

# An Incentive Mechanism for P2P Networks

Richard T. B. Ma\*   Sam C. M. Lee\*   John C. S. Lui\*   David K. Y. Yau†

## Abstract

The current peer-to-peer (P2P) information sharing paradigm does not provide incentive and service differentiation for users. Since there is no motivation to share information or resources, this leads to the “free-riding” and the “tragedy of the commons” problems. In this paper, we address how one can incorporate incentive into the P2P information sharing paradigm so as to encourage users to share information and resources. Our mechanism (or protocol) provides service differentiation to users with different contribution values and connection types. The mechanism also has some desirable properties: (1) conservation of cumulative contribution and social utility in the P2P community, (2) maximization of social utility if all requesting clients have the same contribution value, and (3) incentive-based resource distribution. The resource distribution algorithm and the contribution update algorithm are computationally efficient and can be easily implemented. Experimental results illustrate the efficiency and fairness of our algorithms.

## 1. Introduction

The rapid growth of decentralized and structured or unstructured peer-to-peer (P2P) networks [16, 13, 11] holds great potential for efficient information exchange in the Internet. A P2P network may exhibit a power-law topology [12] such that it can propagate queries quickly and, if implemented efficiently [16], it can locate objects in  $\log(n)$  time, where  $n$  is the number of nodes in the network. However, there are remaining problems in the P2P information sharing paradigm which complicate its deployment. *Free-riding* and the *tragedy of the commons* are two major problems. As reported in [3], nearly 70% of Gnutella users do not share any file with others in a P2P community and nearly 50% of

all search responses come from the top 1% of content sharing nodes. Therefore, nodes that share information and resources are prone to congestion, leading to the tragedy of the commons [7]. Another problem is that many users intentionally misrepresent their connection speeds so as to discourage others from going to their nodes for file download. Worse yet, Gnutella-like systems give no service differentiation between users who do not share any information with or make any contribution to the P2P community.

The objective of this paper is to design and analyze a protocol that provides incentives for users to share information and offers preferential service to users who contribute to the P2P community. We address the following issues:

- 1) How to utilize transfer bandwidth resources efficiently?
- 2) How to fairly serve different nodes which have different connection types and contributions in a P2P community?
- 3) How to avoid problems of free-riding and the tragedy of the commons?

Many current P2P systems use the first-come-first-served policy in providing file transfer services [2]. This may cause large response time or even starvation for requests queued after other long running requests. Alternatively, round robin scheduling can be used. However, evenly distributing the transfer bandwidth between requesting users may not be suitable. First, this may not be an efficient choice for the P2P network since different nodes may have different connection types (e.g., modem, LAN, ADSL) and speeds, and they may achieve different utilities even if given the same amount of transfer bandwidth resource. Instead, one should consider the problem of distributing the transfer bandwidth resource so as to maximize the aggregate utility. Second, it may not be fair since some requesting nodes may have contributed a lot more than other requesting nodes. These considerations lead us to propose a scheduling policy which is based on the aggregate utility, the connection types and contribution values of individual requesting nodes. Such a policy gives a rational user *incentive* to share information and contribute service to a P2P community.

The balance of the paper is organized as follows. In Section 2, we introduce the notations and model of our incentive P2P network. In Section 3, we present the algorithms for resource distribution. In Section 4, we present an algorithm for computing the contribution values of all partic-

\* Department of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, N.T. Hong Kong; {tbma, cmlee, cslui}@cse.cuhk.edu.hk.

† Department of Computer Science, Purdue University, West Lafayette, IN; yau@cs.purdue.edu.

icipating nodes. In Section 5, we report experimental results to illustrate the dynamics between the contribution values of participating nodes and their received bandwidth. Related work is discussed in Section 6. Section 7 concludes.

## 2. Incentive P2P Networks

In this section, we first present a model of the Gnutella protocol – a common P2P network protocol – and some of its inherent problems. Although we compare and contrast our work with Gnutella, the proposed mechanism in this paper can be used to give incentive in more recent P2P protocols, such as [16]. We then present the notations in our design of an incentive-based protocol, and state some of the protocol’s desirable properties.

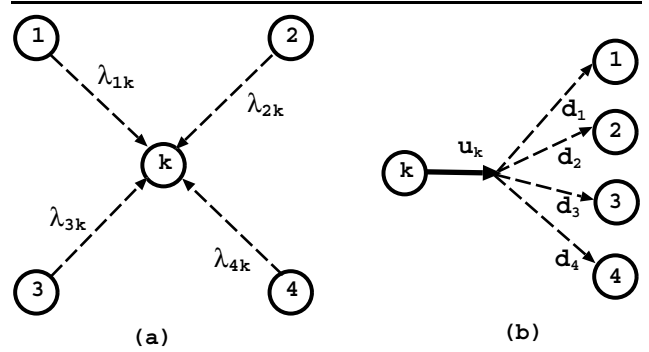
### 2.1. The Gnutella protocol and its inherent problems

Gnutella [1], an open P2P protocol for connection management and distributed search, represents a class of decentralized unstructured P2P networks. In a Gnutella-like network, each node (also called a *servent*) plays the role of both a client and a server. The Gnutella protocol specifies rules for sending/answering queries and maintaining the connectivity between different servents. Each servent joins a P2P network by connecting to some existing servents in the network. A servent, say  $i$ , performs file searching by sending queries to its neighbors, which can in turn forward the query to their own neighbors. Once the file is located in a set of servents, say  $\mathcal{S}$ , servent  $i$  can request a file transfer from any servent  $j \in \mathcal{S}$ .

To formally describe the *logical* and *physical* views of the file transfer process, we define the following notations:

- $\mathcal{N}$ : A set of all servents in a P2P system with  $|\mathcal{N}| = N$ .
- $\Lambda = [\lambda_{i,j}]_{N \times N}$ , where  $\lambda_{i,j}$  represents the average file transfer request rate from servent  $i$  to servent  $j$ .
- $\Psi = (u_1, u_2, \dots, u_N)$ , where  $u_i, i \in \mathcal{N}$ , represents the *maximal* upload bandwidth (in Mbps) of servent  $i$ .
- $\mathcal{D} = (d_1, d_2, \dots, d_N)$ , where  $d_i, i \in \mathcal{N}$ , represents the *maximal* download bandwidth (in Mbps) of servent  $i$ .
- $\mathcal{R}_k$ : The set of servents which may request file download from servent  $k$ ; i.e., any servent  $j$  for which  $\lambda_{j,k} > 0$ .

Figure 1 illustrates the logical and physical views of the file transfer process. Figure 1(a) depicts the query/search process. Servents 1, 2, 3, and 4 have found that servent  $k$  has the file that they are searching for, and they decide to request the file from servent  $k$ . Servent  $k$  has a physical download bandwidth of  $d_k$  and an upload bandwidth of  $u_k$  (both in Mbps). Typically, we have  $d_k \geq u_k$ . For example, for an ADSL connection, we have  $d_k > u_k$ . But for other full-duplex network technologies (e.g, Ethernet and ATM), we



**Figure 1. (a) logical and (b) physical views of file transfers in a P2P network.**

have  $d_k = u_k$ . Therefore, to transfer the requested files, servents 1, 2, 3, and 4 have to *share* the upload bandwidth  $u_k$  of servent  $k$ . This is illustrated in Figure 1(b).

In this paper, we assume that there are  $N$  servents in the P2P network. We define  $\Lambda$  to be an  $N \times N$  matrix with  $\lambda_{i,j}$  denoting the average file transfer request rate from servent  $i$  to servent  $j$ . A servent, who shares certain contents in a P2P community, receives different rates of request from other servents. In general, the request rates depend on (1) the popularity of the contents offered by the target servent, and (2) the target servent’s upload bandwidth capacity.

The Gnutella protocol specifies that the file transfer is to be carried out over HTTP. Most Gnutella client implementations maintain multiple HTTP connections, but limit the maximum number of such concurrent HTTP transfers at any given time. For example, LimeWare [2] uses *upload slots* to limit the number of HTTP connections. When the upload slots are used up, new file transfer requests are queued and LimeWare uses the FCFS scheduling policy to process the waiting requests. With FCFS, requests may experience a long waiting time. For example, if the active HTTP sessions are being occupied with large file transfers, requests in the queue will have to wait for a long time before receiving service.

To reduce the waiting time, an alternative solution is to use some form of processor-sharing discipline in scheduling the file transfer requests. However, such an approach has its own problems. First, the simple strategy of giving an equal share of the transfer capacity to each requesting node can be inefficient, because the download capacity of some of the requesting nodes may be smaller than their allocated capacity share. Second, even if all the requesting servents can fully utilize their bandwidth share, the equal resource allocations ignore the more relevant issue of end user’s satisfaction. For example, different requesting servents may have different utility functions, which quantify the servents’ “degree of happiness” when they receive different amounts of

the transfer bandwidth resource. Lastly, an equal allocation strategy obviously gives no incentive for servents to contribute to their peers.

## 2.2. Incentive protocol: notations & desirable properties

Lack of incentive for sharing leads to the undesirable situation in which a servent behaves like a client most of the time [15]. The design of an incentive protocol for P2P networks is imperative. In such a protocol, the proper allocation of transfer bandwidth to requesting servents should be based on the servents' connection type, utility function, and contribution to the P2P community. Before we present our incentive protocol, we give the necessary notations:

$\Theta = (\theta_1, \theta_2, \dots, \theta_N)$ : A vector which represents the *connection type* (i.e., the upload and download capacities) of all the servents in the P2P network. In particular,  $\theta_i \in \Omega$  is the connection type of servent  $i$ , which is a function of  $i$ 's declared upload bandwidth  $u_i$  and download bandwidth  $d_i$ . The set  $\Omega$  represents all the possible connection types.

$C_i(t)$  represents the *cumulative contribution* of servent  $i$  at time  $t$ , where  $C_i(t) \in \{\mathbb{R}^+ \cup 0\}$ .

$x_i(t)$  represents the bandwidth allocated to servent  $i$  when  $i$  requests a file transfer. The bandwidth assignment is based on our incentive protocol.

$U_i(\theta_i, x_i)$ : A non-negative function which represents the utility of servent  $i$  when it declares its connection type to be  $\theta_i$  and receives a file transfer service rate of  $x_i$ .

Each servent in the system, say  $i$ , has a cumulative contribution value  $C_i(t)$  at time  $t$ . The value of  $C_i(t)$  will increase if servent  $i$  provides service to the community (e.g., by transferring files for other requesting servents). It may decrease if servent  $i$  requests some service from the community (e.g., requesting file transfers from other servents).

We now state some *desirable properties* of an incentive protocol. In later sections, we will prove that our proposed algorithms achieve these properties.

**(1) Conservation of the cumulative contribution and social utility:** The aggregate contribution of all servents at any time  $t > 0$  is equal to the aggregate cumulative utility of all servents from time 0 up to time  $t$ . Formally,

$$\sum_{i=1}^N C_i(t) = \sum_{i=1}^N \int_0^t U_i(\theta_i, x_i(\tau)) d\tau \quad \forall t > 0. \quad (1)$$

**Remark:** This property implies that the contribution by any servent in a P2P network via file transfer service is translated into utilities within the P2P community.

**(2) Maximization of social welfare through resource allocation:** Given a servent  $k$  and all the requesting clients in

$\mathcal{R}_k$ , if  $C_i(t) = C_j(t)$  for all  $i, j \in \mathcal{R}_k$ , an incentive protocol should allocate the transfer bandwidth resource of servent  $k$  so as to maximize the *social welfare*. Formally, maximizing the social welfare implies finding a transfer bandwidth vector  $\mathbf{x}(t) = [x_1(t), \dots, x_N(t)]$  such that

$$\mathbf{x}(t) = \arg \max \left( \sum_{i \in \mathcal{R}_k} U_i(\theta_i, x_i(t)) \right) \quad \text{s.t.} \quad \sum_{i \in \mathcal{R}_k} x_i(t) \leq u_k. \quad (2)$$

**Remark:** This property implies that the incentive protocol should maximize the aggregate utility (or "happiness") of all the requesting servents in  $\mathcal{R}_k$ .

**(3) Incentive-based resource distribution:** The protocol should provide incentive to rational users. Given a servent  $k$  and all the requesting clients in  $\mathcal{R}_k$ , we have two cases:

• **No Congestion:** If the aggregate download bandwidth at time  $t$  of all the requesting servents in  $\mathcal{R}_k$  is less than or equal to  $u_k$ , the upload bandwidth of servent  $k$ , then all servents in  $\mathcal{R}_k$  will receive a transfer bandwidth equal to their respective maximal download bandwidth such that they will achieve equal utility. Formally, if  $\sum_{i \in \mathcal{R}_k} d_i \leq u_k$ , then

$$\begin{aligned} x_i(t) &= d_i \\ U_i(\theta_i, x_i(t)) &= U_j(\theta_j, x_j(t)) \quad \forall i, j \in \mathcal{R}_k. \end{aligned} \quad (3)$$

**Remark:** This property implies that whenever servent  $k$  has sufficient resources, all the requesting servents should receive their maximal download bandwidth such that they are "equally happy".

• **Congestion:** When there is a congestion for servent  $k$  (i.e.,  $\sum_{i \in \mathcal{R}_k} d_i > u_k$ ), the transfer bandwidth allocation should be a function of the contribution and download bandwidth of all the requesting servents in  $\mathcal{R}_k$ . Formally, for any two servents  $i, j \in \mathcal{R}_k$ , if the ratio of contribution to download bandwidth of  $i$  is greater than or equal to that of  $j$ , servent  $k$  will distribute the transfer bandwidth resource such that the utility of servent  $i$  is greater than or equal to that of servent  $j$ . Formally,

$$\frac{C_i(t)}{d_i} \geq \frac{C_j(t)}{d_j} \implies U_i(\theta_i, x_i(t)) \geq U_j(\theta_j, x_j(t)). \quad (4)$$

**Remark:** This property implies that the incentive protocol should provide higher utilities to servents who have higher contributions per unit data request.

In the following, we present the operational setting of our incentive protocol.

## 2.3. Operational setting of incentive P2P protocol

The general setting in which our incentive protocol operates is as follows:

• Each servent declares its connection type to the P2P community, i.e., servent  $i$  has to declare its connection type of

$\theta_i$ . For our incentive protocol, the connection type of server  $i$  depends only on the upload ( $u_i$ ) and download bandwidths ( $d_i$ ).

• To provide fairness and incentive for a P2P community, the utility function, say for server  $i$ , takes on a *concave*, *bounded*, and *normalized* form. The utility function of server  $i$ , which depends on the download bandwidth  $d_i$  and the received transfer bandwidth  $x_i$ , takes the form:

$$U_i(\theta_i, x_i) = U_i(d_i, x_i) = \begin{cases} \log(\frac{x_i}{d_i} + 1) & \text{if } x_i \leq d_i \\ \log(2) & \text{if } x_i > d_i. \end{cases} \quad (5)$$

**Remark:** We take this form of utility function based on the following reasons: (a) A  $\log$  function is a general form of concave function which can represent a large class of elastic traffic [14] and this fits the file transfer service; (b) The utility function has an upper bound of  $\log(2)$ , which implies that once a server receives its maximum download bandwidth, they are *equally satisfied*; (c) The utility function has a value of zero if the received bandwidth  $x_i = 0$ ; (d) The utility function estimates the level of satisfaction given the ratio  $\frac{x_i}{d_i}$ , the amount of allocated bandwidth to the server's maximal download bandwidth.

We adopt concepts from *mechanism design* [10]. Under our incentive protocol:

(1) All servers have to declare their connection types. Hence, server  $i$  has a strategy  $g_i$  which can declare any connection type  $\theta \in \Omega$ , where  $\Omega$  is the set of all connection types in our incentive P2P system. For an honest server, which can be induced by a protocol having the truth revealing property, the strategy of server  $i$  should be  $g_i = \theta_i = (u_i, d_i)$ . That is, server  $i$  declares its real connection type.

(2) We interpret  $C_i(t)$ , the *contribution* of server  $i$ , as the virtual credit that server  $i$  has at time  $t$ . The proposed protocol will update the contributions of all the participating servers in any file transfer activity. In particular, the incentive protocol will increase the contribution when a server offers file transfer service for any requesting server, and it may reduce the contribution of a server who requests a file download. Particularly, the initial value of contribution is assigned to be zero which implies this server has not provided any service to others.

(3) The outcomes of the proposed protocol are (i) how much bandwidth is to be allocated to each requesting server, and (ii) the contribution updates for all participating servers.

### 3. Incentive Protocol for Distributing the Instantaneous Transfer Bandwidth

In this section, we describe the incentive protocol for allocating the instantaneous transfer bandwidth to requesting servers. For ease of discussion, we drop the time dependent notation; i.e., we use  $x_i$  instead of  $x_i(t)$ . Our incen-

tive protocol can achieve *efficiency* for social welfare. Furthermore, it provides *fairness* and *incentive* for sharing resources among all requesting servers. We first illustrate how the incentive protocol maximizes the social welfare. Then we generalize the concept and extend the protocol to include the contribution value of each requesting server.

#### 3.1. Protocol to maximize the social welfare

Consider a server  $k$  that is willing to offer its transfer bandwidth resource for use by a set of requesting servers  $\mathcal{R}_k$ . If all the servers in  $\mathcal{R}_k$  are of the same contribution (or if we ignore the contribution factor for the time being), maximizing the social welfare implies finding a transfer bandwidth vector  $\mathbf{y} = [y_1, \dots, y_{|\mathcal{R}_k|}]$  such that:

$$\begin{aligned} SW(u_k, \mathcal{R}_k) &:= \max \sum_{i \in \mathcal{R}_k} U_i(\theta_i, y_i) \\ \text{s.t. } \sum_{i \in \mathcal{R}_k} y_i &\leq u_k ; 0 \leq y_i \leq d_i \quad \forall i \in \mathcal{R}_k. \end{aligned} \quad (6)$$

Here,  $y_i$  is the allocated transfer bandwidth for server  $i$  in solving the above maximization problem. For our concave and bounded utility functions, we have the following equivalent optimization problem:

$$\begin{aligned} \max \prod_{i \in \mathcal{R}_k} (y_i + d_i) \\ \text{s.t. } \sum_{i \in \mathcal{R}_k} y_i &\leq u_k ; 0 \leq y_i \leq d_i \quad \forall i \in \mathcal{R}_k. \end{aligned} \quad (7)$$

One way to solve the optimization problem is to try to distribute the resource  $u_k$  such that the  $(y_i + d_i)$ s are as *even* as possible for all  $i \in \mathcal{R}_k$ . For instance, if we were without constraints, the solution should satisfy:

$$y_i + d_i = y_j + d_j \quad \forall i, j \in \mathcal{R}_k. \quad (8)$$

We use and enhance the progressive filling algorithm [4] to solve the above constrained optimization problem. Our progressive filling algorithm works as follows:

1. Treat a requesting client  $i \in \mathcal{R}_k$  as a water bucket with a capacity equals to  $2d_i$  and a height equals to  $2d_i$ .
2. Based on the values of  $d_i$ , sort all buckets in ascending order. For bucket  $i$ , the initial water level is  $d_i$ .
3. In addition, we have  $u_k$  amount of "water" (the resource) to distribute to all the buckets. We distribute the water such that the *maxmin fairness* property [4] holds.

One numerical example is shown in [8].

**Theorem 1** The progressive filling (PF) algorithm finds a solution to the bandwidth allocation problem which maximizes the social welfare in Eq. (6).

**Proof:** Please refer to technical report [8].

### 3.2. Incentive protocol

In the above discussion, we distribute the transfer bandwidth  $u_k$  among the requesting servents without considering their contribution values. We now extend the solution to include the contribution value of each requesting servent in order to provide incentive for contributing to the P2P community.

In maximizing the social welfare, Equation (8) implies that for any two requesting servents  $i, j \in \mathcal{R}_k$ , we distribute the transfer bandwidth resource such that  $(y_i + d_i)/(y_j + d_j) = 1$ . To provide incentive, we distribute the transfer bandwidth resource  $u_k$  of servent  $k$  such that the transfer bandwidth vector  $\mathbf{x} = [x_1, \dots, x_{|\mathcal{R}_k|}]$  satisfies

$$\frac{x_i + d_i}{x_j + d_j} = \left(\frac{C_i}{C_j}\right)^r \quad \forall i, j \in \mathcal{R}_k \quad (9)$$

where  $r$  is any nonnegative real number. Clearly this is a generalization of the problem of maximizing the social welfare. E.g., if all the requesting clients have the same contribution values (i.e.,  $C_i = C_j$  for  $i, j \in \mathcal{R}_k$ ), the above formulation is equivalent to Equation (8).

Based on the progressive filling (PF) algorithm given above, we propose an enhanced *contribution dependent* progressive filling (CDPF) algorithm. In essence, the new algorithm tries to satisfy Equation (9) among the requesting servents, if feasible. In doing so, the algorithm also maintains the maxmin fairness property. The CDPF algorithm works as follows:

- (1) Treat a requesting client  $i \in \mathcal{R}_k$  as a water bucket with a capacity equal to  $2d_i$  and a height equal to  $2d_i/(C_i)^r$ .
- (2) Based on the value  $d_i/(C_i)^r$ , sort all the buckets in ascending order. For bucket  $i$ , the initial water level is  $d_i/(C_i)^r$ .
- (3) Distribute  $u_k$  amount of water (the resource) to all the buckets. To fill each unit for bucket  $i$ , we consume  $(C_i)^r$  amount of water.

One numerical example is shown in [8].

**Theorem 2** For any two requesting servents  $i, j \in \mathcal{R}_k$ , the CDPF algorithm distributes the resource such that:

$$\frac{(C_i)^r}{d_i} \geq \frac{(C_j)^r}{d_j} \implies U_i(\theta_i, x_i) \geq U_j(\theta_j, x_j).$$

**Proof:** Please refer to technical report [8].

**Remark:** The significance of Theorem 2 is that our incentive protocol possesses the desired properties (2) & (3) given in Section 2.2.

The CDPF can be implemented by the following code:

---

**CDPF** ( $r, u_k$ , all requesting servents in  $\mathcal{R}_k$ )

1. **if** ( $\sum_{i \in \mathcal{R}_k} d_i \leq u_k$ ) **return**  $\mathbf{x}=\mathbf{d}$ ; /\*no congestion\*/

2. **sort**  $\{ \frac{d_i}{(C_i)^r}, \frac{2d_i}{(C_i)^r} | i \in \mathcal{R}_k \}$  **in ascending order. Store values and node index in array  $S$  and  $T$  respectively;**
3.  $i=1$ ; /\*initialize index variable\*/
4.  $level=d_{T[1]}$ ; /\*initialize feasible water level\*/
5.  $vol=C[T[1]]^r$ ; /\*unit height volume\*/
6. **do** {
7.  $i=i+1$ ;
8.  $nextLevel=S[i]$ ; /\*the next testing water level\*/
9. **if**  $((nextLevel-level)*vol \geq u_k)$  {
10.  $level=level+u_k/vol$ ;
11.  $u_k=0$ ;
12. }
13. **else**{
14.  $u_k=u_k-vol*(nextLevel-level)$ ;
15.  $level=nextLevel$ ;
16. /\*adjust the unit for filling operation\*/
17. **if** ( $S[i]$  is a lower bound of  $T[i]$ )  $vol=vol+(C_{T[i]})^r$ ;
18. **else**  $vol=vol-(C_{T[i]})^r$ ;
19. }
20. **while** ( $u_k > 0$ );
21. **for** (**each**  $i \in \mathcal{R}_k$ )
22. **if** ( $level > \frac{d_i}{(C_i)^r}$ )  $x_i=[level-\frac{d_i}{(C_i)^r}]*C_i^r$ ;
23. **return**  $\mathbf{x}$ ;

---

Our algorithm will assign the bandwidth  $x_i$  as equal to  $d_i$ , which is the maximal download bandwidth of servent  $i$ , for all the requesting servents if the aggregate maximal download bandwidth is less than or equal to the upload bandwidth resource (i.e.,  $\sum_{i \in \mathcal{R}_k} d_i < u_k$ ). Otherwise, our algorithm sorts all the lower bounds  $\{ \frac{d_i}{(C_i)^r} \}$  and upper bounds  $\{ \frac{2d_i}{(C_i)^r} \}$  in ascending order. Then it tests whether the amount of resource  $u_k$  can fill the buckets for reaching these bounds. We initialize the starting water level to be the minimum value of the sorted bounds. The initial marginal amount of water for filling a higher water level is  $(C[T[1]])^r$ , where  $T[1]$  is the index of the servent which has the minimum lower bound. Within each iteration of the do-while loop, we test whether the bound for the next bucket in  $S$  can be reached. If it can be reached, we first reduce the remaining resource in  $u_k$ , then we adjust  $(C[i])^r$  if servent  $i$ 's bound can be reached, and we assign the transfer bandwidth to all the eligible requesting servents. If the bound cannot be reached, the algorithm terminates and we have found the transfer bandwidth distribution.

**Theorem 3** The CDPF algorithm has computational complexity  $O(n \log(n))$ , where  $n$  is the number of requesting servents in  $\mathcal{R}_k$ .

**Proof:** Please refer to technical report [8].

## 4. Contribution Update

After each file transfer activity, we need to update the contribution values of all the participating servents. We give the physical meaning of contribution values and discuss how they should be updated. We will then present our contribution update algorithm and give its complexity analysis. Security issues for the contribution update will be discussed in a later section.

### 4.1. Social gain and social payment

In P2P networks, many factors can influence the contribution value of a given servent – e.g., shared storage or shared bandwidth. It is difficult to rank the importance of these factors. Furthermore, it is the shared *contents*, rather than the resources being shared, that attract download requests. Hence, the same amount of bandwidth or storage offered for sharing by two servents does not imply the same amount of contribution by these servents. Rather, we define a servent's contribution as the utility it can provide to the whole P2P community. When a servent, say  $k$ , transfers files for other servents, it gives utilities to the community. Therefore, we increase the contribution value of  $k$  by the social gain  $\mathcal{SG}_k(u_k)$ , which is defined as:

$$\mathcal{SG}_k(u_k) = SW(u_k, \mathcal{R}_k) = \max \sum_{i \in \mathcal{R}_k} U_i(\theta_i, y_i). \quad (10)$$

On the other hand, when a servent  $i \in \mathcal{R}_k$  receives transfer bandwidth of  $x_i$  based on the CDPF algorithm, the  $x_i$  value may not be equal to  $y_i$ , which is the solution of the PF algorithm for maximizing the social welfare. Therefore, we define  $\mathcal{SP}_i(x_i)$  as the *social payment* for servent  $i$  when it receives a transfer bandwidth of  $x_i$ :

$$\mathcal{SP}_i(x_i) = SW(u_k, \mathcal{R}_k) - [U_i(\theta_i, x_i) + SW(u_k - x_i, \mathcal{R}_k - \{i\})] \quad (11)$$

The physical meaning of  $\mathcal{SP}_i(x_i)$  is the difference between the maximum aggregate utility under social welfare resource distribution (as solved by the PF algorithm) and the maximum aggregate utility under contribution dependent resource distribution (as solved by the CDPF algorithm). In other words, if  $x_i$  bandwidth is assigned to servent  $i$  based on the CDPF algorithm, the P2P community will not receive maximum social welfare. Hence, servent  $i$  should pay for this difference, and we deduct the payment amount from servent  $i$ 's contribution value.

### 4.2. Instantaneous contribution update

When a servent, say  $k$ , provides its transfer bandwidth for use by the P2P community, its contribution value is increased by  $SW(u_k, \mathcal{R}_k)$ . This increase is equal to the maximum

social welfare. On the other hand, the aggregate utility received by the requesting servents in  $\mathcal{R}_k$  is not equal to  $SW(u_k, \mathcal{R}_k)$ . The reason for the difference is that some requesting servents may receive more transfer bandwidth under the CDPF algorithm, as compared with the PF algorithm. Such a servent needs to make a social payment equal to the extra bandwidth received.

In our contribution update mechanism, we compare the resource allocation,  $x_i$ , under CDPF with the resource allocation,  $y_i$ , under PF for servent  $i$ . We choose a servent who obtains the largest amount of extra bandwidth  $x_i - y_i$  and reduce that servent's contribution by  $\mathcal{SP}_i(x_i)$ . The process is repeated until the solution of the CDPF algorithm equals the solution of the PF algorithm.

An example for contribution update is shown in [8].

**Theorem 4** The cumulative contribution is conserved to be equal to the total social utility at all time. That is,

$$\sum_{i=1}^N C_i(t) = \sum_{i=1}^N \int_0^t U_i(\theta_i, x_i(\tau)) d\tau \quad \forall t > 0.$$

**Proof:** Please refer to technical report [8]. Our contribution update mechanism is based on a fluid model. In implementation, we divide time into quanta denoted as  $\Delta t$ . At the beginning of each time quantum, we assign the transfer bandwidth using the CDPF algorithm. At the end of each time quantum, we update the contribution values of servent  $k$  and the servents in  $\mathcal{R}_k$ . The pseudo-code for the contribution update is:

#### Contribution Update ( $r, u_k$ , all requesting servents in $\mathcal{R}_k$ )

1.  $\mathbf{x}$  = CDPF( $r, u_k$ , all requesting servents in  $\mathcal{R}_k$ );  
/\*  $\mathbf{x}$  is the solution of CDPF algorithm \*/
2.  $\mathbf{y}$  = PF( $u_k$ , all requesting servents in  $\mathcal{R}_k$ );  
/\*  $\mathbf{y}$  is the solution of PF algorithm \*/
3.  $C_k = C_k + (\sum_{i \in \mathcal{R}_k} \log(\frac{y_i}{d_i} + 1)) * \Delta t$ ;  
/\* the resource owner  $k$  increases its contribution by the social gain \*/
4. **do** {
5.  $q = \arg \max \{x_i - y_i\}$ ;  
/\*  $q$  is the servent who will reduce its contribution in this iteration \*/
6. **if** ( $x_q - y_q > 0$ ) {
7.  $\mathcal{SP} = \sum_{i \in \mathcal{R}_k} \log(\frac{y_i}{d_i} + 1)$ ;
8.  $\mathcal{R}_k = \mathcal{R}_k - \{q\}$ ;
9.  $u_k = u_k - x_q$ ;
10.  $\mathbf{y} = \text{PF}(u_k, \text{all requesting servents in } \mathcal{R}_k)$ ;
11.  $\mathcal{SP} = \mathcal{SP} - (\log(\frac{x_i}{d_i} + 1) + \sum_{i \in \mathcal{R}_k} \log(\frac{y_i}{d_i} + 1))$ ;  
/\* servent  $q$  reduces its contribution by its social payment \*/
12.  $C_q = C_q - \mathcal{SP} * \Delta t$ ;
13. }
14. } **while** ( $x_q - y_q > 0$ );

In line 3, the algorithm increases the contribution of the resource owner by the social gain  $S\mathcal{G}$ . The contributions of the requesting servents are reduced in the do-while loop. In line 5, we choose the servent  $q$  who gains the most extra resource  $x_i - y_i$ . From lines 7 to 11, we compute the social payment for servent  $q$  and adjust the remaining amount of resource for the remaining requesting servents. In line 12, we decrease the contribution of  $q$  by its social payment.

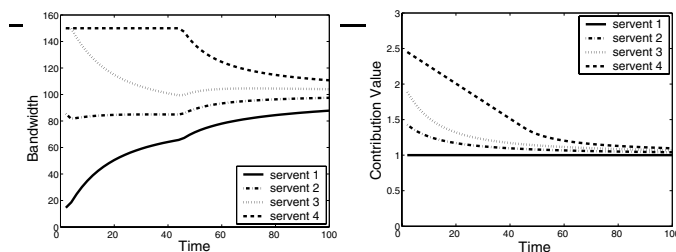
**Theorem 5** The contribution update algorithm has computational complexity  $O(n^2 \log(n))$ , where  $n$  is the number of requesting servents.

**Proof:** Please refer to technical report [8].

## 5. Experimental Results

In here, we present simulation results showing that our mechanism can fairly distribute transfer bandwidth among the requesting servents and can provide higher aggregate utility than other scheduling disciplines like FCFS and processor-sharing.

**Experiment A: servents with similar connection type but different contribution values:** Four servents make requests to servent  $k$ , which has a transfer resource of  $u_k = 400$ . The contribution values of these requesting servents at time  $t_0$  are  $[C_1, C_2, C_3, C_4] = [1, 1.5, 2, 2.5]$ . The connection types of all the requesting servents are the same and their maximal download bandwidth are  $d_1 = d_2 = d_3 = d_4 = 150$ . Each simulation lasts 100 units of time in the interval of  $[t_0 \leq t \leq t_0 + 100]$ . Figure 2 illustrates the bandwidth assignment  $x_i(t)$  and their respective contribution values  $C_i(t)$  during the simulation period.

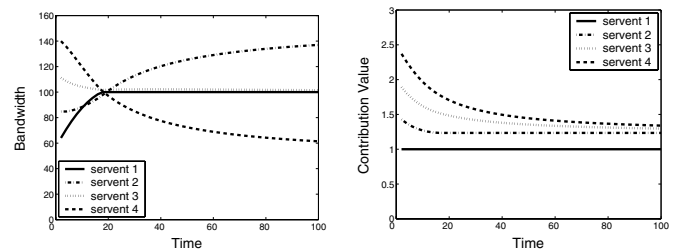


**Figure 2. (a) Instantaneous bandwidth assignment and (b) instantaneous contribution values for competing servents.**

In experiment A, we observe that although all the servents have the same connection type, CDPF algorithm assigns *higher bandwidth* to the servents which have larger contribution values. At the same time, our mechanism decreases the contribution values of these servents (since they

are getting more social resources). Finally, all servents tend to have equal contribution values and equal bandwidths, which maximize the aggregate utility among all servents.

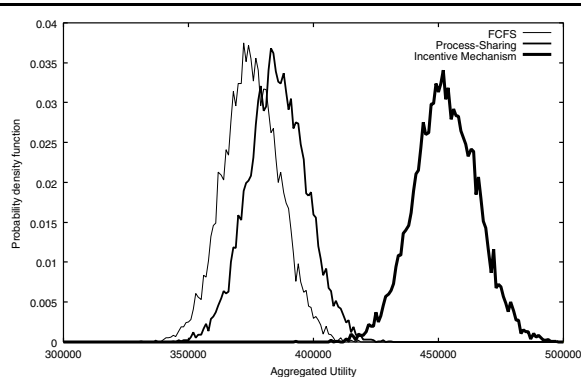
**Experiment B: servents with different connection types and contribution values:** The connection types of the requesting servents are different and they are  $d_1 = 100, d_2 = 150, d_3 = 200, d_4 = 250$ , respectively. All the other settings are the same as in Experiment A. For time  $t_0 \leq t \leq t_0 + 100$ , we illustrate the bandwidth assignment  $x_i(t)$  and contribution  $C_i(t)$  in Figure 3. In experiment B, we



**Figure 3. a) Instantaneous bandwidth assignment and (b) instantaneous contribution values for competing servents.**

observe that our instantaneous bandwidth assignment also converges to the solution of the PF algorithm, which provides maximized aggregate utility for the P2P network. Initially, servent 1 gains less bandwidth than the solution of the PF algorithm. Therefore, the CDPF algorithm does not decrease the contribution of servent 1. For servent 2, 3, and 4, they gain a larger bandwidth than by the PF algorithm at the beginning. Afterwards, their contribution values converge to the same value. So, their instantaneous bandwidth converges to the solution of the PF algorithm.

**Experiment C: achieved utility under different resource distribution algorithms:** We compare the efficiency of our incentive mechanism with that of the FCFS and processor-sharing disciplines. The average file transfer request rate matrix,  $\Lambda$ , is randomly generated in 10,000 experiments. There are fifty servents and they can make requests to each other. There are five different connection types and each servent has an equal probability of being any of the connection type. The file request rate and the file service rate are Poisson. Under the FCFS discipline, there are at most five servents receiving service at the same time. Any further requests are queued and served in FCFS order. Under the processor sharing discipline, each requesting servent gets an equal share of the available bandwidth from the provider servent. The distribution for the incentive mechanism is as described above. The probability density function for the aggregate utility under these three resource distribu-



**Figure 4. Probability density function for aggregate utility under FCFS, process-sharing and incentive mechanism.**

tion algorithms are illustrated in Figure 4. The x-axis is the value of the aggregate utility and the y-axis is the frequency achieving the value of aggregate utility. The proposed incentive mechanism *always* gives a higher aggregate utility than the other algorithms.

## 6. Related Work

In [6], the authors address one possible mechanism for Napster-like P2P network. Our work is different from theirs in the sense that our mechanism uses “virtual credit” so that it will not reduce the willingness of users to participate in a P2P network. In [5], the authors discuss the economic behavior of P2P storage networks. In [17], the authors model P2P networks as a Cournot Oligopoly game and give elegant control-theoretical solution focusing on global storage system. Our work focuses on the file-transfer and bandwidth allocation of a P2P system and we use the mechanism design approach in designing a competitive game in a P2P system. Lastly, algorithmic mechanism design [9, 10, 15] provides a theoretical framework for designing incentive mechanisms.

## 7. Conclusion

We have presented an incentive mechanism for P2P networks. Our mechanism distributes resources among servers based on each server’s utility function, connection type, and contribution. Our mechanism achieves both higher aggregate utility and fairness for a P2P network. Under our mechanism, the contribution value of a server will be increased if it provides service to the P2P community. A server who has a larger contribution value will receive a higher utility when it competes with other servers for file download services. Therefore, servers in the community have incentive to share information, thereby resolving the

free-riding problem. Furthermore, our mechanism may decrease the contribution values of servers who access a congested resource. Therefore, it also provides incentive for servers to access information from non-congested servers and resolves the tragedy of the commons problem.

**Acknowledgement** J. C. S. Lui was supported in part by an RGC Earmarked Grant. D. K. Y. Yau was supported in part by the NSF under grant number CCR-9875742 (CAREER), and in part by CERIAS.

## References

- [1] The Gnutella Protocol Spec. v0.4 1, document revision 1.2.
- [2] Limewire : a Gnutella client software.
- [3] E. Adar and B. Huberman. Free riding on Gnutella. *Technical report, Xerox PARC, 10 Aug. 2000. FirstMonday*.
- [4] J.-Y. Boudec. Rate adaptation, congestion control and fairness: A tutorial, <http://icapeople.epfl.ch/leboudec>.
- [5] A. C. Fuqua, T. Ngan, and D. S. Wallach. Economic behavior of peer-to-peer storage networks. *To appear in Workshop on Economics of Peer-to-Peer Systems (Berkeley, California), June 2003*.
- [6] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge. Incentives for sharing in peer-to-peer networks. *Proceedings of the 2001 ACM Conference on Electronic Commerce*.
- [7] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.
- [8] R. Ma, S. Lee, J. Lui, and D. Yau. Incentive p2p networks: Theory and implementation. *Technical report of Dept. of CSE, Chinese University of Hong Kong*.
- [9] N. Nisan and A. Ronen. Algorithmic mechanism design. *In Proc. 31st Annual Symposium on Theory of Computing*.
- [10] D. Parkes. Chapter 2, Iterative Combinatorial Auctions: Achieving economic and computational efficiency ph.d. dissertation, university of pennsylvania. May, 2001.
- [11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. *In Proc. of ACM SIGCOMM, 2001*.
- [12] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. *CS Dept., University of Chicago, 2001*.
- [13] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in CS*, 2218:329–350, 2001.
- [14] S. Shenker. Fundamental design issues for the future internet. *IEEE Journal on Selected Areas in Communication*, 13(7), September 1995.
- [15] J. Shneidman and D. Parkes. Rationality and self-interest in peer to peer networks. *Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.
- [16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM, 2001*.
- [17] W. Wang and B. Li. To play or to control: a game-based control-theoretic approach to peer-to-peer incentive engineering. *in the Proceedings of the Eleventh International Workshop on Quality of Service (IWQoS 2003)*.