2006－AL－105（7）
2006／3／17

P2Pシステムにおける確率的弱コーラムシステムを用いた自己適応的探索手法

呉 エキ　泉 泰介　大下 福仁　角川 裕次　増澤 利光
大阪大学 情報科学研究科

抄録 — 本論文では，新しいＰ２Ｐ探索手法である，自己適応的探索手法（ＡＰＳＰ）を提案する．ＡＰＳＰは確率的弱コーラムシステム（ＰＷＱＳ）を用いた探索手法に基づき，効果的な拡張を施したものである．本論文ではＡＰＳＰの実現しているメッセージ複雑度が最小であることを証明する．また，システムの動的変化に適応するため，システム内のノード数などのパラメータを推定するアルゴリズムを提案する．さらに，ＡＰＳＰと推定アルゴリズムについてシミュレーションを実施し，ＡＰＳＰによりメッセージ複雑度が削減されること，推定アルゴリズムによりシステム内のノード数が高い精度で推定可能であることを検証する．

An adaptive searching protocol in Peer-to-peer systems based on Probabilistic Weak Quorum System

Yu Wu　Taisuke Izumi　Fukuhito Oosita　Hirotsugu Kakugawa　Tosimitsu Masuzawa
Graduate School of Information Science and Technology, Osaka University

Abstract — In this paper we propose a new Peer-to-peer searching protocol: Adaptive Probabilistic Searching Protocol (APSP). It is an efficient extension of the original searching protocol based on Probabilistic Weak Quorum System (PWQS). APSP has much lower message complexity, which is proved the minimum, than the original protocol. And it is adaptive to network dynamics by estimating system parameters with an estimation algorithm we present in this paper. The estimation algorithm requires small additional message cost and the simulation result proves its efficiency. With the estimation algorithm, APSP keeps the minimum message cost when network environment changes.

1 Introduction

Peer-to-Peer (P2P) network [1] is an overlay network of peer computers without centralized servers. With the prominent merit of efficient utilizing of system resource, load distribution, fault tolerance and low initial investment, P2P network is wildly used in file sharing; high performance distributed computing and collaborative user working systems.

P2P file sharing system is a kind of popular P2P system which develops quickly these years. In such a system, a user (peer) can exchange his files with other peers in the system. Files in P2P file sharing systems are distributed in member peers instead of in a centralized server. To find the peer which stores a target file, namely, the searching problem becomes the basic and unique problem of constructing a P2P system. We abstract manual, data sheet, music file, etc. as *object*. The term 'object' will be used instead of 'file' in this paper.

Because there is no centralized server that maintains a list of objects and their location in network, a peer must find the location of a target object by communication with other peers. Different from a system with centralized servers, P2P systems have dynamic member peer sets. Because peers, with objects, may join and leave the system at any time, to keep the information of peer sets and shared objects up-to-date is necessary in finding an target object. The dynamic nature makes it difficult to construct a searching protocol in P2P file sharing systems with a small overhead.

1.1 Related works

Many solutions of searching problem have been proposed.

Gnutella[2] uses a flooding based searching strategy. In Gnutella, when a peer searches an object, it broadcasts a query message to neighbor peers it knows, and the query is forwarded by flooding. Forwarding of a query will stop when the time-to-live (TTL) value of the query decreased to zero. If the target object is in the distance that query can reach, searching is success. Gnutella is a P2P system that is easy to implement, and does not need maintenance of network structure. But in such a flooding based search protocol, a target object must be within a certain distance, and increase of search range causes an explosion of query message. It is not scalable in message complexity.

Chords[3], FreeNet[4], and CAN[5] are examples of distributed hash table (DHT) based P2P systems. These systems use hash value to identify

peers and objects. An object in such systems are stored in the peer which have a contiguous hash value. By forwarding the searching query to a peer which have a more contiguous hash value, the query will arrive the owner peer of the target at last. DHT based searching algorithms are deterministic algorithms and considered scalable for their sublinear searching message complexity. But maintaining a tight DHT structure in a dynamic network is a heavy work which costs much system resource. In practical use, because those systems match objects to peer based on hash value, they are suitable in the case of searching a particular object. But, it is difficult to enhance searching performance, such as fuzzy search, in DHT based searching algorithms.

The searching protocol based on Probabilistic Weak Quorum System (PWQS)[6] is a new P2P searching protocol proposed recently. It uses a randomized algorithm which can find a target object of a high probability. Different from the searching protocols based on DHT, it does not use an logical structure and does not use hash value to identify peers and objects. That makes it easy to maintain and to enhance searching performance. And it is also proved scalable and fault tolerant. More detail of the searching protocol based on PWQS can be found latter in this paper.

1.2 Our contribution

In this paper we present APSP: Adaptive Probabilistic Searching Protocol, which is an efficiency and extension of the original searching protocol based on PWQS.

The original protocol set same system parameter to each object. Each object will be found with a same probability and expected to use same number of query messages in one searching. We focus on the heterogeneity of objects. In original protocol, a popular object costs a large amount of searching query messages because it is searched frequently. And an unpopular object needs same maintenance cost but is searched rarely. The object's searching rank in P2P file sharing systems is proved following a Zipf-like distribution[7][8]. That means a part of objects in the system are searched with a very high frequency and most of the objects is rarely searched. Thus, we consider the original protocol does not have an optimized message complexity.

The goal of APSP is to optimize the message complexity of the original searching protocol. Its principle is simple. In APSP, a popular object is expected to be found with less query messages by spent more maintenance cost on it. Although its maintenance cost is increased, the total messages will be greatly decreased because it is searched with a high frequency but maintained in a fixed, much lower frequency. In the opposite, an unpopular object is expected to be found with more query messages but the maintenance cost of it is decreased. We also extend the protocol to be an adaptive one. Make it keeps the minimum message complexity when network environment changes. This is realized by estimating the varying network parameters with the estimation algorithms we presented. Our protocol inherits the fault tolerant nature from the original and needs almost no additional system cost.

1.3 Organization of this paper

In section 2, we describe the system model and introduce the PWQS and the principle of searching based on PWQS. In section 3, we introduce APSP. In section 4, we show our simulation results. And in section 5, we give concluding remarks.

2 Preliminaries

2.1 System Models

Peer-to-Peer system.

A distributed Peer-to-Peer(P2P) network is a set of peers $U = \{p_1, p_2 ... p_n\}$. A P2P network is also appeared as an overlay network. In an overlay network, a peer p_x can directly communicate with any other peer p_y through a logical link if p_x knows p_y's name, no matter whether there is physical link between p_x and p_y. In this paper, the term *name* will be used to denote a unilateral link points to p_y. A *name* can be anything that contain a destination peer's location information such as network address etc.

P2P file sharing system is a kind of application of P2P network. In a file sharing system, member peers exchange their objects (files) with each other. An object is the abstract of a manual, data sheet, music file, etc. In P2P file sharing systems, all object are distribute to peers. Searching for a object, is the process to find the peer which stores the target object. For each object, a keyword can be determined from its title, author or file name etc. We state the probabilistic P2P searching problem as follows.

Definition 2.1. The *probabilistic peer-to-peer searching problem* is the problem to find a peer that holds an object of the given keyword with a given probability σ,$0 < \sigma < 1$.

2.2 Searching protocol based on PWQS

Probabilistic Weak Quorum System.

A Probabilistic Weak Quorum System (PWQS)[6] is a communication structure between peers.

Definition 2.2. (Set system [9]) A *set system* $\mathcal{Q}$ of U is a family of sets of U, i.e., $\mathcal{Q} = 2^U$. Each element Q in $\mathcal{Q}$ is called a quorum.

A quorum to be accessed is selected probabilistically. Probability of quorums is defined by access strategy defined as follows.

Definition 2.3. (Access strategy [9]) Let $\mathcal{Q}$ be a *set system* of U. Then an access strategy of $p_i \in U$ is a probability distribution $w_i : \mathcal{Q} \mapsto [0, 1]$ such that $\sum_{Q \in \mathcal{Q} w_i(Q)=1}$.

Definition 2.4. (Access strategy vector) Let $\mathcal{Q}$ be a set system of U and w_i be an access strategy for each $p_i \in U$. Then, *access strategy vector* of U is a vector $\bar{w} = (w_1, w_2, ..., w_n)$.

The definition of PWQS is

Definition 2.5. (Probabilistic weak quorum system) For any integer k, a tuple $< k, \mathcal{Q}, \bar{w}, \varepsilon >$ is a *probabilistic weak quorum system* of order k under U if and only if the following conditions are satisfied.

- $\mathcal{Q}$ is a set system of U,
- $\bar{w} = (w_1, w_2, ..., w_n)$ is an access strategy vector, and
- For any $k+1$ peers $p_x, p_{y1}, p_{y2}, ..., p_{yk} \in U$,

$$\sum \{w_x(Q_x)\Pi_{j=1}^{k} w_{yj}(Q_{yj})\} \geq 1 - \varepsilon$$

where the sum is taken over every combination of $\{Q_x, Q_{y1}, Q_{y2}, ..., Q_{yk}\} \in \mathcal{Q}$ such that $Q_x \cap (Q_{y1} \cup Q_{y2} \cup ... \cup Q_{yk}) \neq \phi$.

Intuitively, any quorum is likely to have nonempty intersection with at least one quorum among any k quorums, and such an event occurs with a probability at least $1 - \varepsilon$.

Searching based on PWQS. The principle of searching an object based on PWQS is explained as follows:

1. Preliminarily, each peer p_{yi} that owns an object randomly selects a quorum $Q_{yi} \in \mathcal{Q}$. Then, it sends an index, which contains the keywords of the object and the name of p_{yi}, to each peer in Q_{yi}.
2. When a peer p_x wishes to search an object, it randomly selects a quorum $Q_x \in \mathcal{Q}$. Then, it sends a query message to each peer in Q_x.
3. A peer p_i in $Q_x \cap Q_{yi}$ informs p_x that p_{yi} owns the target object by sending the target object's index to p_x.
4. If the number of peers that hold the target object is k, there exists such a peer $p_i \in Q_x \cap \bigcup_{i=1}^{k} Q_{yi}$ with a high probability, σ, by the definition of PWQS.

In [6], a protocol based on the searching principle above is presented. It executes three threads in parallel at each peer.

- *Name thread* collects a set of peers' name. The size of the set will be larger than quorum size q and each name in it is randomly selected from $U(t)$. So we can obtain a quorum which is approximately regard as being randomly selected randomly from $U(t)$ by randomly selecting q peers from the name set.
- *Index thread* broadcasts objects' indexes to other peers and manages received indexes. For fault tolerant, all indexes are refreshed in a cycle of TTL_i. We call the cycle TTL_i as the maintenance cycle of index.
- *Query thread* sends queries to other peers when searching and answers received query.

Theorem 2.1. The *success probability* [6] of the searching protocol is

$$\sigma \geq 1 - e^{-\frac{q^2 \cdot k}{n}}, \tag{2.1}$$

where

σ	is success probability,
n	is the total number of peer in network,
q	is the size of quorum (A preliminarily decided system parameter),
k	is the number of replicates of the goal object.

3 Efficiency of the searching protocol based on PWQS

In this section, we present a efficient solution to optimize the message complexity of the searching protocol. The new protocol optimizes the system message complexity by updating system parameter adapting to network environment changes.

3.1 Message complexity

The system message complexity M of the searching protocol based on PWQS is

$$M = M_n + M_i + M_s \tag{3.1}$$

where

M_n is the messages cost for collecting peer's name by *name thread*,

M_i is the messages cost for broadcasting and maintenance of index by *index thread*,

M_s is the messages cost for searching objects by *query thread*.

Because each object is searched independently and its indexes are broadcasted and maintained independently, we have

$$\begin{aligned} M &= M_n + \sum_{j=1}^{|object|} M_{ij} + \sum_{j=1}^{|object|} M_{sj} \\ &= M_n + \sum_{j=1}^{|object|} (M_{ij} + M_{sj}), \end{aligned} \tag{3.2}$$

where

M_{ij} is the messages cost for broadcasting and maintenance of object o_j's index

M_{sj} is the messages cost for searching object o_j

We define the message complexity of an object o_j as $M_j = M_{ij} + M_{sj}$ and rewrite equality 3.2 to

$$M = M_n + \sum_{j=1}^{|object|} M_j, \tag{3.3}$$

According to the principle of searching based on PWQS, we know the original protocol broadcasts q index for each of k replicates of each object in a cycle of TTL_i and sends q query messages when searching for an object.

For simplicity and without losing of generality, we divide continuous time to TTL_i periods. The system message complexity in a period time TTL_i is

$$M = M_n + \sum_{j=1}^{|object|} (k \cdot q + f_{sj} \cdot q), \tag{3.4}$$

where, f_{sj} is the number of times an object o_j is searched in TTL_i.

3.2 Optimizing strategy

We present our optimizing strategy as follows.

1. Optimizing system message complexity by minimizing every object's message complexity M_j separately.
2. Each object's message complexity is minimized by adjusting the number of its index based on how often it is searched.
   - Popular objects have more index so that they will be found easier.
   - Unpopular object have less index so that they have less index cost.

Because each object is searched independently and its indexes are also broadcasted and maintained independently, the system message complexity M is the minimum when every object's message complexity M_j is the minimized

$$\min(M) = M_n + \sum_{j=1}^{|object|} \min(M_j) \tag{3.5}$$

APSP no longer uses the fixed parameter q and $k \cdot q$ as the number of query messages used to search object and the number of indexes of o_j. We use q_{sj} and q_{ij} instead of them respectively. Then, we redefine the object's message complexity in TTL_i as

$$M_j = q_{ij} + f_{sj} \cdot q_{sj} \tag{3.6}$$

In APSP, each object's index number is decided by its searching frequency. Therefore, before find the object, a searcher peer is not able not know how many indexes of the target object there are in the system and how many query messages are necessary to meet a given success searching probability. We change our searching method to: continue searching randomly selected peers until the target object is found.

Theorem 3.1. The expectation of the number of messages used to find an object o_j is:

$$E[q_{sj}] = \frac{n}{q_{ij}} \tag{3.7}$$

Proof. Because every peer to search is randomly selected and the searching will be finished as long as the first index is found. The probability of finding an index with each search query message is q_{ij}/n, and the number of messages cost, q_{sj}, follows a geometric(al) distribution[10] of probability q_{ij}/n. Thus, the expectation of q_{sj} is n/q_{ij}. □

Merge equality 3.5 and 3.6, we have the following theorem trivially.

Theorem 3.2. An object's message cost's expectation is the minimum:

$$E[M_j]_{min} = 2\sqrt{n \cdot f_{sj}}$$

when its index number is set to:

$$q_{ij} = \sqrt{n \cdot f_{sj}} \tag{3.8}$$

Summary of optimizing strategy

The principle of APSP is describe as follows:

1. Preliminarily, each peer p_y that owns an object o_j randomly selects a set of $\sqrt{n \cdot f_{sj}}$ peers. Then, it sends the object's index to all of them.
2. When a peer p_x wishes to search an object, it continues sending query messages to randomly selected peers until find it.
3. The expectation of message complexity of object o_j in TTL_i is $2\sqrt{n \cdot f_{sj}}$. And it is proved the minimum.
4. By minimize every object's message complexity, the system message complexity is also minimized. $M_{min} = M_n + \sum_{j=1}^{|object|} 2\sqrt{n \cdot f_{sj}}$

3.3 Estimation of dynamic network parameters

According to theorem 3.2, we know the core of APSP is to set the index number of each object o_j to $\sqrt{n \cdot fs_j}$. But in a dynamic P2P system the values of n and f_s are varying with time. In order to realize out protocol, we present estimation algorithms to estimate n and f_s.

Estimation of f_s

In our estimation algorithm, we divide continuous time to separate time period, the length each period is equals to TTL_i. We assume in a period of time, labeled t, network environment is transitorily static and it changes only when time label changes. The algorithm is described as follows:

- The peer which is the owner of object o_j count the number of the times that o_j is accessed in time period t. The number is marked as $f_{sj}(t)$.
- In time period $t+1$, $f_{sj}(t)$ is used as the estimated value of $f_{sj}(t+1)$, marked as $f'_{sj}(t+1)$.

However, when the search frequency of o_j varies impetuously, $f'_{sj}(t+1)$ may be distinctly different from $f_{sj}(t+1)$. But this just results the object cost a bit more messages and it is proved in most of time, network environment does not change so impetuously[11].

This algorithm costs no additional messages.

Estimation of n

The algorithm of estimate n is mainly based on Theorem 3.2. It is considered if we know $E[q_{sj}]$ of an object o_j, the owner of o_j can estimate n through equality 3.6. The algorithm is described as follows:

- The time model used in the estimation algorithm of f_{sj} is continue to be used.
- In time t, $f'_{sj}(t)$ is estimated with the algorithms above and every object have the estimated amount of indexes $q_{ij}(t)$.
- When a peer searches for an object o_j, it counts the query messages sent, marked as $\hat{q}_{sj}(t)$, and give $\hat{q}_{sj}(t)$ to the owner of o_j when asks for the object.
- The owner of an object o_j compute $\hat{n}(t)$ by multiplying $\hat{q}_{sj}(t)$ and $f'_{sj}(t)$. $\hat{n}(t)$ is considered as an sample of $n(t)$
- When time label changes, peers who have objects compute $n'(t)$ by taking the average of $\hat{n}(t)$s.
- In the next time period, $n'(t)$ is used as $n(t+1)$.

If we do not consider the change between $n(t)$ and $n(t+1)$, the correctness of the estimated value $n'(t+1)$ is related to how many samples $n'(t)$ is used.

Theorem 3.3. With the assume that $n(t)$ is static in a period of time labeled t, estimation $n(t)$ by taking the average of $n'(t)$, $n'(t)$ is the sample of $n(t)$, the relationship among $n(t)$, $n'(t)$, and the number of samples s can be described as follows. (For simple, time label is omitted.)

$$Pr(|n' - n| \geq \delta n) \leq \frac{1}{\delta^2 \cdot s} \tag{3.9}$$

Proof.
First, we prove the expectation[10] of n' is $E[n'] = n$:

$$\begin{aligned} E[n'] &= E[\frac{\sum_{j=1}^{s} \hat{n}_j}{s}] \\ &= \frac{E[\sum_{j=1}^{s} \hat{n}_j]}{s} \\ &= \frac{E[\sum_{j=1}^{s} q_{ij} \cdot \hat{q_{sj}}]}{s} \end{aligned}$$

Because each searching is independent[10],

$$E[n'] = \frac{\sum_{j=1}^{s} q_{ij} \cdot E[\hat{q_{sj}}]}{s}$$

And according to equality 3.6

$$\begin{aligned} E[n'] &= \frac{\sum_{j=1}^{s} q_{ij} \cdot \frac{n}{q_{ij}}}{s} \\ &= \frac{\sum_{j=1}^{s} n}{s} \\ &= n \end{aligned}$$

Next, we prove the variance[10] of n' is $VAR[n'] = \frac{n^2}{s}$:

$$\begin{aligned} VAR[n'] &= VAR[\frac{\sum_{j=1}^{s} \hat{n}_j}{s}] \\ &= \frac{VAR[\sum_{j=1}^{s} \hat{n}_j]}{s^2} \end{aligned}$$

Because each searching is independent[10],

$$\begin{aligned} VAR[n'] &= \frac{\sum_{j=1}^{s} VAR[\hat{n}_j]}{s^2} \\ &= \frac{\sum_{j=1}^{s} VAR[pi_j \cdot ps_j]}{s^2} \\ &= \frac{\sum_{j=1}^{s} pi_j^2 \cdot VAR[ps_j]}{s^2} \end{aligned}$$

Because qs follows a geometric distribution[10] of probability $1/qi$:

$$VAR[qs] = \frac{1 - qi}{qi^2}$$

Therefore,

$$\begin{aligned} VAR[n'] &= \frac{\sum_{j=1}^{s} pi_j^2 \cdot \frac{(1-pi_j)n^2}{pi_j^2}]}{s^2} \\ &\leq \frac{\sum_{j=1}^{s} n^2}{s^2} \\ &= \frac{n^2}{s} \end{aligned}$$

Finally, according to Chebyshev's Inequality[10]:

$$Pr(|n' - E[n']| \geq \eta) \leq \frac{VAR[n']}{\eta^2}$$

Let $\eta = \delta E[n']$, we have:

$$\begin{aligned} Pr(|n' - E[n']| \geq \delta E[n']) &\leq \frac{VAR[n']}{\delta^2 E^2[n']} \\ Pr(|n' - n| \geq \delta n) &\leq \frac{VAR[n']}{\delta^2 n^2} \\ &\leq \frac{n^2/s}{\delta^2 n^2} \\ &= \frac{1}{\delta^2 \cdot s} \end{aligned}$$

□

For example.

If we want our estimation $n'(t+1)$ is bounded by $\pm 0.1n(t)$ with a probability higher than 90%, we need compute the average of at least $\frac{1}{0.1^2 \times 0.1} = 1000$ samples.

In implementation, some peer which have few objects may not be able to collect so much samples in TTL_i. This can be solved by exchange the data of $\hat{n}_j$ among peers.

4 Simulation results

In this section, we present our simulation results to show the efficiency of our protocol by compare its message complexity with the original's. We also test our estimation algorithm to see how it works by comparing the simulation result with the theoretical result.

In our simulation, we use the same time model with the estimation algorithms'.

- Continuous time is divide to time periods of the length TTL_i. Every period is called a *round*.
- In round t, network environment is transitorily static and changes only when time label changes.

We test a system with about 10000 peers, which is variable in the simulation of dynamic system environment, and a fixed number, 10000, of objects.

We do not test the performance of *name thread* and assume it works ideally. This means when select a *name*, as a destination of sending query or broadcasting index, it is uniformly random selected form system. And the message cost of *name thread* will not appear in our results.

In our simulations, the searching frequency of objects will follow a Zipf-like distribution[7] of an α between 0.63 and 1.24. It is proved that the searching frequency of real P2P systems follows such a distribution by the study of Gnutella[7].

4.1 Simulation results of the protocol's efficiency

In this subsection, we test the efficiency of our protocol by comparing the message complexity with the original.

First, we compare the message complexity of two protocols in a static environment. The number of peers is 10000 and the distribution of object's searching frequency follows a Zipf distribution with a Zipf $\alpha = 1$.

It should be point out that, the original searching protocol uses a fixed number q to search for an object. It does not mean every search costs exactly q messages. The *query thread* will stop sending query messages when an index of the target object is found. And the success probability of the original protocol greatly effects our result. We simulate the original protocol with a probability of 1. It

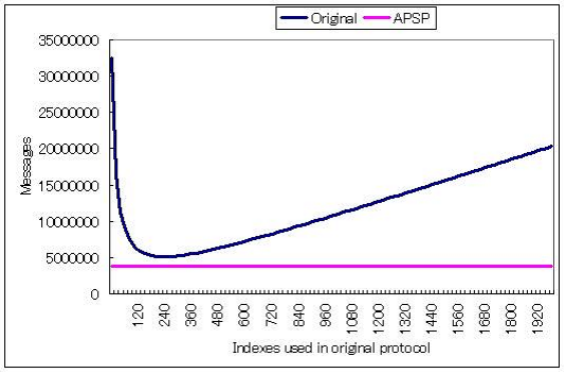

Figure 1: Message complexity in a round

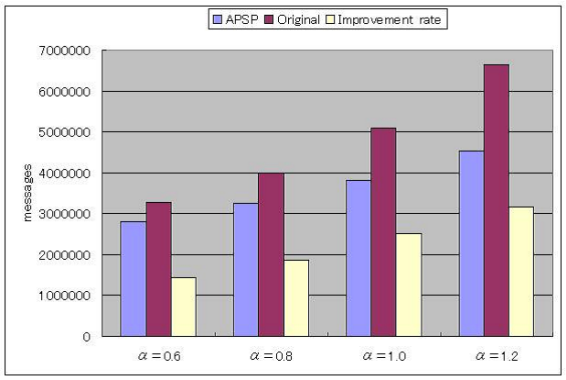

Figure 2: message complexity with different α

means we search for an object until it is found in the simulation of original protocol. This is reasonable because the searcher peer may try again when the target object is not found.

Figure 1 is the result of the two protocol's message complexity work in a static environment we defined above. The value of Y axis is the massages cost in a *round* of two protocols. The value of X axis is the number of indexes used of the original protocol. To the new protocol, the value of X axis is pointless because it decides the number of indexes of every object based its searching frequency.

From the result we find the original protocol cost the least messages when the index number is set to 260, which is the value we expected. Compare the message complexity of our protocol, marked as M_{new}, and the message complexity of original protocol, marked as M_{org} at the index number of 260, we have the *lowest improvement rate*, $\min(\rho) = (\min(M_{org}) - M_{new})/\min(M_{org})$, of our protocol. In this case, it is about 25%. This result proves our protocol has an obviously lower message complexity.

Next, we test the *lowest improvement rate* $\min(\rho)$ of our protocol in different environments. In this test, the rank of objects follow Zipf distributions of different Zipf α[7]. Because the study of Gnutella[7] shows that α is between 0.63 and 1.24, we test the case $\alpha \in \{0.6, 0.8, 1, 1.2\}$. The message complexity of APSP, the lowest message complexity of the original protocol and the lowest improvement rate of is shown in figure 2. It shows that the *lowest improvement rate* of our protocol is better when α is higher. That is the result we expected.

4.2 Simulation results of the estimation algorithm

In this subsection we test the estimation algorithm of the number of peers, $n(t)$, in system. We take

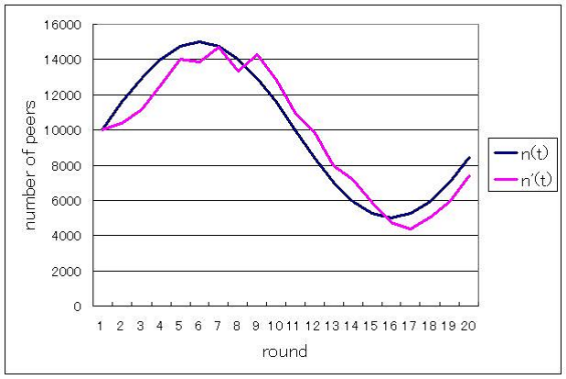

Figure 3: Estimation of n

the average of 1000 samples and expects in most of times, about 90%, the estimation $n'(t)$ is in the range of $0.9n(t)$ and $1.1n(t)$.

The result is shown in figure 3. In order to see how our estimation algorithm works in a varying environment, we set the number of peers to $n(t) = 10000 + 5000\sin(2\pi t/1000)$. In the result, we can clearly see $n'(t)$ has a delay of about one round from $n(t)$. This is reasonable because we use the average of date collected in $n(t-1)$ as the estimation of $n(t)$.

5 Conclusions

In this paper, we proposed APSP, a new P2P searching protocol, which is an efficiency of the original searching protocol based on PWQS. It optimizes the system message complexity by adjust the number of indexes of every object based on the frequency the object is searched. The core of our protocol is to set an object o_j's number of indexes to $\sqrt{f_{sj}n}$, and the the object's message complexity in a period time TTL_i, is $2\sqrt{f_{sj}n}$, which is proved the minimum. Here, TTL_i is the maintenance cycle of

all indexes, f_{sj} is the number of times the object is searched in TTL_i and n is the number of peers in system.

The simulation result shows that the our protocol used obviously less message than the original and almost equals to the theoretical value.

We also extend the protocol to be adaptive to network dynamics. In a dynamic network environment, n and f_{sj} is varying with time. It is difficult to know these two global system parameter in a distributed system, especially n. We proposed an algorithm to estimate them. The estimation algorithm cost almost no additional messages and is proved collect and efficient by the simulation result.

Future work.

The problem we must solve in the future is to decide how long the TTL_i is. The problem will be considered with the research of the time a peer continues connect to a system. Set TTL_i too long makes it difficult to keep the amount of indexes between index refreshes. Set TTL_i too short makes it cost much messages to maintain indexes.

The searching protocol based on PWQS has many excellent features which makes it easy to improve searching performance. For example, replicate system and caching system will be easier to construct in such an system without a logical structure. And because we do not use hash value to identify objects, fuzzy searching can be realized in our protocol naturally. These work will be discussed and researched in the future.

Acknowledgement

This work is supported in part by MEXT: "The 21st Century Center of Excellence Program", JSPS: Grant-in-Aid for Scientific Research ((B)15300017, Young Scientists (B) 15700017), MEXT: Grant-in-Aid for Scientific Research on Priority Areas (16092215) and MIC: Strategic Information and Communications R& D Promotion Programme (SCOPE).

References

[1] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, March 2002.

[2] M. Repeanu. Peer-to-peer architecture case study: Gnutella network. In *First International Conference on Peer-to-Peer Computing (P2P'01)*, page 0099, 2001.

[3] Ion Stoica, Robert Morris, and David Karger. Chord: A scalable peer-to-peer lookup service for internet application. In *Proceedings of SIGCOMM*, pages 149–160, 2001.

[4] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore Hong, W. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservabilty*, pages 44–66, 2000.

[5] Sylvia Ratnasamy, Paul Francis, Mark Handley, M. Karp, Frans, and Scott Schenker. A scalable content-addressable network. In *Proceedings of SIGCOMM*, pages 161–172, 2001.

[6] K. Miura, T. Tagawa, H. Kakugawa, and IEEE Computer Society. A quorum-based protocol for searching objects in peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(1):25–37, January 2006.

[7] Kunwadee Sripanidkulchai. The popularity of gnutella queries and its implications on scalability. URL `http://www.cs.cmu.edu/~kunwadee/research/p2p/paper.html`, 2001.

[8] E. Adar and B. Huberman. Free riding on gnutella. In *First Monday*, volume 5, October 2000.

[9] Moni Naor and Avishai Wool. The load, capacity and availability of quorum systems. *SIAM Journal on Computing*, 27(2):423–447, 1998.

[10] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. CAMBERIDGE, 2005.

[11] Ismail Ari, Bo Hong, Ethan L. Miller, Scott A. Brandt, and Darrell D. E. Long. Managing flash crowds on the internet. In *the 11th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2003)*, 2003.