Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 04-004

Fighting Freeloaders in Decentralized P2P File Sharing Systems

Ivan Osipkov, Yongdae Kim, and Anand Tripathi

January 14, 2004

Fighting Freeloaders in Decentralized P2P File Sharing Systems

Ivan Osipkov, Yongdae Kim, Anand Tripathi
University of Minnesota - Twin Cities
{osipkov,kyd,tripathi}@cs.umn.edu

Abstract Currently used P2P file-sharing systems such as Gnutella suffer from a freeloading problem due to absence of enforcible decentralized mechanism to motivate peers to provide services to others. Freeloading leads to a de facto centralized system that is susceptible to DoS attacks and provides poor service to its participants. In this paper, we propose a decentralized mechanism that avoids on-line polling and forces peers to contribute their bandwidth to the system. We discuss additional positive properties of the model and, finally, discuss how to deal with collusion and Sybil attacks.

1 Introduction

The most famous and currently active decentralized P2P file-sharing systems are Gnutella and Kazaa. In these systems, once a peer has located a source of download the uploading peer is contacted directly and asked for the file. The contacted peer may either accept or decline to provide the service, but it is unable to decide whether the contacting peer is also willing to provide service for others, *i.e.* whether it is a free-rider. As a consequence, there is no egotistic motivation not to be a free-rider. It turns out that Gnutella and Kazaa indeed suffer from this problem. In Gnutella, 70% of peers do not provide any service to the rest of community, and 50% of all queries are served by 1% peers [1]. As a result, Gnutella becomes a de facto centralized system where the hard-working peers form overloaded centers becoming bottlenecks. The de facto centers in Gnutella become targets for RIAA court battles [2] and DoS attacks [8, 3, 9].

Besides the DoS-related security concerns, the above systems are poor from economic point of view. In [7], the authors found that 30% of all requests in Kazaa go to 1% of files which generate 80% of the traffic. With freeloading being a problem, these files are not being spread by free-riders who nevertheless are the primary downloaders, the number of peers that offer them is inadequate, and these peers experience congestion which forces them to put others into long queues in this way degrading service.

A typical approach to deal with freeloading problem is through polling. One can use witnesses that attest to each transaction a peer goes through [13, 11]. These witnesses are middle-men for each transaction accumulating credit/debit or reputation level for its assigned peer. Increasing the witness set to all peers, one can also simply poll other peers on a random basis [5]. However, it may not be always easy to recruit witnesses in a secure manner or obtain a large enough data bank to extrapolate reputation, especially when 70% of peers may be freeloaders which hardly would agree to participate in the protocol. In addition to that, polling adds overhead, is a vehicle for DoS attacks and, due to dynamic nature of P2P groups, may not give accurate picture about a peer.

Another way to deal with freeloading is through micropayments [12]. In such schemes one needs to have a broker trusted by all involved peers that is willing to generate digitally signed cash coins and arbitrate disputes, thus forcing a concept of a central authority into the model. Such approaches are suited more for selling/buying of files than simple sharing.

Yet one more way would be to force users store others' files [4, 10]. However, in current file-sharing systems storage is not as valuable as bandwidth, and the fact that a peer stores a file does not necessarily mean that it will provide it to others for download. In addition, forcing peers to store specific files would probably be unacceptable to the users of Gnutella-like systems. If instead we decide to enforce only a lower bound on the number of files published, smart peers will overcome this restriction by publishing files that no one is interested in. We need to concentrate on bandwidth as the currency of the system.

1.1 Focus and Contributions

In this paper, we are concerned with alleviating the freeloading problem in a real-world P2P system where overwhelming majority of users are not malicious. We will assume that peers perform transactions according

to the agreed terms and that there exists a routing layer (such as Gnutella) which provides query mechanisms. We present a decentralized model on top of Gnutella layer with the following characteristics:

• First, peers have to perform the service proportional to their use of the system in order to participate in it. In addition to curbing freeloading, this implies that popular files will be spread faster and offered more, fostering better load-balance of the system.

• Second, all (recent) participation history of a peer is presented (and kept) by the peer itself. The history consists of digitally signed receipts and thus unforgeable. We do not use third-party brokers or polling to ascertain participation level of a peer.

• Third, it is in the interests of peers to keep all parts of their participation history. Therefore, the receipts that attest to their use of the system (and thus decrease the peers' credit) are not dumped.

• Fourth, the system provides a limit on the amount of credit a peer may accumulate and introduces *progressive taxation*. The more credit the peer has, the more credit it will lose during downloads and the less credit it will gain during uploads. This allows new users to quickly start full participation in the system.

• Fifth, the system provides distributed digitally-signed data that can be used to detect the most abusive cases of collusions. These proofs are undeniable and peers can decide what to do with such criminals

1.2 Organization

Section 2 contains discussion of the basic protocol including the assumptions and properties inherent to it. Section 3 discusses additional security concerns such as collusion and Sybil attacks, and suggests how our protocol can be made resilient to them. We conclude this paper with future work in Section 4.

2 Fighting Freeloaders

In our model, each peer locally maintains a history of its activities in the system. This history includes when the uploads and downloads have occurred in the past and what the role of the peer was. When peer A approaches B with a download request, A will submit to B its history and B will decide on whether to allow a download and when. Depending on A's history, it may be put in B's priority queue. Also, B will provide its history to A. The peers will negotiate between each other on the details of the download. After the download is complete, A and B will exchange receipts that testify that there was a download in a specified time-frame. Peer A will receive a receipt from B stating that it was the downloader, and B will receive a receipt from A that it was uploading. These receipts will be part of history and each peer will hold on to its receipts only.

So far, there is nothing that prevents A from dumping the above receipt which, in fact, has a negative effect on its history. To deal with that, we will include in the receipts the time-frames when the download/upload occurs and we will charge peers for the time-frames which are *unaccounted* for in the receipts. *Thus if A dumps the above receipt, peers will regard this time-frame as unaccounted for and will charge A for that time-frame at least as much as for download.*

2.1 System Assumptions

Our system assumptions are:

• the peers' clocks are pairwise loosely synchronized

• each peer has a public/private key pair with a certificate, *i.e.* existence of PKI is assumed.

2.2 Basic Protocol

For simplicity, *we temporarily assume that every peer has the same fixed bandwidth b Kb/sec, and, once committed, this bandwidth is reserved until the end of the transaction.* These assumptions will be removed later. The digital signatures that peers exchange at the end of the protocol are defined as follows:

Definition *S-receipt belonging to A and signed by C is $(sign\{s, T_1, T_2, A, C\}_C, cert_C)$ where (T_1, T_2) is a time-frame contained within the actual time-frame during which the transaction was carried out, s denotes that this is an S-receipt, A and C indicate that A was the sender and C was the receiver, and $cert_C$ is a public-key certificate of C.Similarly, R-receipt is of the form $(sign\{r, T_1, T_2 A, C\}_C, cert_C)$ where r denotes that this is an R-receipt, A is the receiver and C the sender.*

Each interaction between peers A and B when A wants to download from B consists of 4 stages:

Stage 1(*Exchange of Previously Collected Receipts*) Peer A sends its relatively recent *R-receipts and S-receipts* with its download request to B. Peer B examines the receipts and puts A into a priority queue. Peers with better history have higher priority. Once A gets its turn, peer B replies with its own set of *receipts.*

Stage 2 (*Price Negotiation*) Peers A and B negotiate on the parameters of the receipts that each one of them expects to receive from the other after the download is complete.

Stage 3 (*Download*) Once mutual agreement has been

reached the download commences.

Stage 4 (*Receipt Exchange*) At the end of successful download, both peers exchange *receipt*s on which they agreed before.

2.3 Anatomy of Our Protocol

Exchange of Previously Collected Receipts Note that A and B have to exchange only recent R-receipts and S-receipts, due to credit aging as explained below. Priority queue is maintained by each peer to control bandwidth. Peer with higher credit in the priority queue will be served first.

Price Negotiation Peer B examines the receipts of A as follows. Let R_i^{start} (R_i^{end}) be the set of starting times (end times) specified in the *R-receipts* when A was the receiver. Let S_i^{start} (S_i^{end}) be the set of starting times (end times) specified in the *S-receipts* when A was the sender. Let (H_i^{start}, H_i^{end}) be the time-intervals unaccounted for in the *receipts* and T be the current time.

Credit calculation: Credit of A is defined as follows:

$$\begin{aligned} C_A = \quad & g(\sum f_1(S_i^{start}, S_i^{end}, T), \\ & \sum f_2(R_i^{start}, R_i^{end}, T), \\ & \sum f_3(H_i^{start}, H_i^{end}, T)) \end{aligned}$$

where $f_i(t_1, t_2, T)$ and $g(x, y, z)$ are functions to be specified later.

Choice of parameters for the R-receipt: Let (T_1, T_2) be the actual time-interval of the download. Peer B now chooses $T_1 \le T_1^r \le T_2^r \le T_2$ such that

$$C_A - C_A^{new}$$

reflects the loss of credit that A should incur, where C_A^{new} is the credit of A at the end of the download including the receipt that will be given by B. Peer B will propose to send $(sign\{r, T_1^r, T_2^r, A, B\}_B, cert_B)$ to A at the end of the download.

Choice of parameters for the S-receipt: Next, B chooses $T_1 \le T_1^s \le T_2^s \le T_2$ in such a way that

$$C_B^{new} - C_B$$

reflects the increase in the credit that B is seeking, where C_B^{new} is the credit of B at the end of the download with the receipt that it will receive from A. Peer B will ask A to send $(sign\{s, T_1^s, T_2^s, B, A\}_A, cert_A)$ at the end.

Download Provided A and B have finished negotiation on what receipts they will send to each other at the end, the file transfer starts.

Receipt Exchange Provided that each party is satisfied with the download, A sends $(sign\{s, T_1^s, T_2^s, B, A\}_A, cert_A)$ to B, and B sends $(sign\{r, T_1^r, T_2^r, A, B\}_B, cert_B)$ to A. Care must be taken when exchanging the receipts so that each party receives what it expects.

2.4 Defining f and g

Functions f_i takes three parameters: initial time t_1, final time t_2, and current time T where $t_1, t_2 \le T$. These functions calculate the worth of the time-frame (t_1, t_2) at time T.

Of each function $f = f_i$ we require the following:

- As the time-frame ages its worth should decrease, *i.e.* for fixed t_1,t_2 the function f is decreasing in T.
- The sum of function f values for non-overlapping time-frames should have a finite upper bound.
- The function should always be non-negative.

It follows that due to aging, old receipts become less important than recent ones. By choosing f appropriately one can vary exactly how fast time-frames will age.

Function $g(x, y, z)$ takes three parameters: x is the total worth of all *S-receipts*, y is the total worth of all *R-receipts* and z is the total worth of all unaccounted time-frames. The basic requirements are:

- The function g should be increasing in x, decreasing in y and decreasing in z
- The function g must decrease in z at least as rapidly as in y.

The first requirement states that uploads have a positive effect on total credit and downloads (and unaccounted times) have a negative one. The 2nd requirement states that peers are charged for unaccounted times at least as much as for downloads.

As an example, if one chooses to use exponential aging, one can take $f(t_1, t_2, T) = k \cdot e^{-T} \cdot (e^{t_2} - e^{t_1})$. As for g, by taking $g(x, y, z) = x - y - 2 \cdot z$ the above requirements will be satisfied.

2.5 Removing Temporary Assumptions

Let b Kb/sec be the system *unit* bandwidth, *i.e.* every peer's bandwidth is a multiple of b. If peer A has bandwidth $k \cdot b$, this peer simulates k *virtual* peers with pseudonyms related to A's original pseudonym. If A decides to give more bandwidth than just one channel to another peer, then it will receive receipts for every channel that it gives. Since one can detect if two pseudonyms are related, the peer will not be able to sign its own receipts and thus launch a Sybil attack. Since these channels are independent, a peer now can download on one

channel and upload on another.

Proposition 1 *Serving several peers at the same time or serving them one at a time will generate the same credit for the peer, provided the total bandwidth used is the same in both cases.*

2.6 Properties/Consequences of the Model

Freeloading Our scheme uses a completely offline history management mechanism. According to the above model, a peer's credit is decreased when a download occurs, and it is decreased even more for the time-intervals for which it can not account using the receipts. Thus it is in the interests of a peer not to dump its *R-receipts*. Since receipts testify to participation level of a peer, freeloaders will be always put at the end of priority queue making sure that non-freeloaders get serviced. In this way freeloading problem is alleviated.

Long Term Equilibrium To obtain quality of service proportional to peer's bandwidth, the peer must have a viable amount of credit for each channel, which means that it also has to provide service to others proportional to its bandwidth. As a consequence, popular files spread more since they are the main "credit-makers", which decreases congestion around peers who offer them.

Negative Credit and Credit Bounds Let C be the total credit of a peer. From the requirements for f_i and g, it follows that $k_1 \leq C \leq k_2$ where k_i are constants that depend on the choice of the functions. Some of these constants may be negative, but let us note that the sign of credit does not go into decision whether to allow a download. Credit is used to order peers in the priority queue only.

New User Joins According to out model, as credit of peer increases it has to do progressively more work for the same amount of credit increase. This comes from the fact that the more credit a peer has, the more this credit will age at the end of the upload in absolute terms (the situation with downloads is symmetric). This implies that a new user will gain from uploads more than a "rich" peer making it easier to obtain a viable amount of credit and fully participate in the system quickly.

History size Since *receipts* age, a peer needs to submit only recent ones thus bounding the size of information that needs to be exchanged.

3 Additional Discussion

3.1 Collusion Attacks

In a decentralized P2P system, one user can always vouch for another if it chooses to do so. To deal with that, typically, a range of users would have to be asked to ascertain peer's participation level. Since we specifically aim to avoid polling, our system is susceptible to some extent to collusion attacks. Namely, one peer can vouch for another one at the expense of its own credit. The peer that vouched loses credit and thus ends up losing time by longer waits. More precisely, we have the following two propositions.

Assume that $g(x, 0, 0) > 0$ below (*i.e.* credit can become positive). How large g can become can be adjusted by changing g.

Proposition 2 *Assuming that function g is additive in its vector-argument, $f_1 = f_2$, $f_2 \leq f_3$, $g(0, 0, x) \leq g(0, x, 0)$ and $g(x, x, 0) \leq 0, \forall x$, normal transactions between peers can not raise their total credit above 0.*

Proposition 3 *Under the assumptions of the previous proposition, if transactions between peers raise their total credit above 0, then the set of their receipts contains conflicts that prove that there was a collusion. One can determine identity and prove malicious intent of at least one of the violators.*

Proofs are skipped due to space limitations. By above two propositions, if colluders do not want to be provably exposed, the total amount of credit that they may simulate is limited. If this credit is divided fairly, each colluder gets limited advantage. If large portion of the credit is transfered to only one colluder, the other colluders will not be able to utilize the system effectively.

Note that the receipts that colluders generate have to be presented to others as quickly as possible (due to aging) and, therefore, may be cached by other peers in whose own interests it is to do so in order to prevent providing service to freeloaders. One can build a distributed system that uses this data to detect "provable collusion".

3.2 Sybil Attacks and PKI

In collusions, it would be unlikely that a colluder would agree to lose credit in order for another one to gain credit. But, if a peer can generate several different identities, and only one of his "identities" wins and the others lose, the peer still wins. It would be hard to generate more than one identity if trusted central CA were used in our model's PKI. However, this solution would be unnatural for a decentralized P2P system, and it is unlikely that peers would go through trouble of obtaining a public key certificate from CA such as *Verisign* just to share files. A weaker alternative would be the PGP which is easy to use even though its chain of trust may not always be reliable and one can, if desired enough,

generate several identities and launch Sybil attacks (one can improve PGP's security at the expense of making it harder to obtain a certificate [14], though). Let us notice, however, that in most cases IP addresses for each of generated identities would belong to the same subnet. If we would weigh additional receipts from the same subnet less (motivating peers to work with different subnets) launching Sybil attacks would be much harder.

3.4 Reducing Number of Stages

The above protocol consists of 4 stages. Note that in the first stage peers exchange their histories. Assume peer A is searching for a file. We then can piggyback its history (and possibly "price" it is willing to pay) onto the query message. When a peer B that has the file receives the message, it can inspect A's history and decide whether to reply. When B replies, it can piggyback its own history onto the message with a proposed "price" for the download. In this way, we can merge the 1st stage completely with Gnutella protocol,and the 2nd stage at least partially.

4 Conclusion and Future Work

In this paper, we proposed a decentralized, non-polling approach to curbing freeloading in P2P file-sharing systems. The model fosters a better long term equilibrium of the system, allows new users to join quickly and lets only insignificant collusion attacks to pass through. The problem of Sybil attacks is widely recognized [6] and can be dealt with by clustering receipts using their IP addresses. As an efficiency improvement, the protocol's stages prior to download can be piggy-backed onto Gnutella messages. Our model assumed existence of PKI such as PGP. Even though PGP can hardly be considered secure, the authors believe that it may still be adequate for applications such as Gnutella and Kazaa.

Although we believe that our model holds promise, it still needs to be analyzed and simulated thoroughly which is the subject of our current research. We assumed that peers are not involved in malicious behavior such as spreading of bad content, DoS attacks, and (perhaps non-malicious) system malfunctions that affect other peers. Such behavior should be dealt with via a reputation system, which is orthogonal and complementary to our model. The negotiations on the "price" should take into account reputations of the peers. Our model provides data to detect collusions that lead to significant increases in credit. But to take action on the discovered collusions one needs a good reputation model. Construction of a reputation model and its integration with our current proposal is another topic we are actively working on.

References

[1] E. Adar and B. A. Huberman. Free riding on gnutella. Technical report, XeroxPARC, Aug. 2000.

[2] F. Ahrens. Recording industry sues file swappers. `http://www.washingtonpost.com/wp-dyn/articles/A43155-2003Sep8.html`, Sept. 2003.

[3] M. Collins. Musiccity.com site crashes. `http://edition.cnn.com/2002/BUSINESS/03/05/musiccity/`, 2002.

[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. In *SOSP*. ACM, October 2001.

[5] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS*. ACM, 2002.

[6] J. R. Douceur. The sybil attack. In *Proc. of the IPTPS02*, March 2002.

[7] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the kazaa network. In *Workshop on Internet Applications*, Santa Clara, CA, 2003. IEEE.

[8] F. Manjoo. Gnutella bandwidth bandits. `http://www.salon.com/tech/feature/2002/08/08/gnutella\_developers/`, 2002.

[9] R. Menta. Morpheus was hacked. `http://66.155.75.37/stories/2002/morpheushack.html`, 2002.

[10] T. Ngan, D. S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In *IPTPS*, Berkeley, CA, Feb. 2003.

[11] V. Vishnumurthy, S.Chandrakumar, and E. G. Sirer. Karma : A secure economic framework for peer-to-peer resource sharing. citeseer.nj.nec.com/582637.html.

[12] B. Yang and H. Garcia-Molina. Ppay: Micropayments for peer-to-peer systems. In *CCS*. ACM, Oct. 2003.

[13] H. Zhang, D. Dutta, A. Goel, and R. Govindan. The design of a distributed rating scheme for peer-to-peer systems. In *Workshop on Economic Issues in Peer-to-Peer Systems*, Berkeley, CA, Jun 2003.

[14] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.