Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks[*]

Levente Buttyán and Jean-Pierre Hubaux
Institute for Computer Communications and Applications
Department of Communication Systems
Swiss Federal Institute of Technology – Lausanne
EPFL-DSC-ICA, CH-1015 Lausanne, Switzerland

18 January 2001

Abstract

In mobile ad hoc networks, it is usually assumed that all the nodes belong to the same authority; therefore, they are expected to cooperate in order to support the basic functions of the network such as routing. In this paper, we consider the case in which each node is its own authority and tries to maximize the benefits it gets from the network. In order to stimulate cooperation, we introduce a virtual currency and detail the way it can be protected against theft and forgery. We show that this mechanism fulfills our expectations without significantly decreasing the performance of the network.

1 Introduction

A mobile ad hoc network is a wireless network formed by some nodes in a self-organized way without relying on any established infrastructure or centralized administration. Since *all* services essential to the operation of the network are assumed to be provided by the nodes themselves, the functioning of mobile ad hoc networks heavily depends on the cooperative behavior of their nodes. Communication between two distant nodes A and B, for instance, relies on intermediate nodes that forward packets for the benefit of A and B.

So far, applications of mobile ad hoc networks have been envisioned mainly for crisis situations (e.g., in the battlefield or in rescue operations). In these applications, all the nodes of the network belong to a single authority (e.g., a single platoon or rescue team) and have a common goal. For this reason, the nodes are naturally motivated to cooperate.

However, with the progress of technology, it will soon be possible to deploy mobile ad hoc networks on a large scale for civilian scenarios as well. Example applications are networks of cars, and provision of communication facilities in remote areas. In these networks, the nodes typically do not belong to a single authority (at the very limit, each node belongs to a different authority: to its user) and they do not pursue a common goal. In addition, these networks could be much bigger and have a longer lifetime, and they could be completely *self-organized*, meaning that the network would run solely by the operation of the end-users. In such networks, there is no good reason to assume that the nodes cooperate and provide services to each other. Indeed, the contrary is true: service provision is not in the interest of the nodes, because it consumes energy and it does not have any direct advantages. Note that the nodes of mobile ad hoc networks are often battery powered, and thus, energy is a precious resource that they may not want to waste for the benefit of other nodes.

[*] Technical Report DSC/2001/001, Swiss Federal Institute of Technology – Lausanne, Department of Communication Systems

Lack of cooperation may have fatal effects on the operation of the network. As an example, let us consider Figure 1, which illustrates that the throughput of the network decreases dramatically as the percentage of the nodes that deny packet forwarding increases. The different curves belong to networks of different sizes (100, 200, 300, and 400 nodes) but with the same node density. The figure also shows that larger networks (those that we are interested in) are more vulnerable to this kind of misbehavior by the nodes. These results are based on our own simulations described in detail in Section 4, but similar results have been presented in [11] as well.

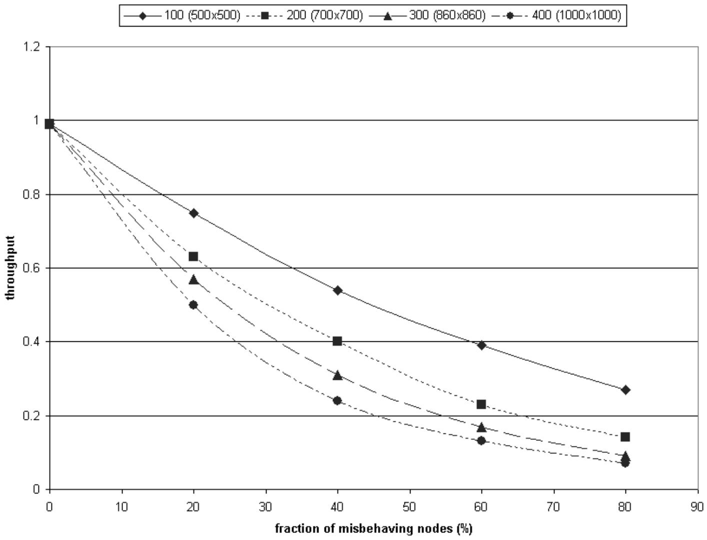

Figure 1: The effect of non-cooperating nodes on the throughput

In this paper, we are concerned with the problem of non-cooperating nodes in large, self-organized, mobile ad hoc networks for civilian applications. We assume that each node belongs to a different authority (user), which has full control over the node. In particular, the user can tamper with the software and the hardware of the node, and modify its behavior in order to better adapt it to her own goals (e.g., to save battery power). We understand that regular users usually do not have the required level of knowledge and skills to modify their nodes. Nevertheless, our assumption is still reasonable, because criminal organizations can have enough interest and resources to reverse engineer a node and sell tampered nodes with modified behavior on a large scale.

On the other hand, we assume that users are not interested in altering the low level protocols of their nodes. These protocols are essential to be able to participate in the network; their modification would certainly disturb the operation of the network, but it would not provide too many advantages for the user of the node. To be more precise, we assume that the physical and data link (MAC) layer of the nodes function correctly, and thus, the nodes are able to directly communicate with other nodes in their power range. However, we assume that users can modify all other layers, including the network layer.

This results in a network where nodes are "selfish": they tend to use services provided by other nodes, but not to provide services for free to the community. In such a network, it is essential to stimulate the nodes to cooperate. In this paper, we propose an economic approach for this purpose. We introduce a virtual currency, called *nuglets*, and mechanisms for charging/rewarding service usage/provision. Nodes that use a service must pay for it (in nuglets) to nodes that provide the service. This makes nuglets indispensable for using the network, and thus, each node is interested in increasing its stock of nuglets. A way to achieve this is to provide services to other nodes[1].

[1] Similar to money in real life, nuglets can be lost. This loss has to be compensated somehow, otherwise the

Besides stimulating for the provision of services, this mechanism also encourages the users of the nodes to make a moderate usage of the network, and to keep their devices turned on (so that they can be used as relays) even if they are not expecting any packet to be sent to them at that time.

In this paper, we focus on the application of nuglets to stimulate the provision of a specific service: packet forwarding. The present proposal was developed in the framework of the Terminodes Project [9, 8, 18]. However, it is generic, and in particular, it could work in conjunction with any routing algorithm. The results presented here are follow-ups of the results of an earlier paper [3]. We considerably improved the mechanisms proposed in [3], removed some constraining assumptions, included an economic model of the nodes, and performed appropriate simulations that demonstrate that the introduction of the nuglets does not significantly decrease the performance of the network.

The outline of the paper is the following: In Section 2, we introduce charging models for the packet forwarding service. One of the models is discussed further in Section 3, where we propose some extensions to it. In Section 4, we describe our simulations and present the results that we obtained. We propose security mechanisms to protect the investigated charging model in Section 5, and report on related work in Section 6. Finally, in Section 7, we conclude the paper.

2 Basic charging models for packet forwarding

Packet forwarding is a service provided by intermediate nodes to the source and the destination of the packet. Therefore, either the source or the destination should pay for it. In this section, we present two conceptual models for charging for the packet forwarding service. In the first one, called *Packet Purse Model*, the source of the packet is charged, whereas in the second one, called *Packet Trade Model*, the destination is charged. The two models can also be combined to provide a hybrid solution.

2.1 The Packet Purse Model (PPM)

In this model, the source of the packet pays for the packet forwarding service. The service charge is distributed among the forwarding nodes in the following way: When sending the packet, the source loads it with a number of nuglets sufficient to reach the destination. Each forwarding node acquires some nuglets from the packet that covers its forwarding costs. The exact number of nuglets charged by the forwarding nodes may depend on many things, including the amount of energy used for the forwarding operation, the current battery status of the forwarding node, and its current number of nuglets. If a packet does not have enough nuglets to be forwarded, then it is discarded.

The main advantage of this model is that, besides stimulating cooperation, it may also deter nodes from sending useless data and overloading the network. Prevention of overloading is an important issue, since, as mentioned in [7], the available bandwidth per node declines when the number of nodes increases (assuming that the traffic does not exhibit locality properties).

The main disadvantage is that it is difficult to estimate the number of nuglets that are required to reach a given destination. If the source under-estimates this number, then the packet will be discarded, and the source loses its investment in this packet. Therefore, the source should over-estimate the number. The surplus that remains in the packet when it arrives to the destination can be kept by the destination, or if the destination provides information services, it can be used to pay for these. However, the source should be careful with the over-estimation, because packets (with nuglets inside) can be lost during transit due to several reasons (e.g., buffer overflow at forwarding nodes).

system becomes poorer and poorer. One way to solve this problem is to let users buy nuglets for real money. Nuglets can be created by international treaty organizations and their agencies. This would mean that providing services is, actually, not the only way to earn nuglets. However, it can be made the preferred way by appropriately choosing the price of one nuglet.

2.2 The Packet Trade Model (PTM)

In this approach, the packet does not carry nuglets, but it is traded for nuglets by intermediate nodes. Each intermediary "buys" it from the previous one for some nuglets (except for the first intermediary that receives the packet for free from the source), and "sells" it to the next one (or to the destination) for more nuglets. In this way, each intermediary that provided a service by forwarding the packet, increases its number of nuglets, and the total cost of forwarding the packet is covered by the destination of the packet. If the next intermediary does not want to buy the packet for the given price, then the forwarding intermediary may try to sell it for a lower price, or to another intermediary, or it may drop the packet.

An advantage of this approach is that the source does not have to know in advance the number of nuglets required to deliver a packet. Furthermore, letting the destination pay for the packet forwarding makes this approach applicable in case of multicast packets as well. A serious disadvantage is that this approach for charging does not directly deter nodes from overloading the network.

2.3 A hybrid model

The previous two models can be combined in the following way: The source loads the packet with some nuglets before sending it. The packet is handled according to the Packet Purse Model until it runs out of nuglets. Then it is handled according to the Packet Trade Model until the destination buys it.

This approach combines the advantages of the Packet Purse and Packet Trade Models: First, since the source still has to pay, it will try to avoid sending useless traffic and overloading the network. Second, the source can under-estimate the number of nuglets that it puts in the packet, because the packet is not discarded when it runs out of nuglets.

2.4 Protection of the models

Clearly, the models described above must be secured and protected against various attacks. An important general problem to solve is the prevention of nuglet forgery. Further problems in the Packet Purse Model include the protection of the packet purse from illegal modifications during transit; preventing the detachment of a packet purse from its original packet and the re-use of it with another packet; and ensuring that the forwarding node receives the service charge if, and only if, it really forwards the packet. A similar problem in the Packet Trade Model is to ensure that the forwarding node receives the payment from the next hop if, and only if, the next hop receives the packet. We will address most of these problems in Section 5. For now, let us assume that these problems are solved and the models are protected in some way or another.

3 Extensions to the basic Packet Purse Model

In the rest of the paper, we are exclusively concerned with the Packet Purse Model. In the presentation of the basic model, we left open the difficult question of how to control the number of nuglets that are taken out from the packet by the forwarding nodes. We assume that a forwarding node can determine the minimum number of nuglets it needs to forward the packet. Ideally, this number should depend on the amount of energy to be used for the forwarding operation (i.e., nodes that forward the packet to a more remote neighbor should acquire more nuglets), the status of the battery at the forwarding node (i.e., nodes which have low battery may esteem battery power higher and should acquire more nuglets as a compensation for the energy used), and maybe some other factors. Clearly, the node should demand this or a higher amount of nuglets from the packet. But what prevents a node from demanding the whole content of the packet purse of a packet? In order to solve this problem, we propose the following two extensions to the basic Packet Purse Model:

- **PPM with Fixed Per Hop Charges:** A simple approach is the following: The source of the packet determines a fixed per hop charge u before sending the packet, and puts u in the packet besides the nuglets. A protection mechanism (discussed later in Section 5) ensures that each forwarding node acquires exactly u nuglets for the forwarding operation. Thus, in this approach, the reward for packet forwarding does not depend on the amount of energy used, the status of the battery, or any other factor. However, the forwarding node is allowed to drop the packet if it estimates that forwarding would not be beneficial for u nuglets.

- **PPM with Auctions:** In this extension of the Packet Purse Model, each forwarding node (and also the source) runs a sealed bid second price auction to determine the next hop. The bidders are the potential next hops towards the destination of the packet. Each bidder b_i determines a price p_i, for which it is willing to forward the packet, and sends it to the forwarding node in a sealed form. When the forwarding node receives all the bids, it determines the winner of the auction. The winner is the bidder b_j with the lowest bid[2] $p_j = \min_i p_i$ (if there are more, then one is chosen randomly or according to some other criteria). Let the second lowest bid be $p_k = \min_{i,\ i \neq j}\ p_i$. The forwarding node puts the value of p_k in the packet, and sends it to b_j. A protection mechanism (discussed later in Section 5) ensures that b_j acquires p_k nuglets ($p_k \geq p_j$) if it forwards the packet further.

  A remarkable feature of the second price auction is that if a bidder values the object on sale at v, then it is a dominant strategy for him to bid exactly v [6]. In our case, this means that each (rational) bidding node is encouraged to bid the smallest price for which it is still worth for it to forward the packet (i.e., to behave correctly and demand a honest service charge). We should note that here we assume that the bidders do not collude and they have no information about the total number of bidders participating in the auction. In particular, a bidding node does not know whether it is alone or there are other bidders around competing for the same packet.

The two approaches described above address the same problem of controlling the number of nuglets that forwarding nodes can take out from the packet in different ways. In the first approach the source fixes this number before sending the packet. In the second approach, this number is determined hop-by-hop by running a second price auction among some neighbors of the forwarding node.

The first approach has two main advantages: first, it is very simple to implement; and second, it is generic, and it can easily be added to any existing routing algorithm, such as DSR [10] or AODV [13]. A disadvantage is that it is not very flexible: if there is a forwarding node on the route that demands more nuglets than the fixed per hop charge in the packet, then the packet is dropped, although it may contain much more nuglets than the demand. We note, however, that in many ad hoc routing algorithms there is a route discovery mechanism, which could be used to obtain information about the "greediness" of the nodes on various possible routes towards the destination. This may help the source to fix a per hop charge and select the least expensive route. In addition, each node is interested in participating in the route discovery and honestly reporting on the charge it demands, because otherwise, if it does not participate or lies, the source may select a route that does not go through the node, or fix a per hop charge that is not beneficial for the node. This means that the node will not be able to earn nuglets.

The second approach is more complex, because it requires each forwarding node to run an auction among some of its neighbors. This could be based on message exchanges between the forwarding node and the bidding neighbors, but this would mean a considerable overhead both in terms of bandwidth and latency. An alternative approach based on software agents could work in the following way: Each node sends an agent to its neighbors that represents the node at the

[2] Auctions are normally used to sell an object, and for this reason, the bidder with the highest bid is selected as winner. In our model, the auction is used to buy the service provided by the nodes, and therefore, the bidder who is willing to provide the service for the smallest charge is selected as winner. This modification does not change the properties of the second price auction in our model, because we consistently reversed every rule (e.g., in our model, the winner earns the value of the second lowest bid, while in the standard second price auction the winner has to pay the value of the second highest bid).

neighbors and implements the algorithm that the node would use in the auctions. A forwarding node then runs the auction locally among the agents of the appropriate neighbors. Finally, the packet is sent to the owner of the winning agent.

Another disadvantage of the second approach is that it can only be combined with routing algorithms in which the nodes are allowed to have multiple entries with different next hops for the same destination in their routing tables (e.g., TORA [12]). In these algorithms, forwarding nodes can use the PPM with Auctions to select the actual next hop from the list of available next hops.

The advantage of the second approach is that in each forwarding step, the node that demands the lowest charge is selected as next hop, and in this way, this approach tries to minimize the number of nuglets spent during the delivery of packets. Intuitively, this decreases the probability that a packet will be dropped because it runs out of nuglets. Furthermore, it is reasonable for the nodes to make the number of nuglets that they charge for the forwarding operation dependent on their battery status. In particular, a node that has a shortage of battery may esteem battery power higher and should demand more nuglets for the compensation of the energy spent for forwarding. In this way, however, the node decreases its chances to win auctions and forward packets. This is a nice property, because, as it is pointed out in [5], the lifetime of the network can be lengthened by routing the traffic in such a way that the energy consumption is balanced among the nodes in proportion to their energy reserves.

4 Simulation

In order to see the effects of the introduction of the proposed charging models on the performance of the ad hoc network, we conducted appropriate simulations written in plain C++ language. Our goal was to study the overall behavior of the mechanism and to show that the performance of the network does not decrease significantly. Our simulation results show that this is indeed true.

4.1 Simulation description

The basis of our simulations is a geodesic packet forwarding algorithm developed in the context of the Terminodes Project [8]. However, we considerably simplified the original algorithm described in [2] in order to ease the implementation of its simulator. This does not affect our results, since we are not interested in the performance of the packet forwarding algorithm itself, but rather in the extent to which this performance changes when we extend the algorithm with our charging models.

The simplified geodesic packet forwarding algorithm that we use works in the following way: We assume that each node knows its own geographic position and the geographic positions of each of its neighbors. Furthermore, the source of a packet knows the geographic position of the destination. Before sending the packet, the source puts the coordinates of the destination in the header of the packet. Then, it determines which of its neighbors is the closest to the destination, and sends the packet to this neighbor. Since the packet contains the coordinates of the destination, each forwarding node can perform the same operation, and sends the packet to the neighbor which is the closest to the destination. If the forwarding node does not have any neighbor that is closer to the destination than the node itself, then the packet is dropped.

We simulated this simplified geodesic packet forwarding algorithm, and its extensions with the PPM with Fixed Per Hop Charges and the PPM with Auctions. In the first extension, the fixed per hop charge depends on the number of nuglets that the source itself would need to forward the packet to the maximum distance in its power range. This number is determined using the economic model of the nodes presented later in this subsection. The number of nuglets that the source loads in the packet depends on the estimated number of hops towards the destination and the fixed per hop charge.

In the extension with the PPM with Auctions, the participants (bidders) of the auction are (the agents of) those neighbors that are closer to the destination of the packet than the forwarding node is (Figure 2). The behavior of (the agents of) the nodes in the auctions is determined by the economic model of the nodes presented later in this subsection. The number of nuglets that the source loads in the packet depends on the estimated number of hops towards the destination and the result of the auction run by the source when sending the packet.

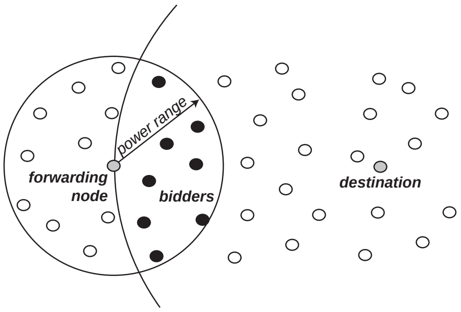

Figure 2: Geodesic packet forwarding with the PPM with Auctions

In the simulations, we modeled the economic behavior of the nodes in the following way: Each node has a battery utility function u and a nuglet utility function v. In our model, the utility of battery for a node depends on the battery level b of the node and the number n of its nuglets in the following way:

$$u(b,n) = k\ \frac{\min(n^{\alpha}, N^{\alpha})}{\min(b^{\beta}, B^{\beta})}$$

where α, β, k, N, and B are appropriate constants, and $\alpha, \beta > 0$. It can be seen from the expression that if the battery level b of the node is below a given threshold B, then battery utility increases as the battery level decreases. If the battery level of the node is above this threshold, then the battery utility is constant and does not depend on the battery level of the node. Furthermore, battery utility is proportional to the number of nuglets if this number is below another threshold N. This is reasonable, because if the node has a very small number of nuglets, then it cannot send any packets even if it has enough battery power, and thus, the utility of its battery is low.

Similarly to battery utility, nuglet utility depends on the number n of the node's nuglets and its battery level b:

$$v(n,b) = \tilde{k}\ \frac{\min(b^{\tilde{\alpha}}, \tilde{B}^{\tilde{\alpha}})}{\min(n^{\tilde{\beta}}, \tilde{N}^{\tilde{\beta}})}$$

where $\tilde{\alpha}$, $\tilde{\beta}$, $\tilde{k}$, $\tilde{N}$, and $\tilde{B}$ are appropriate constants, and $\tilde{\alpha}, \tilde{\beta} > 0$.

Nodes make their decisions based on their utility functions. Let us assume that a node is about to forward a packet, and the cost (in terms of battery power used) of this operation would be c. Let us further assume that the node is offered p nuglets as a reward for the forwarding operation. The node accepts the offer and forwards the packet, if this results in a positive payoff, which means that $p\,v(n,b) - c\,u(b,n) > 0$. Since in our simulation the number of nuglets is always a non-negative integer, the minimum number p_0 of nuglets for which it is still beneficial for the node to forward the packet is:

$$p_0 = \begin{cases} c\ \frac{u(b,n)}{v(n,b)} + 1 & \text{if } c\ \frac{u(b,n)}{v(n,b)} \text{ is an integer} \\ \left\lceil c\ \frac{u(b,n)}{v(n,b)} \right\rceil & \text{otherwise} \end{cases}$$

In the simulations, the parameters of the economic model are set in the following way: $\alpha = \tilde{\alpha} = 1$, $\beta = \tilde{\beta} = 0.5$, $B = \tilde{B} = 20$ (the value that corresponds to the full battery is 100), and $N = \tilde{N} = 1000000$. k and $\tilde{k}$ are set in such a way that $p_0 = 1$ whenever $n > 1000000$, $b > 20$, and c is the cost of transmitting a full sized packet with a full rate to a maximum distance in the power range.

The simulated networks are composed of 400 nodes that are placed randomly (uniformly) on a 1000 m × 1000 m torus (in order to eliminate border effects). The nodes do not move. Each node has the same power range of 100 m. Each node generates packets with the rate of 0.5 packet/second. The destination of the packet is chosen randomly (uniformly). The packet size is 4000 bits in the case of the basic geodesic packet forwarding algorithm, and 4400 bits in the extensions, in order to represent the overhead due to the additional information in the packet (e.g., nuglets, per hop charge, etc.). The transmission rate is 1 Mb/s. Battery consumption of the nodes is such that starting with a full battery, they can send full size packets with full rate (1 Mb/s) to a maximum distance (100 m) for 1 hour. Battery consumption is proportional to the square of the distance covered by the transmission. The initial value of the nuglet counters is set at 800000.

We varied the initial level of the batteries among 80%, 50%, 20%, 15%, 10%, and 5%. For each battery level and each packet forwarding algorithm, we ran 10 simulations and calculated the average of the results. In each case, we simulated about 2.5 hours of operation of the network.

4.2 Simulation results

Figure 3 shows the cumulative throughput of the network for the three packet forwarding algorithms when the operation of the network is started at various initial battery levels (80%, 50%, 20%, 15%, 10%, and 5%). Throughput is defined as the ratio between the number of successfully delivered packets and the total number of packets sent. As expected, both extensions perform worse than the basic, nuglet-less geodesic packet forwarding algorithm, since in the extensions, packets are bigger in size, and they can also be dropped by forwarding nodes due to the low value of the packet purse or the fixed per hop charge. However, except for very low battery levels, the difference among the performances of the three algorithms is not significant (less than 5%). Moreover, again except for low battery levels, the difference between the performances of the basic geodesic packet forwarding algorithm and its extension with the PPM with Auctions is negligible (less than 1%).

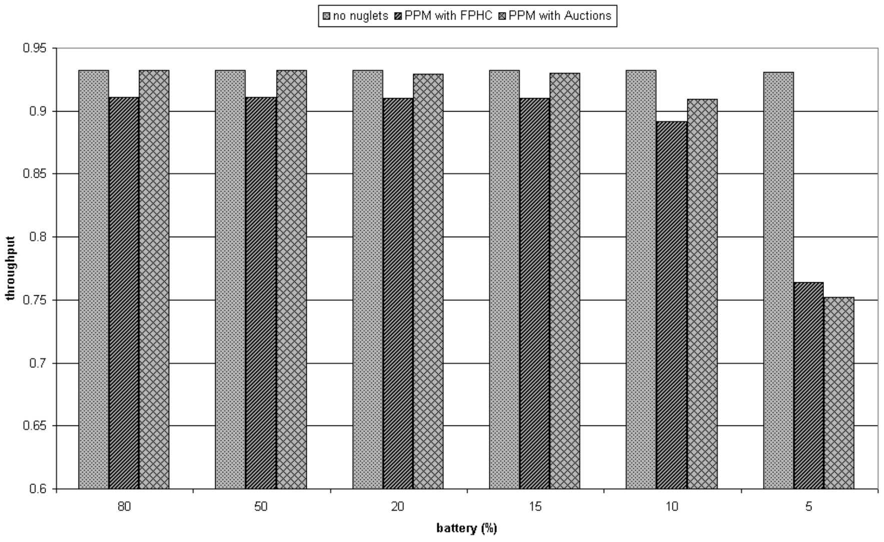

Figure 3: Throughput at various battery levels

5 Protection mechanisms

The first question to be answered when implementing any of the charging models described in Section 2 is how to represent nuglets. One possibility would be to represent them as digital coins. The main problem of this solution is that it requires an authority (usually referred to as the *bank* in the literature on digital money) that issues the coins and maintains accounts for the users. Furthermore, digital coin proposals also make use of on-line or off-line payment gateways, which are servers through which users perform transactions with the bank. However, self-organized mobile ad hoc networks should not rely on any established infrastructure and central administration. Therefore, the digital coin approach does not seem to be appropriate for our purposes.

Instead of using coins, we represent nuglets by counters in the nodes. Each node has a nuglet counter, the value of which corresponds to the wealth of the node. However, in order to prevent the node from illegitimately increasing its own counter, we require that the counter is maintained by a trusted and tamper resistant hardware module in each node. We call this module *security module.* The packet purse (i.e., the nuglets that are loaded in the packet) is protected from illegitimate modification and detachment from its original packet by cryptographic mechanisms.

In order to ensure that the node acquires the nuglets from the packet if, and only if, it really forwards the packet, the security module requires an acknowledgement from the next hop before it increases the nuglet counter of the node. The next hop is encouraged to send acknowledgements by tipping it a few nuglets.

5.1 Assumptions

Before going into the details of our implementation of the Packet Purse Model, we summarize our main assumptions.

- **Security module.** As we mentioned before, we require that each node is equipped with a trusted and tamper resistant hardware module, which we call security module. This module maintains the nuglet counter of the node and manages some cryptographic parameters (keys) that are used in the protocols presented later. In case of the PPM with Auctions, the security module hosts the software agents of the neighboring nodes, and runs the routing algorithm.

  We assume that the security modules are manufactured by a limited number of trusted manufacturers. Furthermore, since the security module is tamper resistant, its behavior cannot be modified. Therefore, security modules are trusted for always functioning correctly.

  One can imagine a security module as a security co-processor, which contains the processor itself, some memory, battery, and tamper detection circuitry. For more information on tamper resistant modules, we refer to [14, 1].

- **Public key infrastructure.** We use public key cryptography to establish symmetric session keys between security modules of neighboring nodes. These session keys, in turn, are used for hop-by-hop cryptographic protection of the packet purses and acknowledgements.

  We assume that each security module has a private key and a corresponding public key. The private key is stored in the security module and kept secret. The public key is certified by the manufacturer of the security module and the certificate is stored in the security module. In addition, we assume that the manufacturers cross-certify the public keys of each other, and each security module stores the public key certificates of all manufacturers issued by the manufacturer of the security module. Finally, we assume that each security module stores an authentic copy of the public key of its manufacturer, which is loaded in the module securely at manufacturing time. Note that storing all these certificates is feasible, because we limited the number of manufacturers.

  In this system, each security module can easily obtain the authentic public key of any other security module in the network. Let us suppose, for instance, that A wants to obtain the public key of B. B can simply send its public key certificate to A, who can verify it with the public key of the manufacturer of B. A possesses an authentic copy of this public key, since it stores the public key certificates of all manufacturers issued by its own manufacturer, and an authentic copy of the public key of its own manufacturer.

Our system is a rather pragmatic solution for the reliable distribution of public keys and we had to limit the number of manufacturers in order for it to work. The design of a general purpose public key infrastructure for large, self-organized ad hoc networks is an interesting and non-trivial problem that is beyond the scope of this paper. An approach towards the solution of this problem is described in [16].

- **Tracking of neighbors.** We assume that the neighborhood of a node does not change very fast. This makes it feasible for the node to keep track of its neighbors by running a *hello protocol* at regular time intervals. Besides discovering its neighbors, the security module uses the hello protocol to establish and maintain security associations with the security modules of the neighboring nodes.

- **Acknowledgements.** We assume that the data link (MAC) layer uses acknowledgements in order to make the communication between neighboring nodes reliable. This assumption is reasonable, since the nodes communicate via unreliable wireless links, and without hop-by-hop acknowledgements the end-to-end probability of losing a packet would be high, especially in large networks. Note that IEEE 802.11, the most commonly assumed MAC layer in mobile ad hoc networks, also uses acknowledgements.

5.2 The security module

The security module is the heart of our protection mechanism. It is the only part of the node that we can trust for correct behavior. This trust is based on the assumptions that the security module is manufactured by a trusted manufacturer and it is tamper resistant.

As we mentioned before, we assume that the security module is implemented as a security co-processor. This means that one cannot put every function at will in the security module. Actually, one should put as little as possible in it, in order to rely on as few assumptions as possible. Therefore, our design principle is the following: we implement a few, critical functions in the security module, and the majority of the functions in the node itself, in such a way that tampering the latter functions does not give any advantages to the user of the node.

The security module stores the following long term data: a system-wide unique identifier of the module; the nuglet counter of the node; the private key of the security module; the public key certificate of the security module issued by the manufacturer of the security module; the public key certificates of all manufacturers issued by the manufacturer of the security module; and the public key of the manufacturer of the security module.

In addition, the security module maintains a table in which each entry corresponds to a neighboring security module and consists of the following fields:

- **Identifier.** The security module stores the system-wide unique identifier of the neighboring security module.

- **Session key.** When the node of this security module A and the node of another security module B become neighbors, they run the hello protocol. Using the hello protocol, A and B establish a symmetric session key k_{AB}. This session key is used to protect the packet purses and acknowledgments that A and B send to each other. This protection is based on symmetric key cryptography for efficiency reasons. The establishment of the session key, on the other hand, is based on public key cryptography and uses the aforementioned public key infrastructure. Clearly, this mechanism can be used to provide other security functions as well (e.g., link-by-link confidentiality and integrity protection of the content of the packets).

- **Sequence numbers.** Besides the session key, A and B set up sending and receiving sequence numbers, which are used to detect replayed packet purses. A's sending sequence number associated with B is denoted by $c_{A \to B}$ and the corresponding receiving sequence number is denoted by $c_{A \leftarrow B}$. B has similar numbers $c_{B \to A}$ and $c_{B \leftarrow A}$, which are associated with A. A and B initialize their receiving sequence numbers to random values and use the hello protocol to set their sending sequence numbers such that the following holds: $c_{A \to B} = c_{B \leftarrow A} + 1$ and $c_{B \to A} = c_{A \leftarrow B} + 1$. Then, each time A sends a packet purse to B, it includes the current value of its sending sequence number $c_{A \to B}$ in the purse, and then increments the number. When A receives a purse from B, it verifies whether it contains a sequence number that is greater by one than its current receiving sequence number $c_{A \leftarrow B}$. If so, then it accepts the purse and increases its receiving sequence number to the received value, otherwise it rejects the purse.

- **Debt counter.** Finally, A maintains a counter $d_{A \to B}$ associated with B, which holds the value of A's debt to B. This counter is needed for the following reason: When the node of A forwards a packet to the node of B, A does not increase the nuglet counter at once, but waits for an acknowledgement from B, in order to be sure that the packet has really been forwarded. To stimulate the node of B for sending acknowledgements, A tips it a few nuglets by increasing $d_{A \to B}$ (and decreasing its own nuglet counter accordingly) when the acknowledgement arrives. Since the acknowledgement is much smaller than a normal packet, the value of the tip is just a small fraction of the reward for forwarding a normal packet. A and B clear their debts at regular time intervals using the hello protocol, which they run regularly to maintain their association.

In case of the PPM with Auctions, the security module also hosts the software agents of the neighboring nodes that participate in the auctions. These agents are exchanged and their states are updated using the hello protocol. Furthermore, the security module stores routing information, the exact form of which depends on the actual routing algorithm used.

5.3 Packet purse and acknowledgement format

According to the Packet Purse Model, each packet has to carry some nuglets that are required to forward the packet. These nuglets are stored in the Packet Purse Header (PPH), which is an additional header between the MAC Layer Header and the Network Layer Header as it is illustrated in Figure 4. The PPH is created by the security module of the source and re-computed by the security module of each forwarding node. It is cryptographically protected in order to prevent nuglet forgery and other illegitimate modifications during transit.

The PPH contains the identifier of the security module A that computed it, the identifier of the security module B of the intended next hop, the sending sequence number $c_{A \to B}$, the number of nuglets n in the packet, the number of nuglets u that B is allowed to take out from the purse, and a Purse Authentication Code (PAC), which is computed from the fields of the PPH and the cryptographic hash value of the content of the packet using a keyed cryptographic hash function g with the session key k_{AB} between A and B. In case of the PPM with Fixed Per Hop Charges, u is the fixed per hop charge, whereas in case of the PPM with Auctions, u is a value determined by the result of the auction run by A. The PAC allows B to detect illegitimate modifications of the PPH or the detachment of the PPH from the original packet.

When a node receives a forwarded packet from another node, it must send an acknowledgement. The Acknowledgement (ACK) is computed by the security module B of the node and it is piggybacked on the MAC layer acknowledgement that the node sends to the sender of the packet anyway. The ACK contains the identifier of the security module A of the previous hop, the identifier of B, the sending sequence number $c_{A \to B}$ that was received in the PPH, and an Acknowledgement Authentication Code (AAC) that is computed from the received PPH using a keyed cryptographic hash function g, where the key is the session key k_{AB} between A and B.

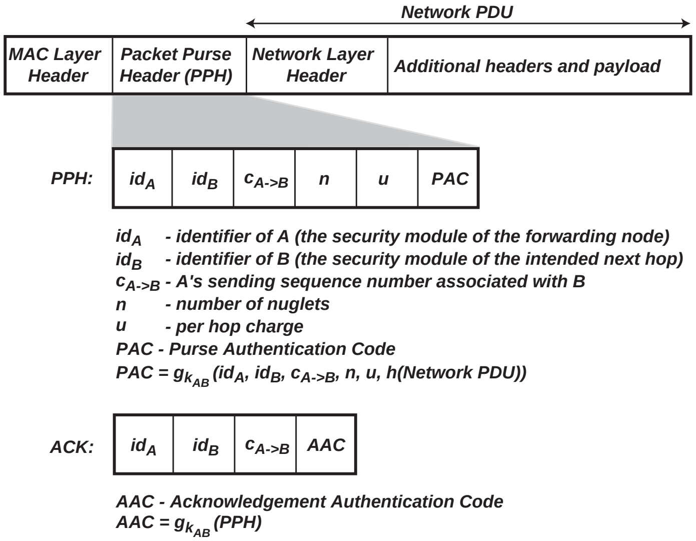
Network PDU
MAC Layer Header
Packet Purse Header (PPH)
Network Layer Header
Additional headers and payload
PPH:
id_A
id_B
c_A->B
n
u
PAC
id_A - identifier of A (the security module of the forwarding node)
id_B - identifier of B (the security module of the intended next hop)
c_A->B - A's sending sequence number associated with B
n - number of nuglets
u - per hop charge
PAC - Purse Authentication Code
PAC = g_k_AB (id_A, id_B, c_A->B, n, u, h(Network PDU))
ACK:
id_A
id_B
c_A->B
AAC
AAC - Acknowledgement Authentication Code
AAC = g_k_AB (PPH)

Figure 4: The Packet Purse Header (PPH) and the Acknowledgement (ACK)

5.4 Packet forwarding protocol

What follows is the description of the packet forwarding protocol when the PPM with Fixed Per Hop Charges is used. At the end of this subsection, we will summarize how packet forwarding in the PPM with Auctions differs from this.

The packet forwarding protocol is illustrated in Figure 5. Let us assume that node N_B received a packet from node N_A, determined that the next hop is N_C, and is willing to forward the packet for the per hop charge in the PPH. In order to acquire any nuglets from the packet, N_B must pass the PPH to its security module B together with the identifier of the security module C of the next hop and the cryptographic hash value of the content of the packet.

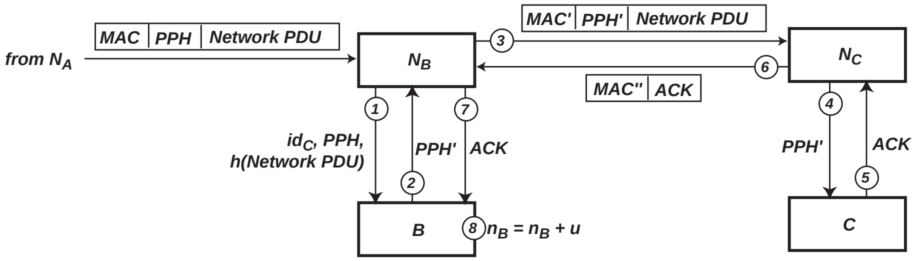

Figure 5: The packet forwarding protocol

B first verifies the PPH by checking if the sending sequence number in the PPH is greater by one than its receiving sequence number $c_{B \leftarrow A}$ associated with A. If so, then this PPH is not a replay (i.e., it has not yet been processed by B), and B proceeds by setting $c_{B \leftarrow A}$ to the received sequence number value. B also verifies the authenticity of the PPH by re-computing the Purse Authentication Code and comparing the computed value to the received one. If they match, then it knows that the PPH has indeed been created by A and has not been modified.

After successful verification, B calculates a new PPH by including the identifiers of B and C, and the sending sequence number $c_{B \rightarrow C}$; decreasing the number of nuglets in the PPH by the fixed per hop charge; and computing the new Purse Authentication Code using the cryptographic hash function g and the session key k_{BC}. Then, B increases its sending sequence number $c_{B \rightarrow C}$.

Finally, B stores the new PPH internally, and outputs a copy for N_B.

N_B attaches the new PPH to the packet and sends it to N_C. N_C must acknowledge the reception of the packet. For this purpose, N_C passes the PPH to its security module C, which constructs the ACK, and returns it to N_C. N_C then sends the ACK to N_B piggy-backed on the MAC layer acknowledgement.

When N_B receives the ACK, it passes it to its security module B. B tries to find the corresponding PPH in its internal memory by matching the identifier of C and the sending sequence number received in the ACK to the identifiers and sending sequence numbers in the stored, pending PPHs. If B finds the corresponding PPH, then it verifies the authenticity of the ACK by re-computing the Acknowledgement Authentication Code from the PPH and comparing it to the value received in the ACK. If they are equal, then B increases its nuglet counter by the per hop charge fixed in the PPH, and deletes the PPH from its internal memory. It also tips N_C some small number of nuglets by increasing the debt counter $d_{B \to C}$ and decreasing its nuglet counter accordingly.

In case of the PPM with Auctions, the security module of the forwarding node determines the next hop, therefore, when the node wants to forward a packet, it only has to pass the PPH and the hash of the content of the packet to its security module. The security module runs the auction among the software agents of the potential next hops internally, and constructs the new PPH according to the result of the auction. Then, it outputs the new PPH and the identifier of the next hop, and the node forwards the packet. The rest of the protocol is the same as in the case of the PPM with Fixed Per Hop Charges.

5.5 Robustness

The protection mechanism described above is robust and resists against various attacks. Nuglet forgery is prevented, because it would require either an illegitimate increase of the nuglet counter, or the generation of fake packet purses or acknowledgements. The former is impossible, because the nuglet counter is manipulated by the security module, which functions correctly and its behavior cannot be altered. The latter is prevented by the use of cryptographic checksums (i.e., the PAC and the AAC), which can be computed correctly only by the security module. The PAC also protects the integrity of the PPH during transit. Furthermore, the PPH cannot be detached from the packet and re-used with another one, because the calculation of the PAC involves the cryptographic hash value of the content of the packet. Replay of the packet purse is prevented by the use of an ever increasing sequence number that is placed in the purse. This solution is preferable to the application of time-stamps, because it does not require the security module to rely on an internal physical clock and to run clock synchronization protocols, which would need to be secured as well.

5.6 Overhead

At first sight, our protection mechanisms may seem to add a significant computational overhead to the system. But this overhead is small when compared to all the functions that are required to accomplish packet forwarding. In particular, the calculation and verification of the PPH and the ACK require only cryptographic hash function computations, which can be done very efficiently. Public key cryptographic operations are used only rarely (only in the hello protocol). Moreover, most of the processing load will be supported by the security module; to some extent, it can be accomplished in parallel with the processing performed by the main processor of the node. Finally, as mentioned in [15], the energy required to perform computation is negligible when compared to the energy required to perform transmission.

Another issue is the communication overhead represented by the length of the PPH and the ACK. Assuming that the identifiers of the security modules are 8 bytes long, the sending sequence numbers are 4 bytes long, the nuglets and the per hop charges are represented on 4 and 2 bytes, respectively, and the authentication codes are 20 bytes long, we get that the PPH is 46 bytes long (hence the 400 bits overhead in the packets in the simulator described in the previous section), and the ACK is 40 bytes long. Considering the usual size of payloads in wireless networks, this seems to be an acceptable overhead.

6 Related work

To the best of our knowledge, the only papers addressing the problem of misbehaving nodes are [11] and our previous paper [3]. The authors of [11] consider the case in which some malicious nodes agree to forward packets but fail to do so. In order to cope with this problem, they propose two mechanisms: a *watchdog*, in charge of identifying the misbehaving nodes, and a *pathrater*, in charge of defining the best route avoiding these nodes.

The paper shows that these two mechanisms make it possible to maintain the total throughput of the network at an acceptable level, even in the presence of a high amount of misbehaving nodes. However, the operation of the watchdog is based on an assumption that is not always true (as reckoned by the authors): the *promiscuous* mode of the wireless interface. Another problem is that the selfishness of the nodes does not seem to be castigated; on the contrary, by the combination of the watchdog and the pathrater, the misbehaving nodes will not be bothered by the transit traffic while still enjoying the possibility to send and to receive packets. The proposed mechanisms could be enriched in such a way that a misbehaving node would be locked out by its neighbors. However, this possibility itself could be exploited to mount denial of service attacks.

Compared to our approach, this proposal exhibits the advantage of being easier to implement, as it does not require a currency, and therefore, it is independent of any cryptographic function. However, our approach achieves more: not only does it foster cooperation, but in addition, it prevents overloading. Moreover, the concept of currency can be used to reward more sophisticated services that go beyond simple packet forwarding.

In the Internet, proposals similar to ours have been made, in which the users who benefit from a given service remunerate the users who provide that service. A recent example is MojoNation [17], which is focused on Web publishing and named after the virtual currency it uses (the Mojo). MojoNation members acquire Mojos by contributing resources (e.g., disk space or CPU time) to the community, or performing services (e.g., hosting a tracking or relay service), and they can use their Mojos to browse and download the content (e.g., MP3 files) available within MojoNation.

7 Conclusion

In this paper, we investigated how the cooperation of nodes in self-organized mobile ad hoc networks – notably the willingness to forward packets for the benefit of others – can be stimulated by introducing a virtual currency (nuglets) in the system. Besides stimulating cooperation, nuglets are a way to encourage the user to make a moderate usage of the network, and to keep her device turned on (so that it can be used as a relay) even if she is not expecting any packet to be sent to her at that time.

We discussed possible charging models for packet forwarding, and described the economic model that we used in our simulations. We also detailed the way the exchange of nuglets can be protected against fraud. We studied the behavior of networks in three cases: i) without nuglets ii) with nuglets and fixed per hop charges and iii) with nuglets and per hop charges based on auctions. Our results show that the introduction of the nuglets does not significantly decrease the performance of the network. We pointed out that the Packet Purse Model with Fixed Per Hop Charges can easily be integrated with any existing routing protocol such as DSR or AODV.

We believe that introducing a kind of virtual currency can serve several other purposes in mobile ad hoc networks. First, it can be used to remunerate not only communication services, as described in this paper, but also information services. Second, it can be used as a way to pay for the usage of backbones or satellite links, when a node has to communicate with a very distant party. In this case, the virtual currency will have to be converted in some way into “hard” currency.

In terms of future work, we intend to study the proposed mechanism when integrated with a mainstream routing protocol (e.g., DSR or AODV) and to simulate it under ns; we do not expect important differences with the results obtained so far, however. We also intend to explore possible approaches for the estimation of the initial packet purse by the source of the packet, and we will work on techniques aimed at compensating the overall loss of nuglets due to packet dropping.

References

[1] R. Anderson and M. Kuhn. Tamper Resistance – a Cautionary Note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, Oakland, California, November 1996.

[2] L. Blazevic, S. Giordano, and J.-Y. Le Boudec. Self-Organizing Wide-Area Routing. In *Proceedings of SCI 2000/ISAS 2000*, Orlando, July 2000.

[3] L. Buttyán and J.-P. Hubaux. Enforcing Service Availability in Mobile Ad-Hoc WANs. In *Proceedings of the 1st IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, August 2000.

[4] L. Briesemeister and G. Hommel. Role-Based Multicast in Highly Mobile but Sparsely Connected Ad Hoc Networks. In *Proceedings of the 1st IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, August 2000.

[5] J.-H. Chang and L. Tassiulas. Energy Conserving Routing in Wireless Ad-hoc Networks. In *Proceedings of the IEEE Infocom Conference*, 2000.

[6] P. Dutta. *Strategies and Games – Theory and Practice*. MIT Press, 1999.

[7] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, March 2000.

[8] J.-P. Hubaux, J.-Y. Le Boudec, S. Giordano, M. Hamdi, L. Blazević, L. Buttyán, and M. Vojnović. Towards mobile ad-hoc WANs: Terminodes. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Chicago, September 2000.

[9] J.-P. Hubaux, Th. Gross, J.-Y. Le Boudec, and M. Vetterli. Towards self-organized mobile ad hoc networks: the Terminodes Project. *IEEE Communications Magazine*, January 2001.

[10] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, T. Imielinski and H. Korth (edts.), Kluwer Academic Publisher, 1996.

[11] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of the ACM Mobicom Conference*, Boston, August 2000.

[12] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the IEEE Infocom Conference*, 1997.

[13] Ch. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999.

[14] A. Pfitzmann, B. Pfitzmann, and M. Waidner. Trusting Mobile User Devices and Security Modules. *IEEE Computer*, February 1997.

[15] G. J. Pottie and W. J. Kaiser. Wireless Integrated Sensor Networks. *Communications of the ACM*, May 2000.

[16] L. Zhou and Z. Haas. Securing ad-hoc networks. *IEEE Network*, November-December 1999.

[17] Mojo Nation web site. `http://www.mojonation.net/`.

[18] Terminodes web site. `http://www.terminodes.org/`.