Scalability Evaluation of a Peer-to-Peer Market Place based on Micro Payments

Giancarlo Ruffo
Dip. di Informatica - Università di Torino
ruffo@di.unito.it

Rossano Schifanella
Dip. di Informatica - Università di Torino
schifane@di.unito.it

Abstract

A fair peer-to-peer market place should protect intellectual properties as well as account peers that act as distributors of the source. FairPeers *is a scheme in which some central authorities are necessary, with the drawback that when the number of transactions grows, these entities can represent single points of failure. This paper proposes a generic model to analytically evaluate such a market place and estimate its performance in terms of scalability w.r.t. the total number of printed coins and the overall transactions that can occur in the given peer-to-peer system.*

1 Introduction

There is an increasing interest in the application of Micro-Payment schemes in Peer-to-Peer systems. This is mainly due to the low cost of a generic transaction, that is the case in common file sharing applications, where the purchased item is simply a mp3 song or a divx clip. Even if early studies on accountability (e.g., [1]) argue that in such a domain, the owner's convenience is questionable because greater profits per good are generated with a flat-fee model, many researchers discuss that micro-payments can be used as incentives for users to share their own resources, reducing the well known free-riding problem [2].

Moreover, micro-payments can be used for giving *fairness* to a profit sharing environment; in fact, in a peer-to-peer system, an user can download a file from the node running in the machine of the owner of the original object (e.g., the author), or simply from another node that stores one of the copies of the file (in other words, from a *distributor*). In a fair domain protecting intellectual properties, the owner should be accounted for each copy of the file she authored in the system. Nevertheless, the distributor must be accounted as well, because she has contributed to the market giving part of her own bandwidth, cpu and storage. In a previous work [3], a micro-payment scheme enabling profit and file sharing has been presented, implementing the above concept of fairness. This scheme, which layers on PPay (proposed in [4]), and here hence called *FairPeers*, is evaluated in this paper, in terms of scalability and applicability. In fact, such schemes must be implemented in an *hybrid* peer-to-peer topology, which differs from a pure one because some central entities are allowed. These central units are the Broker (which guarantees against frauds), the Copyright Grantor (which certifies the ownerships of files), and the Certification Authority (that certifies identities). The main goals of this paper are (1) to propose a generic model that can be used to analytically evaluate such a market place, and (2) to analyse scalability of PPay and FairPeers. In particular, we want to understand how load of the central units can be limited without reducing the number of secure transactions in an heterogeneous p2p systems.

The paper is organized as it follows: in Section 2, PPay and FairPeers protocols are briefly outlined. The evaluation model used to analyse the given domain is introduced in Section 3, and the application of such model to the different parts of the scheme is presented from Section 4 to Section 6. Finally, the conclusions are given in Section 7.

2 Protocols Description

2.1 An overview of PPay

PPay is a micro-payment scheme proposed by Yang and Garcia-Molina [4] (known as PPpay) that tries to minimize the interaction with the broker allowing direct transactions between peers. To achieve this goal, the concept of *floating and self-managed currency* is adopted. The coins can float from one peer to another, and the owner of a given coin manages the currency itself, except when it is created or cashed; i.e., the user manages all the security features of the owned coin(s). As other micro-payments systems, also in PPay some coin fraud is possible, but it is unprofitable. In fact, frauds are detectable and malicious users can be punished as well. Moreover, a fraud can be operated only over small amounts of currency, and risk is higher than benefit. In the following, we briefly report the basic definition of the scheme. Refer to [4] for any further details and security considerations.

Proceedings of the 2nd Int'l Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P 2005)
0-7695-2417-6/05 $20.00 © 2005 IEEE

$\gamma = \{X, sn\}_{SK_B}$	(raw coin)
$\lambda_X = \{X, lim_l, lim_u\}_{SK_B}$	(limit certificate)
$\gamma' = \{X, sn\}_{SK_X}$	(user signed raw coin)
$\alpha_{XY} = \{Y, seq_1, \gamma\}_{SK_X}$	(assigned coin)
$\varrho_{XYZ} = \{Z, \alpha_{XY}\}_{SK_Y}$	(reassignment request)
$\alpha'_{XZ} = \{Z, seq_2, \gamma\}_{SK_X}$	(reassigned coin)
$\alpha^B_{XZ} = \{Z, seq_2, \gamma\}_{SK_B}$	(broker's reassigned coin)
$\pi_{XYZ} = \{Z, Y, seq_3, \alpha_{XY}\}_{SK_Y}$	(layered coin)

Table 1. Coins and Messages in PPay

Let X, Y and Z be three users of a p2p system, and let B be the broker. When setting up her own account, a user, say X, purchases digital coins from B. An user can buy a set of **raw coins** γ, signed by B, or a **limit certificate** λ_X that allows her to print her own raw coins γ'. Each raw coin has a serial number sn to detect double spending frauds. The serial number in a coin printed by a user, must belong to the interval defined in the corresponding limit certificate (i.e., $lim_l \leq sn \leq lim_u$).

When X wants to purchase an item or a service from Y, he will send to Y an **assigned coin** α_{XY}, that contains a sequence number of seq_1. The re-assignment of this coin will have a greater sequence number. Now Y is the owner of the coin, and he can decide to cash it or to re-assign it to another user (e.g., Z). If Y chooses the last opportunity, he has to send a *reassignment request* ϱ_{XYZ} to X. After receiving ϱ_{XYZ}, X processes it and sends to Y and Z the new assignment α'_{XZ}, where $seq_2 > seq_1$. Of course, after α_{XZ} has been released, α_{XY} is no longer valid.

If X is down when Y wishes to reassign his own coin (or she simply denies to serve), the **downtime protocol** is used instead: the broker plays the role of the trusted intermediary, and she generates the newly assigned coin α^B_{XZ} in place of X. B sends the reassigned coin to X when this peer comes back on line, because X should be responsible for detecting frauds committed when it was off-line. Downtime protocol introduces a drawback due to high percentage of off-line periods in a lifetime of a peer: Broker's load significantly grows up to reassignment requests. Moreover, Broker must continuously check when peers came back on-line, because they must send back the newly assigned coin.

Another reassignment strategy is given by **layered coins**. In this case, Y can reassign γ itself by sending to Z the assigned coin α_{XY} enveloped in a layer π_{XYZ}. If Z wishes to reassign the coin again, he has to add another layer to π_{XYZ}. Each layer represents a reassignment request and the broker and X can peel off all the layers to obtain all the necessary proofs. This protocol is still considered secure, but it has the (relatively) negative drawbacks that fraud detection is delayed, and that floating coins grow in size.

2.2 FairPeers: Copyright Management and Fair Profit Sharing

The PPay micro-payment scheme enables users to download items from each other in a peer-to-peer market place, but copyright management is a missing topic. In fact, peers are payed back if they upload a file, but original authors (when different from file owners) are not properly involved in any transactions. In [3], we presented a protocol overlaying PPay, that here we call **FairPeers**, where copyright management is focused as well as profit sharing is granted. In fact, both resource authors and owners are payed from buyers, and this turns against free-riding, because users are stimulated to let other people download their own files.

Let A be the *author* of a given copy-protected content, namely φ. When φ is inserted in the network, A is the only authorized user that can sell φ. When another peer buys item φ, then it gets also the right to re-sell it to a third peer, and so on. To pursue the rights management purpose of this proposal, as previously described, and to correctly maintain dependency between author and data source, we require the existence of a trusted third party – which we call the **Copyright Grantor** – that produces a *certificate* binding A to φ. As a consequence, another central service is introduced.

The main goal of the present study is to evaluate if these central entities (i.e., Broker, Certification Authority, and Copyright Grantor) scale well in terms of number of transactions and bandwidth usage w.r.t. the different payment strategies proposed in the protocol. As a consequence, here we summarize the main features of this protocol, in order to make understandable the evaluation model discussed in Section 3. Please refer to [3] for further details and security considerations.

2.3 The Copyright Certificate

When an author A wishes to sell a digital content he created, he contacts the *Copyright Grantor* (CG for brevity) and submits the file to the CG. In our model we assume that the CG does not bear *any* responsibility about the submitted content. More precisely the CG checks whether or not the given item was already submitted by someone else, but, in case of dispute, only the user is required to prove the originality of the item. Next, CG creates the following *copyright certificate* and sends it to A, that will store it for future use:

$$\kappa^{\varphi}_{A} = \{\varphi_1, A, LS_{\varphi}\}_{SK_{CG}} \quad (1)$$

where φ_1 contains some meta-data (e.g., genre, co-authors, session artists, production year) about φ, including a hashed (i.e. using some collision resistant hash function such as SHA-1) version of φ itself. φ_1 can be used to verify the association between the certificate and the item. Moreover, LS_{φ} is the life span of the given certificate.

Before the CG assigns the copyright certificate to the author, the latter sends his own item to the grantor. Such transaction should be non repudiable by A, because in the case of a dispute (e.g., plagiarism), CG should provide reliable information to a third party. Hence, author A begins the transaction with the Copyright Grantor sending the following message:

$$\varphi, \{A, \varphi_1\}_{SK_A}.$$

Observe that κ_A^φ defined in relation (1) is a certificate of authenticity for φ: it walks with the file, and it includes a hashed version of φ, signed by a trusted authority, that can be always verified by the receiving peer to accomplish an integrity check [1].

2.4 Purchasing an item

Let A, X and Y, respectively the author, the uploader and the downloader of φ. As a consequence, Y purchases from X, and must pay A, too. In the protocol we describe below, We assume that peers communicate using authenticated channels, e.g., an handshake phase in which peers exchange each other their public key certificate initiates the communication.

(a) $Y \rightarrow X : \{$"*get* φ", $\varphi_1\}_{SK_Y}$;
(b) $X \rightarrow A, Y : \{\varphi_1, X, Y\}_{SK_X}$;
(c) $A \rightarrow X, Y : \{A, X, Y, \kappa_A^\varphi\}_{SK_A}$;
(d) $Y \rightarrow X : c'$.
(e) $Y \rightarrow A : c''$.
(f) $X \rightarrow Y : \varphi$;

In step (a) X sends a (signed under his own private key) request for item φ. The message needs to be signed because in case of dispute, X needs to prove that he actually wanted to buy φ. In message (b) X asks author A to send his own copyright certificate to Y. Note that this is crucial for our protocol to work: having the certificate Y can prove, in case of dispute, that she was allowed by the author to sell the item. Y also receives such a request in copy, because for transparency.

If A is on line, he sends the copyright certificate to both X and Y (message (c)).

Upon receiving κ_A^φ, Y can then proceed and pay both the author A and the owner X assigning them two different coins (messages (d) and (e)). To conclude the protocol X sends φ to Y.

In the case where X coincides with A, the protocol is simplified: the identities of the author and the uploader collapse into one, message (b) is avoided, and Y assigns two coins to the A. It remains to discuss the case when A is not on line. This case is dealt similarly to the downtime protocol strategy described above. If A is not on line, i.e., handshake between Y and A cannot be completed after a given time out, the Copyright Grantor can behave in place of him in messages (b) and (c). The broker will receive the coin that Y will send in message (f).

Once A is back on line, he contacts the Copyright Grantor to get information about recent transactions. If some customer happens to have bought items he authored, then he goes to the Broker and gets the corresponding coin(s).

As a final observation, consider that such a protocol does not cope with dishonest peers that cooperate in order to cheat authors, outside the discussed scheme. This is because dealing with colluders is a very hard task: as a matter of fact there is basically *no way* to prevent users from using external (with respect to the adopted protocol) systems in order share contents without involving authors in the process (for example X could burn an audio CD with all the songs performed by A and sell it to Y at a lower price). Dealing with colluders is a major Digital Rights Management problem and is out of the scope of this proposal.

3 Modeling Transfers of Coins

Afterwards, we wish to evaluate analytically the given schemes in order to answer to the following questions: (1) How does PPay perform in relation to the different reassignment strategies? (2) Can we apply the same results to FairPeers? (3) Does the given schemes scale well w.r.t. the central entities?
Our investigation makes no assumptions on the architecture of the peer-to-peer network below and on the implemented lookup strategy. This makes our results generalizable.

The scenario depicted in this paper considers a group of peers that interact reciprocally during a time interval Δt. During Δt, peers exchange items paying them via coins printed by the broker. We suppose that, when the system starts, the peers does not have any coin, and that C coins are printed during the network's life. We also assume that, at the end of the given period, every coin is cashed by the broker. The order whereby the coins are generated does not impact the model: we can suppose that, when a peer joins the system, he buys a group of raw coins from the broker, otherwise, a peer can buy a coin only when he wants to purchase an item.

We make another assumption which is reasonable in the real world: the number of allowed coin reassignments is bounded. This limit is very important, because intuitively higher it is, more the load of the broker is reduced. Conversely, it cannot be too high, because the detection of double spending frauds are delayed. We will call m the maximum number of reassignments for a given system.

[1] Even if we are not making any assumption about the overlay network responsible for look-up, the reader should observe that φ_1 can also be used as the search key

Let us define $a_0, a_1, \ldots, a_m$ where a_i represents the number of coins reassigned i times and that have been cashed by the broker during the time interval Δt. For example, let us suppose that during Δt, 10 coins are printed. Four of them are never reassigned [2] three are reassigned twice, and other three are reassigned once. If the limit m is set to three reassignments, then we have that $a_0 = 4, a_1 = 3, a_2 = 3$, and $a_3 = 0$.

Hence, we observe that C is equal to $\sum_{i=0}^{m} a_i$.

Observing that a coin reassigned i times corresponds to $i + 1$ different transactions between peers (an assignment is due to the first transaction, and the i reassignments are consequences of the other transactions), we can derive the overall number of the transactions executed in the system, namely T:

$$T = \sum_{i=0}^{m} (i+1) a_i \qquad (2)$$

In this scenario, a significant role is played by the distribution of a_i values, that can heavily modify the results of our study. In fact, even if we know the value of C, we can easily understand that an environment characterized by coins that are never reassigned scales very differently w.r.t. another network wherein the majority of coins are reassigned m times. Unfortunately, we do not have any idea how users will behave in such a market place, because no one has experimented such technologies in the real world. This has the consequence that neither the weight distribution can be established in an unique way, nor any empirical measure based on monitored peer-to-peer traffic can be used. We think that it would be wrong using measures achieved in the present p2p networks, because past analysis (e.g. [2] [5]) were conducted in domains where users download files for free without gaining any profit. We strongly believe that the main feature of the *FairPeers* network can heavily change behavioral models, e.g., reducing the Free-Riding phenomenon. For such reason, we decided to evaluate the entire system making several hypotheses, and comparing reassignment strategies under these different settings.

In particular, in our analysis, we focused on the following cases:

- `Best case:` the best expectation, in terms of performance, is that the reassignments chain *always* achieves the highest length, i.e., each coin is cashed only when has been reassigned m times. Formally, we have $a_0, a_1, \ldots, a_{m-1} = 0$ and $a_m = C$.

- `Worst case:` the worst case occurs when a raw coin is assigned once, but never reassigned. In such a case, we obtain $a_0 = C$, and $a_1 = a_2 = \ldots = a_m = 0$. In terms of broker's load this case collapses in a client-server market place, wherein the broker is involved in every transaction.

- `Uniform case:` assuming an uniform distribution of a_i values, we obtain a system characterized by the relation $a_0 = a_1 = \ldots = a_m$, whereby we obtain $a_i = \frac{C}{m+1}$.

- `Pareto case:` in this case the coefficients follow a Pareto distribution in the form $P(X) = \frac{a}{X^{1+a}}$, where a is set to 2 in our experiments.

- `Zipf case:` the distribution selected is the Zipf distribution $P(X) \cong \frac{1}{X^a}$ where $a = 0.8$.

[2] i.e., These four coins were printed by the broker for some given peers, which assigned the coins to other peers. The receiving peers cashed the coins rather than reassigning them.

Name	Value	Description
$\|seq\|$	4 bytes	Sequence number
$\|sn\|$	4 bytes	Serial number
$\|id\|$	2 bytes	Peer's identity
$\|sign\|$	128 bytes	Signature
$\|raw\|$	$\|id\| + \|sn\| + \|sign\|$	Raw coin
$\|assigned\|$	$\|id\| + \|seq\| + \|raw\| + \|sign\|$	Assigned coin
$\|limit\|$	$\|id\| + 2 \cdot \|sn\| + \|sign\|$	Limit Certificate
$\|check\|$	1	Signature verification
$\|gen\|$	40	Signature generation
t	0.8	Off-line peer's rate
T	500	Transactions

Table 2. Cost of atomic actions and modeling parameters

In a real domain, it is likely probable that an high number of coins will be cashed after few reassignments. For this reason, we introduced Zipf and Pareto distributions among our hypotheses.

Furthermore, Table 2 lists the set of system parameters and the cost of atomic actions used in our investigation. At first, we will focus on the space (in bytes) occupied by the basic components of the protocol. Then, we focus on the computational cost of digital signature generation and verification, assigning to a generation operation a weight 40 times higher than a verification [6]. Finally, we introduce the **total number of transactions** T accomplished during the given time interval and the **downtime rate** t of peers, i.e., a peer will be off line with probability t. In this model the number of the users do not affect the final results, because we focus on comparing how different coins assignment strategies load the central logical units, independently from the number of transactions. Of course, the off-line peer rate can be variable. Future work is scheduled to analyse several scenarios in function of t.

4 Broker's load analysis

In this section, we make an evaluation of the broker's load, exploring how the reassignment strategies affect the

performance and, consequently, the scalability and applicability of the entire system. We define the set of activities in which the broker is involved and, for each of them, we estimate the broker's load w.r.t. (1) the number of executed cryptography operations and (2) the amount of exchanged bytes during the following interactions:

Printing: the broker mints a new coin for an user X. The raw coin is digitally signed by the broker.

Reassignment: when an user X assigns a raw coin to another user Y, the broker in not involved in any way. On the contrary, the reassignment of the coin from Y to another peer Z can engage the broker accordingly to the reassignment strategy implemented in the system.

Cashing: a coin floats following the purchase chain until a peer decides or is forced (e.g. when the maximum number of layers is achieved) to cash it. Then, the broker executes the given signature checks and credit the user account if no fraud is detected.

Due to these considerations, we can define the broker's load ℓ_B based on the three components described above:

$$\ell_B = C * \omega_P + (T - C) * \omega_R + C * \omega_C \qquad (3)$$

where $(T - C)$ is the number of reassignments performed during the time interval and ω_P, ω_R, ω_C are functions that, respectively, returns the weight due to printing, reassigning or cashing a coin.

Coins can be reassigned by way of three different strategies, namely Basic, Layer and Hybrid.

Basic: each reassignment involves the owner of the coin according to the scheme based on messages ρ and α' defined in Table 1. If the owner is down, the broker receives the reassignment request from the buyer and sends the reassigned coin α^B to the engaged peers. He has also the charge to contact the owner when he comes back again on-line.

Layer: when a peer wants to reassign a coin, he adds a layer without contacting neither the owner nor the broker. The coin floats with an extra layer for each re-assignment until the limit m on the number of layers is reached or until a peer decides to cash it.

Hybrid: this strategy is a trade-off between those described above. At first, a peer try to reassign the coin by way of the owner, but, if the latter is down, the coin is layered instead.

In the remaining of this section, we will depict in more details the weight functions according to the given reassignment strategies, and performing a comparison analysis in terms of the given cost parameters.

4.1 Computational and spatial analysis

The evaluation has been made following two different perspectives, the former related to the execution of cryptographic primitives, and the second in function of the amount of bytes exchanged during an interaction with the broker.

Strategy	ω_P	ω_R	ω_C
Basic	$\lvert gen\rvert$	$t \cdot (3\lvert check\rvert + \lvert gen\rvert)$	$2\lvert check\rvert$
Layer	$\lvert gen\rvert$	0	$(2+i)\lvert check\rvert$
Hybrid	$\lvert gen\rvert$	0	$(2+t\cdot i)\lvert check\rvert$

Table 3. Weight functions in term of digital signature generations and verifications

Strategy	ω_P	ω_R
Basic	$\lvert raw\rvert$	$t \cdot (\lvert id\rvert + \lvert sign\rvert + 3\lvert assigned\rvert)$
Layer	$\lvert raw\rvert$	0
Hybrid	$\lvert raw\rvert$	0
	ω_C	
Basic	$\lvert assigned\rvert$	
Layer	$\lvert assigned\rvert + i \cdot (2\lvert id\rvert + \lvert seq\rvert + \lvert sign\rvert)$	
Hybrid	$\lvert assigned\rvert + t \cdot i \cdot (2\lvert id\rvert + \lvert seq\rvert + \lvert sign\rvert)$	

Table 4. Weight functions in term of exchanged bytes

Table 3 and Table 4 list, respectively, the value of the weight functions according to these complementary views.

Let us notice that the cost of the printing phase is constant in both the perspectives, regardless of the reassignment strategy chosen. On the contrary, the ω_R function shows a quite different behavior: in the `Basic` scheme a reassignment statement engages the broker just when the owner is down; this case is taken into account with probability t, which is defined in Table 2. In fact, value t is multiplied by the cost of an instance of the downtime protocol. Otherwise, the `Layer` and the `Hybrid` strategies do not involve the broker in the reassignment phase, then, the cost is clearly null.

Now, let us consider the cashing weight ω_C: in the `Basic` strategy, we have a fixed cost due to the fact that the coins that the broker manages have a constant dimension and structure, i.e. the format of the assigned coin α, as defined in Table 1. This situation is quite different w.r.t. the other strategies wherein cashing costs depend on i, namely the number of layers of a coin. Finally, notice that the `Hybrid` scheme is based both on the number of layers and on the parameter t that models if the reassignment is made by means of the owner or by layering the coin.

4.2 Results

Now, we present and discuss the results of our evaluation. Top of Figure 1 describes the environment in function of the number of digital signature operations performed by the broker. Bottom of Figure 1, instead, shows the bytes exchanged during the interactions with the broker. Each figure shows two important aspects: (1) in what way the distribution of a_i coefficients modifies the behavior of the broker in a fixed system configuration and (2) how the different strategies can affect the broker's load. Notice that the `Hybrid` strategy does not appear in the figures because, in any case,

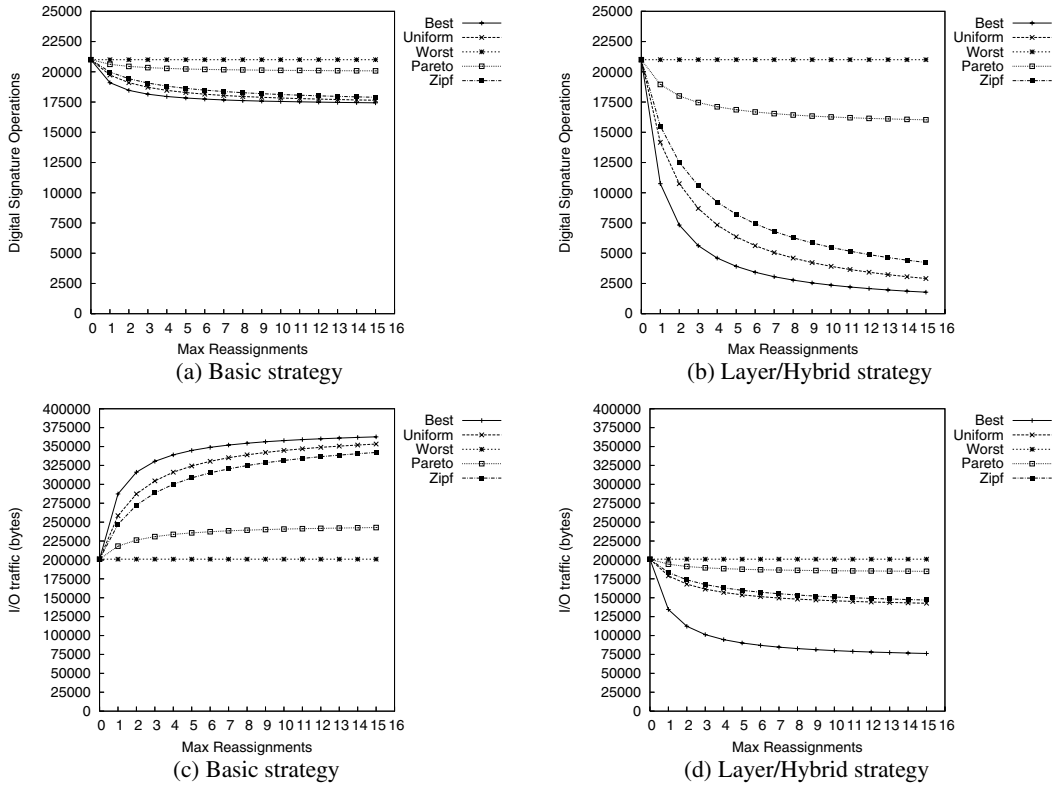

Figure 1. Broker's load analysis: computational and spatial perspective.

it underlies the `Layer` scheme not more than 3%.

First of all, it is evident that the `Layer/Hybrid` strategies outperforms the `Basic` strategy whatever distribution of a_i coefficients is selected, excluding the `Worst case` which is trivially not affected by the reassignment scheme.

Except for the `Best` and the `Worst` cases, `Pareto` underlies both `Zipf` and `Uniform` cases. This is trivially due to the characteristics of the given distributions: the Pareto distribution decreases very rapidly and this has implications in the high number of coins with few reassignments, and in the few coins reassigned many times. This degradation is mitigated if a Zipf distribution is used instead, and it is flatten by an uniform distribution. A gradual performance decay is expected in environments where few reassignments take place, that are inclined to the `Worst` case.

Figure 1(a) shows an apparently surprising result. The `Worst` case gives the best performances when the `Basic` strategy is adopted and the curves are turned upside down w.r.t. the previous graphs. The reason of this phenomenon is the cost of a single instance of the downtime protocol: the more coins are reassigned, the higher the broker is contacted and overloaded. Since the weight given to the downtime rate t is quite high, reassignments in this strategy are counterproductive for the broker.

Increasing the number of reassignments does not always carry a clear performance improvement. In fact, in the `Layer/Hybrid` strategies and for both perspectives (in Figures 1(b) and 1(d)), we notice a flat tail, that suggests that we have not any further benefit when the number of layers increases indefinitely. For instance, focusing on the computational analysis, we have a cost reduction of the 68% passing from $m = 0$ to $m = 7$, but only of the 30% if m is changed from 8 to 15. To the other side, if m is too high, fraud detections are delayed, and the level of security decreases.

We can conclude that performances are improved with the adoption of `Layer/Hybrid` strategies and that more are the layers, higher is the overall improvement. Even if it is reasonable to fix a maximum number of layers not greater than 7, also for security reasons, a real system should incentive users to let circulate the coins as much as possible, in order to avoid `Pareto`-like behaviors. For instance, such incentives can take the form of a service fee discount only if the coin is cashed when no other reassignments are possible.

5 Limit certificate impact

The *Limit Certificates* represent an attempt to reduce the broker's load by way of allowing a peer to print a coin by

Proceedings of the 2nd Int'l Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P 2005)
0-7695-2417-6/05 $20.00 © 2005 IEEE

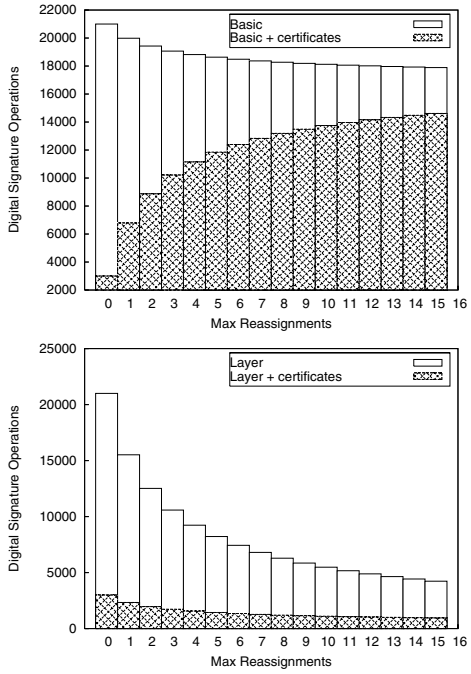

Figure 2. Limit Certificates impact

himself. In this section we will try to answer to the following questions: (1) How does the limit certificates affect the broker's and the peers' performance? (2) Are the considerations outlined in Section 4 still valid?

From the point of view of the broker, a strategy based on limit certificates influences only the printing phase. Let us suppose that a limit certificate authorizes a peer to mint himself n coins, furthermore, we hypothesize that the number of coins cashed by the broker is still C, such as defined in Section 3. Therefore, we can depict the broker's load by way of this relation:

$$\ell_B = L * \omega_P + (T - C) * \omega_R + C * \omega_C \qquad (4)$$

where $L = \frac{C}{n}$ and ω_P has the "computational" value equal to $|gen|$ (because the generation of a limit certificate asks for a broker's signature) and the "spatial" value equal to $|limit|$ (see Table 2 for definitions). Notice that, because a limit certificate influences only the printing component, we can select whatever reassignment strategies we prefer. As a consequence, ω_P and ω_R have the same values of the previous analysis. Applying the same hypotheses described in Section 3, we estimate the broker load in a environment characterized by a Zipf distribution of a_i coefficients (other cases are not included for brevity). Figure 2 shows the impact of the adoption of limit certificates in relation to the different reassignment strategies. As expected, the broker load is heavily reduced, in terms of computational and bandwidth usage costs.

Furthermore, we observe that, in both spatial and computational perspectives, using limit certificates let the load of the broker decrease at exactly the same rate of the corresponding reassignment strategies.

6 Analysis of FairPeers

For simplicity, let us define two different types of transactions (see Section 2.4):

(1) *Buyer-Author* wherein an user, namely a *Buyer*, purchases an item directly from the *Author*

(2) *Buyer-Merchant* wherein the *Buyer* purchases the service from another user, the *Merchant*, that purchased it from someone else.

In the first case, both the coins are received by the *Author*, on the contrary, in the second scheme the *Author* and the *Merchant* are distinct entities. Furthermore, we can say that the first type of transaction is likley to happen with rate μ and, the second with probability $(1 - \mu)$.

In this section, we describe how the central entities involved in the FairPeers protocol are affected in terms of computational and traffic load.

6.1 Broker

In such a context, a *Buyer* spends two different coins to buy the service. Now, if we assume that in a time interval Δt, T transactions occur within the system (see Table 2), we can observe that w.r.t. to the previous analysis, the amount of coins is doubled. We also observe, that the coins management policies and the reassignment strategies are not affected by the implementation of the FairPeers scheme. As a consequence, we can generalize all the scalability considerations we presented in Sections 4 amd 5.

The broker is also involved in a *Buyer-Merchant* transaction, namely when the author is off line. In such a case, as described in Section 2.2, the broker receives the coin c'' from the *Buyer* and he acts on behalf of the `Author`, sending him the coin as soon as he comes back on line. The probability that the broker is involved in a such activity can be determined by the expression $t \cdot (1 - \mu) \cdot T$. In any case, it is clear that this activity is not very expensive: the broker must simply store c'' and send back it to the Author as soon as possible[3].

In a real environment, the `Author` can be taxed when the broker guarantees for hime. In this case, the `Author` will be stimulated to stay on line as longer as possible in order to safeguard her own profits. Thus, the downtime rate t could be lower than the common off line times measured in existing peer-to-peer networks.

[3]The broker can carry out a strict policy by means of checking the coin c'' or he can delay the fraud detection step when the coin has been cashed.

	ω_{CG}
Computational	$\|check\| + \|gen\|$
Spatial	$\|\varphi_1\| + 2\,\|\kappa_A^{\varphi}\| + 8\,\|id\| + 3\,\|sign\|$

Table 5. Copyright Grantor weight function

6.2 Copyright Grantor

The first kind of interaction that involves the CG, concerns the generation of a copyright certificate: when an author wishes to sell a digital content, he contacts the CG that sends back him a certificate that represents a proof of authenticity of the item. The certificates generation is performed just once for each digital content introduced in the market place. Like all the certificates, the copyright certificate contains a lifespan that delimits its validity. Again, the CG should manage the renew process: it is evident that such a task is not very expensive due to the low frequency.

The CG is involved also in a *Buyer-Merchant* transaction, namely when the `Author` is unreachable. He behaves in place of the author in steps (b) and (c) of the scheme described in Section 2.4. The CG behaves similarly to the broker during an instance of the *downtime protocol*.

Formally speaking, we can define the CG load ℓ_{CG}:

$$\ell_{CG} = t \cdot (1 - \mu) \cdot T \cdot \omega_{CG} \tag{5}$$

where ω_{CG} is defined in Table 5. Notice that are valid the same considerations about the *Author* described in the previous section.

We can observe that the CG is loaded less than the broker. In fact, the generation and the renew of a certificate is not an expensive activity, especially if compared with the printing phase in a PPay environment with limit certificates. Moreover, the cost of the downtime scheme which the CG is involved in, is similar to the downtime protocol in PPay. Finally, the CG does not manage the cashing of a coin and the detection of a fraud, that are resource-consuming activities.

6.3 Certification Authority

The Certification Authority is a central service that certifies users' and peers' identities, as well as in a common distributed PKI. Certificates are generated when a node joins the network. After a validity period, the certificates expire and must be renewed. A certificate binds an identity with a public key, and it is signed with the CA's private key. Of course, the CA must be a trusted third party, and each peer in the network must be able to verify a certificate's validity, namely the CA's signature. If each peer is responsible to store his own certificate and to give it to other peers upon request, the CA should be not contacted in any other cases. It is trivial that the Certification Authority is linearly scalable with the number of peers in a network, and its own load will be always very limited.

7 Conclusion

In [4], PPay has been presented providing a micro payment scheme fully integrable in a peer-to-peer system. The FairPeers market place, which overlays PPay, originally presented in [3], completes the PPay environment giving a setting for a fair profit sharing between peers, where the transactions are made without violating digital rights. This is done by way of the introduction of three central units, that we proved to be highly scalable when the coin's reassignment strategy is selected as well. Observe that in [4], and limited to the case of pure PPay, a performance evaluation analysis has been presented also. This analysis framework was tight related to a below peer-to-peer architecture, namely GUESS, and parameters used in such a study strongly biased the conclusions. The present analysis is based on different hypotheses, instead, and no specific architecture assumption is made. The conclusions are somehow more general and underline limits and strength of layered coins and reassignment policies.

Acknowledgement

This work has been supported by the Italian Minister of Research (MIUR), whithin the project *WebMinds* (FIRB).

References

[1] M. J. Freedman R. Dingledine and D.Molnar. *Peer-To-Peer: Harnessing the Power of Disruptive Technologies, Chapter 16*. O'Reilly, 2001.

[2] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, September 2000.

[3] D. Catalano and G. Ruffo. A fair micro-payment scheme for profit sharing in a p2p network. In *Proc. of HOT-P2P 04*. IEEE Press, October 2004.

[4] B. Yang and H. Garcia-Molina. Ppay: micropayments for peer-to-peer systems. In *Proc. of the 10th ACM CCS*. ACM Press, 2003.

[5] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. of SOSP '03*. ACM Press, 2003.

[6] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.