A Lightweight and Secure Session-Aware Ad-Hoc Charging Protocol

João Paulo Barraca, Susana Sargento, Rui L. Aguiar

***Abstract*— Ad-hoc networks can be used to increase the radio range of hotspot scenarios and the number of prospective clients. However, the constraints created by the nature of wireless links and low power devices pose many challenges to this integration. Charging and efficient forwarding are two key functionalities in an ad-hoc network in order to allow its integration in the existing infrastructure networks. This paper proposes a lightweight and secure Session-Aware charging protocol (SACP) for these environments. This protocol is able to provide guarantees of cooperative behavior in traffic forwarding, with minimal network overhead. The performance results, evaluated in multiple environments, show the protocol merits in terms of throughput and network overhead.**

***Index Terms*— SCP, DSR, SACP, MANET, Charging.**

I. Introduction

NETWORK Operators are increasingly developing new services and business models capable of exploiting the possibilities made available by wireless technologies. Services accessed through wireless infrastructures, usually deployed as hotspots, are becoming popular, in a response to the expansion of the concepts “always on”, “anywhere” and “anytime”.

Mobile Ad-hoc networks (MANETs), through their multi-hop characteristics, can be used to enlarge the revenues of the operators by increasing the radio coverage of the hotspots with low cost and easy deployment. In complex environments, nodes will have different capabilities, with some nodes having privileged connections to the core network. Ad-hoc networking allows the exploitation of this complexity, and of this connection diversity.

When creating these scenarios, operators face the problem both of charging the users for the traffic they produce and of motivating the users to forward traffic. In a standard hotspot, all traffic from and to a node uses an equipment owned by the network operator, where charging proofs are created and sent to the central accounting and charging server. On the other hand, in ad-hoc networks, the traffic never reaches this server if both sender and receiver nodes are located inside the same ad-hoc network. Solutions like [7] solve this issue by forcing routes to include some nodes in the infrastructure network, but they inevitably lead to sub-optimal routes in an already constrained environment. Moreover, in a MANET the mobile ad-hoc nodes are expected to act as routers forwarding traffic from other nodes. These requirements pose another problem: if the network operator does not motivate users to contribute to the packets forwarding, no routing of information will exist inside the ad-hoc network, and the number of potential ad-hoc clients of the services offered in the infrastructure will be reduced. This “motivation” requires the deployment of mechanisms for rewarding users for the battery, processing power and bandwidth required for forwarding traffic. This rewarding may consist of a virtual currency, such a credit increase, or can be translated in benefits to additional services, depending on the particular business model preferred.

Some proposals for the charging and rewarding processes in ad-hoc networks connected to infrastructure networks have already been proposed ([6][7][4]). However, these proposals present several problems: non-optimal routes [7]; do not assure the correctness of the charging information sent to the infrastructure [6]; require a viable channel to the charging information or introduce large network overhead [4][6].

In this paper we propose a charging and rewarding mechanism, denoted as SACP (Session-Aware Charging Protocol), evolved from the SCP [4] protocol. SACP is able to provide correct charging and rewarding information without the need of sub-optimal routes, and with small network overhead and processing requirements. More specifically, we propose the notion of session to be assigned to packets of a flow using the same route; in this way, only some packets need to carry the full route information. The results achieved through simulation show that the performance of the network, both in terms of throughput achieved and network overhead, is much increased in SACP compared to previous solutions.

This paper is organized as follows. Section II addresses some related work which is the basis for the work presented in this paper. Section III presents the proposed charging and rewarding solution and its detailed functioning in the most relevant phases. The simulation results are presented in section IV, and the conclusions are addressed in section V.

Manuscript received January 21, 2005. This work was supported in part by the European Union FP 6 project Daidalos (IST-2202-506997).

João Paulo Barraca, Susana Sargento and Rui L. Aguiar are with Universidade de Aveiro, Instituto de Telecomunicações, 3810-193 Aveiro Portugal; e-mail: jpbarraca@av.it.pt, {ssargento, ruilaa}@det.ua.pt..

II. Related Work

Several proposals exist for charging in ad-hoc networks. Some of them assume the existence of an entity in the infrastructure network that collects the proofs of each flow or packet, while others consider micro-payments where nodes can pre-buy some credits in a "bank" and use tokens directly to pay for the traffic they produce.

Considering the first group of proposals, which is the one that will be dealt in this paper, we highlight the following approaches: [6], where all the forwarding nodes need to collect the proofs to later report to the infrastructure entity; [7], in which all traffic needs to cross the access point, and [4] that optimizes the procedure by having only the last forwarding node collecting the information. This latter approach is denoted Secure Charging Protocol (SCP).

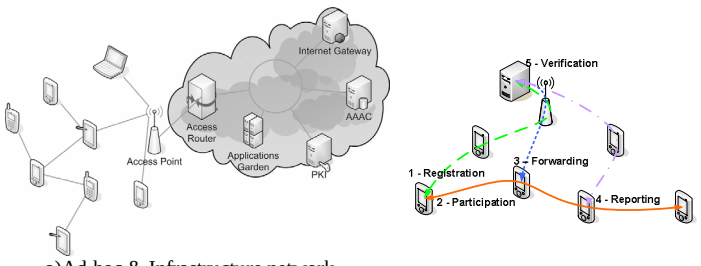
SCP is a solution for secure charging in ad-hoc networks which, besides charging users, also addresses the problem of cooperation in ad-hoc networks by rewarding forwarding nodes with a fraction of the credit used by the sending node. In this proposal, all packets are marked with a proof containing some route information and a hash chain to secure the route avoiding malicious manipulation by a mis-behaved node. The forwarding nodes update the proofs contained in the packets with their information, with the last forwarding node responsible for collecting and reporting the proofs. An accounting and charging server, located in the infrastructure network, stores user profiles, credit values, and collects the proofs sent by the nodes. Its initial proposal only addressed the Dynamic Source Routing protocol [1] and required the list of forwarding nodes to be included in all packets. Further work [5] extended the initial proposal to support non Source Routing (SR) protocols like AODV [2] or OLSR [3]. Thus, the sending node does not need anymore to add the full route to every packet: the route is built at each forwarding node and the hash chain is accordingly updated. SCP main drawback is the overhead in the network, since in each additional IPv6 node hop, 16 more bytes are added in the header, corresponding to the node's address. This requires adjustment of all upper layer protocols which rely on the size of the packet as reported by the IPv6 header (e.g. the checksum of TCP packets). If some header which relies in the packet size field of the IPv6 header is present in the packet, probably that packet will be considered as corrupted and discarded. Also, due to the continuous packet size increase, fragmentation may be repeatedly required at every node, increasing both the processing requirements of the forwarding process and the overhead of control data.

DSR includes an optimization process, the Automatic Route Shortening (ARS) [1]. This process uses the notion of end-to-end flow, which corresponds to the packets and route of a flow. This flow may have different routes during its lifetime (not simultaneously). In ASR, the route information is cached at the nodes during a specified time, and does not need to be sent in all packets. With this process, the overhead of DSR is much reduced. Using DSR with SCP, if a route changes and a new end-to-end flow associated with this route is not established, a new node in the route will not be rewarded. Since this establishment is not controlled by SCP, and only triggered by network events, it results in large amount of errors in the rewarding mechanism. These problems arise because there is no integration between the routing and charging mechanisms. The approach proposed in this paper considers as basis the SCP protocol, but introduces some of the concepts of ARS, and integrates them into the charging and rewarding mechanism in order to reduce the control overhead of the charging process. This approach is further independent of the routing protocol (can be used without DSR) and can be adapted to most currently proposed ad-hoc routing protocols.

III. SACP Operation

In this section we present SACP, a charging and rewarding protocol for ad-hoc networks in scenarios with interconnection to the infrastructure network managed by a network provider. The SACP protocol is based on rewarding concepts and the SCP protocol, but improves over previous proposals [6][7][4][5]: it includes the capability for flow admission without the need of sub-optimal routes, and has smaller network overhead and processing requirements in every node.

In this proposal we assume that there is at least one ad-hoc access network connected, through an Access Router (AR), to the infrastructure network (Figure 1a). The infrastructure network is managed by a network operator and all equipments are trusted: the AR, the Public Key server which stores keys both of the users and the operator, the Authentication, Authorization, Accounting and Charging (AAAC) server that collects and verifies the proofs, and the Internet gateway which connects the infrastructure network to other networks. This is a simplified model of an operator network, with the required elements to the proper operation of SACP. We assume that users have a contract with the network operator, and a pair of public and private keys. The equipments in the ad-hoc network are considered to be non-trusted and potentially selfish, not wasting resources for others' profit.



a)Ad-hoc & Infrastructure network architecture

b) SACP phases

Figure 1 – Architecture and SACP operation

The general SACP operation is depicted in Figure 1b. When a node enters in the ad-hoc network, it goes through a registration process to become known by the AR (and AAAC), to be authenticated and authorized, and to receive the cryptographic material required for the charging process (Registration phase). When communicating with other nodes inside or outside the ad-hoc network, each forwarding node processes the packet and forwards it; the fields containing information on the route are cryptographically secured, so they cannot be wrongly modified along the path (Participation and Forwarding phases). SACP includes concepts of ARS, with routes cached at the nodes for a specified amount of time. Therefore, the packets only carry information about the route for the charging process when the route information in the nodes is about to expire or topology changes are detected.

The node belonging to the flow's path one hop way from the receiver, which we denote as the last forwarding node, is responsible for sending the proofs to the AR (Reporting phase). These proofs contain information on the path(s) of the flow, and are sent to the AR when the number of proofs collected in the node reaches a specific number, or when a timeout expires. Notice that, in the case of traffic with destination outside the ad-hoc network, the last forwarding node is replaced by the AR. When receiving the proofs, the AR sends them to the AAAC to verify the truthfulness of the information, through the cryptographic information contained in the proofs, and retrieves the information of the ad-hoc route (Verification phase). The AAAC is then able to correctly charge the sending (and, eventually, the receiving) nodes, as well as correctly reward the forwarding nodes.

The Registration and Verification phases are similar to the same phases in SCP, as well as the security assumptions and specificities. In the next sub-sections we detail the remaining phases of the mechanism which are specific to SACP.

A. *Participation*

When a node generates a packet, it checks if it has already information of this flow and its current destination. This information is denoted by session. The information about this session is stored in the node. If this session does not exist, it creates a new one (storing its information in the node). Then, it activates a timer, the RouteUpdateTimer that, upon expiration, triggers the event of adding the route information in the outgoing packets at specified intervals. This route information is included in RouteUpdate messages. Moreover, the node sets the number of packets that may be sent before the requirement of a new route update message, the PacketsUntilRU, to a predefined value. It also generates a new Flow ID for this session, different from 0 and from recent values. The first packet will then carry a RouteUpdate message and will have the generated Flow ID set in the IPv6 header. The fields of the RouteUpdate message are:

{ NHops || RouteList || Sequence || Hash Chain || MAC }

NHops indicates the number of addresses present in the list of nodes in the path, the RouteList; it starts with the value 0 at the sender and is incremented by one at each hop towards the destination. The RouteList will be inexistent in the packet just sent, and will grow by one IPv6 address at each forwarding node. The Sequence number is a random number used to avoid repetition attacks and the MAC (Message Authentication Code) field contains the signature of the header created with the sending node's private key, to assure the identity of the sending node. The Hash Chain will be initially created by hashing a pseudo header with data from the packet and the node itself. As nodes may be charged and rewarded according to the Quality of Service of the packets being sent or forwarded, this information is placed in the pseudo header.

If the RouteUpdateTimer expires or if PacketsUntilRU reaches 0, the RouteUpdateTimer is rescheduled, PacketsUntilRU is reset to the preconfigured value, a new Flow ID is generated, and a RouteUpdate message is sent. The combination of a counter variable with a timer allows the refresh of the route after a certain timeout or after a certain number of packets, thus minimizing errors in the rewarding mechanism. Further packets will not carry a RouteUpdate header, but instead will carry the new Flow ID and a PacketProof header with the following structure:

{ Sequence || HashChain || MAC }

The Hash Chain field is constructed through the same pseudo header as before. This much smaller proof requires less processing from the forwarding nodes and causes less network overhead. However, it may not contain information about the "real" forwarding nodes: the ones rewarded will be the ones that forwarded the packet containing the RouteUpdate header and the correspondent Flow ID. However, as will be shown in the next section, the number of wrong assumptions related to the forwarding nodes is very small (less than 1.2%).

B. *Forwarding*

Upon receiving a packet, each forwarding node checks if the MAC field is correct. If it is not, the packet is dropped.

Otherwise, if the packet has a RouteUpdate message, the node increases the NHops variable and adds its IPv6 address to the RouteList. In the case of a TCP packet, this operation will require adjusting TCP fields, like the checksum or maximum segment size. However, this is only required when RouteUpdate messages are present. Then, the node concatenates the charging pseudo header constructed from the packet with the Hash Chain and calculates a new hash. The result of the new hash will be inserted in the Hash Chain field.

After this process, since the node needs to remember that it is included in the route indicated by the Flow ID, it caches the pseudo header until a configured timeout expires or until a packet of the same session arrives with a new Flow ID.

If the packet carries a PacketProof header with a known Flow ID, the forwarding node just updates the Hash Chain and forwards the packet. On the other hand, if the packet only carries a PacketProof header but the Flow ID is unknown, the forwarding node decides if is willing to forward the packet for free. The decision will depend on the nodes interest in forwarding packets without being rewarded, and in the number of packets already forwarded for free. We assume that nodes are willing to forward a very small percentage of the traffic for free if that represents better connectivity in the network. A situation without errors happens when all packets have the full route encoded; the error ratio grows both with higher mobility of nodes and higher periods between RouteUpdate. However, the purpose of the mechanism is to minimize these situations.

C. Reporting

The last forwarding node is responsible for collecting the proofs present in the packets, either it being a RouteUpdate or a PacketProof message. Proofs are packed in a ProofReport message and are sent to the AR, periodically or as soon as it is reached a number of stored proofs enough to fill a packet with maximum size. The AR verifies all information and the Hash Chains contained in the ProofReport, and issues a ProofResponse to acknowledge the reception of the proofs. Both the ProofReport and ProofResponse messages are protected by public key cryptography.

IV. Results

We simulated both SCP and SACP in order to evaluate the performance and efficiency of these mechanisms. The SCP version used is the one proposed in [5]. Both mechanisms were implemented in NS-2 [9] in two scenarios with different mobility patterns and different network loads. The PHY layer we used was IEEE802.11 with maximum values for rate and range of 2Mbits and ~300 meters respectively. Note however, that this PHY was selected only by convenience of usage inside ns2, and our work would be applicable to any PHY that could be used in the selected scenarios. Although the numerical results would be different, the comparison between the different charging protocols should have similar results. To analyze the performance penalty caused by the overhead of the charging mechanisms, simulations were also performed with the same scenario and flow patterns, but without any charging protocol. This reference level allows us to evaluate each analyzed aspect discarding normal dynamics of mobile ad-hoc networks.

The first scenario simulates a Freeway with eight lanes, four in each direction. The movement of the nodes follows a normal distribution with average of 33m/s (120Km/h) and a variation of 17m/s (60Km/h). When the simulation starts, nodes are placed in random positions and start moving according to the direction of the lane. The second scenario simulates an area of 1 square Km, where the movement of the nodes is determined by a random waypoint model (RWP), and the speed varies according to a uniform distribution in the interval between 0 and 5 m/s (18Km/h). In this scenario, the initial position of the nodes is also random. All scenarios are composed by forty nodes moving according to the established pattern. One of the nodes acts both as an access point and an AR, providing a source for external flows and receiving the proofs reported.

The traffic patterns created addressed both TCP and UDP flows to better simulate the dynamics of a mobile ad-hoc network. We considered a mixed traffic scenario with internal traffic (between nodes in the ad-hoc cloud) and external traffic (between ad-hoc nodes and the outside); the internal traffic was set to 1/3 of the traffic to the access point. TCP flows are modeled through FTP (File Transfer Protocol) applications, while UDP flows are composed by CBR (Constant Bit Rate) flows (Audio and Video) and on-off exponential (Exp) flows. In each experiment, the flows are initiated according to a Poisson process with a certain mean time interval between calls, and each flow has an average duration exponentially distributed. The characteristics of these flows are summarized in Table 1, representing a load factor of L. The factor represents the number of flows created at each sending node when generating the traffic patterns; the simulations were executed always with fractions of the factor L.

Generator	Packet Size (Bytes)	Flow Duration (s)	Inter-arrival time (s)	On/Off times (s)	Rate (Kb/s)	Name
CBR	80	120	30	-	48	Audio
CBR	1000	120	60	-	256	Video
EXP	512	120	30	0.5	128	Exp

Table 1 – Specification of the UDP flows simulated

The results addressed in the next sub-sections are a mean of 5 runs of 1000 seconds each.

A. Overhead

The overhead created by each protocol is one of the measures of the efficiency of each proposal, with impact in the performance of the network. Since the size of the proofs increases along the path, both in SCP and SACP in the *RouteUpdate* messages, we determine the overhead as follows:

$$\text{Overhead} = \frac{\text{ControlBytesSent} + \text{ControlBytesReceived}}{\text{DataBytesSent} + \text{DataBytesReceived}}$$

The values of *ControlBytesSent* and *ControlBytesReceived* include both the proofs in the packets and the communication between the last forwarding nodes and the AR. The results presented consider the *RWP* scenario with UDP flows (the overhead difference between SCP and SACP in other scenarios is similar).

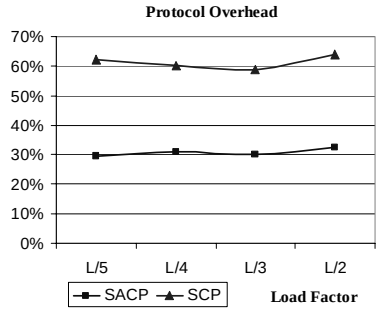
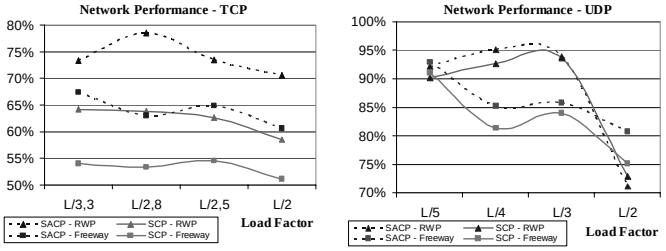


Figure 2 – Charging protocol overhead *vs* network load

According to the results obtained in Figure 2, we observe that SACP can reduce the total overhead of the charging protocol in 50% (from 60% to 30%) as compared to plain SCP. As can be observed, the overhead tends to increase as the network load increases. This happens because of the higher number of retransmissions when the network is under heavy load. Other results also showed this tendency, especially with UDP flows, since TCP reduces the congestion and minimizes collisions.

B. Performance

Since additional information is sent into the ad-hoc network during the charging procedure, it is expected to observe degradation in the total throughput of the flows. The extent of the degradation will be related both with the encoding of the routes and the procedure of reporting proofs to the central AAAC server. We define performance as the average throughput of all flows in the network compared to a network without any charging protocol for different load factor values.



a) TCP flows b) UDP flows

Figure 3 – Throughput *vs* network load

Figure 3a presents the throughput results of TCP flows in both mechanisms and with the two simulated scenarios. We observe that the TCP connections are strongly influenced by the route encoding procedures. A larger header added to packets reduces the available bandwidth and triggers the ECN mechanisms of TCP adjusting the throughput of the flow. The difference between the two proposals is between 7% and 15%, with SACP achieving higher performance. We also notice that the increase in the mobility of the nodes (*Freeway*) and subsequent higher routing overhead also decreases the performance ratio (this is consistent with [8]).

The results related to UDP traffic (Figure 3b) show that UDP is less influenced by the decrease in the overhead than FTP, since UDP does not have congestion detection and recovering mechanisms. Nevertheless, the results are usually better in SACP with a difference of approximately 5%. The results from UDP traffic also show that the performance is largely influenced by the load in the network. This behavior is due to the radio link saturation of the AR when a large number proofs needs to be reported. In overall, the impact in the network performance is lower in SACP.

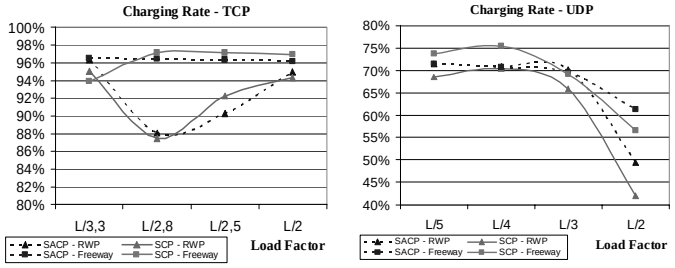
C. Charging Rate

To compare the efficiency of the charging process, we defined a metric denoted as charging rate. It is defined as the percentage of proofs received at the AAAC server from the total of proofs collected, by the forwarding nodes:

$$\text{ChargingRate} = \frac{\text{NumberProofsReturned}}{\text{TotalNumberProofs}}$$

Since in the simulations the nodes stop moving after the traffic flows stop, there is some probability of having nodes with stored proofs that do not have a route to the AR, and then, to the AAAC. Therefore, a low charging rate means that proofs were collected, but it was impossible to report those proofs in the specified time.

The results presented in Figure 4 show that both mechanisms are able to report the majority of the proofs to the AAAC. The difference of the results obtained with each implementation is also very small. We can see that, for specific load factors, the charging rate decreases. In these cases, the wireless medium can be completely saturated and the AR does not receive all the proofs until the end of the simulation. Notice, however, that the proofs are not discarded; they are collected and sent to the AR whenever the load of the wireless link decreases.



a) TCP flows b) UDP flows

Figure 4 - Charging rate *vs* network load

In UDP flows there is a large decrease in the charging rate with the load factor. Further analysis revealed that the average number of hops for UDP packets is higher than the one for TCP packets. This is caused by the congestion avoidance mechanisms of TCP that reduces the throughput of flows located farther from the sending node, thus avoiding higher congestion levels. In UDP this mechanism does not exist, and then, some nodes farther away from the AR accumulate a large number of proofs without being able to report them.

D. Reward Errors

As stated above, SACP may induce errors in the rewarding process. These errors occur whenever a route, being used by a flow, changes before a new *RouteUpdate* message is sent. The periodicity of this message will be reflected in the percentage of errors occurred. The value used in the simulations for the *PacketsUntilRU* variable is 2, which means that every third packet will include a *RouteUpdate* message (usually *RouteUpdateTimer* is larger than the transmission time of 3 packets in a flow). According to Figure 5, the reward error values are rather low for the *RWP* scenario, always lower than 0.35%. The *Freeway* scenario, due to the increased mobility associated with nodes, achieves higher values; nonetheless, they are always below 1.2%. For the operator, this error is meaningless, since the probability of a route to increase or to decrease is sensibly the same. For the user, this error means that some forwarded packets will not be rewarded, and some non-forwarded packets will be rewarded. In the end, for a specific user, the error associated with the rewarding will be very small.

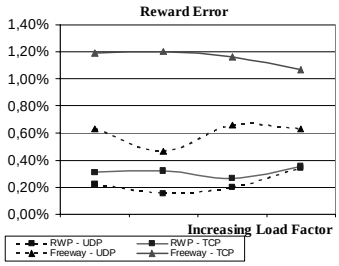


Figure 5 - Reward error ratio *vs* network load

E. Period between updates

The value of the number of packets without routing updates was chosen in order to minimize the errors due to route changes in already established flows. In this sub-section we analyze the influence of the periodicity of the *RouteUpdate* messages, defined as the number of packets without *RouteUpdate* messages (*PacketsUntilRU*), in the reward error of the mechanism. As shown in Figure 6, for the *RWP* scenario, with TCP and load *L*/2.8, the percentage of errors increases with the period between *RouteUpdates*, following a linear distribution. Although this increase is obvious, notice that the reward error for a *RouteUpdate* periodicity of 60 packets (very large value) is lower than 2%. Notice that, for this periodicity value, the network overhead should be very small. The choice of the optimum value will depend on the maximum amount of errors expected.

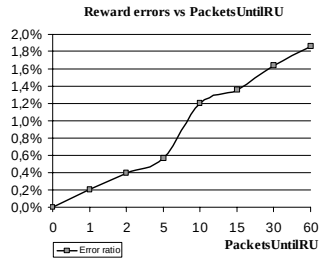


Figure 6 – Reward error *vs* periodicity of *RouteUpdate* messages

The option to implement this mechanism as a static value was taken due to the fact that dynamic acknowledgment of route changes can potentially lead to an increase in the amount of errors. Consider, for example, a worst case in which a route with some forwarding nodes is used by a flow with a high packet rate. In this specific case, the time taken to notify the sending node can be significantly large, and a large amount of packets already in transit can be wrongly rewarded.

V. Conclusion

The work presented in this paper addresses a feasible approach for charging and rewarding in ad-hoc networks used in the context of hotspot scenarios. This approach is based on the concept of session to be assigned to packets of a flow using the same route, preventing the requirement of the route information in all packets. The results achieved show that the impact in the network performance and the network overhead in SACP are much reduced, when compared to SCP, while maintaining similar efficiencies in both charging and rewarding mechanisms. We also demonstrated some of the major issues influencing the performance both of SACP and SCP: both load and nodes mobility can have a high impact in the performance of both mechanisms.

References

[1] D. B. Johnson et al., ”The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)”, draft-ietf-manet-dsr-10.txt, IETF Internet Draft, July 2004.

[2] C. Perkins et al., “Ad hoc On-Demand Distance Vector (AODV) Routing”, IETF RFC 3561, July 2003.

[3] T. Clausen and P. Jacquet, “Optimized Link State Routing Protocol (OLSR)”, IETF RFC 3626, October 2003.

[4] B. Lamparter et al., “Charging Support for Ad Hoc Stub Networks”. Elsevier Journal of Computer Communication, Special Issue on Internet Pricing and Charging: Algorithms, Technology and Applications, 2003.

[5] J. Girão, J. P. Barraca, et al., ”QoS-differentiated Secure Charging in Ad-hoc environments”, International Conference on Telecommunications 2004 (ICT2004), August 2004.

[6] S. Zhong et al., “Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad Hoc Networks”, In Proceedings of IEEE INFOCOM'03, San Francisco, April 2003.

[7] N. Salem et al, “A charging and rewarding scheme for packet forwarding in multi-hop cellular networks”, Proc. 4th ACM Intern. Symposium on Mobile ad hoc Networks & Computing, Maryland, USA, June 2003.

[8] G. Holland, N. Vaidya “Analysis of TCP performance over mobile ad hoc networks”. Wireless Networks, Volume 8 , Issue 2/3 March-May 2002, Selected Papers Mobicom'99, p275-288, 2002, ISSN:1022-0038.

[9] The Network Simulator - ns-2, www.isi.edu/nsnam/ns/ as in Jan. 2005.