Optimal Search Performance in Unstructured Peer-to-Peer Networks With Clustered Demands

Saurabh Tewari, *Student Member, IEEE*, and Leonard Kleinrock, *Fellow, IEEE*

***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 across the set of nodes in the network). Clustering in file popularity patterns is evident from measurements on deployed peer-to-peer file sharing networks. In this paper, we provide mechanisms for modeling clustering in file popularity distributions and the consequent non-uniform distribution of file replicas. We derive relations that show the effect of the number of replicas of a file on the search time and on the search cost for a search for that file for the clustered demands case in such networks for both random walk and flooding search mechanisms. The derived relations are used to obtain the optimal search performance for the case of flooding search mechanisms. The potential performance benefit that 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 (the optimal file replica distribution to obtain these performance gains, however, does depend on the search network topology).**

***Index Terms*— Clustered demands, flooding, optimal search cost, optimal search time, peer-to-peer networks, random walk.**

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 which allow the neighbors of a peer to be chosen randomly (e.g. Gnutella [9]) are referred to as *unstructured* peer-to-peer networks to distinguish them from structured networks (e.g. Chord [27]) where peers are assigned unique node IDs and a peer's set of neighbors is derived from its node ID. These structured networks further map each unique resource to a particular node in the network which allows the searches to be more efficient but their lack of flexibility introduces other issues [16]. 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[1] and we refer to resources as files throughout the rest of the paper. 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 this *average search time* 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 peers queried equals the additional traffic introduced in the network by the query. Therefore, we approximate the search cost by the *query-processing load*, i.e., the average number of query messages 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. 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 [29] under the assumption of a uniform distribution of the file replicas. However, measurements on deployed peer-to-peer file sharing networks show clustering in interests [10], [13], [14] (i.e., the popularity of a set of files varies across the set of nodes) and more replicas of a file are found in the regions where the file is more popular [14]. In this paper, we provide results analogous to those given in [29] for the clustered demands case.

Thus, the problem addressed in this paper is: *given a particular file popularity distribution, what is the optimum performance that can be achieved?* File popularity distribution is driven by the user behavior but a system designer can choose the type of overlay topology that may be constructed, the search mechanisms on this search overlay and the placement and the number of replicas of each file in this network. To limit the scope of the problem we assume that a search for a file

Manuscript received December 15, 2005; revised July 1, 2006. This paper was presented in part at Globecom, St Louis, MO, 2005 and at ICC, Istanbul, Turkey, 2006.

The authors are with the UCLA Computer Science Department, Los Angeles, CA 90095 USA (e-mail: stewari@cs.ucla.edu; lk@cs.ucla.edu).

Digital Object Identifier 10.1109/JSAC.2007.0701xx.

[1]We would like to emphasize that the problem of search has applications beyond the currently deployed peer-to-peer file sharing networks over the Internet and even though we use the term "files," our results do not depend on any modeling assumption specific to file sharing networks.

has no apriori information on which node may have the file,[2] and consider the two extremities among the possible search mechanisms, *random walk* and *flooding*. Our overlay topology model allows for a tunable level of clustering in the topology and we consider the level of clustering in the topology to be an important design parameter. Our model allows control over the placement and the number of replicas of each file by allowing independent control over the number of replicas of a file in each topology cluster. In addition to providing analytic expressions for the optimum performance possible, we also provide analytic expressions for the optimum system design in terms of the level of clustering in the topology and the number of replicas of each file in each cluster in the overlay topology.

Our analysis assumes that the search overlay and the distribution of file replicas on this overlay does not change during a search (i.e. from the time a particular node initiates a search till that search concludes). Over time, both the topology and the distribution of file replicas (as well as the file popularity distribution) can change as peers may join or leave the network, obtain replicas of files and, possibly, delete local replicas of other files. However, as long as the overall statistical properties of the network (i.e. the overlay topology and the replica distribution) are maintained, our results hold. Over a longer term, even the file popularities can change and new files may be introduced. Therefore, an ideal algorithm will dynamically discover the clustering in file popularity distributions and adjust overlay topology in response. We do not provide such algorithms but will briefly mention some of the proposed algorithms that operate along these lines when we discuss the related work in Section II. While these algorithms have been shown to improve performance over overlay topologies that do not exhibit clustering, our characterization of the nature of the optimal solution will allow researchers to further tune their algorithms so as to achieve the optimal performance.

The main contributions of this paper are given in Sections III–VIII. In Section III, 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 IV while Section V derives the analogous results for the flooding search. In Section VI, we derive the query-processing load expressions for random walk and flooding searches. The optimal search time and the optimal search cost expressions for flooding search are derived in Section VII. In Section VIII we extend the results for flooding search beyond the specific mode of demand clustering allowed by our network model described in Section III to incorporate any arbitrary demand clustering. Related work is discussed in Section II and our conclusions are given in Section IX.

II. Background and Related Work

Flooding and *random walking* are the two extremes among the generalized search strategies [8] 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.[3] 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 until a copy is found.

Unstructured peer-to-peer networks are typically modeled using random graphs (since the neighbors of a peer are randomly chosen). We provide simulation results in [29] for the search time as a function of the number of replicas of a file for a measured Gnutella2 [9] trace topology and a superpeer topology (each superpeer has many *leaf* peers connecting to it and the superpeers themselves are connected in an Erdos-Renyi random graph [1] topology) constructed using similar parameters as the measured trace which indicate that Erdos-Renyi random graphs are a good choice[4] when nodes are similar in capacities and file interests (i.e. when files and file popularities are uniformly distributed). Given that file popularities exhibit clustering in most real-world situations [10], [13], [14], tuning the underlying overlay topology to reflect the clustering (e.g. having a search overlay topology matching the data-sharing graphs of [10]) can improve the system performance. Our results in Section VIII quantify the maximum possible improvement but we have not provided mechanisms to achieve this performance in a dynamic setting. Discovering nodes with similar interests (e.g. by gossiping or based on past query responses) followed by structuring the overlay so as to cluster nodes with similar interests together (e.g. using interest-based shortcuts or by replacing some of the existing links with links to nodes with similar interests) [4], [11], [17], [21], [26] are promising approaches in this direction. Our overlay topology model of Section III can be viewed as an Erdos-Renyi random graph with link probability q to which interest-based shortcuts are added so as to construct the clusters of nodes with intra-cluster link probability p (we expect $p > q$). Some alternative approaches are [2] and [20], [25] which use a centralized server and the superpeers in a superpeer-based network, respectively, to find nodes with similar interests.

In addition to the search mechanisms suggested in aforementioned proposals, hybrid search mechanisms that use edge criticality (as discussed in [8]) can also provide efficient searches in these topologies. Search methods that direct query propagation based on past query responses (e.g. [32]) also leverage clustering in file popularities even if their overlays do not have explicit clustering. The proposal in [6] uses guide-rules to guide searches to nodes with similar interests which they note as similar to having separate unstructured overlays for different communities of interest. Finally, even though the

[2]Caching query results at nodes other than the requesting node can reduce the query-processing load but keeping these pointers accurate beyond one-hop from the originial source can be a challenge in the highly dynamic environment [13] in typical peer-to-peer networks.

[3]For example, one way to terminate the flooding search as soon as the file is found is to sequentially increase the maximum allowed depth of the search query one hop at a time.

[4]When node capacities are very skewed, a power-law random graph is a topology choice which distributes the query-processing load unevenly amoung the peers but yields faster search methods [3]. These topologies are suited to the case where the node capacities can take many different values but if they can take approximately only two values (e.g. 300 kbps and 10 Mbps), superpeer topologies are more appropriate to obtain faster searches.

proposal in [18] does not refer to clustering, their approach can work well when file popularities exhibit clustering since limited-hop floods would discover the files in the local cluster and rarely-requested files (e.g. those in the non-local clusters) would be found via a DHT-based mechanism (i.e. over a structured network). We note that there are many other approaches in leveraging clustering in file popularities (such as the work on semantic overlays [28]) that we cannot mention due to space limitations. We also emphasize that while our topology model has small-world features, our search problem does not map to finding a particular node whose "address" was known, this having been dealt with in the seminal work on small-world networks in [12]. Approaches such [15], [19] have built upon results in [12] to construct networks that use the DHT abstraction for searches. The DHT abstraction provides one-to-one mapping from files to nodes in the network and, hence, finding a file translates to finding the node responsible for that file. In contrast, our work assumes that multiple peers may have the desired file (as has been observed for popular files in deployed systems [14], [18]) and, hence, our search is not for a particular peer (that has the file) but for any one of the peers that have the desired file. In fact, one of our key contributions is the analytical relation that we establish between the search performance and the number of replicas of the desired file. While [18] provides expressions for the search performance as a function of the number of replicas, their performance metrics were different. Like other hop-limited flooding (and random walk) search proposals (e.g. [26]), they are interested in the probability of success in finding matching files and the number of distinct file sources found within the maximum number of hops allowed. There are other possible metrics (e.g. the number of distinct nodes found in a given limit on the number of messages allowed [8]) as well but the search time and the search cost that we find are more direct measures of the performance. References [5], [29] discuss the relation between the search performance in terms of these metrics and the number of replicas of the desired file under the assumption of a uniform distribution of file replicas (which assumes that all peers have similar interests) and the optimal search performance under the constraint of finite per-node storage. Since our work in this paper is similar to that in [5], [29] (except that we remove the assumption of a uniform distribution of replicas), we briefly describe the key results from [5], [29] next.

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 and, hence, the (average) search time for file i, τ_{iR}, and the (average) number of messages[5] per search for file i, Q_{iR}, is M/n_i. Reference [29] provides analogous results for flooding 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 and 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. Since the number of nodes queried in a hop distance of τ is d^τ, the (average) number of nodes queried per flooding search for file i, Q_{iF}, is still M/n_i. Thus,

$$\tau_{iF}(n_i) = \log_d(M/n_i) \tag{1}$$

$$Q_{iF}(n_i) = Q_{iR}(n_i) = \tau_{iR}(n_i) = M/n_i \tag{2}$$

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 $\log_d(M/n_i)$ hops because it queries exponentially more nodes with each additional hop.[6] If there are N unique files in the system, the per-node request rate for file i is λ_i, and each node can store K (equal-sized) files, then, as we show in [29], the search time for flooding averaged over all requests is minimized when $n_i \propto \lambda_i$ and the optimal average search time, τ_F^{opt}, and the query-processing load at this distribution, $Q_F^{\tau opt}$, are given by

$$\tau_F^{opt} = -\sum_{i=1}^{N} \frac{\lambda_i}{\lambda} \log_d \frac{\lambda_i}{\lambda} - \log_d K \tag{3}$$

$$Q_F^{\tau opt} = N/K \tag{4}$$

where $\lambda = \sum_{i=1}^{N} \lambda_i$. The optimal average search time for random walk, τ_R^{opt}, and the optimal query processing load for

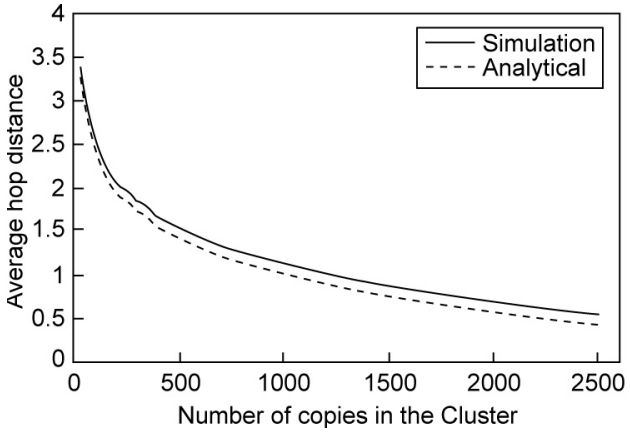


Fig. 1. Search time with uniform distribution (5,000 node network, average degree 5).

[5]Note that this includes the possibility of already queried nodes being queried again (if these nodes could be eliminated from subsequent probes, the probability of finding the file with each new probe would increase as well).

[6]Since a node does not forward a query twice, the exponential growth assumption is optimistic (see [8] for growth patterns in Erdos-Renyi random graphs). Thus, (1) slightly underestimates the actual search time as shown in Fig. 1. In [29], we provide more 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 [29] 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.

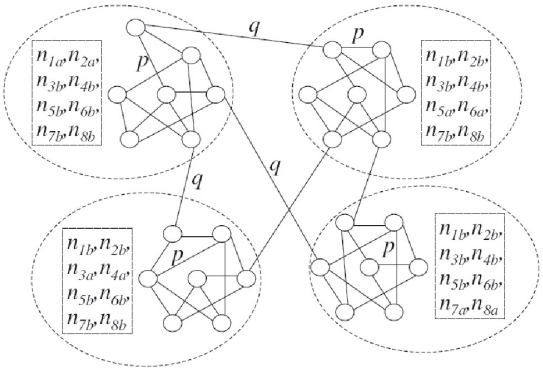


Fig. 2. Two-level clustering example - $M = 28$ nodes in $L = 4$ equal-size clusters; total $N = 8$ files, files 1, 2 are more popular in cluster A and have equal lesser popularity in clusters B, C, D and there are n_{1a}, n_{2a} replicas of files 1 and 2 respectively in cluster A and n_{1b}, n_{2b} replicas of files 1 and 2 respectively in each of the clusters B, C, D; $n_{ia} > n_{ib}$ and, for searches for file 1 and 2, cluster A is the high-replica-density cluster and clusters B, C, D are the low-replica density clusters (as shown in n_{ia}, n_{ib} labeling, files 3, 4 are more popular in cluster B, files 5, 6 more popular in cluster C, files 7, 8 are more popular in cluster D); topology model: average degree = 3.5, fraction of links within the cluster = 0.9 (Probability that any given pair of intra-cluster nodes are connected $= p \sim 0.449$, probability that any given pair of inter-cluster nodes are connected $= q \sim 0.017$).

flooding and random walk, Q_F^{opt} and Q_R^{opt} respectively, are given by

$$Q_F^{opt} = Q_R^{opt} = \tau_R^{opt} = \frac{\left(\sum_{i=1}^{N} \sqrt{\lambda_i}\right)^2}{\lambda K} \tag{5}$$

achieved when $n_i \propto \checkmark\lambda_i$. In this paper, we seek to obtain results analogous to (1)–(5) when the file replica distribution and the demand patterns are not uniform.

We note that [29] and [5] respectively provide distributed algorithms to achieve the $n_i \propto \lambda_i$ and the $n_i \propto \checkmark\lambda_i$ replica distributions without any need to measure individual file request rates λ_i, the total request rate λ, the per-node storage size K or the total number of nodes M even though the proportionality constants for both distributions include these terms. We believe[7] that using techniques similar to those in [5], [29] along with topology clustering techniques similar to those in [4], [11], [17], [21], [26] will allow us to achieve the optimal replica distributions given by the expressions we provide in Sections VII and VIII without a need for explicitly measuring individual file request rates, the per-node storage size or the total number of nodes.[8]

[7]Even though our expressions suggest that optimal replica distributions exist for topologies with arbitrary clustering, allowing tuning of clustering in the topology allows for more robust distributed replication algorithms. The expressions for optimal replica distributions provided by Theorems 7 and 8 in Section VIII are in terms of link probabilities and the probability of finding the file at different nodes-the former is addressed by [4], [11], [17], [21], [26] and the latter by [5], [29] (the expressions in Section VII are a special case of those in Section VIII).

[8]We note that, if necessary, it is possible to estimate file request rates as well as the number of nodes (see [23] and [19] respectively for examples of possible approaches).

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.[9] 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 replica density* of file i replicas whereas a cluster where the file is not as popular has a *lower replica 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.

Since each link is equally likely in the topology generated by an Erdos-Renyi random graph, we need to define an alternate topology that allows for clustering. A topology model that gives us a continuum of topologies with the Erdos-Renyi random graph at one extreme and the fully disconnected clusters (i.e. no inter-cluster links) at the other extreme is the following random graph variant. Assuming L clusters of equal sizes, the nodes are partitioned into L clusters, the probability that any given pair of intra-cluster nodes is connected is p, the probability that any given pair of inter-cluster nodes is connected is q, and the average per-node degree is d (see Fig. 2 for an example of such a topology model). 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 $(p = q)$. Notice that a flooding search in these topologies still expands to d other nodes in the next hop independent of whether the search process is at a node in the high-replica-density cluster or a low-replica-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 [29] still holds.

We note that, depending on the level of clustering, our topology model can be considered as a small-world network. The intra-cluster link probability p is equivalent to the clustering coefficient [10] in the small-world topologies.[10] If the network size (i.e. M) increases and the average per-node degree d remains the same, p, the clustering coefficient, would have to decrease unless the number of clusters also increases so

[9]Section VIII provides the results without these assumptions.

[10]There are $0.5p(M/L)(M/L-1)$ links among the nodes in a cluster out of the $0.5(M/L)(M/L-1)$ maximum possible links.

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 havin 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-replica-densi cluster
n_{ib}	Number of replicas of file i in a "low-replica-densi 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-replica-densi cluster
λ_{ib}	Request rate of file i per node in a "low-replica-densi 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-replica-density cluster with search method x [a]
Q_{ixa}	Query-processing load for file i from the high-replica-density cluster with search method x [a]
τ_{ixb}	Average search time for file i from a low-replica-density cluster with search method x [a]
Q_{ixb}	Query-processing load for file i from a low-replica-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-replica-density cluster with flooding search.

it is not clear if our topology has "heavy" clustering [24]. However, even if p is small, topologies where $p >> q$ are clearly more clustered than an Erdos-Renyi random graph (see Fig. 2). A more intuitive metric for the level of clustering in topology such as ours is the *fraction of links within the cluster*, $(M/L)(p/d)$; if there are more links from a node to other nodes in the same cluster, the topology clearly has clustering. We also note that unless the replica distribution is also "clustered" (i.e. different number of replicas in different clusters), clustering in topology does not change the search performance averages (see results in Sections IV and V). Since the relative replica density (the probability of finding the file) in the different clusters directly affects the search performance, we define the ratio of replica densities in the high-replica-density cluster and the low-replica-density cluster, $\sum_i (n_{ia}/n_{ib})$, as the measure for the level of clustering in the replica distribution and name it the *replica density skew*. In summary, we say a peer-to-peer network has "heavy" network clustering when the fraction of links within the cluster is large and, it has "heavy" demand clustering when the replica density skew is large (as the demand pattern drives the replica distribution). A peer-to-peer network has strong clustering

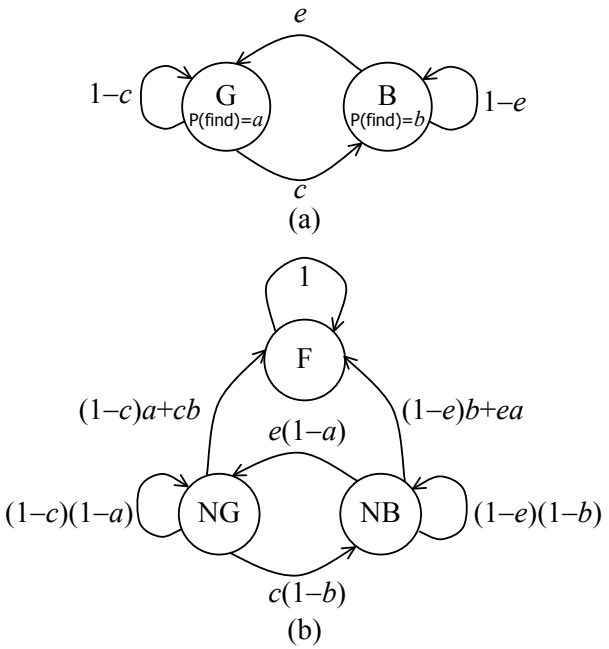


Fig. 3. Random walk in the modified random graph for the non-uniform file distribution case.

when both the demand clustering and the network clustering are large, or one is extremely large and the other is only moderately large.

Table 1 summarizes the notation used in this paper.

IV. RANDOM WALK SEARCH IN NETWORKS WITH CLUSTERING

The existence of inter-cluster links implies that a query can get forwarded to nodes in different clusters 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-replica-density cluster or to a node in one of the low-replica-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 high-replica-density cluster, the probability of one query path "escaping" to a low-replica-density cluster is $c = q(M - M/L)/d$. In contrast, when the query is at a node in the low-replica-density cluster, the probability of escaping to the high-replica-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 high-replica-density cluster as "good" nodes, and the nodes in the lower-replica-density clusters as "bad" nodes.

Fig. 3(a) 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. 3(a) to that in Fig. 3(b). 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}. We solve for these average first passage times using the results from [7] and, since, for the random walk, the search time is also the query-processing load, we obtain the following theorem.

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

$$\begin{aligned}\tau_{iRa}\left(n_{ia}, n_{ib}\right) &= Q_{iRa}\left(n_{ia}, n_{ib}\right) \\ &= \left[\frac{n_{ia}L}{M} - \frac{q\left(L-1\right)\left(n_{ia}-n_{ib}\right)}{\left(n_{ib}L/M\right)\left(d-Mq\right)+Mq}\right]^{-1}\end{aligned} \tag{6}$$

if the search is initiated at a node in the low-replica-density cluster, and is

$$\begin{aligned}\tau_{iRb}\left(n_{ia}, n_{ib}\right) &= Q_{iRb}\left(n_{ia}, n_{ib}\right) \\ &= \left[\frac{n_{ib}L}{M} + \frac{q\left(n_{ia}-n_{ib}\right)}{\left(n_{ia}L/M\right)\left(d-Mq\right)+Mq}\right]^{-1}\end{aligned} \tag{7}$$

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

We notice from (6) and (7) that the search time for a query initiated by a good node increases if cross-cluster links are present (i.e. $q \neq 0$) but if a bad node initiated the query, the search time decreases. As expected, in the uniform distribution case ($n_{ia} = n_{ib} = n_i/L$), (6) and (7) revert to (2).

V. Flooding Search in Networks With Clustering

Unlike the case of no clustering where we found in Section II 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 i.e. the likelihood of reaching a good node on the next hop is higher from a good node than it is from a bad node i.e. $P(G|G) > P(G|B)$ or $1-c > e$. Similarly, the likelihood of reaching a bad node on the next hop is higher from a bad node than it is 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,[11] 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.[12] Thus, a flooding search initiated by a good node is likely to query more good nodes in $log_d x$ steps than a random walk search would in x steps starting at the same node and a flooding search initiated by a bad node will query more bad nodes in $log_d x$ steps than a random walk search will in x steps starting at the same node. Hence,

$$\tau_{iFa}\left(n_{ia}, n_{ib}\right) \leq \log_d\left[\tau_{iRa}\left(n_{ia}, n_{ib}\right)\right] \tag{8}$$

$$\tau_{iFb}\left(n_{ia}, n_{ib}\right) \geq \log_d\left[\tau_{iRb}\left(n_{ia}, n_{ib}\right)\right] \tag{9}$$

Thus, in networks with clustering, the random walk search times only provide us with bounds[13] 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 (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). The best we can do is to bound the search time on the other side as well. For the flooding search time for a search initiated at a good node, one approach to obtain a lower bound 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 $P(F|NG)$. In its derivation of (1), [29] shows that the search time to find file i, when the probability of finding the file at a node on the next hop is P, is $-\log_d P$. Hence, the search time for a flooding search from a good node is no better than $-\log_d\left[P\left(F|NG\right)\right] = \log_d\left[\left(1-c\right)a + cb\right] = -\log_d\left[a - c(a-b)\right]$. Thus,[14]

$$\tau_{iFa}\left(n_{ia}, n_{ib}\right) \geq \log_d\left[\frac{n_{ia}L}{M} - \frac{q\left(L-1\right)\left(n_{ia}-n_{ib}\right)}{d}\right] \tag{10}$$

The upper bound for τ_{iFb}, the search time for a flooding search initiated at a bad node can be similarly found by ignoring all the "good" possibilities and assuming that even after hop distance > 1, the nodes forwarding the queries are all bad 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 $P(F|NB)$. Hence, the search time for a flooding search from a bad node is no

[11]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. It is easy to see that the average number of good nodes with flooding, $3(1-c)^3 + 2[3(1-c)^2c] + [3c^2(1-c)$, is greater than the number with random walk, $3(1-c)^3 + 2[2ce(1-c) + c(1-c)^2] + [(1-c)c(1-e)e + c^2e]$, when $1-c > e$.

[12]We can see this by similar arguments as in Footnote 11.

[13]The bounds presented in this section are approximate bounds as the underlying analytical approach (Section II) 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.

[14]Since the probability of finding the file at hop distance 0 is $P(F)$ whereas the expression $-\log_d\left[P\left(F|NG\right)\right]$ assumes $P\left(F|NG\right)$ to be the probability at all hop distances including 0, a correction factor of $-\left[1-P\left(F\right)\right]/\left[1-P\left(F|NG\right)\right]$ 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 (10). 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 (11) as well.

worse than $-\log_d[P(F|NB)] = -\log_d[(1-e)b+ea] = -\log_d[b+e(a-b)]$. Thus,

$$\tau_{iFb}(n_{ia},n_{ib}) \le \log_d\left[\frac{n_{ib}L}{M}+\frac{q(n_{ia}-n_{ib})}{d}\right] \quad (11)$$

Combining (8), (9), (10), and (11), we get the following theorem:

Theorem 2: The search time for a flooding search in the clustered peer-to-peer network defined in Section III is *approximately*[13,14] bounded by

$$\begin{aligned}&-\log_d\left[\frac{n_{ia}L}{M}-\frac{q(L-1)(n_{ia}-n_{ib})}{d}\right]\\ &\le \tau_{iFa}(n_{ia},n_{ib})\\ &\le -\log_d\left[\frac{n_{ia}L}{M}-\frac{q(L-1)(n_{ia}-n_{ib})}{(n_{ib}L|M)(d-Mq)+Mq}\right] \quad (12)\end{aligned}$$

if the search is initiated at a node in the high-replica-density cluster, and is *approximately*[13,14] bounded by

$$\begin{aligned}&-\log_d\left[\frac{n_{ib}L}{M}+\frac{q(n_{ia}-n_{ib})}{(n_{ia}L|M)(d-Mq)+Mq}\right]\\ &\le \tau_{iFb}(n_{ia},n_{ib})\\ &\le -\log_d\left[\frac{n_{ib}L}{M}+\frac{q(n_{ia}-n_{ib})}{d}\right] \quad (13)\end{aligned}$$

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

As in the case of random walk search, we see from (12) and (13) that presence of cross-cluster links (i.e. $q \neq 0$) increases the search time for a query initiated by a good node and decreases the 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 (12) and $x = n_{ia}L/M$ for (13) or, in other words, when n_{ib} or n_{ia} are very small or q is small (in which case (10) and (11) provide a good approximation). We also see that the bounds become equal in 3 cases: (a) when $q = 0$, (b) when $n_{ia} = n_{ib}$, and (c) when $d - Mq = 0$. $q = 0$ implies the clusters are disjoint. 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 level of clustering, hence this special case is a generalization of our previous results. 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. 4 we compare the bounds in (12) and (13) to simulation results under varying levels of clustering (i.e. different fractions of links within the cluster and different replica density skews). As expected we find that the bounds are tight under moderate clustering (Fig. 4(a)) and as clustering becomes stronger (Fig. 4(b), (c)) the bounds start to separate but the search time gets closer to (10) and (11). 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 as expected.[13] Based on the analytical arguments given earlier (immediately following Theorem 2) and the simulation results in Fig. 4, we conclude that (10) and (11) are a reasonable approximation for the flooding search time in the network model for clustered demands described in Section III. Therefore, throughout the rest of the paper, we use

$$\tau_{iFa}(n_{ia},n_{ib}) \sim -\log_d\left[\frac{n_{ia}L}{M}-\frac{q(L-1)(n_{ia}-n_{ib})}{d}\right] \quad (14)$$

$$\tau_{iFb}(n_{ia},n_{ib}) \sim -\log_d\left[\frac{n_{ib}L}{M}+\frac{q(n_{ia}-n_{ib})}{d}\right] \quad (15)$$

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 and network topology clustering in Section VIII.

VI. Query-Processing Load With Clustered Demands

As discussed earlier, for the network model described in Section III, the query-processing load in the network can be estimated by d^τ when the average search distance is τ. Hence, using (14) and (15) in Section V, we obtain:

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

$$Q_{iFa}(n_{ia},n_{ib}) \sim \left[\frac{n_{ia}L}{M}-\frac{q(L-1)(n_{ia}-n_{ib})}{d}\right]^{-1} \quad (16)$$

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

$$Q_{iFa}(n_{ia},n_{ib}) \sim \left[\frac{n_{ia}L}{M}-\frac{q(L-1)(n_{ia}-n_{ib})}{d}\right]^{-1} \quad (17)$$

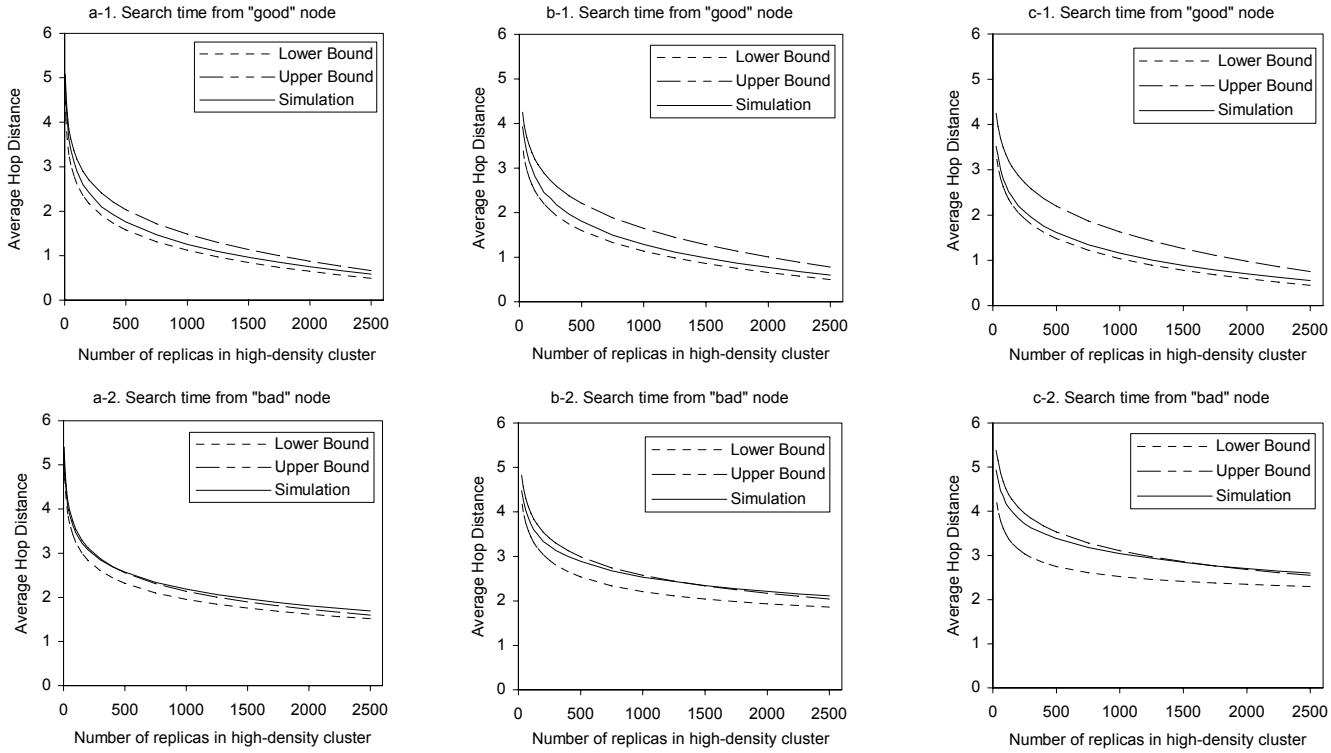
for searches initiated in a low-replica-density cluster. ■

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, observing the inequality signs in (12) and (13) and noticing that exponentiation is monotonic, we obtain the following corollary.

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

$$\begin{aligned}&Q_{iRa}(n_{ia},n_{ib}) \ge Q_{iFa}(n_{ia},n_{ib})\\ &Q_{iRb}(n_{ia},n_{ib}) \le Q_{iFb}(n_{ia},n_{ib}) \quad ■\end{aligned}$$

Corollary 1 suggests that, for arbitrary replica distributions, it may be better to use flooding searches in the high-replica-density cluster and random walk searches in the low-replica-density clusters (at the cost of significantly larger search times for searches from the low-replica-density clusters). However, if a mixed strategy cannot be used (e.g. because accurately determining that the file being queried is an unpopular file may not be trivial), averaged across all requests, random walk will have a lower query-processing load only if $\sum_{i=1}^{N}(\lambda_{ib}Q_{iRb}Q_{iFb} - \lambda_{ia}Q_{iRa}Q_{iFa}) > 0$. Thus, only in a small number of cases, will random walk have a better



(a) 70% links intra-cluster, Replica density skew = 10

(b) 70% links intra-cluster, Replica density skew = 100

(c) 90% links intra-cluster, Replica density skew = 10

Fig. 4. Flooding search time simulation vs. bounds (25,000 node network, 5 equal-sized clusters, average degree 5, varying degree of clustering).

query-processing load than flooding but, in all cases, it has an exponentially higher search time. Therefore, we do not discuss optimization of the search performance with random walk searches in our results in the next section. We also note that this result indicates that the proposal in [18] can be a good approach when topology has clustering matching the clustering in demands.

VII. Flooding Search Performance Optimization

The average search time for flooding search τ_F and the query-processing load for flooding search Q_F over all file requests in the entire network may be written as:

$$\tau_F = \sum_{i=1}^{N} \left[\frac{1}{L} \frac{\lambda_{ia}}{\lambda} \tau_{iFa} + \left(1 - \frac{1}{L}\right) \frac{\lambda_{ib}}{\lambda} \tau_{iFb} \right] \tag{18}$$

$$Q_F = \sum_{i=1}^{N} \left[\frac{1}{L} \frac{\lambda_{ia}}{\lambda} Q_{iFa} + \left(1 - \frac{1}{L}\right) \frac{\lambda_{ib}}{\lambda} Q_{iFb} \right] \tag{19}$$

The average search time and the query-processing load will be smaller if more replicas of each object can be stored but we are limited by the capacity of each node to store only K files. Therefore, we wish to allocate the total available storage so as to minimize the average search time (or, alternately, the query-processing load). Thus, the optimization is subject to the constraint that the total number of replicas of all the files should not exceed the total storage available, i.e. $\sum_{i=1}^{N} n_i \leq KM$. Since $n_i = n_{ia} + (L-1)\, n_{ib}$, the optimization constraint can be written as

$$\sum_{i=1}^{N} \left[n_{ia} + (L-1)\, n_{ib} \right] \leq KM \tag{20}$$

We summarize the constrained optimization results in Theorems 4 and 5 below.

A. Average Search Time Optimization for Flooding Search

Substituting (14), (15) in (18), the Lagrangian [22] for the average search time optimization under the constraint in (20) is

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

Solving for $\partial H / \partial n_{ia} = 0$ and $\partial H / \partial n_{ib} = 0$, we get

$$n_{ia} L/M - q\,(L-1)\,(n_{ia} - n_{ib})\,/d = k \lambda_{ia} \tag{21}$$

$$n_{ib} L/M + q\,(n_{ia} - n_{ib})\,/d = k \lambda_{ib} \tag{22}$$

where k is a constant $\sum_{i=1}^{N} n_i = KM$. Solving for n_{ia}, n_{ib} and k, we get the following theorem.

Theorem 4: The average search time for a flooding search in the clustered peer-to-peer network defined in Section III is minimized when

$$n_{ia} = \left[KM\,(d\lambda_{ia} - Mq\lambda_i) \right] / \left[\lambda L\,(d - Mq) \right] \tag{23}$$

$$n_{ib} = \left[KM\,(d\lambda_{ib} - Mq\lambda_i) \right] / \left[\lambda L\,(d - Mq) \right] \tag{24}$$

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

$$n_i = MK\lambda_i/\lambda \tag{25}$$

$$\tau_F^{opt} = -\sum_{i=1}^{N} \left[\begin{array}{c} \frac{1}{L}\frac{\lambda_{ia}}{\lambda}\log_d\left(\frac{\lambda_{ia}}{\lambda}\right) + \left(1-\frac{1}{L}\right) \\ \frac{\lambda_{ib}}{\lambda}\log_d\left(\frac{\lambda_{ib}}{\lambda}\right) \end{array} \right] - \log_d K \tag{26}$$

$$Q_F^{\tau opt} = N/K \tag{27}$$

i.e. the optimal average search time is independent of q (the level of clustering in the 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. ■

B. Query-Processing Load Optimization for Flooding Search

Substituting (16), (17) in (19), the Lagrangian [22] for the query-processing load optimization under the constraint in (20) is

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

Solving for $\partial H/\partial n_{ia} = 0$ and $\partial H/\partial n_{ib} = 0$, we get

$$n_{ia}L/M - q\,(L-1)\,(n_{ia} - n_{ib})\,/d = k\sqrt{\lambda_{ia}} \tag{28}$$

$$n_{ib}L/M + q\,(n_{ia} - n_{ib})\,/d = k\sqrt{\lambda_{ib}} \tag{29}$$

where k is a constant s.t. $\sum_{i=1}^{N} n_i = KM$. Solving for n_{ia}, n_{ib} and k, we get the following theorem.

Theorem 5: The query-processing load for a flooding search in the clustered peer-to-peer network defined in Section III is minimized when

$$n_{ia} = \frac{(dL - Mq)\sqrt{\lambda_{ia}} - Mq\,(L-1)\sqrt{\lambda_{ib}}}{L\,(d - Mq)\sum_{i=1}^{N}\left[\sqrt{\lambda_{ia}} + (L-1)\sqrt{\lambda_{ib}}\right]} KM \tag{30}$$

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

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

$$Q_F^{opt} = \frac{1}{K}\left(\sum_{i=1}^{N}\left[\frac{1}{L}\sqrt{\frac{\lambda_{ia}}{\lambda}} + \left(1-\frac{1}{L}\right)\sqrt{\frac{\lambda_{ib}}{\lambda}}\right]\right)^2 \tag{32}$$

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

$$\tau_F^{opt} = -\frac{1}{2}\sum_{i=1}^{N}\left[\begin{array}{c} \frac{1}{L}\frac{\lambda_{ia}}{\lambda}\log_d\left(\frac{\lambda_{ia}}{\lambda}\right) \\ +\left(1-\frac{1}{L}\right)\frac{\lambda_{ib}}{\lambda}\log_d\left(\frac{\lambda_{ib}}{\lambda}\right) \end{array} \right] + \log_d\left(\sum_{i=1}^{N}\left[\frac{1}{L}\sqrt{\frac{\lambda_{ia}}{\lambda}} + \left(1-\frac{1}{L}\right)\sqrt{\frac{\lambda_{ib}}{\lambda}}\right]\right) - \log_d K \tag{33}$$

■

C. Interpretation of Optimal Search Performance Results

We find it very interesting that the optimal search performance is independent of q, the level of clustering in topology of the search network. This suggests that our bounds may be the fundamental bounds on the search performance given a demand distribution. We do note that, in our case, the impact of clustering in the topology is evidenced in the file replica distribution needed for the optimal search performance as the file replica distribution does depend on the clustering in the topology. For example, when $q = 0$, i.e. the clusters are disconnected, and as expected, we get $n_{ia} \propto \lambda_{ia}$, $n_{ib} \propto \lambda_{ib}$ from (23) and (24) for search time optimization and $n_{ia} \propto \checkmark\lambda_{ia}$, $n_{ib} \propto \checkmark\lambda_{ib}$ from (30) and (31) for query-processing load optimization. We also see that the optimal average search time expression, which was related to the entropy in the file request probabilities $\{\lambda_i/\lambda\}$ in the uniform file distribution case, changes only in that the entropy expression now includes the spatial distribution of file requests. In [31], we provide an information-theoretic argument for this relation between the minimum search time and the entropy in file request probabilities. The optimal query-processing load also changes only in that the expression includes the spatial distribution of file requests in the clustered demands case. Finally, we note that (25), the overall replica distribution across the network, is the same as the proportional replica distribution shown to be optimal for the search time in the uniform file distribution case. Thus, minimizing the average search time in the clustered demands case also results in download time optimality and fairness in download load distribution [30].

We also note that the l.h.s. in (21), (28), (22), and (29) is equal to the probability of finding the file over a random outgoing link from a node in the high-replica-density cluster and a low-replica-density cluster respectively. Thus, while the expressions for the optimal replica distribution are complex in the case of clustered demands, we still have the invariants as summarized in the following theorem.

Theorem 6: For flooding search in the clustered peer-to-peer network defined in Section III, 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 $\pi_{ij} \propto \checkmark\lambda_{ij}$

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

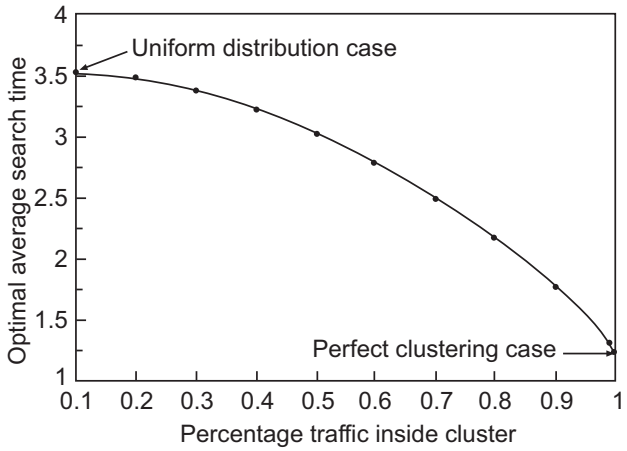


Fig. 5. Benefit of clustered demands: optimal search time.

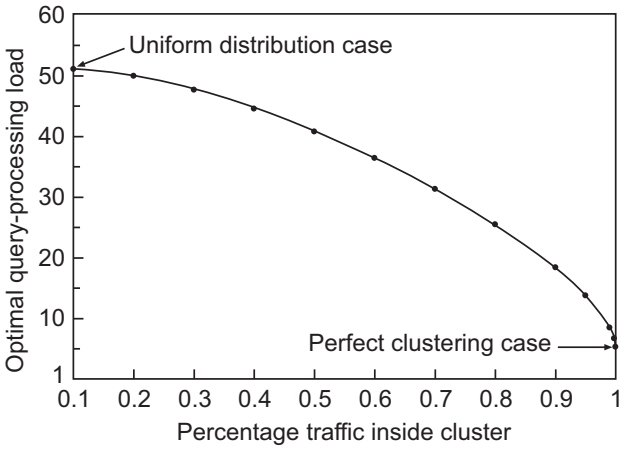


Fig. 6. Benefit of clustered demands: optimal query-processing load.

part[15] of (26) and (32) in Figs. 5 and 6 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. $\lambda_{ib} = 0$ and $\lambda_{ia} = L\lambda_i$. Figs. 5 and 6 clearly demonstrate the potential advantage of clustering. As expected, we see that with perfect clustering, 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 in the network (this can be directly seen from (1) and (2) since each search is now over only the M/L nodes within the cluster).

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. 6, $Q_F^{\tau opt} = 100$ independent of the fraction of traffic inside the cluster while the optimum query-processing load $Q_F^{opt} \sim 50$ in the uniform distribution case but goes down to ~ 8.5 when 99% of the file requests are from inside the cluster. In other words, the additional search cost beyond the minimum possible in optimizing for search time increases from a factor of ~ 2 to a factor of more than 10. On the other hand, the search time incurred upon optimizing the query-processing load increases only slightly. For example, for the peer-to-peer network shown in Fig. 5, the search cost upon optimizing the query-processing load is approximately double[16] that of the optimal search time in the uniform distribution case and this factor increases to only 2.3 times when 99% of the traffic is inside the cluster (in fact, this factor was never more than 2.5 in this network regardless of the level of clustering). However, depending on the application scenario, if the clustering in demands is not very strong or if the query-processing load is not a big concern but the search time is critical, one may still optimize the search time.

[15]To eliminate the dependence on d and K, in Figs. 5 and 6, we plot $\tau'_{opt} = -\sum_{i=1}^{N}\left[\frac{1}{L}\frac{\lambda_{ia}}{\lambda}\ln\left(\frac{\lambda_{ia}}{\lambda}\right) + \left(1-\frac{1}{L}\right)\ln\left(\frac{\lambda_{ib}}{\lambda}\right)\right]$ and $Q'_{opt} = \left(\sum_{i=1}^{N}\left[\frac{1}{L}\sqrt{\frac{\lambda_{ia}}{\lambda}} + \left(1-\frac{1}{L}\right)\sqrt{\frac{\lambda_{ib}}{\lambda}}\right]\right)^2$ instead of (26) and (32) respectively.

[16]The penalty over the optimum search time in optimizing the query-processing load is equal to $-\frac{1}{2}\sum_{i=1}^{N}\left[\frac{1}{L}\frac{\lambda_{ia}}{\lambda}\log_d\left(\frac{\lambda_{ia}}{\lambda}\right) + \left(1-\frac{1}{L}\right)\frac{\lambda_{ib}}{\lambda}\log_d\left(\frac{\lambda_{ib}}{\lambda}\right)\right]$ $+\log_d\left(\sum_{i=1}^{N}\left[\frac{1}{L}\sqrt{\frac{\lambda_{ia}}{\lambda}} + \left(1-\frac{1}{L}\right)\sqrt{\frac{\lambda_{ib}}{\lambda}}\right]\right) = \left(\tau_{opt} + \log_d Q_{opt}\right)/2.$

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 III. Specifically, while we still assume that each node has d neighbors, the clusters can be of different sizes now, a cluster can have more links to one cluster than to another and the request rate for a file can be different in each cluster. We define the following notation for our discussion in this section.

τ_{ik} : average search time to find 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 $p_{jk} = n_{jk}/C_j$. As before, we have M nodes and N files in the network. Using this notation, we obtain the total request rate in the network

$$M\lambda = \sum_{k=1}^{N}\sum_{i=1}^{M}\lambda_{ik} \tag{34}$$

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

$$\sum_{k=1}^{N} p_{jk} \leq K \tag{35}$$

The metrics of interest are

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

and

$$Q = \sum_{k=1}^{N} \sum_{i=1}^{M} \frac{\lambda_{ik}}{M\lambda} Q_{ik} \tag{37}$$

As we noted from our results in Section V, a good approximation for the search time for file j from node i in a flooding search is $\tau_{ij} \approx -\log_d(\pi_{ij})$ 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

$$\tau_{ij} \approx -\log_d \sum_{j=1}^{M} q_{ij} p_{jk}. \tag{38}$$

Since our search network still expands as d^τ in τ hops, from (34), we obtain

$$Q_{ik} = \left(\sum_{j=1}^{M} q_{ij} p_{jk} \right)^{-1} \tag{39}$$

Substituting (38) in (36) and (39) in (37), we obtain

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

$$Q = \frac{1}{M} \sum_{k=1}^{N} \sum_{i=1}^{M} \frac{\lambda_{ik}}{\lambda} \left(\sum_{j=1}^{M} q_{ij} p_{jk} \right)^{-1} \tag{41}$$

Applying constraint (35), we get the Lagrangian for minimizing the average search distance as

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

Solving for $\partial H / \partial p_{jk} = 0 \forall j, \forall k$, we get

$$\sum_{j=1}^{M} q_{ij} p_{jk} = K \lambda_{ik} / \lambda \quad \forall i, \forall k \tag{42}$$

Substituting (42) in (40) and (41) gives us the following theorem.

Theorem 7: 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

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

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 (35), the Lagrangian is

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

Solving for $\partial H / \partial p_{jk} = 0 \forall j, \forall k$, we get

$$\sum_{j=1}^{M} q_{ij} p_{jk} = \beta' \sqrt{\lambda_{ik}} \quad \forall i, \forall k \tag{43}$$

where $\beta' = MK / \sum_{k=1}^{N} \sum_{i=1}^{M} \sqrt{\lambda_{ik}}$. Substituting (43) in (40) and (41) gives us the following theorem.

Theorem 8: 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.

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

and the minimum query-processing load is

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

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

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

■

Thus, we find that the invariants noted in Theorem 6 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.

IX. Conclusions

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 and the search cost for that file and substantially expanded the existing knowledge on this topic. First, we provided a simple model to incorporate clustering in peer-to-peer network models so they better reflect real networks. For this model, 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 extended 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. We used this observation to generalize our search time and query-processing load results for flooding search to peer-to-peer networks with arbitrary demand and arbitrary network topology clustering. Using these approximate expressions, we derived expressions for the optimal search cost and the optimal search time in an unstructured peer-to-peer network when the demand exhibits clustering. The previous work on the optimal search performance in unstructured peer-to-peer networks had assumed uniformity in the replica and the demand distributions. Our results capture the search performance gains afforded by the clustering in file popularities that has been observed in deployed 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 clustering in the search network topology is accounted for in the optimal replica distribution which does depend on clustering in the search network topology.

REFERENCES

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

[2] M. Bawa, G. S. Manku, and P. Raghavan, "SETS: Search enhanced by topic segmentation," in *Proc. ACM SIGIR*, July 2003.

[3] Y. Chawathe, S. Ratnasamy, L. Breslau, L. Lanham, and S. Shenker, "Making gnutella-like P2P systems scalable," in *Proc. ACM SIGCOMM*, Aug. 2003.

[4] V. Cholvi, P. A. Felber, and E. W. Biersack, "Efficient search in unstructured peer-to-peer networks," *European Trans. Telecommun.*, vol. 15, no. 6, Nov.-Dec. 2004.

[5] E. Cohen, and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proc. ACM SIGCOMM*, Aug. 2002.

[6] E. Cohen, A. Fiat, and H. Kaplan, "Associative search in peer to peer networks: Harnessing latent semantics," in *Proc. IEEE INFOCOM*, Mar. 2003, vol. 2, pp. 1261-1271.

[7] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1. New York: John Wiley & Sons, Inc., 1950.

[8] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes for unstructured peer-to-peer networks," in *Proc. IEEE INFOCOM,* Mar. 2005, vol. 3, pp. 1526-1537.

[9] Gnutella protocol specification. [Online.] Available: http://www.the-gdf.org/

[10] A. Iamnitchi, M. Ripeanu, and I. Foster, "Small-world file-sharing communities," in *Proc. IEEE INFOCOM*, Mar. 2004, vol. 2, pp. 952-963.

[11] M. Khambatti, K. Ryu, and P. Dasgupta, "Structuring peer-to-peer networks using interest-based communities," in *Proc. International Workshop Databases, Inform. Syst. Peer-to-Peer Computing (P2PDBIS)*, Sep. 2003.

[12] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. 32nd ACM Symposium Theory of Computing*, 2000.

[13] A. Klemm, C. Lindemann, M. K. Vernon, and O. P. Waldhorst, "Characterizing the query behavior in peer-to-peer file sharing systems," in *Proc. ACM Internet Measurement Conf. (IMC)*, 2004.

[14] F. Le Fessant, S. Handurukande, A. M. Kermarrec, and L. Massouli, "Clustering in peer-to-peer file sharing workloads," in *Proc. IPTPS*, Feb. 2004.

[15] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth efficient management of DHT routing tables," in *Proc. 2nd Symposium Networked Syst. Design and Implementation (NSDI)*, May 2005.

[16] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the evolution of peer-to-peer systems," in *Proc. 22nd ACM Symposium Principles Distr. Computing (PODC)*, July 2002.

[17] A. Loser, et. al., "Searching dynamic communities with personal indexes," in *Proc. International Semantic Web Conf.*, 2005.

[18] B. T. Loo, J. M. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica, "Enhancing P2P file-sharing with an internet-scale query processor," in *Proc. VLDB*, 2004.

[19] G. S. Manku, "Routing networks for distributed hash tables," in *Proc. 22nd ACM Symposium Principles Distr. Computing (PODC)*, June 2003, pp. 133-142.

[20] W. Nejdl, "Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks," in *Proc. International World Wide Web Conf. (WWW)*, May 2003.

[21] C. H. Ng, K. C. Sia, and C. H. Chang, "Peer clustering and firework query model," in *Proc. International World Wide Web Conf. (WWW)*, May 2002.

[22] D. A. Pierre, *Optimization Theory with Applications*. Dover, 1986.

[23] V. Ramasubramanian and E. G. Sirer, "Beehive: O(1) lookup performance for power-law query distributions in peer-to-peer overlays," in *Proc. Networked System Design Implementation (NSDI)*, Mar. 2004.

[24] O. Sandberg, "Searching in a Small World." University of Gothenburg and Chalmers Technical University, 2005. [Online.] Available: http://www.math.chalmers.se/~ossa/lic.pdf

[25] S. Singh, et al., "The case for service provider deployment of super-peers in peer-to-peer networks," in *Proc. International Workshop Economics Peer-To-Peer Systems*, 2003.

[26] K. Sripanidkulchai, K. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *IEEE INFOCOM*, Mar. 2003, vol. 3, pp. 2166-2176.

[27] I. Stoica, R. Morris, D. Karger, F. M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-To-peer lookup service for internet applications," in *Proc. ACM SIGCOMM*, Aug. 2001.

[28] C. Tang, Z. Xu, and S. Dwarkadas,, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 175–186.

[29] S. Tewari and L. Kleinrock, "Analysis of search and replication in unstructured peer-to-peer networks," UCLA Computer Science Dept Technical Report UCLA-CSD-TR050006, Mar. 2005.

[30] S. Tewari and L. Kleinrock, "On fairness, optimal download performance and proportional replication in peer-to-peer networks," in *Proc. IFIP Networking*, May 2005.

[31] S. Tewari and L. Kleinrock, "Entropy and search distance in peer-to-peer networks," UCLA Computer Science Dept Technical Report UCLA-CSD-TR050049, Nov. 2005.

[32] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search for peer-to-peer networks," in *Proc. International Conf, Peer-to-Peer Computing (P2P)*, 2003.

Saurabh Tewari is a Ph.D. candidate in the Computer Science department at the University of California at Los Angeles. He received an MBA from the University of Texas at Arlington in 2000, an M.S. in Electrical and Computer Engineering from Carnegie Mellon University in 1995 and a B.Tech. in Electrical Engineering from IIT Kanpur in 1992. His research interests span the general areas of networking and distributed systems particularly peer-to-peer networks and other highly-available scalable systems design approaches. Saurabh has worked for over eight years in various research and software and hardware development positions at Alcatel on optical burst routing, scalable router architectures, IP routing software stack, broadband optical transport and ATM switching projects and, at Optomation, an optical sensors and systems. He is a student member of the IEEE and the ACM.

Leonard Kleinrock received his Ph.D. from MIT in 1963. He has served as a Professor of Computer Science at the University of California, Los Angeles since then, serving as Chairman of the department from 1991-1995. He received his BEE degree from CCNY in 1957 and his MS degree from MIT in 1959. He also received Honorary Doctorates from CCNY in 1997, from the University of Massachusetts, Amherst in 2000, from the University of Bologna in 2005, and from Politecnico di Turino in 2005. He was Co-founder and the first President of Linkabit Corporation, the company that spawned numerous wireless spinoffs in San Diego. He is Co-founder and Chairman of Nomadix, Inc., a high-technology firm located in Southern California. He is also Founder and Chairman of TTI/Vanguard, an advanced technology forum organization based in Santa Monica, California. He developed the mathematical theory of packet networks, the technology underpinning the Internet, while a graduate student at MIT nearly a decade before the birth of the Internet which occurred in his laboratory when his Host computer at UCLA became the first node of the Internet in September 1969. He wrote the first paper and published the first book on the subject; he also directed the transmission of the first message ever to pass over the Internet. He has published approximately 250 papers and authored six books on a wide array of subjects including packet switching networks, packet radio networks, local area networks, broadband networks, gigabit networks, nomadic computing, performance evaluation, and peer-to-peer networks. During his tenure at UCLA, Dr. Kleinrock has supervised the research for 46 Ph.D. students and numerous M.S. students. These former students now form a core group of the world's most advanced networking experts. A number are full professors at leading universities, and many are associated with major research firms in the area of computer-communications. Dr. Kleinrock is a member of the National Academy of Engineering, a member of the American Academy of Arts and Sciences, an IEEE fellow, an ACM fellow, an INFORMS Fellow, an IEC fellow, a Guggenheim fellow and a founding member of the Computer Science and Telecommunications Board of the National Research Council. Among his many honors, he is the recipient of the L.M. Ericsson Prize, the NAE Charles Stark Draper Prize, the Marconi International Fellowship Award, the Okawa Prize, the IEEE Internet Millennium Award, the ORSA Lanchester Prize, the ACM SIGCOMM Award, the NEC Computer and Communcations Award, the Sigma Xi Monie A. Ferst Award, the CCNY Townsend Harris Medal, the CCNY Electrical Engineering Award, the UCLA Outstanding Faculty Member Award, the UCLA Distinguished Teaching Award, the UCLA Faculty Research Lecturer, the INFORMS Presidents Award, the ICC Prize Paper Award, the IEEE Leonard G. Abraham Prize Paper Award and the IEEE Harry M. Goode Award and the NEC Computer & Communications Prize.