Darmstadt University of Technology

A Token-based Accounting Scheme for P2P-Systems

Nicolas C. Liebau, Vasilios Darlagiannis, Andreas Mauthe, Ralf Steinmetz

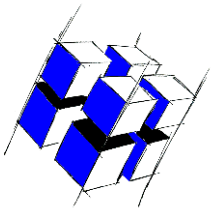
Multimedia Communications Lab KOM, Darmstadt University of Technology, Germany
E-Mail: [Nicolas.Liebau|Vasilios.Darlagiannis|Andreas.Mauthe|
Ralf.Steinmetz]@kom.tu-darmstadt.de

KOM Technical Report 05/2004
Version 1.0
Juli 2004

Multimedia Communications (KOM)

Department of Electrical Engineering & Information Technology
Merckstraße 25 • D-64283 Darmstadt • Germany

Phone: +49 6151 166150
Fax: +49 6151 166152
Email: info@KOM.tu-darmstadt.de
URL: http://www.kom.e-technik.tu-darmstadt.de/



Abstract

This paper presents a token-based accounting scheme for decentralized autonomous systems, such as peer-to-peer systems. The scheme uses tokens as proof of resource usage or service usage. Conforming to the peer-to-peer paradigm, the tokens are issued using a decentralized mechanism. Within peer-to-peer the proposed accounting scheme can be used to cope with the shortage of information. Therewith, it constitutes the basis for coordination and control mechanisms as well as pricing for commercial scenarios in completely decentralized systems.

The presented scheme is compared against an alterative approach showing the advantage of the token-based mechanism in terms of communication costs.

1 Introduction

The design of the first peer-to-peer (P2P) systems was based on the assumption that participating peers share their own resources with other peers while they benefit from resources that are shared by others. Through resource replication and utilization of otherwise unused resources, P2P systems can provide higher robustness and host more content/information at lower costs than traditional client/server-based applications. Actual P2P file sharing systems like eDonkey2000 [11] or KaZaA [17] host huge amount of content in a reliable way. However, the time needed to retrieve a piece of content from such a system is in most cases considerably higher than in traditional client/server-systems. Users of P2P file sharing applications are accepting this performance constraint because they are retrieving content at virtually no cost. One reason for this reduced performance of P2P systems in comparison to client/server-systems is in the opportunistic behavior of the participants who try to maximize there own utilization. Participants try to benefit as much as possible from the resources provided by the other members of the system; however, they try to avoid providing resources themselves. The most familiar example is the free-riding phenomenon in P2P systems [1]. This behavior pattern is fostered through the strong anonymity and the enormous lack of information in P2P systems. Actions cannot be traced back to users. Therefore, resource or service usage and provisioning is not attributable to users. Thus, it is hard to give incentives for resource provisioning, or, as a further step, to implement and enforce rules about participant behavior in the system. The result is the aforementioned weaker performance of P2P systems in comparison to a client/server alternative. The weaker performance also makes P2P systems unattractive for commercial applications.

To overcome this disadvantage the lack of available information must be resolved. An accounting mechanism for P2P systems is required to deliver the missing information. Using this information coordination of the available resources can increase the performance of a system significantly. Coordination can be achieved e.g. through the introduction of rules and rule enforcement supported by the information an accounting mechanism has been collecting. However, the design of such mechanisms for decentralized autonomous systems is not trivial because the control mechanism cannot be decoupled completely from the accounting mechanism. The absence of a controller that analyzes the gathered information and coordinates the system entities requires that the accounting mechanism includes the coordination functionality. Thus, the accounting mechanism must enable the ability to constraint the participants' behavior.

For a distributed accounting system that also enables coordination, it is required that the collected accounting data is held in a robust und secure way so that no important information is lost. Further, the accounting information must be collected and held in a trustworthy manner. If the information is used for system coordination participants my be tempted to modify information for their own benefit. Moreover, the accounting mechanism should be scalable and the net benefit of using it should be positive across the complete system. If the accounting mechanism should be used in different scenarios it should be flexible to support different kind of coordination mechanisms.

To tackle the discussed problem this paper proposes a token-based accounting mechanism. Tokens serve as signed receipts for transactions between peers. Further, tokens represent the transaction history of peers and allow for monitoring and control of the account balance of all participants in a system by means of appropriate aggregation mechanisms.

The remainder of this paper is organized as follows. Section 2 outlines the alternative approaches for accounting mechanism in P2P systems and shows the related work. Section 3 describes the proposed token-based accounting mechanism. Section 4 focusses on security aspects of the mechanism and section 5 gives details about the current implementation. In section 6 the concept is validated against remote account-based accounting mechanisms. In section 7 the conclusions are presented.

2 Related Work

There are several design alternatives for distributed accounting systems. Essentially accounting data is collected in form of receipts. The information stored in a receipt can vary from a single number to detailed transaction data. For the purpose of coordination for every peer the data stored in its receipts is aggregated to an account balance. The balance determines if a peer is allowed to use further resources from the system or if it first has to provide more own resources. However,

major characteristic that distinguishs different accounting schemes is the location of the stored receipts.

Local Accounts. Using local accounts, one receipt is generated for each transaction and participating peer. Receipts are stored locally on the peers. To enhance the trustworthiness of receipts they can be signed by the transaction partner. P2P accounting systems using local accounts scale well because there is no communication with further parties. Today local accounts are used e.g. in eMule's credit system [12] to determine other peers' position in the local download queue. This mechanism tries to achieve local fairness; global coordination is not its goal. With local accounts all information about a peer is derived directly from the peer. Even if the receipts are signed by the transaction partner, fraud is easily possible through collaboration.

Remote Accounts. This alternative tries to overcome the trust problem of local accounts by storing accounting information at third party peers. Each account is located at set of "account peers" to achieve robustness. The "account peers" are usually organized in a Distributed Hash Table (DHT) for efficiency reasons. In [3, 10, 27] this approach is applied. In [3] issued tokens are used as a kind of a virtual currency, which is transferred between the remote accounts during a transaction. For trust reasons receipts can be signed either by transaction partners or (ideally) by multiple trustworthy peers. The trust level in such a system is high. This is achieved through additional network traffic per transaction for querying accounts, signing receipts, storing receipts and keeping the accounts consistent.

Central Accounts. This alternative uses a central network administrator to collect receipts and to distribute the usage of network resources among the participants in a fair way. For Grid Computing, for instance such a system is presented in [5]. However, our goal is to avoid central elements in P2P systems.

Token-based Systems. An alternative to using receipts is to use tokens. Tokens are issued receipts. Peers spend tokens with other peers to receive a service. If a peer runs out of tokens the peer is not eligible for using more system resources. The tokens must be protected against forging and double spending. Storing tokens is not different from normal receipts. Often tokens are used as a virtual currency. Doing so**,** the trust problem of local storage is bypassed, because these tokens do not contain any accounting information that might be altered. Token-based systems require that the token issuer is trustworthy. There are three alternatives for the token issuer: (a) Each single peer can issue tokens. This way the trust problem is bypassed. However, introducing rules and rule enforcement become impossible because there is no control on the amount of tokens issued. Such an approach is shown in [19]. The authors claim that eventually a completely free stable market will develope. Further, in [26] self-issued tokens are used for accounting in grid computing. In [22] two systems based on so called „Proof Of Works“ (POWs) are presented. (b) A central, trusted "bank" issues the tokens. Mojo Nation uses this solution as well as some existing micro payment schemes like eCash [24] or NetCash [18]. A micro payment scheme especially tailored to P2P systems is presented in [28]. The goal of this work is to reduce the load on the central broker. However, the use of a central entity is contrary to our goal of designing a decentralized P2P system. (c) A quorum of peers signs the tokens using a shared private key. If the private key is kept secret such a system combines scalability and trustworthiness. This solution is used in the presented approach.

3 The Token-Based Accounting System

Prerequisites. The token-based accounting system assumes that users can clearly be identified through a permanent id, (e.g. through a private/public key pair proven through a certificate issued from a certification authority). Depending on the application scenario, alternative approaches like [7] are also applicable. Apart from the certification authority it is intended to avoid any central element.

Also we assume that the majority of peers in the system do not act malicious.

Further, we assume the use of a reputation mechanism in the P2P-system. This system is used to publish fraudulent behavior that technical mechanisms cannot detect. The reputation mechanism assigns a reputation value to each peer that represents the trustworthiness of the peer. A possible solution is e.g. presented in [16].

3.1 Overview.

The primary goal of the proposed system is to collect accounting data and to enable system-wide coordination of resource service usage on basis of the collected information. To enable the usage of receipts for coordination in a distributed system, the receipts must have the basic characteristic of the resources and services they represent, viz they must be scarce. Therefore, the receipts must be issued. Accordingly, every user has a limited amount of receipts it can use in transactions. Thus, in the presented approach tokens are used rather as issued receipts than as a virtual currency. As a result, the tokens must not have the characteristics of micro payments of anonymity and untraceability [6]. Therefore, tokens have a clear owner that is contained in the token. This enables storing the tokens locally. Otherwise, if anonymity should be maintained, untraceable tokens have to be stored at trusted remote accounts to control double spending.

Each peer holds an account with a specific amount of tokens clearly issued to it. A peer spends a token by sending it to its transaction partner in order to receive a service. Accordingly, when a peer provides a service it collects foreign tokens from other peers. Peers cannot spend foreign tokens. Using the token aggregation pro-

cess, peers exchange the collected foreign tokens against new ones. To achieve trustworthiness new tokens are signed with the system's shared private key using threshold cryptography [8]. Thus, a token must be signed by a quorum of peers to become valid. The *token structure* ensures protection against forgery, double spending and being robbed. The three basic protocols of the token-based accounting system are T*oken Aggregation*, *Check for Double Spending*, and *Payment*.

3.2 Token structure.

Figure 1 shows the information contained in a token. A new unused token contains the first 5 information fields starting from the top of the figure. The issuing date and time in milliseconds together with the serial number and the owner id serve as unique identification of a token. This is required to enable the detection of double spending. Further, this way double spending can be traced to the owner. During the creation of a batch of new tokens the serial number is randomly selected for every token. Thereby, guessing which tokens exist in the system becomes hard. The account id is used to allocate a token clearly to a specific application. Cross application usage and trade of tokens is possible. This field is optional. The fifth field contains the signature of the information contained in the first four fields, signed with the system's private key. This prevents forgery.

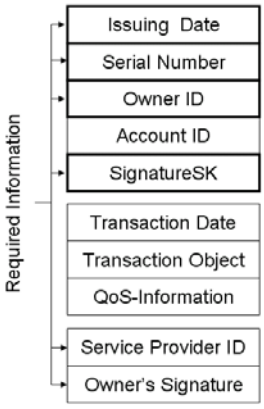


Figure 1:Token Structure

Since a token is basically a receipt, it contains further information about the transaction for which a token is used. The service consumer is the token owner.

Before the owner sends the token to the service provider, he also adds the service provider's id to the token as well as information about the transaction (such as transaction object, date and information about the quality of the service provisioning). The owner finally signs the complete token using his private key. Subsequently the contained information cannot be changed by the service provider. The required information in a token is the information needed for unique identification, the system signature, the service provider as well as the service provider's signature. This prevents tokens from being roobed. Because unused tokens contain the owner, only the owner can spend them. Used tokens are signed and contain the receiver of the token. Only the receiver is allowed to exchange tokens against new, own tokens.

3.3 Token Aggregation

The Token Aggregation process is used to exchange foreign tokens a peer collected against new tokens issued to that peer. The eight-step Token Aggregation procedure is shown in Figure 2.

First the exchanging peer (EP) locates a "trusted peer" (TP) (1). Trusted peers are eligible to exchange tokens and possess one part of the system's private key [8]. EP sends its N collected foreign tokens (Fn_1, ..., Fn_N) to TP (2). TP checks the foreign tokens for their validity. Only tokens signed by the owner and spent only once are valid for exchange.

Using the aggregation function $M = A(Fn_1, ..., Fn_N)$ TP calculates the amount M of new tokens EP must receive in return for the foreign tokens . The aggregation function is public and can take any form. TP now creates M new, unsigned tokens (Un_1, ..., Un_M) (3).

To sign the new tokens with the system's private key using threshold cryptography [8] TP now locates further trusted peers (4). The number of required trusted peers to sign a token is determined by the used secret sharing scheme. The larger the quorum of trusted peers, the more trustworthy is the system. EP is not allowed to choose the quorum of trusted peers itself. This avoids potential collaboration and fraud between the peers.

TP sends the new tokens to this quorum of trusted peers (5). Each peer of the quorum signs now the tokens with its part of the system's private key (6). The resulting partial tokens (Pn_1, ..., Pn_M) are transmitted back to EP (7) . Finally, EP combines the partial tokens to new complete tokens (Tn_1, ..., Tn_M) (8).

The aggregation function is a core piece of the token-based accounting mechanism. It adds an additional degree of freedom because the token exchange rate can be different than 1:1. The aggregation function can be used to implement different usage policies and different economic systems. For instance, the user's reputation value can be taken into account to reward trustworthy behavior or asymmetric DSL-links can be modeled so that 1 MB upload worths more than 1 MB download.

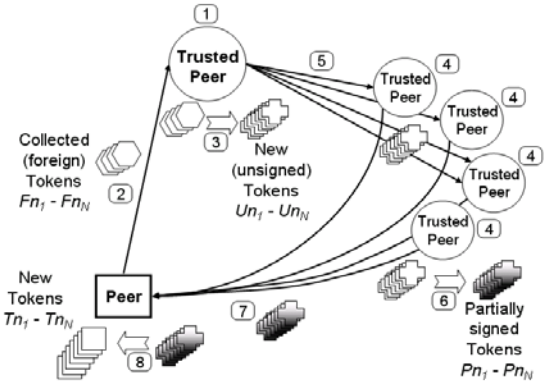


Figure 2:Token Aggregation Process

3.4 Check for Double Spending

To check for double spending a token must be clearly identifiable. To facilitate the check in an efficient manner. For every peer there exists a set of account peers, the account holders. They are organized in a DHT manner, such as Pastry [23] (see Figure 3).

Account holders hold a list containing the tokens currently issued to the peer for which they hold the account. The list is filled with the required information during token aggregation. After the new tokens have been created (Figure 2, step 3), the trusted peer sends a list of the new tokens to the exchanging peer's account holders (Figure 3, step 3).

During the token validity check of the token aggregation process, the trusted peer will ask the account holders responsible for a token, if the token is valid (Figure 3, step 2). The account holders will remove the token from the list. Accordingly, if the token is not in the list, it is an invalid token. Either it was spent before or it was forged. The account holders will inform the trusted peer, which will then discard this token. Also, the reputation mechanism of the P2P system will be informed about the incident.

In order to avoid attacks on messages to account holders, every message sent to the account holders must be signed with the sender's private key. To keep the list between the account holders consistent**,** all account holders for one specific account exchange the list whenever the set of account holders change. This is at joins and leaves of peers of that set. Update is only necessary, if the sender does not receive all confirmation messages.

3.5 Transactions

At transactions the token-based accounting mechanism can be used for resource usage, service usage, or a combination of them. Resources in a P2P system are bandwidth, storage capacity, CPU power, and content [9]. A service is valued differently than resource usage. A service for example detects water marks in pictures. Since special software is needed to provide such a service, it is valued higher than the sum of the used resources. A token can contain information about the used resources and value information of the service itself. The information is added to a token before it is sent to the service provider. By this means information contained in a token can be used as basis for an external payment mechanism.

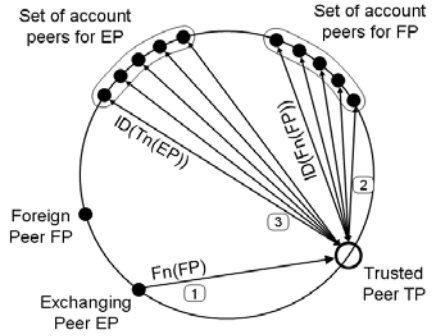


Figure 3:Check for Double Spending

Standard transaction. The standard transaction process is shown in Figure 4. After a service has been requested by the service consumer C, the service provider P informs C about the terms and conditions of the service, including the number of tokens P expects in return for the service. If C accepts the terms and conditions, the service provisioning phase begins.

During this phase tokens can be transmitted before, after, or during the service provisioning. For example a token can be transmitted after 1 MB transferred or after 1 minute service received. Before a token is transmitted, C retrieves the token from its local account and fills in the required accounting information. C has no intention to fraud on the information, because it influences only the token exchange of the P. Then C signs the token with his own private key and sends it to P. P checks the signature of the received token using C's public key, which can be contained in the token as owner id or transmitted with the service request. Thus, it can be verified, that the token sender is also the token owner.

P can choose not to continue to provide the service, if the contained accounting data was incorrect. As a result of a transaction C's amount of own tokens decreases and P's amount of foreign token increases.

Trustable transaction. In a scenario where tokens are used as virtual currency, a more trustworthy settlement process might be required. Here, the transaction party that delivers last has an incentive to cheat on the other party. It still receives the full benefit but does not have to deliver its part of the deal. Therefore, we have designed and implemented a trustable payment procedure that eliminates the incentive to cheat for the transaction partners. In addition, double spending of tokens is not only detectable, but becomes impossible. Figure 5 shows the procedure.

After a service request is received**,** P notifies C about the conditions and terms of the transaction, including the required amount of tokens. C answers with the token ids of the tokens it intends to spend with the transaction. Now P contacts the account holders responsible for C „AH(C)" and checks if the tokens are valid. AH(C) mark in the token list these tokens as "planned to spend". Using the same tokens in another

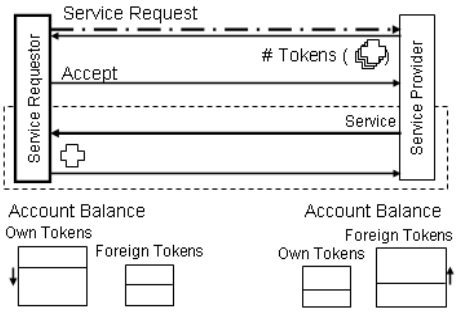


Figure 4:Transaction

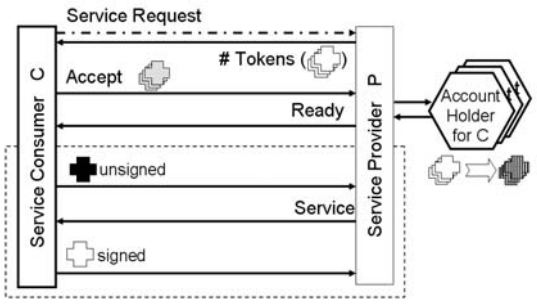


Figure 5:Trustable Transaction Procedure

transaction becomes impossible. If all tokens are valid, P informs C that the transaction phase can begin. C starts the transaction by sending an unsigned token to P. C loses the token. However, since it is not signed by C, P cannot exchange it against own tokens. P has no incentive not to provide the service. Therefore, P now provides the agreed service. Because C already lost the token, it has no intention keeping the token for itself. C will sign the token and send it to P.

If C should fail to send the signed token, P can present the unsigned token to AH(C). The possession of the token proofs that the transaction had started and the token will be removed from the list and is finally lost for C. The aforementioned reputation system will provide further incentives against malicious behavior.

On the other hand, if both peers are consenting to cancel the transaction, C does not lose its tokens. The "planned to"-mark just needs to be removed from the tokens in the token list at AH(C).

4 Trust & Security Considerations

It is crucial for the use of an accounting mechanism that the information it provides is correct. Therefore, the token-based mechanism has been designed to provide a high degree of trust for distributed systems.

Robbery. The number of tokens available to a peer influences its ability to benefit from the system. Therefore, tokens were designed to eliminate robbery. Tokens contain the owner id that cannot be changed without detection through the system signature. Spent tokens contain the token receiver secured through the owner's signature.

Forgery. The system signature on each token ensures that the basic token data cannot be changed and that no peer can create tokens by itself. Thus, the system signature prevents forgery and is crucial for the trustworthiness of the system. Accordingly, fraudulent collaboration of trusted peers must be avoided. Therefore, the choice of trusted peers to form the quorum for signing tokens is not done by the exchanging peer. This alleviates the problem of bribing. Further, peers can only become trusted and receive a part of the shared system private key, if their reputation is above a specific threshold value. The actual threshold value depends on the used reputation system. Further, threshold cryptography provides different proactive mechanisms to secure the key from being compromised. The key parts will be updated periodically using proactive secret sharing [21]. This makes the old key parts obsolete without changing the actual key. The system's public key stays the same. Further, a new system will be created periodically using the decentralized method presented in [4]. This is enforced through tokens being valid only for a specific period of time. Therefore, the unique token id contains the creation date and time. Outdated tokens can be exchanged against new tokens using the token aggregation process. If the system's private key is kept secret the system can be considered secure.

Double Spending. The verification for double spending relies on the data hold at the account holders. Thus, users might try to corrupt their token list at the account holders. This is avoided by not allowing peers to send any queries or enquiries to the account list. Further, the token list at the account holders is a positive list. If a peer plans to double spend a token, it has to avoid that the token is marked in the list as „planned-to-spend“ and later removed from it during token aggregation. Though in both actions the peer is not involved.

Malicious peers trying to remove tokens from the token list of another peer must guess token ids of existing tokens. That is very hard because and the creation date and time in milliseconds and the random serial number have to be guessed correctly. Therefore, this kind of messages is obvious malicious behavior and will be reported to the reputation system.

In P2P systems (even if using a DHT) it cannot be guaranteed that a remote account at the account holders is never lost. In such a case the account owning peer would not be eligible to receive services anymore. Since in the token-based system the tokens are stored locally, users can secure themselves against loss by making backup of their tokens. The loss of an account at the account holders will just influence the ability to check for double spending. Since a peer can not notice if its remote account is lost, it must assume that double spending would still be detected. Hence it will be discouraged to cheat.

5 Implementation

The token-based accounting mechanism has been implemented within the scope of the European project „Market Management of P2P Services“ (MMAPPS). MMAPPS builds a middleware for P2P systems supporting commercial applications. The middleware implementation is based on JXTA 2.2.1 [15]. Crypto-based peer ids (CBIDs) are applied to enable clear identification of message senders. The token-based accounting mechanism is used within an MP3-file sharing application as an incentive mechanism. Tokens are aggregated 1:1. As the utilized cryptography scheme RSA with 1024 bit keys has been selected. The tokens contain the user's public key as owner id.

Messages. Users might be concerned about the traffic volume token messages consume. Any additional new token contained in a message increases its size by 540 byte. Used tokens contain as accounting information the service level agreement id, content id, file name of transferred file, and applied tariff. As provider id the service provider's public key is included in the token. A message containing one used token has a size of ca. 1800 bytes. Each additional used token in a message increases the message size by ca. 1043 bytes. If we compress the tokens with a normal zip-algorithm, message sizes can be reduced with a factor of about 3. Table 1 shows the details. The figures contain an XML-overhead for message identification of ca. 140 bytes.

Table 1: Message Sizes in Bytes

	New Tokens		Used Tokens	
	first Token	add. Token	first Token	add. Token
uncompressed	1.284	540	1.797	1043
compressed	976	ca. 140	1293	ca. 280

Secured TLS JxtaBiDiPipes are used to exchange tokens between transaction partners and to transfer tokens for the token aggregation procedure. Messages with the account holders and the quorum of trusted peers are exchanged using the JXTA Resolver Service. These messages contain only token ids and are signed. Depending on the amount of token ids sent, messages have a size of about 400 bytes, including an XML-overhead of 140 bytes. Performance figures on these JXTA protocols are given in [14].

6 Comparison With Remote Accounts

In this section the performance of token-based accounting mechanism is compared against a competitive approach that uses remote accounts to hold the collected accounting information. To facilitate the comparison we focus on KARMA as presented in [27].

6.1 KARMA Overview

KARMA is an economic framework for P2P systems focusing on security considerations. KARMA utilizes a scalar value called *karma* to represent each peer's balance of contributed/consumed resources. KARMA follows a Remote Accounts approach proposing Pastry as its underlying DHT. Groups of nodes defined as *bank-sets* keep track of the karma belonging to users. Each bank-set is constructed of *k* peers. In every transaction the bank-set is updated to account the karma spent or collected. Users with negative karma are not allowed to consume system resources. Figure 6 illustrates the steps taken to complete a transaction.

6.2 Use Cases Analysis

The comparison of the two accounting approaches is both qualitative in terms of offered functionality and quantitative in terms of achieved performance. Performance is expressed as the number and the size of the messages required performing an operation. However, for KARMA no data about message size is given.

Two main use cases are of interest in this comparison. i.e. the system maintenance procedure and the user transactions.

6.2.1 Maintenance

The maintenance phase includes two different types of activities that are realized in different ways in the two systems. The first type includes maintenance actions to handle the high churn rate of peers joining and leaving. The second type involves maintenance actions that handle security-related or economic aspects such as preserving secrets or introduced inflation, respectively.

KARMA. KARMA activates the maintenance procedure when a peer belonging to a bank-set leaves the system or when a new peer joins the system. In the first case another peer replaces the departed one. In the latter the new peer is integrated into the correct bank-set. In both cases a number of messages is exchanged to restore the information hold by that peer and to redistribute the assigned responsibilities.

KARMA introduces *epochs* to handle the inflation/deflation problem. The complexity of this procedure is $O(n^2)$, where *n* is the number of peers in the system.

Token-based accounting. The maintenance of the account holders in the token-based approach is similar to the maintenance of the bank-set in KARMA.

The token-based accounting does not enforce a specific economic system. Of course a similar concept to epochs could be applied to the token-based accounting mechanism.

Additional costs arise from the requirement to keep the system's private key secret. This involves calculating key updates at one quorum of trusted peers and distributing new key parts afterwards to the rest of the trusted peers.

Comparison. Table 2 summarizes the complexity of the maintenance actions, where *n* denotes the total number of peers in the system, *k* denotes the size of the

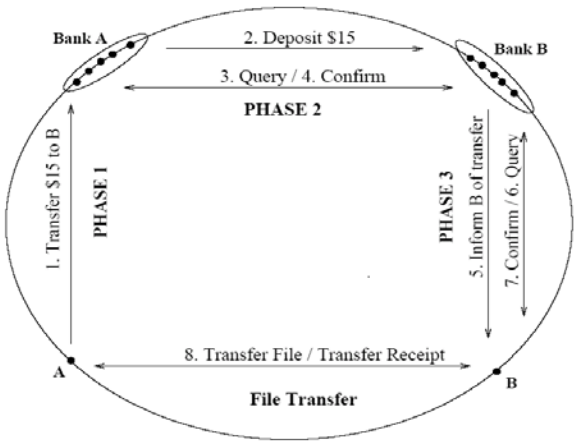


Figure 6:KARMA Transcation Procedure [27]

bank-sets, *m* denotes the number of trusted peers in the system, and a (*l*, *m*) secret sharing scheme is used.

Table 2: Maintenance complexity

	KARMA	Token-based
epoch	$O(n^2)$	--
node arrival	O(k)	O(k)
node departure	O(k)	O(k)
key update calculation [21]	--	$O(l^2)$
key update distribution [21]	--	$mO(l^2)$

6.2.2 Transactions

The major part of traffic in P2P systems results from transactions.

KARMA. A typical transaction in KARMA requires the close interaction of the two bank-sets (that store the accounting information of both the producer and the consumer). The actions are illustrated in Figure 6. KARMA utilizes three-way hand-shaking techniques to ensure the validity of critical actions, such as Karma transfer. Synchronization of the involved bank-sets takes place after each Karma transfer. However, this turns to be a costly procedure with questionable usage for well organized attacks. Finally, KARMA introduces an additional mechanism to ensure action synchronization among the bank-set members.

Token-based Accounting. For comparison reasons we calculate the transaction costs of the token-based accounting mechanism as one trusted transaction with check for double spending and an additional token aggregation procedure.

Comparison. Table 3 summarizes the number of messages required in every step to complete the transaction. *k* denotes the size of the bank-set or account holder set. An (*l*, *m*) secret sharing scheme is used. To compare the two approaches in a more realistic manner, the amount of messages exchanged for up to 100 transactions is shown in Figures 7 for KARMA and Figure 8 for the token-based accounting approach. For the latter a quorum size was set to 7. Also, it must be assumed that users do not exchange each foreign token immediately but do the exchange in batches. The calculations assume a maximum batch size of 20 tokens. Further, a transaction has a size of five tokens (each token send separately) and no transaction has the same transaction partners. That would reduce the needed amount of messages further.

Table 3: Transaction complexity

KARMA		Token-based	
1 Transfer Request	O(k)	Unsigned token transfer	1
2-4 Deposit/Query/Confirm	$O(k^2)$	Signed token transfer	1
5-7 Inform/Query/Confirm	O(k)	Check for double spending	O(k)
Pre-step synchronization	O(k)	Aggregation: send tokes to trusted peer	1
Post Synchronization	$O(k^2)$	Update token lists	O(k)
		New tokens to quorum	O(l)
		New tokens to owner	O(l)

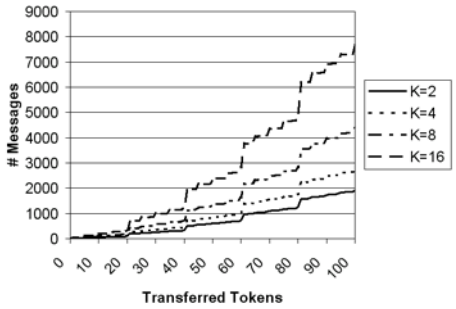


Figure 8:Transaction messages for token-based accounting

The difference arises from the different use of the account holders respectively the bank-set. In token-based accounting a message sent to the account holders does not require the proof of origin, like in KARMA. A peer just needs to proof that it received specific tokens from a peer. Therefore, the correctness of a message is derived from its content. In contrast in KARMA the correctness of a message is derived through the correctness of the message sender. Further, account synchronization in the token-based approach is only needed if the sender of an update message does not receive a confirmation. Therefore, just O(*k*) messages are needed during a transaction.

7 Summary & Conclusions

One of the biggest challenges for a wider deployment of P2P systems is to retrieve, collect and use information about the resource utilization within the system. It is crucial that the information is secure and reliable while the core features of P2P (i.e. decentralization, autonomy of peers, flexibility and dynamics) are still maintained.

This paper presentes a flexible and trustworthy token-based accounting mechanism for P2P-systems. Its purpose is to collect accounting information of transactions. This information can be used to coordinate the behavior of the system's entities to achieve a higher system performance. Further, the collected information can be used as basis for pricing and price

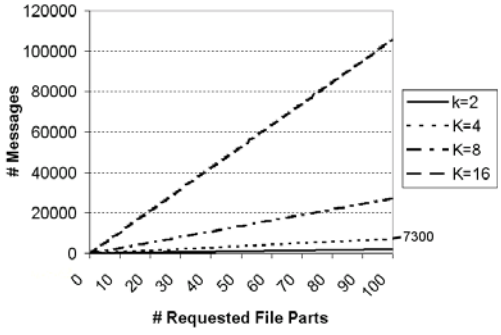


Figure 7:Transaction messages in KARMA

finding processes. Moreover, this builds the foundation for the development of a market within P2P systems. Further, the collected accounting information could be basis for a payment system to support commercial applications.

Since the responsibility of creating tokens is delegated to a randomly selected quorum of peers, fraudulent behaviour is prevented. Only if all peers in the quorum would be malicious, tokens can be forged. Further, a trustable payment mechanism is available that does not require to involve a third party. Thus, this approach is especially scaleable.

The token-based accounting scheme is very flexible through the introduction of the aggregation function. Here the exchange ratio of use tokens against new tokens can be defined by the usage policy. Hence different economic models can be implemented.

In comparison to an accounting mechanism using remote accounts the token-based approach has clear advantages in terms of the number of exchanged messages.

Acknowledgements

This work has been performed partially in the framework of the EU IST project MMAPPS “Market Management of Peer-to-Peer Services” (IST-2001-34201). The authors like to acknowledge discussions with all of their project partners.

References

[1] E. Adar, B. A. Huberman: *Free Riding on Gnutella*, In: First Monday, volume 5, number 10, October 2000.

[2] H. Appel, I. Biehl, A.Fuhrmann, M. Ruppert, T. Takagi, A. Takura, C. Valentin: *Ein sicherer, robuster Zeitstempeldienst auf der Basis verteilter RSA-Signaturen*; Technical Report, No. TI-21/99, Technische Universität Darmstadt, 1999

[3] A. Agrawal, D. J. Brown, A. Ojha, S. Savage: *Bucking Free-Riders: Distributed Accounting and Settlement in Peer-to-Peer Networks*; Technical Report, CS2003-0751, UCSD, June 24, 2003.

[4] D. Boneh, M. Franklin: *Efficient Generation of Shared RSA keys*; in Journal of the ACM (JACM), Vol. 48, Issue 4, pp. 702--722, July 2001.

[5] A. Barmouta, R. Buyya: *GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration*; 17th Annual International Parallel & Distributed Processing Symposium (IPDPS 2003) Workshop on Internet Computing and E-Commerce, April 22-26, 2003, Nice, France.

[6] D. Chaum, A. Fiat, and M. Naor. *Untraceable electronic cash.* In CRYPTO '88, vol. 403 of LNCS, pp. 319--327. Springer Verlag, 1990.

[7] Crypto-ID Project, http://crypto-id.jxta.org/

[8] Y. Desmedt and Y. Frankel: *Threshold cryptosystems*; In Proc. CRYPTO '89, volume 435 of LNCS, pages 307-315. Springer-Verlag, 1989.

[9] R. Dingledine, M. J. Freedman, D. Molnar: *Accountability*; In Peer-To-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly & Associates, Chapter 16, pp. 217 - 340, 1st edition, March 15, 2001.

[10] D. Dutta, A. Goel, R. Govindan, H. Zhang: *The Design of A Distributed Rating Scheme for Peer-to-peer Systems*; In: Proceedings of the Workshop on the Economics of Peer-to-Peer Systems, Berkeley, California, June 2003.

[11] eDonkey2000; http://www.edonkey2000.com.

[12] eMule Project; http://emule-project.net/

[13] P. Golle, K. Leyton-Brown, I. Mironov and M. Lillibridge: *Incentives for Sharing in Peer-to-Peer Networks*, WELCOM'01

[14] E. Halepovic, R. Deters: *The Costs of Using JXTA*. In: Proc. of Third International Conference on Peer-to-Peer Computing 2003; pp. 160-167.

[15] Project JXTA: http://www.jxta.org.

[16] S. Kamvar, M. Schlosser, H. Garcia-Molina: *EigenRep: Reputation Management in P2P Networks*; To appear in Proceedings of the 12th International World Wide Web Conference, May, 2003.

[17] KaZaA: http://www.kazaa.com.

[18] G. Medvinsky, B. C. Neuman: *NetCash: A design for practical electronic currency on the Internet*; In Proceedings of 1st the ACM Conference on Computer and Communication Security November 1993.

[19] T. Moreton, A. Twigg: *Trading in Trust, Tokens, and Stamps*; In: Proceedings of the Workshop on the Economics of Peer-to-Peer Systems, Berkeley, California, June 2003.

[20] Project Mojo Nation: *Peer-driven Content Distribution Technology*; http://www.mojonation.net/, February 2000.

[21] T. Rabin. *A simplified approach to threshold and proactive RSA*. In: Proceedings of Crypto, 1998.

[22] R. L. Rivest, A. Shamir: *PayWord and MicroMint: Two Simple Micropayment Schemes*; Security Protocols Workshop, pp. 69-87, 1996.

[23] A. Rowstron, P. Druschel: Pastry: *Scalable, distributed object location and routing for large-scale peer-to-peer systems*; IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[24] B. Schoenmakers: *Basic Security of the ecash^{TM} Payment System*; State of the Art in Applied Cryptography, Course on Computer Security and Industrial Cryptography, Leuven, Belgium, June 3--6, 1997 Revised Lectures, B. Preneel, V. Rijmen (eds.), volume 1528 of Lecture Notes in Computer Science, Berlin, 1998.

[25] A. Shamir: *How to share a secret*; in CACM, 22(11), pp. 612-613, November 1979.

[26] W. Thigpen, T. J. Hacker, L. F. McGinnis, B. D. Athey: *Distributed Accounting on the Grid*; In Proceedings of the 6th Joint Conference on Information Sciences, pp.1147-1150, 2002.

[27] V. Vishnumurthy, S. Chandrakumar, E. G. Sirer: *KARMA : A Secure Economic Framework for Peer-to-Peer Resource Sharing*; In: Proceedings of the Workshop on the Economics of Peer-to-Peer Systems, Berkeley, California, June 2003.

[28] B. Yang, H. Garcia-Molina: *PPay: Micropayments for Peer-to-Peer Systems*; In: Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), Washington D.C., October 2003.