Implementing Charging in Mobile Ad Hoc Networks

João Girão[1,2], João Paulo Barraca[1,2], Bernd Lamparter[1], Dirk Westhoff[1], Dr. Rui Aguiar[2]

Abstract – In order to keep up with new networking needs, it becomes necessary to adopt mechanisms for charging network usage in a universal way. The Secure Charging Protocol (SCP) aims at answering this complex authentication, authorization, accounting and charging (AAAC) problem, and provides a view based on a different business model, one that has been adjusted to cope with technological changes.

This document discusses SCP as a possible solution to the AAAC problems in MANETs and addresses the improvements made to this protocol in terms of Quality of Service (QoS) and User Interfaces. An implementation of this protocol on PDAs is also described.

I. INTRODUCTION

Nowadays, networks have outgrown the typical Ethernet switch, connected to a router, serving access to a couple of computers. Wireless technologies have broadened the concept of networks to a sharing medium, where all are responsible for a little part in the whole. Words like "hotspot" have become popular marketing suggestions and, step by step, the average user becomes aware of the new possibilities new technologies offer.

Many researchers have worked on making the most out of the advantages of this evolution by considering decentralised scenarios where any node is not only a client but also a server. The concept of Mobile Ad-hoc networks (MANET) [1] became a major research area, under this framework. MANETs can range from networks made only of nodes that forward packets for each other and therefore have no need for a centralised structure, until networks with special static points of access to a main network or even the Internet. Nevertheless, MANETs are always highly volatile networks which support high mobility, decentralised routing algorithms, and wireless technologies (e.g. 802.11a/b/g and Bluetooth), providing new opportunities for research on technologies and business (and AAAC) models.

This paper briefly discusses MANETs and associated research issues in section 2. Section 3 introduces the SCP, and the QoS extensions developed. Section V discusses the implementation done, while section VI presents our main conclusions.

1 – NEC Europe, Networking Labs

2 – Universidade de Aveiro, D.E.T.

II. MANETS

MANETs presented a new set of problems and challenges related to Route Discovery mechanisms, AAAC, QoS and Security.

Conventional Route Discovery methods assume a fixed network and that, once a route is discovered, it may only be changed by failure of a router, e.g. seldom. This is not the case in MANETs. The unstable nature of the wireless networks exponentially increases the number of failures, which is further compounded by the intrinsic mobility of the nodes.

The search for a new Routing Protocol, that could be efficiently used in MANETs, lead to the development of several groups of protocols, most notably proactive and reactive routing protocols.

A Proactive Routing Protocol [2] is one that tries to keep as much information as it can *a priori* so it can quickly determine a route once a packet arrives to a certain node. They are known for having a slow start but can normally reduce the time it takes for a packet to first leave the node.

Reactive Routing Protocols [3, 4], as the name suggests, react on the event of sending a packet if a route is not known to the target. While minimising the quantity of information sent over the wireless link, they suffer from longer delays because of route discovery just before sending the packet.

Although an agreement on which is the best Routing Protocol has not yet been reached (and other types of protocols exist), currently AODV, OLSR and DSR have achieved an overall support.

Another issue in MANETs is security. Solutions for layer 2 security (privacy) are normally disliked since they are not uniform between technologies. Moreover, the well known 802.11b LL2 security mechanism (WEP) has proved to be flawed. Typically, IPSec [5, 6] would be considered as a solution. However, security methods such as the aforementioned are normally put aside due to the mobile nature of the network and the complexity of maintaining sessions (and stable routes) in this environment. Also, the network overhead is rather significant, something not desirable at all in these

mediums which are usually formed by a combination of low-power mobile nodes.

Current security approaches merge both network layer and application layer security, but do not seem to be applicable when AAAC issues are considered.

The security problem is naturally associated to AAAC issues. AAAC architecture concepts [7] can be applied to MANETs with slight modifications. First of all, distributed accounting must be implemented, as all nodes are possible accounting and data collection points. Authentication and authorization must be considered case by case. The charging mechanisms are closely coupled with accounting issues.

QoS is a special hard problem in such a heterogeneous environment: traditional QoS parameters remain relevant, but new parameters are now also important. The processing time, e.g., required for a traffic flow becomes important in small devices such as PDAs and cell phones.

Still, packet delay and bandwidth continue to be the most important QoS parameters. However, since it is hard to predict the path and even more to guarantee it will remain the same, in MANETS we have to look at minimums and maximums rather than fixed values. This is the approach of INSIGNIA, a well-known QoS protocol for MANET [8].

III. THE SCP PROTOCOL

For discussing charging issues, a clear view of business environment should exist. More than Access and Application services, future ISPs will obtain revenue from organizational and social issues, maximizing cooperation between users. The Secure Charging Protocol [9] is based on a Business Model appropriate to these networks, where the service provided is the inter-cooperation, and thus forwarding of traffic by multiple nodes. Other solutions of this sort have also been presented [10] although their approach is slightly different.

Especially in the cases of small devices in ad-hoc environments, it is not usually to the owner's advantage to loose processing and bandwidth in forwarding other people's packets. Therefore, a user will want his packets forwarded but not to forward other's traffic. By providing “incentives”, this practice intends to have users receive money for the service they provide. Users that use this cooperative service will therefore be charged, but also paid by their services. The more packets a node forwards the more money it will receive at the end. This money can then be reinvested in its own usage of the cooperative service, to have his packets forward by other nodes.

Two P factors, P+ and P-, are defined, to reflect the ratio between the money a user has to pay for sending packets and the amount he receives for forwarding. These concepts are the basis for a service an ISP can provide anywhere a MANET can be deployed, using the ISP access points. The actual protocol that has been developed to support this approach is the SCP.

The Secure Charging Protocol is an AAAC protocol that focuses on securely retrieving accounting information for traffic flows, while retaining relevant information associate with the routes that were taken by each of the packets.

A first step in understanding the protocol is to review the security issues that were taken into account. For example, the receiving node (a paying node) cannot be trusted to report the correct traffic information. In a similar way, we cannot assure data will not be tampered with along the way.

To solve these questions, a set of cryptographic primitives such as signing and authentication were used within the protocol itself. Accounting information is sent by the last forwarding node, which can accurately determine which packets were received by the receiving end. An Access Router is defined, which is trust-worthy, and will collect all relevant information.

The protocol operation is divided into three parts (Fig. 1):

- Registration
- Data Transfer
- Accounting

During the registration phase nodes acquire their identity within the domain. An Access Request is sent to the Access Router containing relevant information on the node's domain, plus a username and password. In response, the Access Router returns valid public and private keys within a certificate, a shared key and charging information.

During the data transfer, when a node needs to send out data, it must sign it with its own private key so that the information can be verified as originating from that particular node. (This is of particular importance to avoid charging wrong person). Together with the payload, a hash chain is begun so that the route can be confirmed at Access Router (AR) level. The hash chain is built by hashing a known data together with the shared key of each node the packet transverses.

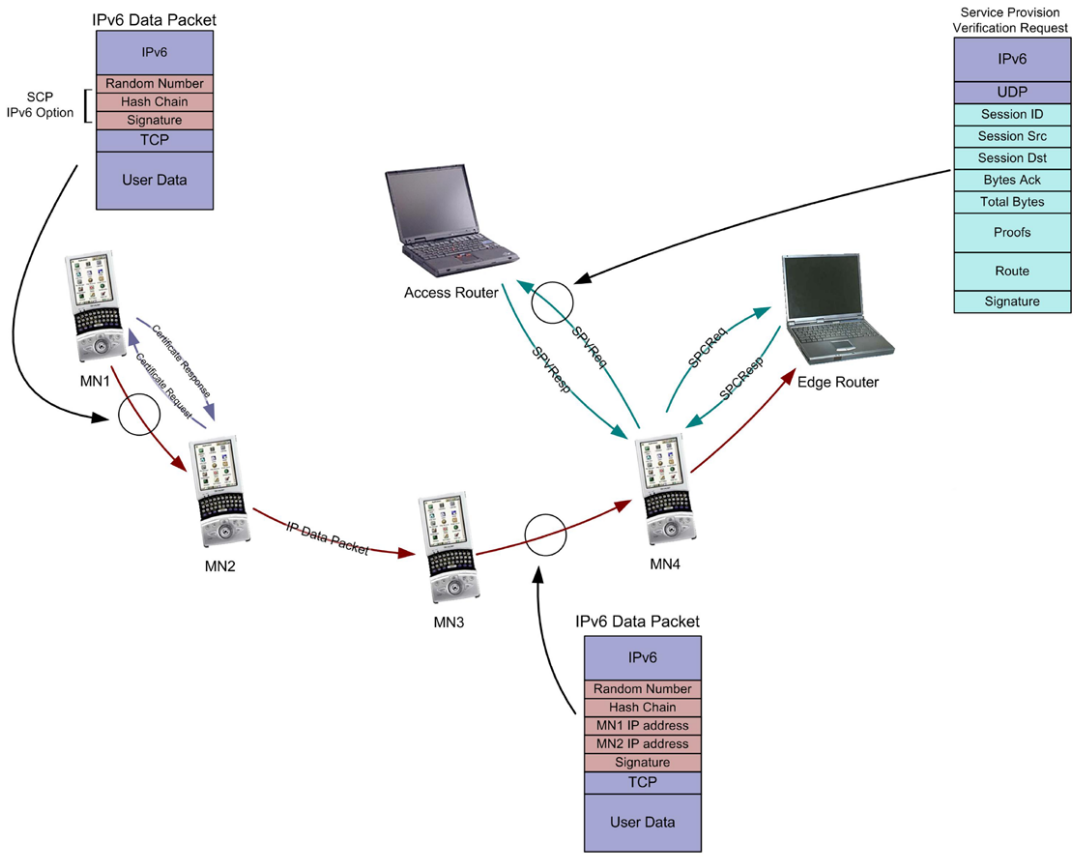

Figure 1 *- Typical SCP network*

Each time the packet hops through a node, the data is confirmed as being correct and the hash chain is updated before the packet is forwarded. This way the node is not wasting processing time on invalid packets.

The last forwarding node is responsible for the accounting. It keeps database information on packets, routes and hash chains and periodically sends this information to the Access Router when reachable.

The accounting information is confirmed with the receiving node and both his answer and the information held by the last forwarding node are sent to the accounting server. By reproducing the order of hashing in the AR, the end result is the same and can be compared to value provided by the network.

a.QoS

One of the most complicated things to assure is that a node complies with a QoS class. This is especially hard in a MANET where nodes have different capabilities. In this context, QoS is handled as class differentiation: each node support a finite number of classes (queues) that receive different processing times for the packets in them. The definition of these classes is an issue for the operator.

The QoS information is marked in the flow label of the IPv6 packet and confirmed by encoding it in the hash chain. When a packet arrives to the input queue, it is sorted onto whichever QoS class it belongs. The heaviest processing functions (in this case, the signature verification function) are then used within a weighted round robin scheme to distribute processing time unevenly.

IV. IMPLEMENTATION OVERVIEW

The implementation of SCP was done for typical PDAs, the Zaurus, running the Linux operating system. The implementation was performed in such a way that it could run on most Linux platforms and on different architectures (currently i386 and StrongArm).

One of the main objectives of the implementation was to have a threaded modular

prototype that could easily be extended, having pieces of code replaced without affecting the overall performance and public interface. The modular approach allows for any extension to be both simple and less time consuming because only a particular piece of the code has to be changed. The threaded approach was chosen, in order to provide the SCP implementation with flexibility, non-blocking asynchronous tasks and the possibility of service differentiation on a per-packet basis.

The SCP implementation consists of two layers: the SCP finite state machine, in the high level, and the network interface: the Packet Handler.

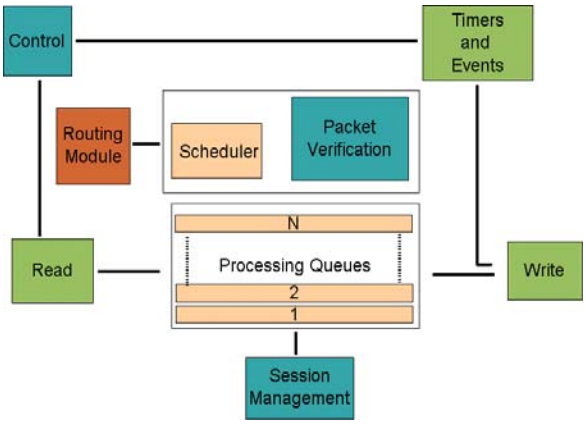

Figure 2 *- Packet Handler Implementation Overview*

The Packet Handler is the basis for SCP. All SCP functions and packet differentiation are done at this level. Functions are then called depending on the packet type from the finite state machine which characterises SCP.

There are three main threads in the Packet Handler. Two belong to the Packet Manager which interfaces directly with the network. One is for packet reading (referenced as Read in the schematic) and the other for sending (represented as Write). These processes are asynchronous to the rest of the program; they depend only on the state of the packet.

The other thread is the round robin that distributes the processing power by calling the verification function as many times for each queue as it has been configured to (we consider this thread to be a scheduler and therefore it was named as such in the figure).

The SCP state machine implementation (which functions at a higher level than the one depicted) is divided in to two classes that share some of the code and belong independently to the Node or the Access Router.

Here, all the functions regarding SCP packets, storing of data and cryptographic methods are grouped so they can be easily accessed from the underlying layer.

At this level we will find the Timer thread that is responsible for timed events such as a packet being re-sent or the cleansing of some stale data.

a. *The Routing Protocol*

SCP was implemented in such a way that it is independent of the underlying routing protocol, although some optimizations can be made if the protocol is source based.

A system was added so that the routing part of SCP could be easily integrated into the protocol itself and even changed at runtime through the use of a module. By using a dynamic module we allow for the external insertion of functionality by a process of loading code from a file.

The architecture chosen is based in considering that a session is a communication between two IPs and can be multiplexed into several sub-sessions depending on the route the packet took to reach its destination. This way, no matter what the routing protocol is, we can store accounting information independently.

Another important step in the separation between accounting and routing is the fact that every packet carries with it the route it took since it was sent. Even though this may be considered unnecessary overhead, it has proved to be of extreme relevance for the protocol to work.

SCP was first implemented with a static route module, and later on integrated with an implementation of AODV6 by the University of Helsinki [11]. Slight modifications had to be made to SCP due to technology overlaps and the hooks used by this AODV6 implementation were reproduced within SCP even for further integration with other routing protocols.

b. *Database Usage in Account Management*

Once the protocol implementation was concluded, there had to be a way of linking the accounting information for charging purposes. As a result, an interface to a database was made so that data could be stored in a relational way. We can now browse the database using the SQL language and group the relevant information.

To prevent the database interface from interfering with the performance of the implementation, special care was taken to separate

both. An asynchronous event list was chosen as the communication method between the SCP Access Router and the database system.

In different threads inside the Access Router, the time a processing unit has to wait for the database access has been minimised. Once an accounting packet arrives, data is submitted to the event system that runs parallel to SCP and is then stored in the database.

A database was also chosen to hold the user database for authentication purposes. A user is validated by matching his account information sent in the Access Request packet with the data stored in the database via Pluggable Authentication Modules. A special module for this Linux facility was created using a MySQL database interface.

c. Monitoring Runtime Protocol Information

The implementation done also includes a monitoring server that reads on a specified UNIX socket for information requests from other programs in the same machine. This server is implemented as a separate thread with access to all the information of the main SCP process. After a program connects to the socket, it can request information about general variables, traffic statistics, and also details about the active sessions and known nodes. Information like traffic statistics or general information can be important for the end user to access, while the details about the active sessions and the known nodes are somewhat more important for debugging and testing purposes of the protocol.

A graphical interface running in both ARM and x86 architectures was developed to exploit this monitoring facility. This graphical interface is implemented using the QT libraries distributed by Trolltech Corporation [12] allowing a seamless integration both with the OPIE environment of the OpenZaurus distributions, and the QT based graphical environment distributed with the majority of the linux distributions.

The interface is divided in 4 tabs: general information, traffic statistics, sessions and nodes.

d. General

This tab shows the user a general view of the protocol. He can see his *userid*, the *domain* where he belongs, the IPv6 address of the current Access Router, the number of active sessions, the number of nodes known to the user, the *P factors* attributed by the ISP and the ratio between these factors. The most important fields in this tab are the P *factors* and their ratio, which are of special importance, since they allow the user to have an idea of the involved costs of use.

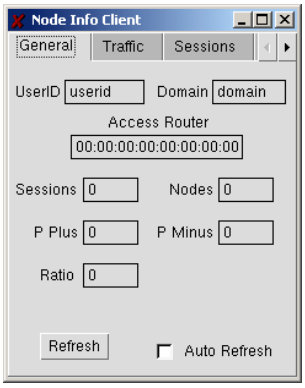

Figure 3 *- General information tab*

e.*Traffic*

By selecting this tab the user can access some statistics about the traffic he is sending, receiving and forwarding, as well as the Average Packet Size. The *Real Ratio* field indicates the ratio between sent, received and forwarded traffic, allowing the user to check, using the *Difference* field below, if the traffic crossing the network interfaces differs from the traffic expected by the ISP. If the *Difference* is positive, the user will not receive any bonus. On the other hand, if it's negative the user is forwarding more traffic than the ISP expects and it will receive a bonus.

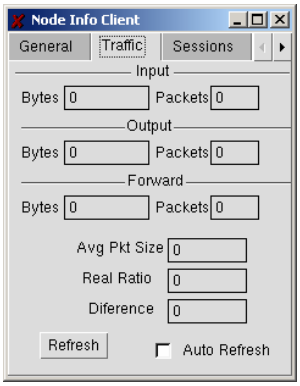

Figure 4 *– Traffic information tab*

f. *Sessions*

The sessions tab shows the list of active sessions and detailed information about a selected session. When a user pushes the refresh button, the program will show the list of active sessions; the user

can then select one particular session to see its details. They are: source and destination IP, the name of the host, number of bytes received or sent and number of bytes forwarded. Only one field with information about session traffic is used at a time; in one session the node can only be receiving, forwarding or sending. The *Session Flags* field display some debugging flags concerning the session; one example is the position of the node in the session (last, first or middle).

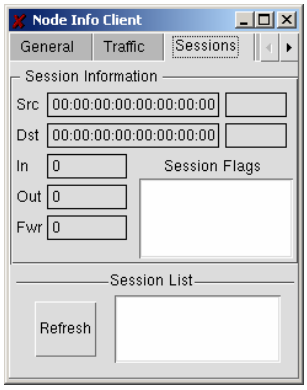

Figure 5 *– Session information tab*

g.*Nodes*

The nodes tab is useful to list all nodes currently known to the SCP protocol, get their IP addresses and names, and also check if the protocol has a valid certificate from that node. It is mainly for debugging purposes.

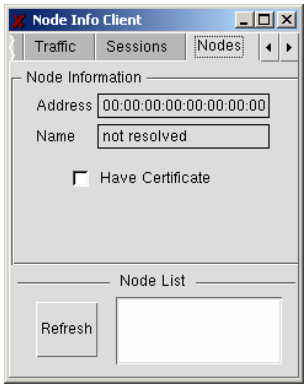

Figure 6 – *Nodes information tab*

V. ISSUES PENDING FUTURE DEVELOPMENTS

After the conclusion of this implementation, a couple of aspects could be improved in order to reach a commercial implementation.

a. *Security Issues*

There are still a few pending security issues, mainly in what regards nodes working together to form a colluding attack and Denial of Service. These aspects have to be considered carefully in order to define what kind of appropriate measures can be taken. The later could be improved by experience although no solution for both these problems is suggested in the description of the protocol [scp].

b.*P+ and P- factors*

As part of the charging scheme, P+ and P- are calculated as the factors for charging and rewarding respectively. At the moment, their calculation does not take the network profile and configuration into account, which means that the prices will not reflect the network topology. As a result, nodes can be unfairly charged for traffic and the ISP might even loose money.

A mathematical function must be implemented taking these parameters into account so that the ISP can decide on its own benefit margins. This function can easily be integrated on the existing implementation.

c. *Edge Routers*

It is desirable that a MANET has some form of communication with the exterior. Normally, this would mean an external gateway, however, in this case, the gateway position will be assumed by a special node: an Edge Router.

This node is special in the sense it holds a connection to the outside network as well as one to the inside network and can therefore provide forwarding capabilities to other networks. The charging mechanisms must be reviewed so that this node acts as the last forwarding node and accounts for the traffic going outside of the network.

d.*Integration with other Routing Protocols*

At the moment, SCP has been integrated with AODV6 and static routing. However other MANET Routing Protocols may be desirable depending on the kind of network SCP will be inserted into. Also, a transparent change of the routing module at runtime will be required to prevent loosing accounting information when moving between networks.

VI. CONCLUSION

The presented implementation offers a secure and modular implementation of the Secure Charging Protocol, ready for evaluation of the current mechanisms offered by the protocol specification. It also provides a stable base for future development, evaluation and testing of new features and proposals. In these regards, the SCP implementation has proved to outperform the expectations. The implementation was tested in networks with 4 hops, in Zaurus machines, with total packet delays of around 15ms per node. The end to end delay has to be correlated from the verification rate of intermediate nodes.

With the implementation of the suggested developments and the inclusion of tighter security in the transmission of the messages, the implementation will be ready for network testing under real world environments, and provides a first pre-commercial prototype of a SCP product.

VII. REFERENCES

[1] S. Corson and J. Macker, "*Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*", RFC 2501, January 1999

[2] T. Clausen, Ed. and P. Jacquet, Ed., "*Optimized Link State Routing Protocol (OLSR)*", RFC 3626, October 2003

[3] David B. Johnson, et al., "*The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*", draft-ietf-manet-dsr-09.txt, April 2003

[4] Perkins, C., et al., "*Ad hoc On-Demand Distance Vector (AODV) Routing*", RFC 3561, July 2003

[5] S. Kent and R. Atkinson, "*IP Encapsulating Security Payload (ESP)*", RFC 2496, November 1998

[6] R. Atkinson, "*IP Authentication Header*", RFC 1826, August 1995

[7] B. Aboba, et al., "*Criteria for Evaluating AAA Protocols for Network Access*", RFC 2989, November 2000

[8] G-S. Ahn, et al., "*INSIGNIA*", draft-ietf-manet-insignia-01.txt, October 1999

[9] B. Lamparter, K. Paul, D. Westhoff, "*Charging Support for Ad Hoc Stub Networks*", In Elsevier Journal of Computer Communications Special Issue on Internet Pricing and Charging: Algorithms, Technology and Applications, Elsevier Science, Aug. 2003.

[10] Sheng Zhong, Jiang Chen, Yang Richard Yang *"Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks"*, In Proceedings of IEEE Infocom 2003, San Francisco, USA,

2003. [11] HUT AODV for Ipv6, Helsinki University of Technology, http://www.tcs.hut.fi/~anttit/manet/aodv, as in 15-10-2003

[12] Web: Trolltech Creator of QT, Trolltech corp, http://www.trolltech.com, as in 14-10-2003