32. Accounting in Peer-to-Peer-Systems

Nicolas C. Liebau, Vasilios Darlagiannis, Oliver Heckmann
(Darmstadt University of Technology)
Andreas Mauthe (Lancaster University)

32.1 The Purpose of Accounting

Today's Peer-to-Peer (P2P) systems seem to work fine for file sharing applications. Though, is that true for all kinds of applications or is it just true for file sharing of copyright protected content? In the latter case, clear incentives for sharing exist - users download copyright protected media content for free. Even here free riding is a widespread behavior [11]. Therefore, many file sharing systems introduced incentive systems, like the eMule credit system [192] or BitTorrent [320]. Obviously, for legally used Peer-to-Peer file sharing applications giving incentives to users for sharing their resources in a fair manner is an important feature.

Another important feature of Peer-to-Peer systems is fair resource allocation. In fact, some people define the free rider problem as part of the resource allocation problem [167]. However, we distinguish between incentives and resource allocation. The goal of resource allocation is to allocate the available resources in a fair manner among the system's entities such that bottlenecks are avoided. Resource allocation is an important feature for applications where quality of service is critical.

We can imagine the Peer-to-Peer paradigm being used for business applications where users sell and offer services. Here, it is an important feature that users can prove they delivered a service or paid for a service received. This requires trustworthy bookkeeping.

All three of these features - incentives, resource allocation, and business support - have a common requirement. Information is needed about the actions in the Peer-to-Peer system as well as evaluation of this information. Peer-to-Peer systems cannot easily provide this kind of information because it is completely distributed. The lack of information leads to the free rider problem because users feel completely anonymous. Furthermore, this missing information makes it difficult to do resource allocation. And without knowing who bought/sold a service, business applications are impossible. The missing functionality is obviously accountability. Accountability in information systems is defined as the process of tracing information system activities to a responsible source [443]. If we can introduce accountability into Peer-to-Peer systems, we can solve the problem of missing information and with it add the aforementioned features to Peer-to-Peer systems.

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 547-566, 2005.

32.2 Why Is Not Accounting in Peer-to-Peer Straight Forward?

Looking at Client/Server systems, accounting was never seen as a major issue because it is intuitively easy. It was not even a real research topic (see e.g. RADIUS [509]). To do accounting in Client/Server systems the server simply logs every access activity of the client. The server can do so because all communication is done either between client and server or between two clients over the server. Two clients never communicate directly.

In Peer-to-Peer we have exactly the opposite situation. There is no server but almost exclusively direct Peer-to-Peer communication. There is no unit in the network that could log all the communication within the system. All that can be done in Peer-to-Peer is limited to local observations. A peer logs itself the communication in which it is involved. Accordingly, there is no straightforward solution for gaining system-wide knowledge about the activities in the system using this locally gathered information.

Furthermore, the problem is very complex because the collected information must be stored and used in a trustworthy manner. Otherwise the accounting system will not be reliable. For example KaZaA's participation level [342] is an accounting system with a trustworthiness problem. The system stores locally at the peer information about the peer's contribution to the system in form of uploads. This information is evaluated and determines the maximum download speed for the peer itself. Obviously, there is a strong incentive to manually increase the participation level in order to cheat. And cheating is easy, especially since *KaZaA Lite* (also known as *K++*), which does not include the download limiter, is freely available [344].

This real life example leads directly to the next issue with accounting in completely distributed autonomous systems: If there are defined rules of behavior then there must also be a way to enforce them. In the case of KaZaA the participation rule cannot be enforced because cheating cannot be detected. Accordingly users get benefits only from using the KaZaA Lite client application instead of using the original KaZaA client application.

Consequently, three aspects make accounting in Peer-to-Peer difficult: The required information is distributed throughout the whole system, the information must be stored and used in a trustworthy manner, and behavior rules for incentive systems or resource allocation must be enforceable.

Finally, a fourth aspect should not be forgotten: An accounting system for Peer-to-Peer should use the scarce resources of the system (e.g. bandwidth) economically.

32.3 The Solution Space – A Classification of Peer-to-Peer Accounting Schemes

This section outlines the different design parameters for building an accounting system that meets the described requirements. The problem can be structured into two main parts: *Information collection* covers the options how and in which form information is collected. *Information storage* concerns the options of where the collected information can be stored.

32.3.1 Information Collection

In Peer-to-Peer systems there are only a few options where accounting information could be collected. Assuming real Peer-to-Peer connections (without a third peer in the middle as a "trusted party"), the information can only be collected at the service provider peer, at the service receiver peer, or at both peers. This collection process usually consists of a metering process and an evaluation process. The metering process measures the used bandwidth (the amount of data received/sent), time (the duration of the service), or collects some service specific signals like "file complete". The evaluation process interprets this information to generate accounting events. These events determine when an accounting record is created.

Accounting records contain the accounting information and can take several forms.

Plain Numbers

The simplest form is plain numbers. For every peer there exists an account balance stored somewhere in the Peer-to-Peer systems. For example, for each MByte uploaded the peer's account balance is increased by one. Plain numbers can be changed very easily, therefore, this kind of accounting record is especially vulnerable to fraud. Nevertheless, this form of information consumes very little space and bandwidth when it is transferred between peers.

Receipts

Receipts are documents of fixed form that can contain all kinds of accounting information, e.g. transaction partners, transaction object, metering information, and an evaluation of it. This evaluation could be the information how this receipt modifies the peer's account balance. This part of information would be similar to plain numbers. In comparison with plain numbers, the additional information of a receipt adds some trustworthiness to the account-

ing information, because more information is available, which could help to detect fraud.

Signed Receipts

Signed receipts are normal receipts with a signature to add integrity to the information contained in the receipt. This is a very important step to achieve trustworthiness of information. Additionally, a signature authenticates a receipt. Working with signed receipts requires that every peer owns a private/public key pair. Such a key pair can e.g. be issued by a central authority or peers could create the key pair themselves and using Public Key Infrastructure (PKI) the keys would be certified. E.g. JXTA [412] uses self signed certificates. This approach has the disadvantage that these keys are not persistently associated with a peer. A peer can easily create a new key pair if it wants to revoke its identification. This problem can be elaborated for example by using a Web of Trust as implemented by PGP [543] or the cryptoID-project of JXTA [139]. A central authority implementing a PKI does not have this disadvantage, though its usage seems contrary to the Peer-to-Peer paradigm. Nevertheless, the responsibilities of a central authority within the Peer-to-Peer system could be reduced to issuing certificates with keys and approving the validity of keys. The services offered in the system would continue to be delivered in a Peer-to-Peer manner.

Receipts are normally signed by the author of the contained information. Indeed, receipts could also be signed by the transaction partner to include an agreement about the information in the receipt. This can also be accomplished by bilaterally signing a receipt. However, one should bear in mind that each signature increases the size of the receipt. And size matters for the efficiency of the accounting system in terms of created overhead. Depending on the cryptography mechanism used, the size of the signature could be large. For example, today's RSA signatures are typically chosen to be not smaller than 1024 bit.

Tokens

In the context of this chapter the term *token* is used for any kind of issued document. An issuer issues a specific number of these documents (tokens). Thus, the number of tokens available in a Peer-to-Peer system can be limited. This results in a specific characteristic that is otherwise hard to achieve - through issuing a limited number of tokens they become a scarce resource. Accordingly, tokens can represent other scarce resources. However, they are not that easy to handle like normal receipts, because two major problems must be addressed: *Forgery* and *Double Spending*, both of which have to be avoided. With respect to double spending a token must be clearly identifiable. Thus, a they must contain a unique identification. Also, there must be a

mechanism built into an accounting system to check for double spending. To avoid forgery it must be ensured that only the issuing authority can create a token and therefore, a token must be signed by the issuing authority with its private key.

There are several options for the token issuing authority in an accounting system. For example, in digital cash systems a central bank issues the tokens (e.g. [546]). This option ensures trustworthiness and control; however, this does not conform to the Peer-to-Peer paradigm. In [636] a micropayment scheme for Peer-to-Peer systems is presented that relieves the central bank many of its responsibilities. The second option is that each peer issues its own tokens. Here tokens are very similar to signed receipts. It is also difficult to control the number of tokens issued by every peer. Therefore, determining the economic value of a peer's token is difficult, too. The third option is a compromise between the first and second option. It is issuing tokens by a subgroup of peers and controlling the subgroup's actions in order to control the total number of tokens in the system. In the second part of this chapter an accounting system using this approach is presented.

Token-based accounting systems can further be distinguished by the usage of the tokens. Tokens can either be used as kind of micropayment system or as receipt, i.e. a token represents exactly a specific part of a provided/consumed service (e.g. 1 MByte data transfer). Tokens become receipts by adding accounting information to the token. Here, a token can represent all kinds of transactions or parts of transactions.

Proof of Work

Proof of Work (PoW) is another micropayment concept where a user has to show that he performed some computationally difficult problem to be eligible to receive a service [167]. Nevertheless, PoW cannot be redeemed by the receiver for something of value to him. Therefore, they can be used to avoid denial of service attacks or flooding attacks. Further, they do not present an economic measure [167]. Also, to create a PoW, CPU cycles that are a limited and that might be traded in the Peer-to-Peer system (and will then represent a scarce resource themselves) are needed. Therefore, PoW in general can be regarded as improper for accounting systems.

32.3.2 Information Storage

In whichever form accounting data is collected, it must be stored at some place in the network, a so called account. The account location is an important design parameter because it strongly influences the traffic overhead of an accounting system. For every transaction, one or more accounting records accumulate and must be transferred to the account. Also, the location influences

how easy it is for a peer to manipulate accounting records and defraud the system. There are three basic alternatives for the account location: accounts can be located locally at the peer that collects the data, at a central server, or remotely at a peer other than the collecting peer. Centrally held accounts obviously are contrary to the Peer-to-Peer paradigm. For every transaction communication with the central entity would be necessary to transfer the record. Therefore, this solution is out of the question. The advantages and disadvantages of the two other alternatives shall now be elaborated.

Local Accounts

Storing accounting records at the place where they accrue has the obvious advantage of reduced traffic, because the records do not need to be sent to the account holder. However, using local accounts poses an important trust problem. Using plain numbers or self-signed receipts to store information enables users to easily change the contained information in order to defraud the accounting system. The appearance of KaZaA Lite as competitor to KaZaA is an example to this behavior [344]. Therefore, receipts should be signed either by the transaction partner or a third party. Alternatively, tokens should be used. Accordingly, accounting records need to be transferred between the transaction partners. Therefore, in terms of bandwidth usage, storing accounting records locally is only advantageous in comparison to other storage location, if the transaction partner needs to get the records for some reason. For example, both transaction partners have to agree about the contained accounting information. Further advantages are that users have immediate control over the collected accounting records. No redundancy need be built into the system, because a peer's accounting records are not needed when it is offline. Also, users are themselves responsible for doing a data backup.

Remote Accounts

The alternative location for storing accounting records is third peers. Using third peers, hence separating account holders from account owners, clearly makes it more difficult for any one peer to fraudulently manipulate accounting records. Accordingly, depending on the Peer-to-Peer application's requirements, special mechanism to ensure information integrity such as signing might not be needed.

However, using remote accounts requires the exchange of more administrative messages between peers. This stems from the need for redundancy. Because the account holder is not always available, several account holders per peer, each holding a replication of the account, are required. All replications of an account need to be kept consistent. Therefore, mechanisms to detect potential inconsistencies as well as mechanisms for determining the ac-

tual account status in a situation of disagreement are required. For example, a Byzantine quorum might be used for the latter case.

32.4 Proposed Accounting Schemes

In this section the presented structure of Peer-to-Peer accounting systems is used to classify the known accounting systems. However, most of the related work was not designed as accounting systems but as economic frameworks. Economic frameworks contain an accounting mechanism and use it to introduce economic based incentives into the Peer-to-Peer system. That is, users have to "pay" accounting units to receive a service and gain accounting units if they provide a service. Accordingly, these systems are concerned with the economic value of an accounting unit. Therefore, they either include mechanisms to regulate the amount of accounting units in the Peer-to-Peer system or try to give users guidelines to assess the economic value of an accounting unit.

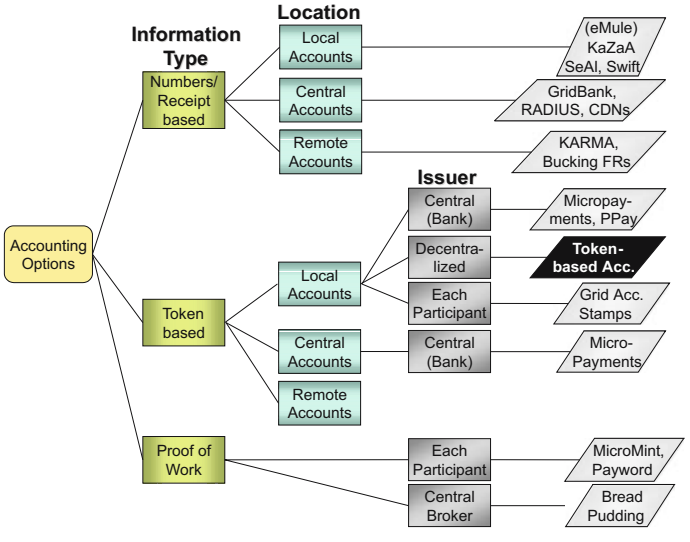


Fig. 32.1: Classification of the Related Work

32.4.1 Plain Numbers-Based Systems

Currently the two most widely used Peer-to-Peer accounting mechanisms are KaZaA's participation level [342] and eMule's credit system [192]. Both sys-

tems account for the amount of data uploaded and downloaded and store the collected information locally. KaZaA's system uses the ratio of uploads to downloads (measured in amount of data transferred) to calculate a peer's actual maximal allowed download speed. The higher this ratio is the higher is the maximal allowed download speed. This is a typical incentive mechanism for file sharing systems. However, KaZaA's system is easy to cheat because the accounting information is stored locally. In fact in the KaZaA clone KaZaA Lite [344] the participation level is removed. In contrast, eMule's credit system is used to determine a requestor's position in the provider's download queue. The position is determined by the amount of data the requestor uploaded to the provider. This system has the obvious advantage, that it cannot be cheated. The provider keeps his accounting records that only influence his own behavior. The disadvantage is that this system only accounts for local observations. A peer could have upload to the system much more than it downloaded. However, if it downloads from a peer to which it did not upload before, it will get a bad position in the provider's download queue.

Another system that uses local accounts to store plain numbers as accounting information is Swift [584]. In contrast to the both systems mentioned before, it is not used in practice yet. Swift basically is a behavior model for Peer-to-Peer file sharing to support fair large scale distribution of files in which downloads are fast. Each peer maintains a credit for every other peer it is connected to. A peer will only upload to a peer with a positive credit balance. Because the accounting data only affects the local peer behavior, peers have no incentive to falsify the collected information.

A system taking into account a peer's actions in the overall system is Karma [603]. Karma stores for every peer in the system a value that represents Its balance of uploads against downloads. This balance is stored at remote peers. These remote peers are called a peer's bank set. The bank set consists of multiple peers, for redundancy reasons. The balance of a peer must not be lost. Accordingly, a bank set is rather large - a suggested size is 64 peers. For every transaction the bank sets of the provider and receiver peer communicate to adjust the peers' balance according to the transaction value. Further, Karma includes the concept of an epoch. At the beginning of each epoch every peer's balance is adjusted accordingly in order to avoid inflation.

In [14] another system using remote accounts is presented. So called accountants store a peer's balance. Like in Karma the accountants are third peers. To ensure that the balance is not lost a set of accountants is required for each peer. With every transaction the balance of the two transaction partners is updated to the new value. A non-mediated and a mediated settlement protocol are presented.

There are other systems known to do accounting using numbers. However these systems are not compliant with the Peer-to-Peer paradigm. These systems use a central server to do the accounting. Examples are the accounting

mechanism in RADIUS [509], accounting mechanisms in content distribution networks [95], and the accounting mechanism for GRID called Gridbank [61].

32.4.2 Receipt-Based Systems

SeAl [453] is a Peer-to-Peer accounting system that uses locally stored receipts. SeAl is working based on favors. For every transaction a receipt is created and stored locally at the receiver and provider. As a result of the transaction, the receiver owes the provider a favor. A favor can be paid back by the receiver by providing a service to the provider. Also the provider can use a favor by redirecting service requests of other peers to the receiver. Furthermore, peers can publish Blacklist Reports about peers behaving against the system's rules. For each service request a score is calculated (using paid back favors and Blacklist Reports) that influences the request's position in the provider's request queue. Accordingly, not all accounting data is stored locally. However, for Blacklist Reports there exist no accounts for storage.

32.4.3 Token-Based Systems

One class of token based accounting systems are micropayment systems that use tokens as a micro currency. For payment these tokens are transferred between users. (Micropayment systems just modify centrally hold bank accounts on request belong to the plain numbers-based systems.) All micropayment systems use a central broker or bank. Thus, they are not appropriate for Peer-to-Peer systems. A micropayments system tailored to Peer-to-Peer systems is presented in [636]. It relieves the broker of some task and these tasks are facilitated by the peers of the system. As a result the broker can even go off-line for short time periods and the system can still continue to operate.

Mojo Nation [441] was one of the earliest Peer-to-Peer systems to use a payment protocol. Users had to use a virtual currency called Mojos to obtain a service from another peer. Mojo Nation still required a centralized trusted third party to issue the Mojos and to resolve double-spending issues.

A system using stamps for peers' "evidence of participation" is presented in [431]. Every peer issues personalized stamps and trades these with other peers. If peer A requests a service from peer B, peer A has to pay a specific amount of peer B's stamps back to B. There is no limit how many stamps a peer issues. However, if a peer issues too many stamps in comparison to its offered services the stamps will devalue. Thus, the peer will have difficulty obtaining other stamps, as rational nodes will not wish to purchase its stamps. This way the stamps protocol combines a virtual currency and reputation.

32.4.4 Proof of Work-Based Systems

In [520] two micropayment systems based on Proof of Works are presented. In both systems each participant issues his own secrets. As mentioned before, using such systems poses the problem for users to determine the economic value of another user's virtual currency. In order to overcome the need to identify the exchange rate of two users' virtual currency in [321] Proof of Work-based systems are extended to work with a central broker that issues secrets centrally. This way only one virtual currency exists in the system. However, there is no way known to construct such a system according to the Peer-to-Peer paradigm.

32.5 Token-Based Accounting Scheme

Within the scope of the EU Project "Market Management of Peer-to-Peer Services" (MMAPPS) we were in need for a highly flexible and trustworthy accounting scheme for Peer-to-Peer systems. In response to that need we developed the here presented token-based accounting scheme.

32.5.1 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 [139] 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 Peer-to-Peer 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 [333].

32.5.2 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 micropayments, namely anonymity and untraceability [112]. 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 [164]. 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*.

32.5.3 Token Structure

Figure 32.2 shows the information contained in a token. A new unused token contains the first five 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 are possible. The account id 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.

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

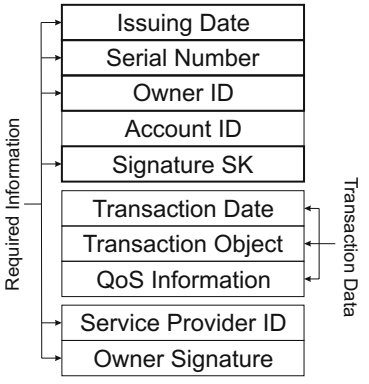


Fig. 32.2: Token Structure

as well as the service provider's 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.

32.5.4 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 32.3 (a).

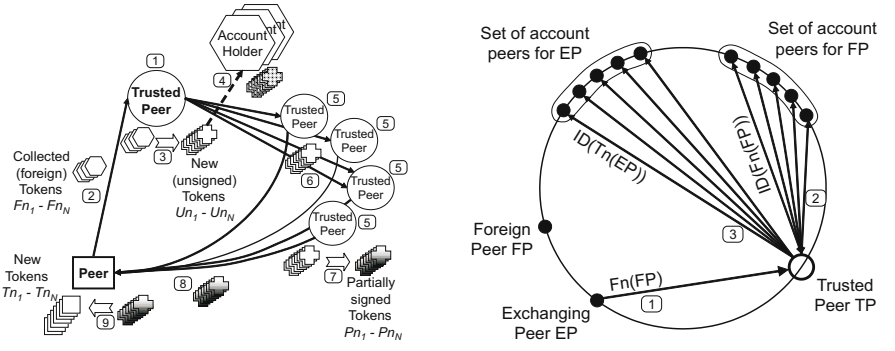
First the *exchanging peer* EP locates a *trusted peer* TP (1). Trusted peers are eligible to exchange tokens and possess one part of the system's private key [164]. EP sends its N collected foreign tokens $(Fn_1, ..., Fn_N)$ to TP (2). TP checks the foreign tokens for their validity. Only tokens signed 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 [164] 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

Fig. 32.3: Token Operations

32.5.5 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 [527] (see Figure 32.3 (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 32.3 (a), step 3), TP sends a list of these new tokens to the exchanging peers account holders (Figure 32.3 (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 32.3 (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 Peer-to-Peer 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 changes. 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.

32.5.6 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 32.4 (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 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 incentive to falsify this 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.

Transaction partners could try to gain tokens by not paying tokens after receiving a service or by not delivering the service after receiving tokens. In

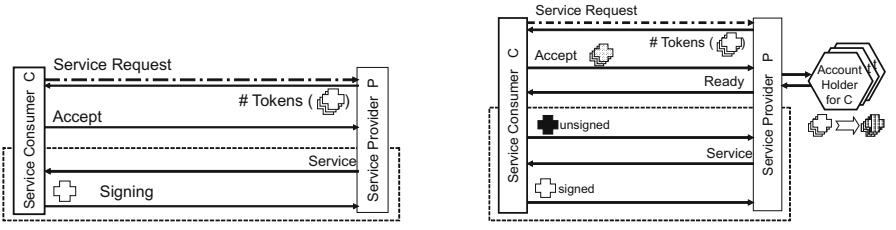
order to avoid that transactions can be split into several parts. Then C sends a signed token to P after P delivered a part of the service; e.g. C sends a tokens after each MByte received data of a 5 MByte file transfer. A further approach that eliminates the incentive for transaction partners to cheat on the other partner is now presented.

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 32.4 (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 proves 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. P informs $AH(C)$ in order to remove the "*planned to spend*"-mark in the token list.

32.5.7 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.



(a) Normal Transaction (b) Trustable Transaction

Fig. 32.4: Transaction Procedures

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

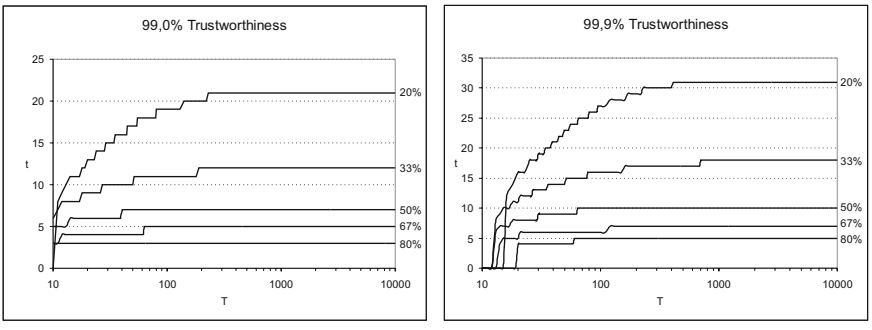
The system signature on each token ensures that the basic token data cannot be changed and that no peer can create tokens itself. Thus, the system signature prevents forgery and is crucial for the trustworthiness of the system. Accordingly, fraudulent collaboration of trusted peers must be avoided.

This can be achieved if in a quorum of trusted peers there 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 32.5 shows the required quorum size for specific trust levels. For example to achieve a trust level of 99.9% with 50% bad trusted peers in the system a quorum size of ten is required. However, because the trusted peers are selected using the aforementioned reputation system the percentage of bad trusted peers can be assumed to be much lower than 50%. 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 (bad) trusted peers must also collaborate. Thus, the quorum

peers must know which other peers have been chosen for the quorum. Thus, the archived trust level is higher.



(a) Trust level = 99.0 (b) Trust level = 99.9

Fig. 32.5: 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 [498]. 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 [81]. This is enforced by 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 Peer-to-Peer 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.

32.5.8 Performance Analysis

We have implemented the token-based accounting system based on JXTA 2.2.1 [412]. Measurements of message sizes were used to simulate the accounting scheme with the simulator presented in [152].

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 32.1 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.

Account Consistency		System Key Related Operations	
Node Arrival	O(k)	Key Update Calculations	O(t)
Node Departure	O(k)	Key Update Distribution	O(t)

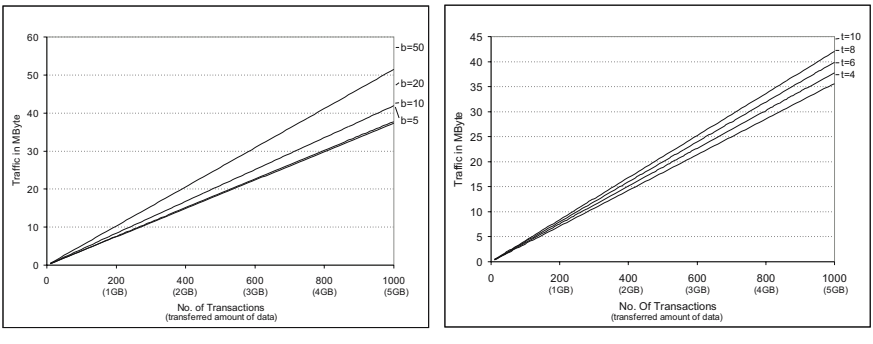
Table 32.1: Account Holder Set & System Key Maintenance Complexity

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:

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

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 32.6 (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 32.6 (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

Fig. 32.6: By Token-Accounting Scheme Generated Traffic

32.5.9 Summary & Conclusions

One of the biggest challenges for a wider deployment of Peer-to-Peer 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 Peer-to-Peer are still maintained.

The presented token-based accounting scheme is flexible and trustworthy. Its basic 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 Peer-to-Peer 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 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.