The Polynomial-Assisted Ad-hoc Charging Protocol

João Paulo Barraca *jpbarraca@av.it.pt*
Susana Sargento *ssargento@det.ua.pt*
Rui L. Aguiar *ruilaa@det.ua.pt*
Instituto de Telecomunicações, Universidade de Aveiro

Abstract

The area of trustworthy charging in self-organized environments has been developed quite recently. This paper introduces a new secure charging-protocol, the Polynomial-Assisted charging protocol, for these environments. The protocol relies on polynomial composition for speeding the identification process in small groups. This protocol is able to provide strict guarantees of cooperative behavior in traffic forwarding, with minimal network overhead. Protocol performance is evaluated in multiple ad-hoc environments and results are compared with previously proposed work. The performance results show the merits of this protocol in multiple types of environments.

1. Introduction

One of the major concerns for future networks is how to provide Internet access all around the world without any limitations and with reduced costs for the users and providers. As a consequence of this situation, hotspots are appearing all around the globe and in the most different and remote places. This business strategy is profitable both for the provider, which increases its revenues, and for the user, that can be connected to the Internet anytime and anywhere. Mobile Ad-hoc networks may be used to increase the radio coverage of hotspots with low cost and easy deployment.

One of the major requirements for using typical telecommunication services is the charging of services. Thus, when associating ad-hoc networking with traditional telecommunications, charging control is required in the ad-hoc network. However, due to the dynamic nature of ad-hoc networks, with nodes dynamically joining and leaving, this charging is not trivial. Another essential issue in ad-hoc networks is the requirement for mobile nodes to cooperate in traffic forwarding. Due to restrictions of node resources, the forwarding nodes may behave selfishly, not making available its resources to forward the traffic from other nodes. A basic economic idea [1] aims to provide some rewards to nodes that behave appropriately. This rewarding can be a credit increase or any other incentive defined by the operator business model. Rewarding forwarding nodes in an ad-hoc network is much more complex than charging, since the overall path of the packets needs to be known, in order to identify the forwarding nodes. Security mechanisms need also to be in place to make sure that the information of the nodes in the path is not modified in transit.

There are already in the literature some proposals for the charging and rewarding processes in ad-hoc networks connected to infrastructure networks. SPRITE [1], SALEM [2] and Secure Charging Protocol (SCP) [3] are some of the proposals. However, these proposals present a multitude of inconvenients: require the use of non-optimal routes [2] (all traffic crosses the access point), do not assure that the information on the forwarding nodes sent to the infrastructure is the correct one [1] (node can eavesdrop the network, store the proofs gathered and later report them without packets forwarding), or introduce large overhead in the network [3] as the number of hops increase (when the route is not known previously, each intermediate node adds its IPv6 address to the proof).

In this paper we propose a charging and rewarding mechanism, denoted as PACP (Polynomial-Assisted Charging Protocol), able to provide correct charging and rewarding information, securing the processes of proofs creation and delivery, without the need of sub-optimal routes, and with small network overhead and processing requirements in all nodes in the path. More specifically, we propose a polynomial encoding of the information on the intermediate nodes that provides small overhead and assures a very simple forwarding process. The results achieved through simulation results show that the performance of the network, in terms of throughput achieved and network overhead, is much increased in PACP compared to previous solutions (such as SCP [3]). Also, PACP presents increased performance of the charging and rewarding mechanism, in terms of charging and rewarding rates.

This paper is organized as follows. Section 2 presents the proposed charging and rewarding solution and its detailed functioning in several phases, from the

Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)
1530-1346/05 $20.00 © 2005 IEEE

node registration until the charging and rewarding of its traffic in the infrastructure network. Simulation results are presented in section 3, and the conclusions and topics for further work are addressed in section 4.

2. A Solution to Charging Support

In this section we present a charging and rewarding protocol, denoted by PACP, for ad-hoc networks in scenarios with interconnection to the infrastructure network managed by a network provider. One such scenario is an extended hotspot environment, where one (or more) of the ad-hoc nodes is connected to a hotspot, and this interconnection capability is shared to all nodes in the ad-hoc network. Note that this proposal has a wider application scope than this extended hotspot scenario, since it also copes with eventual disconnections to the infrastructure networks: the ad-hoc nodes are able to store a significant amount of charging information that would be then reported to the infrastructure network upon availability of the connection. The PACP protocol is based on SCP concepts (very successful approach in the literature), but improves manifold over all the proposals in the literature [1][2][3][4][5]: it includes the capability for flow admission without the need of sub-optimal routes, the network overhead introduced by the protocol is reduced and the processing requirements in every node are decreased. Moreover, the complexity of the forwarding process is low due to the use of polynomial encoding of the intermediate nodes.

Figure 1 depicts the architecture of the ad-hoc network in the extended hotspot scenario. Inside the ad-hoc network, the traffic is routed through reactive or proactive ad-hoc routing protocols, like Ad-hoc On demand Distance Vector routing (AODV) [7] or Optimized Link State Routing (OLSR) [8]. The ad-hoc network is connected to the infrastructure network through an Access Router (AR). This element is a node (often infrastructure) that routes packets between the external networks and the ad-hoc cloud, collects the charging proofs, both for rewarding and charging, and sends them to the A4C server in the infrastructure network. This A4C server handles all authorization, authentication and charging issues: it receives the proofs containing information on the senders, receivers and forwarding nodes, and processes

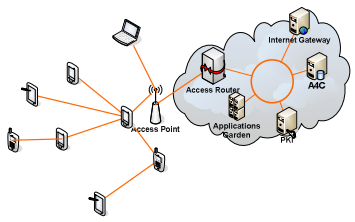


Figure 1 – Ad-hoc Hotspot managed by Operators

the charging and rewarding information. Since the PACP protocol is secure, the ad-hoc nodes need to maintain cryptographic material to be able to send and receive the traffic. This key management is provided by the Public Key Infrastructure (PKI) server. The ad-hoc nodes can access operator services available in the infrastructure network (located in the Application Garden), and can access any node in the Internet through the Internet Gateway.

The rewarding mechanism operates under the assumption that if the nodes between sender and receivers do not have benefits to forward a packet, they will simply start acting selfishly and drop packets instead of forwarding them. This behavior is undesirable both for users, located far from the access point, and for network operators. However, if nodes are rewarded for the packets they forward, they become interested to cooperate in the ad-hoc cloud. To reward the forwarding nodes, the operator, through the A4C, needs to have information on the paths crossed by each packet in the ad-hoc network. This is one of the main challenges of any rewarding protocol: to implicitly (with low overhead), scalable and securely carry and transport information required for the correct charging and rewarding process. Notice that security is of major importance, since we need to assure that the ad-hoc nodes report the correct ad-hoc paths to the A4C.

Briefly, the PACP works as follows (see Figure 2). When a node enters in the ad-hoc network, it goes through a registration process to become known by the AR (and the A4C), to be authenticated and authorized, and to receive the cryptographic material required for the charging process (Registration phase in section 2.2). When communicating with other nodes inside or outside the ad-hoc network, the node implicitly (through polynomial encoding) includes in the data packet the identification of the route that will be updated in each node in the ad-hoc network towards the destination. The fields containing information on the route are fixed size and cryptographically secured, so they cannot be wrongly modified along the path (Participation and Forwarding phases in sections 2.3 and 2.4, respectively). 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 in section 2.5). These proofs implicitly 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 A4C 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 in section 2.6). The A4C is then able to correctly charge the sending (and, eventually, the receiving) nodes, as well as correctly reward the forwarding nodes. In the next sub-sections we detail the security basics required for the protocol operation and the several phases of the protocol. The messages and contents depicted in Figure 2 will be detailed in the following sub-sections.

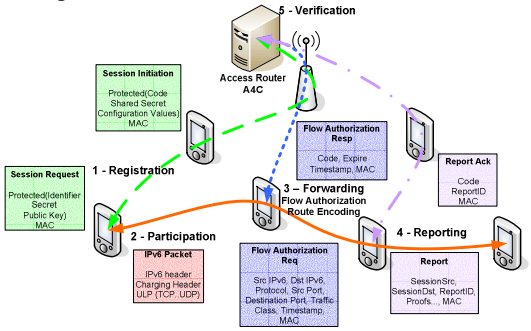


Figure 2 – PACP operation

Although presented in an ad-hoc network concept, the proposed mechanism can also be used in infrastructure networks when distributed control mechanisms are needed.

2.1. Security Assumptions

We consider that all packets contain a Message Authentication Code (*MAC*) used to verify the integrity and ownership of the packets. The network provider has an available public key that can be used by the ad-hoc nodes both to authenticate packets from the AR and to protect control packets sent to the network operator. Because of the limitations of nodes typically composing an ad-hoc network, the cryptographic algorithm and duration of the associated keys need to be carefully evaluated. The algorithm needs to have a very high security per bit ratio and should be easily implemented either in dedicated hardware or in software (ECC has been the algorithm of choice when dealing with low power equipments and low bandwidth environments [9]). We consider the key pair belonging to the operator to be valid for a long period. These keys should be strong enough to avoid impersonation of the trusted equipments belonging to the provider. On the other hand, the keys associated to mobile nodes are only needed during the time the node is connected to the network, and thus the key validity can be small, leading to shorter keys. The use of short keys minimizes the overall overhead introduced by the *MAC*. Also, the delays associated with cryptographic verification and encryption are smaller, increasing network performance and battery life of the mobile nodes.

We assume that all equipments participating in the ad-hoc network are operated by one or more users, each one including a user profile. The profile is composed by identification data, public and private keys, a charging profile and a traffic profile. The user profile should be fully known to the user and partially to the network operator A4C (the operator cannot access the user private key). Based on these assumptions, the PACP mechanism defines the five previously referred phases that compose the charging and rewarding process.

2.2. Registration Phase

When a mobile node joins an ad-hoc network, it needs to be authenticated and authorized in the network before being able to communicate and access the services available. The A4C is the element responsible for this authentication and authorization process. Communication in the registration process is performed through the AR. For brevity sake, we consider that the AR and the A4C server are co-located.

To start the registration phase, the ad-hoc node sends a *Session Request* message to the AR providing its authentication data and public key. The sensitive fields of this message are protected with the network operator public key and signed using the nodes private key. The AR replies with a *Session Initiation* message indicating if the node is allowed to participate in the network. In the case of a positive answer, the node's public key is stored locally and a shared secret is generated and sent in the *Session Initiation* message with the result code and configuration objects. The payload of this message needs to be protected with the node's public key. After this process, the node is ready to send packets into the ad-hoc network.

2.3. Participation Phase

After a successful registration process, the nodes need to include a tracking header (see Figure 3) in all control and data packets sent. This header will be used as a charging and rewarding proof and will carry the identification of the nodes in the route. The tracking header is composed by: a *Control Code* (*Code*), a *Sequence Number* (*SeqN*), a *Route Hash* (*RHash*), a *Hop Counter* (*HopC*), a *Route Identifier* (*RID*), an *Index* (*Xi*), a *Hash Chain* (*Hash*) and a *MAC*. If the sending node is directly connected to the receiving node, the *RHash*, *HopC*, *RID* and *Xi* fields, which are required in the tracking header for route identification, should be omitted to reduce network overhead.

The *Code field* includes External and Optimization bits. The External bit, when 1, identifies the case where the endpoint is inside the ad-hoc network and the sending node is a node in an external network; in this case the charging header is added by the AR to incoming packets. The Optimization bit, when 1, identifies the direct connection between sending and receiving nodes (remove of the route identification related objects). The *SeqN* field is used to avoid Replay Attacks.

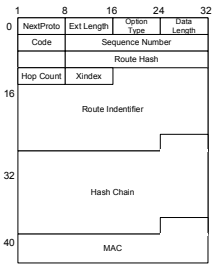


Figure 3 - Packet with tracking header

The *RHash* field is a hash chain with 24 bits computed over the IPv6 addresses of the sending node and interactively computed over the IPv6 address of each (and at each) forwarding node. The *HopC* represents the size of the route. *RHash* and *HopC* provide a rough mechanism to detect route changes.

RID is the field that contains the encoded route. This field represents the values of the polynomial. It is updated at every node (see below) and will be used to reconstruct the path. *Xi* is the polynomial term and will also be used to, together with *RID*, reconstruct the path.

The Hash Chain is a 128 bits value divided in 2 blocks of 64 bits. The first block is called *Charging Block* and the second one *Rewarding Block*. The sending node initializes the entire Hash Chain with the result of the MD5 [11] calculated over a pseudo header (concatenation of the following fields: source address, destination address, traffic class, protocol code, source port, *SeqN* and the node's shared secret). Packets without the tracking header or with an invalid *MAC* will be automatically discarded by forwarding nodes.

2.4. Forwarding Phase

When receiving a packet, each forwarding node needs to validate the *MAC* field and update the fields identifying the route and the *Hash Chain*. Then, the packet can be forwarded.

Forwarding nodes will update the 64 left most bits of the *Hash Chain* (corresponding to the rewarding block) with the less significant bits resulting from a MD5 over a new pseudo header with the same structure as the one used by the sending node. Notice that the values of this pseudo header are all the same but the shared secret, which corresponds to this forwarding node. With this process, each node can assure its identity.

The route is encoded as points in a polynomial that are later reconstructed by the A4C server. This encoding process allows the proofs to be small and with fixed size, reducing the overhead necessary to identify the route and decreasing the complexity associated to variable length packets. Making the proof size invariable of the route length is a major benefit as compared to other proposals. For example, in the case of SCP, the size of the packets is incremented by 16 bytes at each hop (length of the forwarding node's IP), thus requiring recalculation of the TCP checksum at every node, adjustments in the TCP MSS and IP Fragmentation. Other upper layer protocols may even require additional processing or just may be incompatible with SCP. Therefore, the invariability of the packet size implies a significant reduction in the operations associated with the forwarding process.

Route Encoding

We define two sizes for the encoded route identifiers: 64 and 128 bits. Identifiers with 64 bits are used to encode local subnet suffixes or user identification; identifiers with 128 bits can be used as globally valid identifiers or used to provide interoperation with identity approaches like Host Identity Protocol (HIP) [6], where the Host Identity Tag (HIT) is used. Encoding IPv6 addresses as polynomials was first proposed by [10] to solve the IP Traceback problem under Distributed Denial Of Service (DDOS).

The idea behind the polynomial encoding is that for any polynomial f(x) of degree *d* in the prime field *GF(p)*, with *p* being the smallest prime greater than 2^d-1, it is possible to recover *f(x)*, given *f(x)* evaluated at $d+1$ unique points. If IP_i represents the IP address of the *i* forwarding node, and x_i an unique packet identifier in the route *R* with *n* nodes, the A4C Server can recover all the IP addresses if it receives $Fr(x)=IP_1x^{n-1}+IP_2x^{n-2}+\ldots+IP_{n-1}x+IP_n$. Recovering this information is done using inversion of Vandermonde [12] matrixes under a prime field *GF(p)* for a number of packets $d \geq n$. The complexity of the problem is $O(n^2)$ and it can be solved in real-time for the average route length of ad-hoc networks. The recovering process through the Vandermonde matrix is depicted in the following expression.

Following the proposed approach, the sending node sets the *RID* field to 0 and X_i to a random value. Each forwarding node *n* computes $RID(X_i)=[(RID(X_i)*X_i + IP_n) \bmod p]$ and updates the *RID* field in the packet with the new value. This should be performed in chunks of 8 bits and using a *p* value of 257 (smaller prime larger than 2^8). The *RID* will be constructed iteratively in the forwarding

$$\begin{bmatrix} x_1^{n-1} & \cdots & x_1^3 & x_1^2 & x_1 & 1 \\ x_2^{n-1} & \cdots & x_2^3 & x_2^2 & x_2 & 1 \\ x_3^{n-1} & \cdots & x_3^3 & x_3^2 & x_3 & 1 \\ x_4^{n-1} & \cdots & x_4^3 & x_4^2 & x_4 & 1 \\ \vdots & \iddots & \vdots & \vdots & \vdots & \\ x_n^{n-1} & \cdots & x_n^3 & x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} IP_1 \\ IP_2 \\ IP_3 \\ IP_4 \\ \vdots \\ IP_n \end{bmatrix} = \begin{bmatrix} Fr(x_1) \\ Fr(x_2) \\ Fr(x_3) \\ Fr(x_4) \\ \vdots \\ Fr(x_n) \end{bmatrix}$$

Solved in GF(p)

nodes and the A4C server will be then able to reconstruct the path.

One restriction of this approach is that the A4C server needs a number of packets larger or equal to the number of encoded IP addresses to be able to reconstruct the path and identify the forwarding nodes. If the number of packets traversing the same route is smaller than the number of hops, the forwarding nodes will not be rewarded for these packets (the charging process is not affected, and the sender/receiving nodes will be correctly charged). This restriction can be minimized by caching recently reconstructed routes at the A4C server. However, this is not considered to be a limitation of the proposal because nodes will still be interested in cooperating by forwarding packets if they know that they will be rewarded by the majority of the forwarded traffic. Moreover, the simulation results that will be depicted in section 3 confirm that the percentage of times it is not possible to identify the forwarding nodes is very small (below 0.6%).

Flow Authorization

In the registration phase, the AR includes in the *Session Initiation* message the objects that define the requirements for flow admission. Thus, the forwarding node needs to ask the AR to authorize the flows whenever it receives a new packet of an unrecognized flow. To authorize a flow, a forwarding node sends a *Flow Authorization Request* message to the AR specifying the source and destination endpoints (*SrcIP* and *DstIP*), Protocol (*ULProto*), Source and Destination ports (*SrcPort* and *DstPort*), Traffic Class (*TrafCl*) and a Timestamp: *{SrcIP || DstIP || ULProto || SrcPort || DstPort || TrafCl || Timestamp}*. The AR issues a *Flow Authorization Response* message allowing or denying the flow to be forwarded. The answer sent by the AR will take into account the user profile of the endpoints, credit information reported by the A4C server or management policies defined by the network operator. Note that, potentially, this phase could be waived in many scenarios.

2.5. Reporting Phase

Like in SCP, the last forwarding node of each packet flow is responsible for gathering the proofs present in each packet. All information in the proofs is signed to provide protection against changes. Therefore, the node collecting the proofs cannot manipulate the reports in an attempt to obtain some additional profit.

The reports are sent periodically, or as soon as it is reached a number of stored proofs enough to fill a packet with MTU size. If the end-point of a flow is located outside the ad-hoc network, the AR collects the proofs directly and reports them to the local A4C server. Notice that the control packets will be forwarded without additional verification by the intermediate nodes, allowing a multiple hop return channel for the collected proofs. Since all proofs are cryptographically protected, nodes will not be able to alter the proofs without compromising the rewarding process. The proofs are reliably sent to the infrastructure network; if an acknowledgment of the proofs reception is not received, a back off timer is activated and the proofs are sent again.

2.6. Verification Phase

After receiving the proofs, the AR sends them to the A4C server, which will verify its authenticity. If the proofs are authentic, the relevant nodes are charged and rewarded; otherwise, the proofs are discarded.

To verify the sending node (charging validation), the A4C calculates the MD5 sum from the information reported as performed by the sending node, and compares the 64 most significant bits of the result with the 64 most significant bits from the reported *Hash Chain*. If the values differ, the proof is considered to be invalid and it should be immediately discarded. If the values match, the proof is stored.

When the number of reported proofs is, at least, equal to the reported route length, the identification of the forwarding nodes can be performed. In this identification process, the A4C solves the Vandermonde matrix using the values of the *RID* and *Xi* fields and considering the number of hops specified in the route length. The obtained result will be a list of IPv6 addresses that will be repeated every *HopC* terms. The last forwarding node needs to match the IPv6 of the node that reported the proofs. If the *RID* fields are corrupted, the solutions of the matrix will not be unique. For each proof (rewarding validation), the A4C will compute the *Hash Chain*, in the same way performed by the forwarding nodes, and compare the less significant values of each proof with the values calculated. If all these procedures complete successfully, the proofs are considered to be valid and the nodes are rewarded; otherwise, the proofs are discarded.

If the proofs received refer to a route for which there is not enough information to identify the forwarding nodes, the charging procedure is executed and the proof is stored for a period of time. If the timer expires the proof is discarded and no rewarding occurs. If another proof for the same route arrives at the A4C server, they are both cached until the number of proofs stored matches the number of hops reported. When this happens, the rewarding procedure occurs and the new route is cached. The caching of the routes at the A4C

server will increase the efficiency of the rewarding algorithm.

3. Simulation Results

We implemented PACP and SCP in ns-2 [14] with the objective of comparing the performance of these mechanisms and the performance of the network when these mechanisms are active. For this purpose, we developed a mechanism capable of intercepting all IPv6 packets and send these packets to a charging agent for further processing. We also developed the mechanisms for the management, transmission and verification of the proofs. In the simulations performed we considered the reporting period to be 15 sec. Although the charging protocol is independent of the ad-hoc routing protocol, in our simulations we used AODV.

3.1 Simulation Scenarios

In order to evaluate the behavior of the PACP protocol in different situations, three simulation scenarios were created. All of them are composed by 40 ad-hoc nodes, where the first node is the AR (also considered a mobile node). 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. The third scenario has the same characteristics of the previous but the speed of the nodes follows a uniform distribution in the interval between 0 and 1 m/s (3.6Km/h) (RWP-Slow).

The simulations were performed with several traffic patterns, both using TCP and UDP flows. 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 specified mean time interval between calls, and with an average duration exponentially distributed. The characteristics of these flows are summarized in Table 1, representing a total 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 (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

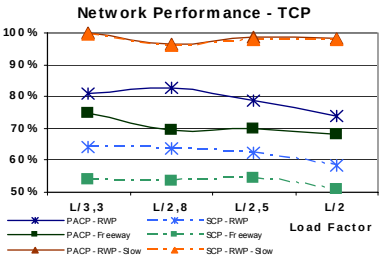
All simulations were divided in two distinct phases. The first phase, between 0 and 1000.0 seconds, contains active traffic flows. At 1000.0 seconds, all data flows stop and only the charging agents remain active in the network during the next period. With this procedure, we verify and analyze the time each protocol requires to flush all pending proofs, reflecting the efficiency of the reporting procedure. To analyze the performance penalty caused by the overhead of the charging mechanisms, simulations were also performed without any charging protocol.

3.2 Network Performance

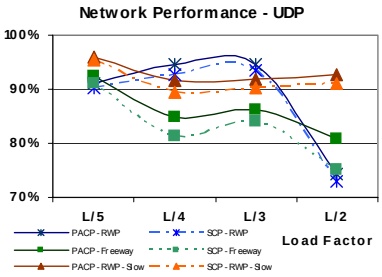
All charging protocols will degrade the performance of the network because of the additional overhead introduced. Since the network performance may be dependent of many different parameters, we used the basic network, without charging protocols, as a baseline, with the same flow generators and movement patterns. We then define network performance as the average throughput of all flows in the network when compared to the network without any charging protocol. A high network performance will result in a high efficiency of the mechanisms operating in the ad-hoc network.

Figure 4a depicts the network performance results obtained with SCP and PACP, for TCP traffic and different scenarios. We observe that TCP flows have best performance, near 100%, in scenarios with higher stability of the routes (RWP – Slow). In this scenario, the slow start and ECN mechanisms [13] have small effect. As observed in the figure, in a stable scenario, the difference of performance between PACP and SCP protocols is negligible. When mobility increases, the performance of the network decreases: the impact of the charging protocol becomes higher and the difference between both proposals also increases. In the freeway scenario with a load factor of $L/2$, the difference between the network performance of both proposals is nearly 20%. Therefore, the impact of PACP is significantly lower than SCP, mainly due to the lower overhead introduced by PACP.

In Figure 4b we depict the same results for UDP flows. The decrease of the network performance throughput with mobility is also noticed in this case. However, the difference between the two proposals is lower, since we do not have the congestion detection and recovering mechanisms of TCP. For the same freeway scenario with load factor *L*/2, the difference is now only 6%. In the scenarios with mobility, there is a decrease of the network performance with increased load, for load factors larger than *L*/3. This behavior is due to the radio link saturation of the AR when a large number of proofs is reported. In overall, the impact in the network performance is lower in PACP and the difference between both proposals increases with mobility.



a) TCP flows



b) UDP flows

Figure 4 - Network performance

3.3. Protocol Overhead

Another important performance parameter is the overhead of each proposal. Since the size of the proofs sent in the data packet grows with the number of hops when SCP is used with AODV, the overhead is calculated as follows:

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

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 PACP proposals in other scenarios is similar).

Figure 5 depicts SCP and PACP overhead with varying traffic loads. We observe that the overhead in SCP is much larger than the one of PACP: in a load of *L*/2, the overhead in SCP is 64%, 40% larger than the one of PACP. This difference clearly identifies the main polynomial encoding advantages of PACP. The overhead also increases with the load in the network. In higher loads, the number of collisions is higher introducing a larger subsequent need for packets retransmission, especially when report packets are retransmitted.

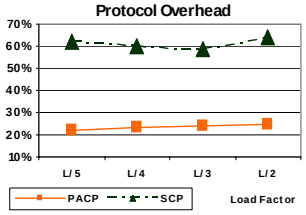


Figure 5 - Protocol overhead

3.4. Charging Rate

The charging rate represents the efficiency of the reporting process. It is defined as the percentage of proofs received at the A4C server from the total of proofs collected, by the forwarding nodes, in the network. In our simulations, the nodes stop moving after the traffic flows stop, at 1000 seconds. For this reason, there is some probability of having nodes with stored proofs that do not have a route to the AR, and then, to the A4C. Therefore, a low charging rate means that proofs were collected, but it was impossible to report those proofs in the specified time. This value of non-reported proofs also depends on how fast a charging protocol reports the proofs, and on the number of proofs accumulated. Also, the higher the throughput of the network, the higher the number of proofs collected and the time required reporting those proofs.

The results obtained in Figure 6 show that the charging rate is directly related with the mobility of the nodes. Higher mobility implies a higher probability of a node finding a short (or direct) route to the AR, sending all the remaining proofs. The RWP scenario with low mobility then presents the worst results. In this scenario there is a high average number of intermediate nodes for a given packet. Since 2/3 of the traffic comes from the AR, nodes far from it store the majority of the proofs and are later unable to report those proofs due to lack of connectivity or saturation of the intermediate nodes. Although not shown due to space limitations, a scenario with TCP flows is less affected, since 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 reduce the throughput of flows located farther from the sending node, thus avoiding higher congestion levels.

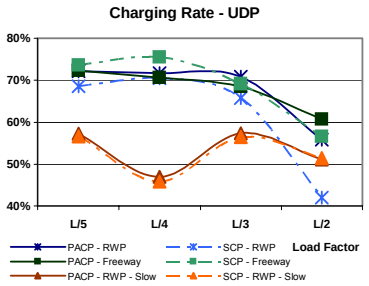


Figure 6 - Charging Rate

3.5. Rewarding Accuracy

As described before, PACP has the drawback of not identifying all the forwarding nodes if the number of proofs reported is less than the number of hops the proof reports. Our implementation made use of a mechanism where routes are cached by the A4C server during a period of time to decrease the number of routes with unknown forwarding nodes.

We define Reward Error as the percentage of times the A4C server is unable to identify the forwarding nodes. Notice that, in this sense, SCP is not affected by errors in the rewarding process because the entire route is added explicitly in every packet. Nevertheless, packets will also be forwarded and not rewarded in SCP because of the ad-hoc network normal behavior (lost connections).

Figure 7 presents the reward error values of PACP for all the scenarios and for UDP and TCP traffic with increasing load factor (different for UDP and TCP). The results show that this reward error is very small, with the highest value of 0.6% obtained in the Freeway scenario. In the RWP scenarios, the errors are smaller than 0.2% of the proofs received.

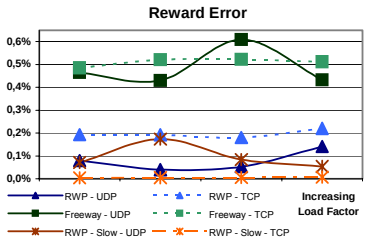


Figure 7 - Routes with unknown intermediate nodes

The results obtained are not strongly affected by the load, while the mobility of the nodes is a major aspect. Some variations with the load are explained by the randomness of a wireless medium with forty nodes. When mobility increases, more changes in the routes exist decreasing the number of proofs per route. When a route of a given flow is very unstable, some routes will not have the number of proofs required to identify all nodes and an error will occur. In some scenarios, the number of errors will be higher in UDP flows. Notice that, as already referred, the mean number of hops in TCP is lower. Therefore, the probability of reward errors is lower, since in a flow with smaller number of hops, fewer proofs are needed to identify the forwarding nodes.

4. Conclusion

We presented an efficient method for charging and rewarding of network services in mobile ad-hoc networks. This method is based on polynomial usage for route identification, providing several advantages over existing proposals: small overhead, lower computational effort, better network usage. In particular, comparing to SCP, our proposal shows clear advantages both in network overhead and in performance of the charging mechanisms.

Further research on the PACP will be focused towards the support of multicast services and pre-paid profiles. In order to evaluate and quantify the performance of the solution in real ad-hoc networks, an implementation of PACP is being developed.

Acknowledgements: This work is in part supported by the EU FP 6 Res. Dev. Daidalos (IST-2202-506997).

References:

[1] S. Zhong et al., "Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad Hoc Networks", IEEE INFOCOM'03, San Francisco, Mar 30 – Apr 3, 2003.

[2] N. Salem et al, "A charging and rewarding scheme for packet forwarding in multi-hop cellular networks", 4th ACM Int. Symp. Mobile ad hoc networking & computing, Jun 2003, Annapolis, Maryland, USA.

[3] 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.

[4] A. Weyland and T. Braun, "Cooperation and Accounting Strategy for Multi-hop Cellular Networks". IEEE Workshop on Local and Metropolitan Area Networks, CA, USA, Apr 2004

[5] L. Buttyn, and J.-P.Hubaux, "Stimulating Cooperation in Self-Organizing Mobile AdHoc Networks". ACM/Kluwer Mobile Networks and Appl. 8(5). Oct 2003.

[6] R. Moskowitz et al., "Host Identity Protocol", Internet Draft v1, draft-ietf-hip-base-01.

[7] RFC 3561, "Ad hoc On-Demand Distance Vector (AODV) Routing".

[8] RFC 3626 "Optimized Link State Routing Protocol (OLSR)".

[9] I. Riedel, "Security in Ad-hoc Networks: Protocols and Elliptic Curve Cryptography on an Embedded Platform", Diplomarbeit, Ruhr-Universität Bochum, Apr 2003.

[10] D. Dean et al, "An algebraic approach to IP traceback", ACM Trans. on Inf. and System Security, v.5 n.2, May 2002.

[11] RFC 1321, "The MD5 Message-Digest Algorithm".

[12] W. H. Press et al, "Numerical Recipes in FORTRAN: The Art of Scientific Computing". Cambridge University Press, 1992.

[13] G. Holland and N. Vaidya. "Analysis of TCP performance over mobile ad hoc networks". In Proceedings of MobiCom, 1999.

[14] The Network Simulator NS2, http://www.isi.edu/nsnam/ns, as in November 2004.