Search Time in Unstructured Peer-to-Peer Networks With Clustered Demands

Saurabh Tewari, Leonard Kleinrock
Computer Science Department
University of California at Los Angeles
Los Angeles, CA 90095, U.S.A.
{stewari,lk}@cs.ucla.edu

Abstract— **Search time as a function of the number of replicas of a queried object provides a key component to understanding system behavior in peer-to-peer networks. The analytical work in this area so far has assumed a uniform distribution of file replicas throughout the network with an implicit or explicit assumption of uniform file popularity distribution whereas, in reality, there is clear evidence of clustering in file popularity patterns. In this paper, we provide mechanisms for modeling clustering in file popularity distributions and the consequent non-uniform distribution of file replicas. We provide results for the search time in such networks for both random walk and flooding search mechanisms.**

Keywords- Flooding, Peer-to-Peer Networks, Random Walk, Search Time

I. INTRODUCTION

Peer-to-peer networks offer the promise of systems that automatically scale in capacity as the number of users increases and yet are extremely robust, automatically adapting to failures of nodes/links as well as to changes in usage patterns, all at virtually no cost. These loosely organized networks of autonomous entities (user nodes or "peers"), which make their resources available to other peers, represent a new computing paradigm where the service consumers are, now, the service providers as well. So, for example in peer-to-peer file sharing networks, users share files and if one wants to download a file and another user is sharing that file, one would download it directly from that user. Upon obtaining the desired file, one may also begin to share that file allowing other users to download from them. Thus, a file is likely to have multiple replicas in the network with the more popular files having more replicas (i.e. more sources to download the file from). The replication of files provides the robustness while its correlation with popularity provides the automatic scaling according to usage patterns.

This flexibility, however, comes at a cost: one has to find a peer who is sharing the desired file. For a user of a peer-to-peer content distribution system, the measure of system performance is the time it takes to fulfill a request for a particular file which now consists of two components: the time it takes to find who has that content, and the time to actually download the content. The download component of the performance, especially as related to the number of replicas of each file, has been well addressed in [15]. In this paper, we address the average search time, i.e., the average time it takes to find a peer that is sharing the desired file, and explore the relation between the search time for a file and the number of replicas of that file (henceforth in this paper, when we say "search time" we mean "average search time").

Some work has been done in this area assuming a uniform distribution of file replicas. These results are discussed in the next section. Measurements on the deployed peer-to-peer file sharing networks [8] show that there is significant amount of clustering in interests, i.e., the popularity of a set of files in (geographical) regions differs from region to region. Further, most replicas of a file are found in the region where that file is popular. Our first contribution in this paper is a model of peer-to-peer networks that allows for incorporating varying degrees of clustering while retaining a fair amount of analytical tractability. This model is discussed in Section 3. Our second contribution, given in Sections 4 and 5, is the search time with random walk search and with flooding search in peer-to-peer networks with clustering.

II. BACKGROUND AND RELATED WORK

As discussed, the time to find which peer has the desired file is a key performance metric. Hence, the search mechanism to find the desired file in the network has received a lot of attention. One approach is to have a centralized index of the files each peer is sharing where one could just look up which peer has the desired file. However, this introduces a single point of failure and congestion in the system. Another approach is to treat each peer as providing a certain amount of storage to the system and have a one-to-one mapping of the object to be stored in the system and nodes in the system. In this class of networks, called *structured* networks, the nodeId where the desired object is stored (if it was in the network) can be computed so one does not need to "search" for files (finding the node associated with that nodeId still incurs a delay) [10, 13]. Most of the deployed peer-to-peer applications, however, use yet another approach where the nodes have control over the files they share and to obtain the desired file, a peer must query other nodes to find a node that is sharing the desired file[1]. It is

[1] To make the search more efficient, a common variation is to have a "superpeer" keep a list of files shared by 50-100 other "peers".

these *unstructured* peer-to-peer networks that we focus on in this paper.

Since a node cannot realistically keep the addresses of all other peers, each node keeps addresses of a few other peers (called its *neighbors*) through which it reaches the rest of the nodes. That is, an overlay network is constructed to accomplish search tasks. The topology of this search network depends on the peer-to-peer protocol and the typical model used for these networks is the Erdos-Renyi random graph [2] (a power-law random graph is another choice but it distributes the query-processing load unevenly among the peers while yielding faster search methods [4, 12]). The Erdos-Renyi random graph is a good choice when nodes are similar in capacities and file interests (i.e. when files and file popularities are uniformly distributed). We choose this model for our work herein.

A second major design choice is in how the search is conducted over this search network when no information is available about which nodes may have the file. The two main approaches are *flooding* and *random walking*. In flooding, the node that wants the file sends a query to all its neighbors and they, in turn, forward the query to all their neighbors (except the one which sent the query) until a copy of the file is found. In random walking, the query is sent to one randomly selected neighbor and if that neighbor does not have the file, it forwards the query to one of its neighbors (selected randomly) other than the neighbor that sent it the query. Thus, the number of nodes queried with each additional hop grows exponentially in flooding leading to lower search time than random walk where this growth is linear [14].

The two main works on search times are [5, 14]. Both approximate the search time for a file in the network by the average number of hops it takes for a query to reach a node that has the file and we will also use the same approximation for the search time in this paper.

Reference [5] gives the search time for a file as a function of number of replicas of the file when the search method is a random walk. Say there are n_i copies of file i in the network and a total of M nodes in the network. If these n_i copies are uniformly distributed in the network (at most one copy to a node), a randomly selected node has a probability n_i/M of having the file. Thus, random walking for file i is a sequence of Bernoulli trials with n_i/M as the probability of success. Hence the (average) search time for file i with random walk τ_{iR} is:

$$\tau_{iR}(n_i) = M/n_i \quad (1)$$

Reference [14] provides analogous results for flooding and then goes on to compare flooding and random walking and shows the benefits controlled flooding provides over random walking. It gives the flooding search time under the uniform distribution assumption to be:

$$\tau_{iF}(n_i) = \log_d(M/n_i) \quad (2)$$

where τ_{iF} is the (average) search time for file i with flooding, d is the average degree (i.e. the average number of neighbors of each node) of the search network with n_i and M as defined earlier. Intuitively one can interpret this result as follows. A search for file i needs to query M/n_i nodes on average to find the file. Since a random walk queries one additional node per hop, it takes M/n_i rounds to find the file while flooding can query that many nodes in just $\log_d(M/n_i)$ hops because it queries exponentially more nodes with each additional hop[2].

There is considerable work on peer-to-peer networks but due to space constraints we will only mention some of the search-related work here. Reference [1] gives analytical results on search time but when there is only one copy of each object. Flooding search is also analyzed by [17] but they focus on node reachability in hop-limited flooding. Simulation results in [11] also show the logarithmic relationship between search time and number of replicas (as we show in [14]). Our work in this paper gives analytical support to the proposal in [3] of constructing networks with clustering to improve the search process. Our analysis complements work in [9] which uses replication (query result caching) to improve search (the metric is query hit rate in hop-limited flooding search). Random walk search has been studied by many others (e.g. [4], [7], [12]) but the network model and/or the metrics of interest were different.

III. NETWORK MODEL

As discussed above, we already have the search time expressions for both random walk and flooding searches if the file distribution is uniform. Our task in this paper is to investigate the effect *clustering* has on search times. As mentioned, measurements on real systems indicate that most replicas of a file are located in the regions where the file is more popular. Therefore, intuitively one would expect that average search times should be shorter in the presence of clustering (there are more replicas nearby in regions where most requests for that file are made).

Let us assume that our peer-to-peer network has M nodes and that these M nodes are clustered in, say, L clusters. For ease of discussion, we make the following assumptions. Each cluster is of the same size (thus, each cluster has M/L nodes). There are only two levels of popularity of each file and there is only one cluster where a file is more popular. We assume that the cluster where a file is more popular will have more copies of that file i.e. if the n_i copies of file i are split as n_{ia} in the cluster where the file is popular and n_{ib} in each of the remaining clusters, where $n_i=n_{ia}+(L-1)n_{ib}$, $n_{ia}<M/L$ and $n_{ia}> n_{ib}$. One may then say that the cluster where file i is more popular has a *higher density* of file i whereas a cluster where the file is not as popular has a *lower density*. Since clustering has already been accounted for, we assume that within each cluster the files are uniformly distributed over all the nodes in the cluster.

One possible model for the search network is to assume that the clusters are totally disconnected (i.e. there are no inter-cluster links) and within each cluster, the network follows the Erdos-Renyi random graph topology. For this model of

[2] Since a node does not forward a query twice, the exponential growth assumption is optimistic. Thus, (2) slightly underestimates the actual search time. In [16], we provide simulation plots for the average search distance for different topologies as well as an analytical proof for (2) when $M\to\infty$ and n_i/M is small. Our work in [16] indicates that (2) is an approximate expression for the search time which captures the dependence of search time on the number of replicas very well while underestimating the search time by a small amount.

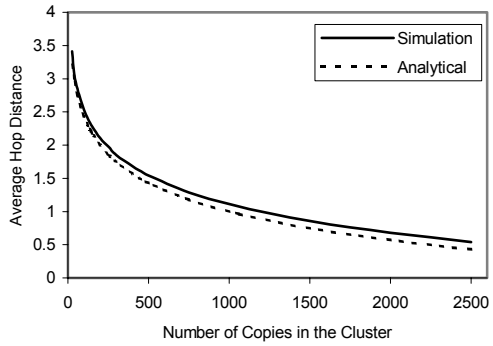


Figure 1. Search Time with Perfect Clustering (25,000 node network, 5 equal-sized clusters, Average Degree 5)

clustering, the search time expressions can be obtained from our earlier results [14] for the uniform file distribution case. Specifically, the search time for flooding when a node in the high density cluster initiates the search, τ_{iFa}, is:

$$\tau_{iFa}(n_{ia}, n_{ib}) = \log_d(M/n_{ia}L) \quad (3)$$

while the search time for flooding when a node in the low density cluster initiates the search, τ_{iFb}, is:

$$\tau_{iFb}(n_{ia}, n_{ib}) = \log_d(M/n_{ib}L) \quad (4)$$

Similarly, the search time for a random walk when a node in the high density cluster initiates the search, τ_{iRa}, is:

$$\tau_{iRa}(n_{ia}, n_{ib}) = M/(n_{ia}L) \quad (5)$$

whereas the search time for a random walk when a node in the low density cluster initiates the search, τ_{iRb}, is:

$$\tau_{iRb}(n_{ia}, n_{ib}) = M/(n_{ib}L) \quad (6)$$

Comparing (3)-(6) to (1), (2) we see that perfect clustering reduces the random walk search time (and the query-processing load for both flooding and random walk [14]) by a factor of L while the flooding search time decreases by $\log_d L$. Fig. 1, where we compare simulation results with (3), shows that (3) captures the effect of the number of replicas very well (since (3) is based on (2), the slight underestimation by (3) is expected).

While assuming disconnected clusters makes for an easy first-order analysis, actual peer-to-peer networks do not have such fully disconnected clusters. There is evidence of strong clustering but intercluster links do exist in real networks so neither an Erdos-Renyi random graph over the entire network nor the fully disconnected clusters model is an appropriate topology. Therefore we must define a new kind of topology that is an intermediate between these two extremes. Ideally, we would like to define a continuum of topologies with the Erdos-Renyi random graph at one extreme and the fully disconnected clusters at the other extreme. One such topology is the following random graph variant. Consider a network in which the probability of including an intra-cluster link is p and the probability of including an inter-cluster link is q and the

TABLE I. NOTATIONS USED

M	Number of nodes
L	Number of clusters
d	Average degree of the search overlay topology
q	Probability of any given pair of inter-cluster nodes having a direct link
n_i	Number of replicas of file i in the entire network
n_{ia}	Number of replicas of file i in the "high-density" cluster
n_{ib}	Number of replicas of file i in the "low-density" cluster
τ_{iR}	Average search time for file i with random walk search
τ_{iF}	Average search time for file i with flooding search
τ_{iRa}	Average search time for file i from high-density cluster with random walk search
τ_{iRb}	Average search time for file i from low-density cluster with random walk search
τ_{iFa}	Average search time for file i from high-density cluster with flooding search
τ_{iFb}	Average search time for file i from low-density cluster with flooding search

average per-node degree is d as before i.e. assuming L clusters of equal sizes, the nodes are partitioned into L clusters and the probability that any given pair of intra-cluster nodes is connected is p and the probability that any given pair of inter-cluster nodes are connected is q. Thus, each node has an average of $(M/L)p$ links to nodes within its cluster and $(M-M/L)q$ links to nodes outside its cluster. Hence, the average degree $d = (M–M/L)q + (M/L)p$ and if one were to hold the average degree constant, defining one of p or q defines the other. Notice that this topology does provide the desired continuum of topologies from the completely disjoint clusters (with $q=0$) at one extreme and the random graph ($p=q$) at the other. A side benefit for flooding search is that it does not matter whether a search process is at a node in the higher-density cluster or the lower-density cluster, it will expand to d other nodes in higher or lower-density clusters in the next hop. Thus, the average number of nodes queried per search expands exponentially and the d^{τ} expression for number of nodes queried given the average search distance of τ [14] still holds.

We summarize the notation used in this paper in Table 1.

IV. RANDOM WALK SEARCH IN NETWORKS WITH CLUSTERING

In the case of no clustering and the case of disconnected clusters we discussed so far, a search only queried nodes of the "same type" (i.e. all the nodes queried by the search had the same probability of having the desired file). However, this is not the case in the clustered peer-to-peer network as the existence of inter-cluster links implies that a query can get forwarded to a node in a different cluster where the probability of a node having the file may be different. Thus, in our model, when a query is forwarded, the event of interest is whether it goes to a node in the high-density cluster or to a node in one of the low-density clusters. Among the d outgoing links at each node, the probability that a link is an inter-cluster link is $q(M-M/L)/d$. Therefore, for a query at a node in the higher-density cluster, the probability of one query path "escaping" to a lower-

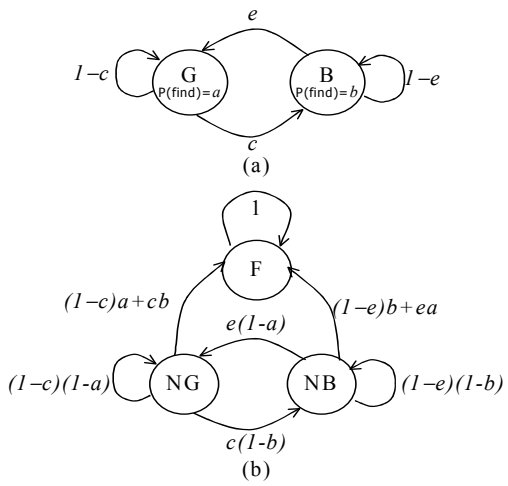


Figure 2. Random walk in the modified random graph for the non-uniform file distribution case

density cluster is $c = q(M-M/L)/d$. In contrast, when the query is at a node in the lower-density cluster, the probability of escaping to the higher density cluster is $e = q(M/L)/d$ as there are only M/L nodes that are of interest for this event. For ease of discussion, throughout the rest of the paper, we refer to the nodes within the higher-density cluster as "good" nodes, and the nodes in the lower-density clusters as "bad" nodes.

Fig. 2a shows a Markov chain model for the random walk on our modified random graph with a non-uniform file distribution prior to finding the file: state G represents the random walk being at a "good" node and state B represents the random walk being at a "bad" node. The random walk transitions between state G and state B until it finds the file. The probability of finding the file when the system transitions to state G (i.e. at a good node) is $a = n_{ia}L/M$, and the probability of finding the file when the system transitions to state B (i.e. at a bad node) is $b = n_{ib}L/M$. Since we need to determine the average number of steps until the file is found for the random walk search time, we transform our Markov chain in Fig. 2a to that in Fig. 2b. The state NG denotes the event that the search visits a good node but does not find the file and the state NB denotes the event that the search visits a bad node but does not find the file. State F is an absorbing state denoting the event that the file is found independent of whether the previous node is good or bad. Thus, the average first passage time from state NG to state F is the search time for a random walk search initiated by a good node, τ_{iRa}, and the average first passage time from state NB to state F is the search time for a random walk search initiated by a bad node, τ_{iRa}.

The relevant equations [6], therefore, are:

$$\tau_{iRa} = 1 + (1\text{-}c)(1\text{-}a)\,\tau_{iRa} + c(1\text{-}b)\,\tau_{iRb}$$

$$\tau_{iRb} = 1 + e(1\text{-}a)\,\tau_{iRa} + (1\text{-}e)(1\text{-}b)\,\tau_{iRb}$$

Therefore:

$$\tau_{iRa} = \frac{(c+e)(1-b)+b}{ab+cb(1-a)+ae(1-b)} = [a - \frac{c(a-b)}{b(1-c-e)+(c+e)}]^{-1}$$

$$\tau_{iRb} = \frac{(c+e)(1-a)+a}{ab+cb(1-a)+ae(1-b)} = [b + \frac{e(a-b)}{a(1-c-e)+(c+e)}]^{-1}$$

Substituting the values for a, b, c and e, we get the following theorem:

Theorem 1. The (average) search time for a random walk search in the clustered peer-to-peer network defined in Section 3 is:

$$\tau_{iRa}(n_{ia}, n_{ib}) = [\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{(n_{ib}L/M)(d-Mq)+Mq}]^{-1} \qquad (7)$$

if the search is initiated at a node in the high-density cluster, and is:

$$\tau_{iRb}(n_{ia}, n_{ib}) = [\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{(n_{ia}L/M(d-Mq)+Mq}]^{-1} \qquad (8)$$

if the search is initiated at a node in the low-density cluster. ■

Comparing (7), (8) with (5), (6) respectively, we see that the search time for a query initiated by a good node increases if cross-cluster links are present but if a bad node initiated the query, the search time decreases. As expected, if there were no cross-cluster links (i.e. q=0), (7), (8) revert to (5), (6) respectively. Further, in the uniform distribution case, $n_{ia} = n_{ib} = n_i/L$ and (7) and (8) revert to (2) as expected.

V. Flooding Search In Networks with Clustering

Unlike the case of no clustering where we found in Section 2 that the flooding search time is the logarithm of the random walk search time, in networks with clustering the mapping between flooding and random walk is not straightforward. Clustering implies more intra-cluster links than inter-cluster links. Therefore, if a query gets to a good node, it is more likely to have come from a good node than a bad node i.e. $P(G|G) > P(G|B)$ or $1\text{-}c > e$. Similarly, a query getting to a bad node is more likely to have come from a bad node than from a good node i.e. $P(B|B) > P(B|G)$ or $1\text{-}e > c$. Thus, searching from a good node, flooding is likely to see more good nodes than a random walk upon querying the same number of nodes[3], and searching from a bad node, flooding is likely to see more bad nodes than a random walk upon querying the same number of nodes[4]. Thus, a flooding search initiated by a good node is

[3] For example, say, the average degree is 3 and let us compare the average number of good nodes among the next 3 nodes queried by a good node. The average number of good nodes with flooding, $n_F = 3(1-c)^3 + 2[3(1-c)^2c] + [3c^2(1-c)]$. The average number of good nodes with random walk, $n_R = 3(1-c)^3 + 2[2ce(1-c)+c(1-c)^2] + [(1-c)c(1-e)+c(1-e)e+c^2e]$. Thus, $n_F - n_R = 4[(1-c)^2c - ce(1-c)] + [3c^2(1-c) - (1-c)c(1-e) - c(1-e)e - c^2e] = c(c^2 - 4c+3+2ce-4e+e^2) = c[(1-c)^2+2(1-c)(1-e)+(1-e)^2-1] = c[(2-c-e)^2-1] = c(1-c-e)(3-c-e) > 0$ since $1-c > e$.

[4] Using the example of average degree 3 again, we compare the average number of bad nodes among the next 3 nodes queried by a bad node. The average number of bad nodes with flooding, $n_F = 3(1-e)^3 + 2[3(1-e)^2e] + [3e^2(1-e)]$. The average number of bad nodes with random walk, $n_R = 3(1-e)^3 + 2[2ce(1-e)+e(1-e)^2] + [(1-c)e(1-e)+c(1-c)e+e^2c]$. Thus, $n_F - n_R = e(1-c-e)(3-c-e) > 0$ since $1-e > c$.

likely to query more good nodes in $\log_d N$ steps than a random walk search would in N steps starting at the same node. Hence,

$$\tau_{iFa}(n_{ia}, n_{ib}) < \log_d[\tau_{iRa}(n_{ia}, n_{ib})] \quad (9)$$

Similarly, a flooding search initiated by a bad node will query more bad nodes in $\log_d N$ steps than a random walk search will in N steps starting at the same node and hence

$$\tau_{iFb}(n_{ia}, n_{ib}) > \log_d[\tau_{iRb}(n_{ia}, n_{ib})] \quad (10)$$

Thus, in networks with clustering, the random walk search times only provide us with bounds[5] on one side for the flooding search times. These bounds, however, are useful since getting an exact expression for the average search time is very difficult. The best we can do is to bound the search time on the other side as well.

Let us first attempt to obtain a lower bound on the flooding search time for a search initiated at a good node. The difficulty in getting an exact expression is that at hop distance > 1, the query could be at bad nodes as well as good nodes and computing the relative distribution of these nodes is hard. Since we want a lower bound, a crude approach is to ignore all the "bad" possibilities and assume that even after hop distance > 1, the nodes that are forwarding the queries are all good nodes. With this assumption, at any hop distance ≥ 1, when a node queries one of its neighbors, the probability that the file is found is $P(F|NG)$. Hence, the search time for a flooding search from a good node is no better than $-\log_d[P(F|NG)] = \log_d[(1-c)a+cb] = -\log_d[a-c(a-b)]$. Thus[6],

$$\tau_{iFa}(n_{ia}, n_{ib}) > -\log_d[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}] \quad (11)$$

We can use the same approach to find an upper bound for τ_{iFb}, the search time for a flooding search initiated at a bad node. We can ignore all the "good" possibilities and assume that even after hop distance > 1, the nodes forwarding the queries are all bad nodes. With this assumption, at any hop distance ≥ 0, when a node queries one of its neighbors, the probability that the file is found is $P(F|NB)$. Hence, the search time for a flooding search from a bad node is no worse than $-\log_d[P(F|NB)] = -\log_d[(1-e)b+ea] = -\log_d[b+e(a-b)]$. Thus,

$$\tau_{iFb}(n_{ia}, n_{ib}) < -\log_d[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}] \quad (12)$$

Combining (9, 10, 11, 12), we get the following theorem:

Theorem 2. The search time for a flooding search in the clustered peer-to-peer network defined in Section 3 is *approximately*[5,6] bounded by

$$-\log_d[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}] < \tau_{iFa}(n_{ia}, n_{ib}) < -\log_d[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{(n_{ib}L/M)(d-Mq)+Mq}] \quad (13)$$

if the search is initiated at a node in the high-density cluster, and is *approximately*[5,6] bounded by

$$-\log_d[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{(n_{ia}L/M)(d-Mq)+Mq}] < \tau_{iFb}(n_{ia}, n_{ib}) < -\log_d[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}] \quad (14)$$

if the search is initiated at a node in the low-density cluster. ■

Comparing (13), (14) to (3), (4) respectively, we see that the presence of cross-cluster links increases the search time for a query initiated by a good node and decreases the average search time for a query initiated by a bad node. Since the lower and upper bounds differ only in the denominator of the term incorporating the effect of clustering, the bounds will be tight unless $(1-x)(d-Mq)$ is large where $x=n_{ib}L/M$ for (13) and $x=n_{ia}L/M$ for (14) or, in other words, when n_{ib} or n_{ia} are very small or q is small (in which case (11), (12) provide a good approximation). We also see that the bounds become equal in 3 cases: when $q=0$, when $n_{ia}= n_{ib}$, and when $d-Mq=0$. q=0 implies the clusters are disjoint so we revert to (3) and (4) as expected. The other two cases have important implications. When, $n_{ia}= n_{ib} = n_i/L$ (i.e. the file distribution is uniform) both bounds again become equal to (1). However, (1) was under the assumption of an Erdos-Renyi random graph search network whereas our network can have an arbitrary degree of clustering. In the $d=Mq$ case also, the bounds become equal and we revert to (1) even though our file distribution has clustering but the search network is an Erdos-Renyi graph as assumed for (1).

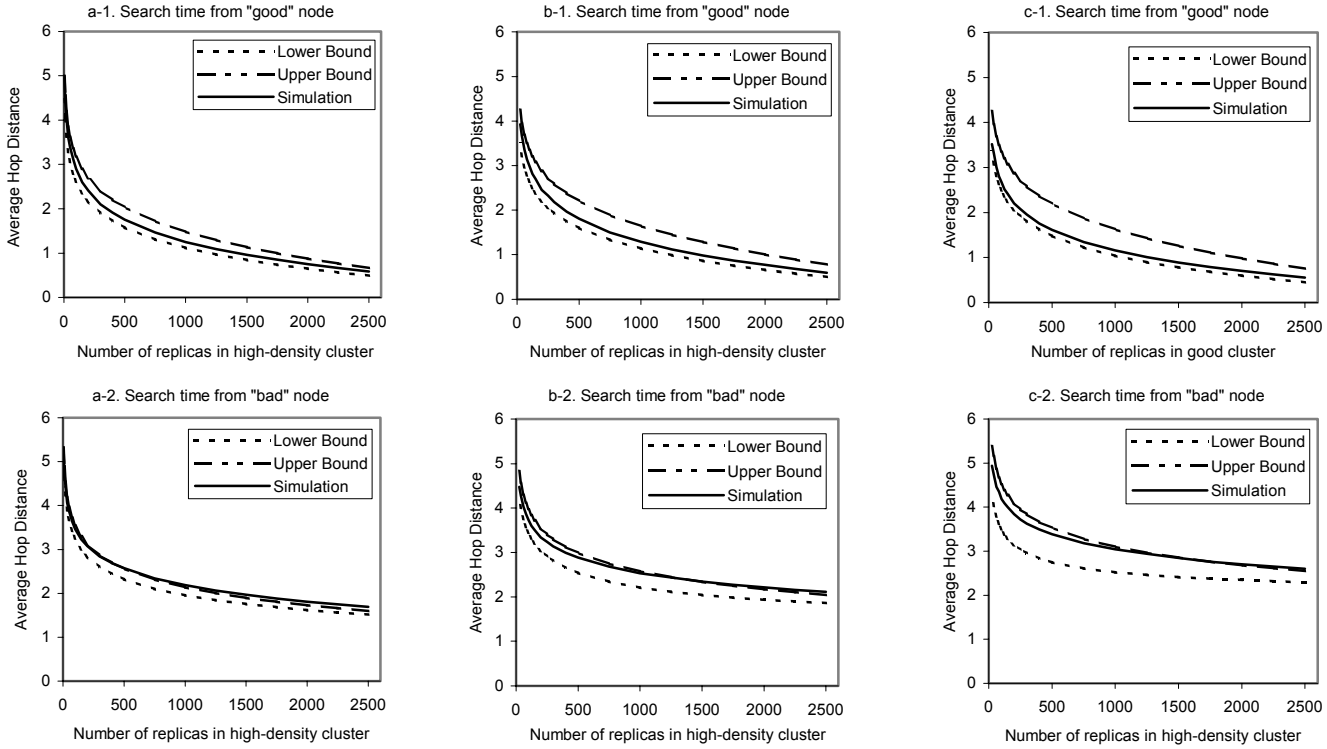
In Fig. 3 we compare the bounds in (13), (14) to simulation results under varying degrees of clustering in inter-cluster link probability and the ratio of replica density in the high-density cluster to that in the low-density cluster. As expected we find that the bounds are tight under moderate clustering (Fig. 3a) and as clustering becomes stronger (Fig. 3b,c) the bounds start to separate but the search time gets closer to (11), (12). Thus, in either case we have a good estimation of the average search time. Finally, we also note that the search time slightly exceeds the approximate upper bound is as expected[5].

VI. CONCLUSION

In this paper, we investigated the relationship between the number of replicas of a file in unstructured peer-to-peer networks and the search time for that file and substantially expanded the existing knowledge on this topic. We provided a model to incorporate clustering in peer-to-peer network models so they better reflect real networks. We were able to find an

[5] The bounds presented in this section are approximate bounds as the underlying analytical approach (Section 2) underestimates the search time by a small amount (see Fig. 1). Thus, the actual search times should lie within the given bounds plus a small offset.

[6] Since the probability of finding the file at hop distance 0 is $P(F)$ whereas the expression $-\log_d[P(F|NG)]$ assumes $P(F|NG)$ to be the probability at all hop distances including 0 [16], a correction factor of $-[1-P(F)]/[1-P(F|NG)]$ is required. Since this correction factor is negligible when the probability that the querying node itself has the file is small, we omit this from (11). A similar correction factor applies in the case of a flooding search from a "bad" node but its magnitude is even smaller and hence we omit it from (12) as well.



(a) 70% links intra-cluster, Replicas-in-low density cluster = 0.1*Replicas-in-high-density-cluster

(b) 70% links intra-cluster, Replicas-in-low density cluster = 0.01*Replicas-in-high-density-cluster

(c) 90% links intra-cluster, Replicas-in-low density cluster = 0.01*Replicas-in-high-density-cluster

Figure 3. Flooding Search Time Simulation vs. Bounds (25,000 node network, 5 equal-sized clusters, Average Degree 5, Varying Degree of Clustering)

exact expression for the random walk search time in a peer-to-peer network with clustering. We were also able to find bounds on the flooding search time in these networks. Using these bounds, we extend the previously known results for flooding search time which assumed a uniform file distribution and an Erdos-Renyi random graph to when the file distribution is not uniform but the search network is an Erdos-Renyi random graph, and when the file distribution is uniform but the search network has clustering. Even though there is still room for improvement in our results, they provide the peer-to-peer system designers a valuable set of tools to make informed design choices on questions such as how many replicas they would like to have for a file, or the TTL hop limit to set for TTL-scoped flooding searches.

REFERENCES

[1] Baryshnikov, Y., Coffman, E., Jelenkovic, P., Momcilovic, P., Rubenstein, D., "Flood Search Under the California Split Rule," Operations Research Letters, Vol. 32, No. 3, May 2004.

[2] Bollobas. B, Random Graphs, Academic Press, London, 1985.

[3] Cholvi, V., Felber, P.A., Biersack, E.W., "Efficient Search in Unstructured Peer-to-Peer Networks," European Transactions on Telecommunications, Vol. 15, Issue 6, November-December 2004.

[4] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N. and Shenker, S. "Making Gnutella-like P2P Systems Scalable," in Proc. of ACM SIGCOMM, August 2003.

[5] Cohen, E. and Shenker, S., "Replication Strategies in Unstructured Peer-to-Peer Networks," in Proc. of ACM SIGCOMM, August 2002.

[6] Feller, W., An Introduction to Probability Theory and Its Applications, Vol. 1, John Wiley & Sons, Inc., New York, 1950.

[7] Gkantsidis, C., Mihail, M., Saberi, A., "Random Walks in Peer-to-Peer Networks," in Proc. of ACM INFOCOM, March 2004.

[8] Le Fessant, F., Handurukande, S., Kermarrec, A. M., Massouli, L., "Clustering in Peer-to-Peer File Sharing Workloads," in Proc. of IPTPS, February 2004.

[9] Lindemann, C., Waldhorst, O.P., "A Distributed Search Service for Peer-to-Peer File Sharing in Mobile Applications," in Proc. of IEEE Peer-to-Peer Computing, September 2002.

[10] Rowstron, A. I. T., Druschel, P., "Pastry: Scalable, Decentralized Object Location, And Routing For Large-Scale Peer-To-Peer Systems," in Proc. of IFIP/ACM Middleware, November 2001.

[11] Rubenstein, D., Sahu, S., "Can Unstructured P2P Protocols Survive Flash Crowds?," IEEE/ACM Trans. on Networking, April 2005.

[12] Sarshar, N., Oscar Boykin, P., Roychowdhury, V. P., "Percolation Search in Power Law Networks: Making Unstructured Peer-To-Peer Networks Scalable," in Proc. of IEEE Peer-to-Peer Computing, September 2003.

[13] Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H., "Chord: A Scalable Peer-To-Peer Lookup Service For Internet Applications," in Proc. of ACM SIGCOMM, August 2001.

[14] Tewari, S., Kleinrock, L. "Analysis of Search and Replication in Unstructured Peer-to-Peer Networks," in Proc. of ACM SIGMETRICS, June 2005.

[15] Tewari, S., Kleinrock, L. "On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks," in Proc. of IFIP Networking, May 2005.

[16] Tewari, S., Kleinrock, L. "Analysis of Search and Replication in Unstructured Peer-to-Peer Networks," UCLA Computer Science Dept Technical Report UCLA-CSD-TR050006, March 2005.

[17] Zhang, X., Song, G., Zhang, Q., Zhu, W., "Performance Analysis in Unstructured Overlays," in Proc. of IEEE ICC, May 2003.