Cashflow: A Channel-Oriented, Credit-Based Virtual Currency System for Establishing Fairness in Ad-Hoc Networks with Selfish Nodes

Lukas Wallentin, Joachim Fabini, Christoph Egger, and Marco Happenhofer

Institute of Broadband Communications, Vienna University of Technology
Favoritenstraße 9/388, A-1040 Vienna, Austria
Lukas.Wallentin@tuwien.ac.at

Abstract. Cooperation in ad hoc networks cannot be taken for granted when nodes belong to distinct authoritative domains. In this paper we present Cashflow, a virtual currency system to motivate nodes to participate in ad hoc networks and to prevent selfishness. This system is different to previously proposed virtual currency systems in that it uses a channel concept for data transmission as well as a market system for pricing. The combination of channel and market concept results in a number of positive system characteristics. For instance, Cashflow provides implicit access regulation and load balancing mechanisms to avoid overload situations in the network. Additionally it considers node context for data transmission and pricing. Cashflow also leaves the decision about the participation degree to the user, which is an important but frequently neglected topic for the user acceptance of virtual currency systems.

Keywords: virtual currency system, ad hoc network, micropayment, load balancing, credit system, selfish nodes, fairness.

1 Introduction

Ad hoc networks have already been an active research field for many years. Originally, the benefits of wireless networks without the usage of additional infrastructure made ad hoc networks attractive for military usage as well as in emergency situations [1]. Today mobile devices have become part of our daily life, generating new application fields for ad hoc networks, which result in new requirements. For instance, multimedia applications require quality of service support while mandating interoperability with other network types like cellular networks or the Internet [2].

Early ad hoc networks and their associated applications have been designed with the assumption of closed user groups, where all network nodes belong to the same authority and therefore are cooperating in archiving a common goal. However, new issues arise if nodes belong to different authorities. In such scenarios, cooperation cannot be taken for granted and the prevention of selfishness is a new challenge for the design of ad hoc networks.

Currently there are two concepts to prevent selfish behavior and stimulate cooperation in ad hoc networks: reputation systems and virtual currency systems [3]. Reputation systems enforce fair behavior by excluding selfish nodes from the network. Additionally reputation systems stimulate cooperation since nodes can rely on other nodes service as compensation for own contributions. Virtual currency systems on the other hand adapt the “pay for service”-concept to ad hoc networks. Nodes must compensate for other nodes service by paying a fee using credits. A node can either earn credits by providing service to other nodes, or buy them from outside the system. It is worth noting that virtual currency systems not only stimulate cooperation between nodes, but additionally could raise interest in ad hoc networks for a number of business applications.

Virtual currency systems proposed in literature mainly focus on how to pay and on how much to pay for data transmission along a given path. As pricing function, in most cases they use auctions or fixed price schemes, based on game theoretical considerations. These schemes do not consider the context of the node nor adapt to the users needs. However, we argue that the consideration of node context and the users ability to control the degree of participation in the network, is crucial for the acceptance of virtual currency systems as enabler for ubiquitous ad hoc networks.

Focusing on these issues, we present Cashflow, a virtual currency system to motivate nodes to participate in ad hoc networks and to prevent selfishness. This system distinguishes itself from other virtual currency systems by using a channel concept for data transmission as well as a market system for pricing. The combination of a channel and market concept results in a number of positive system characteristics. It gives users control over their participation degree, allows Cashflow to consider the context of nodes and provides implicit load balancing and access control functionality.

The rest of the paper is organized as follows. In Section 2 we discuss related work. Section 3 presents Cashflow and discusses its concepts in detail whereas Section 4 concludes with a summery and an outlook on future work.

2 Related Work

Participation and fair behavior of nodes in ad hoc networks cannot be taken for granted, if nodes belong to different authorities like companies or private users. In general, nodes are not interested in providing services like routing and packet forwarding for other nodes without getting a reward. Hence, mechanisms are needed to prevent selfish behavior and stimulate cooperation. Currently two classes of systems are suggested in literature to achieve this task. The first class of systems are reputation systems as detailed in the introductory section. Several reputation systems have been suggested in literature, including OCEAN [4], CORE [5] and CONFIDENT [6,7].

The second class of motivation systems are virtual currency systems. The main principle behind these systems is that nodes pay relaying nodes a fee in form of (sometimes virtual) money for forwarding packets. The packet purse model of the virtual currency system Nuglets [8] uses this concept. The source node of a packet adds nuglets, the virtual currency used by this system, into packets it wants to transmit to a target node. Intermediate nodes extract nuglets from packets as reward for their service before forwarding them. If there are not enough nuglets left in a packet to compensate for the forwarding service, the packet gets dropped. This leads to the problem that the source node has to estimate the number of nuglets needed for the transmission along a path. If it underestimates the needed amount, the packet will not reach its destination. Otherwise, if it overestimates the fee, the target will receive undeservedly nuglets. To overcome this problem, Nuglets proposes also the packet trade model [8,9] where nuglets are used by nodes to buy packets from previous nodes and sell them for a higher price to next-hop nodes. Consequently, target nodes effectively pay for whole transmissions. A combination of both proposed schemes is possible. As pricing strategy an auction scheme or a fix price strategy is suggested by the authors.

Nuglets has two tradeoffs. First, it needs tamperproof hardware to guarantee that no node extracts more nuglets than owed from packets. Second, it is possible that nuglets get lost due to lost or discarded packets in the system. Sprite [10] solves these problems by introducing a credit system instead of a real virtual currency. In this system, the payment and accounting is performed by a credit clearance service. When a node forwards a packet, it keeps a receipt. Later is sends the receipts to the credit clearance service, which compensates the node. Additionally the credit clearance service determinates the charges based on transmission success. This is done to prevent bogus behavior.

iPass [11,12] uses a credit clearance service for charging and accounting too. The difference between Sprite and iPass is the way they determine the fee. While Sprite uses game theory to determine the charge, iPass uses an dynamic auction scheme for pricing in order to prevent overload situations. In iPass, packets include bids. In overload situations nodes forward only packets with high bids and discard other packets. Over a feedback mechanism from the destination node, source nodes can adjust their bids to gain a higher throughput.

These four virtual currency systems have in common that they are not interconnected with routing protocols. They solely solve the problem how to pay and how much to pay along a given route, but provide no mechanism to transmit packets along the cheapest route between source and target. Commit [13], which is based on the virtual currency system Ad hoc-VCG [14], targets this issue. In Commit the fee nodes charge is based on the energy needed for packet transmission. When a node wants to transmit data to another node a special routing protocol is used to find the cheapest and the second cheapest route. If the fee of the second cheapest route lies beneath a certain value, the source node transmits the packet along the cheapest path but pays the fee of the second cheapest. This is done due to game theoretical considerations [13].

All these virtual currency systems have in common that node context is not considered as well as they do not give user influence on their participation degree. An additional drawback is that none of these systems combine load depending pricing with fee based routing which can yield a load balancing and access control system. The virtual currency system Cashflow, which will be presented in the rest of this paper, addresses these issues.

3 Cashflow

In this section we present the concept of Cashflow, a virtual currency system to motivate nodes to participate in mobile ad hoc networks. This system has been designed to fit the needs of specific usage scenarios which will be described in Section 3.1. The specific requirements of the usage scenarios result in an architecture based on channels for communication, a market concept for pricing and a credit system for charging and accounting. Sections 3.2 to 3.4 discuss these components of the Cashflow system. Section 3.5 provides an usage example.

3.1 Usage Scenarios

Cashflow was designed to encourage nodes, belonging to different authorities, to participate in a common wireless network. Cashflow's concept targets on scenarios like shared office buildings, where the communication infrastructure is temporarily not available or a hotel, where members of a traveler party, accommodated in different rooms, want to exchange pictures using their mobile equipment without leaving their room.

In both described scenarios, communication between different nodes is possible, provided that intermediate nodes, equipped with wireless communication technology, are willing to join a common wireless network and provide services like routing and forwarding for other network users. Additionally it can be assumed that in most cases not all nodes are moving at the same time. Taking the office example, a user might use his laptop in his office for some time, than brings the laptop to a meeting, maintaining its position for the meeting duration, and after the meeting ends, he takes the laptop back to his office. Therefore, this laptop shows a nomadic mobility pattern during a normal work day.

Besides the location, also the context of this laptop changes over time. During usage in the users office, the laptop might run on AC power, while during the meeting, the internal battery is used as power source. While running on battery, the energy consumption is more important compared to the time while running on AC power. Therefore the users interest to participate changes depending on the power source. The energy source might not be the only parameter influencing on the user's willingness to provide services for other users. Since routing and forwarding needs computer resources like processing time and memory, users running resource-intensive applications like games, 3D rendering applications or simulations, needing their computer resources for themselves, might not be interested to provide larger parts of their resources for network services. Therefore it was assumed for the design of Cashflow that it is important to consider the context of the node for the fee calculation.

The described scenarios also allow to make some expectations concerning the traffic. Because of the size of digital images and other digital documents, it is assumed that in many cases data transmission between nodes will include a number of packets and not just a single one. Additionally, protocols like TCP require a number of packet exchanges for the transmission of even a single data bit. Consequently for the development of Cashflow it was assumed that, in contrast to other wireless networks like sensor networks or vehicular ad hoc networks, nodes frequently exchange a sequence of packets within a short time and not only communicate sporadically.

Summarizing, Cashflow was developed with a focus on scenarios where the following assumptions are fulfilled:

1. Network nodes are equipped with wireless communication technology.
2. There is at least one direct or indirect connection between every network node.
3. There is no additional infrastructure or the existing infrastructure cannot be used.
4. Network nodes belong to different users.
5. Nodes can be motivated to provide services for other nodes using some kind of compensation.
6. Network nodes consist of mobile office equipment like laptops, smart phones and personal digital assistants.
7. Nodes are free to move, but most network nodes show a nomadic behavior, meaning that not all nodes are always moving.
8. Typically, nodes transmit a number of packets to other nodes, not just single packets.
9. Nodes are willing to pay a certain maximum fee for services of other nodes.
10. Nodes want to pay as little as possible for services of other nodes.
11. All nodes have sporadic access to the internet, but it is not given that nodes have internet access during their participation in a wireless network.

Additionally Cashflow assumes that some of the following node characteristics might be true:

12. User of nodes have different preferences concerning their participation in networks, independent from the reward they receive for their services.
13. The node's context influences its willingness to participate in wireless networks.

These assumptions have established a framework for the development of Cashflow. Some assumptions like 1 to 3 are essential in all ad hoc networks, since otherwise the deployment of a mobile ad hoc network is not possible or does not make any sense. Other assumptions like 4 and 5 are essential for the integration of a virtual currency system. If all nodes belong to one single authority, cooperation can be taken for granted and therefore a system like Cashflow would be obsolete in such kind of network. This is also true in the case that nodes cannot be motivated to participate using rewarding schemes. The remaining assumptions basically define real life scenarios. Some of these assumptions result in system restrictions, others provide optimization potential. For instance, the use of regular mobile office equipment like laptops and smart phones in such a network results in the requirement that the system should not rely on special hardware like tamperproof hardware. The usage of tamperproof hardware would simplify the integration of virtual currency system from the viewpoint of security. On the other hand, the assumption of a specific mobility and traffic pattern allows to optimize the routing as well as the data transfer for these specific scenarios.

All these assumptions lead to a system which is optimized for a special type of scenario. Obviously Cashflow can also be used in scenarios where some of the assumptions are not true. In this case, some functions of Cashflow might not be used or an additionally overhead might be associated with these functions. For example, in the case that Cashflow is used in a scenario where tamperproof hardware is given, other payment strategies might perform better than the standard system currently integrated into Cashflow.

3.2 Channel Concept

Starting point of Cashflow's channel concept is that in many cases nodes exchange a number of packets consecutively and not just sporadically single packets. The basic idea of the channel concept is that if a node wants to transmit a number of packets, it opens a channel to the destination node. It is important to note that channels in Cashflow are bidirectional, meaning that the source node, the initiator of the channel, can transmit data to the destination node and vice versa.

A channel is characterized by two parameters. First parameter is the duration. A channel is open for a certain time during which source and destination nodes can exchange data. After elapse of this time the channel is closed and the source node must open a new channel if it wants to communicate with the destination node again. However, the source node, and only the source node, can request an extension of the duration time, which might be granted by the intermediate nodes.

Second parameter is throughput. It specifies how many packets will be transmitted per second through the channel. Since packets have a defined maximum size, intermediate node can estimate the maximum bandwidth a channel needs. Consequently nodes can reject channel requests, if the needed bandwidth exceeds the available one and therefore avoid overload situations. Besides the avoidance of overload situations, by using this channel concept, nodes can regulate their maximum throughput depending on their current situation as detailed in assumptions 12 and 13.

Not only intermediate nodes benefit from the channel concept. As soon as a node has successfully opened a channel, it can rely that the connection to the destination node will not be interrupted abruptly because of upcoming traffic of other nodes. In other virtual currency systems it might happen that an intermediate node stops forwarding packets on behalf of a node, because another node requests its services and is willing to pay a higher price for the service than the node is currently paying. Even if the node whose transmission has been interrupted continues its transmission by interrupting traffic from other nodes likewise, still some packets gets dropped as consequence of the interruptions. Besides the additional delay caused by the retransmission of dropped packets by the source node, the source node must pay all intermediate nodes along the path to the node which has dropped the packet. The reason is that these intermediate nodes have forwarded the packet and therefore earned a reward, even if the packet did not reach the target. (Compare with Sprite[10], Nuglets [8]) Therefore, we argue that avoidance of such interruptions by means of Chashflow's channel concept matches nodes requirements.

For the establishment of channels it is important that routing protocols support the detection of stable routes. In Cashflow, nodes rate their links to neighbor nodes based on their duration and on the signal to noise and interference value. Since a nomadic mobility pattern is assumed, strong links with a long duration are preferred for channels compared to weak and sporadic links, because of the lower path break probability. By using a modified version of the route discovery algorithm suggests in [15] it is possible to find the cheapest route between two nodes which additionally fulfills restrictions concerning the rating of links.

3.3 Pricing Scheme

The channel concept influences the way nodes determine their service fee. Cashflow uses a market concept to calculate fees charged by nodes for forwarding packets. The basic idea is that nodes continuously adapt the fee they charge for the so called standard channel, depending on supply and demand. The standard channel is a channel with a fixed throughput and duration value, known to every node. Based on the price of the standard channel, a node can derive the price of any other channel with any parameterization. When a node wants to open a channel along a given path, it sends a channel request packet including information about the channel along the path to the destination. The destination node replies with a channel response packet. Intermediate nodes include the value of the fee they would charge for the requested channel into the response packet, so that the original source nodes finally gets the information about the channels costs. Based on the channels fee and the price the source node is willing to pay, it can either accept the offer and open the channel or reject it. By adjusting the price, nodes can make themselves, respectively their routing service, more or less attractive for other nodes.

An alternative point of view is that nodes can express by the fee they charge their willingness to host additional channels. Given that each node has a certain preferred throughput value. If the current throughput is significantly higher than the preferred throughput the node could increase the fee with the result that some of the open channels might not be extended by the channel's source node after the durations time expiration. In addition the probability increases that nodes requesting a channel will reject the channel after receiving the offer because of the high price. Therefore by increasing the fee for channels, nodes can reduce the throughput. The opposite is true when a node decreases its fee. In this case, channels which use this node become cheaper and therefore more attractive for other nodes.

Cashflow additionally uses the fee as instrument for load balancing. Since nodes increases their prices in high load situations, channels through high loaded network segments are more costly than through parts with low load. Assuming that nodes prefer to get a service for the lowest possible fee, nodes are interested to open channels using the cheapest path to the destination. Consequently they avoid highly loaded parts of a network if possible. This functionality requires a route discovery protocol, able to find the cheapest route from a source to a destination route. Such a protocol has been developed as part of Cashflow and is presented in [15]. However Cashflow can also be used in combination with other routing protocols, but might result in a performance penalty of the load balancing functionality.

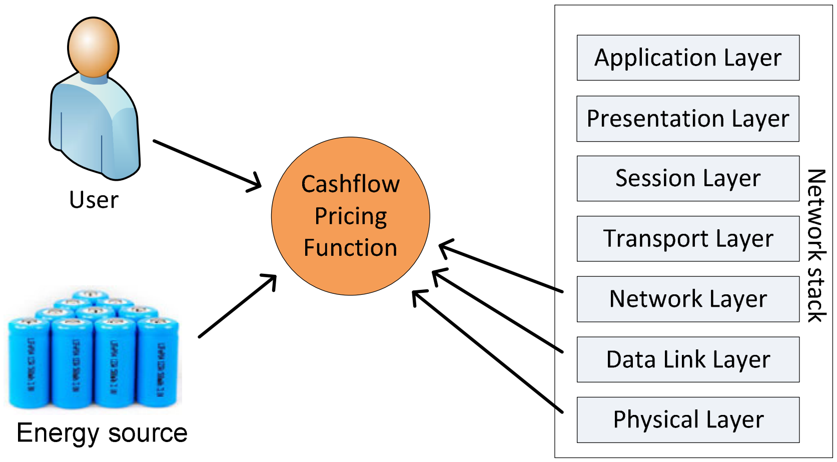

Fig. 1. Data-sources for price calculation used by Cashflow

Besides load balancing, the combination of the channel with the market concept can also be used to prevent overload situations of the shared medium. When nodes receive a channel request they estimate the additional usage of the shared medium based on the channel's parameters. If the additional usage would cause an overload, they charge an infinite high price, which corresponds to a reject. Additionally, it increases the fee it would charge for a standard channel, since such a reject indicates an imbalance of supply and demand. Similar to this overload prevention, Cashflow allows each node to define a maximum throughput. If a node receives a channel request and its parameters indicate that the additional throughput caused by this channel would exceed the maximum throughput, again the node can reject the request by charging an infinite fee. Using this mechanism, the user keeps control about his participation degree.

In short, a number of parameters influences the node's fee. Figure 1. gives an overview of the sources currently considered. It is worth noting that additional sources could be integrated into the pricing function. As pictured in Figure 1, the user of a node is a source of parameters for the fee calculation. The user defines the minimum fee a node charges for forwarding packets. Additionally the preferred and the maximum throughput is defined by the user as well as the allowed variance of the throughput before the pricing function adapts the current fee. Moreover, the node' user configures the maximum usage of the shared medium. The last two parameters the user specifies is the so called minimum and maximum battery penalty, which is added to the fee whenever the node runs on battery. The actual amount of the penalty depends on the battery's charging level. If the battery is fully charged, the minimum battery penalty is added to the fee. As the charge of the battery decreases the penalty increases until it reaches the maximum penalty when the battery is nearly empty.

To realize such a charge dependent penalty, the pricing function needs to use the energy source as input as shown in the figure. The result of the integration of the energy source as parameter in the fee calculation is that nodes using battery are less attractive as relay node for other nodes, especially is the battery penalty is high because of low charge, compared to AC powered nodes.

The next source pictured in Figure 1 is the node's physical radio interface layer. It provides two parameters for the pricing function. First it delivers information about the transmission rate used for data transmission between two nodes. The pricing function uses this information to charge a higher fee for transmissions over links with low bit rate than over faster links. We argue that this is in the interest of the whole network since when a higher bit rate is used, the shared medium is occupied for a shorter time than if a low bit rate is used for the transmission of the same data amount. The second input is the average signal to noise and interference (SNIR) ratio from signals from neighbor nodes. This value is used to estimate the link quality between the nodes. Links with weak quality are charged higher, since the probability that retransmissions are needed is higher than for strong links.

The MAC-layer is an additional input source for the pricing function. By collecting information about the shared channels usage [16,17], the pricing function can react if the preferred usage rate is in average exceeded for a certain time. If the average usage lies above the maximum usage of the shared medium, as defined by the user, the pricing function increases its fee. Additionally a node can use this information to reject channel requests if it is expected that the additional load caused by the requested channel would result in an overload of the shared medium. Therefore the usage of the information provided by the MAC-layer results in an overload protection, since nodes in high load areas of a network tent to increase their fee with the consequence that routes through high load areas become less attractive for other nodes.

The last input source in the current version of Cashflow is the network layer where the channel functionality of Cashflow is implemented. As stated above, the node's user can configure the preferred and the maximum throughput, as well as the tolerated throughput variance. The network layer provides information about the number of currently open channels as well as the resulting throughput for the pricing function. Using this information, the pricing function can adapt the fee according to the difference between preferred and actual throughput.

The fee itself is calculated by two functions. The first function calculates the basic fee a node charges. It continually updates this fee depending on the current throughput and the shared mediums usage. Using this basic fee, another function calculates the fee a node charges for specific channels, depending on channel parameters and the nodes context. The following code describes the functionality of the function which continually adapts the base price:

```
procedure continuesBasicFeeCalculation()
{
  global basicfee=getMinimumFee();
  while(TRUE)
  {
    sleep(1000);
    if(getCurrentTP() > (preferredTP+TPVariance))
       basicfee=basicfee + feeIncrease;
8:      if(getCurrentTP < (preferredTP-TPVariance))
9:         basicfee = basicfee - feeDecrease;
10:     if(wasBlocked == TRUE){
11:        basicfee = basicfee + largeFeeIncrease;
12:        wasBlocked = FALSE;}
13:     if(getCurrentMediumUsage > maximumMediumUsage)
14:        basicfee = basicfee + largeFeeIncrease;
15:     if(basicfee < getMinumumFee())
16:        basicfee = getMinimumFee();
17:    }
18: }
```

At the start of this function, it sets the global variable *basicfee* to the minimum fee the user decided to charge for forwarding packets along a channel. Then the function enters a loop. After waiting for some time, in this example for 1000 milliseconds, it updates the price. In line 6 to 9 of the code, the function checks if the current throughput lies above or beneath the preferred throughput, including the accepted variance and increases or decreases the value of the basic fee accordingly. In line 10, the function checks if a request for a channel was rejected because the additional load would have exceeded the maximum throughput. If so, the basic fee is increase by a value which is larger than the normal increase. In line 13 the algorithm verifies that the current usage of the shared medium is below the maximum usage as preferred by the user. If the usage is higher, the fee is increased by the same value as if a channel has been rejected. After all these fee modifications, line 15 verifies that the new fee is not smaller than the minimum fee. This can happen if there is a low load situation over a longer time in the network. If the new fee would undercut the minimum fee, it is set to the value of the minimum fee. Summarizing, this function adapts the fee depending on supply and demand.

In a next step, the value of the global variable *basicfee*, calculated by the *continuesBasicFeeCalculation* function, is used by the offering function to calculate a concrete offer for a channel request. The basic fee represents the fee for the already described standard channel, if no additional penalty if charged. The offering function receives four parameters: two channel parameters representing the duration and size, in terms of packets per second, relative to the standard channel and the IDs of the next and the previous node, as it can be seen in line 0 of the following code:

```
0: procedure getOffer(size, duration, prevHop, nextHop)
1: {
2:   fee = basicfee * size * duration;
3:   if(batteryPowerd == TRUE){
4:     fee = fee + minBatPenalty;
5:     toMax = (maxBatPenalty-minBatPenalty);
6:     fee = fee + toMax - toMax * charge; }
7:   if(prevHop.speed == SLOW)
8:     fee = fee * speedPenalty;
9:   if(nextHop.speed == slow)
10:    fee = fee * speedPenalty;
11:  if(prevHop.SNIR == low)
12:    fee = fee * signalPenalty;
13:  if(nextHop.SNIR == low)
14:    fee = fee * signalPenalty;
15:  if((expectedTP(size)) > maxTP){
16:    wasBlocked = TRUE;
17:    fee = INFINITE; }
18:  if((expectedMediumUsage(size)) > maxMediumUsage){
19:    wasBlocked = TRUE;
20:    fee = INFINITE; }
21:  if(duration > maxDuration)
22:    fee = INFINITE;
23:  return fee;
24:}
```

Source code line 2 calculates the fee for the requested channel, depending on its size, in terms of packet throughput per second, and duration. The variable *basicfee* contains the value calculated by the *continuesBasicFeeCalculation* function described before. In line 3 to 14 different penalties are added to the fee. The first penalty is the battery penalty. Nodes running on battery add at least a minimum battery penalty in case their battery is fully charged. However during the time the charge will decrease and the node will as a consequence increase the penalty. Line 7 to 10 add extra penalties to the fee if the radio interface of the previous or the next node only allows slow communication. For example, this would be the case if a node supports the IEEE802.11g standard but its neighbors are equipped with IEEE802.11b technology. In the provided code we only distinguish between slow and fast transmission speeds. However, in a real implementation a number of different levels would be used. Similar the rating of the signal quality. Again, in the given code we only differentiate between strong and weak signals and not between a number of signal quality levels. The basic idea is to add a penalty if only a weak connection to the neighbor is given, since the node might needs to perform several retransmissions for transferring data over the weak link. In line 15 the node estimates the cumulative throughput of existing and requested channels and verifies that the result lies beneath the maximum preferred throughput as configured by the user. If the estimated throughput exceeds the preferred one, the node rejects the request by setting the channels fee to infinite. Additionally it signals the function calculating the basic fee that a request has been rejected by setting *wasBlocked* to true, resulting in an increase of the basic fee. The algorithm likewise attempts to estimate the impact of the requested channel on the shared mediums usage. If the new channel would cause a local overload situation the channel is rejected. Before returning the calculated offer, the algorithm verifies that the requested channels duration does not exceed the maximum duration the node accepts.

Summarizing, the pricing function adapts the fee a node charges depending on supply and demand and adds penalties to the fee if required by the node's context.

Nodes charging high fees are not attractive for other nodes as relay and consequently will be avoided. Therefore nodes can influence their throughput by adapting their fee. The pricing functions presented in this section should be taken only as an example. It is possible to implement a number of different pricing strategies into the system. The only requirement is that the price increases when demand overpasses supply.

3.4 Credit System

For the actual payment, Cashflow uses a credit system, where nodes use a kind of virtual credit card to pay rewards for used services. It can be implemented completely in software and consequently does not rely on special tamperproof hardware. Therefore also other virtual currency systems like Sprite and iPass uses similar credit systems for payment.

Figure 2. shows the basic concept behind the payment system, which is based on public key infrastructure. It is assumed that each node has a bank account and that there has already been a secure key exchange between the bank and the nodes. To use the services of other nodes, a node needs a bank credit certificate which acts as virtual credit card. To gain such a certificate, the node sends a signed request to the bank including the account number and its public key as pictured in step 1 in Figure 2. If the node is creditworthy, the bank issues a certificate which includes the account number, the node's public key and an expiration date. This certificate is signed by the bank so that each other node can verify that the credit certificate is valid and will therefore be accepted by the bank. As soon as a node has its credit certificate it can join an ad hoc network using Cashflow. The node does not need to have a connection to the bank server to use the credit certificate. The connection to the bank is only needed for issuing the certificate and for accounting. Since a credit certificate could be valid for some weeks or even months, nodes only need sporadic access to the bank.

When a node requests a channel (step 3 in Figure 2) along a given path, it includes its credit certificate into the request packet to prove its creditworthiness. When the request reaches the destination node, the destination node replies to the request by sending a response packet. This response packet, step 4 in Figure 2, is transmitted back along the same path to the source node. While forwarding the response packet,

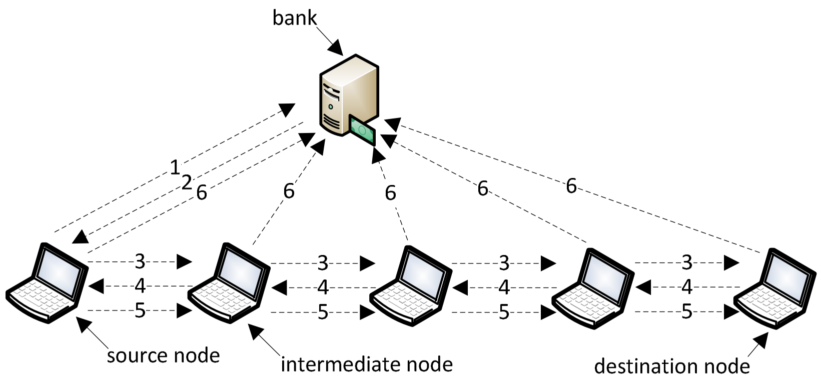

Fig. 2. Credit concept used by Cashflow

each intermediate node inserts a hash code, generated out of the request packet, the source node's account number, the channel ID and the fee it charges for the channel as a signed message. These signed fields represent the offer of an intermediate node. Such an offer also can be seen as a kind of contract which is currently only signed by intermediate nodes. When the source node receives the response, it can calculate the complete price for the channel by summing up all the fees intermediate node charge. If the channel is too expensive for the node, it can drop the response and consequently no channel will be established. If the source node wants to use the channel, it has to sign the offer of each intermediate node.

In step 5 the source node sends the signed offers back to the intermediate nodes, which they transmit in step 6, as soon as they have access to the bank, to get the money transferred from the source node's account to their account. Besides the offer signed by the source and the intermediate node, all the nodes, including the source and the destination node, send statistical channel data to the bank. This statistical data include information about the actual number of packets received by the node. Using this information, the bank can detect malicious nodes and exclude them from the network or make fraud unattractive for the nodes by paying only fractions of the fee, as suggested in [10].

As already stated above, such credit systems are also used by other virtual currency systems. A detailed analyses of security issues in such systems can be found in [12]. The difference of the credit system used by Cashflow compared to other systems is that the source nodes pays for a channel, along which it can transfer a number of packets, and not for each single packet. The benefit of this concept is a reduction of the credit system's overhead since the nodes does not have to keep a receipt for every packet. Additionally the price of a channel is known to nodes before they actually open the channel, allowing them to reschedule transmissions.

3.5 Putting Everything Together

After the description of the different concepts used by Cashflow, this section briefly introduces the interplay between all the components by providing an usage scenario. To enable nodes to participate in ad hoc networks using Cashflow, they first must acquire a certificate from the bank. As soon as the bank has issued the certificate, nodes can participate in ad hoc networks until their certificates expire without having a connection to the bank. All nodes within an ad hoc network periodically update the basic fee they charge, based on supply and demand of their forward service. Additionally they monitor the quality of wireless links to other nodes.

When a node needs to transmit data to another node which is not a direct neighbor, it might have to perform a route search to find the cheapest route to the target node, whereby restrictions concerning link quality must be considered. For the detection of the cheapest route, the route discovery algorithm not only considers the basic fee nodes charge but also penalties caused by the context of intermediate nodes. Provided the target node is part of the network, the route search results in at least one path to the target node. Additionally the node receives information about the price level of different nodes along discovered routes, which they can compare with prices received from previous route discoveries.

Using routing information, a node can request a channel along discovered paths. The request results in an offer for the channel or a reject, if nodes along the route would be overloaded by the channel. The requesting node can decide to either accept the channel or reject it, if the fee is too high. When the channel is opened, the source node can exchange data with the target node until the channel expires. However, the source node can request an extension of the channel if it wants to continue to exchange data. For the charging, intermediate nodes keep receipts of the channels.

As soon as a node can connect to the bank server, it transmits its receipts to transfer the earned fees to its account. Additionally it transmits statistical data which is used for the detection of malign nodes. Depending on the balance of the bank account, the user might have to transfer money from an external source to the bank account, or the bank disburses the money to the node. This is important, since the bank will not issue new certificates to users nodes after the expiration of old certificates if the users account has been overdrawn.

4 Conclusion

To the best knowledge of the authors, Cashflow presents the first attempt to realize a virtual currency system where the decision on the participation degree is left to the users. Additionally Cashflow is sensitive to node context, allowing to reflect the willingness of nodes to forward packets by the fee they charge. This was realized by using a market concept for pricing where the node fee is determined by the difference of supply and demand, as well as by user and context specific parameters. A credit system was introduced for charging and accounting which makes Cashflow, in contrast to other virtual currency systems, independent of tamperproof hardware. An additional benefit of the credit system is its ability to detect malign nodes and exclude them from the system, making misbehavior in the network unattractive.

In contrast to other virtual currency systems, Cashflow uses a channel concept for data transmission. This concept allows to inform nodes about transmission fees before they start to transmit data. Using this information, nodes can suspend transmissions in the case of temporarily high transmission fees. Hence users have full control concerning transmission costs which is not a matter of course. By combining the market based pricing system with the channel concept of Cashflow and a specifically designed routing algorithm [15], which allows to find the cheapest path between nodes, the system provides an implicit access control and load balancing algorithm.

Concerning future work, Cashflow was mainly designed to give users control with respect to their participation degree and to include node context into the pricing. Load balancing and access control functionality can be seen as a kind of positive side effect of the system. Therefore the efficiency of load balancing and access control mechanisms have not been yet completely evaluated and not optimized. However, first simulation results as pictured in Figure 3 and 4 are promising. Both figures show heat maps of a grid network with 32 to 32 nodes, whereas nodes within the area with the dotted black border are inactive and only used for scanning purposes. In both cases four nodes on the left side try to transmit data to four nodes on the right side. The heat maps show network activity whereas red areas indicate overload situations. In the left scenario the pricing function is deactivated which corresponds to an

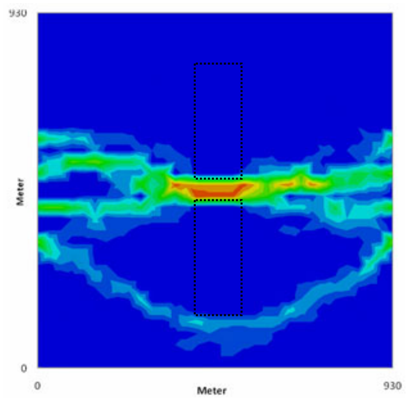

Fig. 3. No load balancing

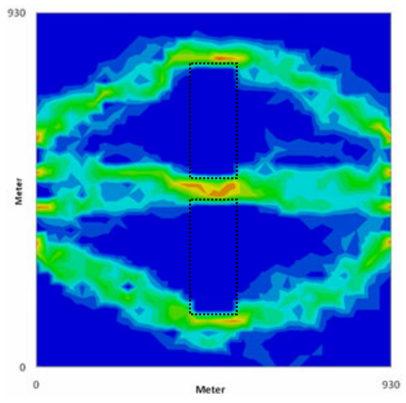

Fig. 4. Load balancing

deactivation of the load balancing function. In most cases the nodes try to use the shortest way to the target nodes resulting in an congestion between the two areas of inactive nodes. If the pricing function is activated, the high price makes paths through the bottleneck unattractive and result in a prevention of local overload, as pictured in Figure 4. Additionally the complete throughput increases by 92.7 percent in this scenario because of the usage of alternative paths.

Besides the evaluation and optimization of the access control and load balancing system, the integration of Cashflow in hybrid and internet connected networks and the extension of the system to a platform for additional paid services will be part of future research.

References

1. Perkins, C.E.: Ad Hoc Networking. Addison-Wesley Longman, Amsterdam (2001)
2. Cavalcanti, D., Agrawal, D., Cordeiro, C., Xie, B., Kumar, A.: Issues in integrating cellular networks WLANs, AND MANETs: a futuristic heterogeneous wireless network. IEEE Wireless Communications 12(3), 30–41 (2005)
3. Misra, S., Woungang, I., Misra, S.C.: Guide to Wireless Ad Hoc Networks, 1st edn. Springer, Berlin (2009)
4. Bansal, S., Baker, M.: Observation-based Cooperation Enforcement in Ad Hoc Networks. Technical report, Computer Science Department, Stanford University (2003)
5. Michiardi, P., Molva, R.: CORE: a Collaborative Reputation mechanism to enforce node cooperation in mobile ad hoc networks. In: Proceedings of the Communication and Multimedia Security 2002 Conference (2002)
6. Buchegger, S., Boudec, J.-Y.L.: Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In: 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (2002)
7. Buchegger, S., Boudec, J.-Y.L.: Performance analysis of the CONFIDANT protocol: Cooperation of nodes - fairness in dynamic ad-hoc networks. In: Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC). IEEE, Los Alamitos (2002)

8. Buttyan, L., Hubaux, J.P.: Enforcing service availability in mobile ad-hoc WANs. In: Proceedings of the First ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc) (2000)
9. Buttyan, L., Hubaux, J.P.: Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. In: Tech. Rep. DSC/2001/001, Swiss Federal Institution of Technology (2001)
10. Zhong, S., Yang, Y.R., Chen, J.: Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad Hoc Networks. In: Proceedings of INFOCOM. IEEE, Los Alamitos (2003)
11. Chen, K., Nahrstedt, K.: iPass: An Incentive Compatible Auction Scheme to Enable Packet Forwarding Service in MANET. In: 24th IEEE International Conference on Distributed Computing Systems (ICDCS 2004), pp. 534–542 (2004)
12. Chen, K.: Cooperative and non-cooperative flow control in mobile ad hoc networks, PhD-Thesis, University of Illinois at Urbana-Champaign, Urbana (2004)
13. Eidenbenz, S., Resta, G., Santi, P.: Commit: A sender-centric truthful and energy-efficient routing protocol for ad hoc networks with selfish nodes. In: Proc. of 5th IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc & Sensor Networks, IPDPS (2005)
14. Eidenbenz, S., Anderegg, L.: Ad hoc-VCG: A truthful and cost- efficient routing protocol for mobile ad hoc networks with selfish agents. In: Proc. ACM MobiCom, pp. 245—259 (2003)
15. Wallentin, L., Fabini, J., Egger, C., Happenhofer, M.: A Cross-Layer Route Discovery Strategy for Virtual Currency Systems in Mobile Ad Hoc Networks. In: Proceedings of the Seventh International Conference on Wireless On-demand Network Systems and Services (WONS 2010), Kranjska Gora, Slovenia, pp. 91–98 (2010)
16. Davis, M.: A Wireless Traffic Probe for Radio Resource Management and QoS Provisioning in IEEE 802.11 WLANs. In: ACM Symposium on Modeling, Analysis and Simlulation of Wireless and Mobile Systems (ACM MSWiM 2004), Venice (2004)
17. Davis, M., Raimondi, T.: A Novel Framework for Radio Resource Management in IEEE 802.11 Wireless LANs. In: Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2005), Trentino (2005)