An Efficient Algorithm for Resource Sharing in Peer-to-Peer Networks

Wei-Cherng Liao, Fragkiskos Papadopoulos, Konstantinos Psounis
E-mail: weicherl, fpapadop, kpsounis@usc.edu

Technical Report # CENG-2005-15

Electrical Engineering Department
University of Southern California
Los Angeles, CA 90089
November 14, 2005

An Efficient Algorithm for Resource Sharing in Peer-to-Peer Networks

Wei-Cherng Liao, Fragkiskos Papadopoulos, and Konstantinos Psounis

University of Southern California, Los Angeles, CA 90089
`E-mail: weicherl, fpapadop, kpsounis@usc.edu`

Abstract. The performance of peer-to-peer systems depends on the level of cooperation of the system's participants. While most existing peer-to-peer architectures have assumed that users are generally cooperative, there is great evidence from widely deployed systems suggesting the opposite. To date, many schemes have been proposed to alleviate this problem. However, the majority of these schemes are either too complex to use in practice, or do not provide strong enough incentives for cooperation.
In this work we propose a scheme based on the general idea that offering uploads brings revenue, and performing downloads has a cost. We also introduce a theoretical model that predicts the performance of the system and computes the values of the scheme's parameters that achieve a desired performance. Our scheme is quite simple and very easy to implement. At the same time, it provides very strong incentives for cooperation and improves the performance of P2P networks significantly. In particular, theory and realistic simulations show that it reduces the query response times and file download delays by one order of magnitude, and doubles the system's throughput.

1 Introduction

Peer-to-peer (P2P) systems provide a powerful infrastructure for large-scale distributed applications, such as file sharing. While cooperation among the system's participants is a key element to achieve good performance, there has been growing evidence from widely deployed systems reporting that peers are usually not cooperative. For example, a well known study of the Gnutella file sharing system in 2000 reveals that almost 70% of all peers only consume resources (download files), without providing any files to the system at all [1]. This phenomenon is called "free-riding".

Despite the fact that this phenomenon was identified several years ago, recent studies of P2P systems show that the percentage of free-riders has significantly increased [2]. This is not because industry and academia have ignored the problem. There is a large body of work on incentive mechanisms for P2P networks, varying from centralized and decentralized credit-based mechanisms, *e.g.* [3–6], to game-theoretic approaches and utility-based schemes, *e.g.* [7, 8], to schemes that attempt to identify and/or penalize free-riders, *e.g.* [9–11], the last two being proposed by the popular KaZaA and eMule systems. The problem of free-riders is hard to tackle because the solution has to satisfy conflicting requirements: minimal overhead, ease of use, and at the same time good amount of fairness and resilience to hacking.

In this paper we propose and study the performance of a simple algorithm where users use tokens as a means to trade *bytes* within the system. All users start with an initial number of tokens. A user earns K_{up} tokens for each *byte* he/she uploads to the system and spends K_{down} tokens for each *byte* he/she downloads from the system. The user also gains K_{on} tokens for each second his/her machine is on the system (*i.e.* it is online). A user can initiate a download only if the number of tokens that he/she has is large enough to download the complete file. Despite its simplicity, this scheme can provide very strong incentives for cooperation, since the amount of bytes a user downloads from the system is directly related to the amount of bytes he/she uploads to the system. The exact relation can be determined by tuning the parameters K_{on}, K_{up}, K_{down} as desired, which makes the scheme quite flexible.

The proposed algorithm relies on the general idea that users should be awarded for offering uploads and staying online, and be penalized for performing downloads. While others have proposed solutions that use the same idea in the past, *e.g.* [6, 8], there are a number of questions that either have not been addressed or have been studied only via simulations: (i) What is the right value for the parameters K_{on}, K_{up}, K_{down}? More general, how should one tune the parameters that dictate the gain from uploads and loss from downloads? (ii) What is the exact effect of such an algorithm on performance over a wide range of parameter values? (iii) Would a *small* number of free-riders, that manage to subvert the scheme, degrade overall performance noticeably? (iv) Is it possible to trade off one performance metric for another by playing with the parameters, e.g. trade off download delay for total system capacity? Our theoretical analysis of the performance of the resulting system, coupled with extensive realistic simulations, gives concrete answers to all these questions. Interestingly enough, it shows that the query response times and file download delays can be reduced by one order of magnitude while being able to sustain higher user download demands.

An important aspect of any solution to the free-riding problem is if the user's participation information is determined and maintained locally and without any interaction with the other peers of the system (localized solutions), or it is determined and maintained by either a centralized authority (non-localized centralized solutions), or by the continuous exchange of information among the system's participants (non-localized distributed solutions). Localized solutions are simple and impose very little overhead but they are easy to subvert. Non-localized solutions are hard to subvert but are complex to use in practice. (Most existing systems, such as KaZaA, eMule, and BitTorrent, all adopt localized approaches.) In any case, our proposed scheme can be easily implemented in any way, and we describe later in the paper how to implement it efficiently with each one of the approaches.

The rest of the paper is organized as follows: In Section 2 we briefly discuss prior work on providing incentives for P2P systems through localized and non-localized solutions. In Section 3 we provide a detailed description of the proposed scheme. In Section 4 we provide a set of equations that relates the parameters of the scheme, and which can be used to predict important performance metrics. In Section 5 we present realistic experiments with P2P systems on top of TCP networks. In Section 6 we briefly present some implementation thoughts and finally conclude in Section 7.

2 Related Work

There has been a large body of work on incentive mechanisms for P2P networks. Three of the most popular localized schemes are the ones implemented by the eMule [11], the KaZaA [12], and the BitTorrent [13] systems. eMule rewards users contributing to the network by reducing their waiting time in the upload queue based on a scoring function (called *QueueRank*). Similarly, in KaZaA, each peer announces its *Participation Level*, computed locally as a function of download and upload volumes, and peers that report high participation levels get higher priority [10]. A disadvantage of both KaZaA and eMule is that they provide relatively weak incentives for cooperation since peers that have not contributed to the system at all can still benefit from it, if they are patient enough to wait in the upload queues. Other problems include that they favor users with high access bandwidth, which may result in frustration or a feeling of unfairness [14], and that they are vulnerable to the creation and distribution of hacked daemons that do not conform to the desired behavior [15]. BitTorrent uses a different scheme that is specific to its architecture. Each peer periodically stops offering uploads to its neighbors that haven't been offering uploads to him/her recently. This scheme is hard to subvert. However, it suffers from some unfairness issues and it only works with "BitTorrent-style" systems, that is, in systems where files are broken into pieces, users are grouped into swarms by the file of interest, and downloading a file involves being connected to almost all of one's neighbors in order to collect and reassemble all the pieces of the file.

Non-localized proposals are primarily concerned with creating systems that cannot be subverted. Some of them make use of credit/cash-based systems. They achieve protection from hackers by either using central trusted servers to issue payments (centralized approach), *e.g.* [3, 4], or by distributing the responsibility of transactions to a large number of peers (distributed approach), *e.g.* [6]. Other distributed approaches use lighter-weight exchanged-based mechanisms, *e.g.* [16], or reputation management schemes, *e.g.* [5]. These mechanisms are indeed hard to subvert but they are also quite complex to use in practice.

In this paper we decouple the issue of how to design an algorithm to prevent free-riding from the issue of how to implement this algorithm in a P2P system. We first propose an efficient scheme that provides very strong incentives for cooperation. We show this via both theory and simulations. Then, we show that the scheme is generally applicable to any P2P system and comment on how to implement it using either a localized or a non-localized approach. Our main contribution is the theoretical analysis of the performance of the P2P system with and without the proposed scheme. The analysis yields a set of equations that are used to tune the parameters of the scheme, which results in improving the performance of the system by one order of magnitude.

3 A Simple and Effective Algorithm

As mentioned earlier, the algorithm uses tokens as a means to trade bytes within the system. Each user is given an initial number of tokens M when he/she *first* joins the network. This allows new users to start downloading a small number of files as soon as they join the system. (Note that this is not a requirement for the scheme to work.) When a user *rejoins* the system he/she uses the amount of tokens he/she previously had.

Users spend K_{down} tokens for each byte they download from the system and earn K_{up} tokens for each byte they upload to the system. This forces users to offer files for upload proportionally to the number of files they want to download. Further, users gain K_{on} tokens/sec while being online. This mechanism of accumulating tokens serves two purposes. First, it allows users who are not contacted frequently for uploads to gain tokens by just being online, which is more fair towards users with low access bandwidth [14]. Second, it provides an incentive for users to keep their machines on the system even when they are not downloading a file, which helps to prevent the so-called problem of low "availability" [17]. Note that the value of K_{on} should be relatively small, in order to prevent users from gaining enough tokens by just keeping their machines on without providing any uploads. Finally, a user can initiate a download only if the number of tokens he/she currently possesses is greater or equal to the number of tokens required to download the requested file.

This scheme provides strong and direct incentives for cooperation. Everyone understands what *"You can currently download 1MB of data"* means. Hence, it is expected to alter the usual behavior of free-riders significantly. In particular, free-riders will be *"forced"* to provide some uploads to the system, as an attempt to gain tokens fast enough in order to sustain their desirable download demands. This is in contrast to previous schemes, for example [10, 11], where it is unclear how exactly they affect free-riders who do not provide *any* uploads to the system.

Under the proposed scheme some free-riders may decide to share their files as soon as they are out of tokens. Others, may adopt a more "dynamic" behavior and decide to adjust the number of uploads they provide to the system as a function of the number of tokens they currently have. A common characteristic of all these behaviors is that they provide a significant amount of resources to the system, which was "invisible" before. Since the entire population of free-riders will deviate from their usual selfish behavior, and since the vast majority of users in deployed systems are free-riders, P2P system performance is expected to improve tremendously. This is verified in Section 5.

4 A Mathematical Model For The Proposed Scheme

Choosing the appropriate values for K_{on}, K_{up}, and K_{down} is an important issue in the proposed scheme. An arbitrary setting of these parameters may lead to several undesired situations. For example, giving a large value to K_{on} may provide tokens for the free-riders fast enough, so that there won't be any incentive for them to start sharing their files with the system. In this case, the scheme won't have a significant effect on system's performance. As another example, giving relatively small values to both K_{on} and K_{up} may reduce the token accumulation rate of cooperative users so much that they can't sustain their download demands.

Further, another important issue is how to predict important performance metrics given the values of K_{on}, K_{up} and K_{down}, without relying on experiments that can be either too expensive or unrealistic. Therefore, it is crucial to have a mathematical model that can be used to tune the parameters of the scheme and to predict the system's performance.

4.1 System Dynamics

We assume a system that implements the proposed scheme is which we call "system with the tokens". Recall that K_{down} and K_{up} are expressed in tokens/byte and K_{on} in tokens/sec. Now, let C_{down} and C_{up} denote the user's file download and upload speeds respectively, both expressed in bytes/sec. A user spends $K_{down}C_{down}dt$ tokens if he/she is downloading files from other peers during time $(t, t + dt)$. The user earns $K_{on}dt$ tokens if he/she is online during time $(t, t + dt)$ and $K_{up}C_{up}dt$ tokens if other users are uploading files from the user under study during time $(t, t + dt)$. Let $T(t)$

denote the number of the user's tokens at time t, with $T(0) \geq 0$. We can then write the following differential equation:

$$\frac{dT(t)}{dt} = K_{on} I_{on}(t) + K_{up} C_{up} I_{up}(t) - K_{down} C_{down} I_{down}(t), \tag{1}$$

where

$$I_{on}(t) = \begin{cases} 1 & \text{if the user is online in (t, t+dt)} \\ 0 & \text{otherwise} \end{cases},$$

$$I_{up}(t) = \begin{cases} 1 & \text{if the user is online and provides uploads in (t, t+dt)} \\ 0 & \text{otherwise} \end{cases},$$

$$I_{down}(t) = \begin{cases} 1 & \text{if the user is online and performs downloads in (t, t+dt)} \\ 0 & \text{otherwise} \end{cases}.$$

Taking expectations on both sides of Equation (2), and interchanging the expectation with the derivative on the left hand side[1], we get:

$$\frac{dE[T(t)]}{dt} = K_{on} P_{on}(t) + K_{up} C_{up} P_{up}(t) - K_{down} C_{down} P_{down}(t), \tag{2}$$

where $P_{on}(t)$ is the probability that the user is online at time t, $P_{up}(t)$ is the probability that the user provides uploads to the system at time t, and $P_{down}(t)$ is the probability that the user performs downloads from the system at time t. Note that Equation 2 can be regarded as a fluid model describing the token dynamics.

$P_{on}(t)$, $P_{up}(t)$, and $P_{down}(t)$ depend on how the user behaves given the number of tokens that he/she has at some point in time, and on his/her download demands. Along these lines, one can define user profiles and solve the differential equations. However, modeling a user's behavior is still challenging in current research community and it is out of the scope of this paper. In the next section, we employ a simple user profile to demonstrate the usefulness of Equation (2).

4.2 A User Profile: Linear Model

In general, free-riders will be motivated to provide uploads to the system when they do not have enough tokens to sustain their download demands and they may lose their willingness to provide uploads as the amount of tokens they possess is larger than the amount of tokens they need. Therefore we assume users will provide uploads to the system with a high probability, say P_{upmax}, when they are short of tokens. After their tokens reach a threshold, T_{th1}, their willingness to keep providing upload service will be decreased. And the probability that they are willing to provide uploads to the P2P system is linearly decreasing as the amount of tokens they possess increases. This behavior model is depicted in Figure 1 and characterized by Equation (3).

$$P_{up}(t) = \begin{cases} P_{upmax} & \text{if T(t)} \leq T_{th1}, \\ P_{upmax}\left(1 - \frac{P_{upmax} - P_{upmin}}{P_{upmax}} \frac{T(t) - T_{th1}}{T_{th2} - T_{th1}}\right) & \text{if } T_{th1} \leq \text{T(t)} \leq T_{th2}, \\ P_{upmin} & \text{if T(t)} \geq T_{th2}. \end{cases} \tag{3}$$

[1] Taking into account that $T(t)$ is bounded in practice, we can use the bounded convergence theorem to justify the interchange.

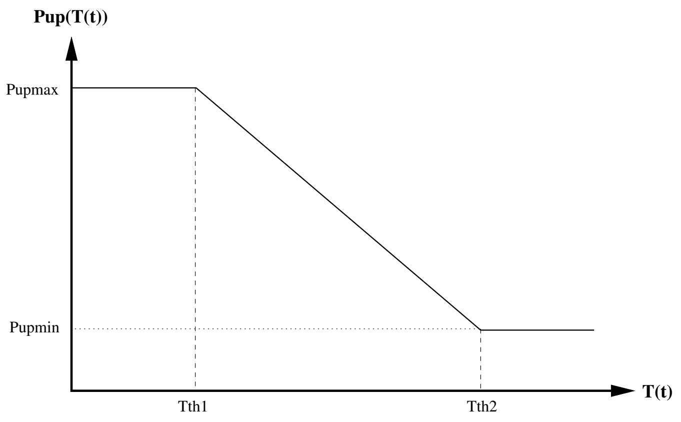

Fig. 1. Linear function of P_{up}.

For ease of exposition, let $P_{on}(t) = 1$ and $P_{on \cap down}(t) = P_{down}$. Equation (3) along with Equation (2) yield:

$$\frac{dE[T](t)}{dt} = \begin{cases} K_{on} + C_1 - C_2 & \text{if } \tilde{T}(t) \leq T_{th1}, \\ K_{on} + C_1 \left(1 - C_3 \dfrac{T(t) - T_{th1}}{T_{th2} - T_{th1}}\right) - C_2 & \text{else.} \end{cases} \tag{4}$$

where

$$\begin{aligned} C_1 &= K_{up} C_{up} P_{upmax} \\ C_2 &= K_{down} C_{down} P_{down} \\ C_3 &= \frac{P_{upmax} - P_{upmin}}{P_{upmax}}. \end{aligned}$$

We can solve this differential equation with an initial condition: $\tilde{T}(0) = 0$, having:

$$E[T](t) = \begin{cases} (K_{on} + C_1 - C_2)t & \text{if t} \leq t_0, \\ \dfrac{(C_4 T_{th1} + K_{on} + C_1 - C_2) e^{C_4(t-t_0)} + C_2 - K_{on} - C_1}{C_4 e^{C_4(t-t_0)}} & \text{else} \end{cases} \tag{5}$$

where

$$\begin{aligned} C_4 &= \frac{K_{up} R_{down} (P_{upmax} - P_{upmin})}{(T_{th2} - T_{th1})} \\ t_0 &= \frac{T_{th1}}{K_{on} + K_{up} C_{up} P_{upmax} - K_{down} P_{down} C_{down}} \end{aligned}$$

We plug values into Equation (5) to study the dynamics of users' tokens. In Figure 2 we can observe that the user's tokens accumulate linearly at the beginning, which represents the first part of Equation 5 (where $t < t_o$). After the user's tokens reach T_{th1} (at time t_o), the user decreases their willingness to provide upload and hence the token accumulating rate start decreasing. Finally, the user adapts to the equilibrium at which the token consuming rate equals to the token earning rate and thus the user's tokens converge to a steady state.

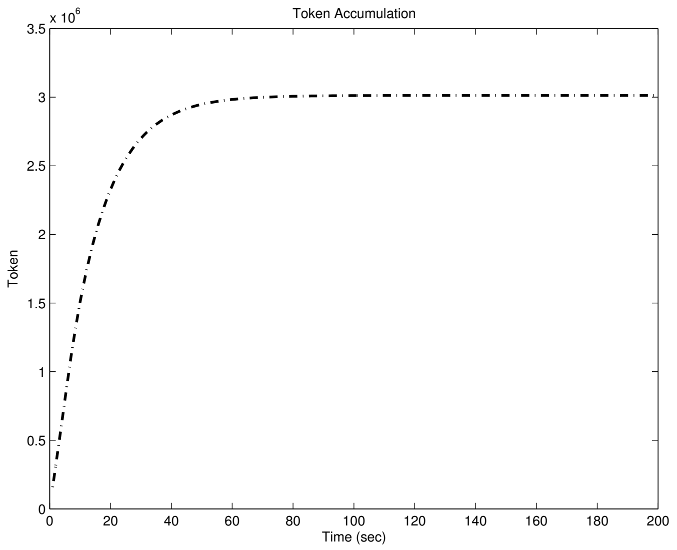

Fig. 2. Token accumulation in linear model.

Since users' tokens will reach the steady state, we are interested in analyzing the steady state of Equation 4. The steady state can be obtained by setting Equation (4) equal to zero. As a result, we have:

$$\begin{aligned} T_{ss} &= \frac{(K_{on} + C_1 - C_2)(T_{th2} - T_{th1})}{C_1 C_3} + T_{th1} \\ &= \frac{(K_{on} + K_{up}C_{up}P_{upmax} - K_{down}P_{down}C_{down})(T_{th2} - T_{th1})}{K_{up}C_{up}P_{upmax}(P_{upmax} - P_{upmin})} + T_{th1} \end{aligned} \tag{6}$$

and

$$P_{upss} = \frac{K_{down}C_{down}P_{down} - K_{on}}{K_{up}C_{up}} \tag{7}$$

Figure 3 shows the effect of choosing different values of K_{up} in the proposed scheme. Given all other parameters fixed, we can observe from both Equation 6 and Figure 3 that T_{ss} will increase and thus P_{upss} decreases as K_{up} increases. This phenomenon is a natural consequence of the selfish behavior of free-riders. As K_{up} increases, a user can earn more tokens when he/she provides the same volume of uploads and thus his/her willingness to provide uploads to the system will be decreased. As a result, P_{upss} will converge to a smaller value and T_{ss} will converge to a higher value as K_{up} increases.

The purpose of giving users K_{on} tokens when they stay on the system had been discussed in Section 3. Obviously, a small value of K_{on} can not give users strong incentives to keep their machines online. However, setting K_{on} to a large number is neither a good design because this will void the proposed scheme. The effect of using different values of K_{on} in the scheme is shown in Figure 4. We can observe that the effect of choosing different values of K_{on} is subtle when K_{on} is comparatively small ($K_{on} = 1$ or $K_{on} = 1000$). This is because users have to provide considerable uploads to the system in order to gain tokens to sustain their download demands when K_{on} is much smaller than the token earning rate of providing uploads to the system,

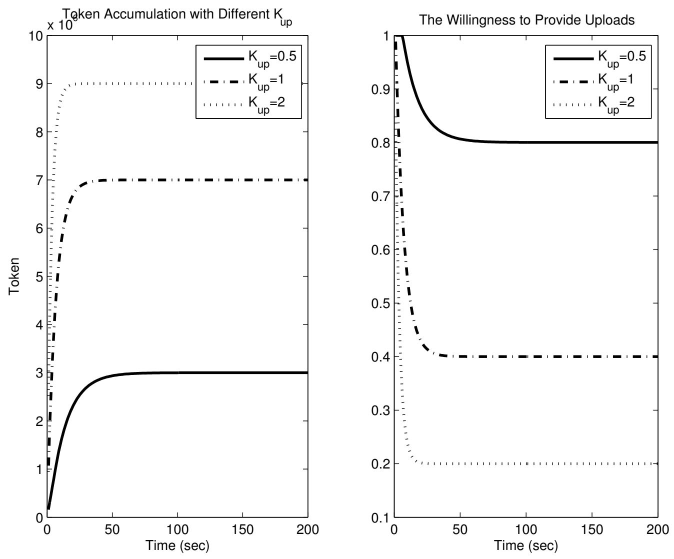

Fig. 3. Effects of setting different values to K_{up} in linear model.

(i.e. $K_{up}C_{up}$). However, if we assign a large value to K_{on}, users can gain tokens fast even without providing upload service to the system. Consequently, free-riders are no longer motivated to provide uploads to the system.

4.3 Steady States

Given a user's profile, we can completely study the dynamics and the steady state of the system. However, realistic user's profiles are still mysterious to the research society. Fortunately the existence of a steady-state of the proposed scheme can be easily justified by taking into consideration that in the long-run a typical user will spend as many tokens as he/she gains. Therefore, we study in general the steady state of the proposed scheme in this section.

The steady states of Equation 2 can be studies by setting $\frac{dE[T(t)]}{dt} = 0$ and dropping the time dependence from the probabilities in the qquation. Without loss of generality assume $P_{on} = 1$. Let R_{up} be the long-run average rate of file upload requests per second that the user handles, which we refer to as the *upload rate*. Also, let R_{down} be the long-run average rate of file download requests per second that the user initiates, which we refer to as the *download rate*. Last, let S denote the average file size in the system in bytes. Then, it is easy to see that $P_{up} = \frac{R_{up}S}{C_{up}}$ and $P_{down} = \frac{R_{down}S}{C_{down}}$. Equation (2) in steady state gives:

$$K_{on} + K_{up}R_{up}S - K_{down}R_{down}S = 0$$

Taking the average over all users yields:

$$K_{up} = K_{down}\left(\frac{E[R_{down}]}{E[R_{up}]}\right) - \frac{K_{on}}{E[R_{up}]S}. \quad (8)$$

Note that the same formula holds when the averaging is over free-riders only, in which case the rates are denoted by $E[R_{up}|FR]$ and $E[R_{down}|FR]$, and when the

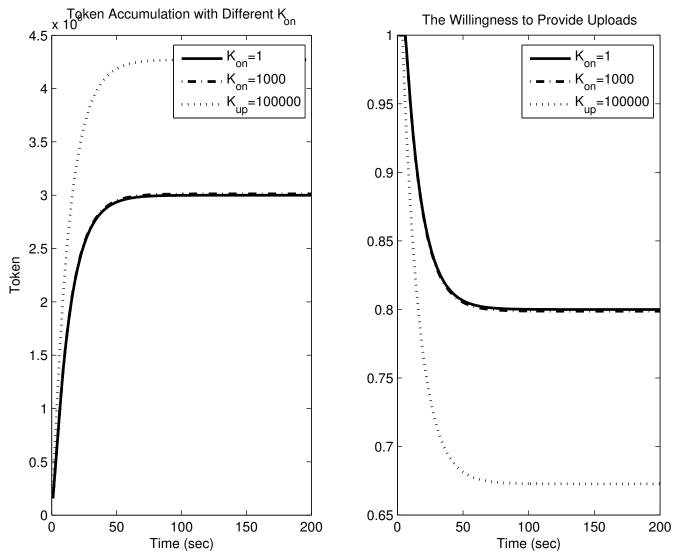

Fig. 4. Effects of choosing different value of K_{on} in linear model.

average is taken over over non-freeriders only, in which case the rates are denoted by $E[R_{up}|NF]$ and $E[R_{down}|NF]$. Equation 8 relates the parameters of the scheme K_{on}, K_{up}, and K_{down}, with the average download and upload activity of the users. We will later use it to select the parameter values that yield a target performance. But first, we need to compute the average download and upload rates, which is the next topic. In addition the performance improvement of the proposed scheme will also be analyzed.

4.4 Performance Metrics

Let N be the number of peers in the system and let a proportion α of them be free-riders. Assume that free-riders are uniformly distributed over the system. Also, assume that both cooperative users and free-riders have the same download demands. In particular, they have the same query request rate, denoted by R_q queries/sec, and the same preference over files, that is, each query is for file i with some probability, $Q_f(i)$, irrespectively of the query issuer. Finally, assume that free-riders respond to a query only if the amount of tokens they currently have is less than the amount required to download a file while cooperative users always respond to query requests.

User Download Rate (R_{down}) Let $P_{ans}(i)$ be the probability that a query request for file i is successfully answered. Now recall that in the system with the tokens a user can initiate a download only if the amount of tokens he/she has is larger than the amount required to download the file. Let P_{tkn}^{FR} and P_{tkn}^{NF} denote respectively the percentage of time that a free-rider and a non-freerider have enough tokens to initiate a download. Then, we can express the expected download rate for a free-rider and a non-freerider, respectively, as follows:

$$E[R_{down}|\text{FR}] = \sum_i R_q Q_f(i) P_{ans}(i) P_{tkn}^{FR}, \tag{9}$$

$$E[R_{down}|\text{NF}] = \sum_i R_q Q_f(i) P_{ans}(i) P_{tkn}^{NF}, \tag{10}$$

where the summation is taken over all files i. Clearly, the expected rate over all users in the system is:

$$E[R_{down}] = E[R_{down}|\text{FR}]\alpha + E[R_{down}|\text{NF}](1-\alpha). \tag{11}$$

Notice that while both classes of users have the same download demands (i.e. the same R_q and $Q_f(i)$), their expected download rates may differ, since in general $P_{tkn}^{FR} \neq P_{tkn}^{NF}$. However, we can find a relation between P_{tkn}^{FR} and P_{tkn}^{NF} by making the following two observations. First, observe that the rate of by which a user earns tokens is proportional to the percentage of time that the user is providing uploads. This in turn is proportional to the percentage of time that the user responds to query requests. Write $E[R_t|\text{FR}]$ for the average token earning rate of a free-rider and $E[R_t|\text{NF}]$ for the average token earning rate of a non-freerider. A free-rider responds to a query request only when he/she doesn't have enough tokens, i.e. with probability $1 - P_{tkn}^{FR}$. In contrast, a non-freerider always responds to a query request, i.e. with probability 1. Therefore we can easily see that $E[R_t|\text{NF}] = \frac{E[R_t|\text{FR}]}{1-P_{tkn}^{FR}}$. Now, observe that the token earning rate should be proportional to the token spending rate, which in turn is proportional to the download rate. Therefore $E[R_{down}|\text{NF}] = \frac{E[R_{down}|\text{FR}]}{1-P_{tkn}^{FR}}$. This fact, with Equations (9), (10), and the fact that $P_{tkn}^{NF} \leq 1$, yield:

$$P_{tkn}^{NF} = \min\left(1, \frac{P_{tkn}^{FR}}{1-P_{tkn}^{FR}}\right). \tag{12}$$

Now, lets find a relation for $P_{ans}(i)$. First, assume that due to overlay congestion, which may occur during the forwarding process on peers [18], each message (either a query request or a query response) has a probability p of being dropped at some peer. [2] Then, if L is the average number of overlay hops until a query is answered, $P_{drop} = 1-(1-p)^L$ is just the probability that the query response is lost. Next, observe that if $K \leq N$ is the average number of peers that a query request can reach, then the request can be answered by an average of $K((1-P_{tkn}^{FR})\alpha + 1(1-\alpha))$ peers. Finally, let $P_f(i)$ be the probability that a peer has file i (file popularity). We can then write:

$$P_{ans}(i) = 1-(1-P_f(i)(1-P_{drop}))^{K((1-P_{tkn}^{FR})\alpha+1-\alpha)}. \tag{13}$$

The probability functions $Q_f(i), P_f(i)$, as well as the expected query rate R_q, are determined from measurement studies of P2P systems, e.g. see [19, 20]. Finally, quantities like K and L can be easily computed for many kinds of overlay graphs and search algorithms, e.g. see [21] and references therein.

User Upload Rate (R_{up}) In any system, the total number of downloads equals the total number of uploads, and thus the expected download and upload rates over *all* nodes are also equal. This however does not necessarily mean that all peers provide uploads. For example, in a system that does not implement the proposed

[2] This assumption is introduced to make the model more general. A well designed system usually has $p \approx 0$, which is accomplished by setting the buffer size of the TCP socket sufficiently large.

scheme $E[R_{down}] = E[R_{up}]$ but we know that only non-freeriders provide uploads, i.e. $E[R_{up}|\text{FR}] = 0$, and hence $E[R_{up}|\text{NF}] = \frac{E[R_{down}]}{(1-\alpha)}$. On the other hand, in the system with the tokens each free-rider answers to a query request with probability $1 - P_{tkn}^{FR}$. As a result, this system seems like having $N((1-\alpha) + \alpha(1 - P_{tkn}^{FR}))$ non-freeriders. It is easy to see that the expected upload rate of each non-freerider is now given by:

$$E[R_{up}|\text{NF}] = \frac{E[R_{down}]}{(1-\alpha) + \alpha(1 - P_{tkn}^{FR})}. \tag{14}$$

And, since $E[R_{up}] = E[R_{down}]$, the expected upload rate of each free-rider equals:

$$E[R_{up}|\text{FR}] = \frac{(1 - P_{tkn}^{FR})E[R_{down}]}{(1-\alpha) + \alpha(1 - P_{tkn}^{FR})}. \tag{15}$$

Query Response Time Query response time consists of two major parts: network delay and retrial time. Network delay comes from the congestion of the network and the retrial time is the amount of time spent due to the timeout mechanism (retransmissions) of unsuccessful queries. Since the theoretical network delay can not be easily calculated and we argue that the network delay does not dominate the query response time[3], we hence regard the query retrial time as an estimated query response time. In average, the number of retrials needed to received a response for file i is $\frac{1}{P_{ans}(i)} - 1$ and hence the retrial time of file i is $\text{TO}(\frac{1}{P_{ans}(i)} - 1)$ where TO is the timeout of an unsuccessful query. Therefore, the average query response time over all files i is:

$$\sum_i \text{TO}\left(\frac{1}{P_{ans}(i)} - 1\right) Q_f(i). \tag{16}$$

File Download Time We define file download time as the amount of time spent in downloading a file from the system. Because of the complexity of the underlying network, it is very difficult to estimate network congestion and network delay of a P2P system. However, our intention is to study the performance enhancement of the system with tokens rather than to evaluate the performance itself. Given system's performance of an original system, we are interested in estimating the performance improvement the token system in terms of file download time.

To this end, we view the whole P2P system as a *black service box.* The arrival rate to this service box is $NR_qQ_f(i)P_{ans}(i)$. Because users only initiate downloads when their queries are answered, the inter-arrival time to this service box is a Geometric distribution. Since Geometric distribution is the counterpart of Exponential distribution in the discrete time, the arrival process to this service box can be viewed as a Poisson process. Therefore, the service box can be viewed as a M/G/1-PS (processor sharing) system. In our simulation the service time of the service box is fixed because file size of all files are the same. As a result, we use M/D/1-PS to predict the performance of our system. In a M/D/1-PS system, the total delay (includes service time and queueing delay) of the system is proportional to $\frac{\rho}{1-\rho}$ where ρ is the load of the system. In a P2P system, the service capacity of the system is the number of users who are willing to provide upload. Namely, the service capacity is $N(\alpha P_{tkn}^{FR} + 1 - \alpha)C_{up}$. Since the download demands of the system is $NE[R_{down}]S$,

[3] For now we assume this and we are infestigating ways of relaxing this assumption. In fact the simulation results justify this assumption.

the load of the system is $\frac{E[R_{down}]S}{C_{up}(\alpha P_{tkn}^{FR}+1-\alpha)}$. Given file download time of the original system, D_{ori}, we can estimate the file download delay of the system with tokens, D_{tkn}, by:

$$\frac{D_{tkn}}{D_{ori}} = \frac{\frac{\rho_{tkn}}{1-\rho_{tkn}}}{\frac{\rho_{ori}}{1-\rho_{ori}}} = \frac{\rho_{tkn}(1-\rho_{ori})}{\rho_{ori}(1-\rho_{tkn})}. \tag{17}$$

File Download Delay The sum of query response time and file download time is the time spent from a user issued a query request to the moment he/she finished the download task. We define this time difference as the file download delay. File download delay is a metric that end users will concern and so we use it as a metric to evaluate the performance of P2P systems.

4.5 Choosing the right values for K_{on}, K_{up}, K_{down} and predicting performance

We use P_{tkn}^{FR} as the design parameter of our system since it represents the level of contribution of free-riders and comprises an indication of how fast users accumulate tokens. We are given the probability functions $Q_f(i)$, $P_f(i)$, the values of R_q and p, and the structure of the overlay graph. We want to find a set of values for K_{on}, K_{up} and K_{down} that will satisfy a target P_{tkn}^{FR}.

First, observe from Equation (8) that it is the *relative* values of K_{on}, K_{up}, and K_{down} that are important for the proper operation of the system. However, also recall that K_{on} should be sufficiently smaller than the token spending rate of free-riders. This is to prevent them from accumulating enough tokens by just staying online without offering any uploads. Thus, we should have $K_{on} \ll K_{down}E[R_{down}|\text{FR}]S$.

With the above observations in mind we proceed as follows in order to satisfy the target P_{tkn}^{FR}:

1. Fix K_{down} to some arbitrary value.
2. Use Equation (12) to compute P_{tkn}^{NF}. (To guarantee that cooperative users will not be penalized, i.e. $P_{tkn}^{NF} \approx 1$.)
3. Use Equations (9) and (13) to compute $E[R_{down}|\text{FR}]$ and use Equations (10), (11), and (15) to compute $E[R_{up}|\text{FR}]$.
4. (v) Assign a value which is one order of magnitude smaller than $K_{down}E[R_{down}|\text{FR}]S$ to K_{on}. (The specific value turns out not to affect the performance sizeably.)
5. Use Equation (8) to find the corresponding value for K_{up}.
6. Use Equation (16) and (17) to estimated the performance of the system.

Conversely, if we are given the values of K_{on}, K_{up}, and K_{down}, which are being used in a system, we can use our equations to predict quantities like $E[R_{down}|\text{FR}]$, $E[R_{down}|\text{NF}]$, $E[R_{up}|\text{FR}]$, P_{tkn}^{NF}, and so on. [4]

In the next Section we verify our analysis via experiments on top of TCP networks, and show the impact of the proposed scheme on system's performance.

[4] Note that we can also use Equations (9)...(15) to compute upload/download rates in a system that does not implement the scheme, by setting $P_{tkn}^{FR} = 1$.

5 Experiments

5.1 Simulation setup

For our experiments we use GnutellaSim [22], a packet-level peer-to-peer simulator build on top of ns-2 [23], which runs as a Gnutella system. We implement the file downloading operation using the HTTP utilities of ns-2.

We use a 100-node transit-stub network topology as the backbone, generated with GT-ITM [24]. We attach a leaf node to each stub node. Each leaf node represents a peer. The propagation delays assigned to the links of the topology are proportional to their length and are in the order of ms. We assign capacities to the network such that the congestion levels are moderate. The capacity assigned to a peer's access link is 1.5Mbps.

In order to test the algorithm on a general gnutella-like unstructured P2P network we use Gnutella v0.4, which uses pure flooding as the search algorithm and does not distinguish between peers. The TTL for a query request message is set to 7 (the default value used in Gnutella).

All peers join the system initially and never go offline. For simulation purposes we implement the following user behavior: each user initiates query requests at the constant rate of 1 query every 20sec. Once a timeout for a query request occurs, the corresponding query is retransmitted. The maximum number of retransmissions is set to 5, and the timeout is 60sec.

There are 1000 distinct files in the system. The number of replicas of a certain file is described by a Zipf distribution with a scaling parameter equal to 1. Query requests are issued according to the same distribution, and replicas of a certain file are uniformly distributed among all peers. These settings are in accordance with measurement studies from real P2P networks [19, 20]. We distinguish two systems: (i) the original system which does not implement the proposed algorithm, and (ii) the system with the tokens. In both systems, 85% of peers are free-riders in accordance to the percentage reported in [2]. Finally, the file size is set to 1MB.

5.2 Simulation Results

Download and Upload Rates For various values of the design parameter P_{tkn}^{FR} we compute the corresponding values of K_{on}, K_{up} and K_{down} according to the procedure described in the previous Section. We then assign these values to all users of the system and compare the theoretical download and upload rates with the experimental results. Figures 5(i) and 5(ii) show respectively the expected download and upload rate over all non-freeriders, over all free-riders, and over all users of the system, as a function of P_{tkn}^{FR}. The horizontal line in Figure 5(i) represents the expected download rate of a user in the original system. (Clearly, in the original system $E[R_{down}] = E[R_{down}|FR] = E[R_{down}|NF]$.) The horizontal line in Figure 5(ii) represents the expected upload rate of a non-freerider in the original system. (Recall that $E[R_{up}|FR] = 0$ in this system.)

It is clear from the plots that analytical and simulation results match. Further, we can make several interesting observations. First, notice that as P_{tkn}^{FR} increases, the download rate for both classes of users first increases and then starts decreasing until it reaches the value of the original system. Second, observe that while the upload rate of free-riders behaves in a similar manner, the upload rate of non-freeriders continuously increases until it reaches its original value. Based on these observations

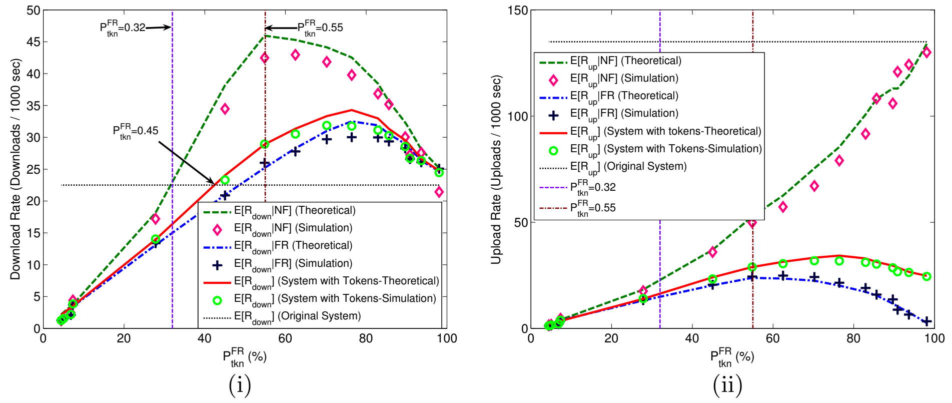

Fig. 5. (i) User's expected download rate, and (ii) user's expected upload rate.

we divide the plots into three regions. The first region corresponds to $P_{tkn}^{FR} < 0.32$. In this region, both classes of peers are constrained to a lower download rate compared to the original system, since the probability of having tokens to initiate a new download after a successful query, is pretty low. Notice that for $P_{tkn}^{FR} = 0.32$, cooperative users can sustain the same download rate they had in the original system. The second region corresponds to $0.32 \leq P_{tkn}^{FR} \leq 0.55$. In this region, users accumulate tokens at a higher rate than before. Since there are more responses than in the original system, users can use the extra tokens to initiate more downloads. Notice that cooperative users earn tokens faster than free-riders since they always respond to query requests. At $P_{tkn}^{FR} = 0.55$, non-freeriders achieve their maximum download rate, which is approximately twice the one they had in the original system. Finally, the third region corresponds to $0.55 < P_{tkn}^{FR} \leq 1$. In this region free-riders accumulate tokens faster than before and reduce their query response rate since they do not need to provide as many uploads as before. This causes cooperative users to handle more uploads. Futher, since the query response rate regulates the download rate, the latter also decreases. At $P_{tkn}^{FR} = 1$, the two systems have approximately the same performance, as expected.

Impact On Delays Figures 6(i) and 6(ii) show respectively the average query response time (that includes retransmissions) and the average download delay as a function of P_{tkn}^{FR}. First, we can observe from the figure that the theoretical results can capture the simulation results.[5] The plots can also be divided in the same three regions as before. For $P_{tkn}^{FR} < 0.32$, the low user download rate imposes a low load into the network. This yields the low delays. For $0.32 \leq P_{tkn}^{FR} \leq 0.55$, as the user download rate increases, the load in the network and hence the delays also increase. Note that both delays are still significantly smaller than the original system, even at higher download rates. This is because a significant portion of the load is now handled by the free-riders. For $0.55 < P_{tkn}^{FR} \leq 1$ the delays increase at a higher rate, even though the download rate decreases. This is because free-riders provide fewer

[5] We can observer that the theoretical result is slightly smaller than the simulation results. This comes from the facts that we do not take the network delay into consideration in estimating query response time.

and fewer uploads. As P_{tkn}^{FR} approaches 1, the performance of the two systems is approximately the same.

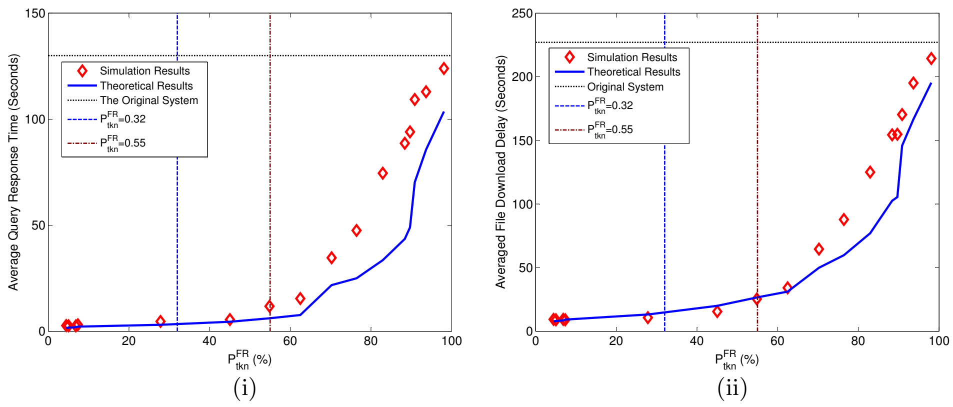

Fig. 6. (i) Average query response time, and (ii) average file download delay.

To fairly compare the delays between the two systems, we should consider the case where the load is the same, i.e. where $E[R_{down}] = 22$ downloads/1000sec. This value corresponds to $P_{tkn}^{FR} = 0.45$, and as we can see from the plots this corresponds to approximately one order of magnitude lower query and file download delays. This is a gigantic amount of improvement on the system's performance.

As a final note, the best operating region is the second, where $0.32 \leq P_{tkn}^{FR} \leq 0.55$. In this region, we can either choose to operate the system at $P_{tkn}^{FR} = 0.32$, where cooperative users can sustain the same download demands as in the original system, or sacrifice a bit from the performance improvement with respect to the reduced delays to support higher user demands.

5.3 In Case of Hacking

Theory and simulations show syetem's performance can be greatly improved by adopting the proposed scheme. However, any incentive scheme may suffer malicious hacking. Therefore, we are insterested in studying how much performance degeneration will we suffer if we have a small percent of malicious users in the system. Assume we have 5% of free-riders who are capable to circumvent the proposed scheme. They can download files as long as they receive responses from the system. We comapre the system with malicious users and the system without malicious users in Figure 7. First, we can observe that the performance of these two systems are very close. The amount of performance degeneration is almost negligible. Note that the user download rate in the system with malicious users in general[6] is smaller than the rate in the system without malicious users and the average file download delay in the system with malicious users is larger than the delay in the system without mailicious users. These results are natural consequences of imposing malicious users in the system. Because mailicious users do not contribute to the system, the service capacity of the system decreases and thus the user download rate decreases. For the same

[6] In the plot, there are particular counter examples due to the simulation tolerance.

reason, fewer users in the system provide uploads results in the higher file download delay.

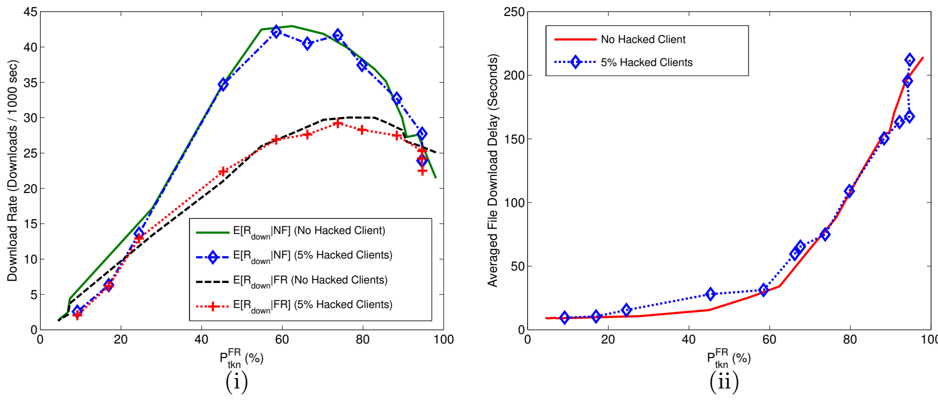

Fig. 7. (i) user download rate, and (ii) average file download delay.

6 Implementation

As mentioned before, this scheme can be implemented either locally or non-locally. Implementing this scheme locally is quite simple. The local P2P client takes care of bookkeeping by increasing the user's tokens for each acknowledged byte he/she uploads and for being online, and by decreasing the tokens for each byte the user downloads. However as we have already mentioned, this approach is quite vulnerable to hacked clients.

There are several directions for making the hacking of localized solutions hard. One can utilize encryption techniques e.g. [25] that make unauthorized modifications to data (such as the scheme's parameters) hard. In addition, one can also use technologies like DRM (Digital Rights Management) in order to protect the entire client's code from being altered e.g. [26], and re-distribute new clients frequently in order to minimize the number of hacked clients that can be connected to the network. Further, one could also employ techniques such as tamper-proofing and self-checking in order to verify the client's code integrity during the join process and/or on every download request e.g. [27–30]. Of course, the only way to guarantee that all P2P clients are original is to have a trusted platform where both the hardware and the operating system can be trusted. This is clearly not an option in practice. However, interestingly enough, both theory and simulations dictate that our scheme is quite resilient to a small number of free-riders. In particular, we have showed that the system performance is virtually unchanged when the malicious free-riders comprise 5% of the populaton of the system. Hence, all one needs to do is to make it hard for users to use hacked clients.

The scheme can be also implemented in a secure non-localized centralized manner, where peers exchange messages with a centralized authority that updates and maintains their amount of tokens. Peers communicate with the centralized authority once they finish uploading and downloading, periodically while being online, and to get permission to initiate a new download. This is similar to the main idea that many centralized "cash-based" systems, *e.g* [3, 4], follow. Finally, the scheme can be also implemented in a secure non-localized decentralized manner, for example by utilizing and extending the framework suggested in [6].

7 Conclusion

In this paper we studied a simple algorithm that provides strong incentives for cooperation in file sharing P2P networks. We provided a set of equations that relates the system's dynamics and which can be used for parameter tuning and performance prediction. We demonstrated the effectiveness of the algorithm via experiments with TCP networks. We also showed via simulation the algorithm is resilient to a small number of mailicious free-riders. Future work consists of performing larger scale experiments and extending our analytical methodology to prevalent P2P syetems, e.g. BitTorrent and eMule.

References

1. E. Adar and B. Huberman, "Free riding on gnutella," `http://www.firstmonday.dk/issues/issue5_10/adar`, October 2000.
2. D. Hughes, G. Coulson, and J. Walkerdine, "Free riding on gnutella revisited: the Bell Tolls?," *IEEE Distributed Systems Online Journal*, vol. 6, no. 6, June 2005.
3. "Mojonnation," `http://www.mojonation.net/Mojonation.html`.
4. J. Ioannidis, S. Ioannidis, A. Keromytis, and V. Prevelakis, "Fileteller. paying and getting paid for file storage," in *Proc. of 6th International Conference on Financial Cryptography*, March 2002.
5. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "EigenRep: Reputation management in P2P networks," in *Proc. of 12th International World Wide Web Conference (WWW 2003)*, May 2003.
6. V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "KARMA: A secure economic framework for P2P resource sharing," in *1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.
7. C. Buragohain, D. Agrawal, and S. Suri, "A game-theoretic framework for incentives in P2P systems," in *Proc. of International Conference on Peer-to-Peer Computing*, Sep 2003.
8. L. Ramaswanmy and L. Liu, "Free-riding: A new challenge to peer-to-peer file sharing systems," in *Proc. of the 36th Hawaii international conference on system sciences*, 2003.
9. M. Feldman, C. Papadimitriou, I. Stoica, and J. Chuang., "Free-riding and whitewashing in Peer-toPeer systems," in *SIGCOMM Workshop*, 2004.
10. "KaZaA participation level," `http://www.kazaa.com/us/help/glossary/participation_ratio.htm`.
11. "The emule project," `http://www.emule-project.net/`.
12. "KaZaA media desktop," `http://www.kazaa.com/`.
13. "Bittorrent," `http://www.bittorrent.com/protocol.html`.
14. H. Bretzke and J. Vassileva, "Motivating cooperation in peer to peer networks," in *Proc. of User Modeling UM03*, June 2003.
15. "Hack KaZaA participation level," `http://www.davesplanet.net/kazaa/`.
16. K. Anagnostakis and M. Greenwald, "Exchanged-based incentive mechanisms for peer-to-peer file sharing," in *Proc. of 24th International Conference on Distributed Computing Systems*, 2004.

17. R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding availability," in *Proc. of 2nd IPTPS*, 2003.
18. Qi He and Mostafa Amar, "Congestion control and massage loss in gnutella networks," in *Proc. of Multimedia Computing and Networking*, 2004.
19. S. Saroiu, K. P. Gummadi, R. J. Dunn, S.D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," in *Proc. of the Fifth Symposium on Operating System Design and Implementation (OSDI)*, December 2002.
20. J. Chu, K. Labonte, and B. N. Levine, "Availability and locality measurements of peer-to-peer file sharing systems," in *Proc. of SPIEITCom: Scalability and Traffic Control in IP Networks*, July 2002.
21. C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes for unstructured peer-to-peer networks," in *Proc. of INFOCOM*, 2005.
22. "Packet-level Peer-to-Peer Simulation Framework and GnutellaSim," `http://www.cc.gatech.edu/computing/compass/gnutella/`.
23. "Network simulator," `http://www.isi.edu/nsnam/ns`.
24. K. Calvert, M. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, 1997.
25. "Data encryption standard," `http://www.itl.nist.gov/fipspubs/fip46-2.htm`.
26. T. Sander, *Security and Privacy in Digital Rights Management*, Springer; 1st Edition, 2001.
27. D. Aucsmith, "Tamper resistant software: An implementation," in *Proc. 1st International Information Hiding Workshop*, May 1996.
28. H. Chang and M. Atallah, "Protecting software code by guards," in *Proc. of 1st ACM Workshop on Digital Rights Management*, May 2002.
29. Y. Chen, R. Venkatesan, M. Cary, R. Pang, S. Sinha, and M. Jakubowski, "Oblivious hashing: A stealthy software integrity verification primitive," in *Proc. of 5th International Information Hiding Workshop*, October 2002.
30. C.S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation - tools for software protection," *IEEE Transactions on Software Engineering*, vol. 28, no. 6, June 2002.