Modeling and Analysis of Random Walk Search Algorithms in P2P Networks

Nabhendra Bisnik and Alhussein Abouzeid
Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York
Email: bisnin@rpi.edu, abouzeid@ecse.rpi.edu

***Abstract*— In this paper we develop a model for random walk search mechanism in unstructured P2P networks. Using the model we obtain analytical expressions for the performance metrics of random walk search in terms of the popularity of the resource being searched for and the parameters of random walk. We propose an equation based adaptive search mechanism that uses estimate of popularity of a resource in order to choose the parameters of random walk such that a targeted performance level is achieved by the search. We also propose a low-overhead method for maintaining an estimate of popularity that utilizes feedback (or lack there-off) obtained from previous searches. Simulation results show that the performance of equation based adaptive search is significantly better than the non-adaptive random walk.**

Keywords : Simulations, Statistics, System Design, Peer-to-Peer Networks, Modeling, Performance Evaluation.

I. Introduction

Locating a resource or service efficiently is one of the most important issues related to unstructured decentralized peer-to-peer networks. The objective of a search mechanism is to successfully locate resources while incurring low overhead and delay. In order to fulfill this objective several random walk based search algorithms [12], [8], [3], [14] have been proposed for resource discovery in decentralized peer-to-peer networks. However no analytical model has been proposed to quantify the effect of parameters of random walk (like number and TTL of random walkers) on the performance of random walk search.

In this paper we present a mathematical model for random walk search in a peer-to-peer network. Using the model we derive analytical expressions for the performance metrics, such as delay, overhead and success rate, of the search in terms of the popularity of the resource being searched and the parameters of random walk. We propose an algorithm that uses the analytical expressions in order to adaptively set the parameters of random walk so that it maintains a certain minimum level of performance. The algorithm is referred to as *Equation Based Adaptive Search (EBAS)*[1]. We also propose a method for maintaining popularity estimates of the resources in the network. The popularity estimator is based on an exponentially weighted moving average that smoothes out the high frequency (noise) components.

[1]This is similar to "equation based congestion control" [5].

We perform extensive simulations in order to verify the analytical results of our model and compare the performance of EBAS with pure random walk. The simulation results show that the performance of random walk is significantly enhanced when parameters are set in accordance to the analytical results. The methodology of EBAS may be extended to other random walk based search mechanisms in order to appropriately set parameters of random walk.

The contributions made by this paper can be summarized as follows:

1) Analytical expressions for the performance metrics of random walk as function of the number and TTL of random walkers and popularity of the resource being searched for.
2) An algorithm, called EBAS, to adaptively adjust the parameters of random walk in order to achieve a desired performance.
3) A feedback-based algorithm for maintaining popularity estimate of the resources in the network with the help of feedback from previous queries.
4) Simulation results that evaluate the proposed adaptive search methodology and compare it against previous non-adaptive techniques (pure random walk).

We now summarize the terminology used in this paper. A "search" is *successful* if it results in the discovery of at least one node that has the resource being searched for. The *Success Rate* is defined as the fraction of successful searches given that the resource being searched for is present in the network. The *popularity*, p, of a resource is defined as the probability that a randomly chosen node has the resource i.e. it is the ratio between the number of nodes that have the resource and the total number of nodes in the network. The search *delay* for a successful search is the duration of time between the start of a search for a resource and the time the resource is found. *Overhead* of search is the total number of bytes transmitted in the network for locating the resource. A *resource* may be a media file, an e-book, storage space or idle processor cycles. The performance of a search method is measured in terms of three metrics; (i) success rate, (ii) overhead and (iii) delay. It is desired that a search method has high success rate, low delay and low overhead.

The rest of the paper is organized as follows. In section II we discuss the advantages of random walk and highlight the

importance of adaptively choosing the parameters of random walk. Section III positions our work in context with other related research in this area. Section IV presents the network model and the analytical results for the random walk performance metrics as a function of the random walk parameters. Section V develops EBAS algorithm. Simulation results and comparisons are presented in Section VI. Section VII concludes the paper and outlines future work.

II. Background

Gnutella [2], a popular unstructured and decentralized P2P application, employed flooding of search queries in order to locate files in the overlay network. This resulted in huge overhead due to the search queries. In August 2000 Gnutella community grew to such enormous size that entire bandwidth of Gnutella users connecting to the Internet using dial-up connection was "chocked" because of the large amount of queries they forwarded [1]. Although the unstructured P2P architecture is very appealing due to its fault tolerance, self-organization and low overhead associated with node arrival and departure, efficient search methods are very important for scaling of such networks.

Random walk is a popular alternative to flooding for locating resources in P2P networks. A node, which we call the querying node, that needs to locate a resource sends k queries (i.e. packets) to randomly selected neighbors. Each of these k queries is often referred as a *random walker*. Each random walker has a time to live (TTL) field that is initiated with some value $T > 0$ that limits the number of times the random walker is forwarded. When an intermediate node receives a random walker, it checks to see if it has the resource. If the intermediate node doesn't have the resource, it checks the TTL field, and if $T > 0$, it decrements T by 1 and forwards the query to a randomly chosen neighbor, else if $T = 0$ the query is not forwarded. On the other hand, if the intermediate node has the resource, the query is not forwarded and a reply is sent to the querying node.

Unlike flooding, the overhead of random walk is independent of the underlying topology. If the overlay network has average degree d, then flooding the network with a query having TTL T will produce d^T packets on the average. Studies in [9] have shown that average node degree of Gnutella network is 3.5 and on the average 95% of all nodes are within 7 hops. If flooding is used for querying in this network, with TTL set to 7, the overhead will be 3.5^7. On the other hand the overhead of a search involving k random walkers, each with TTL T, is bounded by $k \times T$ packets irrespective of the underlying topology. Specifically the overhead of random walk varies linearly with k and T while the overhead of flooding grows exponentially with T.

The performance of random walk largely depends on the choice of k and T. Intuitively, the average number of nodes required to be probed for discovering a resource is inversely proportional to the popularity of the resource. Choosing low values of k and T for searching a resource with low popularity would result in low success rate and high delays while choosing high values of k and T for searching resource with high popularity would result in excessive overhead. The parameters of random walk must be chosen according to the popularity of the resource being searched. The popularity of a resource may not be known a-priori at the querying node. In addition, the popularity may change due to arrival/departure of nodes, replication/deletion/exhaustion of resources or other random changes in the network. Thus, the setting of the parameters of the random walk has to be done in an adaptive manner.

The examples of P2P applications where an adaptive search based on popularity of resources would be most effective are:

1) Applications where the same resource is searched for multiple times by a node. For example a P2P application for distributed computing may search for a certain size of storage space or number of processor cycles whenever more data is ready for storing or processing. In this case the popularity estimate of a resource can be maintained using feedback from previous search results. Notice that popularity of a resource may change not only due to joining and leaving of nodes but also due to exhaustion of the resource over time.
2) Applications where the same resource is not searched for multiple times by a node but resources may be categorized such that resources belonging to the same category are likely to have the same popularity. For example, music files may be categorized on the basis of artists or genre (or both). In this case the popularity estimate of the category can be maintained based on feedback from previous search results of resources belonging to that category. Notice that popularity of a resource may change not only due to arrival and departure of nodes but also due to replication or deletion of the resource over time.

III. Related Work

Studies aimed at modeling P2P networks have revealed that the overlay network has characteristics similar to small world networks [10], [9]; high clustering coefficient, power-law distribution of node degree and small average path lengths. A method for growing a connected graph with properties of small world networks has been described in [7]. This method is used in our paper to generate overlay P2P topologies for validation of our analytical results and performance evaluation of our proposed algorithm. In [6] authors show that random walk on a peer to peer network has statistical properties similar to independent sampling from a uniform distribution. This result is used in the analytical derivations in Section IV.

Several search algorithms for unstructured P2P networks have been proposed. These algorithms could be classified into *state-full* and *stateless*. "Stateless" algorithms do not maintain any state information about the network or outgoing links. Flooding of search queries was originally used by Gnutella [2]. In iterative deepening [15], the querying node performs flooding with increasing depth until the search is successful. The search query is forwarded to randomly selected subset of neighbors in modified BFS [11]. In k-random walk [12], the querying node employs k random walkers to search for the desired resource. A two level random walk is proposed in [8]. All these are examples of stateless search algorithms.

In contrast "state-full" algorithms attempt to improve the performance of random walk by maintaining state information at the nodes in order to direct the search queries. They are expected to improve the performance (in terms of higher success rate) at the cost of higher complexity. Forwarding random walkers to neighbors with probability proportional to their degree is proposed in [3]. Directed BFS [15] seeks to improve performance by forwarding queries to neighbors that are more likely to return successful results. Intelligent BFS [11] uses peer ranking mechanism to forward queries to "good" neighbors. The local indices in [15] seeks to improve performance by maintaining indices of data stored by all nodes within k hops at each node. Adaptive Probabilistic Search (APS) is proposed in [14]. APS employs k random walkers to search for the required resource. Each node maintains indices table whose (i, j) entry corresponds to probability with which the node will forward random walker searching for resource i to neighbor j. The probabilities are updated after each query. These search mechanisms are example of state-full search algorithms.

EBAS, the search algorithm proposed in this paper, lies in the category of state-full search algorithms. Unlike most of the state-full search algorithms EBAS does not maintain state of the outgoing links or immediate neighborhood. Instead each node maintains estimates of popularity of the resources in the network.

Although random walk has been used as a basic component of most of probabilistic search algorithms proposed in the literature, quantifying the effects of various parameters of random walk has not been studied to the best of our knowledge.

IV. Analytical Model

In this section we first present the underlying assumptions of our model used for analysis. We then derive analytical expressions for success rate, expected overhead and expected delay of (stateless) random walk in terms of popularity of resource (p), number of random walkers (k) and TTL (T).

A. Modeling and Assumptions

The P2P network is assumed to have small world properties [2] [10], [9] (see Section III). In order to keep the analysis simple we assume that all query packets have the same size and we measure the overhead in terms of number of packets. We also assume that the time required to transmit a query packet over each hop (i.e. from a node to its neighbor) is constant, denoted by ρ. In each time step of duration ρ one node is probed by each random walker. Delay is expressed in terms of number of time steps elapsed before a random walker visits a node with resource. This is also equal to number of nodes visited by the random walker that is first to discover the resource.

We present analytical results for "*pure*" random walk. By "pure" random walk we mean that state information is not maintained by any node. This implies that each node forwards random walkers without any bias towards any particular neighbor and that multiple random walkers visiting the same node are forwarded independently. A random walker is terminated in two cases: 1. If the random walker visits a node that has the resource (successful termination). 2. If TTL of the random walker expires (unsuccessful termination).

[2]This assumption implies that the network has no bad cuts or bottlenecks.

B. Analytical Results

In the analysis below, we invoke the following result [6]: Random walk achieves statistical properties similar to independent sampling for every reasonable network and network model (e.g. small-world networks). So each node visited by a random walker may be assumed to be an independent sample from a space of uniform distribution of the nodes. From the definition of popularity and this result, the probability that a node visited by a random walker has the resource equals the popularity of the resource.

Success Rate: A random walker is terminated unsuccessfully if none of the nodes visited by it has the resource. The probability that a node visited by a random walker does not have the resource is $(1-p)$. So the probability [3] that a random walker is unsuccessfully terminated is $(1-p)^T$. The probability of unsuccessful termination of k independent random walkers is given by $(1-p)^{k \cdot T}$. Thus the probability of success of a search, employing k random walkers each with TTL T, for a resource with popularity p is given by

$$p_s = 1 - (1-p)^{kT} \tag{1}$$

Expected Overhead: Let O_i denote the overhead incurred by the i^{th} random walker where $i \in [1, 2...k]$. Then,

$$P(O_i = j) = \begin{cases} p(1-p)^{j-1} & j = 1, 2, .., T-1 \\ (1-p)^{T-1} & j = T \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Thus,

$$\begin{aligned} E[O_i] &= \sum_{j=1}^{T-1} j.p(1-p)^{j-1} + T.(1-p)^{T-1} \\ &= \frac{1-(1-p)^{T-1}}{p} + (1-p)^{T-1} \end{aligned} \tag{3}$$

where $E[\cdot]$ denote the expectation operation. Let O be the total overhead incurred by a search that deploys k random walkers. Then

$$\begin{aligned} E[O] = & \quad E[\textstyle\sum_1^k O_i] \\ = & \quad k \cdot (\tfrac{1-(1-p)^{T-1}}{p} + (1-p)^{T-1}) \end{aligned} \tag{4}$$

Expected Delay: Let the delay of a search be equal to D.

$$P(D = j) = \begin{cases} (1-(1-p)^k)(1-p)^{k \cdot (j-1)} & j = 1, 2, \\ & .., T-1 \\ 1-(1-p)^{k \cdot (j-1)} & j = T \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

[3]This result and following results in this section are based on assumption that random walk on P2P network corresponds to independent sampling from uniform distribution of nodes, as shown in [6]. In practice there is some correlation between nodes visited in consecutive steps of random walk. Also the performance metrics would be influenced by placement of resources relative to querying node.

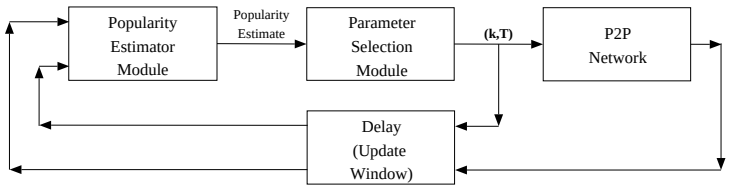

Fig. 1. Block diagram representing the main components of feedback based adaptive search.

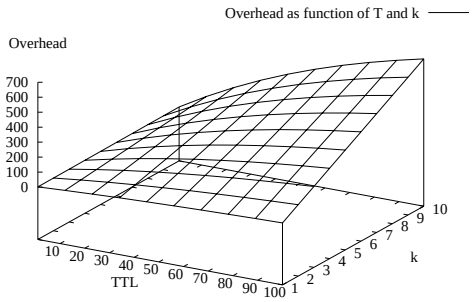

Fig. 2. Expected overhead as a function of parameters k and T for $p = 0.01$.

Hence

$$\begin{aligned} E[D] = & \sum_{j=1}^{T-1} j.(1-(1-p)^k)(1-p)^{k.(j-1)} \\ & +T.(1-p)^{k.(T-1)} \\ = & \frac{1-(1-p)^{k.(T-1)}}{(1-(1-p)^k)} + (1-p)^{k.(T-1)} \end{aligned} \quad (6)$$

V. Equation Based Adaptive Search

EBAS consists of two components as shown in Figure 1; a *parameter selection module* and a *popularity estimator*. They are described in the next two subsections.

A. Parameter Selection Module

The objective of the parameter selection module is to select k and T for discovering a resource with popularity p such that a target performance is guaranteed. Our methodology can be summarized as follows. First, we formulate the desired performance as bounds on the success rate, delay, and overhead. Second, we find the range of feasible (k, T) pairs (if exist) that satisfy the desired bounds. Third, within a feasible set of (k, T) pairs, we characterize the tradeoff between the delay and overhead. Finally, we discuss one possible approach to implement the parameter selection module and apply our methodology to a specific example.

The desired performance of the random search algorithm could be formulated as the following set of constraints:

$$p_s \geq 1-\epsilon, \quad 0 < \epsilon \ll 1 \quad (7)$$

$$E[O] \leq \alpha, \quad \alpha \geq 1 \quad (8)$$

$$E[D] \leq \delta, \quad \delta \geq 1 \quad (9)$$

where ϵ, α and δ are design parameters that characterize the minimum acceptable performance of the search algorithm for a resource with popularity p.

Substituting the above constraints (7), (8) and (9) in (1), (4) and (6), respectively, results in three inequalities that relate k and T with the design parameters for any $0 < p \leq 1$. For example, substituting (7) in (1) yields

$$k \cdot T \geq \frac{log(\epsilon)}{log(1-p)} \quad (10)$$

For the other two inequalities, by substituting (4) and (6) in (8) and (9) respectively, it is not possible to explicitly separate the design parameters on one side and k, T, on the other, and thus must be solved numerically. Let $S(p)$ denote the set of (k, T) feasible pairs for popularity p. $S(p) = \phi$ if there does not exit a (k, T) pair that satisfies each of these three inequalities.

Within a feasible set, there is a tradeoff between overhead and delay. The following numerical example illustrates this tradeoff. For $p = 0.01$, Figures 2 and 3 show how the expected overhead and delay vary with k and T. It is observed that the overhead increases with increase in k and T while delay decreases with increase in k. Thus, it is up to the designer to choose the operating point (i.e. the (k, T) pair) from the feasible range.

The following numerical example illustrates how a (k, T) pair is chosen to satisfy the design constraints. Consider a case when a node needs to search a resource with $p = 0.01$. Suppose that the constraints for the application are given by $\epsilon = 0.05$, $\alpha = 175$ and $\delta = 50$. From (10), $k \cdot T \geq 298$ for satisfying the constraint on success rate. For $k \cdot T = 300$, Figure 4 shows the expected delay and overhead as a function of T. One feasible value of T is 150, and the corresponding k is 2. So the node sends 2 random walkers with TTL set to 150 to search for this resource.

One approach to implement this module is as follows. Each node has a table, called *parameter selection* table with values of k and T corresponding to intervals of popularity. When a node needs to search for a resource with popularity p, it looks up the entry in the parameter selection table corresponding to the interval in which p lies and initiates random walk with parameters k and T specified in the entry. Table V-A is an example of a parameter selection table. So it is ensured that the desired success rate is achieved while overhead and delay are within the specified bounds. This is in contrast to pure random walk whose parameters remain constant often resulting in low success rate or excessive overhead.

B. Maintaining Estimate of Popularity

The popularity of a resource is required for the parameter selection module to set the values of the parameters of random walk. However, the popularity of resources in the network change over time because of arrival and departure of nodes or because of exhaustion/deletion/replication of resources. Centralized methods for informing nodes about popularity of resources would incur large overhead. In this section we describe methods for maintaining popularity estimate with the help of feedback from previous searches, and adapting the parameters of the random walk as a function of this variable

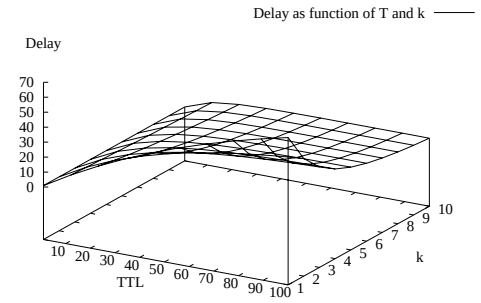

Fig. 3. Expected delay as a function of parameters k and T for $p = 0.01$.

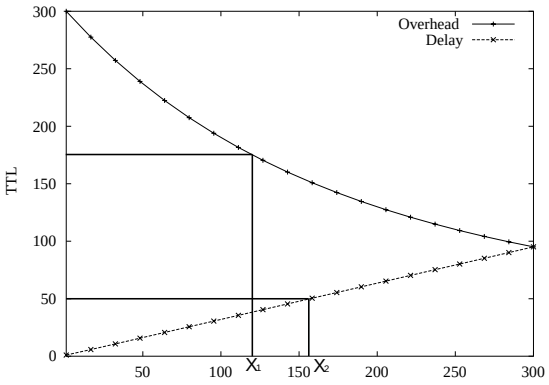

Fig. 4. Figure showing the feasible region $S(p)$. $T \cdot k = 300$ ensures that success rate is over desired value of 0.95.

estimate.

Suppose there are n different resources of interest in the network. Each node maintains popularity estimate of each resource in a *popularity table*. The i^{th} entry of the table corresponds to popularity estimate of i^{th} resource. When a node joins the network, it needs to initialize it's popularity table. The initialization of popularity table is done by selecting a neighbor and querying it for it's popularity table. The neighbor then transfers its current popularity table to the node that has recently joined. Thereafter, the popularity estimate of a resource is updated after feedback from l most recent searches for the resource. The interval between two successive updates is referred to as *update window*.

The algorithm works as follows. Within each update window, say the j^{th} update window, an instantaneous popularity estimate is computed from the parameters k_j, T_j and the success rate during that window. An exponentially weighted moving average is used to update the popularity estimate. New parameters for the random walk k_{j+1} and T_{j+1} for the next (i.e. $j + 1^{st}$) window are then computed by the parameter selection module using the updated popularity estimate.

Specifically, let $\hat{p}_i(j)$ denote the popularity estimate of the i^{th} resource during the j^{th} update window. Let $r_i(j)$ denote the fraction of successful searches in the j^{th} update window. From (1), the instantaneous popularity estimate of the i^{th} resource computed from the searches within the j^{th} window, denoted by $\hat{q}_i(j)$, is given by

$$\hat{q}_i(j) = 1 - e^{\frac{log(1-r_i(j))}{k_j \cdot T_j}} \tag{11}$$

The popularity estimate is updated after each update window using

$$\hat{p}_{(i)}(j+1) = \beta \cdot \hat{p}_i(j) + (1-\beta) \cdot \hat{q}_i(j+1), \quad 0 < \beta < 1 \tag{12}$$

$\hat{p}_{(i)}(j+1)$ is used by the parameter selection module to select k_{j+1} and T_{j+1} that will be used in the next update window.

Popularity Interval	k	T
$[0,\ 1 \times 10^{-3})$	150	15
$[1 \times 10^{-3},\ 2 \times 10^{-3})$	150	13
$[2 \times 10^{-3},\ 3 \times 10^{-3})$	150	11
$[3 \times 10^{-3},\ 4 \times 10^{-3})$	150	9
$[4 \times 10^{-3},\ 5 \times 10^{-3})$	150	8
$[5 \times 10^{-3},\ 6 \times 10^{-3})$	150	4
$[6 \times 10^{-3},\ 7 \times 10^{-3})$	150	4
$[7 \times 10^{-3},\ 8 \times 10^{-3})$	150	3
$[8 \times 10^{-3},\ 9 \times 10^{-3})$	150	3
$[9 \times 10^{-3},\ 1 \times 10^{-2})$	150	2
$[1 \times 10^{-2},\ 1)$	150	2

TABLE I

AN EXAMPLE OF PARAMETER SELECTION TABLE.

VI. SIMULATION RESULTS

This section presents extensive simulations that verify that:

1) The analytical results (Section IV-B) are in agreement with simulation results.
2) The popularity estimator module (Section V-B) effectively maintains an estimate of the variable popularity.
3) The parameter selection module (Section V-A) achieves the target design objectives of equations 7, 8 and 9.
4) EBAS performs better than the non-adaptive random walk.

The simulation scenario and results are presented below.

A. Simulation Scenario

The network graph used for simulations consists of 10^4 nodes. The network is grown using the methodology specified in [7], which is a extension of Barabasi's model [4]. This ensures that the node degree follow power law distribution (defined in [9]) and the network has high clustering coeffient. The average node degree and exponent of power law of the network are 3.5 and 3, respectively, which are close to actual values of Gnutella network as observed in [13] .

In order to populate the network with a resource with popularity p, $p \times 10^4$ nodes are randomly selected and marked to own the resource. In the simulations of EBAS, the update window size, l, is set to 100. In order to make interpretation of results convenient, time is normalized with respect to the update window size. So time $t = j$ refers to the time at which the j^{th} update of popularity estimate takes place.

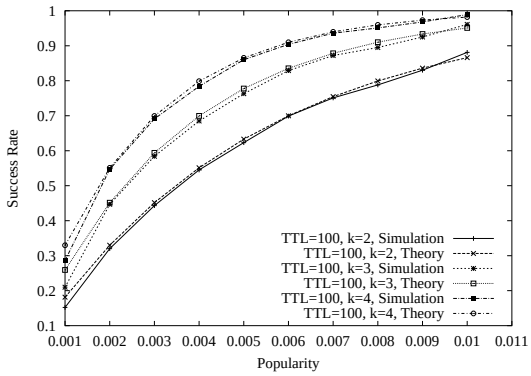

Fig. 5. Comparison of theoretical result for success rate with values obtained from simulation.

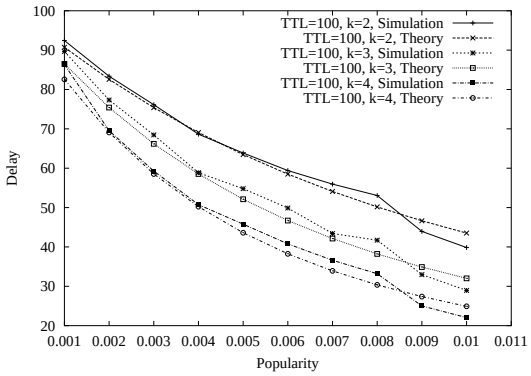

Fig. 7. Comparison of theoretical result for expected delay with values obtained from simulation.

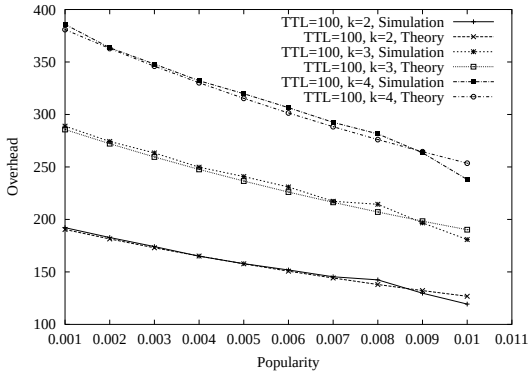

Fig. 6. Comparison of theoretical result for expected overhead with values obtained from simulation.

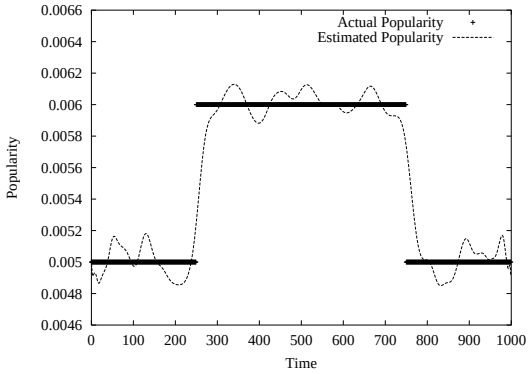

Fig. 8. Estimated popularity closely follows the actual popularity.

B. Verification of Analytical Results

For verifying the analytical results derived in subsection IV-B we simulated pure random walk and compared the values of average success rate, overhead and delay thus obtained against the numerical values obtained from equations (1), (4) and (6) respectively. The average values of performance metrics are obtained by averaging results of 10^4 runs of same search in order to get good estimate of average values.

Figures 5, 6 and 7 show plot of success rate, expected overhead and expected delay, as obtained from (1), (4) and (6), along with average values obtained from simulations. It is observed that simulation results agree closely with the analytical results. The slight deviation in the average values obtained from simulations is because the random walk is statistically similar to independent uniform sampling only after a sufficient number of steps (see [6] for detailed discussion).

C. Performance of Popularity Estimator

In order to test the performance of the popularity estimator we changed the actual popularity of the resource during the simulation and observed how the popularity estimate, calculated using feedback from previous searches, changes with changing popularity. Figure 8 shows the actual popularity and estimated popularity of the resource as obtained from simulation. Initially the actual popularity of the resource is 5×10^{-3} and is increased to 6×10^{-3} at $t = 250$ and later decreased to 5×10^{-3} at $t = 750$. Figure 8 shows that the popularity estimate very closely follows the actual value of popularity. So the proposed popularity estimator is efficient in tracking popularity changes.

D. Satisfaction of Design Objectives

In Section V-A we described the parameter selection module, which chooses the parameters of random walk such that the constraints of (7), (8) and (9) are satisfied. In order to verify this we ran simulations by populating the network with resources of various popularity. The parameters of random walk for discovering each resource are adjusted according to the design constraints. The performance metrics ($\bar{p_s}$, $\bar{O}$ and $\bar{D}$) for each resource were measured. Figure II shows the average values obtained from simulations, along with constraints imposed in choosing the parameters (k, T) for resources with different popularity. The average values obtained from simulation satisfy all the constraints. Thus EBAS fulfills its objective.

E. Comparison of Performance of EBAS with "Pure" Random Walk

We compare EBAS with non-adaptive pure random walk in two scenarios. In one scenario the popularity of the resource increases with time and in the other scenario the popularity decreases with time. In both scenarios, the initial values of the

p	ϵ	α	δ	k	T	$\bar{p_s}$	O	D
0.01	0.05	175	50	2	150	0.95	158	30
0.005	0.05	325	50	3	150	0.96	290	47
0.007	0.05	500	50	4	150	0.97	431	50

TABLE II

THIS TABLE SHOWS THAT THE FRACTION OF SUCCESSFUL SEARCHES, AVERAGE VALUES OF DELAY AND OVERHEAD SATISFY THE BOUNDS IN (7), (8) AND (9) WHEN THE SEARCH PARAMETERS ARE CHOSEN ACCORDING TO THE METHOD DESCRIBED IN SECTION V-A.

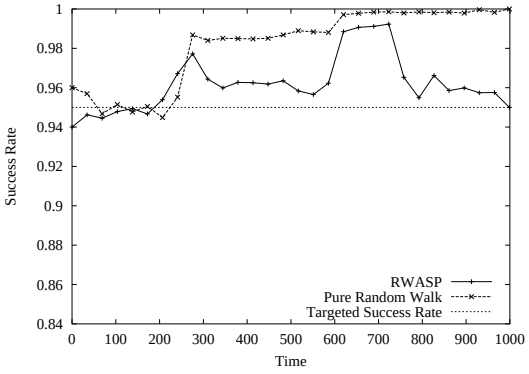

Fig. 9. EBAS maintains success rate above the targeted value for scenario 1.

random walk parameters for EBAS and non-adaptive random walk are equal.

Scenario 1: At $t = 0$ the resource of interest has popularity $p = 0.005$. The popularity is increased to 0.007 at $t = 250$ and further increased to 0.01 at $t = 600$. Figure 9 shows that both EBAS and non-adaptive search strategies maintain success rate above the targeted rate. However, Figure 10 shows that the overhead of EBAS is less than the overhead of the non-adaptive search strategy after the popularity of the resource increases (i.e. after $t = 250$). The initial values of parameters k and T are too aggressive for the higher value of popularity. The adaptive search strategy adjusts the value of parameters causing less overhead while non-adaptive random walk continues searching with fixed k and T thus incurring higher overhead. Figure 11 shows that the number of random walkers decrease adaptively with increasing popularity in case of EBAS while number of random walkers remain constant for the case of pure random walk causing higher overhead.

Scenario 2: At $t = 0$ the resource of interest has popularity 0.01 which is decreased to 0.007 at time $t = 250$ and to 0.005 at $t = 750$. Figure 12 shows that EBAS is able to maintain the success rate above the target value while the success rate of non-adaptive random walk falls much below the acceptable value as the popularity of the resource decreases. Figure 13 shows that in order to maintain success rate above target value, EBAS pays a larger overhead since it adaptively increases the value of search parameters. However, this additional overhead is *necessary* in order to make sure that success rate remains above the required value. Figure 14 shows how the number of random walkers increases adaptively as popularity decreases in case of EBAS while the number of random walkers remains constant in case of pure random walk causing low success rate.

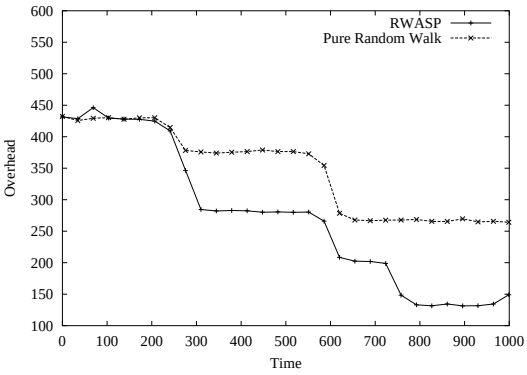

Fig. 10. Comparison of overhead incurred by EBAS and non-adaptive random walk for scenario 1.

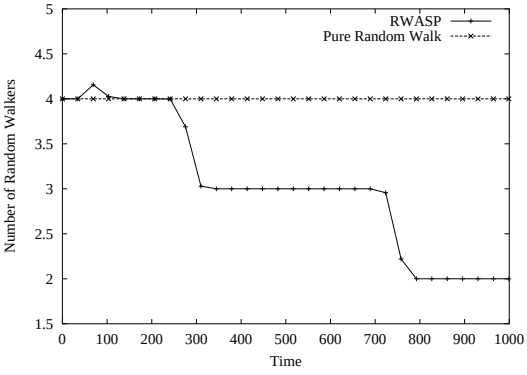

Fig. 11. Comparison of number of random walkers employed by EBAS and non-adaptive random walk for scenario 1.

In summary, the simulation results show that if the parameters of random walk remain fixed then it may lead to large overheads or poor success rate because of changing popularity (or poor initial estimate) of the resource of interest. In contrast, EBAS adjusts the parameters according to the current popularity estimate of the resource thus maintaining the desired performance.

VII. DISCUSSION AND CONCLUSION

In this paper we developed analytical expressions for the success rate, delay and overhead of random search as a function of the random walk parameters and resource popularity. These results were used as a guideline for the design of a search method that uses popularity estimate in order to adaptively adjust the parameters of random walk while maintaining a target performance. Simulation results showed that (i) the analytical expressions match with the simulation results, (ii) our popularity estimation method closely tracks the popularity of the resource as it changes with time, and (iii) the proposed EBAS achieves the target level of performance at negligible overhead.

EBAS stores popularity estimate of all resources in the network. This would require $O(n)$ memory space, where n is the number of resources in the network. State-full search mechanisms, such as APS [14], that maintain rank of neighbors store tables of size $O(n \cdot d)$ where d is the average degree

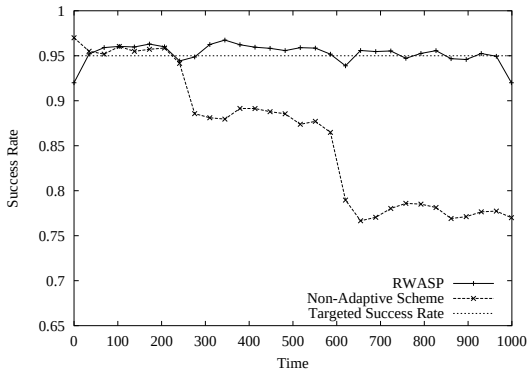

Fig. 12. EBAS maintains success rate above the targeted value even when popularity of the resource decreases (Scenario 2).

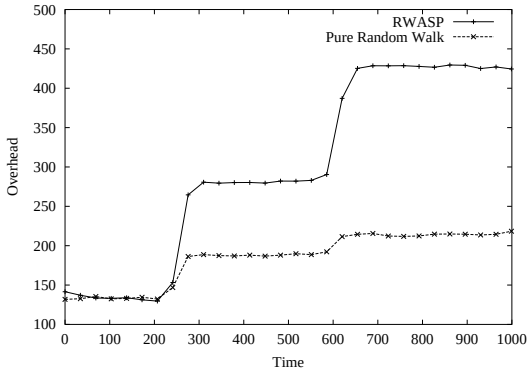

Fig. 13. EBAS pays higher overhead when popularity decreases but maintains a success rate close to the targeted value in scenario 2.

of a node. Thus memory requirement of adaptive random walk is less than other search methods.

Overhead incurred in order to maintain popularity estimates is confined to transfer of popularity estimates to a new node that joins the network. Thus the overhead for arrival of each node is of the order of n. Total overhead caused would depend upon the node arrival rate of the network. This overhead will not effect the scalability of the network if the node arrival rate of the network is not very large. More importantly, the overhead affects only the one-hop neighbor.

This method of adaptively choosing the values of parameters of random walk using popularity estimates can be used along with other state-full search mechanisms like APS. This would further decrease the delay and increase the number of hits of adaptive random walk.

The problem of choosing optimal values of k and T may also be modeled as control theoretic problem. Response of the P2P network to the parameters of random walk is non-linear, thus analysis of performance using control theory is not straightforward. However we could analyze performance mathematically by linearizing the response of P2P network around a specific operating point. The application of control theory to analyze the performance of adaptive random walk is the subject of our future work.

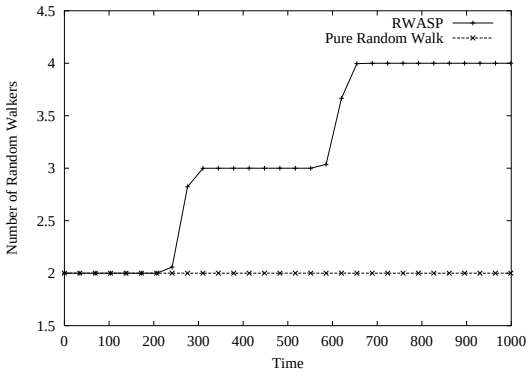

Fig. 14. Comparison of number of random walkers employed by EBAS and non-adaptive random walk for scenario 2.

REFERENCES

[1] Gnutella:to the bandwidth barrier and beyond , http://www.xml.com/pub/r/662.
[2] The Gnutella, http://www.gnutella.com.
[3] L. A. Adamic, R. M.Lukose, B. Huberman, and A. R. Puniyani. Search in Power-Law Networks. *Physical Review*, 64, 2001.
[4] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, pages 43–56, Stockholm, Sweden, August 2000.
[6] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proc. of IEEE INFOCOM*, Mar. 2004.
[7] P. Holme and B. J. Kim. Growing Scale-free Networks with Tunable Clustering. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 65, 2002.
[8] I. Jawahar and J. Wu. A Two Level Random Search Protocol for Peer-to Peer Networks. In *Proceedings of the 8th world Multi-Conference on Systemics, Cybernatics and Informatics*, 2004.
[9] M. A. Jovanovic, F. S. Annexstein, and K. A. Berman. Modelling Peer-To-Peer Network Topologies Through Small World Models and Power Laws. In *IX Telecommunication Forum Telfor, Belgrade*. IEEE, Nov 2001.
[10] M. A. Jovanovico, F. S. Annexstein, and K. A. Berman. Scalability Issues in Large Peer-to-Peer Networks - A Case Study of Gnutella. *University of Cincinnati Technical Report*, 2001.
[11] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A Local Search Mechanism for Peer-to-Peer Networks. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM)*, pages 300–307, 2002.
[12] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 258–259. ACM Press, 2002.
[13] M. Ripeanu and I. Foster. Mapping the Gnutella Network. *IEEE Internet Computing Journal*, 6:50–57, 2002.
[14] D. Tsoumakos and N. Roussopoulos. Adaptive Probabilistic Search for Peer-to-Peer Networks. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P'03)*, pages 102–109. IEEE, sep 2003.
[15] B. Yang and H. Garcia-Molina. Improving Search in Peer-to-Peer Networks. In *Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 5. IEEE Computer Society, 2002.