Performance and Design of P2P Networks for Efficient File Sharing

Amit Kapur, Natarajan Gautam
Harold and Inge Marcus Department of Industrial and Manufacturing Engineering
Penn State University
University Park, PA 16802

Richard Brooks
Applied Research Laboratory
Penn State University
University Park, PA 16802

Suresh Rai
Electrical and Computer Engineering
Louisiana State University
Baton Rouge, LA 70803

Abstract
Peer-to-peer (P2P) technology enables users to share resources (files or mobile codes) directly with each other instead of requiring a large-capacity server to store the files. However, there are several small-capacity servers called indexes in these networks that index information regarding which peer has what files. In this paper we conduct a performance analysis to obtain response time, jitter and loss probability expressions analytically. We use the results in a mathematical programming problem to determine the optimal number of indexes and time out value. We consider three topological models for the network of peer nodes: Erdös-Rényi, small world, and scale-free.

Keywords:
Peer-to-peer, mobile code, delay, jitter and loss probability.

1 Introduction
This paper aims at creating a peer-to-peer (P2P) network [14] infrastructure that supports mobile code [12] technology and is capable of adapting to malicious, possibly catastrophic events. Mobile code technology enables transmission and execution of programs between networked nodes. P2P networks distinguish themselves from traditional client/server or master/slave networks in that there is neither a central point of control nor centralization of data. They potentially support adaptation by allowing network structure to evolve. To realize our goal of creating adaptive network services, we are creating a P2P infrastructure from existing mobile code daemons [3].

The two most widely known P2P implementations are Napster [13] and Gnutella [7]. Napster is a file-sharing network with only one central index. This index contains a database of users and their files. When a user connects to Napster, a list of files available on the user's machine is added to a central index. When the user requests a specific file, a list of participating machines containing the file is returned. The file can be retrieved from any machine on the list. This is an efficient architecture. File names and machine addresses contain tens of bytes. Files being exchanged typically contain megabytes of data. The large data transfers occur between machines chosen virtually at random. This tends to spread data traffic evenly throughout the Internet. On the other hand, its survivability is poor as a single failure or a court order can stop the entire network by switching off the central index.

Gnutella offers a radically different approach [7]. It is fully distributed with no single point of failure. Each node has an index of its own files. File discovery is performed by flooding the network with request packets. There appears to be serious scalability issues with this approach [7, 18]. Another application, Freenet [11] addresses this scalability issue. On the other hand, Gnutella has interesting survivability characteristics. To stop the Gnutella service, it would be necessary to stop every node on the Internet running Gnutella. Other P2P networks like Chord [4], CAN (Content Addressable Network) [15] and Pastry [16] are also decentralized networks with different routing schemes. They all scale very well and are fault tolerant.

On the same lines as Napster and Gnutella, in this paper we consider a P2P network (with n nodes and a arcs) where the nodes represent computers, work stations, or servers that store mobile code (analogous to audio files in Napster) and the arcs represent physical or logical connectivity between two nodes. However, one of the main differences is that we use a protocol with a "time-out" parameter for abandoning a search. This paper studies the question: what is an appropriate number of indexes for a P2P network? We analyze this problem in terms of performance, scalability, and survivability. We consider Napster and Gnutella as the extreme cases of a continuum. The number of indexes varies in the range of 1 to n. Another question studied in this paper is, what is the appropriate time-out value to use?

The inputs to the problem studied in this paper are number of nodes, average arc degree and the type of network. They are inputs over which we have no control. Although we do not state the link and computing (i.e. node) speeds as inputs explicitly, we assume their values are known too. The controllable parameters are: number of indexes and time-out value. We choose them to optimize performance measured in terms of delay, jitter and loss. We assume nodes fail randomly and independently. Nodes fail for several reasons including denial-of-service attacks.

The layout of the paper is as follows: In Section 2, we provide background on networks and their topologies. In Section 3, we provide a request-response process model for the index and client nodes. In Section 4 we present analytical results for the three performance measures: *delay*, *jitter* and *loss probability*. They are used to formulate and solve an optimization problem that selects the appropriate number of indexes and time-out value in Section 5. We conclude the paper and give directions for future work in Section 6.

2 Types of Networks

In this paper we consider three classes of randomly generated graphs [2]:

Random graphs or Erdös-Rényi graphs – Consider a set of n nodes and a probability p^* that an arc exists between any two nodes. Node degree distribution follows a Poisson distribution as n approaches infinity [2]. When these graphs have a single connected component, the average number of hops between nodes grows proportionately to the logarithm of the number of nodes [1].

Small world graphs – A class of graphs with two properties: (*i*) average number of hops increases with the number of nodes in the same order of magnitude as random graphs, and (*ii*) there is a significant clustering of nodes (i.e. many nodes have multiple neighbors in common). For this class, we use the connected caveman model described in [21]. A set of fully connected components is constructed. One arc at random is rewired in each fully connected component so that the set of components is connected in a cycle. A small set of arcs in the resulting structure is rewired at random. The node degree distribution depends on the number of arcs re-wired.

Scale-free graphs – A class of graphs where the probability $p(k)$ that a node has degree k follows a power law distribution $p(k) \alpha k^{-\gamma}$, where γ is a constant. Empirical studies have shown that many real-world systems including the Internet [1, 6] and Gnutella [6] have this property. The average number of hops of scale-free networks grows more slowly with respect to the number of nodes than for Erdös-Rényi graphs.

3 Request-response Process

We divide the network into approximately equal sized groups per index. Each index node has a database of the files available from nodes in its group. The source node requests a particular document from its index node and starts a timer. An index node can also be a source node, in which case the travel time to the first index node is zero. If the index node knows the location of the requested document (i.e. the document and source node are in the same group), it informs the source node of the location of the node containing the document (*destination node*). Otherwise, the index node sends the location of the next nearest index node to the source node. In this way, the source node queries index nodes for the document until all indexes have been searched. If the destination node is identified, the source node requests the document from it. If a transfer is completed before the timer expires then it's a success, otherwise the request is lost. A request is also lost if an index node with document information is down or the destination node is down. In case of a loss, the source node does not retry the same request.

For the analysis in Section 4, we use the expected value (and variance) of the number of hops between any two nodes in the network. The result is derived in an earlier paper [9]. In that paper, the results are compared against the hop distribution computed using Dijkstra's Shortest Path Algorithm [5] and they were extremely accurate. We use μ and σ to denote the mean and standard deviation of the number of hops in a network.

4 Performance Analysis

In a preliminary paper [8], we considered a scenario where requests from peer nodes arrived infrequently so that there were no queues at any of the nodes. Now we consider a scenario with frequent arrivals so that there are queues formed at the nodes. In this section we present performance measures such as *delay*, *jitter* and *loss probability* for such a scenario. In Section 4.1, we model queues at index nodes and destination nodes assuming no node failures and no lost requests. We incorporate time-outs in Section 4.2. In Section 4.3 we compare the analytical results against simulations. The results for the moments of the time to retrieve a document (T) are derived in terms of the following parameters: number of nodes n, average arc degree k, standard deviation of number of nodes s, communication speed (link speeds) l, processing (computing or node) time d, number of different documents or code modules mo, average size of requests q, size of responses B, number of indices I, mean number of hops to index i (as a function of n, k and s) $\mu_i(n,k,s)$, standard deviation of number of hops to index i (as a function of n, k and s) $\sigma_i(n,k,s)$, starting from index 1 the mean number of hops to eventually reach index i (as a function of n, k and s) $\alpha_i(n,k,s)$, and, starting from index 1 the standard deviation of number of hops to reach index i (as a function of n, k and s) $\beta_i(n,k,s)$.

4.1 Performance model with no failures and no time outs

We initially assume an infinite time-out and frequent request arrival so there is resource contention (i.e. queuing) among requests and they may have to be stored in a queue at the indexes and destination nodes. Queues at index nodes and destination nodes were approximated as M/G/1 queues with Poisson inter-arrivals, Uniform service times and a single server. Using the M/G/1 model, the following performance measures are easily obtained: $E[W_i]$, the average waiting time a randomly arriving request spends in the queue of index node i plus the expected time to search the index node, and $E[W_n]$, the average waiting time a randomly arriving request spends in destination node queue plus the expected time to search the destination node. Thereby, the analytical expressions for delay and jitter of the response time (i.e. time between when a request is sent and a document is retrieved) can be derived in the following manner (details in [10]). The expected delay with queues at the index nodes and destination nodes is:

$$E(T) = \frac{2*q}{l} * \sum_{i=1}^{I} \mu_i(n,k,s) + \sum_{i=1}^{I} E(W_i) + E(W_n) + \left(\frac{q}{l} + \frac{E(B)}{l}\right) \sum_{i=1}^{I} \alpha_i(n,k,s) \qquad (1)$$

The jitter (defined as $\sqrt{Var(T)}$) with queues at index nodes and destination nodes can be derived using:

$$Var(T) = E(T^2) - [E(T)]^2 \qquad (2)$$

For equation (2), $E(T)$ can be obtained from equation (1). In order to obtain $E[T^2]$, we define T_i as the random variable denoting the time to retrieve the document conditioned upon the document indexed in index i, therefore

$E(T^2) = \sum_{i=1}^{I} \left(Var(T_i) + [E(T_i)]^2 \right) \frac{1}{I}$, which can be computed using the following equations:

$$Var(T_i) = \frac{4\alpha_i^2(n,k,s)q^2}{l^2} + \left(\frac{d*mo}{I}\right)^2 + \frac{1}{12}\left(\frac{d*mo}{I}\right)^2 + E(I)Var(W_q) + \frac{\beta_i^2(n,k,s)q^2}{l^2} + \left(\frac{d*mo}{2n}\right)^2 + \frac{1}{l^2}\left\{Var(B)*\beta_i^2(n,k,s) + Var(B)*\alpha_i^2(n,k,s) + E(B)^2*\beta_i^2(n,k,s)\right\}$$

$$E(T_i) = \frac{2q\mu_i(n,k,s)}{l} + E(W_I) + 0.5E(W_I) + \frac{\alpha_i(n,k,s)q}{l} + 0.5E(W_n) + \frac{\alpha_i(n,k,s)E(B)}{l}$$

4.2 Finite time-out case

Section 4.1 assumes that the time-out value is infinite. This means that there is no loss of requests. If the time-out value was finite, then requests would be lost if the time to get a response exceeds the time-out value. Since the delay is a sum of a large number of independent random variables, we can approximate using central limit theorem that T ~ Normal [E(T), Var(T)]. Now the probability that time-out occurs (in terms of θ, the finite time-out value) is

$$P(T > \theta) = \varepsilon = 1 - \Phi\left[\frac{\theta - E(T)}{\sqrt{Var(T)}}\right]. \qquad (3)$$

Then, the response time given that document is retrieved before time-out ($E(T_\theta)$) will be

$$E(T_\theta) = \frac{\int_{-\infty}^{\theta} x f(x) dx}{(1-\varepsilon)} \tag{4}$$

where $f(x)$ is the normal probability density function. Then the variance will be

$$Var(T_\theta) = \left\{ \frac{(1-\varepsilon) \int_{-\infty}^{\theta} x^2 f(x)\, dx - \varepsilon (E(T_\theta))^2}{(1-\varepsilon)^2} \right\} \tag{5}$$

The loss probability will be

$$L_\theta = 1 - p_d [1 - (1-p)^{co}](1-\varepsilon) \tag{6}$$

4.3 Comparison of Analytical and Simulation Models

In this section we compare the analytical models derived in Sections 4.1-4.2 with simulation results to verify the analytical results. This is important because we will use analytical models to compute the optimal number of indexes and time-out value in Section 5. We consider both the cases: infinite and finite time-out values. For all the 3 network types, the number of nodes was taken as 100, average arc degree as 3, the number of indexes as 5 and the arrival rate of requests into the system was taken as Poisson with 1 request in 4 seconds. The link speeds were taken standard 56Kbps, the average query size as 60 bytes, the document size as Normally Distributed with a mean of 100Kb and standard deviation of 10Kb and the node speed was taken as 0.001 seconds per document. Also 10 replications were taken for all the simulation runs. Also this simulation was done in Arena as Arena simulates queuing models well. The results are summarized in Table 1 (for infinite time-out value) and Table 2 (for time-out value of 50 seconds).

Table 1: Comparison of Analytical and Arena Simulation results for infinite time-out value.

	Analytical		**Simulation**	
	Delay	*Jitter*	*Delay*	*Jitter*
Erdös-Rényi Graph	30.0351	12.7749	27.61	13.452
Small World	32.4967	13.3071	30.005	13.694
Scale-free	29.3821	12.8987	27.086	13.41

Table 2: Comparison of Analytical and Arena Simulation results for time-out value of 50 seconds.

	Analytical			**Simulation**		
	Delay	*Jitter*	*Loss Prob.*	*Delay*	*Jitter*	*Loss Prob.*
Erdös-Rényi Graph	28.8205	10.8854	0.06	25.317	10.882	0.06071
Small World	30.0634	10.9015	0.095	27.076	10.717	0.07884
Scale-free	27.8087	11.1569	0.0508	24.801	10.946	0.05645

From the above two tables it can be seen that results from analytical and simulation models are fairly close. The difference in the two results is due to a few approximations that we made in deriving the analytical models. One approximation is the Central Limit Theorem we used to incorporate finite time-out value. We also approximated the arrival process of requests at index nodes as Poisson (which will not be the case since the output from an M/G/1 is non-Poisson).

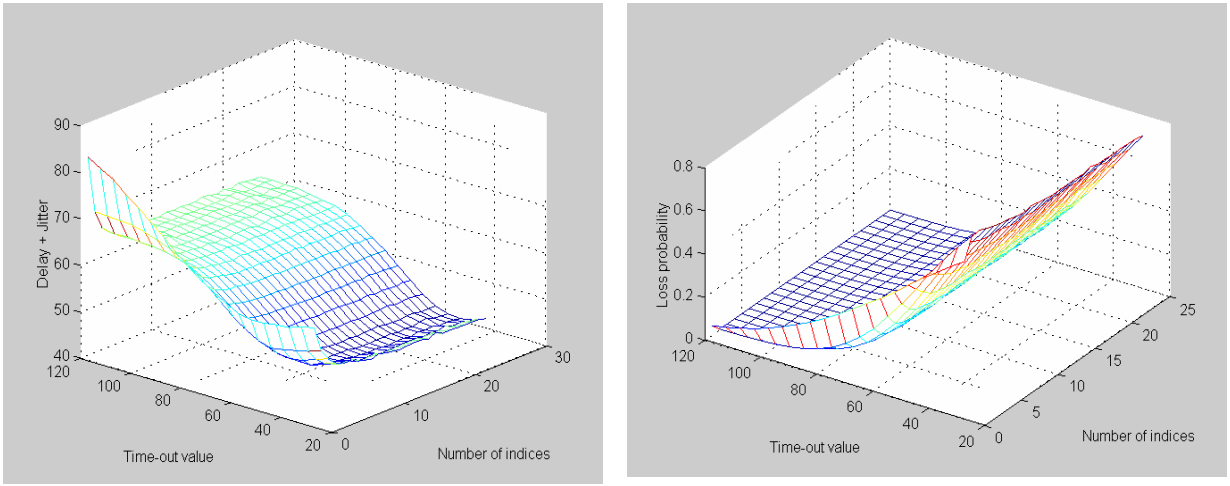
5 Design Optimization Results

We would like to optimize the number of indexes and the time-out value. The number of indexes was varied from 1 to *n* (centralized index to completely distributed indexes) to determine the optimum number of indexes in a P2P network. Also the time-out value was varied from infinite to a certain value. Our objective is to find out the optimal number of indexes and time-out value by minimizing the *delay* and *jitter*, keeping the *loss probability* no more than 7%. During the experiments it was found that the variation in jitter was less than the variation in delay with the

variations in index nodes and the time-out values. Therefore, we normalized the delay and the jitter so that delay does not over shadow the jitter when we solve our problem. Delay was normalized by dividing the entire delay matrix by the maximum value of the matrix. Similarly the jitter was normalized by dividing the jitter matrix with the maximum value of the matrix. It is also possible that the decision maker might want to give some weights to the delay and the jitter depending upon how much influence he/she wants these two to have on the optimal solution. Therefore, if W_d and W_j are the weights assigned to delay and jitter respectively and D_m and J_m are the maximum value in the delay and jitter matrix respectively, then objective function can be written in terms of θ *and* I as:

$$\text{Minimize:} \qquad \frac{1}{D_m} * (1 - W_d) * (E(T_\theta)) + \frac{1}{J_m} * (1 - W_j) * \sqrt{Var(T_\theta)}$$

subject to satisfying the constraints: $L_\theta \leq 7\%$, $\theta < \infty$, and $I \geq 1$.



(a) (b)

Figure 1**:** The optimal solution for 0.8 traffic intensity. (a) Delay + Jitter vs. Number of indexes and Time-out value. (b) Loss probability vs. Number of indexes and Time-out value.

Figure 1 shows the performance parameters vs. number of index nodes and time-out value for the numerical values in Section 4.3 for traffic intensity of 0.8 at each index node for any number of index nodes. The number of nodes was 25 and average arc degree was 5.28. Also the weights for delay and jitter were equal to 0.5 each. From Figure 3 it can be seen that, for a loss probability less than 7%, the objective function is minimum for ***17*** index nodes and time-out value of ***75*** seconds is the optimal solution. Seventeen indexes are better than having only one, as now there is not a single point of failure in the P2P network. It is also better than having all the nodes as indexes as the traffic along the network will not be high. The time-out value of 75 seconds is neither too large nor too small given that there is considerable wait in the queues.

6 Conclusions and Future Work

In a preliminary paper [8], we used simulations to illustrate how well the P2P networks scale by studying the performance measures for varies input parameters (such as number of nodes, average arc degree, number of indexes, time-out value, and probability of node being up). We found that the networks scaled very well. The next step was to study the quality of service experienced by such networks. In the preliminary paper we only considered the simple case where the requests arrived sporadically such that there was no contention for resources. We extend those results in this paper under a more complicated scenario where we consider resource contention, and thereby queueing at the nodes. Here we obtain performance measures such as *delay*, *jitter* and *loss probability* for a request. We test our analytical model against simulations for three different network types: Erdös-Rényi, small world and scale-free. We show that the analytical models, though approximate, do produce good results. Further, we used the analytical results in a mathematical programming problem to determine the optimal number of indexes and time-out value in a P2P network for an acceptable loss probability and minimum weighted sum of *delay* and *jitter*.

In this paper we have not considered a cost analysis of a P2P network. Cost of storing and maintaining data in the network would play an important role when determining the optimum number of indexes and this will be considered

in our future work. Another future consideration is to build approximates for the performance measures under consideration with queues and unreliable nodes.

Acknowledgement and Disclaimer

This research was partially supported by the following grants: ONR award No.N00014-01-1-0859, ARO award No.C-DAAD19 01-1-0646 and NSF award CCR 0073429. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the sponsoring agencies.

References:

1. R. Albert, and A. -L. Barabási, *"Statistical Mechanics of complex networks"*, arXiv: cond- mat/0106096v1, June 2001.
2. B. Bollobás, *"Random Graphs"*, Cambridge University Press, Cambridge, UK, 2001.
3. R. R. Brooks, E. Grele, W. Kliemkiwicz, J. Moore, C. Griffin, B. Kovak, and J. Koch *"Reactive Sensor Networks: Mobile Code Support for Autonomous Sensor Networks"*, *Distributed Autonomous Robotic Systems DARS 2000*, Pp. 471-472. Springer Verlag, Tokyo, October 2000.
4. Dabek F., E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, H. Balakrishnan *"Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service"*, *In the Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII),* Schloss Elmau, Germany, May 2001. http://www.pdos.lcs.mit.edu/papers/chord:hotos01/
5. Dijkstra's 1959 http://www.cs.usask.ca/resources/tutorials/csconcepts/graphs/tutorial/advanced/dijkstra/dijk_descrip.html http://www.cs.usask.ca/resources/tutorials/csconcepts/graphs/tutorial/advanced/dijkstra/dijkstra.html http://isomap.stanford.edu/code/dijk.m
6. M. Jovanovic, *"Modeling large-scale peer-to-peer networks and a case study of Gnutella"*, Master's thesis, University of Cincinnati, 2001.
7. G. Kan, *"Chapter 8: Gnutella"*, in *Peer-to-Peer Harnessing the Power of Disruptive Technologies*, A. Oram, ed. pp. 94-122, O'Reilly, Beijing, 2001.
8. A.K. Kapur, N. Gautam, R.R. Brooks, S. Rai*, "Design, Performance and Dependability of a Peer-to-Peer Network supporting QoS for Mobile Code Applications"*, Proc. Of 10th Int. Conference on Telecommunications Systems, Modeling and Analysis, October 2002.
9. A.K. Kapur, N. Gautam, R.R. Brooks, S. Rai, *"On Design Optimization Problems in P2P Networks for Mobile Code Applications" INFORMS Journal on Computing*, October 2002
10. A.K. Kapur. *Optimal Design of P2P Networks Supporting QoS Issues for Efficient File Sharing*, M.S. Thesis, Industrial Engineering, Penn State University, 2002.
11. A. Langley, *"Freenet"*, pp. 123 – 132b, Orielly 'Peer-to-Peer'.
12. Napster 1999 http://www.napster.com
13. A. Oram, ed. *"Peer-to-Peer Harnessing the Power of Disruptive Technologies"*, O'Reilly, Beijing, 2001.
14. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker, *"A Scalable Content-Addressable Network"*, ACM, 2001 http://www.acm.org/sigcomm/sigcomm2001/p13-ratnasamy.pdf
15. J. Ritter, *"Why Gnutella Can't Scale. No, Really"* http://www.darkridge.com/~jpr5/doc/gnutella.html
16. M. E. J. Newmann, *"Ego-centered networks and the ripple effect or Why all your friends are weird"*, Working Papers, Santa Fe Institute, Santa Fe, NM, http://www.santafe.edu/sfi/publications/workingpapers/01-11-066.pdf.