A Search Theoretical Approach to P2P Networks: Analysis of Learning

Nazif Cihan Taş
Dept. of Computer Science
University of Maryland
College Park, MD 20742
Email: ctas@cs.umd.edu

Bedri Kâmil Onur Taş
Dept. of Economics
TOBB Economics and Technology University
Ankara, Turkey
Email: onurtas@etu.edu.tr

Abstract

One of the main characteristics of the peer-to-peer systems is the highly dynamic nature of the users present in the system. In such a rapidly changing enviroment, end-user guarantees become hard to handle. In this paper, we propose a search-theoretic view for performing lookups. We define a new search mechanism with cost-analysis for refining the lookups by predicting the arrival and leave possibilities of the users. Our system computes a threshold for the number of times that a user has to perform. We compare our results with the naive approach of accepting the first set of results as the basis.

1. Introduction

Peer-to-peer systems are among the most widely known and used computer systems around the world. As the public opinion becomes more and more controversial about the legality of these systems, researchers and computer scientists discovered the potential in the systems as they promise autonomous, scalable, fault tolerant, balanced and anonymous societies.

Today, in addition to the famous Napster [3], Gnutella [2] and FreeNet [1], there are a lot of efficient peer-to-peer designs proposed, such as CAN [6] and Chord [9]. These designs aim to increase the search efficiency by creating high-level logical topologies among the users connected. As such designs focus on increasing the availability through replication, usually, they do not address the end-to-end connection characteristics.

In our paper, we address the problem of selecting download nodes in a peer-to-peer system. Peer-to-peer software available in the market now handle the search request by disseminating the request through the overlay, collecting the results and displaying the results to the end-user as potential candidates for the download. Moreover, some information about the candidate is also given. Then the user choses the download location she desires and the communication begins.

As the previous schema is efficient, it overlooks the dynamic nature of the environment. In highly dynamic environments, such as peer-to-peer systems, the availability of the nodes are not guaranteed. It is highly possible that a user leaves the system before a download process ends, or a new user with very high network resources might enter the system. In these kind of systems, the user should perform another search in order to know about the changes in the system. In this paper, we propose a method to efficiently find a stop criteria for searching. Using this schema, a user applies repetitive searches until the search criteria is met.

This paper is making two contributions. First, by using a novel technique in computer science literature we provide the solution to the search problem in P2P networks. This paper is the first paper that analyzes this question in a search theoretical framework. Second, we also provide the solution where the downloader learns about the probability distribution of the download time of the participants. To best of our knowledge this study is the first study that introduces and analyzes learning dynamics in a P2P network context.

The next chapters are organized as follows: In Section 2, we formulate our problem and represent the computation method for the search stop criteria. In Section 3, we propose a possible extension to the original method. In Section 4, we report on our experiments, and finally in Section 5 we conclude with our results.

2. Problem Formulation

A peer-to-peer (or P2P) computer network is a network that relies on the computing power and bandwidth of the participants in the network rather than concentrating it in a relatively few servers. Since there are many participants, usually there are more than one place that an individual can download a certain file. The individual faces the problem of choosing the place from where she will download the file. Another aspect of P2P networks which makes this problem

Proceedings of the Advanced International Conference on Telecommunications and
International Conference on Internet and Web Applications and Services (AICT/ICIW 2006)
0-7695-2522-9/06 $20.00 © 2006 IEEE

more interesting is there is always a possibility that a new participant with better download speed and better file quality can join the network. So, individual should take into account the possibility of future participants when deciding the download location.

This paper implements *Search Theory with Learning Approach* to produce an algorithm to the search problem that the downloader faces. We deliver the solution to an optimal search problem with learning where the searcher has distinguishable search opportunities. Search theory is a widely-used and developed subject in economics. By implementing the techniques widely used in the economics literature, we provide a simple rule that the downloader can use to pick the location that she will download from. The searcher possessing priors about the distribution of download times can use a search outcome to learn about the search time distribution by updating these priors.

The next section sets up the search problem we consider using a simple model without learning. Section 3 introduces learning into the search problem and provides the solution when the probability distribution of download times is unknown.

2.1. Simple Search Model:

In this section, we develop and solve a basic search model without learning, i.e. the probability distribution of download times is known. The search problem is characterized by a searcher facing a (possibly infinite) number of search opportunities (in our case infinite number of locations that the file can be downloaded from). The solution to the problem involves finding the formula for the reservation download time (R). She should stop her search when the download time of a location is smaller than R. Thus, the algorithm that the downloader (searcher) should follow, involves three steps:

1. Calculate the reservation download time for the file which the downloader would like to download.
2. Stop the search and start download from the location if the download time of that location is smaller than the reservation download time.
3. Run another search if the download times of the locations are larger than R and go back to step 2.

The simple search model that we consider is as the following. Individual runs a search using a search engine in the P2P network. The search engine finds N possible places to download the file. The downloader sorts the download times and decides whether to keep searching or start downloading from the location with the smallest download time.[1] We assume that the downloader can go back and download from a place that she found at previous searches. In this chapter, we assume that the downloader knows the probability distribution of download times, $(F(T))$.

We assume that the downloader gets disutility from download time T_t.

The downloader maximizes the expected payoff minus costs

$$\max_s E\left[e^{-r\tau_s}\left(-T_{\tau_s}\right) - C_s\right] \tag{1}$$

with τ_s being the stopping time under sampling rule S, T_{τ_s} being the smallest download time in τ_s and $E[C_s]$ being the expected discounted sampling costs under S. $e^{-r\tau_s}$ is the discount factor. If learning is Bayesian, the equation (1) is identical to expected utility maximization.

2.2. Nature of the Solution:

Suppose the shortest download time the downloader gets from the previous searches is T, then the expected gain over T from running another search and stopping search with the best download time can be calculated to be

$$\begin{aligned} Q(T) &= T - \left(\beta \int_{-\infty}^{T} T df(x) + \beta \int_{T}^{\infty} x df(x) + c\right) \\ &= \beta \int_{-\infty}^{T} (T - x)\, df(x) + (1-\beta)\, T - c \end{aligned} \tag{2}$$

where $\beta = e^{-r\tau_i} \leq 1$ is the discount factor. c is the time spent for running another search (cost of running another search) and $f(x)$ is the probability density function associated with $F(x)$.

Then the reservation price, R, can be defined as the download time at which the searcher would be indifferent between the following two actions: 1.)Stopping search and download with download time T, and 2.)Running another search and stopping thereafter with the smallest download time, i.e.

$$Q(R) = 0$$

Notice that R can be calculated using the distribution of the download times only, ignoring any value from continued search.

[1] Besides download time the model could include the quality of the file downloaded. To be able to get a simple tractable model we only consider the download time as a factor in the downloaders decision. The model can easily be extended to include the quality of the file downloaded.

3. Optimal Search Strategy with Learning:

The previous section assumes that the probability distribution of the download times $F(T)$ is known. In this section we eliminate this assumption and introduce learning to our model. With learning the distribution of search outcomes (download times), $F(\cdot|D_{t-1})$ now depends on the information contained in the previously observed search outcomes where D_t is the previously observed search outcomes. Thus, now the formula for the reservation time R depends on the available information:

$$Q(T, D_{t-1}) = T - \beta \int_{-\infty}^{T} T df(x|D_{t-1}) \quad (3)$$

$$-\beta \int_{T}^{\infty} x df(x|D_{t-1}) - c$$

$$= \beta \int_{-\infty}^{T} (T - x)\, df(x|D_{t-1}) + (1 - \beta)\, T - c \quad (4)$$

The reservation price $R(D_{t-1})$ is the value of T that solves

$$Q(T, D_{t-1}) = 0 \quad (5)$$

The existence and uniqueness of a solution to equation (4) is shown by Adam [4].

In equation (3) R is a function of current information set D_{t-1}. So, R may therefore change over time as new information becomes available. Yet, how they might change in the future does not enter into the calculation of the reservation prices. Therefore, the reservation prices are independent from the downloader's learning rule.

In this model with learning uncertainty has 2 sources. First, download times are drawn from some probability distribution. Second, there is uncertainty about the prevailing distribution from which offers are drawn. uncertainty about $F(d_t)$ may be represented by beliefs in form of a probability distribution $P(\theta)$ over some parameter θ that indexes the set of possible true probability distribution $F(\cdot/\theta)$ where the true distribution function $F(\cdot)$ is equal to $F(\cdot/\theta)$ for some specific θ. Beliefs,$P(\theta)$, are updated using observed outcomes $P(\theta/D_{t-1})$.

Expected true probability distribution is

$$f(x/D_{t-1}) = E[F(x/\theta/D_{t-1})]$$
$$= \int F(x/\theta)\, P(\theta/D_{t-1})\, d\theta \quad (6)$$

Equation (6) is derived from Bayesian learning. We can give an example using Bayesian learning.

Example: Let there are N possible outcomes and θ_i is probability of observing download time d_i. If learning is Bayesian and the downloader has Dirichlet priors about θ then

$$P(\theta/\alpha_1, \alpha_2 \ldots, \alpha_N) \propto \theta_1^{\alpha_1 - 1} \theta_2^{\alpha_2 - 1} \ldots \theta_N^{\alpha_N - 1} \quad \alpha_i > 0$$

and Talmain [10] shows that the reservation prices are decreasing. Several other learning rules can be examined.

Adam [4] shows that the reservation prices which are based solely on current beliefs are sufficient to determine the optimal sampling strategy. So, with this set up, it is not optimal to give up payoffs and wait and continue to search just to get information and have a better understanding of the distribution. In other words adding learning dynamics into our search model does not change the optimal downloading algorithm. Thus, when the distribution of download times is not known and there is Bayesian learning, Adam [4] indicates that the downloader should calculate the reservation download time, R, using equation (5) and stop the search and start downloading if the download time, d_t, is below R and continue search otherwise. So, in the decision mechanism information dynamics do not matter the downloader only cares about the payoff.

4 Experiments

In this section, we report on our simulation results. We performed simulations on the model without learning and left the experimentation on the learning case as future work. We implemented our simulation testbed in Matlab using a discrete time technique. Our simulation operates from the perspective of a user who searches for a file to download from the system. We started with identifying a time window, w, and assumed that the user makes her search request at the beginning of this time window. Furthermore, she can have the possible download places immediately after the request. Each user picks her entrance time to the system from a uniformly distributed random generator within the window interval. The stay time of the users are computed from an exponentially distributed random variable with mean μ_{stay}. This assignment is consistent with previous work [5].

The download times are the necessary times for the user to complete the download process from that particular user only. As an example, if a user has download time 500, this means that using only that particular network node in the system, the downloading node can have the complete file in 500 seconds. Notice that this value, together with the average file length, represents the bandwidth characteristics of the system. Intuitively, download times depend on the network characteristics and end-user workload. For example, in our simulations, we assumed a file length of 100 Mb.

In a system with 100Kbps bandwidth, this system can be downloaded in 1000 seconds. Hence, with the bandwidth characteristics of general P2P networks [8, 7], it is meaningful to simulate the download times as normally distributed values with mean $\mu_{download-time}$ and standard deviation $\sigma_{download-time}$.

Table 1 gives the assigned values of the parameters which were used unless stated otherwise.

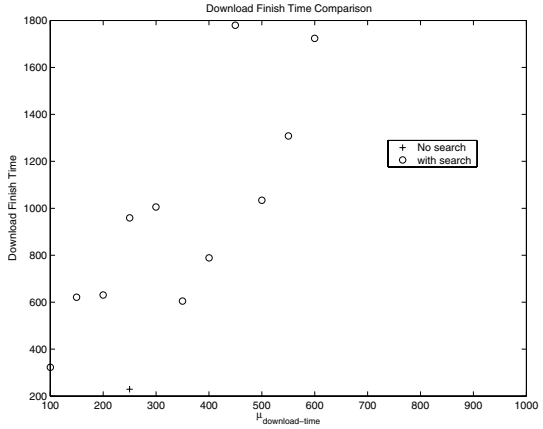

Figure 1. Download Time Completion Comparison

Our first experiment demonstrates the effect of the download times distribution over the goodness of the solution. Figure 1 represents our results. We labeled the classical method with *No Search* tag, and our method with *with Search* tag. This figure shows the time that is needed for the user to complete the file download. We used the parameter assignments as specified in Table 1. However we varied the $\mu_{download-time}$ value between 100 and 1000. We assigned $\sigma_{download-time} = \mu_{download-time}/4$ for all cases.

Parameter	Value
Number of Users	1 M
w	100K
beta	0.9
c	5
File length	100 Mb
μ_{stay}	100
$\sigma_{download-time}$	250
$\mu_{download-time}$	1000

Table 1. Parameter Values for Experiments

We repeated the experiment for 5 times and took the average for all x-values. The missing points represent the incomplete download jobs for at least once in the average process, i.e. in any of the 5 experiment repetitions at least one incomplete download. The graph shows that in each of the discrete steps of x-axis with the exception of $x = 250$, there was at least one incomplete transfer when we do not use our search algorithm. Hence, our first function in the figure consists of only one point. For each of the other x-values, it performs out of range. These incomplete downloads were either because all of the users involving in the download left the system entirely or because the user could not finish the download in the experiment window time. So, using the regular method, the node will immediately start downloading from the list that it gets at the starting time, and as the current node leaves, it jumps to the next best one in the list until it is out of options. Our experimentation shows that the first result almost always performs poorly. The graph shows that when we use the search methodology we are proposing, for mean values less than 600, we always finish the download in reasonable bounds. For values larger than 600, our method starts to perform out of range as well.

Another expected results from this figure is that as the mean value of download times increases, download finish time also increases because, on the average, the serving power of the users decreases.

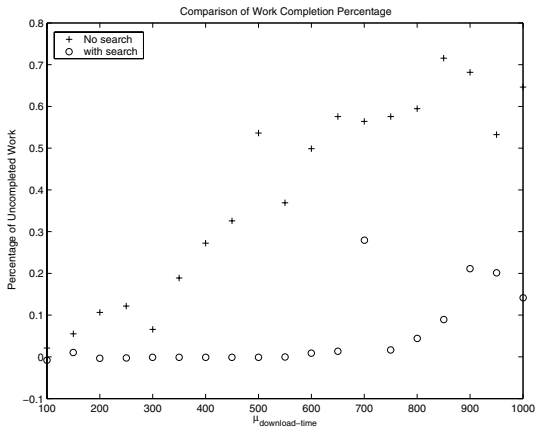

Figure 2. Percentage of Uncompleted Work Comparison

Even though Figure 1 gives us some tips about the nature of the two methods, as we have limited information about the *No search* method, we need additional experimentation. Figure 1 shows us that in a system with large $\mu_{download-time}$ values, no-search option often yields with uncompleted downloads. However, it is natural to expect that result as this μ value increases, the download times goes higher. Our figure shows that our search option could still complete the download before no-search option begins to give uncompleted downloads. In Figure 2, we report on the completion characteristics of two methods. The y-axis in this figure represents the percentage of unfinished download

at the end of the simulation. For example, value 0 means that the download was completed successfully, whereas 0.75 means that only 25% of the work was completed at the end of the simulation. When we use the naive method of no-search, the percentage of the uncompleted work increases much faster than our search method. We believe our method gives better results for two reasons. First, our cost analysis gives us more accurate predictions about the future of the system. And second, since our system introduces some delay before beginning the download, the system becomes more stabilized than the time when the system is started. Since our assumption about the user's starting point is fixed, this is indeed a realistic assumption. However, we believe that there is more experimentation needed for the comparison of these two methods when the user's entrance point to the system can be random too. We leave those experiments as future work and continue with our experimentation results on the sensitivity of our method to the change of the parameters we are using in our simulations.

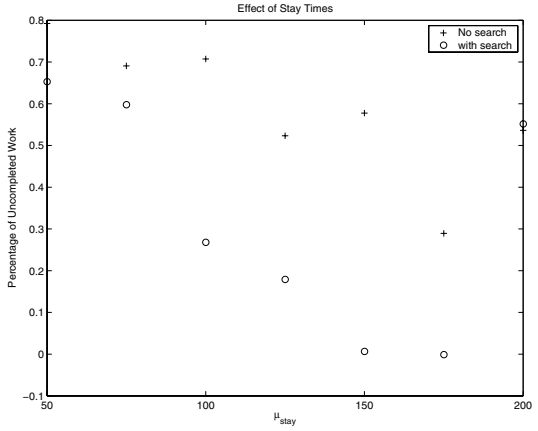

Figure 3. Effect of μ_{stay}

In order to understand the effects of the rapidity of change to effeciency of the system proposed, we also made experiments by varying the stay times of the users. Figure 3 reports on the effect of the μ_{stay} choice to the overall system. As this parameter increases, the average stay time of the users increases, making the average number of users in the system at any time higher. Hence, there is more choice to download the file from and more of the file can be downloaded. However, our search method always outperforms the regular no-search method.

5. Conclusion:

This paper implements *Search Theory with Learning* approach to produce an algorithm to the search problem that the downloader faces. We deliver the solution to an optimal search problem with and without learning where the searcher has distinguishable search opportunities. And, we propose an optimal algorithm that the downloader should follow when she decides from where to download a certain file.

Our experiments show that our method outperforms the naive approach most of the times. The only times that the naive approach performs better is during periods in which very high network resource nodes present. In those cases, the cost of searching more does not pay-off, and it is best to start the download as soon as possible.

References

[1] Freenet homepage.

[2] Gnutella homepage.

[3] Napster homepage.

[4] K. Adam. Learning while searching for the best alternative. *Journal of Economic Theory*, 101(1):252–280, 2001. available at http://ideas.repec.org/a/eee/jetheo/v101y2001i1p252-280.html.

[5] D. G. Deschenes, S. D. Weber, and B. D. Davison. Crawling gnutella: Lessons learned. Technical Report LU-CSE-04-005, Dept. of Computer Science and Engineering, Lehigh University, 2004.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. Technical Report TR-00-010, University of California, Berkeley, CA, 2000.

[7] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network, 2001.

[8] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

[9] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[10] G. Talmain. Search from an unknown distribution: an explicit solution. *Journal of Economic Theory*, 57(1):141–157, 1992.