This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Probabilistic Search in P2P Networks with High Node Degree Variation

Haoxiang Zhang*‡, Lin Zhang*, Xiuming Shan* and Victor O. K. Li*†
*Department of Electronic Engineering
Tsinghua University, Beijing 100084, P.R. China
†Department of Electrical and Electronic Engineering
University of Hong Kong, Pokfulam, Hong Kong, P.R. China
‡Email: Zhanghx99@mails.tsinghua.edu.cn

***Abstract*—A novel Adaptive Resource-based Probabilistic Search algorithm (ARPS) for P2P networks is proposed in this paper. ARPS introduces weighted probabilistic forwarding for query messages according to the node degree distribution and the popularity of the resource being searched. A mechanism is introduced to estimate the popularity and adjust the forwarding probability accordingly such that a tradeoff between search performance and cost can be made. Using computer simulations, we compare the performance of ARPS with several other search algorithms. It is shown that ARPS performs well under various P2P scenarios. ARPS guarantees a success rate above a certain level under all circumstances, and enjoys high and popularity-invariant search success rate.**

I. Introduction

File sharing is one of the most popular applications of P2P networks. Due to legal considerations, Napster [1] and other P2P systems with centralized search indices have been replaced by decentralized systems. There are currently two major categories of decentralized systems, namely, the structured systems and the unstructured systems. Today, most popular P2P applications operate on unstructured networks, organized in an ad hoc fashion, and can easily accommodate a highly transient node population. To locate a resource becomes one of the most challenging issues in this kind of systems.

To locate the target files, Gnutella [2] peers flood query messages across the overlay network. The flooding is limited to a given number of hops, defined by the Time-To-Live (TTL) parameter of the query packets. Although it is simple and capable of discovering the maximum number of objects in the search region, this flooding scheme does not scale well. To tackle this problem, many search methods have been proposed, and they can be categorized as blind search and informed search, based on whether the information about document locations is used to locate the resources.

The basic goal of all such search mechanisms is to balance the volume of control traffic and the search performance, specifically, to control the volume of the query messages while guaranteeing a certain level of search performance.

To accomplish this goal, the popularity of the resource, defined as the probability that a randomly chosen node possesses the resource, may be exploited. A high popularity indicates that the resource can be located easily, and the cost for the search will be correspondingly smaller. On the other hand, the search for resources with lower popularity will produce more messages, reach more nodes and often suffer from low success rate. That is to say, under the same search mechanism, the cost (defined as the number of messages produced or the number of peers contacted) and search performance (defined as the success rate and number of hits) are different between popular and unpopular resources. But both the blind and informed search methods seldom take the popularity into consideration, treating the resources with different popularities in the same way.

Recent measurements of Gnutella, a highly popular P2P file-sharing system, show that the underlying network topology has a power-law node degree distribution [10]. The node degrees exhibit high variance, with a few nodes having very high degrees while many others, low ones. Therefore, whether and how the search methods exploit the power-law distribution in the node degree will greatly impact the search performance and cost.

In this paper we present a novel adaptive resource-based probabilistic search algorithm (ARPS) for P2P unstructured networks. ARPS considers the degree variance and the difference in popularity amongst the resources. In ARPS, a node uses weighted probabilistic forwarding for query messages, varying the forwarding probability according to the popularity of the resource being searched and its own degree. Peers estimate the popularity of the resource in the network based on feedback from previous searches. As we will show in this paper, ARPS exhibits several desirable characteristics on power-law networks, such as high accuracy for unpopular resources, much lower cost for popular ones, and excellent adaptability in scenarios with dynamic popularity. That is to say, ARPS achieves a better tradeoff between cost and performance than traditional methods over a wide range of P2P scenarios.

The rest of the paper is organized as follows. In Section II, we describe our work in the context of related research. Section III presents some analytical results of probabilistic search. Section IV describes the ARPS algorithm. Simulation results are provided in Section V. Section VI concludes the paper and outlines the future work.

1-4244-0353-7/07/$25.00 ©2007 IEEE

II. Related Work

Many search algorithms for unstructured P2P networks have been proposed in the last few years. Modified-BFS [3], a variation of Flooding, determines a ratio of the neighbors to forward the query messages. In Random Walks [4], another alternative to Flooding, a query node sends out k query messages to an equal number of randomly selected neighbors. These queries are often referred to as walkers. Each walker follows its own path. The intermediate node, upon receiving a walker, randomly chooses one neighbor to forward the query if it does not possess the resource. The walker terminates either with a search success or when TTL becomes 0.

Adaptive Probabilistic Search (APS) [5], an adaptive search algorithm, utilizes the feedback from previous searches to make future searches more efficient. Each node maintains a table for the forwarding probability to each neighbor for each resource. APS employs k random walkers to search for the required resource, and each intermediate node forwards the query to one of its neighbors with a probability given by its table index. Index values are updated after each query using the feedback from walkers. If a walker succeeds (fails), the relative probabilities of the nodes on the walker's path are increased (decreased). The update procedure takes the reverse path back to the requester.

ARPS, proposed in this paper, is an adaptive search algorithm, and follows the probabilistic search instead of the random walk approach. Search with probabilistic forwarding is first analyzed in detail in [6]. In the algorithms introduced, if a peer does not find the resource in its LD (Local Directory) and DC (Directory Cache), it will send a search request to each peer in its neighborhood with a certain probability (called broadcast probability). In contrast to ARPS, the forwarding probability in [6] is a static parameter, and determined at system initialization. Also, [6] only consider the random graph topologies.

Unlike other methods, ARPS takes the popularity of the resource into consideration. Peers keep estimating the popularity of the resource it has requested instead of the states of the outgoing links or immediate neighbors. In order to track the popularity of the resources, an adaptive algorithm is introduced. Based on the estimation, a proper forwarding probability is chosen to propagate the query messages. ARPS also considers the node degree variance of peers. According to its own node degree, peers use the weighted probability to forward the query to further reduce the messages. Specifically, in ARPS, peers forward the query messages to its neighbors with small weighted probability if the resource being searched is with high popularity, and vice versa. The weighted probability is smaller for high degree nodes compared with that for low degree ones. As each node has a different degree and has its own estimation of the popularity, the probability is changed at each node the queries reach.

Probabilistic search, by properly choosing the forwarding probability, allows a tradeoff between the search cost and performance.

III. Probabilistic Search

We employ generalized random graphs (GRG) [8] to model the topology of P2P networks, and use the analytical framework in [9] to model the search strategies of probabilistic forwarding. A GRG represents a family of graph instances with a given number of vertices (nodes) where the degree of a randomly chosen node is specified by an arbitrary probability distribution. Results for a GRG actually have to be viewed as averages over the entire set of possible graph instances. The main results on the use of GRG to model the overlay networks are developed from the following two different generating functions for probability distributions.

The generating function for the probability distribution of the vertex degree is

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k, \tag{1}$$

where p_k is the probability that a randomly chosen vertex has degree k.

The generating function of the degree distribution of the nodes reached by following one end of a randomly chosen edge (the starting edge is excluded) is

$$G_1(x) = \frac{G_0'(x)}{G_0'(1)} = \frac{1}{z} G_0'(x), \tag{2}$$

where $G_0'(x)$ is the first derivative of $G_0(x)$, and $z = G_0'(1)$ is the average node degree. The generating function of the number of nodes two hops away is given by

$$\sum_{k=0}^{\infty} p_k [G_1(x)]^k = G_0(G_1(x)). \tag{3}$$

Therefore, the generating function for the number of peers h hops away from a randomly chosen node is $G_0(\underbrace{G_1(\ldots G_1(x)\ldots)}_{h-1})$.

The message forwarding function g [9] represents the query forwarding probability of the peers. In probabilistic search, each peer propagates the query to its neighbor with probability p_f. The function is in the form of:

$$g(d) = p_f, \forall d < \text{TTL}. \tag{4}$$

The probability that the query originator transmits the search message to n of its neighbors is given by

$$q_n = \sum_{k=n}^{\infty} p_k \binom{h}{n} p_f^n (1-p_f)^{k-n}. \tag{5}$$

The generating function for the number of peers that receive the query in the neighborhood of the requester is

$$Q_1(x, g) = \sum_{n=0}^{\infty} q_n x^n = G_0(1 + p_f(x-1)). \tag{6}$$

The generating function for the number of peers at distance h hops from the query originator that receive the message is

$$Q_h(x, g) = Q_1(\underbrace{\overline{Q}_1(\ldots \overline{Q}_1(x, g), g \ldots, g)}_{h-1}, \tag{7}$$

with

$$\overline{Q}_1(x,g) = G_1(1 + p_f(x-1)). \tag{8}$$

The generating function for the total number of peers that receive the query message under the constraint of TTL is given by

$$Q(x,g,\text{TTL}) = \prod_{m=1}^{\text{TTL}} Q_m(x,g). \tag{9}$$

From Equation (9), the average number $\overline{N}$ of query messages sent throughout the network is

$$\overline{N} = Q'(1,g,\text{TTL}). \tag{10}$$

Let p denote the resource popularity. The probability that, among all neighbors of the query originator that receive the query message, n peers own the resource is

$$P(n,p) = \sum_{k=n}^{\infty} q_k \binom{h}{n} p^n (1-p)^{k-n}, \tag{11}$$

with the generating function

$$H_1(x,g,p) = \sum_{n=0}^{\infty} P(n,p)x^n = Q_1(1 + p(x-1), g). \tag{12}$$

The generating function for the total number of resource owners that receive the query message with the constraint of TTL is

$$H(x,g,p,\text{TTL}) = \prod_{m=1}^{\text{TTL}} H_m(x,g,p), \tag{13}$$

where $H_m(x,g,p) = Q_m(1 + p(x-1), g)$.

From Equation (13), the probability P that the resource can be found (at least one hit returns) is given by

$$P = 1 - H(0,g,p,\text{TTL}). \tag{14}$$

As the degree distribution of popular P2P networks exhibits high variance with power-law characteristics, we use Equations (10) and (14) to obtain the numerical values for the power-law overlay topology with $p_k = \frac{\kappa^{-\tau} e^{-k/\kappa}}{Li_\tau(e^{-1/\kappa})}$, where $Li_n(x)$ is the n^{th} poly-logarithm of x, and τ, κ are the power-law exponent and cutoff, respectively.

Figure 1 gives the mean message number produced for different forwarding probabilities. As a branching process, the message number drops dramatically with the decrease of the forwarding probability. Figure 2 presents the numerical results of the success rate versus the popularity of the resource being searched. For popular resource, probabilistic search with small forwarding probability can achieve a success rate close to flooding ($p_f = 1$).

We can conclude from the analytical results that probabilistic search shows more flexibility, since the success rate and message number can be adjusted by choosing different p_f, allowing a tradeoff between cost and performance.

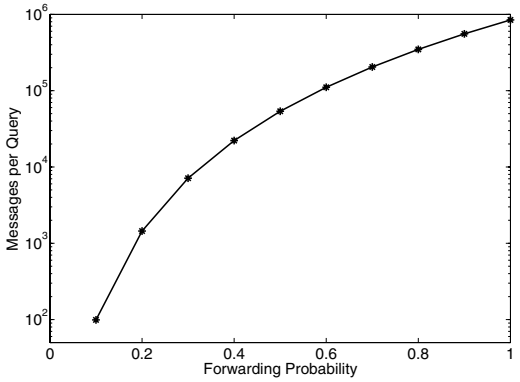

Fig. 1. Message generated vs. forwarding probability (with $\tau = 2.041289, \kappa = 500$)

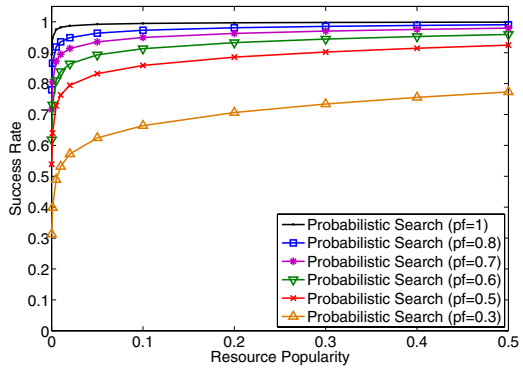

Fig. 2. Success rate vs. resource popularity (with $\tau = 2.041289, \kappa = 500$)

IV. The ARPS Algorithm

A. Algorithm Description

Unlike the analytical model, in real P2P scenarios, because of the self-organized and distributed nature of P2P networks, it is unrealistic for a node to keep accurate information of the resources such as popularity in the network. A mechanism should be introduced to estimate the popularity. Also, as different nodes have different node degrees and are located at various distances from the resources, the popularity estimations will be different. So for each resource, each node has its own forwarding probability, and the probability is changed in a hop-based manner along the path.

In ARPS, each node keeps a local index on the popularity estimate for each resource it has requested or forwarded queries. During a search process, a node first finds the resource in its local index. If there is no match, meaning this is the first search for the resource, the peer will send queries to its neighbors by flooding, with a forwarding probability of 1; otherwise, according to the estimated popularity, a reasonable probability is chosen to forward the message.

B. Algorithm Improvement

In power-law networks, the node degree varies greatly; a few nodes have very high degrees while many others have

low ones. Though the nodes with high degrees generally are limited in number, they play an important role. These nodes are often utilized when designing efficient search algorithm [10]. As Figure 2 illustrates, the forwarding probability can not be too small (less than 0.4); otherwise, the performance of probabilistic search will degrade dramatically. But even with small probability, these high degree nodes produce a significant volume of messages. Moreover, the high degree nodes will be encountered more often. In power-law networks, a large number of links point to only a small subset of high degree nodes; therefore, the link of a randomly chosen node preferentially leads to a high degree node.

In ARPS, the forwarding probability will be adjusted according to the node degree, in order to minimize the number of messages. Here we introduce the parameter η called *Degree Weight*. For node i,

$$\eta_i = \lceil \log_{10}^{D_i/\overline{D}} \rceil. \tag{15}$$

D_i denotes the node degree of peer i, and $\overline{D}$ is the average node degree of the network. η_i describes the degree characteristic of each peer. Each node adjusts the forwarding probability according to the estimated popularity and the *Degree Weight*. So in ARPS, the query is forwarded in a weighted probabilistic manner.

C. Update Equation and Forwarding Probability Selection

The update equation for the estimate of one specific resource at each node is given by

$$p_s(k+1) = \beta \cdot p_s(k) + \Delta_k \tag{16}$$

$$\Delta_k = \sum_{n=1}^{N} \left(\frac{1}{\overline{d_{out}} \cdot p_f(k)} \right)^{h_n}. \tag{17}$$

For each node, suppose it has requested or forwarded requests for R different resources. The node maintains a table of R entries, with an entry for the estimated probability of each resource. Specifically, let $p_s(k)$, $p_f(k)$ denote the current popularity estimate and forwarding probability of the resource r after the k^{th} update, respectively. β is the decay rate, and $\overline{d_{out}}$ denotes the average out degree of the node.

During the new round of search for resource r, after forwarding the query message with the proper probability, the peer decreases the estimated popularity by a factor β. If a query fails to locate a resource when TTL is reached, nothing needs be done to further reduce the index entry, as the estimated popularity has already been decreased because of the unsuccessful search. If the search is successful, suppose there are N backward hits passing along the reverse path to the originator. When the messages arrive at the intermediate peer, h_n is the hop distance from the node to each resource owner. Δ_k incorporates a distance-based function for modifying the relative estimated popularity. Though the estimated popularity will decay with rate β after each search, with the increase of hit number, we get a larger Δ_k, which compensates the estimation decay. If $p_s(k+1) > 1$, Δ_k is set to 1. So we finally get the estimated popularity after the $(k+1)^{th}$ update in the form of:

$$p_s(k+1) = \min(\beta \cdot p_s(k) + \Delta_k, 1). \tag{18}$$

TABLE I
FORWARDING PROBABILITY SELECTION TABLE

Estimated Popularity Interval p_s	Forwarding Probability p_f
$[0, 0.0001)$	1
$[0.0001, 0.005)$	0.9
$[0.005, 0.001)$	0.8
$[0.001, 0.005)$	0.7
$[0.005, 0.03)$	0.6
$[0.03, 0.16)$	0.5
$[0.16, 1]$	0.4

Besides the resource popularity estimation table, each node also contains a forwarding probability selection table, which is a piecewise projection from popularity to forwarding probability. When a node initiates or forwards a query message for the search of a resource with estimated popularity p_s, it looks up the table to find the proper forwarding probability p_f.

Based on the parameter η, each node propagates the message with the weighted probability p'_f. The relation between p_f and p'_f is as follows:

$$p'_f = \max(p_f - \frac{\eta}{10}, 0.1). \tag{19}$$

The minimum probability (0.1) guarantees the search performance, as even a small part of the neighborhood of high degree nodes plays an important role as hubs in network searches.

The numerical value of the piecewise projection in the selection table is obtained from the analytical model in Section III, with a targeted success rate of over 80%.

V. SIMULATION RESULTS

A. Simulation Scenario and Performance Metrics

To simulate the P2P network topology, we generate the power-law graph with exponential cutoff. The degree distribution is the same as the analytical one with an average degree of 3.5 (similar to Gnutella-type graphs [7]). The default graph has 10000 nodes. In order to simulate the resource with popularity p, $\lfloor p \times 10^4 \rfloor$ nodes are randomly selected and marked as the resource owners in the network. In ARPS, the decay rate β is 0.8. As ARPS uses probabilistic search, in each simulation round, at least one query is propagated from the originator.

The performance of ARPS is compared with several other search methods, including Flooding, Modified-BFS and APS. For Random Walks, the requesting node sends out $k = 3$ query messages. In Modified-BFS, nodes randomly choose 3 of their neighbors (if the neighbor number is less than 3, then all the neighbors are chosen) to forward a query message. In APS, we simulate the algorithm in the pessimistic approach with different walker numbers and TTL pairs. The evaluation and

TABLE II
SUCCESS RATE COMPARISON

	$p_f = 0.9$			$p_f = 0.8$		
p	GRG	Sim	Err (%)	GRG	Sim	Err (%)
0.005	0.9519	0.9195	3.40	0.9184	0.8805	4.13
0.01	0.9634	0.9277	3.70	0.9352	0.9065	3.07
0.02	0.9723	0.9348	3.86	0.9489	0.9181	3.24
0.05	0.9814	0.9360	4.63	0.9634	0.9213	4.37
0.1	0.9869	0.9489	3.85	0.9726	0.9320	4.18
0.2	0.9914	0.9576	3.41	0.9809	0.9381	4.36
0.3	0.9938	0.9853	3.38	0.9853	0.9360	5.01
0.4	0.9953	0.9645	3.09	0.9883	0.9427	4.62
0.5	0.9964	0.9801	1.64	0.9907	0.9520	3.91

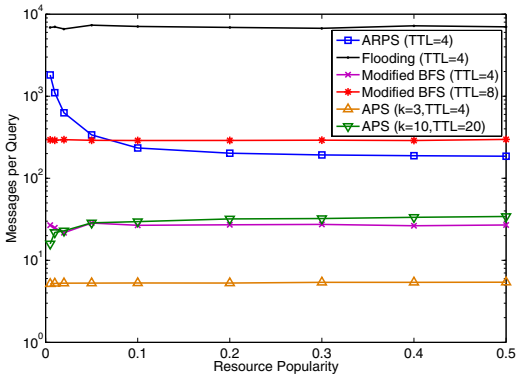

Fig. 3. Messages per query for different search methods.

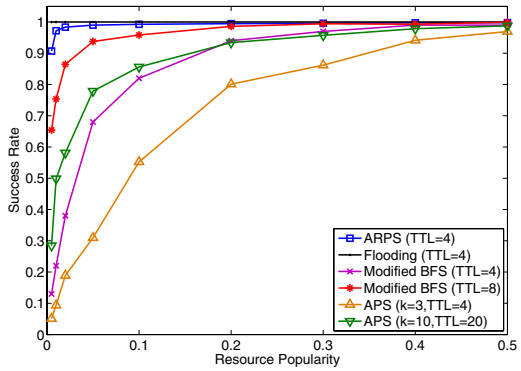

Fig. 4. Success rate for different search methods.

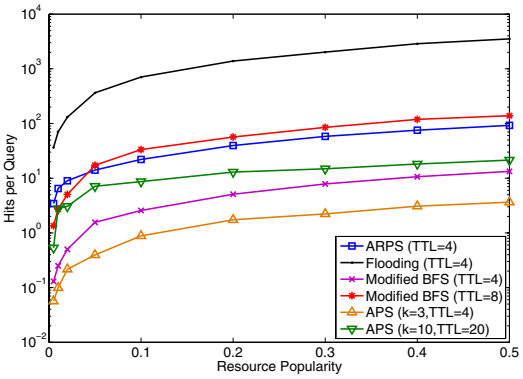

Fig. 5. Hits per query for different search methods.

comparison of the search performance are based on the metrics of messages per request, success rate, and hits per request.

For the dynamic settings, for ease of result interpretation, time is normalized with respect to the search rounds. So $t = j$ refers to the time at which the j^{th} search takes place.

B. Experimental Validation

To verify the analytical results derived in Section IV, we simulate the probabilistic search with different uniform forwarding probability p_f, and compare the simulation results with the analytical ones. This validation is necessary since the analytical model is based on the assumption that the number of newly visited peers at each hop is independent. Table II shows the analytical and simulation results of the success rate. Probabilistic search with different forwarding probabilities are compared. The deviation between the analytical and simulation results is due to the limited number of nodes of the generated power-law graph in the simulation and the independent assumption of the analytical model.

C. Search Performance and Comparison

Figure 4 shows the success rate for the search of resources with different popularities. Flooding gives the highest success rate but at a cost of the poorest scalability under all circumstances. Because of the high degree nodes, Flooding propagates the query message to nearly 70% of all the nodes in the network within 4 hops, generating a huge volume of messages.

Modified-BFS exhibits low success rate for resources with low popularity. By increasing TTL, more query messages are propagated, and Modified-BFS increases the success rate in the search for unpopular resource and returns more hit messages.

Compared with Random Walks, APS achieves much higher performance with the same overhead. But for unpopular objects, the success rate and hit number still remain low. This can be explained by the fact that the method produces limited number of messages and the message numbers stay almost the same for resources with different popularities. Therefore, less popular objects receive considerably fewer queries, leading to miserable success rates. In order to make the algorithm efficient, we have to increase the value of walkers or TTL. As Figures 3 and 4 illustrate, the increased walker number and TTL ($k = 10, TTL = 20$) generates more messages, and improves the success rate to a certain extent. But the improvement is limited due to two reasons: first, the average node degree is limited, and walker number larger than the node degree will not generate extra messages, as fewer than k walkers will be used; second, with larger TTL, the high degree nodes will be encountered much more often, which

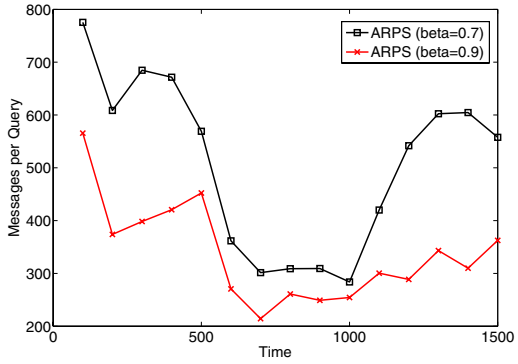

Fig. 6. Messages per query for different decay rates (with average value of 509.74 for $\beta = 0.7$ and 345.33 for $\beta = 0.9$)

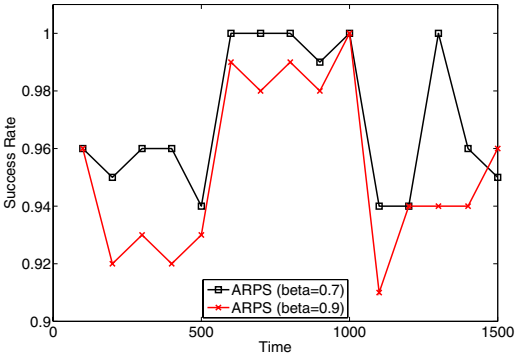

Fig. 7. Success rate for different decay rates (with average value of 97.00% for $\beta = 0.7$ and 95.27% for $\beta = 0.9$)

causes routing loops.

ARPS presents a striking contrast to these methods, with the message number decreasing dramatically at high resource popularity due to its adaptive mechanism. For unpopular resource, p_f is quite high for each hop, and ARPS functions similarly to Flooding. The weighted forwarding probability decreases as the popularity increases, which results in the dramatic dropping of the number of messages generated. Because of the adaptive mechanism, ARPS enjoys a high success rate. In fact, from Figure 4, we find that the success rate is popularity invariant, which is a good characteristic. Figure 5 shows that the hit number is also invariant with popularity. In summary, ARPS achieves a good tradeoff between performance and cost.

D. ARPS Performance with Different Decay Rates

We compare ARPS with different decay rates in the dynamic scenario. At $t = 0$ the resource of interest has popularity $p_s = 0.02$. The popularity is increased to 0.2 at $t = 500$ and decreased to 0.02 at $t = 1000$.

Figure 6 shows that under both decay rates the method succeeds in adjusting the number of messages generated in accordance with the popularity changes. ARPS with small decay rates produces more messages, as the estimated popularity stays low, resulting in high forwarding probability. Figure 7 shows that for both decay rates, ARPS maintains success rates above a certain level. The rate for ARPS with $\beta = 0.9$ fluctuates more, and does not stays as high as the one with $\beta = 0.7$. We conclude that a large β is too aggressive when the popularity is low, or changes rapidly.

VI. Conclusion

In this paper, we propose a novel Adaptive Resource-based Probabilistic Search algorithm (ARPS) for P2P networks. ARPS utilizes weighted probabilistic forwarding in a hop-based manner, with an adaptive mechanism to choose the probability. Furthermore, the high variance of node degrees in power-law networks is taken into consideration in the algorithm. ARPS gives a guaranteed success rate above a certain level under all circumstances, and this high success rate is popularity invariant. At the same time, the number of messages generated adapts to the variation of the resource popularity. All these characteristics make ARPS desirable for P2P scenarios.

ARPS stores the popularity estimates of all resources in the network. This would require $O(n)$ memory space, where n is the number of resources in the network. The memory requirement of ARPS is less than that of APS. Besides the need to adjust p_f, ARPS incurs no additional overhead, which makes the algorithm very simple to use.

Though ARPS exhibits several good characterizes, the performance of ARPS depends on the decay rate to some extent. In the future, we plan to develop a control model to find a better projection from popularity to forwarding probability to further improve the search performance of ARPS.

Acknowledgment

This research is supported in part by the National Science Foundation of China (Grant No. 60672107) and China 973 Project (Grant No. 2007CB307105).

References

[1] "Napster protocol specification," March 2001, http://opennap.sourceforge.net/napster.txt.
[2] Gnutella development forum, The Guntella v0.6 Protocol, 2001.
[3] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," Proc. of the 11th ACM Conference on Information and Knowledge Management, McLean, Virginia, USA, November 2002.
[4] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," Proc. of the 16th ACM International Conference on Supercomputing, New York, New York, USA, June 2002.
[5] D. Tsoumakos, and N. Roussopoulos, "Adaptive probabilistic search in peer-to-peer networks," Technical Report, CS-TR-4451, 2003.
[6] D. A. Menasc, and L. Kanchanapalli, "Probabilistic scalable p2p resource location services," SIGMETRICS Perform. Eval. Rev., vol. 30, no. 2, pp. 48-58, 2002.
[7] M. Ripeanu and I. Foster, "Mapping the Gnutella network," IEEE Internet Computing, vol. 6, pp. 50-57, Jan. 2002.
[8] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," Physical Reviews, E64, 026118, 2001
[9] R. Gaeta, G. Balbo, S. Bruell, M. Gribaudo, and M. Sereno, "A simple analytical framework to analyze search strategies in large-scale peer-to-peer networks," Performance Evaluation, 62(1-4):1-16, 2005.
[10] L. A. Adamic, R. M. Rajan, M. Lukose, A. R. Puniyani and B. A. Huberman, "Search in power-law networks," Physical Reviews, E64, 046135, 2001