

Constructing Fair-Exchange P2P File Market

Min Zuo and Jianhua Li

Department of Electronic Engineering, Shanghai Jiaotong University,
Shanghai, China
{zuomin, lijh888}@sjtu.edu.cn

Abstract. P2P is a promising technology to construct the underlying supporting layer of a Grid. It is known that contribution from all the peers is vital for the sustainability of a P2P community, but peers are often selfish and unwilling to contribute. In this paper we describe how to construct a fair file-exchanging P2P community. We name this community a P2P file market. Our scheme forces peers to contribute by a micropayment-based incentive mechanism.

1 Introduction

P2P(Peer-to-Peer) networking is a promising way to construct the underlying supporting layer of a Grid[1]. Some most attracting features of P2P networks are: they are independent of servers (totally or partly); they are self-organized and robust; and, they are rich in resources (computing power, bandwidth, storage, valuable files, etc.). However, none of these features can be achieved without cooperation of the peers.

In theory, a peer in a P2P community can be both a consumer and a server. But serving others usually means sacrificing some resources of one's own interests. If peers are autonomous and self-interested, or if they are on behalf of rational individuals (as compared with *altruistic* ones), they will tend to act as pure consumers (called "free-riders" in [2]). Free-riding makes a P2P community unfair. This will eventually paralyze the whole community and lead to a "Tragedy of the Commons"[3]. To solve this problem, some kind of *incentive* mechanism is needed. In this paper, we take the file-sharing application as an example. We suggest that a P2P file-sharing community be upgraded into a "file market". In this market, every peer is allowed to share and download files, but they have to pay the provider if they want to download a file. They can earn the necessary "money" by sharing their own files in the community.

The rest of this paper goes as follows: we present some of the design considerations in section 2; then we describe the setup, file-trading, and accounting process in section 3-5; finally, we give some remarks and conclude this paper in section 6.

2 Design Considerations

There are some design considerations we'd like to mention before going into details. First is the overall architecture. There are basically three kinds of entities in the market: peers (P) who exchange files, a trusted third party (TTP) to resolve conflicts, and an accounting center (AC) acting as a central bank. We'd like to point out that, any entity trusted by the involved two parties could act as the TTP in a transaction.

Then is the problem of identity and key management. We assume each peer to be identified by a unique ID and each ID bond with a pair of public and private keys. Anonymous peers may be permitted, but they can only download and provide cost-free files (usually they are deemed as less valuable than priced ones). Here are some of the notations used in this paper: $SIG_X(M)$ denotes peer X 's signature with its private key on a message M ; $E_X(M)$, $D_X(M)$ denote the encryption of M with peer X 's public key, and the decryption with the corresponding private key; $SE_K(M)$, $SD_K(M)$ denote the encryption and decryption of M with symmetry key K ; $H()$ is a public hash function, such as MD5, to generate message digests. We suggest the adoption of *Identity-based Cryptosystem* [4] because of its easiness in key management (IDs are also the public keys). In our file market, the AC also acts as the KGC (Key Generation Center). It sets the public parameters and holds the secret master key to compute a private key for each ID.

Last is the problem of fair exchanges. One of our most important tasks is to ensure the fairness of the trading process. We borrow the ideas in [5] to design a fair exchange protocol for our system. It is an *optimistic* protocol. It guarantees fair exchange of two electronic items between two mutually distrusting parties by the mediation of a third party (TTP), but the involvement of TTP is needed only when a conflict happens. Therefore, if most of the peers are well-intentioned, the TTP is hopefully involved infrequently. With this protocol, the trading process will be able to go in a P2P fashion as much as possible.

3 System Setup

Each new peer has to register at the AC before it can participant in the file market. During each registration, AC performs the following operations: (1) choosing a unique ID for the peer; (2) computing the private key for the ID and sending this private key to the peer; (3) issuing an initial *capital certificate* to the peer; (4) adding an item for this new peer in its account database. The communication of a registration session must be performed through a safe link (eg. SSL/TSL).

A *capital certificate* (CC) is a digital certificate. It contains the subject's ID, the expiration time of this certificate (*Cash-Exp*), the upper bound of the value (*Max-Value*) and number (*Max-Num*) of valid cheques during this period, and the signature of the AC. We can compare it to a cheque-book issued by a bank. It allows the peer to issue a maximum of *Max-Num* pieces of cheques, each valued no more than *Max-Value*. And these cheques must be cashed before the time limit *Cash-Exp*.

The initial values of *Max-Value*, *Max-Num*, and *Cash-Exp* are set to a system default. This initial capital is necessary to make the whole market start to function at the beginning. However, malicious users could make use of the initial capitals to free-ride. For example, one could abundant an ID every time he has used up its initial capital, and register as a newcomer to get another share. This is called "white-washing" in [6]. To combat these abuses, we require that AC perform some sophisticated checking when a newcomer registers its ID. One possible method is to use e-mail confirmation, which is widely adopted in many applications on today's Internet.

4 Downloading and Paying

When a file is put on the market for sale, it is divided into several pieces as in BitTorrent. The metadata information (file ID, file name, description, size, hash values, provider ID, address, port, accepted TTP list, price, etc.) is recorded in the community’s index service and made public to all. A peer makes search queries to find what it wants. A list of matching records is returned to the querier. It then checks the list to see if a file is likely to be what it wants (file name, description, size, etc), if the price is acceptable and affordable, and if there is a TTP they commonly trusted, etc.

4.1 Negotiating

After a peer has decided which file to download, there will be a negotiating phase between the downloader (peer B) and the provider (peer A) before the download starts. The message flow of this phase is illustrated in Fig. 1. CC_X is peer X’s capital certificate (see section 4). The mark “||” denotes a sequence of two or more parts.

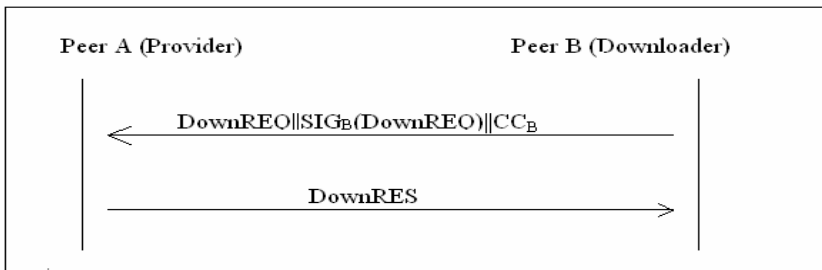


Fig. 1. Negotiating Protocol

$$\text{DownREQ} = \text{FileID}||\text{DownloaderID}||\text{ChequeSN}||\text{TS}. \tag{1}$$

Each peer maintains a personal counter. It is reset to zero each time the peer gets an updated capital certificate, and increased by one each time the peer issues a new payment cheque. “ChequeSN” is the current value of this counter. It should not be greater than *Max-Num* of this peer’s current capital certificate, or the request will be rejected by other peers. “TS” is the timestamp.

When A receives the signed DownREQ and the capital certificate from B, it first verifies the signatures using B’s and AC’s public key respectively. Then A makes sure that ChequeSN is no greater than *Max-Num* in the certificate and there is still enough time before *Cash-Exp*. If all these succeed, A randomly chooses a piece (to be encrypted) in the file and sends its sequence number in the DownRES message. Otherwise, A sends an error message to B, or just ignores this request.

4.2 Downloading

Then begins the downloading. If there are several concurrent downloaders, the provider will redirect some of the data requests between them, so that more bandwidth

can be spared to serve the rarest pieces (Due to space limitation, we won't dwell on the details here.). For each downloader there is an encrypted key piece (KP). This key piece can only be got from the original provider. When a downloader requests for the key piece, the provider will randomly choose a key K, encrypt the piece with K and send the encrypted piece to the downloader.

4.3 Paying

When the downloading finishes, downloader B gets the requested file with one of the pieces encrypted. To reconstruct the file, B has to get the decryption key (K). To get the key, B has to give the provider (A) a payment cheque as they have negotiated in section 4.1. The exchange of K and the cheque must be fair. The following protocol can make sure that B gets K *iff* (if and only if) A gets the cheque, and vice versa. The message flow is illustrated in Fig. 2. "C" denotes the payment cheque:

$$C = ID_{Payee} || ID_{Payer} || ID_{TTP} || ChequeSN || Price || TS || H(KP) || H(SE_K(KP)). \tag{2}$$

The value of "Price" shouldn't be greater than *Max-Value* in the payer's capital certificate. If the price is greater than this max value, two or more cheques (signed together) will be needed to cover the entire payment.

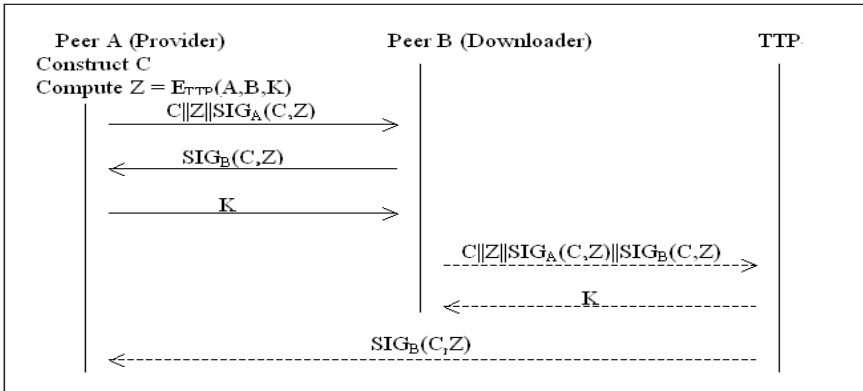


Fig. 2. Paying Protocol - a Fair Exchange Protocol

If the provider (payee) and the downloader (payer) are both honest, then the protocol only needs three message exchanges between them directly, without the interference of a third party (optimistic scenario):

- A1.** Peer A first constructs C and Computes $Z = E_{TTP}(A,B,K)$. Then it sends C, Z and $SIG_A(C,Z)$ to peer B.
- B1.** After B receives C, Z and $SIG_A(C,Z)$, it first check the contents of C and make sure that all fields of C are correct. Then B checks the validity of the signature. If all the checks succeed, B generates $SIG_B(C,Z)$ and sends it to A.
- A2.** If A receives properly signed $SIG_B(C,Z)$ from B, it sends K to B as plaintext.

In the case that the provider is dishonest, it may refuse to give K to the downloader (step A2) after it receives the downloader's signature on the check. In this case the downloader can contact the TTP (dotted lines in Fig. 2):

B2. B sends C , Z , $SIG_A(C,Z)$ and $SIG_B(C,Z)$ to the TTP.

TTP1. The TTP parses C to learn the ID of the provider and the downloader. Then it verifies the two signatures. If both signatures are correct, it decrypts Z with its own private key. If $D_{TTP}(Z)$ is a triplet and the first two parts are A and B , then it: (i) sends the third part (K) to B ; (ii) sends $SIG_B(C,Z)$ to A .

In another case, if the downloader is dishonest and tries to directly access the TTP to learn K and bypass step B1 after it receives C , Z and $SIG_A(C,Z)$, it has to send both $SIG_A(C,Z)$ and $SIG_B(C,Z)$ to TTP. Then, if K is sent to the downloader, $SIG_B(C,Z)$ will also be sent to the provider.

When this protocol finishes, A gets the payment and B can reconstruct the file using the key. Here a fair file-exchange is completed.

5 Accounting

Accounting operations are initiated by peers periodically. To reduce the burden of AC, peers are discouraged from frequently contacting AC by a certain amount of accounting fees imposed on each accounting request.

When a peer contacts AC and request for accounting, it gives AC a package of all the payment cheques it has earned since its last accounting. Each cheque in the package should be a quadruplet in the form of $(C, Z, SIG_x(C,Z), K)$.

AC maintains an account database for all the registered peers in the community. For each peer, AC stores the information about its balance, capital certificate, credit-cheques (this peer as the payee) and debit-cheques (this peer as the payer) that have been cashed but still not expired, etc.

When AC receives the package of cheques, it first rejects the expired ones. Then it checks the IDs and sequence numbers in the cheques to detect "double-accounting" and "double-spending". If it happens, the corresponding cheque will be rejected, and a *debit* (for example 10% of this cheque's nominal value) will be imposed on the payee (double-accounting) or the payer (double-spending) as a punishment. For each remained cheque, AC further checks if the following is correct: (1) $Z = E_{TTP}(ID_{payee}, ID_{payer}, K)$; (2) $SIG_{payer}(C, Z)$ is correct. Those do not satisfy these conditions will be deemed as invalid and rejected. After all these checkings, AC begins to update the balances of the involved peers according to the "Price" in the valid cheques. An accounting fee is also subtracted from the requester's account. At last, AC sends a confirmation message back to the requester, together with a list of accepted cheques.

If the requester's current capital certificate is about to expire or it has issued *Max-Num* checks, it will also apply for a new certificate. If its balance is greater than a certain lower bound, AC will issue a new certificate to it and updates the corresponding information in the database. Otherwise, the requester will have to "sell" some more files to earn enough "money", or it can no longer download any priced files from the community after its current certificate expires.

There is another problem to be considered. Due to the debits and accounting fees, the per-peer capital of the community will decrease as time passes by. That means the whole community will become poorer and poorer until no one can afford a download. However, intuitively as more files are shared and exchanged, the per-peer capital should increase gradually. Thus we add a 0.1% per day “credit interest” for each non-negative account. These interests also serve as an additional incentive for peers to share more files and accumulate more capitals.

6 Remarks and Conclusion

In this paper we describe how to construct a fair P2P file-sharing community. We name it a P2P file market. Our scheme is a “pay-per-transfer” scheme [7], forcing peers to contribute by a micropayment-based incentive mechanism.

The most important features of our scheme include: a fair-exchange protocol ensuring the fairness of peer-to-peer trading process (to our best knowledge, most P2P micropayment schemes do not guarantee fair exchanges), a sophisticated accounting policy, and the support for multi-source parallel download (it is one of the most attracting merits of P2P technology).

In the future, we will further examine if there are other “rational attacks” [8] possibly existing in our system. Also we will try to find an efficient way to distribute the role of the accounting center onto several peers. Finally we hope to implement a prototype in a real user environment soon and observe its performance in practice.

References

1. H. Zhuge, J. Liu, L. Feng, X. Sun and C. He. “Query Routing in a Peer-to-Peer Semantic Link Network”. *Computational Intelligence*, 21(2) pp197-216. (2005)
2. E. Adar and B. Huberman. “Free Riding on Gnutella”. *First Monday*, 5(10), (2000)
3. G. Hardin, “The Tragedy of the Commons,” *Science*, vol.162, pp1243–1248, (1968).
4. A. Shamir. “Identity-Based Cryptosystems and Signature Schemes”. In: *Proc. of Crypto’84*, LNCS-196, Springer Verlag, (1985).
5. Micali S. “Simple and fast optimistic protocols for fair electronic exchange”. In: *Proc. of ACM PODC*, (2003).
6. M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, “Free-Riding and Whitewashing in Peer-to-Peer Systems,” In: *Proc. of ACM SIGCOMM’04 Workshop on Practice and Theory of Incentives in Networked Systems (PINS)*, (2004)
7. B. Yang and H. Garcia-Molina. “PPay: Micropayments for Peer-to-Peer Systems”. In: *Proc. of ACM CCS’03*, (2003)
8. SJ Nielson, SA Crosby, and DS Wallach. “A Taxonomy of Rational Attacks”. In: *Proc. of the 4th International Workshop on Peer-To-Peer Systems*, (2005)