QoS-Differentiated Secure Charging in Ad-Hoc Environments*

João Girão[1,2], João P. Barraca[1,2], Bernd Lamparter[1], Dirk Westhoff[1], and Rui Aguiar[2]

[1] NEC Europe, Networking Labs, Germany
{joao.girao, bernd.lamparter, dirk.westhoff}@netlab.nec.de
[2] Universidade de Aveiro, D.E.T.-I.T., Portugal
{jpbarraca@av.it.pt, ruilaa@det.ua.pt}

Abstract. In order to keep up with new networking needs, it is 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. SCP fits business models especially adequate for ad-hoc networks. This document discusses SCP as a possible solution to the AAAC problems in MANETs and presents the improvements made to this protocol in terms of Quality of Service (QoS). An implementation of this protocol on PDAs and the results achieved are discussed.

1 Introduction

Wireless technologies have broadened the concept of networks to shared uncontrolled mediums where all are responsible for a little part of the whole. Words like "hotspot" and "mesh" have become popular in marketing and, step by step, the average user becomes aware of the new possibilities new wireless technologies offer.

Many researchers have exploited 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. In every case, MANETs are always highly volatile networks which support high mobility, decentralised routing algorithms, and wireless technologies. It is not surprising thus that MANETs have provided new opportunities for research on technologies and business (and AAAC) models. AAAC architecture concepts [2] can be applied to MANETs with slight modifications: distributed accounting must be implemented, as all nodes are possible accounting and data collection points; and authentication and authorization must be considered case by case.

MANETs flexibility is attracting network service providers to incorporate ad-hoc technology support in their products. MANETs seem to provide mechanisms to

* This work is in part supported by the EU Framework Programme 6 for Research and Development Daidalos (IST-2202-506997).

J.N. de Souza et al. (Eds.): ICT 2004, LNCS 3124, pp. 1093–1100, 2004.

extend connectivity range, reduce node's power consumption and increase both node's mobility and failure tolerance - all without a significant investment from the service provider. There are two problems, though. Ad-hoc technologies require a critical mass of well-behaved nodes willing to forward other's traffic. Moreover, the ISP also expects to charge the users accessing the network, in a parallel way to traditional connections. Both problems create the need to develop new charging paradigms and business models, capable of securely charging over "trust-no-one" environments and motivating node's participation in ad-hoc networks.

This document introduces a business model adequate to traffic charging in ad-hoc networks, while increasing the participation level by motivating nodes to forward traffic. A protocol appropriate to this business model, the SCP protocol, and a novel implementation supporting traffic differentiation, will be presented. We will discuss issues concerning security and key sizes and present experimental results obtained. Finally, in the end, we will present the conclusions reached.

2 The SCP Protocol

Charging is clearly associated with business models, and ad-hoc environments present an opportunity for the appearance of novel business models. In the future, ideally ISP revenues will be maximised by increasing cooperation and promoting communication between users, instead of the traditional Access services.

The implementation of this type of business concept, where the operator will benefit by the users' willingness to promote communication (peer-to-peer services), has some challenges. 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 by forwarding other people's packets. Therefore, a user will want his packets forwarded, but not to forward other's traffic. By providing "incentives", novel business models [3] intend to reward users with money for the (forwarding) service they provide. The more packets a node forwards the more money it will receive at the end. This money can then be reused in its own usage of the cooperative service, to have his packets forward by other nodes, or can become a net profit at the end of the contract.

Two price 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. This concept is 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 Secure Charging Protocol [3]. SCP is adequate for the Business Models suggested, where the service provided is the inter-cooperation, the forwarding of traffic by multiple nodes. Notice that other alternative solutions based on incentives have been proposed (e.g. [4, 5]), with different trade-offs in terms of scenarios and required security levels.

SCP is an AAAC protocol that focuses on securely retrieving accounting information for traffic flows, while retaining relevant information associated with the routes that were taken by each of the packets. The protocol aims to handle situations where the receiving node (a paying node) cannot be trusted to report the correct traffic information, and provide per-packet assurances, if required.

To solve the questions associated with AAAC problems in this environment, a set of cryptographic primitives like signing, verifying and keyed hash chains, were used

within the protocol to create digital signatures. These primitives are essential to guarantee the authenticity of the accounting information provided. This accounting information is sent by the last forwarding node, which can accurately determine which packets were received by the receiving end. This node has the interest of providing as much information as possible, as it will benefit (economically) of this (it is the last node in the communication path that will forward data). SCP further defines an Access Router, which is trust-worthy, and will collect all relevant information. Note that this "Access Router" will often be the router interconnecting the ad-hoc network to a physical infrastructure (hotspot), but it is not necessarily so.

The protocol operation (Fig. 1) is divided into three phases: registration, forwarding and charging phases.

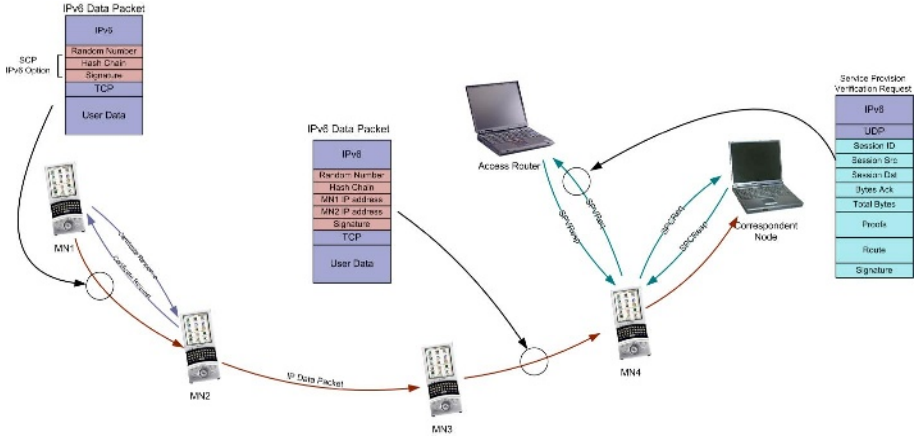

Fig. 1. Diagram showing operation of SCP.

a) 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 over an encryption tunnel the node's certificate, the node's private key, a shared key with the access router and charging information. This information can be used to authenticate node-related information.

b) During 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 people). Together with the payload, a hash chain is initiated using a random number and a shared key so that the route can be confirmed at Access Router (AR) level.

Each time the packet hops through a node, the data is confirmed as being correct and authentic and the hash chain is further increased, by hashing the last value with the shared key of each node, before the packet is forwarded. This way the node is not wasting processing time on invalid packets, and introduces a confirmation of his activities (forwarding) inside the packet. The freshness of this hash is guaranteed by a random number seed introduced by the sender.

c) 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, during the accounting phase.

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, together with the shared key, in the AR, the end result calculated in the server is compared to the value provided by the network, and thus provides a path confirmation mechanism. Having verified the path reported, the AR is then able to derive credits and charges to issue to all nodes in the communication path.

These concepts have been extended to support QoS. Making sure that a node complies with given QoS parameters is usually difficult, and is especially hard in MANETs, where nodes have different capabilities, and links have varying performance. In this context, we handle QoS simply as class differentiation: each node supports a finite number of classes (queues) that receive different processing times. The definition and parameterization of these classes is an issue for the operator.

A packet is transmitted associated with a class. The QoS class information is marked in the flow label of the IPv6 packet and is confirmed because it is part of the signed data of each packet. This creates a register of the QoS class provided for the packet by the nodes. A queue exists for each class. Service differentiation is provided by sorting packets to different QoS classes (different queues). The heaviest processing functions (in this case, the signature verification) are then used within a weighted round robin (WRR) scheme to distribute processing time unevenly, providing differentiated services to the QoS classes. Note that more advanced scheduling disciplines could be chosen, at the expense of larger implementation complexity.

During the charging phase, the AR derives the credits and charges taking in consideration the QoS reported information coming from the last forwarding node.

3 Implementation

SCP was implemented in a very common PDA, the Sharp Zaurus, running the Linux operating system. The implementation was performed in such a way that it can 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. Threading was implemented 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 (Fig. 2). The Packet Handler is the basis for SCP. All SCP verification functions and packet differentiation are done at this level. Functions are then called depending on the packet type (class) from the finite state machine which characterizes 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; and depend only on the state of the packet. The other thread is the WRR mechanism that distributes the processing power by calling the verification

function as many times for each queue as it has been configured to (the Scheduler). Naturally, empty queues' timeslots will be used by pending packets in other queues.

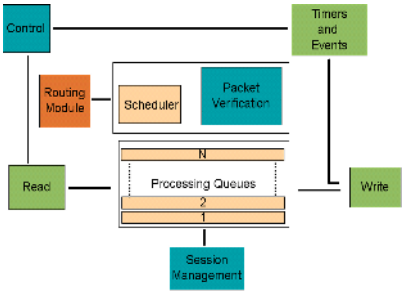

Fig. 2. Packet Handler Implementation and Key Global Functions

The SCP state machine implementation is divided into two classes that share some of the code and belong independently to the Node or the Access Router. (Note that this state machine is quite complex, and thus is not completely represented on the figure). At this level, all the functions regarding SCP verification functions, storing of data and cryptographic methods are grouped so they can be easily accessed from the underlying layer. This is represented as the Verification module in Fig. 3.

At this same level is another thread, the Timer thread, which is responsible for timed events such as a packet being re-sent or the cleaning of some stale data.

The implementation also includes Session Management capabilities. It includes a monitoring server that reads on a specified UNIX domain socket for information requests from other programs in the same machine mainly for debugging and demonstration purposes. 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 [8] 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.

Notice that the modularity characteristics of this software allows for different verification or scheduling algorithms to be supported with minimal effort. Furthermore, the interfacing with the ad-hoc routing module is quite flexible, and different ad-hoc routing mechanisms can be easily supported. Moreover, run-time change of the routing protocol (because of node movement across different networks, e.g.) is supported. Currently the implementation supports AODV6 and static routing.

The choice of the signature algorithm will greatly influence the overall performance of any SCP implementation. Moreover, characteristics of ad-hoc networks like multi-hopping, reduced bandwidth, processing power and battery, and high distrust on the intentions of the nodes, impose some restrictions on the level of security and, therefore, in the signature algorithm to be used. Three key points needed to be evaluated: security level, processing power and network overhead. Security

level should be high enough to avoid impersonation and guaranty non-repudiation of the network traffic. These requirements are related to the key and algorithm to be used and also to the duration of the keys. If the ISP generates a new certificate every hour, key sizes can be very small. This may be not generally appropriate to all ad-hoc networks because the network overhead to exchange certificates could be too high in some networks. We supposed that certificates can have a high duration, and were to be secure enough to comply with the security requirements.

Different keying algorithms will have different signing and verifying times (Table 1). Specially, RSA and ECDSA are very different in terms of timings. The following table compares the time, in milliseconds, needed to perform a sign or verify operation using RSA or ECDSA. These tests [6] were executed in a PDA equipped with a StrongARM 206Mhz processor using the standard OpenSSL library [9].

Table 1. Comparison between ECDSA and RSA.

Algorithm	**RSA**		**ECDSA**	
Key size	**704**	**1024**	**131**	**163**
Sign (ms)	32.4	78	22.1	28.8
Verify (ms)	2.5	4.3	43.4	55.9

According to [8] if we consider an average number of hops equal or lesser than 5 and a level of security equivalent to an RSA-1024, ECDSA proves to be the most appropriate choice. However if the key length is below the previous value, RSA may be a better choice to maximise implementation performance.

RSA and ECDSA key sizes are very different. ECDSA provides a much more compact representation of the key regarding to the same security level, and thus is the most appropriate choice for the signature algorithm in order to minimise network overhead. A smaller key size also implies smaller certificates and consequently less memory occupied at each node by other nodes certificates.

After comparing RSA with ECDSA, ECDSA was found to provide better security, performance and overhead ratios than RSA, when considering the target environments of our SCP implementation. Final choices were of a security level related to an RSA key with 1024 bits or a security-equivalent 163 bits ECDSA key [7].

4 Implementation Results

We used a highly optimised implementation [6] of ECDSA which is capable of performing signing and verifying operations about 3.2 times faster than the OpenSSL implementation as in the CVS snapshot 20021202.

Our test scenario consisted in four nodes creating a bidirectional flux of packets with different packet rates, different packet sizes, different rates of packet verification at each node, and using a key size of 163 bits. The intermediate nodes are Zaurus PDAs and edge nodes are Pentium 500Mhz processors. With these scenarios we could effectively evaluate the protocol's impact in the round-trip delay of a bidirectional connection like a VoIP communication, and QoS differentiation behaviour.

QoS differentiation was quite simple to demonstrate, and performed according with the expected WRR performance. Processing power was scheduled asymmetrically by the four queues implemented according to the weights allocated.

Other measures determined the benefit of verifying only a percentage of the packets forwarded. Fig. 3 shows the round-trip delay without SCP, SCP with verification of all packets crossing a node, and SCP with verification of half of the packets crossing the node. This last approach should not introduce many security problems, since control packets are all verified and, statistically all packets have a high probability of being verified along the route.

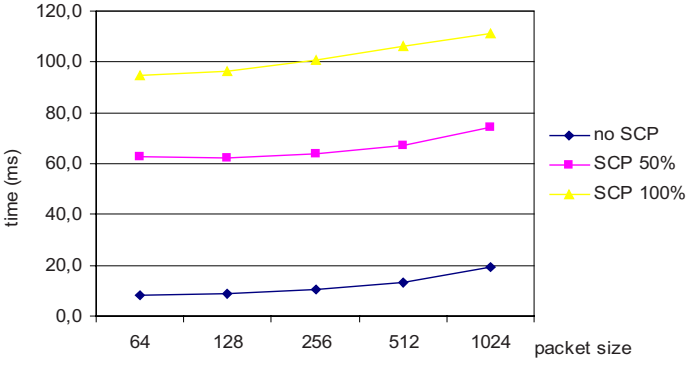

Fig. 3. Round trip delays with different verification ratios

The round-trip delay results show a constant increment on the measured time in function of the packet size. The delay difference between packets without using SCP in the network is the same, only added by an additional delay. This delay is mainly caused by the cryptographic functions verify and sign. (Measurements shown were taken for a single QoS class).

We identified three major functions introducing delay: SCP processing, cryptographic verify and cryptographic sign. Taking the case of a packet with 256 bytes, and doing signature verification in all nodes, the round-trip delay increased from ~10ms to ~100ms. Considering that nodes perform 6 verifications and 2 signatures in a round-trip, Table 2 shows the impact of each component in the overall performance of the protocol, along the different nodes.

It is clear that verification is the key function in this implementation. The choice of a key length appropriate to the security requirements from a specific scenario plays a major role in maximising protocol performance – and in our case we have a 163-bit ECDSA, usually regarded as life-time security. Also, the use of dedicated cryptographic hardware integrated in the mobile nodes, a trend already present in many wireless devices, could push the security of SCP even further by allowing these very secure keys without significant penalty loss, and minimising battery consumption.

Table 2. Percentage of CPU time at each node by function

	MT	FW Node	Third	CN
SCP	0,8%	1,6%	1,6%	0,8%
Sign	6,0%	0,0%	0,0%	6,0%
Verify	13,9%	27,8%	27,8%	13,9%

5 Conclusion

The SCP protocol can perform secure charging in ad-hoc networks without relying on any centralised infrastructure to account traffic, and considering that the nodes are not trusted. This allows the implementation of novel business approaches, promoting users' willingness to cooperate and benefiting from peer-to-peer trends.

The implementation described offers a secure and modular implementation of the Secure Charging Protocol, able to evaluate the current mechanisms offered by the protocol specification, and including service differentiation. 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 key length is very important to determine the security achieved and also the performance penalty imposed with the charging process. The use of dedicated cryptography hardware, like FPGA technology, could greatly improve the results obtained and reduce dramatically packet delay in the case of very long key lengths.

References

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

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

[3] 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.

[4] 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

[5] Naouel Ben Salem, et al, "A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks", ACM MobiHoc '03, 2003.

[6] Ingo Riedel "Security in Ad-hoc Networks: Protocols and Elliptic Curve Cryptography on an Embedded Platform", Diploma Thesis, Ruhr-Universität Bochum, March 2003.

[7] A.J. Menezes, P. C. van Oorschot and S. A. Vanston, "Handbook of Applied Cryptography", CRC Press, 1996

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

[9] Web: OpenSSL Project, http://www.openssl.org, as in 14-10-2003.