$\mathcal{P}^2 2\mathcal{P}$ - Prepaid Peer-to-Peer Services

Barbara Emmert
University of Würzburg, Germany
emmert@informatik.uni-wuerzburg.de

Oliver Jorns
Telecommunications Research Center Vienna, Austria
jorns@ftw.at

Abstract

Since the early days of file sharing, the quest for incentives persuading users to contribute resources to P2P networks exists. Crediting and debiting users' accounts for consumptions and contributions is surely one possibility. But how to do this without enormous additional overhead or infrastructure? We depict a solution to this question by describing the establishment of a lightweight, secure prepaid charging mechanism within a P2P network.

1. Introduction

Free riders, i.e. people using resources without contributing anything, endanger not only the stability of a file-sharing system, but of any P2P system, may it be used for decentralized service discovery or distributed network management. Thus, adequate incentives for resource contributions have to be found. Among others, one obvious incentive could be to credit and to debit resource contribution and usage with money.

Another reason that brings money to the P2P world can be the fact, that resources outside the P2P system are used. One example is Skype, where users have to pay if they call traditional phones. Others envision even the emergence of a competitive market of *DHT service providers* (DSP) that naturally charge for a commercial service delivery [4].

Sophisticated solutions for charging in P2P systems were e.g. proposed by [3]. We present a simpler solution, that avoids the effort of credit checking and collection of consumption data by introducing a fraud resistant decentralized prepaid solution for accounting contributions and consumptions in a P2P network and for charging P2P services. In Section 2, we familiarize the reader with the concept of hash based accounting which we discuss in Section 3 before we conclude in Section 4.

2. Hash Based Accounting

To illustrate how to establish a payment mechanism for a DHT application, we focus on the case of storage applications like OpenDHT [4]. In those systems, any user can issue *put*- and *get*-requests for storing and retrieving data using dedicated keys to any peer. We assume that a *peer* is an arbitrary computer running a piece of code, which makes it a member of the specific P2P system. This application offers a user interface and manages the accounting and forwarding of data storage and retrieval requests. Keep in mind, that even if we use storage applications to describe our ideas, the proposals are suitable for all applications using a DHT routing layer.

To allow any peer to charge a user u's account for a demanded service, we propose to record the account balances within the DHT. In the following, we focus on the situation where resources are paid and a DSP exists, but other scenarios like crediting storage or bandwidth contributions, or a decentralized accounting algorithm could be imagined likewise. In Section 2.1, we present how to keep the accounts of the users securely within the DHT. In Section 2.2 we describe the concept of *authenticated refill vouchers* that enable users to increase their account balances.

2.1. Accounting within the DHT

For charging DHT services, each user, known by his system-wide unique identifier u, needs a personal account $a_u = [c_u, \sigma_u]$. c_u indicates u's actual credit, σ_u is a certificate of identity only needed for ensuring that just the person who established a_u can use it. σ_u can hence be an arbitrary bit string satisfying standard password requirements chosen by u on creating his account, but has to be kept secret.

The most evident approach to hold a_u in the DHT storage system would be to determine the responsible peer by assigning the key H(u) to a_u, where H is the collision-resistant hash function deployed by the underlying DHT algorithm. Of course, anyone could guess this trivial key and hence easily manipulate u's balance. We therefore use *Hash based Message Authentication Codes* (HMAC) [1], which enable two parties holding a common secret key K to generate hash values that are completely different from the outcome of the standard hash function.

K has to be known to all peers to enable them to charge a request of user u: Given K, any peer can calculate the key k_u that determines which peer stores a_u as

$$k_u = \text{HMAC}_K(u) = \text{H}(K \oplus \text{opad}, \text{H}(K \oplus \text{ipad}, u)),$$

where "ipad" and "opad" stand for the bytes 0x36 and 0x5C repeated 64 times respectively [1], $\oplus$ denotes the bitwise XOR and a comma the concatenation of two bit strings.

To decentralize the billing process, it is furthermore required that all peers know the tariff function, i.e. the amount of credit that is needed for storage or retrieval operations. One very simple possibility would be to charge a constant amount f_s for storing items and to grant the free access of data. Another possibility could be a tariff that depends on the time for which the DHT infrastructure is used, similar to the mechanism proposed by Kurtansky and Stiller [2]. Anyhow, by deploying the P2P software, the tariff function can be learned by all peers. Another possibility would be to store it in the DHT under a well known key.

Using the case of a constant storage fee f_s, we illustrate how u could be charged for a data storage request, the case of data access or VoIP services can be imagined likewise:

1. u issues the command to store $data$ under the key k to a peer p, presenting his certificate of identity, σ_u to p.
2. p retrieves $a_u = [c_u, \sigma_u]$ stored under k_u and checks u's account and identity. If u's identity is verified and his account allows the operation, p charges the storage, i.e. updates u's balance to $c_u - f_s$.
3. If c_u is sufficient, p takes care that $data$ will be stored by the peer responsible for k.

If in step 2, $c_u < f_s$, this is reported to u and the put-request is discarded or delayed until u has refilled his account.

2.2. Authenticated Refill Vouchers

In analogy to the plastic made refill cards known in the world of mobile communication services, we propose to purchase certificates that enable users to credit their accounts. We describe the reload-procedure for a user u:

1. u, using σ_u as proof of identity, pays money or contributes resources to increase his account by x.
2. The DSP checks, if σ_u matches u, if yes, it issues the voucher v_x to u.
3. u presents v_x, u and σ_u to an arbitrary peer p.
4. p verifies if v_x is valid and if u is correctly identified by σ_u, if yes, it updates u's account balance to $c_u + x$.

Now, what do we require from an "authenticated refill voucher"? First, it has to certify, that u's account may be increased by x. Next, it has to be non-forgeable, non-reproducible and valid only for u. Therefore, we propose a voucher to be a simple bit string, created as

$$v_x = [x, t, \varsigma] \text{ , with } \varsigma = \text{HMAC}_K(x, t, u).$$

Given K, any peer can recompute ς to verify if v_x is valid for u. The time stamp t prevents that vouchers are reused.

3. Discussion

The use of HMAC enables decentralized fraud resistant charging of P2P services, that is computationally feasible even for hardware constrained machines: Knowing K and using the hash function that is used within the deployed DHT, each peer can verify the identity of any user and can charge resource consumptions to his account. Obviously, it is vital, that no third party gets to know K, as this would allow modifications of accounts and creation of false vouchers. In most deployed DHT algorithms, cryptographic hash functions are used and keys are 160 bit long. For this scenario, all known attempts of recovering K from keys or vouchers fail to be practical [1].

Fraud is nevertheless possible, if user v is able to associate k_u or c_u with another user u. Therefore all communication of users with peers and on the DHT routing layer has to be encrypted. To make collusions and insider attacks impossible, and to prevent that a malicious user of the system communicates K to a third party, we propose furthermore to hide K from the users of the system. If only the applications running on the used nodes are able to access K, as it is e.g. encrypted within the code, this would offer a solution to the mentioned problems. However, a bulletproof security concept is beyond the scope of this work.

Another potential manipulation point lies in the charging mechanism: to prevent abuse, it has to be assured, that only correct fees are charged, and that accounts are updated properly. For this purpose, forwarding of digitally signed user requests and vouchers could be imagined.

4. Conclusion and Outlook

In this work, we presented a lightweight solution for charging P2P services. We outlined how a prepaid billing mechanism can be established easily within an arbitrary DHT infrastructure. Our concept is not only open for other incentive mechanisms, the described accounting structure can be used for various other administrative purposes. The use of HMAC makes it computationally near to infeasible to manipulate this system.

Despite its simplicity, we believe in the potential of our concept. To make it more suitable for applications, future research will be dedicated to the consideration of security and privacy aspects, to the prevention of manipulations and to the examination of various tariff functions.

References

[1] M. Bellare et al. Keying Hash Functions for Message Authentication. In *CRYPTO 96*, Santa Barbara, California, USA, 1996.

[2] P. Kurtansky and B. Stiller. Time Interval-Based Prepaid Charging of QoS-Enabled IP Services. In *WINE 2005*, Hong Kong, China, 2005.

[3] N. Liebau et al. Charging in Peer-to-Peer Systems based on a Token Accounting System. In *ICQT'06*, St. Malo, France, 2006.

[4] S. Rhea et al. OpenDHT: A Public DHT Service and Its Uses. *ACM SIGCOMM Computer Communications Review*, 35(4), 2005.