Token-based Accounting for P2P-Systems

Nicolas C. Liebau[1], Vasilios Darlagiannis[1], Andreas Mauthe[1] und Ralf Steinmetz[1]

Multimedia Communications Lab, KOM
Darmstadt University of Technology, Germany
E-Mail: [Nicolas.Liebau|Vasilios.Darlagiannis|Andreas.Mauthe|Ralf.Steinmetz]@kom.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 or service usage. Conforming to the peer-to-peer paradigm, the tokens are issued using a decentralized mechanism. Within peer-to-peer systems the proposed accounting scheme can be used to overcome information deficits. Thus, it constitutes the basis for coordination and control mechanisms as well as for pricing in commercial scenarios in fully decentralized systems. The presented scheme is compared against an alterative approach showing the advantage of the token-based scheme 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 [2] host huge amount of content. Users of P2P file sharing applications are accepting performance constraints compared with client/server systems because they are retrieving content at virtually no cost. One reason for this reduced performance of P2P 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 [4]. 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 provide the missing information. Using this information, coordination of the available resources becomes feasible in order to improve the overall system performance. 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 itself 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 and 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 may 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 is supposed to be used in different scenarios it should be flexible to support different kinds of coordination mechanisms.

To tackle the discussed problem this paper proposes a token-based accounting scheme. 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.

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, a major characteristic that distinguishes different accounting schemes is the location of the stored receipts.

Local Accounts. Using local accounts, a 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 [3] 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 itself. Even if the receipts are signed by the transaction partner, fraud is easily possible through malicious 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 [5,11,19] this approach is applied. In [5] 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 instance, for Grid Computing such a system is presented in [6]. 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 why their availability could be limited. 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 [15]. The authors claim that eventually a completely free stable market will develop. Further, in [18] self-issued tokens are used for accounting in Grid Computing. (b) A central, trusted "bank" issues the tokens. Mojo Nation uses this solution as well as some existing micro payment schemes like NetCash [13]. A micro payment scheme especially tailored to P2P systems is presented in [20]. 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 [1] are also applicable. Apart from the certification authority it is intended to avoid any central element.

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 presented e.g. in [12].

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 based on 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, i.e. 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 [8]. Therefore, tokens have a clear owner that is contained in the token. This enables local tokens storage. 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 process, peers exchange the collected foreign tokens against new ones. To achieve trustworthiness new tokens are signed with the systems shared private key using threshold cryptography [10]. Thus, a token must be signed by a quorum of peers to become valid. The token structure ensures protection against forgery, double spending and stealing. The three basic protocols of the token-based accounting system are *Token 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 right hand 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.

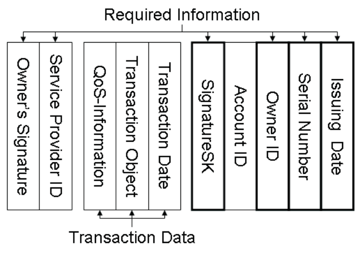


Figure 1: Token Structure

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, it 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 its 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, i.e. the system signature, the service provider as well as the service providers signature. This prevents tokens from being stolen. 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. A token has no intrinsic value; it rather presents an accounting event. The value of a token is determined in the token aggregation process.

3.3 Token Aggregation

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

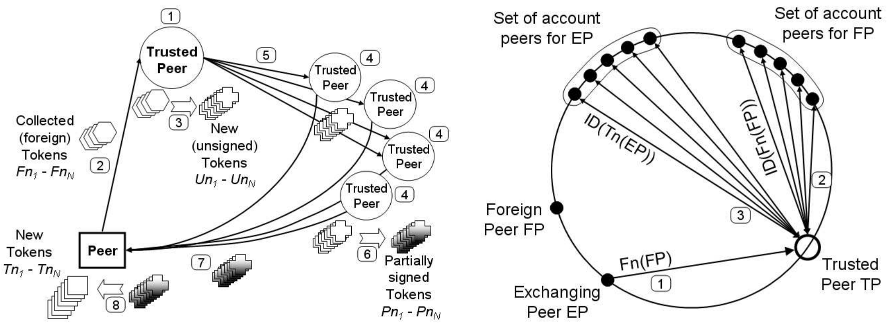
First the *exchanging peer* EP locates a *trusted peer* TP (1). Trusted peers are eligible to exchange tokens and possess one part of the system's private key [10]. EP sends its N collected foreign tokens $(Fn_1, ..., Fn_N)$ to TP (2). TP checks the foreign tokens for their validity. Only tokens signed 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 [10] TP now locates further trusted peers (4). EP is not allowed to choose the quorum of trusted peers itself. This alleviates the problem of potential collaboration and fraud. The number of required trusted peers to sign a token is determined by the used secret sharing scheme. The system's trustworthiness increases proportional with the size of the quorum of trusted 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).

It is important to mention that the aggregation function adds an additional degree of freedom to the system. With an appropriate aggregation function specific economic systems can be implemented.



(a) Token Aggregation Process (b) Check for Double Spending

Figure 2: Token Operations

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 (the account owners) there is a set of account holding peers, i.e. the *account holder set*. The account holder peers are organized in a DHT manner, such as Pastry [17] (see Figure 2 (b)). Account holders hold a list of tokens currently issued to the account owner. The list is filled with the required information during token aggregation. After new tokens have been created (Figure 2 (a), step 3), TP sends a list of these new tokens to the exchanging peers account holders (Figure 2 (b), step 3).

During the token validity check of the token aggregation process, TP will ask the account holders responsible for a token, if the token is valid (Figure 2 (b), 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. TP will discard such a token and the P2P system's reputation mechanism will be informed about the incident.

In order to avoid message manipulation, every message sent to the account holders must be signed with the senders 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 takes place only when peers of that set join or depart from the system. Consistency checks are only necessary, if the sender does not receive all confirmation messages.

3.5 Transactions

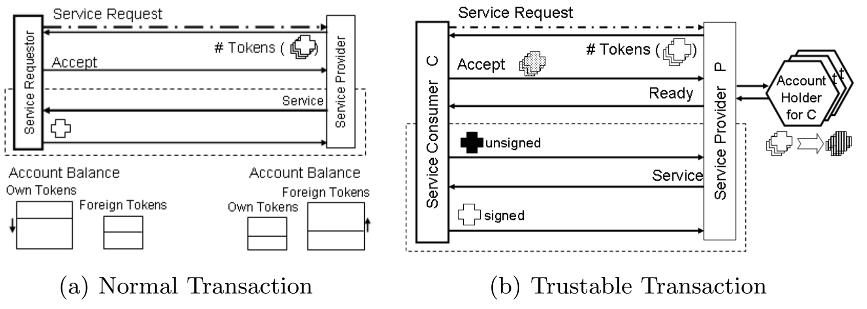
During transactions the token-based accounting system accounts for resource usage, service usage, or a combination of both. Service usage is valued differently than resource usage. A service for example detects water marks in pictures. Since special software is needed to provide such a service, it is 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.

Standard transaction. The standard transaction process is shown in Figure 3 (a). 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 it 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 fills in the required accounting information. C has no intention to falsify the information, because it influences only the token exchange of P. Then C signs the token with its 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 each transaction C's own token balance decreases and P's foreign token balance 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 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 3 (b) 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 for 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 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 provides further incentives against such malicious behavior. On the other hand, if both peers are consenting to cancel the transaction, C does not lose its tokens. The "*planned to spend*"-mark just needs to be removed from the tokens in the token list at *AH(C)*.



(a) Normal Transaction (b) Trustable Transaction

Figure 3: Transaction Procedures

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 system has been designed to provide a high degree of trust for distributed systems.

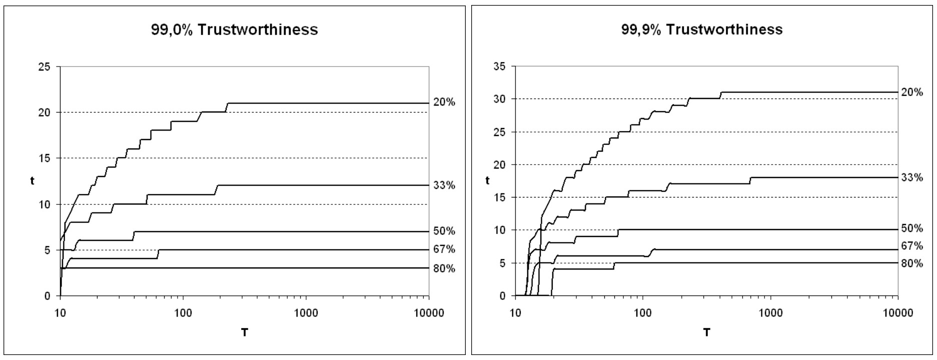
Robbery. Tokens were designed to eliminate robbery. Tokens contain the owner id that cannot be changed without detection through the system signature. Spent tokens contain the token receiver secured through the owner's signature.

Forgery. The system signature on each token ensures that the basic token data cannot be changed and that no peer can create tokens 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.

This can be achieved if in a quorum of trusted peers is at least one trustworthy peer. The probability of a quorum consisting of at least one good peer can be determined using the hypergeometric distribution. The resulting probability p defines the trust level of the system according to:

$$p(T,t,p_g) = \frac{\binom{T\cdot(1-p_g)}{t}}{\binom{T}{t}}, \text{ where } \begin{array}{ll} T & \text{number of trusted peers} \\ t & \text{quorum size} \\ p_g & \text{percentage of good peers} \end{array}$$

Figure 4 shows the required quorum size for specific trust levels. Moreover, because the trusted peers are not aware which other peers belong to a quorum, having only bad peers in a quorum does not mean that this results in fraud. The chosen trusted peers must also collaborate. Thus, the quorum peers must know which other peers have been chosen for the quorum.



(a) Trust Level 99.0

(b) Trust Level 99.9

Figure 4: Required Quorum Size for Trust Levels by Percentage of Good Peers

Furthermore, peers can only become trusted and receive a part of the shared system private key, if their reputation is above a specific threshold value. Accordingly, the proportion of bad peers among the trusted peers can be assumed less than the proportion of bad peers in the whole system. The actual trust threshold value depends on the used reputation system.

Additionally, threshold cryptography provides different proactive mechanisms to secure the key from being compromised. The key parts will be updated periodically using proactive secret sharing [16]. This makes the old key parts obsolete without changing the actual key. The system's public key remains the same. Further, a new system key will be created periodically using the decentralized method presented in [7]. 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 for 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. Rule breaches are reported to the reputation system. 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 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 a 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 Performance Analysis

We have implemented the token-based accounting system based on JXTA 2.2.1 [14]. Measurements of message sizes were used to simulate the accounting scheme with the simulator presented in [9].

To study the performance of the token-based accounting system two use cases have to be distinguished - costs for maintenance and costs for transactions.

Maintenance. Maintenance costs arise from keeping the remote accounts consistent and from the requirement to keep the systems 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. Table 1 summarizes the complexity of the maintenance actions, where k denotes the size of the bank-sets and a (t, T) secret sharing scheme is used, where T denotes the number of trusted peers in the system.

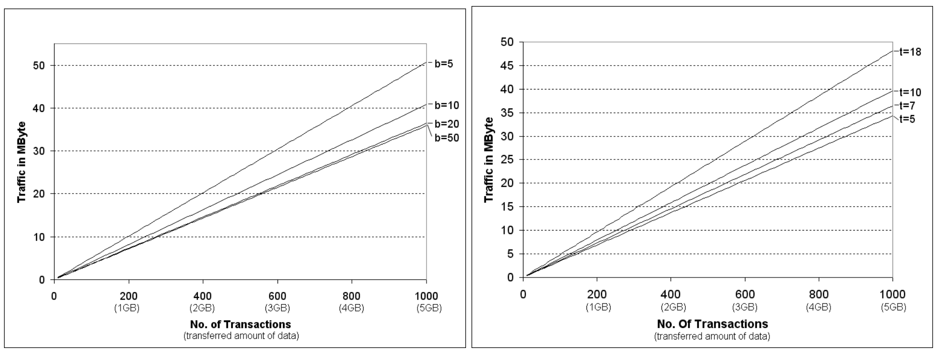
Table 1: Account Holder Set & System Key Maintenance Complexity

Account Consistency		System key related operations	
node arrival	O(k)	key update calculations	O(t^2)
node departure	O(k)	key update distribution	O(t^2)

Transactions. For the analysis we assume a conservative ratio of 67% good peers in the system. Further, we set a trust level of 0,1% which results in a quorum size t of 6 trusted peers. Furthermore, we set the account holder set size k to 4. We model a file sharing scenario, where for 1 MB download 1 token is required and the average file size s is 5 MB. Users exchange tokens in different batch sizes b. The trustable transaction procedure is used. If n transactions are carried out the average number of accounting messages M sent in such a scenario results in:

$$M(n,k,t,b) = n(2s+2k) + \frac{ns}{b}(1 + 2k\frac{b}{s} + 2k + 2t)$$

For 100 transactions exchanging 500 tokens with a batch size of 20 results in 3125 messages. Simulating this scenario the token-based accounting system creates an additional overhead of less than 1% (for the mentioned example it is less than 3,5 MB overhead for file transfers of 500 MB). Figure 5 (a) shows the generated traffic for different batch sizes and up to one million transactions. As it can be expected, the overall traffic generated by the token-based accounting system is reduced as the batch size increases. However, the effect levels off after a batch size of 20. Figure 5 (b) shows the influence of increased quorum size. The effect is not strong. Even with a very high trust level (t=18) the system still generates not more than 1% of overhead. The effect of size of account holder set for the generated traffic is very small and therefore the graph is omitted here.



(a) by batch size (b) by quorum size

Figure 5: By Token-Accounting Scheme Generated Traffic

6 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 are still maintained.

This paper presents a flexible and trustworthy token-based accounting scheme 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 finding processes. Moreover, this builds the foundation for the development of a market within P2P systems. Further, the collected accounting information could be the 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 behavior is prevented. Only if all peers in the quorum would be malicious, tokens can be forged. Also, a trustable payment mechanism is available that does not require to involve a third party. Thus, this approach is especially scalable.

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

The further steps to investigate next are detection of the need for a system key update or system key creation procedure. Also the economic behavior of the system with respect to inflation and deflation will be evaluated using simulations.

References

1. Crypto-id project. http://crypto-id.jxta.org/, 2004.
2. edonkey2000. http://www.edonkey2000.com, 2004.

3. emule project. http://emule-project.net/, 2004.
4. E. Adar and B. A. Huberman. Free riding on gnutella. First Monday, volume 5, number 10, Oktober 2000.
5. A. Agrawal, D. J. Brown, A. Ojha, and S. Savage. Bucking free-riders: Distributed accounting and settlement in peer-to-peer networks. Technical Report CS2003-0751, UCSD, June 2003.
6. A. Barmouta and R. Buyya. Gridbank: A grid accounting services architecture (gasa) for distributed systems sharing and integration. In *17th IPDPS, Workshop on Internet Computing and E-Commerce*, Nice, France, April 22-26 2003.
7. D. Boneh and M. Franklin. Efficient generation of shared rsa keys. *Journal of the ACM (JACM)*, 48(4):702–722, July 2001.
8. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO '88*, volume 403 of LNCS, pages 319–327. Springer Verlag, 1990.
9. V. Darlagiannis, A. Mauth, N. Liebau, and R. Steinmetz. An adaptable, role-based simulator for p2p networks. In *Proceedings of MSV'04*, pages 52 – 59, Las Vegas, Nevada, USA, june 2004.
10. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO '89*, volume 435 of LNCS, pages 307–315. Springer-Verlag, 1989.
11. D. Dutta, A. Goel, R. Govindan, and 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.
12. S. Kamvar, M. Schlosser, and H. Garcia-Molina. Eigenrep: Reputation management in p2p networks. In *Proceedings of the 12th International World Wide Web Conference*, 2003.
13. G. Medvinsky and B. C. Neuman. Netcash: A design for practical electronic currency on the internet. In *Proceedings of the 1st ACM CCS Conference*, November 1993.
14. Sun Microsystems. Project jxta.
15. T. Moreton and 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.
16. T. Rabin. A simplified approach to threshold and proactive rsa. In *Proceedings of Crypto*, 1988.
17. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware*, pages 329–350, Heidelberg, Germany, November 2001.
18. W. Thigpen, T. J. Hacker, L. F. McGinnis, and B. D. Athey. Distributed accounting on the grid. In *Proceedings of the 6th JCIS*, pages 1147–1150, 2002.
19. E. G. Sirer V. Vishnumurthy, S. Chandrakumar. 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.
20. B. Yang and H. Garcia-Molina. PPay: Micropayments for Peer-to-Peer Systems. In *Proceedings of the 10th ACM CCS Conference*, Washington D.C., October 2003.