UCLA-CSD-TR050040

Optimal Search Performance 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—**This paper derives the optimal search time and the optimal search cost that can be achieved in unstructured peer-to-peer networks when the demand pattern exhibits clustering (i.e. file popularities vary from region to region in the network). Previous work in this area had 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. The potential performance benefit that the clustering in demand patterns affords is captured by our results. Interestingly, the performance gains are shown to be independent of whether the search network topology reflects the clustering in file popularity. We also provide the relation between the query-processing load and the number of replicas of each file for the clustered demands case showing that flooding searches may have lower query-processing load than random walk searches in the clustered demands case.**

Keywords- Flooding, Peer-to-Peer Networks, Random Walk, Optimal Search Time, Optimal Search Cost, Clustered Demands

I. INTRODUCTION

Peer-to-peer networks are loosely organized networks of autonomous entities (user nodes or "peers") which make their resources available to other peers. Since each new peer brings additional resources, these networks are fully scalable provided that the resources one offers can be found by the peers who need those resources. Thus, finding the desired resource is a critical issue in peer-to-peer networks. Keeping a centralized index of the resources each peer is offering is an approach that has scalability issues and a single point of failure. Alternatively, a direct approach for finding the desired resource is to have the peer wanting a resource to query other nodes to find a node that has that resource. Since a node cannot realistically keep the addresses of all other peers, an overlay network is constructed where each node keeps addresses of a few other peers (called its *neighbors*) through whom it reaches the rest of the peers. Peer-to-peer networks following this approach are referred to as *unstructured* peer-to-peer networks to distinguish them from structured networks [7, 9] which map each unique resource to a particular node in the network, an approach that can be more efficient but whose lack of flexibility introduces other issues [6]. In this paper we focus on unstructured peer-to-peer networks and address two major concerns in these networks: the time to find a peer who is offering a particular resource (the *search time*), and the amount of additional traffic introduced in the network in the process of locating the peer that is offering that resource (the *search cost*). The reference example is of peer-to-peer file sharing networks and we refer to resources as files throughout the rest of the paper.

As is typical in the related literature [10], we 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 that file, and use *average search time*, i.e., the average time it takes to find a peer that is sharing the desired file, as our first metric for search performance. Our second metric is the search cost. Since a search for a file is done via peers sending query messages to other peers, the number of query messages each peer processes equals the additional traffic introduced in the network by a query. Therefore, we approximate the search cost by the *query-processing load*, i.e., the average number of nodes that are queried per file request. One expects that if many peers are sharing a file, in any reasonable search method, the search time and the search cost for the file will be smaller than if very few peers were sharing that file. In the extreme case, if all nodes could store all files, no search would be required. Since each peer has finite storage space, a system designer seeks to get the optimum search performance possible given the per-node storage constraint. The optimal average search time, the optimal query-processing load and the *file replica distribution* (number of replicas of each file as a function of that file's popularity) at the respective optima have been derived in [10, 11] under the assumption of a uniform distribution of the file replicas. However, measurements on deployed peer-to-peer file sharing networks show a significant amount of clustering in interests [5], i.e., the popularity of a set of files in (geographical) regions differs from region to region. Further, more replicas of a file are found in those regions where that file is more popular.

The main contributions of this paper are given in Sections 3-7. In Section 3, we present a peer-to-peer network model that allows for incorporating clustering in demand and file replica distribution. The search time for a random walk search in this network model is derived in Section 4 while Section 5 derives the analogous results for the flooding search. In Section 6, we derive the query-processing load expressions for random walk and flooding searches. The optimal search time and the optimal search cost expressions are derived in Section 7. Based on our observations in Section 5, in Section 8, we extend the results for flooding search time beyond the specific mode of demand clustering allowed by our network model described in Section 3 to incorporate any arbitrary demand clustering. Section 9 discusses certain properties of the optimal search time and the optimal search cost. Related work, including the results in [10, 11], is discussed in Section 2 and our conclusions are given in Section 10.

II. Background and Related Work

Flooding and *random walking* are the two main alternatives in how the search is conducted over the search network when no information is available about which nodes may have the file. 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.

When nodes are similar in capacities and file interests (i.e. when files and file popularities are uniformly distributed), the Erdos-Renyi random graph [1] is a good topology model[1] for the overlay search network. The optimal search performance under the constraint of finite per-node storage is covered well by [3, 10, 11] with the assumption of uniform distribution of file replicas. Reference [3] 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, for random walk search, the (average) search time for file i, τ_{iR}, and the (average) number of nodes queried per search for file i, Q_{iR}, is M/n_i. Reference [11] provides analogous results for flooding and before going on to compare flooding and random walking and showing the advantage of a controlled flooding search over a random walk search. It gives the flooding search time under the uniform distribution assumption to be $\tau_{iF}(n_i) = \log_d(M/n_i)$ 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 (i.e. the (average) number of nodes queried per flooding search for file i, Q_{iF}, is still M/n_i). 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 $\log_d(M/n_i)$ hops because it queries exponentially more nodes with each additional hop[2].

[1] When node capacities are very skewed, a power-law random graph is a topology choice which distributes the query-processing load unevenly among the peers but yields faster search methods [2, 8].

We summarize these results in Table 2. Table 1 gives the notation used in the paper. In this paper, we seek to obtain results analogous to those in Table 2 when the file replica distribution and the demand patterns are not uniform. Since each link is equiprobable in an Erdos-Renyi random graph, it is not suited for modeling clustering in file interests and we develop a network model that incorporates clustering in file interests as well as the network topology in the next section.

TABLE I. Notation Used

M	Number of nodes
L	Number of clusters
N	Number of unique files
K	Per-node storage size (in number of files)
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 a "low-density" cluster
λ_i	Request rate of file *i* per node (averaged over the network)
λ_{ia}	Request rate of file *i* per node in the "high-density" cluster
λ_{ib}	Request rate of file *i* per node in a "low-density" cluster
λ	$= \sum_{i=1}^{N} \lambda_i$
τ_{ix}	Average search time for file *i* with search method *x* [a]
Q_{ix}	Query-processing load for file *i* with search method *x* [a]
τ_{ixa}	Average search time for file *i* from the high-density cluster with search method *x* [a]
Q_{ixa}	Query-processing load for file *i* from the high-density cluster with search method *x* [a]
τ_{ixb}	Average search time for file *i* from a low-density cluster with search method *x* [a]
Q_{ixb}	Query-processing load for file *i* from a low-density cluster with search method *x* [a]
τ_x^{opt}	Optimal average search time with search method *x* [a]
Q_x^{opt}	Optimal query-processing load with search method *x* [a]
$Q_x^{\tau opt}$	Query-processing load with the replica distribution that minimizes the average search time with search method *x* [a]

[a] For flooding search: x=F, For random walk search: x=R e.g. τ_{iFb}=Average search time for file *i* from a low-density cluster with flooding search

[2] Since a node does not forward a query twice, the exponential growth assumption is optimistic. Thus, (1) slightly underestimates the actual search time. In [11], we provide simulation plots for the average search distance for different topologies as well as an analytical proof for (1) when $M \to \infty$ and n_i/M is small. Our work in [11] indicates that (1) 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.

TABLE II. RESULTS FOR UNIFORM DISTRIBUTION OF REPLICAS ([11])

Replica Distribution	Equation	
Valid for arbitrary replica distributions	$\tau_{iF}(n_i) = \log_d(M/n_i)$	(1)
	$Q_{iF}(n_i) = Q_{iR}(n_i) = \tau_{iR}(n_i) = M/n_i$	(2)
$n_i \propto \lambda_i$	$\tau_F^{opt} = -\sum_{i=1}^{N} \frac{\lambda_i}{\lambda} \log_d \frac{\lambda_i}{\lambda} - \log_d K$	(3)
	$Q_F^{\tau opt} = \frac{N}{K}$	(4)
$n_i \propto \sqrt{\lambda_i}$	$Q_F^{opt} = Q_R^{opt} = \tau_R^{opt} = \frac{(\sum_{i=1}^{N} \sqrt{\lambda_i})^2}{\lambda K}$	(5)

III. A PEER-TO-PEER NETWORK MODEL FOR CLUSTERED DEMANDS

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 in which a file is more popular. Thus, for all files $i = 1$ to N, file i has request rate λ_{ia} per node in one cluster and λ_{ib} per node in each of the remaining $L-1$ clusters where $\lambda_{ia} > \lambda_{ib}$ and $M\lambda_i = \frac{M}{L}\lambda_{ia} + (L-1)\frac{M}{L}\lambda_{ib}$. where λ_i is the average node request rate for file i across the entire network. Let us further assume that the n_i replicas of file i are split as n_{ia} replicas in the cluster where the file is more popular and n_{ib} replicas in each of the remaining clusters where $n_{ia}>n_{ib}$, $n_i = n_{ia}+(L-1)n_{ib}$ and $n_{ia}<M/L$. One may then say that the cluster where file i is more popular has a *higher density* of file i replicas 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 that 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 clustering, the search time and the query-processing load expressions can be obtained from the analogous expressions for the uniform distribution case in Table 2 with (1) and (2) yielding (6), (7) and (8), (9) respectively as shown in Table 3. Comparing (6)-(9) 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 by a factor of L while the flooding search time decreases by $\log_d L$. Fig. 1, where we compare simulation results with (6), shows that (6) captures the effect of the number of replicas very well (since (6) is based on (2), the slight underestimation by (6) is expected).

TABLE III. SEARCH PERFORMANCE WITH DISCONNECTED CLUSTERS

Derived from	Equation	
(1)	$\tau_{iFa}(n_{ia}, n_{ib}) = \log_d(M/n_{ia}L)$	(6)
	$\tau_{iFb}(n_{ia}, n_{ib}) = \log_d(M/n_{ib}L)$	(7)
(2)	$Q_{iFa}(n_{ia}, n_{ib}) = Q_{iRa}(n_{ia}, n_{ib}) = \tau_{iRa}(n_{ia}, n_{ib}) = M/n_{ia}L$	(8)
	$Q_{iFb}(n_{ia}, n_{ib}) = Q_{iRb}(n_{ia}, n_{ib}) = \tau_{iRb}(n_{ia}, n_{ib}) = M/n_{ib}L$	(9)

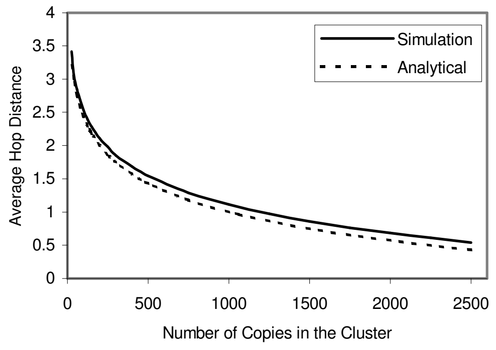


Figure 1. Search Time with Perfect Clustering (25,000 node network, 5 equal-sized clusters, Average Degree 5)

While assuming disconnected clusters makes for an easy first-order analysis, actual peer-to-peer networks do not typically have such fully disconnected clusters. There is evidence of strong clustering but inter-cluster 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. A topology model that gives us a continuum of topologies with the Erdos-Renyi random graph at one extreme and the fully disconnected clusters at the other extreme 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 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 is 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. Varying q provides a continuum of topologies from the completely disjoint clusters ($q=0$) to the Erdos-Renyi random graph ($p=q$). A flooding search in these topologies expands to d other nodes (in the higher-density or a lower-density cluster) in the next hop independent of whether the search process is at a node in the higher-density cluster or a lower-density cluster. Thus, the average number of nodes queried per search expands exponentially and the d^{τ} expression for the number of nodes queried given the average search distance of τ [11] still holds.

In subsequent sections we derive the search performance expressions similar to those listed in Table 2 for this network.

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

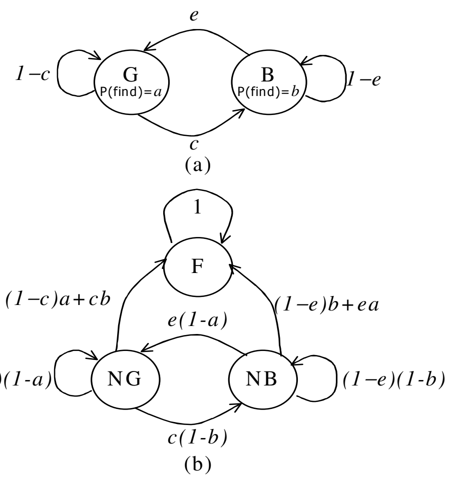


Figure 2. Random walk in the modified random graph for the non-uniform file distribution case

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-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 [4], therefore, are:

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

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

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:

$$\tau_{iRa}(n_{ia}, n_{ib}) = [\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{(n_{ib}L/M)(d-Mq)+Mq}]^{-1} \quad (10)$$

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

$$\tau_{iRb}(n_{ia}, n_{ib}) = [\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{(n_{ia}L/M(d-Mq)+Mq}]^{-1} \quad (11)$$

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

Comparing (10) and (11) with (8) and (9) 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), (10) and (11) revert to (8) and (9) respectively. Further, in the uniform distribution case, $n_{ia} = n_{ib} = n_i/L$ and (10) and (11) 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 $1-c > e$. 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 $1-e > c$. 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

[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 $1-c > e$.

nodes than a random walk upon querying the same number of nodes[4]. Thus, a flooding search initiated by a good node is 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,

$$\tau_{iFa}(n_{ia}, n_{ib}) < \log_d[\tau_{iRa}(n_{ia}, n_{ib})] \tag{12}$$

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

$$\tau_{iFb}(n_{ia}, n_{ib}) > \log_d[\tau_{iRb}(n_{ia}, n_{ib})] \tag{13}$$

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],

$$\tau_{iFa}(n_{ia}, n_{ib}) > -\log_d[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}] \tag{14}$$

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,

$$\tau_{iFb}(n_{ia}, n_{ib}) < -\log_d[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}] \tag{15}$$

Combining (12, 13, 14, 15), 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

$$-\log_d[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}] \;<\; \tau_{iFa}(n_{ia}, n_{ib}) \;<\; -\log_d[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{(n_{ib}L/M)(d-Mq)+Mq}] \tag{16}$$

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

$$-\log_d[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{(n_{ia}L/M)(d-Mq)+Mq}] \;<\; \tau_{iFb}(n_{ia}, n_{ib}) \;<\; -\log_d[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}] \tag{17}$$

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

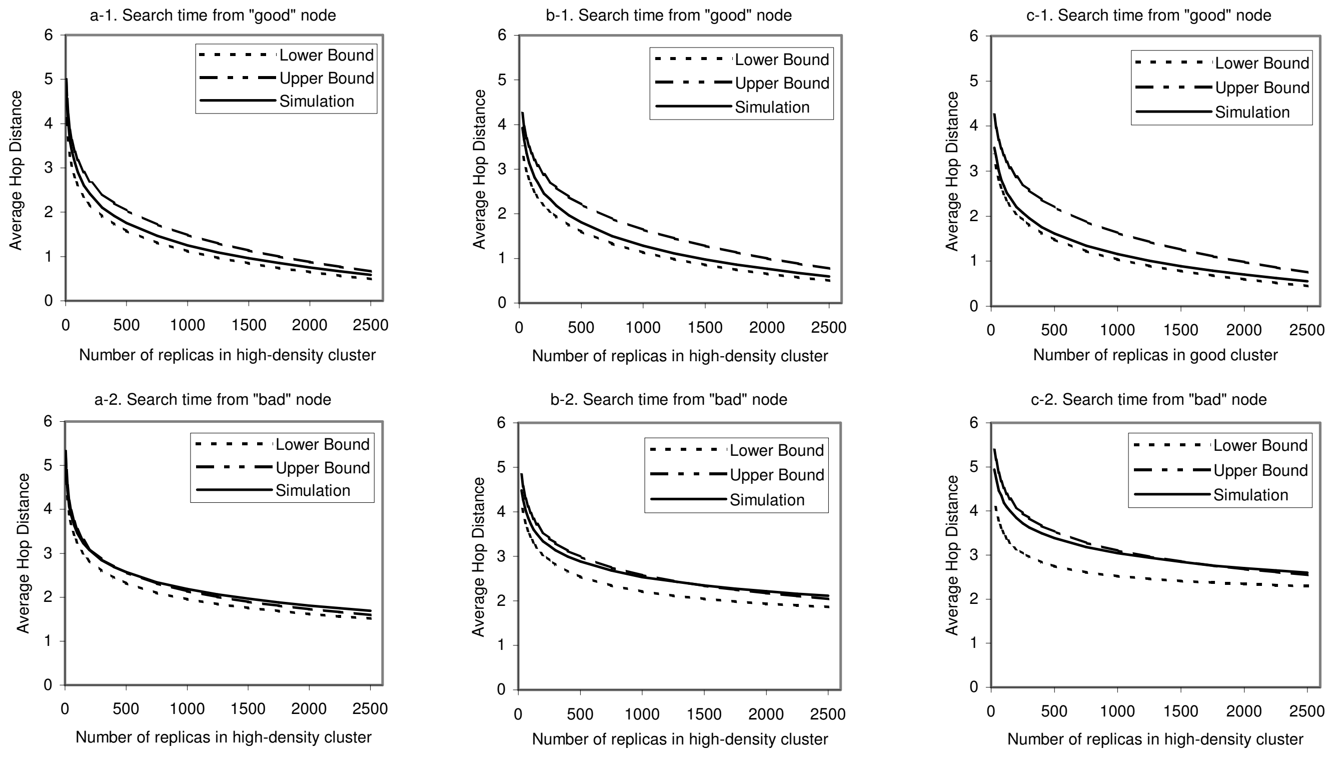
Comparing (16) and (17) to (6) and (7) respectively, we see that the presence of cross-cluster links increases the search time for a query initiated by a good node and decreases the average search time for a query initiated by a bad node. Since the lower and upper bounds differ only in the denominator 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 (16) and $x=n_{ia}L/M$ for (17) or, in other words, when n_{ib} or n_{ia} are very small or q is small (in which case (14) and (15) 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 (6) and (7) 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 (16) and (17) 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 (14) and (15). Thus, in either case we have a good estimation of the average search time. We also note that the search time slightly exceeds the approximate upper bound is as expected[5].

[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, $n_F = 3(1-e)^3 + 2[3(1-e)^2 e] + [3e^2(1-e)]$. 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 $1-e > c$.

[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, 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 (14). 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 (15) 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)

Based on the simulation results in Fig. 3, we conclude that (14) and (15) are a reasonable approximation for the flooding search time in the network model for clustered demands described in Section 3. Therefore, throughout the rest of the paper, we use

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

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

for our analysis. Given that these expressions are reasonable approximations, we utilize the intuition behind these to extend our results to unstructured peer-to-peer networks with arbitrary demand clustering pattern (but with the same clustering in network topology clustering) in Section 8.

VI. Query-Processing Load with Clustered Demands

As discussed earlier, for the network model described in Section 3, the query-processing load in the network can be estimated by d^{τ} when the average search distance is τ. Hence, using the results in Section 5, we obtain:

Theorem 3: The query-processing load for a flooding search in the clustered peer-to-peer network defined in Section 3 is

$$Q_{iFa}(n_{ia}, n_{ib}) \sim [\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}]^{-1} \qquad (18)$$

for searches initiated in the high-density cluster, and is

$$Q_{iFb}(n_{ia}, n_{ib}) \sim [\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}]^{-1} \qquad (19)$$

for searches initiated in a low-density cluster. ■

Notice that unlike the uniform distribution case, the query-processing load for the flooding search and the random walk search are different now. In fact, we can show that:

Corollary 1: For the clustered peer-to-peer network defined in Section 3, (a) From the high-density cluster, a flooding search has a lower query-processing load than a random walk search whereas (b) From a low-density cluster, a flooding search has a higher query-processing load than a random walk search i.e. for searches for file *i*,

$$Q_{iRa}(n_{ia}, n_{ib}) > Q_{iFa}(n_{ia}, n_{ib})$$

$$Q_{iRb}(n_{ia}, n_{ib}) < Q_{iFb}(n_{ia}, n_{ib})$$

Proof:

Let $a = n_{ia}L/M$, $b = n_{ib}L/M$, $c = q(n_{ia}-n_{ib})/d$, $e = Mq/d$. Then $Q_{iFa} = [a-c(L-1)]^{-1}$, $Q_{iRa} = [a-c(L-1)/[b(1-e)+e]]^{-1}$ and $Q_{iFb} = [b+c]^{-1}$, $Q_{iRb} = [b+c(L-1)/[a(1-e)+e]]^{-1}$. Since a < 1, $b < 1$ and $1-e > 0$, we get $b(1-e)+e < 1$ and $a(1-e)+e < 1$. $b(1-e)+e < 1$ $\Rightarrow$ $c(L-1)/[b(1-e)+e] > c(L-1)$ $\Rightarrow$ $a-c(L-1)/[b(1-e)+e] < a-c(L-1)$ $\Rightarrow$ $Q_{iRa}(n_{ia}, n_{ib}) > Q_{iFa}(n_{ia}, n_{ib})$. Similarly, $a(1-e)+e < 1$ $\Rightarrow$ $c/[a(1-e)+e] > c$ $\Rightarrow$ $b+c/[a(1-e)+e] > b+c$ $\Rightarrow$ $Q_{iRb}(n_{ia}, n_{ib}) < Q_{iFb}(n_{ia}, n_{ib})$. ■

We observe that, while the request rate in the high-density cluster, λ_{ia}, should be larger than the request rate in a low-density cluster, λ_{ib}, it is not clear whether, for arbitrary replica distributions, the lower query-processing load offered by a flooding search in the high-density cluster offsets the higher query-processing load incurred by the flooding search in the low-density cluster after weighting the query-processing costs by λ_{ia} and λ_{ib} respectively with $\lambda_{ia} > \lambda_{ib}$.

Corollary 1 also suggests that, for arbitrary replica distributions, it may be better to use flooding searches in the high-density cluster and random walk searches in the low-density clusters (at the cost of significantly larger search times for searches from the low-density clusters). A simple search algorithm could specify using flooding for a short hop-limit (which would allow completion of flooding searches from the high-density cluster) and then switching to some other approach (e.g. using a structured network as [13] suggests) to limit the query-processing cost for searches from the low-density cluster.

VII. Search Performance Optimization

The optimal average search time τ_x^{opt} and the optimal query-processing load Q_x^{opt} over all file requests in the entire network are:

$$\tau_x^{opt} = \sum_{i=1}^{N}\left[\frac{1}{L}\frac{\lambda_{ia}}{\lambda}\tau_{ixa} + (1-\frac{1}{L})\frac{\lambda_{ib}}{\lambda}\tau_{ixb}\right] \qquad (20)$$

$$Q_x^{opt} = \sum_{i=1}^{N}\left[\frac{1}{L}\frac{\lambda_{ia}}{\lambda}Q_{ixa} + (1-\frac{1}{L})\frac{\lambda_{ib}}{\lambda}Q_{ixb}\right] \qquad (21)$$

where x=F for flooding search and =R for random walk search.

For the case of disconnected clusters, where (6)-(9) are the relevant expressions, the same steps as in [15] can be followed for the search performance optimization to yield the optimal results summarized in Table 4.

TABLE IV. Optimal Search Performance: Disconnected Clusters

Optimal Replica Distribution	**Equation**[a]
$\{n_{ia} \propto \lambda_{ia},\ n_{ib} \propto \lambda_{ib}\}$	$\tau_F^{opt} = -\sum_{i=1}^{N}\frac{1}{L}\frac{\lambda_{ib}}{\lambda}\log_d\frac{\lambda_{ib}}{\lambda} - \sum_{i=1}^{N}\left(1-\frac{1}{L}\right)\frac{\lambda_{ib}}{\lambda}\log_d\frac{\lambda_{ib}}{\lambda} - \log_d K$ (22)
	$Q_F^{\tau opt} = \frac{N}{K}$ (23)
$\{n_{ia} \propto \sqrt{\lambda_{ia}},\ n_{ib} \propto \sqrt{\lambda_{ib}}\}$	$Q_F^{opt} = Q_R^{opt} = \tau_R^{opt} = \frac{1}{\lambda K}\left[\sum_{i=1}^{N}\left(\frac{1}{L}\sqrt{\lambda_{ia}} + \left(1-\frac{1}{L}\right)\sqrt{\lambda_{ib}}\right)\right]^2$ (24)

[a] As in [16], we still have the constraints that $n_{ia} \geq 1$, $n_{ib} \geq 1$, $n_{ia} \leq M/L$, $n_{ib} \leq M/L$ and, hence, (22), (23) hold if $L/KM \leq \lambda_{ia}/\lambda \leq L/K$ and $L/KM \leq \lambda_{ib}/\lambda \leq L/K$ $\forall i$ (see [15] for conditions under which (24) holds).

The optimization procedure in the general clustering case is the same as in [15] but the solution is harder to obtain. We summarize the optimization results for flooding search in Theorems 5 and 6. The results for random walk search optimization are not provided.

A. *Average Search Time Optimization for Flooding Search*

The Lagrangian for the average search time optimization is

$$H = -\sum_{i=1}^{N}\left[\frac{1}{L}\frac{\lambda_{ia}}{\lambda}\log_d(\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}) + (1-\frac{1}{L})\frac{\lambda_{ib}}{\lambda}\log_d(\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d})\right] + \gamma(\sum_{i=1}^{N}[n_{ia} + (L-1)n_{ib}] - KM)$$

Differentiating w.r.t n_{ia} and n_{ib} respectively, we obtain:

$$-\frac{1}{\lambda\ln d}\left[\frac{\frac{\lambda_{ia}}{L}[\frac{L}{M} - \frac{q(L-1)}{d}]}{\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}} + \frac{\lambda_{ib}(1-\frac{1}{L})\frac{q}{d}}{\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}}\right] + \gamma = 0$$

$$-\frac{1}{\lambda\ln d}\left[\frac{\frac{\lambda_{ia}}{L}\frac{q(L-1)}{d}}{\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}} + \frac{\lambda_{ib}(1-\frac{1}{L})(\frac{L}{M} - \frac{q}{d})}{\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}}\right] + \gamma(L-1) = 0$$

These equations are satisfied[7] by

$$n_{ia}L/M - q(L-1)(n_{ia}-n_{ib})/d = k\lambda_{ia} \qquad (25)$$

$$n_{ib}L/M + q(n_{ia}-n_{ib})/d = k\lambda_{ib} \qquad (26)$$

where k is a constant whose value is to be determined. Thus, at the optimal replica distribution, $\lambda_i = \lambda_{ia}/L + (1-1/L)\lambda_{ib} = [n_{ia}/M - q(1-1/L)(n_{ia}-n_{ib})/d + (L-1)n_{ib}/M + q(1-1/L)(n_{ia}-n_{ib})/d]/k = [(L-1)n_{ib}+n_{ia}]/Mk = n_i/Mk$, i.e. $n_i \propto \lambda_i$. $KM = \sum_{i=1}^{N} n_i = Mk\sum_{i=1}^{N}\lambda_i = M\lambda k \Rightarrow k = K/\lambda$. The following theorem summarizes these results.

Theorem 5: The average search time for a flooding search in the clustered peer-to-peer network defined in Section 3 is minimized when

$$n_{ia} = [KM(d\lambda_{ia} - Mq\lambda_i)]/[\lambda L(d - Mq)] \qquad (27)$$

$$n_{ib} = [KM(d\lambda_{ib} - Mq\lambda_i)]/[\lambda L(d - Mq)] \qquad (28)$$

if $L/KM \leq \lambda_{ia}/\lambda \leq L/K$ and $L/KM \leq \lambda_{ib}/\lambda \leq L/K$ $\forall i$, and at the replica distribution defined by these equations,

$$n_i = K\lambda_i/\lambda$$

$$\tau_F^{opt} = -\sum_{i=1}^{N}[\frac{1}{L}\frac{\lambda_{ia}}{\lambda}\log_d(\frac{\lambda_{ia}}{\lambda}) + (1-\frac{1}{L})\frac{\lambda_{ib}}{\lambda}\log_d(\frac{\lambda_{ib}}{\lambda})] - \log_d K$$

$$Q_F^{\tau opt} = N/K$$

i.e. the optimal average search time is independent of q (the level of clustering in search network topology) while the query-processing load when the average search time is minimized is independent of the skew in file popularity and the level of clustering in both the search network and the file popularity. ■

[7] $\frac{1}{k}[\frac{1}{M} - \frac{q}{d}(1-\frac{1}{L})] + \frac{q}{dk}(1-\frac{1}{L}) = \gamma\lambda\ln d$

$\frac{q}{dk}(1-\frac{1}{L}) + \frac{1}{k}(1-\frac{1}{L})[\frac{L}{M} - \frac{q}{d}] = \gamma(L-1)\lambda\ln d$

B. Query-Processing Load Optimization for Flooding Search

The Lagrangian for the query-processing load optimization is $H = \sum_{i=1}^{N}\left[\frac{1}{L}\frac{\lambda_{ia}}{\lambda}[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}]^{-1}\right.$

$\left. + (1-\frac{1}{L})\frac{\lambda_{ib}}{\lambda}[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}]^{-1}\right] + \gamma(\sum_{i=1}^{N}[n_{ia}+(L-1)n_{ib}] - KM)$

Differentiating w.r.t n_{ia} and n_{ib} respectively, we obtain:

$$\frac{-\frac{\lambda_{ia}}{\lambda L}[\frac{L}{M} - \frac{q(L-1)}{d}]}{[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}]^2} + \frac{-\frac{\lambda_{ib}}{\lambda}(1-\frac{1}{L})\frac{q}{d}}{[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}]^2} + \gamma = 0$$

$$\frac{-\frac{\lambda_{ia}}{\lambda L}\frac{q(L-1)}{d}}{[\frac{n_{ia}L}{M} - \frac{q(L-1)(n_{ia}-n_{ib})}{d}]^2} + \frac{-\frac{\lambda_{ib}}{\lambda}(1-\frac{1}{L})(\frac{L}{M} - \frac{q}{d})}{[\frac{n_{ib}L}{M} + \frac{q(n_{ia}-n_{ib})}{d}]^2} + \gamma(L-1) = 0$$

Comparing these equations to those yielding (25) and (26), we can see that these equations will be satisfied by

$$n_{ia}L/M - q(L-1)(n_{ia}-n_{ib})/d = k'\sqrt{\lambda_{ia}} \quad (29)$$

$$n_{ib}L/M + q(n_{ia}-n_{ib})/d = k'\sqrt{\lambda_{ib}} \quad (30)$$

where k' is a constant whose value is to be determined. Thus, at the optimal replica distribution, $n_i = n_{ia}+(L-1)n_{ib} = (M/L)(n_{ia}L/M - q(L-1)(n_{ia}-n_{ib})/d + (L-1)[n_{ib}L/M + q(L-1)(n_{ia}-n_{ib})/d]) = (k'M/L)[\sqrt{\lambda_{ia}}+(L-1)\sqrt{\lambda_{ib}}]$. Using $n_i = (k'M/L)(\sqrt{\lambda_{ia}}+(L-1)\sqrt{\lambda_{ib}})$ in $\sum_{i=1}^{N} n_i = KM$, we get $k' = K\Big/\sum_{i=1}^{N}[\frac{1}{L}\sqrt{\lambda_{ia}} + (1-\frac{1}{L})\sqrt{\lambda_{ib}}]$ yielding the following theorem.

Theorem 6: The query-processing load for a flooding search in the clustered peer-to-peer network defined in Section 3 is minimized when

$$n_{ia} = \frac{(dL-Mq)\sqrt{\lambda_{ia}} - Mq(L-1)\sqrt{\lambda_{ib}}}{L(d-Mq)\sum_{i=1}^{N}[\sqrt{\lambda_{ia}}+(L-1)\sqrt{\lambda_{ib}}]}KM \quad (31)$$

$$n_{ib} = \frac{[dL-Mq(L-1)]\sqrt{\lambda_{ib}} - Mq\sqrt{\lambda_{ia}}}{L(d-Mq)\sum_{i=1}^{N}[\sqrt{\lambda_{ia}}+(L-1)\sqrt{\lambda_{ib}}]}KM \quad (32)$$

assuming λ_{ia} and λ_{ib} are such that $1 \le n_{ia} \le M/L$, $1 \le n_{ib} \le M/L$ $\forall i$ in these equations, and at this replica distribution

$$Q_F^{opt} = \frac{1}{K}\left(\sum_{i=1}^{N}[\frac{1}{L}\sqrt{\frac{\lambda_{ia}}{\lambda}} + (1-\frac{1}{L})\sqrt{\frac{\lambda_{ib}}{\lambda}}]\right)^2$$

i.e. the optimal query-processing load is independent of q (the level of clustering in search network topology). ■

As noted in the theorems, the optimal search performance is independent of the level of clustering in search network topology. Thus, the results for the optimal search performance provided in Table 4 for disconnected clusters hold for the general clustered demands network model *except* for the file replica distributions needed for the optimal search performance which are now defined by (27) and (28) for the optimal average search time i.e. (22) and (23) and by (31) and (32) for the optimal query-processing load i.e. (24).

C. Interpretation of Optimal Search Performance Results

We find it very interesting that the optimal search performance does not depend on the underlying search network topology. In fact, it is rather intriguing that the optimal average search time expression for the uniform distribution case seems to be related to the entropy in the file request probabilities $\{\lambda_i/\lambda\}$, and that the only change in the optimal average search time expression in the case of clustered demands is that the entropy now includes the spatial distribution of file requests (λ_{ia}/L is the probability that file i is requested by a node in the high-density cluster and $(1-1/L)\lambda_{ib}$ is the probability that file i is requested by a node in a low-density cluster). Similarly, the optimal query-processing load also changed only in that the expression includes the spatial distribution of file requests in clustered demands case.

Another interesting observation in comparing (25), (26) and (29), (30) to [14, 15] is that while the expressions for the optimal replica distribution are complex in the case of clustered demands, we still have the invariant from the uniform distribution case that the probability of finding the file over a random outgoing link from a node is proportional to the file request rate at that node when optimizing the average search time and is proportional to the square-root of the file request rate at that node when optimizing the query-processing load. We summarize this result in the following theorem.

Theorem 7: For flooding search in the clustered peer-to-peer network defined in Section 3, we have the following invariants independent of the level of clustering in demands and the level of clustering in the search network topology

1) The average search time τ is minimized when $\pi_{ij} \propto \lambda_{ij}$, and

2) The query-processing load Q is minimized when $\pi_{ij} \propto \sqrt{\lambda_{ij}}$

where π_{ij} is the probability of finding file i over a random outgoing link from a node in cluster j and λ_{ij} is the per-node request rate for file i in cluster j. ■

To evaluate the potential benefits of clustering in demands over the uniform distribution case, we plot the interesting part[8] of (22) and (24) in Figs. 4 and 5 respectively for a peer-to-peer network of 100 files with zipf-distributed request rates and 10 equal-sized clusters. Perfect clustering is defined as the case when the entire demand for a file is from its own cluster i.e. λ_{ib}=0 and $\lambda_{ia}= L\lambda_i$. Figs. 4 and 5 clearly demonstrate the potential advantage of clustering. The advantage in search performance afforded by perfect clustering can be summarized in the following theorem.

[8] To eliminate the dependence on d and K, in Figs. 1 and 2, we plot $\tau_{opt}' = -\sum_{i=1}^{N}[\frac{1}{L}\frac{\lambda_{ia}}{\lambda}\ln(\frac{\lambda_{ia}}{\lambda}) + (1-\frac{1}{L})\frac{\lambda_{ib}}{\lambda}\ln(\frac{\lambda_{ib}}{\lambda})]$ and $Q_{opt}' = \left(\sum_{i=1}^{N}[\frac{1}{L}\sqrt{\frac{\lambda_{ia}}{\lambda}} + (1-\frac{1}{L})\sqrt{\frac{\lambda_{ib}}{\lambda}}]\right)^2$ instead of (22) and (24) respectively.

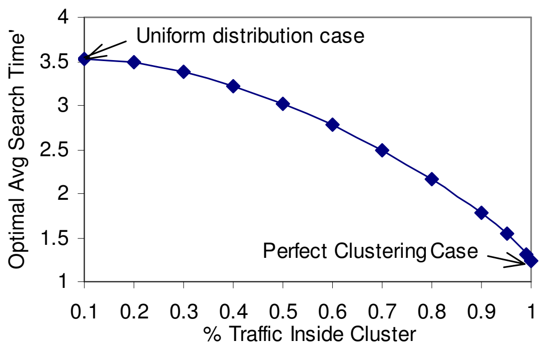


Figure 4. Benefit of Clustered Demands: Optimal Search Time

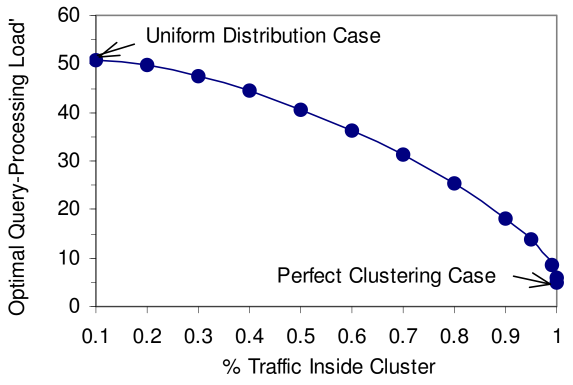


Figure 5. Benefit of Clustered Demands: Optimal Query-Processing Load

Theorem 8: When the entire demand for a file is from its own cluster i.e. λ_{ib}=0 and λ_{ia}= $L\lambda_i$, the optimal average search time τ_F^{opt} decreases by $\log_d L$ and the optimal query-processing load Q_F^{opt} decreases by a factor of L, the number of clusters.

Proof:

Substituting λ_{ib}=0 and λ_{ia}=$L\lambda_i$ in (22) and (24), we get τ_F^{opt} $= -\sum_{i=1}^{N} \frac{\lambda_i}{\lambda} \log_d (\frac{\lambda_i}{\lambda}) - \log_d K - \log_d L$ and $Q_F^{opt} = \frac{1}{L}\left(\sum_{i=1}^{N}\sqrt{\lambda_i}\right)^2 \Big/ \lambda K$.
The theorem follows on comparing these to (3), (5). ■

Finally, we note that the penalty over the optimal query-processing load incurred upon optimizing the average search time increases in the case of clustered demands. For example, for the peer-to-peer network shown in Fig. 5, $Q_F^{\tau opt}$ = 100 independent of the fraction of traffic inside the cluster while Q_F^{opt} ~50 in the uniform distribution case but goes down to ~8.5 when 99% of the file requests are from inside the cluster.

All of the above discussion assumes that the optimal replica distribution can be achieved. In the uniform distribution case, LRU storage management gave near-optimal replica distribution [11] but for the clustered demands case, as we can see in (27) and (28) for the optimal average search time and (31) and (32) for the optimal query-processing load, the desired replica distribution depends on the degree of clustering in the underlying search network topology. However, rather than being a hindrance, this dependence of the optimal replica distribution on the underlying search network topology offers us a very powerful tool to achieve the optimal search performance. In a preliminary study, we were able to achieve the optimal query-processing performance with LRU storage management algorithm by tuning the underlying search network topology. This suggests that it may be possible to achieve the optimal replica distribution with any local storage management algorithm by appropriately tuning the underlying topology. If this approach of tuning the search network topology to reach the optimal replica distribution works in most cases, we may be able to obtain the optimal download performance [12] (by using an LRU-like approach that populates file replicas in near-linear proportionality to the file request rates) and, at the same time, obtain the optimal query-processing load as well by tuning the underlying search network topology appropriately.

VIII. Flooding Search in The General Case

In this section, we extend our results on flooding search performance to peer-to-peer networks that have more complex demand clustering patterns than allowed by our network model of Section 3. Specifically, unlike the network model introduced in Section 3, we will no longer assume the clusters to be equal-sized nor will the demand pattern be restricted to a two-tier demand model (where only one cluster had a different request rate for a file than the rest of the network). We will remove the restriction that the link probabilities be the same from a node to all the nodes in the network that are not in its own cluster (i.e. we will allow a cluster to have more links to one cluster than to another). In our search network topology, each node still has d neighbors on average. We define the following notation for our discussion in this section.

τ_{ik} :	average search time for file k from node i
Q_{ik}:	query-processing load incurred in a search for file k from node i
λ_{ik} :	request rate for file k at node i
q_{ij} :	probability that a random outgoing link from node i goes to node j
π_{ik} :	probability of finding file k over a random outgoing link from node i
p_{jk} :	probability that node j has file k

The network has a number of distinct clusters to which various nodes belong depending on their demand patterns and topology. Thus, if node j belongs to, say, cluster j which has C_j nodes and n_{jk} replicas of file k, then $p_{jk} = n_{jk}/C_{jk}$. As before, we have M nodes and N files in the network. Using these notations, we obtain the total request rate in the network

$$M\lambda = \sum_{k=1}^{N} \sum_{i=1}^{M} \lambda_{ik} \tag{33}$$

Since each node has a total storage capacity of K files, we get the storage capacity constraint

$$\sum_{k=1}^{N} p_{jk} \le K \tag{34}$$

The metrics of interest are

$$\tau = \sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{M\lambda}\tau_{ik} \tag{35}$$

and

$$Q = \sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{M\lambda}Q_{ik} \tag{36}$$

We know from (14), (15) in Section 5 and the results in [10, 11] for the uniform distribution case that $\tau_{ij} = -\log_d(\pi_{ij})$ where π_{ij} is the probability of finding file j over a random outgoing link from node i. Since, in our case $\pi_{ij} = \sum_{j=1}^{M} q_{ij}p_{jk}$ we obtain

$$\tau_{ik} = -\log_d(\sum_{j=1}^{M} q_{ij}p_{jk}) \tag{37}$$

Since our search network still expands as d^{τ} in τ hops we obtain

$$Q_{ik} = \left(\sum_{j=1}^{M} q_{ij}p_{jk}\right)^{-1} \tag{38}$$

Substituting (37) in (35) and (38) in (36), we obtain

$$\tau = -\frac{1}{M}\sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{\lambda}\log_d\left(\sum_{j=1}^{M} q_{ij}p_{jk}\right) \tag{39}$$

$$Q = \frac{1}{M}\sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{\lambda}\left(\sum_{j=1}^{M} q_{ij}p_{jk}\right)^{-1} \tag{40}$$

Applying constraint (34), we get the Lagrangian for minimizing the average search distance as

$$H = -\sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{M\lambda}\log_d\left(\sum_{j=1}^{M} q_{ij}p_{jk}\right) + \gamma\left(\sum_{k=1}^{N} p_{jk} - K\right)$$

Differentiating w.r.t. p_{jk}, we get the set of following L equations $\forall k$

$$-\frac{1}{M\lambda\ln d}\sum_{i=1}^{M}\frac{\lambda_{ik}q_{ij}}{\sum_{j=1}^{M} q_{ij}p_{jk}} + \gamma = 0 \quad \forall i$$

One can easily verify that the above set of equations will be satisfied by

$$\sum_{j=1}^{M} q_{ij}p_{jk} = \beta\lambda_{ik} \quad \forall i, \forall k \tag{41}$$

where β is a constant whose value we determine next.

Since populating more files always improves search performance, all peer caches will be full at the optima i.e. the inequality in (34) will be an equality (see [11] for further discussion on this). Therefore, we obtain

$$MK = \sum_{j=1}^{M}\sum_{k=1}^{N} p_{jk} \tag{42}$$

From (41), we obtain

$$\sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{j=1}^{M} q_{ij}p_{jk} = \beta\sum_{k=1}^{N}\sum_{i=1}^{M}\lambda_{ik} = \beta M\lambda$$

From (33), the r.h.s equals $\beta M\lambda$. Rearranging the order of summation on the l.h.s., we have

$$\sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{j=1}^{M} q_{ij}p_{jk} = \sum_{k=1}^{N}\sum_{j=1}^{M}(\sum_{i=1}^{M} q_{ij})p_{jk} = \sum_{k=1}^{N}\sum_{j=1}^{M} p_{jk} = \sum_{j=1}^{M}\sum_{k=1}^{N} p_{jk} = MK$$

with the last step following from (42). Therefore, $\beta = K/\lambda$.

Substituting the value of β in (41) and substituting (41) in (39), we get

$$\tau_F^{opt} = -\frac{1}{M}\sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{\lambda}\log_d\frac{\lambda_{ik}}{\lambda} - \log_d K \tag{43}$$

Further, with $\beta = K/\lambda$ in (41), we can substitute (41) in (40) to obtain

$$Q_F^{\tau opt} = \frac{N}{K} \tag{44}$$

The following theorem summarizes these results.

Theorem 9: The average search time for flooding search is minimized when the probability of finding file i over a random outgoing link from a node j, π_{ij}, is proportional to λ_{ij}, the request rate for file i at node j, i.e.

$$\pi_{ij} \propto \lambda_{ij}$$

and the minimum average search time is

$$\tau_F^{opt} = -\frac{1}{M}\sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{\lambda}\log_d\frac{\lambda_{ik}}{\lambda} - \log_d K$$

and the query-processing load when the average search time is minimized is

$$Q_F^{\tau opt} = \frac{N}{K}$$

i.e. the optimal average search time is independent of any clustering in the search network topology and the query-processing load when the average search time is minimized is independent of the skew in file popularity and the level of clustering in both the search network and the file popularity. ■

Instead of minimizing the average search time, let us now minimize the query-processing load. When we minimize the query-processing load with constraint (34), the Lagrangian is

$$H = \sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{M\lambda}\left(\sum_{j=1}^{M} q_{ij}p_{jk}\right)^{-1} + \gamma\left(\sum_{k=1}^{N} p_{jk} - K\right)$$

Differentiating w.r.t. p_{jk}, we get the set of following L equations $\forall k$

$$-\frac{1}{M\lambda}\sum_{i=1}^{M}\frac{\lambda_{ik}q_{ij}}{\left(\sum_{j=1}^{M} q_{ij}p_{jk}\right)^2} + \gamma = 0 \quad \forall i$$

One can easily verify that the above set of equations will be satisfied by

$$\sum_{j=1}^{M} q_{ij} p_{jk} = \beta' \sqrt{\lambda_{ik}} \quad \forall i, \forall k \tag{44}$$

where β' is a constant whose value we determine next. As before, we have

$$MK = \sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{j=1}^{M} q_{ij} p_{jk} = \beta' \sum_{k=1}^{N}\sum_{i=1}^{M}\sqrt{\lambda_{ik}}$$

Substituting the value of β' from the above equation in (44) and substituting the result in (40), we get

$$Q_F^{opt} = \frac{1}{K}\left(\sum_{k=1}^{N}\sum_{j=1}^{M}\frac{1}{M}\sqrt{\frac{\lambda_{ik}}{\lambda}}\right)^2 \tag{45}$$

The following theorem summarizes these results.

Theorem 10: The query-processing load for flooding search is minimized when the probability of finding file *i* over a random outgoing link from a node *j*, π_{ij}, is proportional to the square-root of λ_{ij}, the request rate for file *i* at node *j*, i.e.

$$\pi_{ij} \propto \sqrt{\lambda_{ij}}$$

and the minimum query-processing load is

$$Q_F^{opt} = \frac{1}{K}\left(\sum_{k=1}^{N}\sum_{j=1}^{M}\frac{1}{M}\sqrt{\frac{\lambda_{ik}}{\lambda}}\right)^2$$

i.e. the optimal query-processing load is independent of any clustering in search network topology. The average search time at the optimal query-processing load is

$$\tau_F^{Qopt} = -\frac{1}{2M}\sum_{k=1}^{N}\sum_{i=1}^{M}\frac{\lambda_{ik}}{\lambda}\log_d\frac{\lambda_{ik}}{\lambda} - \log_d K + \log_d\left(\sum_{k=1}^{N}\sum_{j=1}^{M}\frac{1}{M}\sqrt{\frac{\lambda_{ik}}{\lambda}}\right) \tag{39}$$

■

Thus, we find that the invariants noted in Theorem 7 hold in the general demand and topology clustering case as well. Further, the optimal average search time is still related to the entropy in file request probabilities at each node and the optimal query-processing load is still related to the square of the sum of the square-roots of the file request probabilities at each node times the probability of that node being selected to make the request. We discuss these results in next section.

IX. Entropy and $N_{EFFECTIVE}$

Comparing (3), (22) and (43), we notice that the minimum average search time in a flooding search depends on the entropy in file request probabilities at each node and the per-node storage size. Specifically, the entropy on which the minimum average search time seems to depend is the entropy in which a file is requested across the entire network. In other words, if the entire network is considered an information source, the entropy term we see refers to the uncertainty regarding which node will make the next request and which file it will request). If we (an external observer) knew which file will be requested by whom next, we (the external observer) can place the file at exactly the right location and the search time will be 0. Our lack of information prevents us from placing the files in such a manner. However, suppose we knew that a particular group of nodes requests only a certain set of files, then we would place those files in or near that group of nodes thereby reducing the search time for those files (and, hence, the average search time) substantially. Thus, it is understandable that the minimum average search time is related to the entropy (the uncertainty) as to which node will request which file next. We also note that an increase in the per-node storage size causes a decrease in the search time. Recall that in our just concluded discussion on the relation of entropy, we said that we can "place" the file(s) that are likely to be requested near the nodes likely to request that file. We can place a file only if storage space is available. In the extreme case where the per-node storage size is large enough to place all the files at all the nodes, the search time would be 0. In general, that is unlikely to be the case but, clearly, if more storage space is available, more of the files likely to be requested, can be placed near the peers that will request them. If the available storage space is small but there is little uncertainty regarding who will request which files, an effective placement can be made (e.g. if each node requests only two files and the per-node storage capacity is two, the search time will be 0). In contrast, if the storage space is not too large and we have very little information regarding who will request what, the placement will not be able to reduce the average search time as much (e.g. if the per-node storage capacity is, say, 5 files but a node can request any of the 1000 files, then file placement will not help much).

Recall that the search time for a file is defined as the number of hops taken to find the file. As noted in [10], since flooding expands to all possible paths with every additional step, the search time for a file from a node is the shortest distance to a replica of the file from that node. From the perspective of a querying node, flooding is a breadth-first search over a tree of newly queried nodes at each hop. If one were to specify the path from the querying node in this tree (i.e. the sequence of branches to be taken at each step) to the nearest replica of the queried file, the path length will be equal to shortest hop distance to the nearest replica of the file. With this analogy of flooding tree and path length to the nearest replica, we can readily see similarities to many other seemingly unrelated problems in computer science where the entropy expression appears.

Optimal coding is the best known of these problems. Recall from [16] that the optimum codeword length for a given symbol is the self-information of that symbol. The classical representation of coding is a coding tree where symbols are the leaf nodes in the tree and the code for a symbol is the path from the root to the symbol on the coding tree. Thus, the average path length in the coding tree is the average code word length (just as it is for the average search time in our case). There are differences, however, e.g. the symbols are leaf nodes in a coding tree and each symbol appears only once in the coding tree while, in our case, each file can be stored at multiple nodes and the files can be stored at non-leaf nodes.

Some other places where the entropy expression shows up are: *(i)* the average number of steps to find a key (i.e. the average depth to which one has to go to find a key in the search tree) in the optimal binary balanced search tree [17], and *(ii)* as the lower bound on the minimum number of comparisons needed to sort a set of keys (where multiple items in the set may have the same value) [18] where the entropy is on the frequency of occurrence of different keys in the entire set. To the best of our knowledge, no precise discussion of the relevance of entropy to these problems exists in the literature.

Let us now investigate the optimal query-processing load expressions. Comparing (5), (24) and (45), we notice that the minimum query-processing load depends on the square of the sum of the square roots of the file request probabilities at each node and the per-node storage size. As discussed for the optimal average search time, we know that a larger cache size allows more files to be placed near the nodes requesting them which decreases the search time and, consequently, the query-processing load as well. A larger skew in file request probabilities implies we have better information on who will request which file and so the placement of files will be more effective; hence, the search time and the query-processing load will be smaller. The square of the sum of the square roots of the file request probabilities at each node is another way (like entropy) of representing the skew in file request probabilities. The advantage of a higher skew in file request probabilities and the advantage of a larger cache size is reflected in the expression for Q_{opt}.

Notice that the worst-case file request probability distribution for query-processing load, Q_{opt}, is when each node has the same request rate for each file. Q_{opt} in this case is N/K where N is the number of files in the network and K is the number of files cache size in. In other words

$$Q_{opt} = \frac{1}{K}\left(\sum_{k=1}^{N}\sum_{j=1}^{M}\frac{1}{M}\sqrt{\frac{\lambda_{ik}}{\lambda}}\right)^2 \le \frac{N}{K}$$

Thus, we may interpret the square of the sum of the square roots of the file request probabilities at each node as the *effective number* of equal request rate files in the network (recall that a larger skew provides us more information allowing us to make a more effective placement of files *as if* there were fewer equal request rate files in the network). Therefore, we can write

$$N_{effective} = \left(\sum_{k=1}^{N}\sum_{j=1}^{M}\frac{1}{M}\sqrt{\frac{\lambda_{ik}}{\lambda}}\right)^2$$

We note that this square of the sum of the square-roots expression has appeared earlier in [14, 15]. [15] introduced the novel approach that the benefit afforded by the skew in the source rates of each participant is represented by a (reduced) effective number of participants given by this square of the sum of the square roots expression. This approach allowed [15] to reduce a given number of skewed sources to a smaller number $N_{effective}$ of sources each of which had the same load. The power of this idea of having only $N_{effective}$ files (or users) is that we can now eliminate a dimension of variability (the file request rate distribution) from the problem and focus on the remaining parameters in the simulation/analysis. Reference [14] also found the square of the sum of the square roots expression in the optimum message delay expression. While [14] did not offer a notion of $N_{effective}$ at the time, we can see that in that problem (of allocating a fixed given capacity to different channels) also, the skew in traffic on each channel afforded improvement in the overall average delay[9] and the square of the sum of the squares expression is the equivalent number, $N_{effective}$, of equal load channels.

In summary, while entropy has a physical meaning (uncertainty), the square of the sum of square-roots expression is the effective number of distinct equal rate resource consumers in the system.

X. CONCLUSION

In this paper, we investigated the relationship between the number of replicas of a file in unstructured peer-to-peer networks and derived the search time and the search cost 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 exact expression for the random walk search time (and, hence, the search cost) 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. We also found that, among these bounds, one side was a reasonable approximation for the flooding search time in the clustered demands case. Using these approximate expressions, we derive expressions for the search cost, the optimal search cost and the optimal search time in an unstructured peer-to-peer network when the demand exhibits clustering. The previous work in this area assumed uniformity in replica and demand distribution. Since real networks show clustering in demands, our results provide a more accurate estimate of the search performance achievable in unstructured peer-to-peer networks. Interestingly, we found that the gains in the optimal search performance afforded by clustering in demand patterns are independent of whether the search network topology matches the clustering in file popularity. The optimal replica distribution, however, does depend on

[9] The best case scenario for that problem was to have all the traffic on one channel and allocate all the capacity to that channel. In that case, the higher capacity provides a lower service time while the utilization factor ρ remains the same. By the same argument, the worst-case scenario is for each channel to have equal load and, thus, equal capacity.

clustering in the search network topology. Since the replica distribution is driven by peer requests, we believe that tuning the search network topology to match the replica distribution generated by peer requests is more practical than matching the replica distribution to the topology. In our simulations, we were able to operate a peer-to-peer network at the optimal search cost by tuning the clustering in the search network topology depending the clustering in demands while using LRU cache management at each peer. In the process of deriving the optimal search performance results, we derived the relation between the query-processing load and the number of replicas of each file for the clustered demands case showing that flooding searches may have a lower query-processing load than random walk searches in the clustered demands case.

References

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

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

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

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

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

[6] Liben-Nowell, D., Balakrishnan, H, Karger, D., "Analysis of the evolution of peer-to-peer systems," in Proc. of PODC, July 2002.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] Loo, B. T., Huebsch, R., Stoica, I., and Hellerstein, J. "The case for a hybrid P2P search infrastructure," in Proc. of IPTPS, 2004.

[14] Kleinrock, L., Communication Nets; Stochastic Message Flow and Delay, McGraw-Hill Book Company, New York, 1964.

[15] Kleinrock, L., "Performance of Distributed Multi-Access Computer-Communication Systems", in Proc. of IFIP Congress 77, August 1977.

[16] Weaver, W., Shannon C. E., The Mathematical Theory of Communication, Urbana, Illinois: University of Illinois Press, 1949.

[17] Knuth, D. E., The Art of Computer Programming, Vol. 3: Sorting and Searching, Reading, Massachusetts: Addison-Wesley, 1998.

[18] McIlroy, P., "Optimistic sorting and information theoretic complexity," in Proc. of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, 1993.