Securing P2P Storage with a Self-organizing Payment Scheme

Nouha Oualha[1] and Yves Roudier[2]

[1] Telecom ParisTech, Paris, France
Tel.: +33 (0)1 45 81 77 41, Fax: +33 (0)1 45 89 79 06
oualha@telecom-paristech.fr
[2] EURECOM, Sophia Antipolis, France
Tel.: +33 (0)4 93 00 81 18, Fax: +33 (0)4 93 00 82 00
roudier@eurecom.fr

Abstract. This paper describes how to establish trust for P2P storage using a payment-based scheme. This scheme relies on the monitoring of storage peers on a regular basis. The verification operations allow assessing peer behavior and eventually estimating their subsequent remuneration or punishment. The periodic verification operations are used to enforce the fair exchange of a payment against effective cooperative peer behavior. Payments are periodically provided to peers based on the verification results. Only cooperative peers are paid by data owners with the help of intermediates in the P2P network, thus accommodating peer churn. Moreover, our payment scheme does not require any centralized trusted authority to appropriately realize a large-scale system. Simulations in this paper evaluate the capability of the payment scheme to work as a sieve to filter out non cooperative peers.

Keywords: P2P storage, cooperation, Storage reliability, payment based scheme.

1 Introduction

The tremendous growth in the amount of data available in today networks and the trend towards increasing self-organization, as illustrated by large scale distributed systems and mobile ad hoc networks, generate a growing interest in peer-to-peer (P2P) data storage. Such an application exposes the data to new threats due to peers since any service deployed in self-organizing networks first and foremost relies on their willingness to cooperate. In this setting, selfishness not only covers explicit non cooperation, but also subtle forms of misbehavior whereby peers may discard some data they promised to store in order to optimize their resource usage or may try to escape their responsibility in contributing to the maintenance of the storage infrastructure.

Approaches to overcome non cooperating behaviors fall into two classes: reputation and payment (or remuneration) mechanisms. Reputation builds a long-term trust between peers based on a statistical history of their past interactions.

On the other hand, payment-based mechanisms feature a more immediate reward for a correct behavior making possible an explicit and discrete association of incentives and cooperation. With payments, peers cooperate because they are getting paid not because of a hypothetic reciprocation in the future as can be found with reputation systems.

Several payment schemes have been introduced to enable ad hoc packet forwarding ([1], [2]). It is however quite easy to determine individual and instantaneous rewards fostering cooperation in exchange of a packet forwarding operation, whereas data storage can only be deemed as correct at data retrieval time. Furthermore, it is useful in this case to be aware of a storage failure as soon as possible. Other payment schemes are not attached to a particular resource sharing application ([19], [20], [21], and [22]); but they unfortunately rather suppose the presence of a centralized authority to mediate peer payments, and they therefore may undermine the scalability of a P2P system.

The contributions of this paper are summarized in the following. A description of the problem statement (Section 2) that particularly defines our threat model and related work (Section 3). The design of a payment based scheme for mitigating the impact of selfishness towards P2P storage (Section 4). The proposed scheme relying on KARMA [14] for peer account administration is based on a cryptographic protocol that enforces the periodic fair-exchange of payment against cooperative behavior. An evaluation of that scheme (Section 5) based on simulation results demonstrating its capability to detect and punish non cooperative peers and an analysis of its security.

2 Problem Statement

P2P cooperative storage systems feature peers that store data for other peers; in counterpart, peers receive some disk space for the storage of their own data. Such systems face two major issues: reliability, since holder peers may crash or delete the data they store; and fairness, since peers may refuse to store data for others. The data stored can be periodically checked using one of the numerous verification protocols discussed in [4] and [5]. [4] discusses how verification can be mostly handled by verifier peers selected and appointed by the data owner to distribute the load of this task. The current paper discusses how to enable such a system using monetary compensations for the verification task.

Threat model. P2P storage relies on cooperation with unknown peers without prior trust relationships. However, peers may misbehave in various ways. In comparison with the study of selfish behaviors in routing over mobile ad-hoc network, we distinguish two types of selfishness:

- **Passive selfishness:** Passively selfish peers store their own data in the system but never concede disk space for the storage of other peers. This problem has long been referred to as free-riding. In a storage system like the one described in [4], free-riders will also refuse to perform data storage verifications

although they will request other peers to check the correct storage of their own data.

- **Active selfishness:** Actively selfish peers accept to store data although they will not in effect store them for the agreed upon period of time but instead destroy them at some point to optimize their resource usage. Actively selfish peers also agree to become verifiers, although they will defect in some way as discussed below. Data owners wrongly depend on actively selfish peers contrary to passively selfish ones. Such a behavior constitutes an attack to the cooperation scheme much more offensive because it wastes network bandwidth and lowers data reliability and trust towards verifiers.

Active selfishness involves addressing the following issues:

- *Resource related collusions:* peers may collude to reduce their resource usage (storage space, computation, or bandwidth utilization). Holders may destroy all replicas of a data but one and still be able to answer verification requests. This attack might be addressed by personalizing data replicas at each holder ([4], [18]). Another type of collusion may occur between a holder and its verifier such that neither the holder stores data nor the verifier performs verification operations. This can be avoided by distributing the verification task to several verifiers, then checking the consistency of their answers to decide on the trustworthiness of both the holder and the verifiers. If verifiers might defect altogether (Quis custodiet ipsos custodes[3]), the owner might ultimately perform sampling verifications itself, which should be done sparingly, for obvious resource related matters.
- *Remuneration related vulnerabilities:* attacks may directly target the incentive mechanism to defeat its filtering capabilities. The Sybil attack represents another potential vulnerability making it possible to generate new peers at will. The Sybil attacker masquerades under multiple simultaneous peer identities in order to gain an unfair advantage over system resources. The payment based mechanisms to be envisioned should therefore support some form of real world based authentication. This attack should at least be mitigated by imposing a real world monetary counterpart to membership for peers joining the storage system. In terms of implementation, the payment mechanism should also prevent or mitigate in some respect double spending attacks.

Purely malicious behaviors are also possible: a malicious peer might aim at obtaining or altering some private data, or at destroying the data stored or the infrastructure through denial of service attacks (e.g., flooding), even at the expense of its own resources. This paper only focuses on selfish behaviors, in particular active ones, and their mitigation through a (micro)payment based incentive scheme.

[3] who watches the watchers?

3 Related work

A number of micropayment schemes have been proposed in the past like PayWord, MicroMint [20], and Millicent [21]. Those rely on a central broker, the bank, that tracks each peer balance and payment transactions. In most of these schemes, the load of the bank grows linearly with the number of transactions; though hash chains in PayWord or the use of electronic lottery tickets [23] greatly reduce such cost. As example, the P2P micropayment system MojoNation [19] has also a linear broker's load and this system has gone out of work because the central bank constitutes as well a single point of failure.

One obvious way of distributing bank's work would be to use super-peers disseminated within the P2P network. These super-peers would provide neutral platforms for performing a payment protocol. The use of such an infrastructure of trusted peers may make sense, in particular in relation with content distribution networks (CDNs) [3]. Such networks involve the deployment of managed workstations all over the Internet, thereby providing a nice platform for other functionalities.

The scale of the P2P system makes it necessary to resort to a type of protocols termed optimistic protocols where the bank does not necessarily take part in the payment, but may be contacted to arbitrate litigations between peers. With such type of protocols, the bank's work is reduced. PPay [22] is a lightweight micropayment scheme for P2P systems where the issuer of any coin is a peer from the system that is responsible for keeping trace of the coin. However, the bank comes into play when the issuer of a coin is off-line. In a very dynamic system, the probability of finding the original issuer of the coin on-line is very low. In this situation, PPay converges to a system with a centralized bank.

If data storage should be achieved in a large-scale and open P2P system, designs based on a trusted authority may be unfeasible or unmanageable. In that case, implementing the optimistic fair exchange protocol would have to be done by relying solely on peers. [7] describes design rules for such cryptographic protocols making it possible to implement appropriate fair-exchange protocols.

To the best of our knowledge, the only fully-decentralized micropayment scheme that has been proposed so far is KARMA [14]. KARMA splits the bank functionality in different bank sets composed of peers selected and appointed randomly from the P2P system for each peer when it first joins the system. The KARMA payment scheme does not require any trusted infrastructure and is scalable.

The payment scheme proposed in this paper relies on such framework to administer peer accounts. Enforcing the fair-exchange of payment against a cooperative service is based on a cryptographic protocol (described in Section 4) that is built on top of the KARMA framework. The framework has been initially applied to the file sharing problem described in [14]. The described application cannot be assimilated to a P2P storage application since in the former case payments are immediately charged after the exchange of the file, whereas in the latter case payments for storage or verification are by installment i.e., they are billed at a due date that corresponds to the confirmation (by verifications) of

the good behavior of the holder or the verifier. Therefore, we will supplement the KARMA framework by an escrowing mechanism (described in detail in Section 4) that guarantees the effective payment of credits promised by the owner towards a cooperative holder or a cooperative verifier.

Tamper resistant hardware (TRH) have also been suggested as a way to enforce payment protocols in a decentralized fashion as illustrated by smart cards in [6]. TRH supported approaches have been suggested within the TermiNodes [1] and CASHnet [2] projects as a support for implementing cooperation incentives. TRH based approaches suffer from other attacks on the payment scheme: if the TRH of a non cooperative or malicious peer is disconnected from the other peers, their credits/tokens might not be available, which might raise starvation issues. However, the use of a secure operating system as a TRH might make it possible alleviate this problem notably by more completely controlling and possibly reducing the device functionalities if the peer does not connect to the system network.

4 Payment-based Storage

In this section, we first give an overview of the payment-based storage scheme, to describe then the cryptographic protocol that achieves such scheme.

4.1 Overview

We propose a mechanism that monitors data storage on a regular basis to determine the payments between data owners, holders, and verifiers. The payment mechanism allows a peer storing data for other peers to be paid for its service. It thus controls the storage functions seen in the previous section by rewarding cooperating peers.

Notations: Let B_P denote the bank-set of the peer P, PK_P the public key of a peer P, SK_P the private key of P, and SK_{B_P} the private key of the bank-set of P. A message M signed by some key K is denoted as M_K or σ_K (bank-set signature is explained in [14]).

Let G denote a finite cyclic group with n elements. We assume that the group is written multiplicatively. Let g be a generator of G. If h is an element of G then finding a solution x (whenever it exists) of the equation $g^x = h$ is called the discrete logarithm problem (DLP) and is assumed hard to solve.

Assumptions: A P2P system generally consists of altruistic peers, selfish peers, malicious peers, and others with behavior ranging in between. We will assume that there are a non-negligible percentage of the peers that are altruistic or at least correctly follow the protocol. Peers of the storage system are structured in a DHT such as CAN [8], Chord [10], Pastry [9], or Tapestry [11]. A DHT consists of a number of peers having each a key $Key(Peer)$ in the DHT space, which is the set of all binary strings of some fixed length. Each participant is

assigned a secure, random identifier in the DHT identifier space: $Id(Peer)$. We assume that the DHT provides a secure lookup service (see [12] and [13]): a peer supplies an arbitrary key (an element in the DHT space), and the lookup service returns the active node in the DHT that stores the object with the key.

Peer selection: To avoid pre-set collusion between verifiers and the owner or verifiers and the holders, holders and verifiers should be randomly chosen. The random selection of peers is generally used for its simplicity since it is less sophisticated and since it consumes less bandwidth per peer. TotalRecall [24] and the P2P storage system of [25] rely on a DHT to randomly select data holders. The selection is realized by randomly choosing a value from the DHT address space and routing to that value. [26] proved the positive effects of randomization in peer selection strategies. For instance, a long-list of randomly chosen potential holders and verifiers can be constructed: the l_1 (respectively l_2) closest peers in the DHT identifier space to the key $HASH_H(Id(Owner), timestamp)$ (respectively $HASH_V(Id(Owner), timestamp)$) constitute the potential holders (respectively verifiers) of the owner ($HASH_H$ and $HASH_V$ are pseudo-random functions publicly known).

Credit escrowing: Each peer has a personal account managed by a set of peers likewise KARMA [14] that are called bank-set. Our payment scheme relies on digital checks. To prevent peers from emitting bad checks, the amount of credits that corresponds to a check value are escrowed, i.e., the necessary number of credits to pay check holder are locked by the bank-set. Consequently, bank-sets keep two types of peer balances: normal credits and locked credits. Credits are escrowed for some time-out (that corresponds to the check's expiry time), after which they are returned to the peer normal balance. The owner desiring to store data in the system must be able to pay its holders and verifiers with checks. That's why, it must escrow credits which are converted to digital checks. These checks are then stored in a blinded version at the corresponding holders and verifiers. Checks include some random numbers that are generated by the owner and certified by its bank-set. The latter have a blinded version of these numbers too (to prevent collusion between one bank-set member and a holder or a verifier). Each blinded digital check has this form:

$$C(payer, payee, g^c) = \{g^c, id(payer), id(payee), price, seq, TTL, \sigma_{SK_{payer}}\} \quad (1)$$

having c a random number, seq check's sequence number, and TTL check's expiry time. The knowledge of c by the payee allows this latter to be paid credits of value price. The bank-set of the payer is not informed of this number c; but only a blinded version of it g^c. The verification operation allows both the verifier and the holder to extract the check in order to be able to present it to their bank-set to be paid in return. The holder must also escrow an amount of credits corresponding to the punishment it gets if it destroys data that it has promised to store. The escrowed credits of the holder are converted to one digital check that is certified by the holder's bank-set. The check is split into multiple

shares each one will be stored at each verifier: a threshold number k of these shares allow reconstructing the full check. Shares of the digital check comprise the following numbers (in blinded version) $\{g^{s_i}\}_{1\leq i\leq n_v}$ which are shares of g^s if $\{s_i\}_{1\leq i\leq n_v}$ are shares of s [16]. If a threshold-based majority of verifiers agree that the holder has destroyed data, they can construct holder's check and present it to the owner such that this latter will be reimbursed.

Data verification: Each verifier appointed by the owner periodically checks storage of data stored at a holder. The verifier does not have the full challenge for the holder, but rather a share of the challenge: a threshold number of messages received by the holder from verifiers allow this latter to reconstruct the full challenge. Distributing the verification task to multiple verifiers prevents potential collusion between the holder and a verifier, since the holder cannot compute the response without the stored data even by having a partial knowledge of pre-computed challenges (metadata) disclosed by a small number of colluding verifiers. The verification operation has three-fold objectives: it allows assessing the availability of stored data, it permits the verifier to remove the blinding factor of the stored digital check in order to get paid for verification, and finally it allows the holder to recover also its check for its payment too. Since, the verifier is paid exactly for each verification operation it actually performs, verification operations are executed in a defined number. Consequently, the payment scheme does not require a verification protocol where verifications are unlimited and may rely on pre-computed challenges for instance.

The originality of the scheme mainly stems from the combination of verification and payment operations such that the scheme works without the data owners being involved. These latter can store their data at multiple holders, appoint verifiers, and then they can forget about the stored data until of course the moment of retrieval. The system of peers operates automatically without the intervention of owners: it periodically checks data storage and pays the cooperating holders.

4.2 Protocol

In this section, we describe a protocol that provides a cryptographic implementation of the scheme. We consider an owner denoted O that stores its data at a holder H. The integrity of such data is periodically checked by a verifier V on behalf of O. The proposed protocol consists in multiple steps described in the following Fig. 1.

5 Simulation experiments

In order to validate the ability of our payment-based storage approach to detect and punish selfish peers, we developed a custom simulator of our payment scheme. This section first describes the framework of simulations, then presents and analyzes the results obtained.

Operations	**Description**
Credit escrowing	**(*O* escrows credits for the payment of *H* and *V*)** *O*: fix number of verification operations to *m* *O*: generate random numbers $\{R_i\}_{1\leq i\leq m}$, v_H, v_V *O*: compute for each $i\in[1, m]$ $T_i = HASH(HASH(d, R_i), v_H)$ $T'_i = HASH(HASH(d, R_i), v_V)$ $O \rightarrow B_O$: $\{g^{T_i}, price\}_{1\leq i\leq m}, id(H), \{g^{T_i}, price\}_{1\leq i\leq m}, id(V)$ $B_O \rightarrow O$: $\{C(O, H, g^{T_i})\}_{1\leq i\leq m}$, $\{C(O, V, g^{T'_i})\}_{1\leq i\leq m}$ **(*H* escrows credits to form its punition *p*)** *H*: generate a random number *s* *H*: generate $\{s_i\}_{1\leq i\leq n_v}$ shares of *s* $H \rightarrow B_H$: $\{g^{s_i}\}_{1\leq i\leq n_v}$, $g^s, id(O)$, *punition* B_H: check $\{g^{s_i}\}_{1\leq i\leq n_v}$ are shares of g^s $B_H \rightarrow H$: $\{C(H, O, g^{s_i})\}_{1\leq i\leq n_v}$
Data storage	**(*O* stores data *d* at *H*)** $H \rightarrow V$: $s_i, C(H, O, g^{s_i})$ $V \rightarrow H \rightarrow O$: $\{ACK\}_{SK_V}$ $O \rightarrow H$: *d*, $\{C(O, H, g^{T_i})\}_{1\leq i\leq m}$, v_H **(*O* delegates verification of *d* to *V*)** *O*: generate for each $i\in[1, m]$ $\{r_{ij}\}_{1\leq j\leq n_v}$ shares of R_i *O*: compute $\{HASH^2(d, R_i)\}_{1\leq i\leq m}$ ($HASH^2$: *HASH* is executed 2 times) $O \rightarrow V$: $\{HASH^2(d, R_i)\}_{1\leq i\leq m}$, r_{ij}, $\{C(O, V, g^{T'_i})\}_{1\leq i\leq m}$, v_V
Data verification	**(*V* sends a share of the i^{th} challenge to *H*)** $V \rightarrow H$: i, r_{ij} **(*H* answers verifiers upon construction of challenge from shares)** *H*: compute $Res=HASH(d, R_i)$ $H \rightarrow V$: *Res* **(*V* checks *H*'s answer)** *V*: check $HASH(Res) =? HASH^2(d, R_i)$
Payment	**(*H* obtains its i^{th} payment)** *H*: compute $T_i = HASH(HASH(d, R_i), v_H)$ $H \rightarrow B_H$: T_i, $C(O, H, g^{T_i})$ $B_H \rightarrow B_O$: T_i, $C(O, H, g^{T_i})$ B_H: increase *H*'s balance B_O: decrease *O*'s balance **(*V* obtains its i^{th} payment)** *V*: compute $T'_i = HASH(HASH(d, R_i), v_V)$ $V \rightarrow B_V$: T'_i, $C(O, V, g^{T'_i})$ $B_V \rightarrow B_O$: T'_i, $C(O, V, g^{T'_i})$ B_V: increase *V*'s balance B_O: decrease *O*'s balance
Data retrieval	**(*O* retrieves *d* from *H*)** $H \rightarrow O$: *d* **(*H* unblocks its escrowed credits)** $O \rightarrow H \rightarrow B_H$: $\{ACK\}_{SK_O}$ If *TTL* times out: unspent escrowed credits are returned (respectively to *O* and *H*)

Fig. 1. Payment protocol

5.1 Simulation framework

The self-organizing storage system is modeled as a closed set of homogeneous peers. The storage system operation is modeled as a cycle-based simulation. One simulation cycle corresponds to the period between two successive verifications.

Churn: Peers arrive to the system in Poisson distribution: there are 100 newcomers per hour, for an average lifetime of 2 weeks. [17] shows that Gnutella peer uptime follows a power-law distribution. We will use the same distribution for peer uptime and downtime. In average, a peer stays online for 1 hour and connects in average 6.4 times in a day.

Storage: Peer storage space, file size, and storage duration are chosen from truncated log-normal distributions. The storage space of each peer is chosen from 1 to 100GB, with an average of 10GB. In each day of simulated time, 2.85 of files are stored per peer for an average period of 1 week. The average file size is 500MB. The stored files will be checked by verifiers each day.

User strategies: We consider three peer strategies: cooperation, passive selfishness (free-riding) and active selfishness.

- **Cooperative:** whenever the peer accepts to store data from another peer, it keeps them stored. Whenever the peer accepts to check the availability of some data at a storage peer, it will periodically perform verification operations on this peer as agreed.
- **Passively selfish:** the peer will never accept to store data and will never accept to verify the availability of some data stored for other peers. The peer is just consumer of the storage system. This type of behavior is also termed free-riding.
- **Actively selfish:** the peer probabilistically accepts to store data for other peers or to verify storage at other peers. Whenever it stores or verifiers for others, it will fulfill its promise only probabilistically. This type of behavior has an instability effect: alternating between cooperation and selfishness at a given probability: probability of participation denoted p and probability of achieving promise denoted q.

5.2 Simulation results

Different scenarios were simulated to analyze the impact of several parameters on the payment mechanism. Simulation studies the transition phase of the network to a stable state where cooperative peers are the only active actors of the system. Used notations are summarized in Table 1.

Exclusion of selfish owners: Fig. 2. demonstrates that selfish peers have less capability over time to store data in the system; on the other hand, cooperative peers are becoming the majority of data owners in the storage system. Passive

Table 1. Description of notations used in simulation figures

Notations	Description
n	Number of peers in the system
r	Data replication factor (number of holders of some given data)
m	Number of verifiers of some given data
p	Probability of participation of an actively selfish peers
q	Probability of data conservation by an actively selfish peer
w	Weight parameter in price formulation (associated with the amount of credits owned by a given peer)

selfish peers are the first to be excluded from the system because they consume all their initial credits (all peers have a default number of credits when they join, in order to facilitate system bootstrap). Active selfish peers are also filtered out from the system because they cooperate only probabilistically. The figure shows also that a decreasing fraction of these active selfish peers are still present in the system. Because they cooperate at some probability; they may temporarily gain some credits and then go without detection. These are considered as the false negatives of our detection scheme. But still, such false negatives are decreasing with time. We may notice that 1 simulated month is sufficient to filter out passively selfish peers; however the filtering may take more than 3 months for actively selfish peers. Yet, this time period can be reduced by adaptively reducing the default initial income for newcomers.

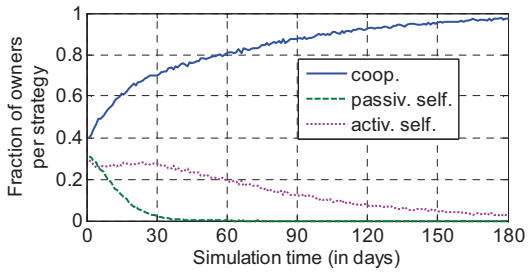

Fig. 2. Averaged ratio of owners per strategy. n=1000, r=3, m=5, w=0.5, p=0.2, q=0.2, 40% cooperators, 30% passively selfish peers, 30% actively selfish peers.

Exclusion of selfish holders: Fig. 3. depicts the fraction of cooperators and selfish peers in the population of data holders. The figure demonstrates that with time cooperative peers will make the majority of holders. This result is due to the fact that actively selfish peers are losing their credits and then becoming unable to escrow credits necessary for the storage of other peers' data; albeit the fact that

they will propose small prices (this explains the small pick in the first simulated month).

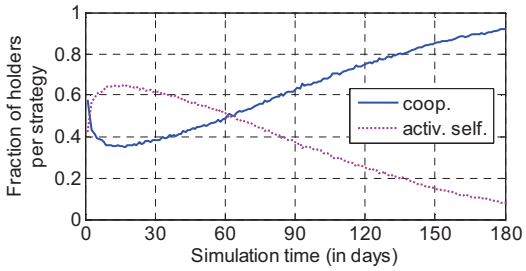

Fig. 3. Averaged ratio of holders per strategy. n=1000, r=3, m=5, w=0.5, p=0.2, q=0.2, 40% cooperators, 30% passively selfish peers, 30% actively selfish peers.

Overhead: Note that we only measure the communication overhead due to holder and verifier selection and storage verification. In particular, we exclude the cost of P2P overlay maintenance and storing/fetching of files, since it is not relevant to our analysis. In a further observation, the bandwidth consumed for verification is dependent on the number, rather than the size, of files being stored. This is in fact a requirement on the verification protocol. Fig. 4 shows the amount of control messages per file. The figure demonstrates that the bandwidth cost decreases with time.

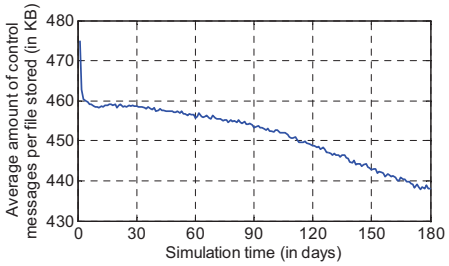

Fig. 4. Average amount of control messages per file stored (in KB). n=1000, r=3, m=5, w=0.5, p=0.2, q=0.2, 40% cooperators, 30% passively selfish peers, 30% actively selfish peers.

Data reliability: Fig. 5. shows that the rate of the amount of data injected into the storage system decreasing. This is due to several factors. First of all, there

is the gradual exclusion of selfish peers that limits the number of peers able to store data in the system. Second, there are possible false positives of our detection system due to the starvation phenomenon where cooperative peers are not able to contribute because they are not chosen as holders or verifiers, and at the end they consume all their credits and get expelled from the system. The figure also depicts the rate of file loss that is falling down as low as zero, owing to the exclusion of selfish holders (explained earlier on).

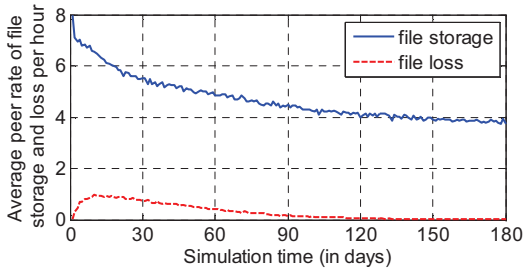

Fig. 5. Average peer rate of file storage and loss per hour. n=1000, r=3, m=5, w=0.5, p=0.2, q=0.2, 40% cooperators, 30% passively selfish peers, 30% actively selfish peers.

5.3 Security considerations

In this section, we analyze the security of the protocol to prevent or at least mitigate the threats described in section 2. The security of our scheme relies principally on replication to deter peers that might try to subvert the protocol. It assumes that there are at least a given number of peers in the system at all times, and uses protocols to ensure that the system will correctly operate unless a substantial fraction of these peers are selfish or malicious.

Selfishness punishment: The proposed payment scheme works as a quota system: peers have to keep a given balance to be able to participate to the storage system. Peers that are passively selfish gradually consume all their credits for their data storage and when their accounts are exhausted they will not be able to use the storage system anymore. In the same way, actively selfish peers keep losing credits because they have been detected destroying data they have promised to store. These peers will also drain their accounts and with time will not able to use the storage system.

Collusion prevention: Holder and verifier selection is random which limits preset peer collusions. The digital check of the holder is shared among verifiers, thus mitigating also collusion between the owner and one or a small number of verifiers. Additionally, challenges sent to the holder are constructed cooperatively

by verifiers to avoid collusion between the holder and one or a small number of verifiers. Finally, collusion between the owner and the holder is less probable because it does not generate any financial profit since the owner must pay verifiers to check holder's storage. The distribution of tasks to several verifiers limits collusion; but it is still feasible if at least k verifiers collude with the holder for instance. The probability of collusion can be computed as:

$$\sum_{i=k}^{n_v} \binom{n_v}{i} p^i (1-p)^{(n_v - i)} \tag{2}$$

where p is the probability that a given verifier is not honest (colluder). However such collusion probability is less than 0.1 for 60% of dishonest peers (i.e., $p = 0.6$) in the system and with $n_v = 10$ and $k = 8$, or 80% of dishonest peers and with $n_v = 20$ and $k = 18$.

Fair-exchange: Holder and verifier payments are strongly related to the correct operation of data verification. This motivates the holder to accept storage verifications and incites verifiers to perform this task periodically on behalf of the owner. The frequency of verifications is determined by the owner at the time of delegating verification. This frequency is the matter of all verifiers: the majority of verifiers use the determined frequency at which the holder collects a sufficient number of random numbers to compute the challenge; therefore very fast or very slow frequencies of some verifiers do not influence (with large probability) the actual frequency of computing the verification challenge. The holder or the verifier cannot cash their checks without verifying the stored data. This is due to two reasons. First of all, the secret random numbers included in the checks are only known to the actual payers since they are held in DLP-based blinded version at the payees and bank-sets too. Second, $HASH$ is a one-way function, so knowing $HASH$(m) does not give any extra information on m. Therefore, the data verification operation strongly relates to the holder and the verifier payment operations. However, the existence of such relation is only guaranteed by the owner. So, if a verifier or a holder is still not paid even though it behaves well, it has the possibility to prove owner's misbehavior to the other participants (using the certified checks) and also to stop cooperating with the owner without being punished. Thus, the owner is encouraged to provide this type of relation to secure the future cooperation of peers handling its data. The bank-set comes into play to guarantee that payments are actually doable since the corresponding amounts of credits are locked to prohibit the payer from emitting bad checks.

Remuneration-related attacks: Attacks on the payment scheme (such as double spending or impersonation) are handled by the KARMA framework. Moreover, the sequence number and the identity of the payee included in each payment receipt prevent replay attacks, because they impose that the digital check is only cashed by the payee one time. We assume that all exchanged messages are signed and encrypted by the keys of the involved parties in order to ensure the integrity of exchanged messages and even the security against man-in-the-middle

attack for instance. Sybil attacks are mitigated a la KARMA by compelling peers to execute a cryptographic puzzle before joining the storage system, the result of which will be used to construct their identities.

6 Conclusion

We described a new payment scheme as an incentive for P2P storage. The scheme is scalable and cost-effective since it does not require any trusted infrastructure. It relies on a cryptographic protocol to enforce the fair exchange of payments against cooperative behavior. The protocol is self-powered: it does not require the mediation of data owners during the data verification process or for payments. Additionally, the design of the scheme takes into consideration several types of peer collusions, and achieves an effective punishment of selfishness. Our simulations outline that this scheme makes the system converge to a stable state where non-cooperative peers are finally filtered out. We are further investigating the formal validation of this approach.

References

1. Levente Buttyán and Jean-Pierre Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks. Technical report, EPFL, 2001.
2. Attila Weyland, Thomas Staub and Torsten Braun. Comparison of Incentive-based Cooperation Strategies for Hybrid Networks. 3rd International Conference on Wired/Wireless Internet Communications (WWIC 2005), pp 169-180, ISBN: 3-540-25899-X, Xanthi, Greece, May 11-13, 2005.
3. Akamai technologies, inc. http://www.akamai.com/.
4. Nouha Oualha, Melek nen, and Yves Roudier. A Security Protocol for Self-Organizing Data Storage. 23rd International Information Security Conference (SEC 2008), Milan, Italy, September 2008.
5. Nouha Oualha, Melek nen, and Yves Roudier. A Security Protocol for Self-Organizing Data Storage. (extended version) Technical Report N RR-08-208, EURECOM, January 2008.
6. Holger Vogt, Henning Pagnia, and Felix C. Grtner, "Using Smart Cards for Fair Exchange", Lecture Notes In Computer Science, Vol. 2232, in Proceedings of the Second International Workshop on Electronic Commerce, p. 101 - 113 , 2001, Springer-Verlag
7. N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. in proceeding of the IEEE Symposium on Security and Privacy, 1998, 3-6 May, p. 86-99, Oakland, CA, USA.
8. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In Proceedings of SIGCOMM, San Diego, CA, Aug. 27-31, 2001.
9. Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proceeding IFIP/ACMInternational Conference on Distributed Systems Platforms, Heidelberg, Germany, Nov. 2001.
10. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of SIGCOMM, San Diego, CA, Aug. 27-31, 2001.

11. Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB//CSD-01-1141, University of California, Berkeley, Apr. 2000.
12. Emil Sit and Robert Morris. Security Considerations for P2P Distributed Hash Tables. IPTPS 2002.
13. Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. Symposium on Operating Systems and Implementation, OSDI'02, Boston, MA, December 2002.
14. Vivek Vishnumurthy, Sangeeth Chandrakumar and Emin Gun Sirer. KARMA: A Secure Economic Framework for P2P Resource Sharing. In Proceedings of the Workshop on the Economics of Peer-to-Peer Systems, Berkeley, California, June 2003.
15. Indrajit Ray and Indrakshi Ray. Fair Exchange in E-Commerce. ACM SIGEcomm Exchange, Vol. 3(2), Spring 2002, pp 9-17.
16. Yvo G. Desmedt, Yair Frankel. Threshold Cryptosystems. Crypto '89, LNCS 435, Springer-Verlag, Berlin 1990, 307-315.
17. Daniel Stutzbach, Reza Rejaie. Towards a Better Understanding of Churn in Peer-to-Peer Networks. Technical Report CIS-TR-04-06, University of Oregon, November 2004.
18. Mark Lillibridge, Sameh Elnikety, Andrew Birrell, Mike Burrows, and Michael Isard. A Cooperative Internet Backup Scheme. In Proceedings of the 2003 Usenix Annual Technical Conference (General Track), pp. 29-41, San Antonio, Texas, June 2003.
19. MojoNation archived website. http://web.archive.org/web/20020122164402/%20http://mojonation.com/.
20. Ronald L. Rivest and Adi Shamir. Payword and micromint: two simple micropayment schemes. In Security Protocols Workshop, 1996.
21. Steve Glassman, Mark Manasse, Martn Abadi, Paul Gauthier, and Patrick Sobalvarro. The millicent protocol for inexpensive electronic commerce. In Proceeding of WWW4, 1995.
22. Beverly Yang and Hector Garcia-Molina. PPay: Micropayments for Peer-to-Peer Systems. ACM Conference on Computer and Communications Security (CCS '03), Washington, DC, USA, October 2003.
23. Ronald L. Rivest. Electronic lottery tickets as micropayments. In Proceeding of financial cryptography, LNCS vol. 1318, Springer Verlag (1997), pp. 307-314.
24. Ranjita Bhagwan, Kiran Tati, Yu-Chung Cheng, Stefan Savage, and Geoffrey M. Voelker. TotalRecall: System Support for Automated Availability Management. ACM/USENIX NSDI, 2004.
25. Nouha Oualha and Yves Roudier. Reputation and Audits for Self-Organizing Storage. In the 1st Workshop on Security in Opportunistic and SOCial Networks (SOSOC 2008), Istanbul, Turkey, September 2008.
26. P. Brighten Godfrey, Scott Shenker, and Ion Stoica. Minimizing churn in distributed systems. ACM SIGCOMM CCR, Vol. 36, N. 4, 2006.
27. Yves Deswarte, Jean-Jacques Quisquater, and Ayda Sadane. Remote Integrity Checking. In Proceedings of Sixth Working Conference on Integrity and Internal Control in Information Systems (IICIS), 2004.