Modeling and Simulation of Adhoc/P2P Resource Sharing Networks

Krishna Kant
Enterprise Architecture Lab
Intel Labs
Oregon

Ravi Iyer
Network Architecture Lab
Intel Labs
Oregon

Abstract

Performance evaluation of adhoc peer-to-peer resource sharing networks is difficult since nearly every aspect of such networks including topology and offered resources can change unpredictably. In this paper we introduce a random-graph based model for studying the evolution of ad hoc peer-to-peer (P2P) communities (such as Gnutella/Freenet). The proposed random graph model generates a non-uniform graph and provides control over the nodal degree distribution. Based on this random graph model, we present analytic and simulation-based approaches to understand performance characteristics and issues. The analytic approach studies basic properties such as reachability and nodal utilization in the network. To address more detailed performance characteristics, we present our simulation framework called SimP^2. We discuss the crucial issue of efficiently simulating a random graph model by looking at several potential approaches. SimP^2 incorporates several performance-related aspects of the P2P environment including reachability metrics, queuing behavior, content caching at each node, impact of user timeouts, etc., which are illustrated via an example. While the current study targets a simple static P2P file sharing environment likw Gnutella or Freenet, SimP^2 is intended for open-source release shortly in order to facilitate the evaluation of other more dynamic networks and perhaps intelligent searches.

1 Introduction

Distributed peer-to-peer (P2P) file sharing started by Gnutella and Freenet continues to attract a subtantial interest due to its applicability to ad-hoc collaboration networks. The recent explosive growth of Wi-Fi

(IEEE 802.11) networks has made ad-hoc file-sharing networks particularly attractive. Such networks may range in size from very small (a few laptops with 802.11 hooked up together into a network) to very large (a large wired network or computational grid, further augmented by 802.11 equipped mobile participants).

A study of the performance of such networks is complicated by the wide variations and dynamic changes in network size, topology, peer capabilities, communication bandwidths, characteristics of the file shared, etc. Yet a performance study is essential to study better means for information location and transfer, and for minimizing unnecessary work. Our goal in this paper is to come up with a viable approach to evaluating various performance charactistics of adhoc P2P networks. This includes not only the overall methodology, but also the development of the necessary tools for this purpose.

In this paper, we facilitate performance studies of adhoc P2P networks by introducing a random-graph model that attemts to capture some basic properties of these networks, thereby providing a concrete way of emulating and analyzing them. We then analyze various performance characteristics of the network through analytic and simulation approaches. Due to the inherent complexity of the problem, the analytic model is limited to studying basic properties such as reachability and nodal utilizations. To address detailed performance characteristics such as queuing delays and message expiry or loss, we present a simulation framework called SimP^2. SimP^2 consists of two parts: (a) Construction of a set of P2P network instances, and (b) Parallel simulation of file search & retrieval over these instances. The first part is based on the random graph model discussed above. The second part allows one to study several aspects of P2P network, both topological and performance related. In particular, it allows the study of reachability properties, total traffic processed by various nodes, queuing behavior of nodes, performance impact of local caching of files, impact of user behavior including timeouts, etc. An important point to note here is that the second part of the tool could well be given networks (or graphs) that don't necessarily come from the first part and thus can be applied in a larger context.

Section 2 provides a very brief overview of the relevant work in P2P file-sharing area and the modeling assumptions made in our work. Section 3 introduces the proposed random graph model for representing adhoc networks. Section 4 presents the analytic approach to evaluating P2P file sharing networks. Section 5 presents our simulation-based approach to evaluating detailed performance characteristics of P2P file-sharing networks. In this section, Section 5.1 discusses some of the difficulties in simulating random graphs and some potential approaches. Section 5.2 provides an overview of our simulation framework called, SimP^2. Section 6 illustrates some uses of SimP^2 by providing some sample performance results

of a case study. Finally, section 7 summarizes the work and points out future work in this area.

2 Peer-to-Peer File-Sharing Networks and Assumptions

The peer-to-peer file-sharing revolution started with Napster, which supported music file-sharing among geographically distributed clients via a centralized server that maintained file-location information. This was soon followed by Gnutella and Freenet, both of which supported sharing of arbitrary files without any centralized location agent. Instead, files were located by an explicit spanning tree type of search for each exchange. Gnutella supported a hop-based limitation of search propagation coupled with a stamping scheme to avoid duplicate node visits.

A P2P file-sharing network is basically an ad-hoc network with arbitrary topology and size that change as existing nodes leave the network and new ones enter. It is possible to model such network directly, however, the very large degrees of freedom generally allow only approximate or asymptotic results. For example, reference [12] presents a model for building low-diameter dynamic P2P networks where nodes arrive according to a Poisson process and stay in the network for an exponential amount of time. They prove a number of interesting asymptotic results concering the connectivity and the diameter of the largest connected component.

In order to obtain more concrete results, we instead disregard dynamic changes to the network. The justification for this approach follows from the observations concerning real Gnutella networks in [6, 1, 14]. It has been observed in these and several other papers that Gnutella is not “democratic” in the sense that all nodes don’t behave in a similar way. Instead, it is possible to identify roughly 3 ”tiers” of nodes as discussed below:

tier1: These are globally known nodes that stay in the network almost indefinitely, have high-speed connectivity and are always available. Effectively these nodes act like “servers” in traditional networks. We call these “distinguished nodes”.

tier2: These are non-permanent nodes that are not free-riding [1], i.e., they contribute significant number of files to the community, are reasonably resource rich, and stay on far longer than an average request-response time. We call these as “undistinguished nodes”.

tier3: These are transient nodes that join the network for short periods of time, connect to only a few

other nodes, and typically do not contribute much to the community.

For the purposes of the analysis in this paper, we ignore the transient nodes and instead assume that the traffic generated by these nodes essentially originates at other nodes to which they connect to. This assumption allows separation of the request-response process in the network from the network topology modification process (due to undistinguished nodes joining/leaving the network). This allows us to work with a network instance with some given number of nodes, and completely avoid simultaneous consideration of the network change dynamics. A separate model could then be used to model network changes and from there compute the distribution of number of nodes of each type. The separability then allows a simple superpositoin of the results. For brevity, we omit network change aspect in this paper.

Because of lack of global knowledge in an uncoordinated P2P network such as Gnutella, requests for documents invariably result in a spanning-tree like search through the network up to a certain number of "hops". If each message is stamped with a globally unique ID, duplicate responses for the same request from a given node can be easily avoided (as in Gnutella). More sophisticated approaches include maintaining locational information either via "snooping" the file transfers or via a special update protocol. The snooping can be facilitated by sending response along the search path (in reverse direction) and is used in Freenet [3]. We shall consider both direct and reverse path responses.

Recently there have been several other proposals to reduce the cost of the searches. In particular, the idea of "distributed hash tables" (DHT) has been used in several recent projects including PASTRY, CAN, Chord and Tapestry. Here, each node is made responsible for a subset of the entire key space by assigning an Id from the hash over the key space. A search message is directed to the node whose Id is closest to the key in the hash space. Thus, a resource can usually be located in a small number of hops by redirection towards the node whose Id has a better match with the key than the current node. References to these and some discussion of how to exploit network proximity in the search can be found in [2].

In this paper, we shall stick to the simple Gnutella type of network. The "intelligent search" techniques discussed above essentially result in yet another layer of overlay network that is different for each search. Note that the P2P network itself is overlaid on the physical Internet; thus, even a direct modeling of Gnutella doesn't quite capture what happens in the physical network. The issues of efficient mapping of P2P networks onto physical network are well known but very difficult problems that we shall ignore in this paper. Our results still provide some insight into the performance with respect to the overlay network being considered.

Along with the above simplifying assumptions, we also assume a rather idealized case of a corporate intranet type of environment where all nodes sit on a high-speed network, all peer nodes are always turned on and connected to the network, and there are no significant down-time issues to be considered for performance modeling purposes.

3 Random Graph Model

Following the discussion in the last section, we consider a 2-tier graph model with "distinguished" and "undistinguished" nodes. The distinguished nodes and connections between them exist right from the beginning and this subnetwork acts as the nucleus of the file sharing network. We assume an initial network with N_d distinguished nodes connected in a regular pattern with degree $M > 0$. The undistinguished nodes join the network sequentially. Each joining node, say X, connects to a total of $\mathcal{K}$ nodes, where $\mathcal{K}$ is a random variable with the mass function $\theta_k, k = 1..K$. Each connection attempt from joining node X is directed to an undistinguished node with probability q_u, and to a distinguished node with probability $(1 - q_u)$. While the network contains less than K undistinguished nodes, any excess connections are directed to distinguished nodes. Given that the total number of nodes in the graph, henceforth denoted as N_t, is much larger than K, this initial perturbation has very little impact on the overall network.

One important property of such a graph construction is nonuniformity: distinguished nodes and the nodes that enter the graph early on have, on the average, higher connectivity than others. Moreover, the connectivity distribution has a heavy tail (unlike the exponentially decaying tail in a classical random graph). In fact, we explicitly introduce a parameter β which controls how heavy the tail would be. When the ℓ_2th node attempts to create a connection with an undistinguished node, it has the choice of connecting to up to $\ell_2 - 1$ of them. The function $\beta(\ell, \ell_2)$ gives the probability of selecting level ℓ node in this case. Note that if $\beta(.,.) = 1$, the average degree of a node entering at level ℓ is governed by the term $\sum_{i=\ell}^{\ell_2-1} 1/i$. For a given ℓ_2, this is a very slowly decaying function of ℓ. By choosing $\beta(\ell, \ell_2)/approx\ell^{-\mu}$ for some $\mu \geq 1$, it is possible to obtain a much faster decay and hence a less heavy tail in the degree distribution. It may be noted that heavy-tailed degree distribution is commonly observed in not only Gnutella [6] P2P networks but also in other contexts including web [10] and disease propagation networks [11].

In order to study the performance issues stated above, we need to characterize a number of properties including nodal degree, reachability, total request and response traffic at each node and queuing properties.

The next section introduces an analytic model for this purpose. The simulation of random graphs is addressed in a subsequent section.

4 Analytic Modeling of P2P Networks

Let us start with the characterization of nodal degree. It is convenient to introduce the notion of a "level" for each node. Since all distinguished nodes are treated identically, we consider all of them to lie at level 0. Each undistinguished node, however, introduces a new level. Thus, when the network has a total of N_t nodes, it has $N_t - N_d + 1$ levels, numbered $0 .. N_t - N_d$, with all level 0 nodes being distinguished, and others undistinguished. Henceforth, we let $L = N_t - N_d$ as the maximum level number.

Let $P_n(\ell_2, \ell)$ denote the probability that a node that entered the network at level ℓ has a degree n when a level ℓ_2 node enters the network. Note that following the inclusion of level ℓ_2 node, the total number of nodes in the network is $\ell_2 + N_d$. Also, ℓ_2 denotes the number of undistinguished nodes *after* the addition of the new node. In the following, we show how to obtain the probability generating function $\Phi(z|L, \ell)$ of $P_n(\ell_2, \ell)$, and hence compute $D_{at}(\ell|L)$, defined as the mean degree *at* a node that enters the network at level ℓ assuming a total of L levels.

4.1 Computation of Nodal Degree

In order to write a recurrence equation for $P_n(\ell_2, \ell)$, we introduce the connection probability $\alpha(\ell_2, \ell)$, defined as the probability that the addition of a new node to the network (at level ℓ_2) will result in one more connection to a level ℓ node.

Let us first suppose that $\ell > 0$, i.e., we are considering how the degree of a nondistinguished node changes with the new node addition. Suppose that the level ℓ_2 node attempts to connect to k nodes, of which i are undistinguished nodes. When $\ell_2 = 1$, $\alpha(\ell_2, \ell) = 0$ for $\ell > 0$ since there are no undistinguished nodes to connect. Otherwise, if $\ell_2 - 1 < i$, only $\ell_2 - 1$ connections are actually possible with undistinguished nodes. It follows that for $\ell_2 > 1$, $\ell > 0$,

$$\alpha(\ell_2, \ell) = \sum_{k=0}^{K} \theta_k \left[\sum_{i=1}^{\min(\ell_2 - 1, k)} \beta(\ell, \ell_2) \frac{i\, B_{k,i}(q_u)}{\ell_2 - 1} + \sum_{i=\ell_2}^{k} B_{k,i}(q_u) \right]$$

where $B_{k,i}(q_u)$ is the binomial mass function $B_{k,i}(q_u) = \binom{k}{i} q_u^i (1-q_u)^{k-i}$ with $B_{0,0}(q_u) = 1$. The equation follows from the fact that the probability of connection attempt to i undistinguished nodes is $B_{k,i}(q_u)$, and a given node will be selected with probability $\frac{\beta(\ell,\ell_2)i}{(\ell_2-1)}$ if $i \leq \ell_2 - 1$. For $i > \ell_2 - 1$, every undistinguished node must be targeted for connection.

For $\ell_2 > K$, the entire expression collapses to the following intuitively obvious expression:

$$\alpha(\ell_2, \ell) = q_u E[\mathcal{K}] \frac{\beta(\ell, \ell_2)}{(\ell_2 - 1)}, \qquad \ell_2 > K \tag{1}$$

Note that only the mean value of $\mathcal{K}$ is relevant in this case; the rest of the distribution doesn't matter.

Next, let us evaluate $\alpha(\ell_2, 0)$. Note that if the new node connects to i undistinguished nodes, it will connect to $k - i$ distinguished nodes. Since there are a total of $N_d \geq K$ distinguished nodes, we have $\alpha(\ell_2, 0) = \sum_{k=0}^{K} \theta_k \zeta(k, \ell_2)$ where

$$\zeta(k, \ell_2) = \sum_{i=0}^{\min(\ell_2 - 1, k)} \frac{k-i}{N_d} B_{k,i}(q_u) + \sum_{i=\ell_2}^{k} \frac{k - \ell_2 + 1}{N_d} B_{k,i}(q_u)$$

If $\ell_2 > K$, the entire expression again collapses to the obvious equation:

$$\alpha(\ell_2, 0) = (1 - q_u) E[\mathcal{K}]/N_d, \qquad \ell_2 > K \tag{2}$$

Assuming that $P_n(.,.) = 0$ for $n < 0$, we can now write the following recurrence equation for $P_n(\ell_2, \ell)$ as:

$$\begin{aligned} P_n(\ell_2, \ell) &= \alpha(\ell_2, \ell) P_{n-1}(\ell_2 - 1, \ell) \\ &+ [1 - \alpha(\ell_2, \ell)] P_n(\ell_2 - 1, \ell) \end{aligned} \tag{3}$$

For $\ell > 0$, this equation applies for $\ell < \ell_2$, i.e., when level ℓ node in consideration is not the same as the node being added to the network. For $\ell = \ell_2$, the degree of the added node is simply $\mathcal{K}$. That is,

$$P_k(\ell, \ell) = \theta_k, \qquad 0 \leq k \leq K \tag{4}$$

For $\ell = 0$, the above recurrence equation fails to apply only initially, i.e., when the network consists of only distinguished nodes. Since we are assuming a regular M-ary connectivity initially, we have:

$$P_n(0, 0) = \begin{cases} 1 & n = M \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Let $\Phi(z|L,\ell)$ denote probability generating function of the degree distribution. We have:

$$\begin{aligned}\Phi(z|\ell_2,\ell) &= \sum_{n=0}^{\ell_2-\ell+K} z^n P_n(\ell_2,\ell) \\ &= [1-\alpha(\ell_2,\ell)(1-z)]\Phi(z|\ell_2-1,\ell), \qquad \ell < L\end{aligned}$$

where we have used the fact that $P_{\ell_2-\ell+K}(\ell_2-1,\ell)=0$. Using this relationship recursively, we get

$$\Phi(z|L,\ell) = \Phi(z|\ell,\ell) \prod_{i=\ell+1}^{L} [1-\alpha(i,\ell)(1-z)] \tag{6}$$

It thus follows:

$$\Phi(z|L,\ell) = \Psi(z) \prod_{i=\ell+1}^{L} [1-\alpha(i,\ell)(1-z)] \tag{7}$$

where the function $\Psi(z)$ is defined as:

$$\Psi(z) = \begin{cases} \sum_{k=0}^{K} z^k \theta_k & \ell > 0 \\ \\ z^M & \ell = 0 \end{cases} \tag{8}$$

This completes the characterization of the degree distribution. Let $D_{at}(\ell|L)$ denote the mean degree *at* a node that enters the network at level ℓ, given a total of L levels. By definition,

$$D_{at}(\ell|L) = \Phi'(z|L,\ell)\Big|_{z=1} = \Psi'(1) + \sum_{j=\ell+1}^{L} \alpha(j,\ell) \tag{9}$$

Now we separately consider the $\ell > 0$ and $\ell = 0$ cases. In the former case, if $\ell \leq K$, the above summation can be divided into two parts, one for the range $\ell+1\,..\,K$ and the other from $K+1\,..\,L$. Then, using equation (1), for $0 < \ell < K$, we get:

$$D_{at}(\ell|L) = E[\mathcal{K}] + q_u E[\mathcal{K}] \sum_{i=K}^{L-1} \frac{\beta(\ell,i)}{i} + \sum_{j=\ell+1}^{K} \alpha(j,\ell) \tag{10}$$

where the last term drops out for $\ell > K$. Now, for $\ell = 0$, we have two ranges, one $0\,..\,K$, and the other $K+1\,..\,L$. Using equation (2), we have:

$$D_{at}(0|L) = M + E[\mathcal{K}](1-q_u)\frac{L-K}{N_d} + \sum_{j=1}^{K} \alpha(j,0) \tag{11}$$

If required, the overall average degree of a node could then be computed as: $D_{at}(L) = (N_d D_{at}(L,0) + \sum_{\ell=1}^{L} D_{at}(\ell|L))/N_t$.

Let us now adapt the above analysis to obtain the undistinguished degree of a node, defined as the number of undistinguished nodes to which a given node is directly connected. Let us denote this as $P_n^{(u)}(\ell_2, \ell)$, defined just like the overall degree $P_n(\ell_2, \ell)$. Since every added node in the network construction process is a undistinguished node, equation (3) holds for $P^{(u)}$ as well, except that the initial conditions are different. In particular, let k_u denote the number of undistinguished nodes that a newly arriving level $\ell > 0$ node connects to. Obviously, $k_u \in 0 \,..\, \min(K, \ell - 1)$; therefore, $\theta_u(k_u, \ell)$, the probability of a newly arriving level ℓ node connecting to k_u undistinguished nodes is:

$$\theta_u(k_u, \ell) = \sum_{k=k_u}^{K} \theta_k B_{k,k_u}(q_u) \tag{12}$$

Note that even if $\theta_0 = 0$ (i.e., an incoming node always connects to at least one node), k_u could still be 0. Also, $\theta_u(k_u, \ell)$ depends on ℓ for $\ell \leq K$ only. Now, equation (4) becomes $P^{(u)}(k_u|\ell, \ell) = \theta_u(k_u, \ell)$. Since, a distinguished node is initially not connected to any undistinguished node, eqn(5) reduces to:

$$P_n^{(u)}(0, 0) = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

With this, the PGF of $P^{(u)}$ is given by an equation like (7), but with Ψ function suitably modified for the changed initial conditions, i.e.,

$$\Psi_u(z) = \begin{cases} \sum_{k=0}^{K} z^k \theta_u(i, \ell) & \ell > 0 \\ \\ 1 & \ell = 0 \end{cases} \tag{14}$$

Let $D_{at,u}$ be defined like D_{at} except that we are now interested in only the undistinguished degree of a node. Equations (10–11) then apply to $D_{at,u}$ as well except for the first term modified to correspond to $\Psi'_u(1)$ instead of $\Psi'(1)$. Note that for $\ell > K$,

$$\Psi'_u(1) = \sum_{k_u=0}^{K} k_u \theta_u(k_u, \ell) = \sum_{k=0}^{K} \theta_k \sum_{k_u=0}^{k} k_u B_{k,k_u}(q_u)$$

which evaluates to $q_u E[\mathcal{K}]$, as expected.

4.2 Reachability Distribution

Next we consider the computation of the number of nodes reached starting from a given node. For this, we first characterize the reachability matrix $\mathcal{R}_h$ whose (i, j)th element gives the probability of reaching

level i from level j in exactly h hops (i.e., via paths with exactly $h-1$ unique intermediate nodes different from nodes i and j). This reachability matrix, along with the nodal degree characterization, yields the average number of nodes reached in h hops. We henceforth let $G(\ell_2, h)$ denote the average number of nodes reached *at* hth hop, starting from level ℓ_2. Similarly, let $G_u(\ell_2, h)$ denote the number of undistinguished nodes reached in h hops starting from level ℓ_2. The computation of $\mathcal{R}_h$, $G(\ell_2, h)$, and $G_u(\ell_2, h)$ is discussed below.

We start with the reachability of all nodes from a given node. For this, we arbitrarily number the distinguished nodes as $1 \,..\, N_d$, and subsequently a level ℓ node as $\ell + N_d$. Let $R_1(n_1, n_2)$ denote the probability of reaching node n_1 from node n_2 in one hop. Let R_1 denote the corresponding $N_t \times N_t$ matrix. To specify R_1, we first compute a related matrix R_0 that enumerates the probability of finding an arc between any given pair of nodes. Since R_0 is a symmetric matrix and for any node n_1, $R_0[n_1, n_1] = 0$; therefore, it suffices to assume $n_2 > n_1$ in the following:

$$R_0[n_1, n_2] = \begin{cases} M/(N_d - 1) & n_1 \leq N_d, n_2 \leq N_d \\ \alpha(n_2 - N_d, 0) & n_1 \leq N_d, n_2 > N_d \\ \alpha(n_2 - N_d, n_1 - N_d) & n_1 > N_d, n_2 > n_1 \end{cases} \tag{15}$$

Given R_0, we construct R_1 by normalizing the columns to 1. This ensures that the total probability of reaching all other nodes from any node is 1. Note that R_1 is not a symmetric matrix, but it is possible to exploit its special structure.

Let $R_h[n_1, n_2]$ denote the probability of reaching node n_1 from node n_2 in h hops. In order to compute the matrix R_h, it is necessary to enumerate all unique paths of length h that do not use any intermediate node more than once. The latter requirement increases the computational cost, since a straightforward recursion on h cannot avoid node reuse. We also need to renormalize the matrix at each step to ensure that the columns sum to 1. We denote the unnormalized matrices as R'_h where we specifically set $R'_h[n, n] = 0$ for all n. Then, $R_h = R'_h \times \text{diag}(\mathbf{e}.R'_h)$ where $\mathbf{e}$ denotes a row vector of all 1's and the function diag converts a row vector into a diagonal matrix by putting the vector elements along the diagonal. Now, for $n_1 \neq n_2$,

$$R'_2[n_1, n_2] = \sum_{n_3=1}^{N_t} R_1[n_1, n_3] R_1[n_3, n_2] \tag{16}$$

where it is not necessary to specifically avoid $n_3 = n_1$ or $n_3 = n_2$ since $R_1(n, n) =$ for all n. However,

for R'_3, we have:

$$R'_3[n_1, n_2] = \sum_{\substack{n_4=1 \\ n_4 \neq n_2}}^{N_t} \sum_{\substack{n_3=1 \\ n_3 \neq n_1}}^{N_t} R_1[n_1, n_4] R_1[n_4, n_3] R_1[n_3, n_2]$$

Similar equations can be written for R'_i, $i > 3$. We now define a modified $(L+1) \times (L+1)$ matrix $\mathcal{R}_h$ where the rows and columns indicate level, rather than individual nodes:

$$\mathcal{R}_h[\ell_1, \ell_2] = \begin{cases} (N_d - 1)\ R_h[1,2] & \ell_1 = 0, \ell_2 = 0 \\ R_h[\ell_1 + N_d, 1] & \ell_1 > 0, \ell_2 = 0 \\ N_d\ R_h[1, \ell_2 + N_d] & \ell_1 = 0, \ell_2 > 0 \\ R_h[\ell_1 + N_d, \ell_2 + N_d] & \ell_1 > 0, \ell_2 > 0 \end{cases}$$

This matrix suffices for further computation. The sole reason for defining R matrices in terms of node numbers earlier was to ensure that no node gets used more than once on a path.

Let $D_{fr}(\ell_2, h|L)$ denote the average degree of the nodes reached in h hops starting *from* a node at level ℓ_2 when the maximum level is L. It is convenient to denote row vectors of $D_{at}(\ell|L)$'s and $D_{fr}(\ell, h|L)$'s for $\ell = 0..L$ as $\mathbf{D}_{at}(L)$ and $\mathbf{D}_{fr}(h|L)$ respectively. Then, $\mathbf{D}_{fr}(1|L) = \mathbf{D}_{at}(L)$, and for $h > 1$

$$\mathbf{D}_{fr}(h|L) = \mathbf{D}_{at}(L)\mathcal{R}_{h-1} \tag{17}$$

In the analysis presented here, the last equation is the only place where the matrix $\mathcal{R}_h$ is used; therefore, explicit computation of $\mathcal{R}_h$ is unnecessary (and very expensive). Instead, it is desirable to compute $\mathbf{D}_{fr}(h|L)$ directly for each h.

Let $G(\ell_2, h)$ denote the average number of nodes reached *at* hth hop, and $G^{(t)}(\ell_2, h)$ the total number of nodes reached in h hops (ℓ_2 is the starting level in both cases). Again, it is convenient to work with the corresponding row vectors henceforth denoted as $\mathbf{G}(h)$ and $\mathbf{G}^{(t)}(h)$. Then, $\mathbf{G}(1) = \mathbf{D}_{at}(L)$ (i.e., the degree of the starting node) and $\mathbf{G}^{(t)}(0) = \mathbf{e}$ (the starting or "root" node). For $h > 0$, we have:

$$\mathbf{G}^{(t)}(h) = \mathbf{G}^{(t)}(h-1) + \mathbf{G}(h), \qquad 1 \leq h \leq H \tag{18}$$

where H denotes the highest hop count of interest. Now, to obtain an expression for $\mathbf{G}(h)$ for $h > 1$, we imagine that the nodes at $(h-1)^{st}$ hop are expanded sequentially in order to count the unique nodes reachable from each of them. Let $g_m(\ell_2, h)$ denote the number of unique nodes accounted for at the mth step of the expansion. Then, in the $G(\ell_2, h-1)$th step, we would have accounted for all $G(\ell_2, h)$ nodes.

With $g_0(\ell_2, h) = 0$ for all ℓ_2 and h, we can write the following recurrence equation for $g_m(\ell_2, h)$, $h > 1$

$$g_m(\ell_2, h) = g_{m-1}(\ell_2, h) + D_{fr}(\ell_2, h|L) \frac{N_t - G^{(t)}(\ell_2, h-1) - g_{m-1}(\ell_2, h)}{N_t} \tag{19}$$

This equation is obtained as follows: At mth step, $N_t - G^{(t)}(\ell_2, h-1) - g_{m-1}(\ell_2, h)$ nodes out of a total of N_t nodes have already been accounted for; therefore, the number of new nodes added is simply the average node degree multiplied by the fraction of unreached nodes. This equation can be rewritten as

$$g_m(\ell_2, h) = a.g_{m-1}(\ell_2, h) + b \tag{20}$$

where the quantities a and b are independent of the index m and are given by $a = 1 - D_{fr}(\ell_2, h|L)/N_t$ and $b = [N_t - G^{(t)}(\ell_2, h)](1-a)$. A repeated expansion of equation (20) along with $g_0(\ell_2, h) = 0$, yields the following explicit recurrence for $G(\ell_2, h)$.

$$\begin{aligned} G(\ell_2, h) = g_{G(\ell_2,h-1)}(\ell_2, h) &= b\frac{1 - a^{G(\ell_2,h-1)}}{1-a} \\ &= [1 - a^{G(\ell_2,h-1)}][N_t - G^{(t)}(\ell_2, h-1)] \end{aligned} \tag{21}$$

Thus, $\mathbf{G}(h)$, and hence $\mathbf{G}^{(t)}(h)$ can be computed.

Next we adapt the above equations to obtain reachability to only undistinguished nodes. For this, we first need to compute the average undistinguished degree of nodes reached in h hops from a given level ℓ_2, given a total of L levels. We henceforth denote this as $D_{fr,u}(\ell_2, h|L)$ and also define the corresponding row vector $\mathbf{D}_{fr,u}(h|L)$. As with $\mathbf{D}_{fr}(h|L)$, we have $\mathbf{D}_{fr,u}(1|L) = \mathbf{D}_{at,u}(L)$, and for $h > 1$

$$\mathbf{D}_{fr,u}(h|L) = \mathbf{D}_{at,u}(L)\mathcal{R}_{h-1} \tag{22}$$

Next, we define the quantities $\mathbf{G}_u(h)$ and $\mathbf{G}_u^{(t)}(h)$ respectively, which have the same meaning as $\mathbf{G}(h)$ and $\mathbf{G}^{(t)}(h)$ except that reachability to only undistinguished nodes is being considered. These quantities can be determined by a slight tweak to the analysis presented above. Note that the paths to undistinguished nodes could well pass through distinguished nodes; therefore, if we are interested in paths of length h, we still proceed as in the last subsection up to length $h-1$ and then estimate the number of reachable undistinguished nodes in the last step. Let $g_{m,u}(\ell_2, h)$ denote the number of undistinguished nodes reached in the mth step for length h paths. Note specifically that the upper bound on m is $G^{(t)}(\ell_2, h-1)$ [instead of $G_u^{(t)}(\ell_2, h-1)$]. With $g_{0,u}(\ell_2, h) = 0$, equation (19) can be adapted as follows:

$$g_{m,u}(\ell_2, h) = g_{m-1,u}(\ell_2, h) + D_{fr,u}(\ell_2, h|L) \frac{L - G_u^{(t)}(\ell_2, h-1) - g_{m-1,u}(\ell_2, h)}{L} \tag{23}$$

This equation is again based on the observation that at mth step the number of new undistinguished nodes added is the undistinguished degree multiplied by the fraction of unvisited undistinguished nodes. The quantities $\mathbf{G}_u^{(t)}(h)$ and $\mathbf{G}_u(h)$ are again related as:

$$\mathbf{G}_u^{(t)}(h) = \mathbf{G}_u^{(t)}(h-1) + \mathbf{G}_u(h), \qquad 1 \leq h \leq H \tag{24}$$

along with the initial conditions $\mathbf{G}_u(1) = \mathbf{D}_{at,u}(L)$ (i.e., the undistinguished degree of the starting node) and $\mathbf{G}_u^{(t)}(0) = [0, 1, \ldots, 1]$. As in the last section, equation (23) can be solved explicitly to give:

$$G_u(\ell_2, h) = [1 - a^{G(\ell_2, h-1)}][L - G_u^{(t)}(\ell_2, h-1)] \tag{25}$$

where $a = 1 - D_{fr,u}(\ell_2, h|L)/L$.

4.3 Performance Metrics

Given the above analysis, we are now in a position to estimate the total request and response traffic arriving at a given node X. This is given by a superposition of the traffic generated by all nodes from which X can be reached in H or fewer hops (including requests generated by the node itself). A precise characterization of the traffic process is intractable; here we only compute its average rate. Let $N_{req}(\ell, H)$ denote the total number of requests reaching *undistinguished nodes* in at most H hops from a level ℓ node. Then,

$$N_{req}(\ell, H) = 1 + \sum_{h=1}^{H} G_u(\ell, h) \tag{26}$$

Turning this around, $N_{req}(\ell, H)$ gives the total request traffic processed by a level ℓ node. Thus, if λ_0 denotes the request rate for each undistinguished node, the total request traffic arriving at a level ℓ node, denoted $\lambda_{req}(\ell)$, is simply $\lambda_0 N_{req}(\ell, H)$.

The response traffic can also be computed similarly. That is, $\lambda_{resp}(\ell)$, the total response traffic arriving at a level ℓ node, is $\lambda_0 N_{resp}(\ell, H)$ where $N_{resp}(\ell, H)$ denotes the total number of responses processed at level ℓ node for a single request. The estimation of $N_{resp}(\ell, H)$ depends on how the responses are generated and forwarded back to the requesting node. We consider the following possibilities in this regard:

1. Every node generates a response (found or not found), and sends it to the requester directly (without going through any intermediate nodes). Then, $N_{resp}(\ell, H) = N_{req}(\ell, H)$.

Table 1: Reachability and traffic for a 100 node network

undist prob	no of hops	tot nodes reached	undist reached	responses per node	tot traf per node
	1	5.9	3.3	4.9	6.1
	2	55.2	44.5	103.6	146.5
0.05	3	99.1	85.8	235.2	320.5
	4	100	90	238.8	328.8
	5	100	90	238.8	328.8
	1	5.9	4.3	4.9	8.4
	2	34.3	23.8	61.7	82.3
0.50	3	91	73.9	231.7	304
	4	99.9	89.4	267.5	356.9
	5	100	89.6	267.7	357.3
	1	5.9	5.3	4.9	10.6
	2	28.6	22.6	50.3	73.6
0.95	3	76.7	63.8	194.6	258.4
	4	98.5	87.4	281.8	369.2
	5	99.7	89.3	287.8	377.2

2. Every node generates a response which travels back along the request path. In this case, a request going to a node that is h hops away, will require response processing at each one of the h intermediate nodes (excluding the response generating node where request processing and response generation are considered a single atomic operation). It follows that:

$$N_{resp}(\ell, H) = \sum_{h=1}^{H} h\, G(\ell, h) \tag{27}$$

Let S_{req} and S_{resp} denote the average service times of requests and responses. Then, the level ℓ node utilization is given by:

$$U(\ell) = \lambda_{req}(\ell) S_{req} + \lambda_{resp}(\ell) S_{resp} \tag{28}$$

As stated earlier, the queuing delays depend on the details of the arrival and service processes and are best determined via a simulation (which can also also issues such as limited queue depths, abandonments and retries). The above equation for utilization is useful for ensuring that the nodal utilization in the simulation model does not exceed a predetermined limit.

Now, we present some experimental results using the analytic model. We choose the parameters in this study as follows:

Table 2: Reachability and traffic for a 500 node network

undist prob	no of hops	tot nodes reached	undist reached	responses per node	tot traf per node
	1	6	3.6	5	6.2
	2	243.7	232.7	480.5	711.5
0.05	3	499.7	488.6	1248.4	1737
	4	500	490	1249.6	1739.6
	1	6	4.7	5	8.5
	2	95.7	84.2	184.3	264.6
0.50	3	483.5	465.1	1347.8	1812.4
	4	500	490	1413.9	1903.9
	1	6	5.8	5	10.7
	2	35.1	29.1	63.2	91.7
0.95	3	163.5	137.1	448.3	582.4
	4	405.7	367.7	1417.2	1782.7

1. The network initially has 10 distinguished nodes, each of which is connected to 4 others.
2. Each newly arriving node connects from 1 to 4 nodes with equal probability. (Thus, the average connectivity is 2.5).
3. The β function is assumed to be identically 1.
4. We consider two network sizes N_t of 100, and 500 nodes.
5. For each network, we consider 3 undistinguished node connection probabilities q_u of 5%, 50%, and 95%.

Tables 1 and 2 show overall averages on reachability, response traffic per node and total traffic per node for 100 and 500 node graphs respectively. For reachability, the tables list the average number of all nodes reached from an arbitrary node (column 3), and average number of undistinguished nodes reached from an undistinguished node (column 4). The response (and hence total) traffic computation assumes that the responses trace the request path backwards. These results (and others not shown here) point to a number of interesting conclusions about the network under consideration.

1. Even with a very limited immediate connectivity per node of 2.5, 4 hops can cover nearly all the nodes in most cases.

2. Distinguished nodes provide a "short cut" between nodes; therefore, with a low value of "undist prob" (or q_u), the overall reachability increases very fast with the number of hops.

3. An interesting traffic parameter is the total traffic divided by the network size. With 4 hops, this metric varies between 3.3 and 3.8. That is, if each request returns a response, the nodal capacity must increase 3-4 times as fast as the network size.

5 Simulation-Based Evaluation of P2P Networks

Our simulation model is based on the random graph representation of the P2P network. The random graph model of P2P networks is attractive in that it concisely represents a large number of network topologies either as a result of dynamic changes to the topology or as simply representing a collection of networks with certain characteristics. Random graph models are good for mathematical modeling, but unfortunately their simulation becomes difficult to handle because of a large number potential instances. In this section we start by discussing possible methods for dealing with this issue.

5.1 Simulation of Random Graphs

Below, we discuss two approaches to random graph simulation – constrained connectivity vs direct simulation.

5.1.1 Constrained Connectivity

Apart from a large number of instances, a fundamental problem in simulation is how to handle multiple network instances. Note that one cannot simply simulate each instance separately and then take the average of the results. For example, an instance where a certain node happens to be very richly connected would result in a high enough load on that node so as to violate the stability condition. This issue can be handled via a parallel simulation of all considered instances. That is, the simulator takes as input, all the instances of interest, and then randomly selects an instance for each transaction. In order to maintain consistency, it is necessary to retain the instance ID as a part of transaction throughout the transaction life.

We propose limiting the number of instances by constraining the connectivity during graph construction. The constraints must ensure that high-probability instances are not discarded. One way to do this is

by forcing the degree of each node to be within a narrow band around the average degree. This is possible only if at least the expected degree of each node can be computed efficiently for each node addition. This is true for the random-graph model we described above; in more complex cases, expected degree can be obtained reasonably reliably via a separate network traversal simulation.

Let D_{in} denote the average degree of node i when the entire graph contains n nodes. Let $L < 1$ and $H \geq 1$ denote low and high degree multipliers of interest. That is, we want to keep the actual degree between $L.D$ and $H.D$. Let d_{in} denote the actual degree of node i when $(n+1)^{\text{st}}$ is added to the graph. All nodes with $d_{in} \geq H.D_{in}$ are ineligible for connection at this point, whereas if node i has $d_{in} < L.D_{in}$, it is chosen immediately for connection. (If several nodes satisfy this property, the one with smallest index is chosen.) Otherwise, the normal probabilistic choice applies among all eligible nodes.

Note that if we set $L = H = 1$, this construction yields an "average case" instance of the original random graph. Using just that instance makes for an efficient simulation that will preserve properties related to nodal degrees (e.g., average traffic at each node). However, the simulation would yield very small variation in basic quantities and thus could severely underestimate queuing delays, drop probabilities, and the like.

As the spacing between L and H increases, the number of distinct instances blows up very fast, and it is not possible to simulate all instances. Instead, we take the approach of randomly choosing a reasonable number of instances (a few 100's) and use only those for simulation. The accuracy would obviously depend on L, H, degree distribution, and the number of instances chosen. Note that a careful sampling will pick more frequently occurring instances with a higher probability. That is, during the parallel simulation we don't need to worry about the probabilities with which those instance appear.

5.1.2 Direct simulation of probabilistic model

This approach does not consider individual graph instances, instead, it uses the random-graph directly during the simulation. The technique requires pre-computation of the relative probability q_{ij} of having an edge from node i to node $j > i$. This can be calculated using the random graph model presented earlier. With this, a query received from node k at node i is sent to node j with probability $q_{ij}/(1-q_{ik})$. This virtual topology for the query is used to return responses as well. In this case, a single simulation automatically visits various instances in the correct proportion. However, this advantage can be a drawback too. Since there is no explicit control over which instances are visited, reliable results take a very long time. The more

serious problem is the lack of fixed topology, which makes complex inter-relationships difficult to handle. For example, if transaction A were to set a flag or timer at node X and transaction B is expected to reset it, both transactions must always pass through node X. This will require a large amount of global information and coordination. In view of these difficulties, we did not follow the direct simulation approach.

5.2 Overview of SimP^2

In order to study the performance of the network described above, we have implemented a tool called SimP^2 (Simulating P2P). This tool provides the following capabilities:

1. Generation of desired number of instances of the graph.
2. Determination of file parameters (size, popularity, number of copies, etc.)
3. Assignment of files to network nodes.
4. Search query and response handling at each node.
5. File transmission and reception.
6. Handling of time-outs, dropped requests/responses, file caching, etc.

The simulation tool is written in C/C++ and uses the Sim++ package as simulation engine. It provides a very flexible input interface. This input interface is actually same as for our Geist e-commerce traffic generation tool [8]. A detailed detailed description of the tool per se is beyond the scope of this paper; instead, we only describe some of the key input parameters needed by the simulation model and our choices for them for producing the results shown in section 6. Although these parameter settings are chosen with some consideration for real-life, every P2P environment is so different that it is impossible to claim them to be "representative" in any way. In any case, *the results included in this paper are merely illustrative and should be regarded as such.*

5.3 Graph Characteristics

The random-graph model used for the experiments was built as follows: the graph has a total of 100 nodes, of which 10 are distinguished. Each distinguished node is connected to exactly two other distinguished

nodes. Each newly arriving undistinguished node connects to exactly 4 other nodes (no distribution used here, though the model supports it). Each of these connections goes to a distinguished node with a probability of 0.5. For simplicity, the weighting function $\gamma(i,n)$ was chosen to be identically 1 for all i and n. The major consequence of this choice is that the average degree of a node goes down very slowly (i.e., as $1/j$ with node number j). That is, the network used in these results was not very highly nonuniform.

As discussed in section 5.1, the simulation requires generating instances under constrained connectivity. We found that using $L = 0.5$ and $H = 1.5$ cuts out a large number of low-probability instances without affecting the accuracy significantly. We also found that using 100 instances in the parallel simulation is adequate, although this clearly is a result of the choice of the γ function that leads to low variability.

5.4 Request Generation

We represent the request generation (or arrival) process at each peer by an on-off process with constant request generation rate during the on period. This is a widely accepted user behavior model and allows approximate emulation of a self-similar [15] with a given Hurst parameter. Although P2P traffic characteristics are not well understood, long-range dependence is likely to be observed (much like the request level web traffic as shown in [9]). Let X_0 and X_1 denote the off and on periods. Assume that X_0 and X_1 are iid with the Pareto distribution $P(X_i > x) = (x/T_i)^{-\alpha}$ with $1 < \alpha < 2$, $x \geq T_i$, $i = 0, 1$, where T_i is the minimum value of the off/on period and α is the decay rate of the Pareto distribution. It is shown in [15] that the Hurst parameter of the combined arrival process is given by: $H = (3 - \alpha)/2$. We assume a mean on/off period of 30 secs and $H = 0.8$.

Each requesting node runs a timer T_u, currently assumed to have a triangular distribution in the range (0.75, 3.5) secs with a mean of 2.0 secs. This distribution attempts to approximately model the kind of timeout distribution observed in telecommunications networks – it is certainly possible to use a more accurate empirical distribution from real measurements. Any responses arriving after the timeout are ignored (or dropped). Currently, the tool has no provision of retrying timed out requests, although such a feature is trivial to add.

5.5 File Characteristics

We start with file-size distribution. We use a 2-segment model with the following characteristics:

1. Uniform distribution in the small-size range, chosen as 400 bytes to 4 KB.

2. Pareto distribution with a minimum value of 4KB and mean of 40 KB, which gives $\alpha = 1.11$.

We note here that 40 KB mean is typical for web pages, but too small for MP3 files. The Pareto tail is a well known characteristic, but the uniform initial part is somewhat arbitrary.

Note that the file size distribution provides only the storage characteristics; for the access characteristics, we introduce the notion of a file category, which provides a link between file size and its popularity. We use 9 categories, forming a geometric series with next value $\sqrt{2}$ times that of current entry. Thus category1 ranges over (400B, 1265B) whereas category9 ranges over (4MB,12.65MB). We choose the file access distribution over categories as a unimodal function with a geometrically decaying tail. The chosen distribution is $(0.07, 0.14, 0.2018, 0.20, 0.14, 0.098, 0.0686, 0.048, 0.0336)$.

A search in a P2P network would typically result in multiple hits, depending upon the search expression and the distribution of the file keys. A direct modeling of this requires choosing many distributions and makes reliable results harder to get. Therefore, we don't delve into the details of the search; instead, we control the "hit" characteristics by specifying the number of *copies* for each file and how those copies are distributed at various nodes. The number of copies may depend both on file category and on the type of node (distinguished vs. undistinguished). The results, however, assume a single triangular distribution with minimum, maximum and mode values as $C_{\min} = 1$, $C_{\max} = 20$, and $C_{\mathrm{mode}} = 5$. Again, one could easily use a different distribution.

The copies of a file are assigned to successive nodes at a fixed distance so as to distribute them evenly across the network. An offset is used for successive files so are to avoid bunching up of copies. Also, no node is assigned more than one copy of a file to a node. The specific assignment algorithm is given below.

```
n_copies = triangular_rv(1, C_max, C_mode);        // Generate copies
distance = n_nodes/n_copies;          // Distance for copy allocation
offset = 1 + n_nodes/no_files;        // If too few files, get an offset
tot_offset = (tot_offset + offset) % n_nodes;
node_no = tot_offset;                 // Node for assigning the first copy
for( copy_no = 0; copy_no < n_copies; copy_no++) {
   assign_file( node_no, file_no, size);
   node_no = (node_no + distance) % n_nodes;
   if(copy_no < n_copies-1
```

```
        && node_no == (tot_offset+wraps)% n_nodes) {
    node_no = (node_no + 1) % n_nodes; wraps++;
    }
}
```

A specific file is now identified by the triplet: (category, file_no, copy_no) where file_no is a unique id (e.g., sequence no) of files in a category. This allows following capabilities:

1. Unique searches specified by the file-id triplet.
2. Non-unique searches specified by (category, file_no).
3. Replication control and fault-tolerant operation.

5.6 Query and Response

Each query specifies a file (category, file_no) with given access characteristics. Shown results do not specify copy_no, which means that multiple hits are possible for each query. A query percolates for upto h *hops*. If a query arrives at a node more than once, it is not propagated. Each node has a finite queue and any query coming into a node will be dropped.

Each query reaching a node generates found/not found response, which travels backwards along the search path. All responses that arrive at the requesting node after the timer T_u has expired are ignored. The model provides the ability to cull requests or responses after some user settable time ($< T_u$) so that it is possible to minimize resource spent on queries whose originator is likely to have abandoned before the response is received.

File retrieval is done by randomly choosing one of the positively responding nodes. For the results considered here, requested file(s) are obtained directly (i.e., do not follow the response path). A retrieved file may be optionally cached at the requesting node using a LRU scheme. Since the peers may join/leave the network and the documents may change, it is important include its impact on the caching. Each document can be assigned a time to live (TTL), after which it is flushed from the cache. Also, the entire cache is also flushed occasionally, which represents a peer leaving and a statistically identical peer joining the network. The number of cycles before cache flushing is assumed to be Zipf with minimum value of 30 secs, maximum value of 120 secs, and $\alpha = 1.0$.

5.7 Query/Response Service

Each query and response needs service at each node visited. File transfer needs service on both ends and has two parts: (a) a basic service time (independent of file-size, given by a distribution), and (b) a file-size dependent component. Each node implements 3 priority levels for efficient processing of transactions of different types: Low for queries, Medium for file transfers, and High for response processing. Such an assignment ensures that the work already in the system has a better chance of completing than the new incoming work.

As indicated earlier, overall queue size at each node is constrained to avoid long queuing delays. For a finer control, it is possible to do a priority level based throttling, but this is currently not implemented. The link service time also has two components: (a) a basic service time (independent of transfer size, given by a distribution), and (b) a size dependent part determined from link bit rate. The link bit rate taken as 3 KB/sec (a estimate of real-life rate on Internet). Assuming that P2P traffic $\ll$ total traffic, it is reasonable to consider links as delay servers. Note that a single link in the model represents the entire path between two peers — currently, no attempt has been made to model the physical topology of the network and the mapping of peers to the physical network.

6 SimP^2 Results and Analysis

In this section, we illustrate the uses of the Tool by examining a number of scenarios and their performance. The base scenario considered for this is one with 3 hops per search, a filestore size of 16M, local file-caching and flushing enabled.

We start with the effect of the number of hops allowed per search. Figure 1 and Table 3 show the simulation results as the number of hops are increased from 1 to 4. From the figure, it is clear that the number of searches that find a file successfully increases to a maximum value when the number of hops equals 2 or larger. However, this data only counts searches that found at least one copy of the needed file. We found that the number of copies found per search continues to increase as the number of hops increase from 2 to 3. Table 3 shows more detailed data in terms of the average utilization & queue length at distinguished & undistinguished nodes. Additionally it also shows data on the number of overall responses received per request including those that arrive after the user's patience threshold has expired. One of the main observation from the results is that the average utilization at the distinguished node reaches a

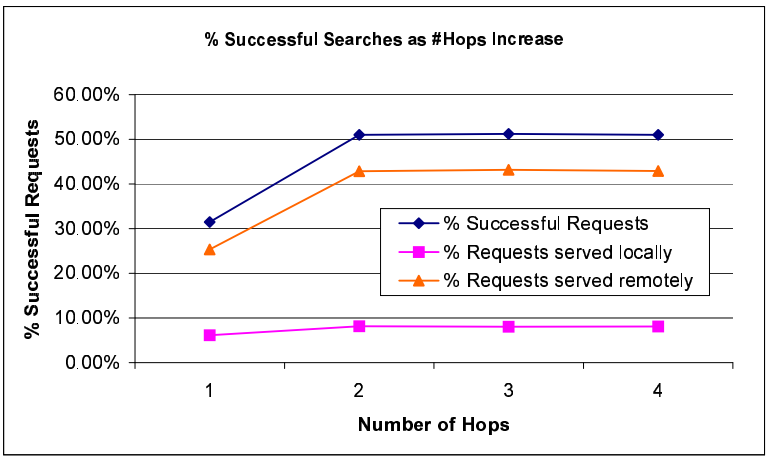

Figure 1: Success Rate vs. Number of Hops

	Hops = 1	Hops = 2	Hops = 3	Hops = 4
Node Utilization (Dist Node)	8.1%	44.3%	86.1%	96.1%
Node Utilization (Other Nodes)	5.2%	25.9%	50.7%	59.0%
Queue Length (Dist Nodes)	1.013	2.35	14.646	30.318
Queue Length (Other Nodes)	1.046	2.264	5.176	6.977
Num Responses Per Request	6.59	51.07	84.17	80.56
% Unexpired Responses	97.22%	95.55%	97.36%	97.66%
% Expired Responses	2.78%	4.45%	2.64%	2.34%
Num Dropped Msgs Per Request	0.00	0.36	7.24	27.77
% Successful Requests	31.47%	50.99%	51.24%	51.03%
% Requests served locally	6.1%	8.14%	8.04%	8.11%
% Requests served remotely	25.3%	42.85%	43.19%	42.92%

Table 3: Effect of Number of Hops Allowed

significant level (> 80%) at 3 hops. For this case, the average utilization at the undistinguished nodes reaches around 50%. Another interesting result is the sharp increase (beyond 2 hops) in the number of dropped messages basically due to the increase in the queue length at the distinguished nodes. Finally, we also found the number of hops allowed had minimal impact on the local vs remote hit rate for searches.

We next studied the effect of caching on the performance characteristics of the network. We basically simulated three scenarios: (1) caching of all files, (2) caching files of a limited size (<40K) and (3) no caching. Table 4 shows relevant simulation results for these configurations. As the amount of caching reduces, the average utilization and queue lengths on the nodes increase moderately. Additionally, the number of responses per request also increases as the amount of caching is reduced. The reason for this is that if a request finds a copy of a file in the local cache at the node, it does not propagate the search to other

	Cache All	Cache < 40K	No Caching
Node Utilization (Dist Node)	86.1%	89.8%	93.0%
Node Utilization (Other Nodes)	50.7%	49.9%	52.4%
Queue Length (Dist Nodes)	14.646	18.252	21.497
Queue Length (Other Nodes)	5.176	3.956	4.179
Num Responses Per Request	84.17	89.32	93.90
% Unexpired Responses	97.36%	99.72%	99.61%
% Expired Responses	2.64%	0.28%	0.39%
Num Dropped Msgs Per Request	7.24	4.48	5.49
% Successful Requests	51.24%	57.67%	28.79%
% Requests served locally	8.04%	5.32%	0.00%
% Requests served remotely	43.19%	52.35%	28.79%

Table 4: The Impact of File Caching

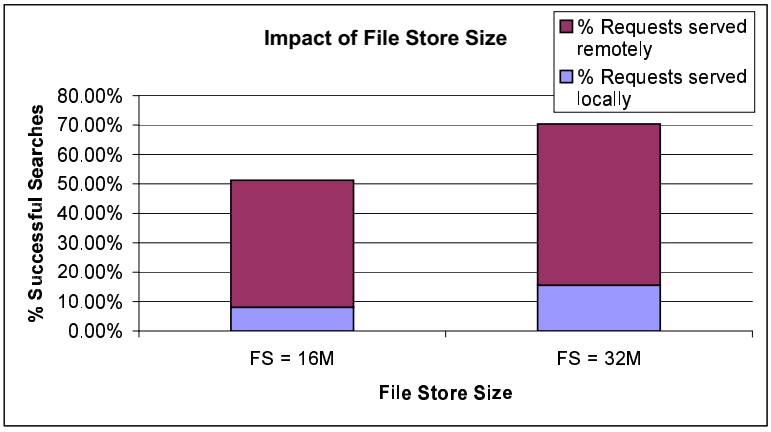

Figure 2: Effect of Increasing File Store Size

nodes. As a result, if caching is enabled, the higher probability of finding the file in the local cache reduces the number of responses it would receive. Of all successful searches, roughly 16% found the files locally in the all-caching scenario. If only files smaller than 40K bytes are cached, then this local hit percentage reduces to less than 10%. Finally, if caching is disabled, the number of successful searches reduces by almost a factor of 2.

We next studied the impact of increasing the size of the file-store from 16M to 32M. In doing this, we found that the overall success rate for the searches increases significantly (from 50% to 70%). In particular, the rate at which the requests were found locally increases from 8% to 15%. As a result, the average utilization on the nodes in the network reduces as the file-store size increases.

In all previous simulation runs, we assumed that the distinguished nodes had the same computing

	Base	DNodePow*2
Node Utilization (Dist Node)	86.1%	51.4%
Node Utilization (Other Nodes)	50.7%	49.5%
Queue Length (Dist Nodes)	14.646	3.522
Queue Length (Other Nodes)	5.176	5.463
Num Responses Per Request	84.17	82.48
% Unexpired Responses	97.36%	97.87%
% Expired Responses	2.64%	2.13%
Num Dropped Msgs Per Request	7.24	9.77
% Successful Requests	51.24%	51.00%
% Requests served locally	8.04%	7.96%
% Requests served remotely	43.19%	43.04%

Table 5: Impact of Doubling Compute Power at Distinguished Nodes

power as the undistinguished nodes. However, it is worthwhile to have a smaller set of powerful distinguished nodes & a large set of mediocre undistinguished nodes. The effect of doubling the computing power at the distinguished nodes is shown in Table 5. The main result is that the utilization of the distinguished nodes reduces significantly (86% to 51%) and so does the queue lengths (from 15 to 4).

Given the assumptions we made for the P2P network, an overall summary of the key observations are as follows: (1) limiting the numbers of hops to about 3 seems to be reasonable (even though the degree of connectivity we chose was rather small), (2) Increasing the number of hops beyond about 3 only clogs the network w/ search requests that might actually get dropped ultimately (3) As compared to the overall file-set size, the cache size needed per node becomes rather small for a medium-sized network and (4) the request/response queue sizes at each node can be an important parameter if the traffic tends to arrive in a highly bursty manner. In our experiments with SimP^2, we actually also studied the impact of increasing queue length, the impact of turning cache flushing ON/OFF and the impact of message expiry in the network. The detailed results, however, are not enumerated here due to lack of space. In general, we found that disabling both the caching & flushing option increases the search hit ratio considerably since files never get replaced and lost in the network. Additionally, we also observed that culling the expired messages in the network make little difference to the performance characteristics of the chosen network.

7 Summary and future work

This paper presented analytic and simulation based approaches to studying the performance of generic peer-to-peer (P2P) file-sharing networks. The underlying P2P network is modeled via a random graph which certain characteristics similar to existing Gnutella networks, although the simulator can use networks generated using other mechanisms as well. A key contribution of the paper is a discussion of some ways for simulating the random-graph model without too much computational expense. The paper also discusses some sample performance results by varying several characteristics of the network and the content location mechanism. The main value of the example is not so much in results per se but in the discussion surrounding the choice of various parameter values.

While the first version of SimP^2 was discussed in this paper, it can be extended to study a variety of other issues of interest for P2P sharing networks. These include (a) intelligent, automatic propagation of files through the network based on recent access history, (b) explicit modeling of user retry behavior in case of user timeouts, (c) explicit modeling of dynamic changes to the network, and (d) a more explicit representation of the underlying physical network and the mapping between this and the virtual P2P network.

SimP^2 can be further extended to study many other scenarios as well. In a large network, many responses may be duplicates or otherwise redundant. A mechanism for early duplicate (or redundancy) detection could make the searches considerably more efficient. For example, schemes that batch responses to weed out duplicates at intermediate nodes might be of interest. Another interesting idea is to meld together the blind-search approach of Gnutella and the distributed hash-table (DHT) based ordered search approach in order to come up with a scheme that can represent a broader spectrum of content lookup.

Finally, SimP^2 is intended to be released in an open-source form so that other researchers can enhance it and do further interesting studies with it.

References

[1] E. Adar and B. A. Huberman, “Free riding in Gnutella”, Xerox PARC report, Oct 2000, available at www.parc.xerox.com/istl/groups/iea/papers/gnutella.

[2] M. Castro, P. Druschel, et. al., “Exploiting Network Proximity in Distributed Hash Tables”.

[3] I. Clarke, “A Distributed Decentralized Information Storage and Retrieval system.” M.S. Thesis, Division of Informatics, Univ of Edinburgh, UK, 1999.

[4] A. Chervenak, E. Dealman, et. al., “Giggle: A Framework for Constructing Scalable Replica Location Services”, Proc. of the IEEE Supercomputing 2002.

[5] R. Iyer, V. Tewari, and K. Kant, “Overload control mechanisms for web servers”, Performance and QoS of Next Generation Networks, Nagoya, Japan, Nov 2000, pp 225-244

[6] M.A. Jovanovic, F.S. Annexstein and K.A. Berman, “Scalability issues in large peer to peer networks – A case study of Gnutella”, Tech. Report, Univ. of Cincinati, 2001.

[7] K. Kant, R. Iyer and V. Tewari, “A framework for classifying peer-to-peer Technologies”, 2nd IEEE/ACM Intl. Symposium on Cluster Computing and the Grid, May 21-26, Berlin, Germany. (available at http://kkant.ccwebhost.com/download.html)

[8] K. Kant, V. Tewari and R. Iyer, “Geist: A generator of e-commerce and internet server traffic”, Proc. of ISPASS, Tucson, AZ, Nov 2001, pp 49-56.

[9] K. Kant and Y. Won, “Server Capacity Planning for Web Traffic Workload”, IEEE transactions on knowledge and data engineering, Oct 1999. pp731-747.

[10] R. Kumar, P. Raghvan, et. al., “The web as a graph”, ACM POD conference, Dallas, TX, 2000.

[11] C. Moore and M. Newman, “Epidemics and percolation in small-world networks”.

[12] G. Pandurangan, P. Raghvan and E. Upfal, “Building low-diameter P2P networks”.

[13] Matei Ripeanu, “Peer-to-Peer Architecture Case Study: Gnutella”, Proc. of 2001 Intl Conf on Peer-to-peer Computing (P2P2001), Linkoping Sweden, 27-29 August 2001.

[14] A. Vahdat, J. Chase, et. al., “Self-organizing subsets: From each according to his abilities, to each according to his needs”.

[15] W. Willinger, M.S. Taqqu et al., “Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN traffic at the source level”, Proc of SIGCOMM 95, pp100-113.