Adaptive Sharing of Large Resources in P2P Networks

Prithviraj Dasgupta
Computer Science Department
University of Nebraska, Omaha, NE 68182, USA
E-mail: pdasgupta@mail.unomaha.edu

ABSTRACT

A peer-to-peer(P2P) system comprises a network of nodes that are capable of sharing and exchanging resources with one another. Recent studies of P2P networks show that many resources exchanged between users are considerably large files that require significant download times, consume the majority of the network bandwidth, and also occupy substantial storage space on the node providing the resource. In such a scenario, it would be inefficient for a node to store a large resource that is rarely, or never requested by other nodes, or, to share a large resource with a node that is already acquiring the resource from another source. These inefficiences can be mitigated if a node dynamically determines and updates its decision to store and share large resources. Here, we describe an agent enabled adaptive strategy for a node to share large resources based on the expected availability of the resource in the network. Experimental results of our adaptive sharing strategy show that a saving of 8-12 downloads/resource, accounting for 50-70 MB of saved data transfer/resource, can be achieved without performance deterioration in the P2P network.

Categories: I. Computing Methodologies
I.2 Artificial Intelligence
I.2.11 Distributed Artificial Intelligence
Subject Descriptor: *Multiagent Systems*

General Terms: Algorithms, Management, Economics.

Keywords: Peer-to-peer networks, resource management, probabilistic sharing, revelation mechanism.

1. INTRODUCTION

A P2P system comprises a network of users located on nodes which are capable of exchanging and sharing resources with each other in a distribued manner. The decentralized nature of a P2P network makes it attractive for sharing thousands of resources without worrying about problems of scalability or load balancing. The recent popularity of file-sharing networks such as Gnutella[5], Napster, and SETI@home indicate that P2P networks are emerging as a major medium for sharing and exchanging resources between users on the Internet. Recent studies of file-sharing P2P networks[6] have shown that many of the resources exchanged between users are large files greater than 100 MB in size that require download times in the order of days and account for 65 percent of the data transferred in the P2P network. In this paper we address the challenges related to the sharing of such large resources in P2P networks.

In P2P networks sharing is motivated by mutual exchange of resources between users[5, 10]. A user shares the resources it possesses with other users so that it can expect to receive resources in return from those users. In such a scenario, it is counter-intuitive for a user to share large resources that are very rarely, or, never requested by other users. Sharing unrequested large resources does not enhance the possibility of a user of receiving resources from other users. However, the large resources still consume substantial storage space on the user's node.

Large resources also consume considerable data transfer bandwidth when they are shared(downloaded) between nodes. It is a challenging problem to reliably transfer large amounts of data in a P2P network that has no central server node to implement a congestion control algorithm for guaranteeing resource downloads. This makes it inefficient for a node to commence sharing a large resource with another node while running the risk of the download process getting aborted before the requesting node completely acquires the resource. Therefore, it makes sense to investigate mechanisms that would enable users to share large resources without incurring the overheads stemming from inefficient storage or sharing of large resources.

Previous research addressing the resource sharing problem in P2P networks uses distributed congestion control algorithms such as controlled flooding[3], and fluid flow models[9]. However, most of these techniques address the problems of P2P resource sharing at the cost of decreased availability of resources in the network that adversely affects the performance of the P2P network. In[6], data traces reported from resource sharing in real-life P2P networks indicate that it is difficult to model the number of requests for large resources using pre-defined distributions such as the zipf distribution used to model Web page requests[4]. Instead of degrading the network performance or modeling P2P resource sharing characteristics with a pre-defined distribution, we envisage that a better technique for sharing large resources would be to dynamically adjust the probability of sharing and storing a large resource based on the demand for the resource among other nodes in the network and the current availability of the resource in the network. In this paper, we propose a software agent enabled mechanism that adaptively determines the sharing decision of a node for a large resource. Our experimental results show that the adaptive sharing strategy can achieve a saving of 8-12 downloads/resource, accounting for 50-70 MB of saved data transfer/resource, without performance deterioration in the P2P network.

2. CHALLENGES OF SHARING LARGE RESOURCES IN A P2P NETWORK

A P2P network enables sharing of resources among participating nodes in a distributed and decentralized manner. New nodes enter the network using the *node discovery protocol* while sharing and exchanging resources between the nodes is implemented through the *resource discovery protocol*. We assume that the network and resource discovery protocols are already available in our P2P network. We consider an unstructured and pure(completely decentralized) P2P network that does not contain a central server node to maintain information about participating nodes and administer resource management. In pure P2P systems, a search query for a resource from a node is flooded across the network. This ensures that every node in the network receives a query originated by any node in the network.

In our proposed mechanism, we assume that users are located on nodes of the P2P network, and each node is provided with an agent that enables the user to determine its resource sharing decision with other users(nodes). In the rest of the paper, we use the terms user, node and agent interchangeably. The problems for sharing large resources in P2P networks are the following:

- **Constrained Storage Space.** The nodes of a P2P network are located on computers with finite storage capacity. The storage capacity of a node constrains the size and number of large resources that can be shared by the node. In a P2P network, a node is motivated to share a resource with other nodes because it expects to receive resources in return from those nodes. A suitable mechanism for sharing resources should adaptively remove from the node's storage, large resources that no other node wishes to acquire, because those resources do not enhance the node's chances of receiving resources from others.

- **Aborted Downloads.** When a large resource is requested by a node, the providing node incurs considerable transfer bandwidth and download time for sharing the resource. Partially completed downloads are inefficient for the node sharing the resouce because it still expends bandwidth and download time to share part of the resouce. However, the providing node does not qualify to receive resources in return because it did not provide complete resources to the requesting nodes. The problem of aborted downloads can be mitigated by a mechanism that avoids unnecessary downloads of a resource as described below.

- **Redundant Downloads.** In a P2P network, a node often requests the same resource from multiple providing nodes to improve its chances of acquiring the resource when download completion is not guaranteed. Such a scenario can lead to aborted partial downloads for the nodes providing the resource as soon as the requesting node obtains the first copy of the resource from one of the providing nodes. Unnecessary downloads can be reduced by a mechanism that adapts a node's decision to share a resource based on the availability of the resource among other nodes in the network and the expected demand of the resource in the network.

In the next section, we describe a technique for sharing large resources in a P2P network to address these issues using an agent based adaptive sharing mechanism. The adaptive sharing mechanism requires each agent to perform local computations only to determine its resource sharing decision so that the mechanism can operate efficiently in a decentralized P2P environment.

3. ADAPTIVE SHARING FOR LARGE RESOURCES

In a P2P network, a node's agent shares a resource with other nodes' agents so that it can expect to acquire resources in return from those agents. The utility received by an agent from sharing a resource is determined by the number of agents that wish to acquire the shared resource. This means that the sharing utility of a resource should be proportional to the number of expected requests from other agents for that resource. As the resource gets shared between agents, the number of agents still wishing to acquire the resource diminishes. Therefore, the sharing utility of a resource should be dynamically updated to reflect the change in the number of expected requests for that resource.

Resource demand data from file-sharing P2P networks show that a resource's demand decreases over time[6]. Newly available resources are requested frequently for download while relatively older resources are seldom requested. Following this P2P resource demand characteristic we assume that an agent values a resource in proportion to the sharing utility of that resource. When a new resource is introduced by an agent in the P2P network it has a certain value that diminishes as other agents download the resource and acquire copies of it. The number of expected requests for a resource at a particular instant should be the same at all agents(nodes) across the P2P network. Since the value of a resource to an agent is determined by the number of expected requests for the resource, therefore, it is intuitive for all agents possessing a particular resource to use the same valuation function for the resource. Also, the value of a resource to an agent ultimately becomes zero when every agent in the network wishing to acquire the resource receives it. To model these characteristics we have chosen the resource valuation function of an agent as a linearly decreasing function of the form:

$$v_t = v_0(1 - \frac{\text{No. of agents possessing the resource at instant t}}{\text{Total no. of agents wishing to acquire the resource}}),$$

where v_t is the valuation of a resource that was introduced in the network t instances ago, and, v_0 is the initial valuation of the resource by the agent that introduces it in the network. Since every agent is rational, the probability of sharing a resource is determined as the relative value of the resource at instant t, viz., v_t/v_0. Therefore, the sharing strategy is independent of the initial resource value v_0 selected by the introducing agent.

The resource sharing function of an agent $\pi : v \times t \rightarrow [0, 1]$ takes the relative valuation of the resource and the duration since the resource was introduced and returns the probability of sharing the resource. The resource is removed from the agent's storage when the sharing function returns a zero probability for sharing the resource. We now show the calculations performed by an agent in its resource sharing function to calculate the resource sharing probability from the parameters reported by other agents. The calculations are shown for one resource. and the same calculations can be used for every resource.

The parameters for our analytical model of the P2P network are the following:

N	No. of agents in the network wishing to acquire a resource
k	Average upload capacity of agents(nodes) in the network
χ	Fraction of agents in the network that share their resources with other agents
v_t	Valuation of a resource by an agent after download round t
v_0	Initial valuation of a resource by the agent that introduces it

s_t	No. of agents sharing the resource during download round t
δ_t	Average no. of download requests for a resource received by an agent during download round t
ρ	Rate of adding new resources in the network
π_t	Probability of sharing a resource during download round t

We consider a download-round as the average time required to download a resource. We consider one download round as the metric for time in our model to remove asynchrony arising out of communication delays in our calculations. Also, notice that the average upload capacity k corresponds to the average number of agents that can acquire the resource in each download round from one agent sharing the resource.

Case I. *Agents do not share resource after downloading.* First, we consider the case where only the agent A_0 that introduces the resource in the network shares it while agents that download the resource from A_0 do not share it. In each round k agents acquire the resource from A_0. Since none of the agents receiving the resource share it, therefore, the value of the resource diminishes by k/N in each download round. The value of the resource v_t after download round t is given by $v_0(1 - kt/N)$. The probability of sharing the resource after t download rounds is then given by

$$\pi_t = \frac{v_t}{v_0} = \frac{v_0(1 - kt/N)}{v_0} = 1 - kt/N \tag{1}$$

Note that, in this scenario, the probability of sharing a resource depends on the average download capacity of agents k. The resource is no longer requested by any agent after $t = N/k$ download rounds.

Case II. *Agents share resource after downloading.* Now we consider the case when agents that download the resource selectively share it with other agents in the network that request the resource. Since each agent providing the resource can support k simultaneous uploads, the number of agents acquiring the resource during download round t is given by:

$$d_t = ks_t, \tag{2}$$

where s_t is the number of agents sharing the resource during download round t. s_t is given by the sum of the number of agents were sharing the resource during the previous download round $(t-1)$, and, the number of agents that acquire the resource during download round $(t-1)$ and also share it. Therefore, we get:

$$s_t = s_{t-1} + \chi d_{t-1}. \tag{3}$$

Substituting $d_{t-1} = ks_{t-1}$ obtained from Equation 2 in Equation 3 we get:

$$\begin{aligned} s_t &= s_{t-1} + \chi k s_{t-1} \\ or, s_t &= s_{t-1}(1 + \chi k) \end{aligned} \tag{4}$$

Solving the above recurrence relation with $s_1 = 1$ (the agent(node) that introduces the resource in the network is the only agent that shares during download round 1) we get

$$s_t = (1 + \chi k)^{t-1} \tag{5}$$

Substituting the value of s_t from Equation 5 in Equation 2 we get

$$d_t = k(1 + \chi k)^{t-1}$$

The total number of agents that acquire the resource since the introduction of the resource in the network until download round t is given by

$$\sum_t d_t = \sum_t k(1 + \chi k)^{t-1} = \frac{(\chi k + 1)^t - 1}{\chi} \tag{6}$$

In a manner similar to the derivation of Equation 1 the valuation of the resource after t download rounds is given by $v_0(1 - \frac{(\chi k+1)^t - 1}{N\chi}$. Therefore, the probability of sharing the resource after t download intervals is:

$$\pi_t = \frac{v_t}{v_0} = 1 - \frac{(\chi k + 1)^t - 1}{N\chi} \tag{7}$$

The number of download rounds τ elapsed before every agent wishing to acquire the resource receives it can be calculated by setting $\frac{(\chi k+1)^\tau - 1}{\chi} = N$ and solving for τ which yields

$$\tau = log_{\chi k+1}(\chi N + 1) \tag{8}$$

3.1 Determining χ, k and N

An agent uses Equation 7 to determine the value of its sharing function. However, in a decentralized P2P setting it is difficult for a single agent to determine the values of the network-wide (global) parameters on the r.h.s of Equation 7. We now inspect these parameters and discuss mechanisms and calculations for each agent to detemine them locally.

- *Determing the average upload capacity of nodes, k*: To enable calculation of k, each agent in the network reports, (i.e. broadcasts) its upload capacity k_i to all other agents(nodes) in the network. This is a valid mechanism for agents to report their k_i values because the node discovery protocol used by P2P networks such as Napster and Gnutella, requires a node to report its upload capacity when it joins the network. Since the upload capacity of an agent(node) does not change over time, these values have to be reported only once by each agent when it joins the network. Each agent then locally calculates k as the sum of individually reported k_i-s averaged over the number of agents reporting, i.e.,

$$k = \sum_{i=1}^{i=N} k_i/\mathcal{A},$$

where $\mathcal{A}$ is the number of agents reporting their values of k_i-s.

- *Calculating the fraction of agents that share the resource, χ*: The agents that do not share the resource are those that report a value $k_i = 0$. The fraction of agents that share the resource is then calculated by each agent as:

$$\chi = 1 - \frac{\text{No. of agents that report } k_i = 0}{\mathcal{A}}$$

where $\mathcal{A}$ is the number of agents reporting their values of k_i-s. Therefore, χ can be calculated locally by an agent from the k_i values reported to it from other agents.

- *Calculating the number of agents wishing to acquire the resource, N*: As mentioned in Section 2, we consider a pure P2P system where resource search queries are flooded across the network and every agent in the network receives a request for a resource initiated by a particular agent in the network. We assume that the time to flood resource requests $\ll$ time to download a large resource. Therefore, an agent receives all requests for sharing a resource from agents that wish to acquire the resource but haven't yet acquired it, before the

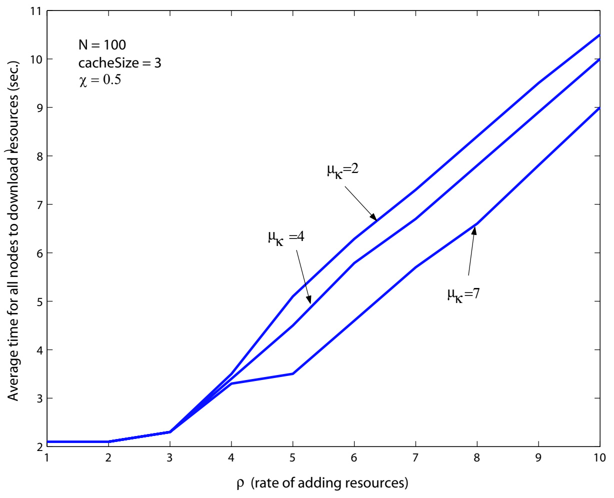

Figure 1: Variation in the average download time for resources with ρ for different values of μ_k.

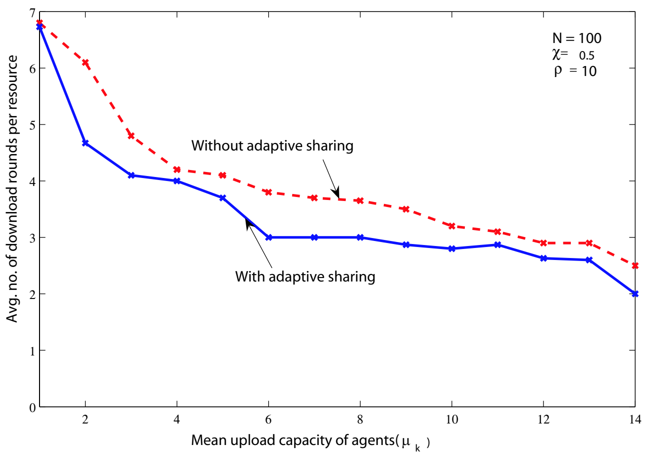

Figure 2: Variation in the average no. of download rounds per resource for different values of average upload capacities of agents(μ_k) with and without adaptive sharing.

agent commences a download round. Then, the total number of agents, N, wishing to acquire a resource is given by the sum of the number of download requests received each agent in the network during a particular download round t and the number of agents that already acquired the resource between download rounds $1..(t-1)$, i.e.,

$$N = \delta_t + \sum_{1}^{t-1} d_t,$$

where t can take values between 1 and τ (τ is the no. of download rounds elapsed before every agent wising to acquire the resource receives it).

Substituting $\sum d_t$ from Equation 6 we get:

$$N = \delta_t + \frac{(\chi k + 1)^{t-1} - 1}{\chi},$$

Substituting this value of N in Equation 7 we get:

$$\pi_t = 1 - \frac{(\chi k + 1)^t - 1}{\chi\delta_t + (\chi k + 1)^{t-1} - 1} \tag{9}$$

In Equation 9, each agent is aware of the number of resource requests, δ_t, it receives. t is the age of the resource since it was first introduced in the network and is measured using the number of download rounds. t can be easily calculated by each agent by subtracting the *date/time of creation* attribute passed with the resource from the current time. Therefore, an agent can calculate the value of its sharing function given by Equation 9 locally from its values of δ_t, t, χ and the reports of the k_i values it receives from other agents.

4. EXPERIMENTAL RESULTS

We simulated our algorithm for adaptive resource sharing on a P2P network with the following parameters: $N = 100$, k_i values for the nodes are drawn from the normal distribution $N(\mu_k, 1.0)$ where μ_k is varied between 1 and 10 to simulate agents(nodes) with different upload capacities. χ is varied between 0.05 and 1.0 to simulate different proportions of sharing agents, while ρ is varied between 1 and 10 to simulate different rates of introducing new resources in the network. The duration of 1 download round for a resource is chosen randomly between 5-10 seconds to simulate different download times for different resource sizes. We refer to the storage space of a node as its cache, and, the storage capacity as the node's cache size. We assume that each agent(node) can hold upto 3 resources in its cache, and, therefore, has a cache size of 3. If the cache gets full, an agent has to remove at least one resource it is sharing from its cache before it can download another resource. In our simulations, the resource with the least value of the sharing function is selected for removal when the cache gets full. Resource removal is assumed to take a constant time of 1 sec. on each node. In all the experiments performed, the adaptive sharing required the same or lesser resource download completion times as the non-adaptive mechanism. Moreover, in all the experiments, the adaptive sharing mechanism achieved significant savings in the number of downloads over the non-adaptive mechanism, by avoiding redundant downloads.

Figure 1 shows the variation in the average download time for a resource for different values of the resource introduction rate, ρ. When resources are introduced rapidly, the downloading activity in the network increases. This increases network traffic and network latency and results in increased download time for all resources. Therefore, as shown in Figure 1, the average time to download a resource increases as the rate(ρ) at which new resources get added increases. When the average upload capacity of the agents, k, increases, the download time decreases because each agent can support more uploads in each round resulting in faster dissemination of the resource in the network. Each curve in Figure 1 exhibits a knee around $\rho = 3$. This happens because with an introduction rate $\rho \leq 3$, the resources in the cache do not need to be replaced as cache size $= 3$. However, beyond $\rho > 3$, one resource needs to be replaced from each node's cache, for each resource added. The delay in replacing a cache entry accounts for the increase in download times for $\rho > 3$.

Figure 2 shows the variation of the average no. of download rounds for a resource for different values of μ_k for $\rho = 10$. As the mean upload capcity μ_k of agents increases, each agent can down-

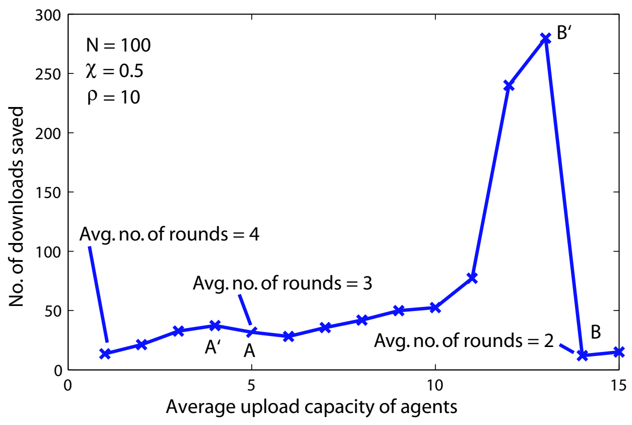

Figure 3: No. of downloads saved with adaptive sharing vs. average upload capacities of agents(μ_k) .

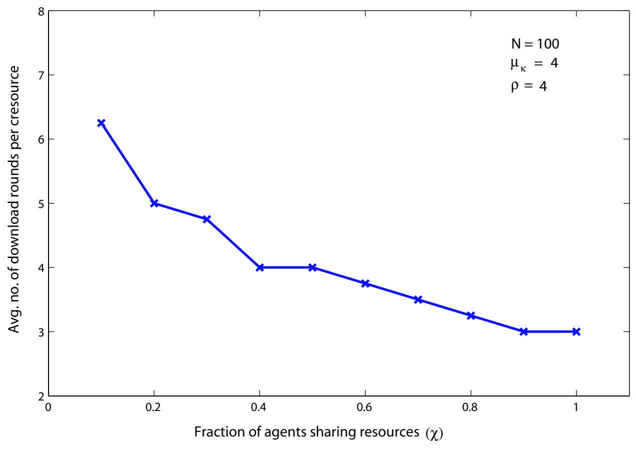

Figure 4: No. of download rounds/resource vs. reported no. of agents sharing the resource with adaptive sharing.

load more resources. Therefore, the average number of rounds to download a resource decreases when μ_k increases. When adaptive(probabilistic) sharing is used, redundant downloads are avoided and this accounts for between 10-35 percent reduction in the average number of download rounds for each resource.

Figure 3 shows the number of resource downloads that were saved for different values of μ_k when resources are shared adaptively. Similar shaped curves were obtained for different values of ρ in the range of $1 - 9$. As shown in Figure 3, the number of downloads saved illustrates a cyclic pattern. When the average no. of rounds to download resources approaches a transition to a lower value, the number of downloads saved drops significantly as shown at points A and B on the curve. Just prior to these points, the number of downloads saved has a considerably high value (points A' and B' on the curve). The reason for this behavior can be attributed to the fact that at points A' and B' only a few agents do not have the resource in the last round as a high value of μ_k ensures that most agents receive the resource before the last round. Therefore, the downloads, if any, in the last one round (for point A') or two rounds (for point B')comprise redundant downloads and are avoided by our adaptive sharing mechanism. Overall, the adaptive sharing strategy achieves a saving of 8-12 downloads per resource per download round accounting for about 50-70 MB of saved data transfer corresponding to every resource in every round.[1]

Figure 4 shows the variation in the average number of download rounds per resource for different values of χ when $\rho = 4$ when adaptive sharing is used. As more agents stop sharing resources (χ gets smaller) the resource becomes less available in the network because there are fewer 'sharing' ndoes from which the resources can be downloaded. For $\chi < 0.1$, the download of the resource is not guaranteed because with such a low value of the fraction of sharing agents, very few agents end up sharing the resource. This scenario illustrates a society of selfish agents that are only interested to acquire resources (if available) but do not share resources themselves with other agents. When compared with non-adaptive sharing, downloads are successful for lower values of χ than with non-adaptive sharing because avoiding the redundant downloads increases the chances of successful downloads even when very few nodes share.

[1]Assuming a download bandwidth of 5 Mbps for each node and a download time between 5-10 secs. per resource.

5. ENSURING TRUE REPORTS FROM OTHER AGENTS

A major concern in P2P networks is the problem of free-riding[1]. Free-riders are selfish agents that acquire resources from other agents that share resources, but themselves never share resources with other agents. Free-riders can exploit our adaptive sharing mechanism by falsely understating their upload capacity, k_i, when they broadcast it. When any one agent i understates its value of k_i, the average upload capacity, k computed by each agent in the network decreases, and, correspondingly the value of π_t increases in Equation 9. Therefore, if an agent i understates k_i it can coerce other agents to share the resource for a longer duration. This would affect the mechanism proposed in Section 3 by causing agents to share large resources even when those resources actually have no demand. Therefore, it is crucial to ensure that each agent i does not falsify its reported value for k_i.

To ensure true reports of the k_i values from agents we propose a Vickrey-Clarkes-Grove(VCG) type revelation mechanism [8]. In VCG mechanisms, each agent i from a set $\mathcal{A}$ of agents, is assumed to have a quasi-linear utility function $u : \Theta \rightarrow \Re$. Each agent i reports its value, also called the agent's type, $\theta_i \in \Theta_i$, (Θ_i is the set of types for agent i)to a central location. A set of alternatives L is used to specify the outcome and one of these alternatives is chosen by the central location using the *choice function*, $l : \Theta_1 \times ...\Theta_{|\mathcal{A}|} \rightarrow L$, specified by the VCG mechanism. Each agent is then assigned the outcome determined by the choice function by the central location. The VCG mechanism also assigns a side payment $t_i(\theta)$ to each agent determined by the VCG mechanism's *payment function* $t : \Theta_1 \times ...\Theta_{|\mathcal{A}|} \rightarrow \Re$. The side payments to each agent ensure that the external effects of the agent's revealed value are internalized, i.e., it is in the agent's best interest to reveal its true type and enable the mechanism to select the alternative corresponding to the best outcome for all agents.

We now show that using the VCG mechanism in our model, each agent reports its true value of k_i. The VCG mechanism requires every agent to report its type to a central location that computes the choice and payment functions. In our model, k_i values are broadcast over the network and each agent receives the reports from all other agents in the network. Therefore, the VCG mechanism described below is applied $\mid \mathcal{A} \mid$ times in our P2P network, once for each agent as the central location, to preserve the distributed nature of the system. Alternatively, these calculations can also be performed on a single trusted node, with the values calculated by the single node being broadcast to all agents in the network.

To implement the VCG mechanism in our model, we have defined the quasi-linear utility function of agent i as:

$$u_i(\theta_i) = v_i(l, \theta_i) - t_i(\theta_i), \tag{10}$$

where, $\theta_i = k_i$, the revealed upload capacity of agent i, and,$v_i(l, \theta_i)$ is the value obtained by agent i from this revelation corresponding to alternative $l \in L$.

The set of alternatives corresponding to the outcome are:

$$L = \begin{array}{ll} 1 & \text{if the resource is received by an agent} \\ 0 & \text{if the resource is not received by an agent} \end{array}$$

The valuation function of each agent i is specified using the following parameters:

γ	Price of the resource if the agent had to purchase it from outside the P2P network.
c	Cost to acquire the resource in the P2P network. This cost includes the search cost and the download bandwidth cost incurred by the agent to acquire the resource.
θ_i	Upload capacity k_i revealed by the agent.

Without loss of generality, we have assumed that the parameters γ and c are identical for each agent. The valuation function for agent i is then given by:

$$v_i(l, \theta_i) = \begin{array}{ll} \gamma - c - \sum_{t=t_i}^{\tau} \theta_i & \text{if } l = 1, \\ -\epsilon & \text{if } l = 0. \end{array}$$

The term $\sum_{t=t_i}^{\tau} \theta_i$ in the above equation gives the number of times agent i will share the resource since acquiring it in download round t_i until the last download round τ. This summation corresponds to the decrease in the resource's value due to sharing. (The value of τ can be calculated using Equation 8.) When $l = 0$, the agent incurs a cost ϵ for searching the resource.

Applying the choice function of the VCG mechanism:

$$l^*(\theta) = \text{arg max}_{l \in L} \sum_i v_i(l, \theta_i),$$

to this setting gives:

$$l^*(\theta) = \text{arg max}_{l \in L} \{((\gamma - c) \mid \mathcal{A} \mid - \sum_{i=1}^{i=|\mathcal{A}|} \sum_{t=t_i}^{\tau} \theta_i), -\epsilon \mid \mathcal{A} \mid\} \tag{11}$$

Therefore, the choice function returns $l = 1$ corresponding to the valuation $\gamma - c - \sum_{t=t_i}^{\tau} \theta_i$ of each agent.

The payment function of the VCG mechanism is given by:

$$t_i(\theta) = \sum_{j \neq i} v_j(l^*_{-i}(\theta_{-i}), \theta_j) - \sum_{j \neq i} v_j(l^*(\theta_i, \theta_{-i}), \theta_j),$$

Substituting this value of $t_i(\theta)$, the utility to agent i is given from Equation 10 we get:

$$u_i(\theta_i) = v_i(l, \theta_i) - \sum_{j \neq i} v_j(l^*(\theta_i, \theta_{-i}), \theta_j) + \sum_{j \neq i} v_j(l^*_{-i}(\theta_{-i}), \theta_j)$$

Agent i's report of θ_i does not affect the last term in the above equation, and so, it can be ignored while calculating $u_i(\theta_i)$. Substituting the values of the first two terms in the above equation from the valuation function under chosen alternative $l = 1$, we get:

$$u_i(\theta_i) = \gamma - c - \sum_{t=t_i}^{\tau} \theta_i + ((\gamma - c)(\mid \mathcal{A} \mid -1) - (\sum_{j=1..i-1, i+1..|\mathcal{A}|-1} \sum_{t=t_j}^{\tau} \theta_j)) \tag{12}$$

The objective of agent i is to reveal type $\hat{\theta}_i$, that might not necessarily be its true type, such that it maximizes its utility given in Equation 12. i.e. max $_{\hat{\theta}_i \in \Theta_i} u_i(\theta_i)$. However, the revealed type $\hat{\theta}_i$ of agent i only has an effect on the choice function and the agent can maximize Equation 12 by revealing its true type. Therefore, agent i reports its true type θ_i, or, in other words, its true upload capacity, k_i to other agents by using the VCG mechanism.

6. RELATED WORK

Resource sharing mechanisms in P2P networks has been an active research topic since the inception of P2P networks. Many researchers [12, 13] have proposed structured overlays using distributed hash tables (DHT-s) for P2P networks to address resource management issues. However, most of these systems address the issue of improving the search latency for resource queries in P2P networks instead of directly addressing the problems of sharing large resources. Several researchers [2, 7, 14] have also addressed the problem of incentive based mechanisms for resource sharing in P2P networks. Most of these systems use a game theoretic approach and employ reputation and trust based mechanisms to incentivize resource exchange between P2P users and mitigate the free-rider problem. These research concentrate on the economic aspects of sharing resources and do not propose solutions for sharing large resources. The issues related to congestion control in decentralized P2P networks have been addressed using distributed congestion control algorithms in [3, 9]. However, it is difficult to apply these congestion control techniques to the P2P resource sharing protocol because they reduce the availablity of a resource and result in a deterioration of performance in the P2P network.

A major motivation behind our research has been the data traces resulting from resource sharing in commercially deployed file-sharing P2P networks which are reported in [6]. The findings in these papers indicate that many resources exchanged between P2P users are large objects that consume significant network bandwidth, and, often require several days to download. They also indicate that, contrary to earlier reports, the number of requests for a resource in a P2P environment does not follow the zipf distribution that is commonly used for Web page requests[4]. This makes it difficult to construct an analytical model for P2P resource sharing based on pre-defined distributions. Therefore, in this paper we have proposed an adaptive mechanism to model the probability of sharing large resources.

7. CONCLUSION AND FUTURE WORK

In this paper, we described an agent based adaptive mechanism for probabilistically sharing large resources in a P2P environment. Our simulation results show that such a mechanism can address the problems of sharing large resources without performance degradation.

In the analytical model of the P2P network considered in this paper, we assumed that resource search queries are flooded in the network according to a Gnutella-like model. In the future we propose to extend our adaptive sharing mechanism to model resource distributions when sophisticated P2P search algorithms such as DHTs[13] and evolutionary search mechanisms are used. Also, we assumed that nodes do not frequently join and leave the P2P network which enables us to model the set of nodes requesting a particular resource as a constant set. However, when nodes dynamically join and leave the network, this set can also change dynamically. We propose to investigate the effect of these changes on the computation of the sharing probability of a resource. Finally, we plan to investigate the effect of sharing incentives based on trust and referrals on the adaptive sharing mechanism described in this paper.

8. REFERENCES

[1] E. Adar and B. Huberman, “Free Riding on Gnutella,” First Monday.

[2] S. Braynov and T. Sandholm, “Incentive compatible mechanism for trust revelation,” AAMAS 2002, pp. 310-311.

[3] L. Costa, M. Amorim, and S. Fdida, “Reducing Latency and Overhead of Route Repair with Controlled Flooding,” Wireless Networks, vol. 10, no. 4, 2004, pp. 347-358.

[4] M. Crovella, M. Taqqu, and A. Bestavro, “Heavy-tailed probability distributions in the World Wide Web,” A Practical Guide To Heavy Tails, eds. R. Adler, R. Feldman, M. Taqqu, Chapman and Hall, 1998.

[5] Gnutella, URL http://www.gnutella.com

[6] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy and J. Zahorjan, “Measurement, modeling, and analysis of a peer-to-peer file-sharing workload,” SOSP 2003, pp. 314-329.

[7] P. Golle, K. Leyton-Brown, and I. Miranov, “Incentives for sharing in P2P Networks,” Proceedings of the ACM Conference on Electronic Commerce, 2001, pp. 264-267.

[8] A. Mas-Colell, M. Whinston and J. Green, “Microeconomic Theory,” Oxford University Press, 1995.

[9] L. Massoulie, “Stability of Distributed Congestion Control with Heterogeneous Feedback Delays,” Microsoft Research Technical Report, 2000.

[10] Mojo Nation, URL http://sourceforge.net/projects/mojonation

[11] The FastTrack Protocol URL http://cvs.berlios.de

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable network,” SIGCOMM 2001, pp. 161-172.

[13] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for internet applications,” IEEE/ACM Transactions on Networking, vol. 11, no. 1, 2003, pp. 17-32.

[14] B. Yu and M. Singh, “Searching Social Networks,” Proceedings of Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, 2003, pp. 65-72.