Endeavouring to Be in the Good Books. Awarding DTN Network Use for Acknowledging the Reception of Bundles

Adrián Sánchez-Carmona[a,*], Sergi Robles[a], Carlos Borrego[a]

[a] *Department of Information and Communications Engineering (dEIC), Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain*

Abstract

This paper describes an incentive scheme for promoting the cooperation, and, therefore, avoiding selfish behaviours, in Delay Tolerant Networks (DTN) by rewarding participant nodes with cryptographic keys that will be required for sending bundles. DTN are normally sparse, and there are few opportunistic contacts, so forwarding of other's bundles can be left out. Moreover, it is difficult to determine the responsible nodes in case of bundle loss. The mechanism proposed in this paper contributes to both problems at the same time. On one hand, cryptographic receipts are generated using time-limited Identity Based Cryptography (IBC) keys to keep track of bundle transmissions. On the other hand, these receipts are used to reward altruistic behaviour by providing newer IBC keys. Finally, these nodes need these IBC keys to send their own bundles. When all nodes behave in a cooperative way, this incentive scheme works as a virtuous circle and achieves a Nash equilibrium, improving very much the network performance in terms of latency. The scheme is not difficult to implement, and it can use an already existing IBC infrastructure used for other purposes in a DTN.

Keywords: Incentive Schemes, Delay Tolerant Networks, Nash Equilibrium, Non-repudiation, Receipt exchange, Cooperation

1. Introduction

Delay Tolerant Networks (DTN) [1] are networks with low connectivity rates and high and variable delays. They support two main networking operations: 1) *to route* own traffic, to transmit a message from its origin to any intermediate node; and 2) *to forward* other's traffic, to receive a message, store and carry

*Corresponding author
Email addresses: adria.sanchez@deic.uab.cat (Adrián Sánchez-Carmona), Sergi.Robles@uab.cat (Sergi Robles), Carlos.Borrego@uab.cat (Carlos Borrego)

[2] it for some time to transmit it when it is possible to its destination or to another intermediate node.

In these networks, all nodes are usually interested in routing and use their resources for their own benefit. On the other hand, all nodes demand that others forward their messages, but no one has a special interest in forwarding because it consumes energy and fills buffer space without any direct benefit. Therefore, it is necessary a mechanism to keep track of their behaviour: to know if they are forwarding, if they are refusing to forward or if they are losing or dropping messages. This knowledge about the performed actions of nodes must be used to encourage them to be cooperative and behave for the benefit of the network.

To solve this situation, we created an incentive scheme where nodes are required to forward if they want to route. The incentive scheme is based on a receipt exchange protocol. The receipt exchange protocol makes use of the principles of non-repudiation protocols to provides a way to discover which nodes are suspect of non-cooperative behaviour. The exchanged receipts are used by an incentive scheme that requires nodes to forward if they want to route, and punishes non-cooperative behaviours.

In the presented scheme, nodes need cryptographic keys, not only to forward messages and perform the receipt exchange protocol but also to route their own messages, because running out of keys means becoming isolated. When the incentive scheme detects suspicious nodes, it punishes them by delivering them lesser amounts of keys or even forcing these nodes to wait a while without keys. Therefore, Identity Based Cryptography (IBC) [3] keys act as an enforcing mechanism, because nodes are forced to forward messages to obtain keys, and they want the keys to route their messages.

Our main contributions can be summarized as follows.

- A receipt exchange protocol designed to overcome the limitations that the non-repudiations protocols present when applied in DTNs. The cryptographic receipts are generated by the incentive scheme using IBC keys that are used to track the actions of the nodes.

- An asynchronous incentive scheme for DTN that uses the policy "guilty until proven innocent" to punish and reward the cooperative nodes. This scheme uses the receipts generated by the receipt exchange protocol and rewards nodes by delivering IBC keys to the nodes.

In this article, we proof that, on the presented incentive scheme, node behaviours form a Nash equilibrium when all participants behave in a fully cooperative way. Besides, the simulations show that, even if nodes have low demand of keys and try to be as uncooperative as they can afford, our system improves the performance of the network in terms of latency.

The remainder of this paper is organized as follows: Section 2 presents the related work, in the field of incentive schemas and in the field of non-repudiation protocols and signature exchanges. Section 3 presents a receipt exchange protocol designed to overcome the limitations of non-repudiation protocols when applied to DTNs. Section 4 explains the incentive schema, its asynchronous

operation and how we relate the amount of keys given to the nodes with their balances. Section 5 analyses the choices to be made by the network's participants and demonstrates that all nodes cooperating and being honest form a Nash equilibrium. Section 6 details the performance evaluation. Section 7 details the simulations and presents the obtained results. Finally, Section 8 concludes the article and provides some future lines of research.

2. Related Work

In this section, we will present the state-of-the-art of incentive schemes. As our proposal relies not just on the incentive scheme but also on the receipt exchange protocol to build the chain of custody of every message, we will summarize how other incentive schemes keep track of the actions performed by the nodes to reward them. Finally, we will briefly summarize some non-repudiation protocols, a field that we used to develop the receipt exchange protocol presented in Section 3.

2.1. Incentive schemes

Incentive schemes have been an active research field; Mobile Ad Hoc Networks (MANET) [4] and DTN are usually the kinds of networks where this research is focused.

There are proposals that are heavily related to the concrete application they were designed to solve: dissemination of advertisements, special offers, discount coupons, and so on over a MANET. In [5], a central authority approves and marks each advertisement to track it, nodes that obtain the advertisements deliver receipts to the relaying node, and relaying nodes use these receipts to claim a reward for their work, but the central authority only rewards relaying nodes when the advertisement is used by an end user. *Coupons* [6] is based on the simple idea of adding the name of each relaying node to the transferred coupon, when the coupon is finally used a central authority rewards all nodes that had relayed it. *SMART* [7], is based on the same principles, but it is adapted for general purpose messages in DTN.

The incentive schema called *Pi* [8] includes the policy of payment-rewarding inside each message, giving to the relaying nodes the opportunity to choose, at every message, if the reward will be enough to compensate the usage of resources. As in almost all schemes, a central authority does the credit clearance after the message arrives at its final destination.

Other proposals, such as *Nuglets* [9], are based on the idea of a counter of virtual currency that every node maintains and updates when they send messages, subtracting the cost of sending a message or relaying other's messages, adding a payment for relaying. Obviously, nodes are motivated to cheat and alter the content of the virtual currency counter, therefore these proposals are supported by a trusted and tamper resistant hardware module that provides security to the incentive schema.

In [10, 11] the performance of the network is improved by forcing nodes to exchange messages one by one in a *Barter* manner, this way nodes are incentivized to accept and carry messages they are not interested in but they could exchange later by more interesting ones. In this proposal, nodes are restricted to exchange sets of messages of the same size, and no measures are taken against cheating, so in each transaction one party can deliver one message less than the other without being punished. Selfish nodes could benefit from this weakness to obtain all messages they are interested in without forwarding any other one, performing transactions where they receive one message and do not deliver one.

Several works present incentive schemes that, from a game theory perspective [12, 13, 14], grant that nodes should behave honestly and provide services to others because it is in its own interest. These kind of schemes, like *Sprite* [15], a scheme designed for Ad Hoc Networks, base their operation on the rationality of nodes. In *Sprite*, relaying nodes obtain a receipt of a message together with the message, and deliver the receipt to a central authority. The central authority re-builds the chain of custody of a message to charge the sender and reward the relay nodes when the message arrives at its final destination.

RAPID [16, 17] is a DTN's incentive schema strongly related to a routing algorithm. This proposal, and many others, such as [18, 19, 20, 21] are based on the Tit-for-tat principles: nodes reciprocate good or bad behaviour on part of the peer, they low service to a neighbour when they detect that a neighbour is misbehaving.

This research topic has been studied even from an economic point of view. In [22], the fear of an audit that proves that a node has been misbehaving becomes the only incentive for nodes to behave honestly. A similar approach is used in *iTrust* [23], where the audit is substituted by a probabilistic inspection that reduces a 90% the effort that the *Trusted Authority* has to do. Other works [24, 25] are focused on the global aspects of the network's economy like taxes, inflation, deflation, "feast and famine" cycles, effects of isolation, etc... and usually do not care about how to track the actions performed by each node.

There are proposals that do not try to incentivize selfish nodes to act in an unselfish way, but try to mitigate the impact of such behaviours in the network. For example, in [26], authors try to mitigate routing misbehaviors in DTN using random nodes of the network as witnesses of each transaction to detect nodes that do not relay messages. Then, the results of these observations are used to re-send messages across another path, or to decrease the reputation of selfish nodes.

2.2. Tracking the actions of the nodes

All incentive schemas need to track the actions done by the nodes of the network. It is needed to distribute rewards to nodes with fairness. The most used mechanism is called layered coin.

The layered coin consists of two or more layers, the first, which is also named the base layer, is generated by the source of the message and is sometimes used to indicate payment policies, the class-of-service requirements, or other remunera-

tion conditions. During the subsequent message relaying process, each intermediate node will generate a new layer based on the previous layers by appending a non-forgeable digital signature. This new layer is also called the endorsed layer, which implies that the forwarding node agrees to provide forwarding service.

Using endorsed layers, it is easy to track the propagation path and determine each intermediate node by checking the signature of each endorsed layer, but the layered coin is only complete when a message arrives at its final destination, and intermediate nodes do not have any proof of their cooperation by forwarding a message. The usage of the layered coin always leads to a synchronous schema where relaying nodes are all rewarded at the same time, after the message has arrived at its final destination.

Another mechanism to track the actions of the nodes is the *watchdog*. In [27], each node monitors the next node in the path of a message to check if the message is relayed or not. This solution is related to the characteristics of Ad Hoc Networks and is not applicable in Delay Tolerant Networks.

2.3. Non-repudiation protocols

Our approach is to provide the nodes with a mechanism to obtain a receipt in exchange for its cooperation. The receipt must be changed with the message in a fair way to avoid cheating. A situation where a node obtains a receipt but does not relay the message is as undesirable as a situation where a node relays the message but does not obtain the receipt. This leads us to consider non-repudiation protocols [28].

Non-repudiation protocols provide ways to exchange messages with receipts in a fair way. The majority of the proposals are based on a Trusted Third Party (TTP) that acts as a moderator or intermediary of each transaction, to ensure the protocol is performed correctly by all participants (online TTP) or to repair damages when one participant cheats to obtain an advantage over the other (offline TTP). Proposals that use a TTP, either online or offline, are not viable due to lack of end-to-end connectivity in DTNs.

Non-repudiation protocols without TTP are based on the idea of splitting the message into n parts and send the parts one by one, receiving an acknowledgement for each one [29]. A variation of this idea can be found in [30], where the message is cyphered and sent at the beginning of the transmission and the key needed to decrypt it is sent by parts. These kinds of protocols are called probabilistic because the receiver can, with a probability of $1/n$, guess what part is the last one and there is no need to send the last acknowledgement in order to obtain the whole message. These probabilistic protocols are not viable due to the extremely variable (and usually unpredictable) size of connectivity windows in DTNs.

Unfortunately, to our knowledge, there are not non-repudiation protocols that could be used in Delay Tolerant Networks.

3. Receipt exchange

The core of this proposal is divided into two different and complementary parts, a receipt exchange protocol presented in this section, and an incentive scheme presented and discussed in Sections 4 and 5.

In this section, we explain the two fundamental inputs we have considered during the design of the receipt exchange protocol. Then, we present the notation used during this section, and we provide an extensive description of all the algorithms and steps involved. After this, we explain the evidences created during the execution of the protocol, and we discuss some security aspects of the usage of IBC.

3.1. Receipt exchange protocol's design

The receipt exchange system we propose is based on combining the Fair Exchange Signature Scheme (FESS) [31] with IBC, a cryptographic scheme where the identity of nodes is used to build their public keys. On one hand, we chose this signature scheme because it needs to exchange a low number of messages; it does not requires the involvement of a third party during the transaction; and because when the algorithm finishes, the two signatures arouse and become effective simultaneously. On the other hand, we chose IBC because this cryptographic scheme avoids key management issues in DTN scenarios [32].

However, we have not only combined FESS with IBC. Firstly, and most important, we have transformed a protocol where two nodes sign a document they know beforehand into a protocol where two nodes forward a message and generate evidences about the transaction done. We have achieved this by changing the goal of the protocol and using the last step of the protocol to send the message, instead of a random *keystone*. Besides, we have introduced the concept of a *voucher* as a description of a transaction; and we have modified the structure of the FESS receipts, adding the needed fields to make it store unequivocal information about the transaction they are related. We have made sure that nodes can not reuse past receipts or parts of them on future transactions of the same message. Note that this is something not considered in FESS, where reusing parts of a past receipt to sign the same document again is not a problem. Finally, we have benefited from hash functions properties to optimize the protocol and reduce the amount of space needed by nodes to store the receipts.

3.2. Definitions

Firstly, we present the notation of the elements and the definitions of the concepts used in the receipt exchange protocol. Table 1 contains the notation used to refer to each element and a brief description of its meaning.

A voucher $v = \langle sender, receiver, whosigns, type \rangle$ of a transaction contains four fields: *sender* is the identity of the sender; *receiver* is the identity of the receiver; *whosigns* indicates who is the issuer of the voucher; and *type* is a flag used to indicate the type of the transaction (`origin`, `relay` or `delivery`). From now on, we use *transaction* to refer indistinctly to the next three cases: a message m sent by its origin to any non-final destination node (*type*: origin);

Notation	Description
sk_i	Private key of user i
pk_i	Public key of user i
SK_i	Private IBC key of user i
PK_i	Public IBC key of user i
m	Message
v	Voucher of a transaction
σ	Receipt of a transaction
$H(m)$	Hash function applied on message m
ID_m	Unique identifier of message m
$E_k(m)$	Cypher of m using key k
$D_k(m)$	Decrypt of m using key k
$S_k(m)$	Signature of m using key k
$V_k(m, s)$	Verification of signature s associated to message m with key k

Table 1: Elements used in our receipt exchange protocol.

a message m sent from a node that is not its origin to a node that is not the destination (*type*: relay); and a message m delivered to its final destination from any node (*type*: delivery).

It is important to differentiate between a voucher and a receipt. A voucher is the description of a transaction between two nodes while a receipt contains a voucher and a signature that binds it to the issuer and to the message.

3.3. Algorithms

The receipt exchange protocol that we present in this paper uses of the following algorithms: Algorithm 1, that generates the public key of each participant; Algorithm 2, that generates the exchanged receipts; Algorithm 3, that validates the exchanged receipts; and Algorithm 4, that validates a receipt when executed *a posteriori* by a third node.

Algorithm 1 *SystemSetup*

Input: $\emptyset$
Output: $\emptyset$

1: Choose p and q, big prime numbers so that $q \mid p - 1$.
2: Choose g with order q so that $g \in \mathbb{Z}_p^*$.
3: **for** i *in* $\langle$*All participants*$\rangle$ **do**
4: Generate the pair of keys (sk_i, pk_i) so that $pk_i = g^{-sk_i} \ mod \ p$, where sk_i is the private key and pk_i is the public key.
5: **end for**

Algorithm 2 *FSign*

Input: v: Voucher of the transaction.
pk_A: Issuer's public key.
sk_A: Issuer's private key.
SK_A: Issuer's private IBC key.
PK_B: Receiver's public IBC key.
k: $H(H(m \mid\mid ID_m))$.
Output: $\sigma = \langle a, v, k, s \rangle$: Receipt of the transaction.
1: Choose w so that $w \in \mathbb{Z}_p^*$.
2: Calculate $a = \langle E_{PK_B}(pk_A), S_{SK_A}(H(pk_A)) \rangle$.
3: Calculate $r = g^w \ mod \ p$.
4: Calculate $e = H(a, v, k, r)$ where H is a one way hash function.
5: Calculate $c = w + sk_A e \ mod \ q$.
6: **return** $\sigma = \langle a, v, k, s \rangle$ where $s = \langle r, e, c \rangle$.

Algorithm 3 *SVerify*

Input: $\sigma = \langle a, v, k, s \rangle$: Received receipt, where $s = \langle r, e, c \rangle$ and $a = \langle E_{PK_B}(pk_A), S_{SK_A}(H(pk_A)) \rangle$.
SK_B: Receiver's private IBC key.
Output: **true** or **false**
1: Decrypt $pk_A = D_{SK_B}(E_{PK_B}(pk_A))$.
2: Calculate $r_s = g^c pk_A^e \ mod \ p$.
3: **if** $e == H(a, v, k, r_s)$ **AND** $V_{PK_A}(H(pk_A), S_{SK_A}(H(pk_A)))$ **then**
4: **return true**
5: **else**
6: **return false**
7: **end if**

Algorithm 4 *KVerify*

Input: $\sigma = \langle a, v, k, s \rangle$: Received receipt.
$\langle m \mid\mid ID_m \rangle$: the message and its identifier.
Output: **true** or **false**
1: **if** $SVerify(\sigma) ==$ **true AND** $k == H(H(m \mid\mid ID_m))$ **then**
2: **return true**
3: **else**
4: **return false**
5: **end if**

3.4. Steps of the exchange

Let A be a node that wants to send a message m to node B and wants to generate and exchange the receipts related to this transaction. Figure 1 shows the schema of the protocol, which we explain in detail in the next paragraphs:

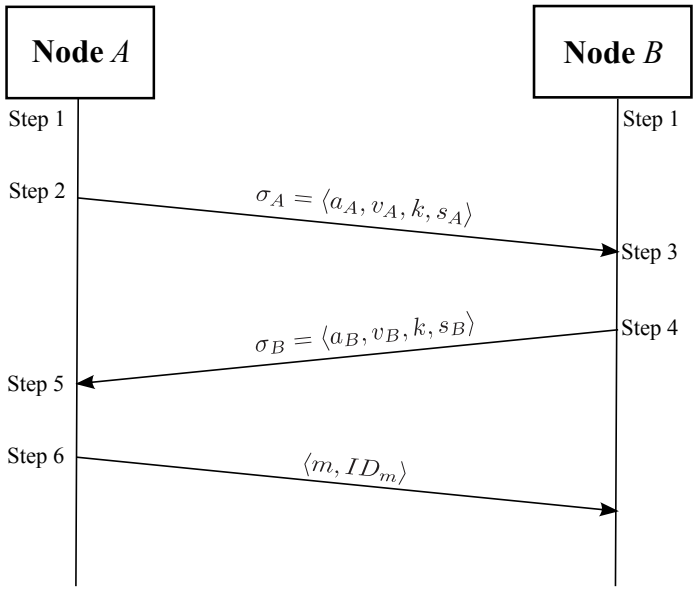

Figure 1: Schema of the receipts exchange protocol. Node A, that initiates the transaction sends the receipt σ_A to B. B receives it, checks if it is correct and sends the receipt σ_B to A. When A receives it and checks its correctness, sends the message, that provides validity to the two receipts.

Step 1. At the deployment phase, before the firsts messages are sent, values p, q and g must be generated by the Public Key Generator (PKG) of the network and delivered to all system nodes. Every node needs, also, its pair of keys $\langle sk_i, pk_i \rangle$, as can be seen in Algorithm 1.

Step 2. Every time node A starts a transmission of message m to node B, the sender begins creating a voucher $v_A = \langle PK_A, PK_B, A, \textit{type} \rangle$. Note that IBC public keys PK_i are used to identify nodes. Then, the sender uses Algorithm 2 to generate the receipt $\sigma_A = \langle a_A, v_A, k, s_A \rangle$ and sends it to B.

Step 3. Node B receives σ_A from A. B checks that the voucher v_A is correct and verifies σ_A using Algorithm 3. If the voucher is valid and the algorithm returns true, B can proceed to Step 4; otherwise, the transmission is aborted.

Step 4. Node B accepts the receipt σ_A issued by A. If B accepts the transaction, then it creates a voucher $v_B = \langle PK_A, PK_B, B, \textit{type} \rangle$, uses Algorithm 2 to generate the receipt $\sigma_B = \langle a_B, v_B, k, s_B \rangle$ and sends it to A.

Step 5. When A receives σ_B, it verifies it using Algorithm 3. A also checks the voucher v_B. If the algorithm returns true and the voucher is correct, it can proceed to Step 6; otherwise, the transmission is aborted.

Step 6. When both nodes have exchanged and verified the receipts, A sends $\langle m, ID_m \rangle$ to B. This last transmission allows B to obtain the message and its identifier $\langle m \mid\mid ID_m \rangle$. Node B verifies that $H(H(m \mid\mid ID_m)) = k$ to accept and end the transaction. The keystone links together each participant with the message itself and the receipt issued on the previous steps. The keystone also provides both nodes unequivocal evidence that they have been in contact because of the transfer of m.

3.5. Created evidences

From the moment the exchange has been completed, B has the receipt σ_A and $H(m \mid\mid ID_m)$, that acts as the keystone. σ_A and $H(m \mid\mid ID_m)$ together form a piece of origin non-repudiation evidence that compromises A as the sender of the message m, so B can prove it has received m from A revealing σ_A and $H(m \mid\mid ID_m)$ to a third party that executes Algorithm 4 and returns true. This way the protocol provides non-repudiation of origin.

Moreover, A has the receipt σ_B and the keystone $H(m \mid\mid ID_m)$. σ_B and $H(m \mid\mid ID_m)$ together form a piece of evidence that A and B have been in contact due to the transfer of m. Node A can prove it revealing σ_B and $H(m \mid\mid ID_m)$ to a third party that executes Algorithm 4 and returns **true**, but cannot prove by itself that the transaction of m has ended correctly.

Notice that, when one of the nodes involved in a two-party transaction acts dishonestly by not forwarding the message, neither of the two evidences are sufficient to prove, unequivocally, which node is guilty and which node is innocent. However, when we reconstruct the chain of custody of this message using receipts from some other two-party transactions, we can identify the two nodes that are suspicious of having lost the message. Then, our incentive scheme punishes both nodes, even when one of them is probably innocent. Later, when we look at the big picture, by reconstructing the chains of custody of lots of messages, we can tell apart the innocent nodes from the guilty ones. The incentive scheme is designed to, in the long run, identify and punish guilty nodes that do not forward messages. We provide discussion about the asymmetric nature of this receipt exchange in Section 4.

3.6. IBC keys obtention: security aspects

The presented proposal has to face all IBC inherent issues related with the obtention of new keys, due to the usage of Identity Based Cryptography. They are out of the scope of this article, so, for the sake of simplicity, we will briefly describe the ones that are related with our incentive scheme and point the suggested method to fight them.

Nodes not always may ask the PKG for new keys before their keys have expired, if this has happened, they do not have a valid secret key that could be used to demonstrate that they are whom they claim to be. To avoid an impersonation attack where a node claims himself to be another, usually with a higher score, when asking for keys at the PKG, the PKG and each node may share a secret and update it, using the Diffie-Hellmann protocol, each time they obtain new keys.

When a new node arrives to the network and wants to become part of it, the node can ask the PKG for a new identity in order to become part of the network. To avoid that a node with a low score could benefit from this and reset it to a higher amount by changing their identity, new identities should be created setting their score equal to the lowest of the network.

If a node loses the secret that proves their identity and their keys expire, it will maintain their identity and do not lose their score only if the network can provide an alternative mechanism that demonstrate the identity of the node; otherwise, it will be treated as a new node.

4. Incentive Scheme

In this section, we explain how to convert the pieces of evidence generated during the receipt exchange phase into proofs of behaviour that could be used by the Incentive Manager. We also discuss the role played by the Incentive Manager, how it rewards and punishes nodes due to their behaviour, delivering a higher or lesser amount of keys, and the policies of the asynchronous incentive scheme.

4.1. Definitions

In order to make the incentive scheme easier to understand, we define a proof of behaviour as a set of receipts and keystones used to prove the behaviour of a node towards one concrete message. We also define *deliverable* proof as any proof that can be delivered to the manager of the incentive scheme.

Notation	Description	Definition
$A \xrightarrow{B} C$	Proof of forwarding. Proves that B has forwarded message m between A and C.	$\sigma_A + \sigma_C + H(m \mid\mid ID_m)$ $(type(v_A) = \texttt{relay},$ $type(v_C) = \texttt{relay})$
$\{A \xrightarrow{B} C$	Proof of forwarding from the origin. Proves that B has forwarded message m between its origin A and C.	$\sigma_A + \sigma_C + H(m \mid\mid ID_m)$ $(type(v_A) = \texttt{origin},$ $type(v_C) = \texttt{relay})$
$A \xrightarrow{B} C\}$	Proof of forwarding to the destination. Proves that B has forwarded message m between A and its final destination C.	$\sigma_A + \sigma_C + H(m \mid\mid ID_m)$ $(type(v_A) = \texttt{relay},$ $type(v_C) = \texttt{delivery})$
$A \rightarrow B\}$	Proof of delivery. Proves that B, the destination of a message m, has received it from A.	$\sigma_A + H(m \mid\mid ID_m)$ $(type(v_A) = \texttt{delivery})$

Table 2: Notation of deliverable proofs. Proofs are formed by the pieces of evidence created during the receipt exchange protocol when sending, forwarding, or delivering a message. We use $type(v_A)$ to denote the value of the field `type` of the voucher v_A inside the receipt σ_A.

Table 2 contains the notation used to refer to each deliverable proof, a brief description of its meaning and the items that compose it, using the notation explained in the previous section.

4.2. Guilty until proven innocent

The receipt exchange protocol we propose is specifically designed to operate in DTNs, it does not need any third party during the transaction, and the number of messages exchanged is minimal. This protocol allows us to obtain non-repudiation proof of origin and reception when all steps are completed. However, when the last message of the protocol is not sent by the sender, discarded by the receiver, or lost during transmission, there is no way of knowing exactly what happened with that transaction. Here, we face the classic Two Generals' Problem[1]: to be sure that the last acknowledgement has been sent and received, another acknowledgement needs to be sent, but that new last acknowledgement has the same issue, and so on.

To deal with this, our incentive scheme uses the policy "guilty until proven innocent", meaning that the two nodes involved with the loss of a message will be marked as suspicious nodes and punished until it is demonstrated that they behaved honestly. From the moment their innocence is proven, punishment is removed, and they are rewarded for their behaviour.

4.3. System parameters

Conceptually, the incentive scheme is very simple. Nodes are rewarded or punished with Cooperation Points (CP) depending on their actions. We define Level of Cooperation (LC) as the amount of CP that a node has obtained with its behaviour and from now on we will use this terminus or its acronym indistinctly. The amount of CP that it is added to or subtracted from the LC of the nodes is defined by the next three parameters: α, β and ϵ, which have the following meaning:

- α: The punishment applied to nodes that are suspicious of not forwarding a message, α must satisfy[2] $\alpha > 0$.
- β: The reward given to nodes when it is proven that they have forwarded a message. The value of β must satisfy $\beta > 0$.
- ϵ: The reward given to a node for each proof delivered to the manager. The value of ϵ must satisfy $\epsilon > \frac{13}{20}\alpha + \frac{4}{5}\beta$.

[1] Suppose there is a valley surrounded by two hills. General A_1 is on one. General A_2 is on the second hill. The enemy, B, is in the valley. If either A_1's or A_2's army attacks B independently they would lose, but together they would win. The problem for A_1 is to communicate a coordinated attack time to A_2 and be sure that A_2 received the message. A_2 also needs to know the acknowledgment got through to avoid attacking alone and lose the battle.

[2] Restrictions for the values of α, β and ϵ have been chosen in order to satisfy the Nash equilibrium. See Section 5 for more details.

It is important to note that, when it is proved that a node had forwarded a message, it is not considered suspicious anymore, so it is rewarded with $\alpha + \beta$ CP (remove the punishment and add the reward).

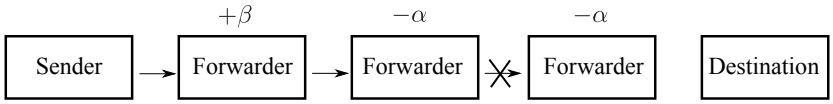

Figure 2: Illustration of the reward and punishment scheme. The arrows depict the relays of the message. The first node, the sender of the message, is not rewarded nor punished. The second node is rewarded with $+\beta$ CP because it is a confirmed relay. The third and the fourth nodes are the two last nodes of the chain of custody, so they are suspected of having lost the message; therefore, they are punished with $-\alpha$ CP.

Figure 2 illustrates the basic idea of the incentive scheme, and how the suspicious nodes are punished while other nodes are rewarded if it is proven that they have forwarded a message.

4.4. Reward and punishment

The chain of custody of every message is indexed by its message identifier ID_m, and it is updated automatically using the proofs of forwarding or the proofs of delivery by applying the following rules:

- Punish with $-\alpha$ CP those nodes that have become suspected of not complying as a result of the last update. Suspicious nodes are the last nodes and the second-to-last nodes of each chain of custody.
- Reward with $+\alpha$ CP those nodes that were suspicious before the update but not after it.
- Reward with $+\beta$ CP those nodes that have become a confirmed relay with the last update. A confirmed relay is a node that is, at least, the third last node of a chain of custody, or any node in the chain of custody of a message that has been delivered. There are two exceptions that do not obtain this reward:
  - The sender, the node that has created the message.
  - Any node that has appeared at least once before in the chain of custody. This way we avoid that nodes can obtain high amounts of CP by colluding with other nodes to forward a message between them an arbitrary amount of times.
- Finally, reward the node that has delivered the proof with $+\epsilon$ CP if the chain of custody has changed thanks to it.

Note that uncooperative nodes that do not accept messages to forward them, do not obtain CP, and that nodes that drop messages are punished with $-\beta$ for every lost message. This punishment is proportional to the damage done to the

overall performance of the network by each one of these two behaviours. Figure 3 shows an example of updating a chain of custody and rewarding nodes when a new proof is used. We provide lots of examples in Section 5.

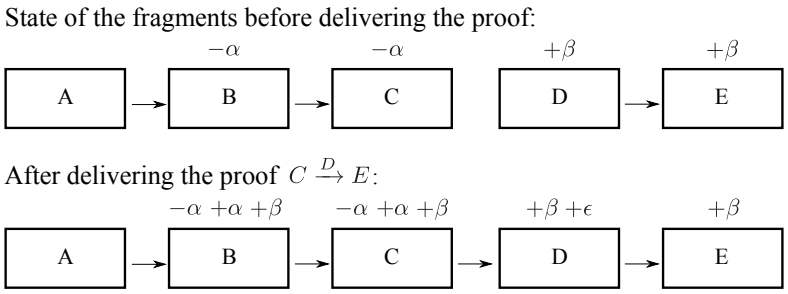

Figure 3: Example of updating a chain of custody. On the upper chain of custody, nodes B and C are suspected of having lost the message, so they are punished with $-\alpha$ CP. Then, node D delivers $C \xrightarrow{D} E$, D gains the reward $+\epsilon$ CP for delivering a valuable proof, and B and C are now confirmed relays (they are previous steps of the confirmed relay D). Therefore, B and C are rewarded with $+\alpha + \beta$ CP to remove the punishment and reward their behaviour. Note that a proof delivered by D that involves C and E has affected too the Level of Cooperation of node B.

4.5. Incentive Manager and the Enforcing Mechanism

IBC-based DTNs are based on the assumption that nodes will, eventually, connect with the Private Key Generator to obtain a set of private IBC keys. Besides, as seen in [33], some DTN routing protocols base their operation on some infrastructure assistance, either via mobile data mules or through the deployment of stationary nodes.

Our proposal benefits from this assumption and uses an offline third-party called Incentive Manager (IM), which must be located on the same node that acts as the PKG[3]. The IM receives the proofs of all transactions, tracks the chain of custody of every message and rewards or punishes nodes due to their behaviour. This way, nodes use the trip to upload proofs and to obtain new IBC keys.

IBC systems usually use keys with a small duration. Depending on the needs of the nodes, the PKG may deliver them sets of a high number of their next keys instead of delivering them only the next one. This way nodes do not need to contact the PKG during a while and can continue routing their messages without running out of keys. Therefore, nodes prefer to obtain more keys when they contact the PKG because this way they obtain more independence and they can operate for more time without asking the PKG again for more keys.

[3]In networks where an alternative communication channel exists, two or more Incentive Managers and/or PKGs can coexist, using this channel to connect between them in order to share the state of the chain of custody of all messages.

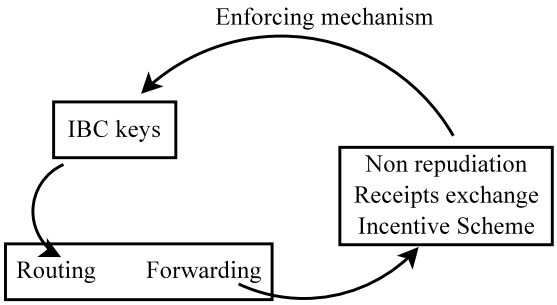

Figure 4

As an enforcing mechanism for our incentive system, we relate the amount of IBC keys given to a node to its Level of Cooperation, as briefly depicted in Figure 4. We calculate K, defined as the number of keys of a fixed duration given by the IM to a node in the basis of the LC of the demanding node related to the LC of all other nodes.

We define max_{LC} as the higher LC of the network and min_{LC} as the lower LC of the network. Then, we normalize the Level of Cooperation of the demanding node inside the interval $[min_{LC}, max_{LC}]$ and map that value to its corresponding value inside the interval $[min_K, max_K]$, defined by the minimum and maximum possible values of K. This way nodes will not be excluded of the network because they will obtain, at least, min_K and the possibility of increase their LC using these keys. This procedure is formalized in Equation 1.

$$K = min_K + \frac{LC - min_{LC}}{max_{LC} - min_{LC}}(max_K - min_K) \tag{1}$$

Note that every node will approximately have the same opportunities to forward messages as their neighbours. A node that does not forward because it has no chance will not obtain any CP, but their neighbours neither, so it will not be punished when keys are delivered based on their relative LC. But a node that decides not to forward messages will remain with the same LC, but will obtain a lesser amount of keys on the basis of their relative LC because their neighbours will probably be forwarding messages and obtaining CP.

Besides, we use an exponential decay function to gradually decrease Nodes' Levels of Cooperation over time. This way we avoid *selfish bursts*, defined as the behaviour of a node accumulating CP and then using it to behave in a very uncooperative way without being punished. Decreasing the LC of all nodes periodically, we allow nodes with a very small (or negative) LC to quickly recover from their past uncooperative behaviour if they start being cooperative, because past actions will weigh less than present actions. With that purpose, we update the LC of every node every t seconds using Equation 2, using the previous value LC_{-1} of their Level of cooperation to calculate the new value LC_t, where T is the time constant.

$$LC_t = e^{-t/T} LC_{-1} \tag{2}$$

Finally, our system includes a mechanism designed to ward off nodes that conform with receiving $K \simeq min_K$. When the IM calculates K and it is lesser than a threshold thr_K, then the IM gives no key to the demanding node and waits an amount of time defined by $tout_K$. When the node demands keys again after $tout_K$ seconds, the IM gives him thr_K keys.

4.6. Asynchronous incentive scheme

To the best of our knowledge, this is the only proposal where the incentive manager works asynchronously and does not need to build the whole chain of custody before rewarding nodes. Every time a node delivers a proof to the IM, it updates the state of the chain of custody, and it distributes rewards and punishments as if the new state will be the last.

This signifies that, despite heavy punishment is applied to a node when other nodes deliver proofs - because it has become suspicious for not forwarding lots of messages - it does not matter. This is so because own node's balance is recalculated when it contacts the IM and uploads its proofs, and all punishment will be removed if it proves that it has forwarded all the messages.

5. Nash Equilibrium

In this section, we start by making some assumptions about the rationality of the network's participants their knowledge about other's behaviours. Afterwards, we discuss the different strategies than can be played by nodes. Finally, we demonstrate that all nodes behaving honestly, forwarding messages, and delivering proofs of forwarding and proofs of delivery, form a unique Nash Equilibrium.

5.1. Previous assumptions

We start assuming that nodes cannot make guesses about other's Levels of Cooperation. Besides, we assume all participants of the network to be rational [34]. As rational nodes never behave in a way that could turn against their interests, they always want to maximize its utility function. During this section, we define the utility function of nodes as their Level of Cooperation[4].

Moreover, when a node delivers a proof to the IM, it is rewarded or punished depending on the previous state of the chain of custody. The chain of custody tracked by the IM is updated every time a node delivers a proof. Therefore, it depends on the proofs previously delivered by other nodes. This means that a node can not know *a priori* the state of the chain of custody. During all the

[4]In Section 7 we consider a slightly different utility function where nodes try not to maximize Level of Cooperation, but to obtain at least a set number of keys each time they contact the PKG.

current section, we will assume that nodes cannot make any guesses about the state of the chain of custody. As a result, we will consider all possible states as equally likely.

Summarizing, nodes are interested in obtaining a higher Level of Cooperation and act according to their interest, and no guesses can be made about the state of the chains of custody.

5.2. Simplified notation

In all tables of this section we used a simplified notation $X \dots YZ$ to summarize a chain of custody in which X is the first known custodian node, and Y and Z are the second-to-last and the last nodes. We use '...' to represent a chain of custody where no node matches the nodes involved in the received proof.

5.3. Game Theory Analysis

In order to analyse the game generated by the presented incentive scheme, we have split the whole game into a set of subgames. Every subgame takes into consideration one of the decisions that a participant of the network has to take: to participate on the network; to accept other's messages to forward them; to cheat or be honest when exchanging receipts; to deliver proofs of forwarding to the IM; and to deliver proofs of delivery to the IM. Therefore, a strategy profile of one player for the whole game contains his strategy profile for every subgame.

Definition 1. *The game generated by the presented incentive scheme is*

$$G =< N, \{s_i\}, \{\pi_i\}, \{p_i\} > .$$

- $N = \{N_0, N_1, \dots, N_n\}$ *is the set of the nodes of the network.*
- $s_i = \{s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}\}$ *is the strategy set of the player* N_i*, where every* s_{ij} *corresponds to the strategy player* i *has chosen for subgame* j*, every strategy set contains two actions, one cooperative and one selfish, that will be explained in next subsections.*
- π_i *is the payoff of the ith player* N_i*, and it is measured in CP.*
- $p_i = \{p_{i0}, p_{i1}, \dots, p_{i4}\}$ *is a mixed strategy for player* i*, where* $p_{ij} = \{p^i_{cj}, 1 - p^i_{cj}\}$ *is a mixed strategy for player* i *for subgame* j*, and* p^i_{cj} *denotes the probability of player* i *of acting cooperatively in subgame* j*.*

5.3.1. Subgame 0. The dilemma of participating

A node that chooses to not participate (NP) never accepts messages not addressed to him. Therefore, it will never obtain CP for forwarding or for delivering proofs of forwarding because it does not generate any. A node that behaves this way will not increase its Level of Cooperation, so, its benefit will be 0 CP. On the other hand, a node that decides to participate (P) will obtain a payoff π_i(P) that depends on the outcomes of subgames 1 to 4. Table 3 provides the payoff matrix for this subgame.

		Any other node j	
		$\mathbf{P}(q^j_{c0})$	$\mathbf{NP}(1-q^j_{c0})$
node i	$\mathbf{P}(p^i_{c0})$	$\pi_i(\mathrm{P})$, $\pi_j(\mathrm{P})$	$\pi_i(\mathrm{P})$, 0
	$\mathbf{NP(1\text{-}}p^i_{c0}\mathbf{)}$	0, $\pi_j(\mathrm{P})$	0, 0

Table 3: The payoff matrix of subgame 0.

Theorem 1. *All nodes choosing to participate (P) with probability $p^i_{c0} = 1$ form a unique Nash equilibrium for subgame 0 if and only if they behave in a way that allow them to obtain more than 0 CP.*

Proof. Node i obtains $\pi_i(\mathrm{P}) \cdot q^j_{c0} + \pi_i(\mathrm{P}) \cdot (1 - q^j_{c0}) = \pi_i(\mathrm{P})$ when chooses to participate (P), and $0 \cdot q^j_{c0} + 0 \cdot (1 - q^j_{c0}) = 0$ when chooses not to participate (NP).

$$\pi_i(\mathrm{P}) > \pi_i(\mathrm{NP});$$
$$\pi_i(\mathrm{P}) > 0$$

Therefore, if $\pi_i(\mathrm{P}) > 0$, then i will always obtain a higher payoff by playing P. As subgame 0 is symmetric, the same applies to any other node.

□

Throughout the current section, we will demonstrate that there are profiles that grant nodes $\pi_i > 0$ CP no matter what other nodes do.

5.3.2. Subgame 1. The dilemma of forwarding

A node that accepts a message to forward but does not forward (NF) it will be punished with $-\alpha$ CP when any proof of forwarding that marks i as a suspicious node is delivered to the IM. On the other hand, a node that decides to forward (F) a message will be eventually rewarded with $+\beta$ if the next node forwards the message too. But it will be punished with $-\alpha$ if the next node decides to not forward the message because it will be considered by the IM as a suspicious node.

		other node j, playing second	
		$\mathbf{F}(q^j_{c1})$	$\mathbf{NF}(1-q^j_{c1})$
node i	$\mathbf{F}(p^i_{c1})$	β, play again as first player	$-\alpha$, $-\alpha$
plays first	$\mathbf{NF(1\text{-}}p^i_{c1}\mathbf{)}$	$-\alpha$, 0	$-\alpha$, 0

Table 4: The payoff matrix of subgame 1.

This subgame is sequential and asymmetric because the decision of the second node does not matter unless the first one chooses to forward (F) the message. Then, if the second node plays NF, both nodes are punished with $-\alpha$. But if it plays F, his payoff is determined by playing the same subgame once again, with

the second node playing first and a third node (the next hop) playing second. Table 4 provides the payoff matrix for this subgame.

Theorem 2. *All nodes choosing to forward messages (F) with $p^i_{c1} = 1$ form a unique Nash equilibrium for subgame 1.*

Proof. Node i obtains $\beta \cdot q^j_{c1} - \alpha \cdot (1 - q^j_{c1})$ when chooses to forward (F), and $-\alpha \cdot q^j_{c1} - \alpha \cdot (1 - q^j_{c1}) = -\alpha$ when chooses NF. Therefore, his payoff is higher when it plays F than when it plays NF because both α, β and q^j_{c1} are defined to be positive.

$$\pi_i(\text{F}) > \pi_i(\text{NF});$$

$$\beta \cdot q^j_{c1} - \alpha \cdot (1 - q^j_{c1}) > -\alpha;$$

$$q^j_{c1} \cdot (\beta + \alpha) > 0$$

Therefore, for the first player, playing F is a dominant strategy.

By backward induction, the second player will play F, because it is the only way to obtain a positive payoff, because playing NF will grant him $p^i_{c1} \cdot -\alpha + 0 \cdot (1 - p^i_{c1}) = -\alpha \cdot p^i_{c1}$, which is negative in any case. □

5.3.3. Subgame 2. The dilemma of cheating

When forwarding a message, an honest node (H) finishes the receipt exchange protocol by sending the message. On the other hand, a node can try to cheat (NH) by starting the exchange receipt protocol to obtain a proof of forwarding but does not sending the last message. This way the transaction remains unfinished because the receiver has not received the message. A node that behaves this way could deliver the gathered proofs to obtain ϵ CP for each one. An honest (H) node could deliver the gathered proofs to obtain not only ϵ CP, but also α CP for no longer being suspicious and β CP for becoming a confirmed relay. This subgame is a one-player game because only the part that starts the transmission con cheat the other part.

Last state	New state	A's reward	B's reward	C's reward	Y's reward	ρ	ϕ
...	$A...BC$	$+\beta$	$+\epsilon - \alpha$	$-\alpha$		1/3	1/5
$C...XY$	$A...XY$	$+\beta$	$+\epsilon + \beta$			0	1/5
$X...YA$	$X...BC$	$+\alpha + \beta$	$+\epsilon - \alpha$	$-\alpha$	$+\alpha$	1/3	1/5
$B...XY$	$A...XY$	$+\beta$	$+\epsilon$			0	1/5
$X...AB$	$X...BC$	$+\beta + \alpha$	$+\epsilon$	$-\alpha$		1/3	1/5

Table 5: Possible situations when delivering $A \xrightarrow{B} C$. A cheater node B would not obtain the reward $+\beta$ CP for forwarding the message, and will always be punished with $-\alpha$ CP for being suspected of having lost the message.

Table 5 shows all possible states of the chain of custody, how it is updated, and the reward or punishment given by the IM to each participant when node B

delivers a proof of forwarding $A \xrightarrow{B} C$. Last two columns correspond to ρ, the probability of the situation if B is a node that plays NH, and ϕ, the probability of the situation if B is a node that plays H. Notice that there are situations with $\rho = 0$, these are situations that can only exist if B forwards messages in an honest way (H). Taking this into account, Table 6 provides the payoff matrix for this subgame.

		other node receiving the message
node i	$\mathbf{H}(p^i_{c2})$	$(\epsilon + \frac{1}{5}\beta - \frac{2}{5}\alpha)$, $-\alpha$
	$\mathbf{NH(1\text{-}}p^i_{c2}\mathbf{)}$	$(\epsilon - \frac{2}{3}\alpha)$, play as first player

Table 6: The payoff matrix of subgame 1.

Theorem 3. *Any node that is honest (H) with $p^i_{c2} = 1$ form a unique Nash equilibrium for subgame 2.*

Proof. Node i obtains $\epsilon + \frac{1}{5}\beta - \frac{2}{5}\alpha$ when chooses to be honest (H), and $\epsilon - \frac{2}{3}\alpha$ when chooses NH. Therefore, his payoff is higher when it plays H than when it plays NH because both α and β are defined to be positive.

$$\pi_i(\text{H}) > \pi_i(\text{NH});$$

$$\epsilon + \frac{1}{5}\beta - \frac{2}{5}\alpha > \epsilon - \frac{2}{3}\alpha;$$

$$\beta > -\frac{4}{3}\alpha$$

Therefore, playing H is a dominant strategy. □

5.3.4. Subgame 3. The dilemma of delivering proofs of forwarding

A node can choose to deliver proofs of forwarding (PF) in order to be rewarded by the IM. Besides, a node can choose to not deliver them (NPF) and wait until the next node delivers a proof of forwarding that provides him a reward.

This way, a node that plays NPF avoids punishment $-\alpha$ CP and will eventually be rewarded with β CP for forwarding the message, but it never obtains $+\epsilon$ CP for delivering proofs. On the other hand, a node that plays PF will be considered suspicious and be punished with $-\alpha$ CP sometimes, but it will obtain $+\epsilon$ CP every time that delivers a proof.

Table 7 has the same structure as Table 5. The last columns correspond to μ, the probability of the situation if A is a node that never delivers proofs of forwarding and λ, the probability of the situation if B is an honest node that always delivers the proofs. Notice that there is only one situation with $\mu = 0$, this is a situation that can only exist if A has delivered a proof previously. Taking this into account, Table 8 provides the payoff matrix for this subgame.

Last state	New state	A's gain	B's gain	C's gain	Y's gain	μ	λ
...	$A...BC$	$+\beta$	$+\epsilon-\alpha$	$-\alpha$		1/4	1/5
$C...XY$	$A...XY$	$+\beta$	$+\epsilon+\beta$			1/4	1/5
$X...YA$	$X...BC$	$+\alpha+\beta$	$+\epsilon-\alpha$	$-\alpha$	$+\alpha$	1/4	1/5
$B...XY$	$A...XY$	$+\beta$	$+\epsilon$			1/4	1/5
$X...AB$	$X...BC$	$+\beta+\alpha$	$+\epsilon$	$-\alpha$		0	1/5

Table 7: Possible situations when node B delivers a proof of forwarding $A \xrightarrow{B} C$ to the IM. Node B will obtain a higher gain than a node A that does not deliver any proofs and trusts that others will deliver proofs that increase A's Level of Cooperation.

		next node j, playing second	
		PF(q^j_{c3})	**NPF**($1-q^j_{c3}$)
node i	**PF**(p^i_{c3})	$(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)$,$(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)$	$(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)$, $(\beta+\frac{\alpha}{4})$
plays first	**NPF(1-**p^i_{c3}**)**	$(\beta+\frac{\alpha}{4})$, $(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)$	0, $(\beta+\frac{\alpha}{4})$

Table 8: The payoff matrix of subgame 3.

Theorem 4. *All nodes choosing to deliver proofs of forwarding (PF) to the IM with $p^i_{c3}=1$ form a unique Nash equilibrium for subgame 3 if $\epsilon > \frac{4}{5}\beta+\frac{13}{20}\alpha$.*

Proof. The first player obtains $(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)\cdot q^j_{c3}+(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)\cdot(1-q^j_{c3}) = (\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)$ when plays PF, and $(\beta+\frac{\alpha}{4})\cdot q^j_{c3}$ when chooses NPF. Therefore, his payoff is higher when it plays PF than when it plays NPF if and only if

$$\pi_i(\text{PF}) > \pi_i(\text{NPF});$$

$$\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha > (\beta+\frac{\alpha}{4})\cdot q^j_{c3};$$

$$\frac{\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha}{\beta+\frac{\alpha}{4}} > q^j_{c3}$$

Being $0 \le q^j_{c3} \le 1$, if the numerator $(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)$ is higher than the denominator $(\beta+\frac{\alpha}{4})$, then this equation holds.

$$\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha > \beta+\frac{\alpha}{4};$$

$$\epsilon > \frac{4}{5}\beta+\frac{13}{20}\alpha$$

This subgame is asymmetric, so, the second player obtains $(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)\cdot p^j_{c3}+(\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)\cdot(1-p^j_{c3}) = (\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha)$ when plays PF, and $(\beta+\frac{\alpha}{4})\cdot p^i_{c3}+(\beta+\frac{\alpha}{4})\cdot(1-p^i_{c3}) = (\beta+\frac{\alpha}{4})$ when chooses NPF.

As we have demonstrated $\epsilon+\frac{\beta}{5}-\frac{2}{5}\alpha > \beta+\frac{\alpha}{4}$ above, then we can state than playing PF is a dominant strategy for both players. □

Corollary 4.1. *The incentive scheme also grants that an honest (H) node will obtain a positive payoff while delivering a proof of forwarding (PF).*

Proof. The payoff obtained by a node that plays H and PF is positive if the next equation holds.

$$\pi_i(\mathrm{H}, \mathrm{PF}) > 0;$$

$$\epsilon + \frac{1}{5}\beta - \frac{2}{5}\alpha > 0;$$

$$\epsilon + \frac{1}{5}\beta > \frac{2}{5}\alpha$$

Given that ϵ is defined to be higher than $\frac{13}{20}\alpha$, then $\epsilon > \frac{2}{5}\alpha$ also holds because $\frac{13}{20}\alpha > \frac{2}{5}\alpha$, so, this equation holds for any value of α and β even when $\beta \approx 0$. □

5.3.5. Subgame 4. The dilemma of delivering proofs of delivery

To obtain proofs of forwarding from the origin or proofs of delivery is important for the IM. When a chain of custody is completed, the IM can delete it and free resources. To a node, there are no differences between forwarding a message from its origin than forwarding it from any other node. The same way, there are no differences between the reward obtained by delivering any proof of forwarding. Therefore, decisions involving proofs of forwarding from the origin are covered by subgames 1 and 3.

A node that has received a message can deliver (PD) the proof of delivery in order to be rewarded by the IM. On the other hand, a node that has received a message can chose not to deliver (NPD) this proof. This way it will never recover from the punishment $-\alpha$ for being considered suspicious, and the same happens to its previous node on the chain of custody. This subgame is a one-player game because only the destination node has the proofs of delivery and can choose to deliver them or not.

Last state	New state	A's gain	Previous B's reward	B's gain	Y's gain	φ
...	$A...AB\}$	$+\beta$		$+\epsilon$		1/3
$X...YA$	$X...AB\}$	$+\alpha+\beta$		$+\epsilon$	$+\alpha+\beta$	1/3
$X...AB$	$X...AB\}$	$+\alpha+\beta$	$-\alpha$	$+\epsilon+\alpha$		1/3

Table 9: Possible situations when delivering $A \rightarrow B\}$. The message has arrived at its destination and all implied nodes are rewarded.

Table 9 shows all possible situations that can occur when node B delivers a proof of delivery $A \rightarrow B\}$. Let φ be the probability of every possible state of the chain of custody at the moment of delivery. Taking this into account, Table 10 provides the payoff matrix for this subgame.

Theorem 5. *Any node that delivers proofs of delivery (PD) with $p^i_{c4} = 1$ form a unique Nash equilibrium for subgame 4.*

		previous node $(i-1)$ on the chain of custody
node i	**PD(p^i_{c4})**	$(\epsilon + \frac{1}{3}\alpha)$, $(\frac{2}{3}\alpha + \beta)$
	NPD(1-p^i_{c4})	$(-\frac{1}{3}\alpha)$, $-\alpha$

Table 10: The payoff matrix of subgame 4.

Proof. Node i obtains $\epsilon + \frac{1}{3}\alpha$ when plays PD, and $-\frac{1}{3}\alpha$ when chooses NPD. Therefore, his payoff is higher when it plays PD than when it plays NPD because both α and ϵ are defined to be positive.

$$\pi_i(\text{PD}) > \pi_i(\text{NPD});$$
$$\epsilon + \frac{1}{3}\alpha > -\frac{1}{3}\alpha$$

Therefore, playing PD is a dominant strategy. □

Corollary 5.1. *The incentive scheme also grants that a node that delivers a proof of delivery (PD) will obtain a positive payoff, and that this will cause another node which has forwarded (F) the message to obtain a positive payoff.*

Proof. The payoff obtained by a node that plays PD is positive because both α and ϵ are positive.

$$\pi_i(\text{PD}) > 0;$$
$$\epsilon + \frac{1}{3}\alpha > 0$$

The payoff obtained by the previous node, which has played F, when node i plays PD, is positive because both α and β are also positive.

$$\pi_{i-1}(F) > 0;$$
$$\frac{2}{3}\alpha + \beta > 0$$

□

5.4. Nash equilibrium

From a game theory perspective, every node of the network has to choose a global strategy, which consists of picking a strategy for each of the subgames.

As we have proven in this section, for any node, choosing a strategy that consists of accepting messages addressed to others (P), forwarding these messages (F), being honest during the receipt exchange protocol (H) and delivering to the IM all kind of proofs (PF and PD) is a strictly dominant strategy. This behaviour grants the node higher benefits than any other possible strategy, and it does not matter what the other nodes of the network do.

Consequently, a profile of strategies where all nodes choose to behave this way form a unique Nash equilibrium because it is impossible for any node to increase its profits by deviating from this strategy.

6. Performance evaluation

In this section we present some details about the proof-of-concept we have implemented. Then, we provide measurements of the computational overhead introduced by the receipt exchange protocol. Finally, we study if the computational overhead introduced by this protocol is affordable for the network.

6.1. Overhead calculation

As a proof-of-concept and in order to obtain a measure of the overhead that the receipt exchange protocol adds to every transaction, we have deployed an implementation of the receipt exchange protocol on two Raspberry Pi devices[5]. We have used this implementation to send 250 messages of sizes between 1MB and 5MB and measured the performance of the protocol and the time needed to compute and exchange the receipts.

Message size	Transmission time without the receipt exchange protocol		Transmission time with the receipt exchange protocol		Absolute overhead		Relative overhead
	Average	σ	Average	σ	Average	σ	
1MB	1.07s	0.14s	4.23s	0.26s	3.16s	0.19s	294%
2MB	2.11s	0.21s	5.40s	0.21s	3.29s	0.16s	155%
3MB	4.46s	0.26s	7.78s	0.38s	3.31s	0.36s	74%
4MB	7.24s	0.22s	10.62s	0.25s	3.37s	0.17s	46%
5MB	10.95s	0.57s	14.31s	0.61s	3.35s	0.26s	30%

Table 11: Average values and standard deviation (σ) of the time needed to complete a transaction with and without the receipt exchange protocol. The absolute overhead is almost constant.

The obtained results are shown in Table 11, where the introduced overhead is shown, and Table 12, where the detail of the overhead introduced by each operation is shown. This results have been incorporated to the simulations via a parameter called *overhead time*.

6.2. Impact of the overhead

We have used simulations[6] to measure the impact of the overhead caused by the receipt exchange protocol. For this, we have compared a scenario where the fully cooperative nodes does not need to use the protocol (modelled by an *overhead time* of 0 seconds) with a scenario where the selfish behaviours of

[5]Raspberry Pi Broadcom BCM2835 SoC full HD, 700MHz Low Power ARM1176JZ-F, 512MB SDRAM, 4GB SD with Raspbian, equipped with a Wireless Edimax EW-7811Un (802.11b/g/n up to 150Mbps), a GPS receiver NL-302U (baud rate: 4,800 bauds) and a dual output 5,000mAh battery.

[6]In order to avoid redundancy, all details about the methodology, the parameters used, the simulated scenario and the simulator will be found on Section 7.

Operation	Executions per transaction	Time consumed per execution		Time consumed per transaction	
		Average	σ	Average	σ
FSign	2	0.90	0.10s	1.81	0.21s
SVerify	2	0.71	0.07s	1.42	0.14s
Send receipt	2	0.03	0.002s	0.06	0.004s
		Total		3.30	0.25s

Table 12: Detail of the different parts of the receipt exchange protocol and the time consumed by each one. To sign the receipts is the most costly operation.

the nodes urges us to use the receipt exchange protocol to enable the incentive system.

The experiments show that, as can be seen in Table 13, without the overhead introduced by our protocol, the ratio of aborted transactions is 2.4%, corresponding to the transactions of messages that can not be finished before the involved nodes move out of reach one from another. When we take into account the overhead introduced by the receipt exchange protocol, the ratio of aborted transactions is 4.5 times higher, because there are more transactions than can not be successfully finished before the end of an opportunistic contact.

Despite the abort rate, the overall performance of the network is not injured. The design of the receipt exchange protocol, where the transmission of the message is the last step of the protocol, causes that when an exchange finishes abruptly the transaction is considered as not done, and the message is not removed from the source. Even with this increased rate of aborted messages, the impact on the network in terms of delivery ratio is negligible, and the impact in terms of latency is, as will be seen in Section 7, very positive.

Scenario	Overhead	Aborted ratio		Delivery ratio		Latency
		Average	σ	Average	σ	
Cooperative	0s	2.43%	0.65%	94.85%	0.03%	$\approx 21,600$s
Selfish	3.3s	11.02%	0.57%	93.74%	0.05%	N/A[7]

Table 13: Average values and standard deviation (σ) of the results. Although the aborted ratio grows when using the receipt exchange protocol, delivery ratio remains almost unaffected, and latency depends on other parameters.

[7]The latency of the network for the selfish case is heavily dependent on the behaviour of the nodes and the parameters of the incentive system. For this reason, the results obtained can not be condensed in a single latency value. See Section 7 for more details.

7. Simulations and results

In this section, we define how we expect rational nodes to behave when they obtain keys depending on their Level of Cooperation. Afterwards, we present a scenario where our proposal can be applied, and we explain and justify the simulation parameters we have used. Finally, we show and explain the results obtained through simulations.

7.1. Re-defining rational behaviour and the utility function

As we proved in the previous section, for all nodes in the network, to behave in a fully cooperative way is a strictly dominant strategy. This apply if they try to maximize their Level of Cooperation, thus, if the utility function of a node is its LC. However, in practical scenarios, nodes would probably be more interested in the benefit they obtain from their LC than in the LC itself.

In order to model this kind of behaviour, we have defined the concept of selfishness and the selfish utility function, formalized in Algorithm 5. The selfishness of a node measures the percentage of times (selfishness $\in [0, 100]$) that a node behaves in a non-cooperative way.

The selfish utility function operates with two variables: the received amount of keys (rK), and an indicator of the desired amount of keys that the node would like to receive when it contacts with the IM ($dK \in [0, 1]$). When a node receives an amount of keys lower than the amount established by dK, the utility it obtains is 0, and when it receives an amount of keys higher or equal than the desired amount, the utility it obtains is its selfishness. This function models the interest of a node that wants to be as selfish as possible, to save resources, but also wants to keep its LC high enough to obtain a certain amount of keys.

Algorithm 5 *Selfish utility function*

Input: rK: Amount of received IBC keys.
dK: Desired amount of IBC keys.
Output: 0 or 1
1: **if** $rK > min_K + dK(max_K - min_K)$ **then**
2: **return** selfishness
3: **else**
4: **return** 0
5: **end if**

We have also defined an algorithm that models the strategy used by nodes to maximize their utility; it is shown in Algorithm 6. This algorithm updates nodes' selfishness with the goal of obtaining the maximum utility possible. The update is performed every time they get new IBC keys from the IM. Essentially, nodes quickly reduce their selfishness when receive fewer keys than the desired amount, and they slowly increase their selfishness while the amount of obtained keys satisfies them.

Algorithm 6 *Node strategy*

Input: rK: Amount of received IBC keys.
dK: Desired amount of IBC keys.

```
1: if K > min_K + dK(max_K − min_K) then
2:    selfishness = max (100, selfishness +1);
3: else
4:    selfishness = min (0, selfishness −2);
5: end if
```

7.2. Scenario

The scenario we have used in all the simulations is based on the scenario presented by Borrego *et al.* in [35]. This scenario consists of a mobile robot sensor network where messages use the time they are being carried by nodes to execute some tasks. These tasks are called sensing jobs and are injected into the network by the heterogeneous applications that coexist in the network. The multi-purpose approach of the network allows applications to deploy their own nodes. These nodes serve the sensing jobs of some applications without restrictions, forwarding their messages, prioritizing their jobs, etc. These nodes could cooperate with all other applications in order to improve the performance of the whole network, but they have no incentive to do that.

We have chosen this scenario because it has particular characteristics that fit well with our incentive schema: 1) nodes have to return periodically to a base station to recharge their batteries or to deliver some data to the sink node, 2) it is a multi-purpose sensor network that works with different kinds of applications, and 3) nodes are as heterogeneous as the applications and can be deployed by different operators with different goals, so the idea of rational nodes that do not always act in a fully cooperative way makes sense.

According to this scenario, we have modelled the operation of the *sensing jobs* that travel among the network of the original publication using messages. A message that travels from one concrete node to another represents a job that has been executed in a node and wants to travel to another one to continue its work, or a new job recently injected into the network which travels to its first destination. These messages are created with a frequency of one message every 40-80 seconds and their sizes are between 500KB and 3MB.

7.3. Simulation parameters

All simulations have been performed using *The Opportunistic Network Simulator* (The ONE) [36]. We have developed or customized a set of classes that model the behaviour and the movement models of all participants.

We ran all simulations in a field of $1,500$ x $1,500$ meters with 100 nodes implementing a custom random walk movement with random speeds between 0.5 m/s and 1.5 m/s that returns to the base station, located at $(0, 0)$, when their IBC keys expire. The Incentive Manager is placed at the base station, without any movement model. Proofs are sent to the IM using direct delivery

routing. As in [35], the communication range has been adjusted to 15 meters to all nodes.

Nodes may cooperate and forward messages, may not forward messages and save their own resources, or may drop messages. Dropping messages is a way to reduce the load of the network and assure that messages of their application will be slightly better treated (because there would be more space in the buffers, less congestion in nodes, etc.). Nodes decide how to behave with each message depending on their selfishness and the application that owns the message. Nodes model this behaviour using Algorithm 6 to update their selfishness. Then, when a node is requested to receive a message, it uses its selfishness value to decide probabilistically to behave selfishly or cooperatively. When acting selfishly, nodes randomly choose either not accepting the message (50%) or dropping it (50%). When a node decides to be cooperative with one message, it uses Spray-and-Wait [37] to route it, using $L = 3$, where L is the maximum number of message copies present in the network. The initial selfishness of each node is randomly chosen at the beginning of the simulation.

Parameter	Value
Punishment α	49 CP
Reward β	1 CP
Reward ϵ	30 CP
Duration of keys	5 minutes
min_K	3
max_K	30
Interval of selfishness	$[0\%, 100\%]$[8]
Simulation time	$1,000,000$ seconds
Update interval t	500 seconds
Overhead time	3.3 seconds
Desired amount of keys dK	Defined for each simulation
Time constant T	Defined for each simulation

Table 14: Parameters used in the simulations.

Table 14 shows the chosen values of all parameters needed to run the simulations. Note that the amount of desired IBC keys of the nodes, dK, and the time constant T are not fixed, we have defined their values at every simulation in order to study their impact on the system. For the sake of simplicity and to avoid interferences with the simulations that make use of different values of dK, we disabled the waiting time of the nodes when $rK < min_K$.

In order to obtain conclusive results, every simulation has been run five times with the same parameters but different random seeds, and all results have been

[8]The interval of selfishness, [Min selfishness, Max selfishness], has been fixed to [0%, 100%] for each simulation except those necessary to obtain results for the best case ([0%, 0%]) and for the worst case ([100%, 100%]).

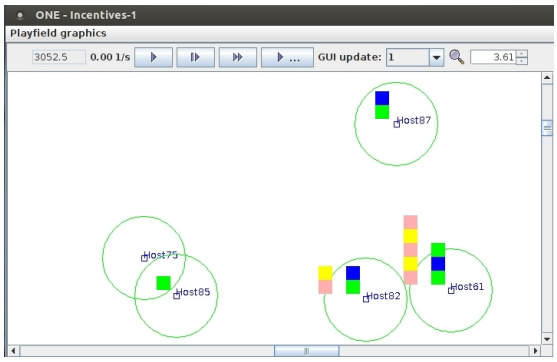

Figure 5: A snapshot of a simulation. On the upper side, a node carries two messages and waits for an opportunity to forward them. On the left side, one node tries to forward the only message it carries, but the other acts selfishly and rejects the transmission. On the right side, two cooperative nodes just forwarded messages between them, generating some proofs of forwarding.

calculated as the average of the five runs. In total, we have executed 150 runs.

7.4. Results. Study of the time constant

This set of tests has been designed to identify how risky and likely selfish bursts are in relation to the time constant T and to study the impact of T regarding the fairness of the incentive schema. We ran simulations with different T values between $5,000$ and $35,000$ seconds and we disabled the update behaviour of all nodes. This models a scenario where nodes ignore the incentives and do not care about their Level of Cooperation. Nodes randomly chose their selfishness at the beginning of the simulation and held it until the end.

To know if this incentive schema allows nodes to save high amounts of CP and then behave selfishly without being punished, we have developed the metric *messages to loose* (*mtl*), that is calculated using the following equation.

$$mtl = \frac{max_{balance} - min_{balance}}{\alpha} \tag{3}$$

What *mtl* defines is the amount of messages that the node with the highest Level of Cooperation of the system can lose before its LC becomes the lowest, assuming all other nodes' LC will stay frozen. We have studied this metric for a set of different T values.

As can be seen in Figure 6, after the transient period, the number of messages that can be lost becomes stable at a higher or smaller value with small peaks, depending on the T used. There is an exception when $T = \infty$, which means that there is no decay function applied, and nodes can infinitely save CP from their past actions, increasing the risk of selfish bursts and the damage they can do.

Note that highly increasing T lead to very small increases of *mtl*. Besides, it is important to note that the average number of messages flowing through

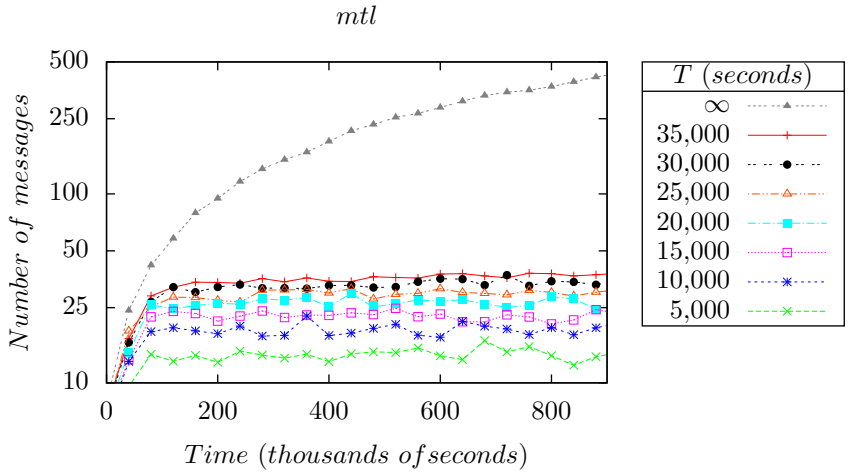

Figure 6: Number of messages that the node with the highest LC can lose before becoming the node with the lowest balance, note the logarithmic scale. Higher values of T imply higher values of mtl because the LC of the nodes decrease slowly and present behaviour are less important compared with past behaviours. When no decay is applied ($T = \infty$), mtl increases linearly.

the network is $1,945.56$. Therefore, mtl, that oscillates between 14.2 and 36.5, suppose a percentage between the 0.73% and the 1.89% of the total messages of the network.

Figure 7 shows the average Level of Cooperation of all nodes inside every selfishness interval. We only show the results of three of the T simulated values, $25,000$, $20,000$ and $15,000$ seconds because they are enough to understand what we address here. All graphics, the three included here, and all others, show the same pattern: after a transient state, all nodes stabilize their LC and hold it without major changes until the end of the simulation. As the time constant T becomes higher, the values where the nodes stabilize their LC are higher too. In all cases, nodes with lesser selfishness have a higher LC almost all the time, except during occasional, little time intervals. In these intervals, it is possible that a category (e.g. 20%-30%) has a higher LC than the immediately less selfish category (e.g. 10%-20%). These intervals become a little more frequent as the time constant T becomes lower.

On the basis of the previous results, we can conclude that a higher T is better for the fairness of the system because it increases the differences between the LC obtained by a selfish node and a cooperative one. But a extreme $T = \infty$, meaning no decay is applied, allow nodes to perform selfish bursts without prejudice, so the time constant has to be chosen high enough to assure fairness, but not too high to avoid this.

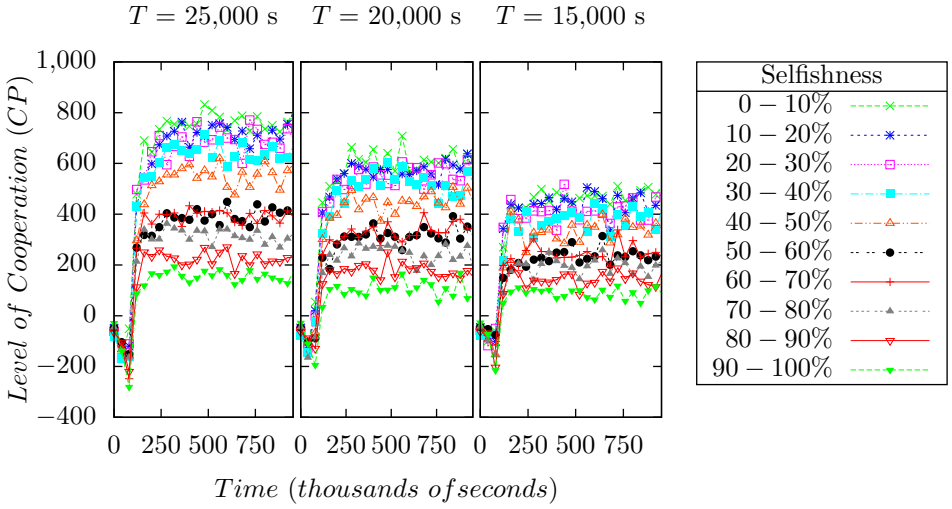

Figure 7: Average Level of Cooperation of all nodes inside every selfishness interval using different values for the time constant T. The three graphics share the Y axle. After a transient period, nodes stabilize their LC. As T becomes smaller, the value where the LC stabilizes and the differences between categories become lesser too, and moments where nodes of one selfish category have a higher LC than nodes of a more cooperative category become slightly more frequent.

7.5. Results. Impact of message expiration

In DTN, messages usually have a time-to-live (*ttl*) and they are discarded when their expiration time is reached. The presented Incentive Scheme punishes the node that discards a message and the previous one. Taking into account that the reason why a node has discarded a message can not be known, it is mandatory to study the fairness of the system when dealing with this. We need to be sure that nodes will not refuse to accept messages because they are afraid of being punished if the messages expire.

We have measured the *mean absolute error* (*mae*), this measures how close is the ranking of nodes elaborated by the IM to an ideal ranking of cooperative nodes elaborated by an omniscient entity that knows all about nodes' actual and past behaviours. We have calculated *mae* using Equation 4.

$$mae = \frac{1}{|N|} \sum_{i=0}^{|N|} |lcr(i) - rbr(i)| \tag{4}$$

Where

- $lcr(i)$ is the *level of cooperation rank*, the position of node i on a list of all nodes of the network, ordered by level of cooperation.

- $rbr(i)$ is the *real behaviour rank*, the position of node i on a list of all nodes of the network, ordered by their real behaviour, elaborated by an omniscient entity.
- $|N|$ is the amount of nodes of the network.

On this set of simulations we measured the fairness of the presented Incentive Scheme to handle message expiration. We have fixed T to $20,000$ seconds. We have ran the experiments without message expiration to obtain the lower bound of the *mae* and the average latency. Then, the *ttl* of each message was set to $10,000$, $20,000$ or $30,000$ seconds (one third, two thirds, and one time the average latency without message expiration).

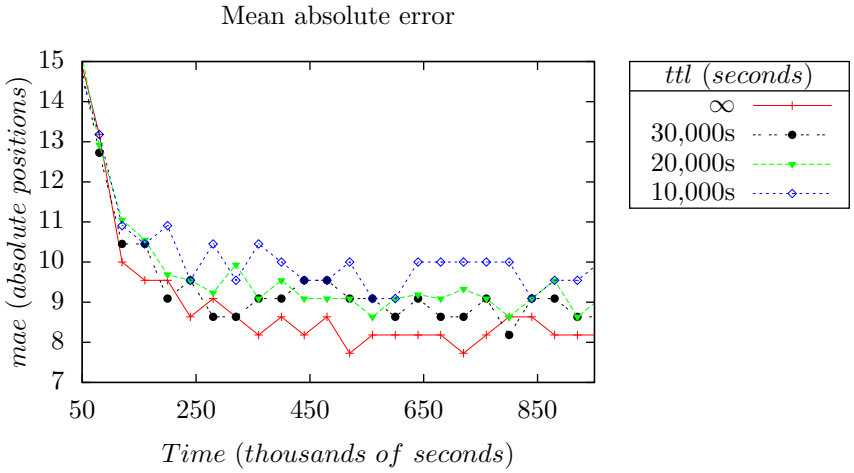

Figure 8: Mean absolute error depending on the *ttl*. The obtained results are very similar. High decreases of the *ttl* lead to small increases of the *mae*.

Figure 8 shows the average *mae* obtained when different *ttl* are used. As it can be seen, each high decrease of the *ttl* leads to a small increase of the mean absolute error. The reason for this is that the punishment applied to nodes when the messages expire is distributed among all nodes. Besides, cooperative nodes are more exposed to message expiration, but they also have a higher LC, meaning they are more resistant to the punishment. As a result, when a message expires, the LC of its custody node is reduced, but its position on the overall ranking remains almost unchanged. This means that the Incentive Scheme is fair with the nodes regarding message expiration.

7.6. Results. Performance of the network

To perform this set of simulations, we have chosen $T = 20,000$ seconds. Besides which, we have enabled the update behaviour of all nodes and we have run simulations with different values of dK.

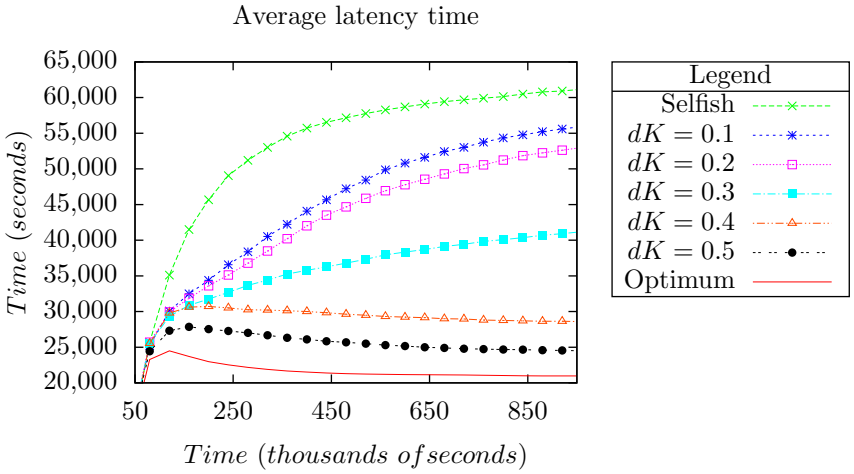

Figure 9: Latency average depending on the dK of nodes. If nodes conform with obtaining 20% or less of the keys, the latency decreases a little. The latency is reduced a 30% if nodes wants to obtain at least the 30% of the keys. When nodes want to obtain the 40% of the keys or more, the latency is reduced more than 50%.

Figure 9 shows the average latency (defined as the time required by messages to travel from their source to their destination) of a network where nodes share the dK parameter. The selfish case has been measured with all nodes behaving with a 100% of selfishness, meaning the only way a message can arrive at its destination is using direct delivery, so it can be considered the worst case, the upper bound. The optimum case has been measured with all nodes behaving with a 0% of selfishness, so it can be considered the best case, the ideal network where all nodes are fully cooperative: the lower bound.

The obtained results are extremely successful because they show that the incentive system improves the performance of the network significantly. Even when dK is as low as 0.3, the latency of the network is reduced by about 30%. When $dK > 0.3$ which definitely is not a strong requirement, the latency of the network is reduced by more than 50% and approaches a lot the lower bound.

Figure 10 shows the average latencies obtained with $dK > 0.5$. As can be seen, the differences are not significant. This is a good point because it means that the incentive system does not need nodes to care a lot about the amount of IBC keys they receive to improve the overall performance of the network. Even in networks where nodes are easily satisfied when they obtain half of the keys they could obtain, or even little less than half, the incentive scheme provides an important increase in network performance.

We can conclude that the presented incentive scheme, that punishes and rewards nodes on the basis of their behaviour by giving them a higher or lesser amount of IBC keys, forces nodes to be more cooperative, and improves the performance of the network by reducing the latency of the messages by more

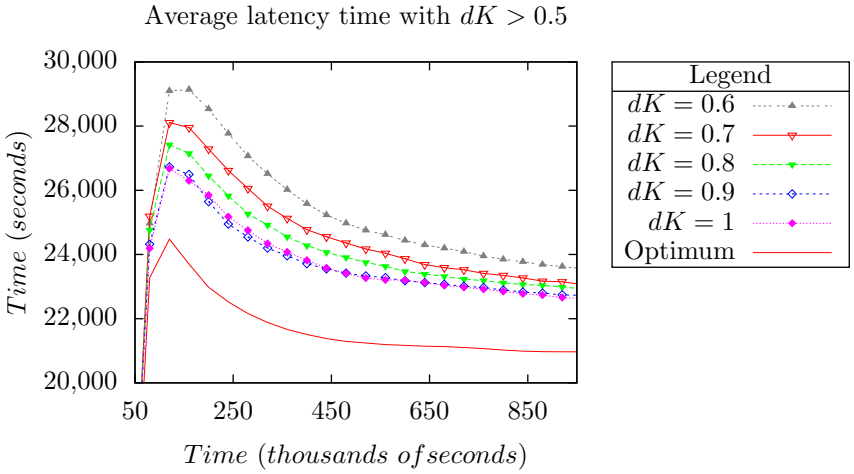

Figure 10: Latency average for values of dK greater than 0.5. Differences between results obtained with values of dK greater than 0.5 are very small. In all cases, the obtained performance of the network is similar to the performance obtained in the Optimum case.

than 50%.

8. Conclusions and future work

We have presented an asynchronous incentive scheme for DTNs. This scheme is based on a receipt exchange protocol designed to overcome the inherent limitations of DNTs and uses the policy "guilty until proven innocent" to punish suspicious nodes and reward cooperative nodes.

Moreover, we have developed a new way of tracking the actions of nodes that allow us to treat nodes in a new, fair way: they will be rewarded for the actions they perform, without depending on other elements like the actions performed by others, the delivery ratio of the routing algorithm used, etc...

The game theory analysis of the incentive scheme has proven that, for each node, accepting and relaying messages, delivering receipts to the Incentive Manager and avoiding cheating is a dominant strategy and the best response to any other node behaviour. This way, we have proved that all nodes behaving this way form a Nash equilibrium.

The results of the simulations show that the usage of the incentive scheme improves the performance of a network, even if nodes try to behave in a selfish way, ignoring their balance. In the concrete scenario of a wireless robot sensor grid network with heterogeneous nodes and applications, latency is reduced by more than 50%. And this improvement is obtained with the only requirement that nodes has to want to obtain at least 40% of the keys they could obtain.

As a future line of research, we plan to modify the system by using different types of IBC keys that only allow nodes to perform a subset of all the possible actions, this way we plan to improve the enforcing mechanism by increasing the punishment to uncooperative nodes, but at the same time giving them more options to redeem and recover. We also plan to be specially focused on adapting this scheme to incentive nodes to behave in a certain way not only in terms of routing and cooperation but also in terms of movement and location, improving the coverage of the network and the quality of the opportunistic contacts. Finally, we think that re-designing the incentive scheme, respecting the main principles, to charge nodes for every message sent will allow the scheme to be useful to a wider range of networks.

Acknowledgements

This work has been partially funded by the Ministry of Science and Innovation of Spain, under the reference project TIN2010-15764 and by the Catalan Government under the reference project 2014SGR691.

References

[1] S. Farrell, V. Cahill, Delay- and Disruption-Tolerant Networking, Artech House, Inc., Norwood, MA, USA, 2006.

[2] K. Scott, S. Burleigh, Bundle Protocol Specification, RFC 5050 (Experimental) (2007).

[3] M. Joye, G. Neven, Identity-Based Cryptography, IOS Press., 1013 BG, Amsterdam, The Netherlands, 2008.

[4] S. Giordano, et al., Mobile ad hoc networks, Handbook of wireless networks and mobile computing (2002) 325–346.

[5] S.-B. Lee, G. Pan, J.-S. Park, M. Gerla, S. Lu, Secure incentives for commercial ad dissemination in vehicular networks, in: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '07, ACM, New York, NY, USA, 2007, pp. 150–159. doi:10.1145/1288107.1288128.

[6] A. Garyfalos, K. Almeroth, Coupons: A multilevel incentive scheme for information dissemination in mobile networks, Mobile Computing, IEEE Transactions on 7 (6) (2008) 792 –804. doi:10.1109/TMC.2008.37.

[7] H. Zhu, X. Lin, R. Lu, Y. Fan, X. Shen, Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks, Vehicular Technology, IEEE Transactions on 58 (8) (2009) 4628–4639. doi:10.1109/TVT.2009.2020105.

[8] R. Lu, X. Lin, H. Zhu, X. Shen, B. Preiss, Pi: A practical incentive protocol for delay tolerant networks, Wireless Communications, IEEE Transactions on 9 (4) (2010) 1483 –1493. doi:10.1109/TWC.2010.04.090557.

[9] L. Buttyán, J.-P. Hubaux, Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks, Tech. rep., Swiss Federal Institute of Technology (2001). doi:DSC/2001/001.

[10] L. Buttyan, L. Dora, M. Felegyhazi, I. Vajda, Barter-based cooperation in delay-tolerant personal wireless networks, 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) 0 (2007) 1–6.

[11] L. Buttyán, L. Dóra, M. Félegyházi, I. Vajda, Barter trade improves message delivery in opportunistic networks, Ad Hoc Networks 8 (1) (2010) 1–14. doi:http://dx.doi.org/10.1016/j.adhoc.2009.02.005.

[12] N. Nisan, Algorithms for selfish agents, in: STAC'99. Symposium on Theoretical Aspects of Computer Science, STACS'99, Springer-Verlag, Springer-Verlag, Trier, Germany, 1999, p. 1–15.

[13] C. Papadimitriou, Algorithms, games, and the internet, in: Proceedings of the thirty-third annual ACM symposium on Theory of computing, STOC '01, ACM, New York, NY, USA, 2001, pp. 749–753. doi:10.1145/380752.380883.

[14] T. Roughgarden, E. Tardos, How bad is selfish routing?, J. ACM 49 (2) (2002) 236–259. doi:10.1145/506147.506153.

[15] S. Zhong, J. Chen, Y. Yang, Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks, in: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, Vol. 3, 2003, pp. 1987–1997. doi:10.1109/INFCOM.2003.1209220.

[16] A. Balasubramanian, B. Levine, A. Venkataramani, Dtn routing as a resource allocation problem, SIGCOMM Comput. Commun. Rev. 37 (4) (2007) 373–384. doi:10.1145/1282427.1282422.

[17] U. Shevade, H. H. Song, L. Qiu, Y. Zhang, Incentive-aware routing in dtns, in: Network Protocols, 2008. ICNP 2008. IEEE International Conference on, 2008, pp. 238–247. doi:10.1109/ICNP.2008.4697042.

[18] D. Levin, K. Lacurts, N. Spring, B. Bhattacharjee, Bittorrent is an auction: analyzing and improving bittorrent's incentives, in: ACM SIGCOMM Conference, 2008, pp. 243–254. doi:10.1145/1402958.1402987.

[19] A. Seth, D. Kroeker, M. Zaharia, S. Guo, S. Keshav, Low-cost communication for rural internet kiosks using mechanical backhaul, in: Proceedings of the 12th annual international conference on Mobile computing and networking, MobiCom '06, ACM, New York, NY, USA, 2006, pp. 334–345. doi:10.1145/1161089.1161127.

[20] F. Milan, J. J. Jaramillo, R. Srikant, Achieving cooperation in multihop wireless networks of selfish nodes, in: Proceeding from the 2006 workshop on Game theory for communications and networks, GameNets '06, ACM, New York, NY, USA, 2006. doi:10.1145/1190195.1190197.

[21] J. J. Jaramillo, R. Srikant, Darwin: distributed and adaptive reputation mechanism for wireless ad-hoc networks, in: Proceedings of the 13th annual ACM international conference on Mobile computing and networking, MobiCom '07, ACM, New York, NY, USA, 2007, pp. 87–98. doi:10.1145/1287853.1287865.

[22] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, Self-recharging virtual currency, in: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, P2PECON '05, ACM, New York, NY, USA, 2005, pp. 93–98. doi:10.1145/1080192.1080194.

[23] H. Zhu, S. Du, Z. Gao, M. Dong, Z. Cao, A probabilistic misbehavior detection scheme toward efficient trust establishment in delay-tolerant networks, IEEE Transactions on Parallel and Distributed Systems 25 (1) (2014) 22–32. doi:http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.36.

[24] B. N. Chun, P. Buonadonna, A. Auyoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, A. Vahdat, Mirage: A microeconomic resource allocation system for sensornet testbeds, in: In Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors, 2005.

[25] D. G. Sullivan, M. I. Seltzer, Isolation with flexibility: a resource management framework for central servers, in: Proceedings of the annual conference on USENIX Annual Technical Conference, ATEC '00, USENIX Association, Berkeley, CA, USA, 2000, pp. 27–27.

[26] Q. Li, G. Cao, Mitigating routing misbehavior in disruption tolerant networks, Information Forensics and Security, IEEE Transactions on 7 (2) (2012) 664–675. doi:10.1109/TIFS.2011.2173195.

[27] S. Marti, T. J. Giuli, K. Lai, M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, in: Proceedings of the 6th annual international conference on Mobile computing and networking, MobiCom '00, ACM, New York, NY, USA, 2000, pp. 255–265. doi:10.1145/345910.345955.

[28] S. Kremer, O. Markowitch, J. Zhou, An Intensive Survey of Fair Non-Repudiation Protocols, Computer Communications 25 (17) (2002) 1606–1621.

[29] O. Markowitch, Y. Roggeman, Probabilistic Non-Repudiation without Trusted Third Party, in: 2nd Conference on Security in Communication Networks, Amalfi, Italy, 1999.

[30] J. Mitsianis, A new approach to enforcing non-repudiation of receipt, manuscript (2001).

[31] J. Liu, R. Sun, W. Ma, Y. Li, X. Wang, Fair exchange signature schemes, in: Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on, 2008, pp. 422 –427.

[32] N. Asokan, K. Kostianinen, P. Ginzboorg, J. Ott, C. Luo, Applicability of Identity-Based Cryptography for Disruption-Tolerant Networking, in: MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking, ACM, New York, NY, USA, 2007, pp. 52–56. doi:http://doi.acm.org/10.1145/1247694.1247705.

[33] Y. Cao, Z. Sun, Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges, Communications Surveys Tutorials, IEEE 15 (2) (2013) 654–677. doi:10.1109/SURV.2012.042512.00053.

[34] J. C. Harsanyi, Rational Behaviour and Bargaining Equilibrium in Games and Social Situations, Press Syndicate of the University of Cambridge, New York, USA, 1977.

[35] C. Borrego, S. Robles, A store-carry-process-and-forward paradigm for intelligent sensor grids, Information Sciences 222 (2013) 113 – 125.

[36] A. Keränen, J. Ott, T. Kärkkäinen, The ONE Simulator for DTN Protocol Evaluation, in: SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, ICST, New York, NY, USA, 2009.

[37] T. Spyropoulos, K. Psounis, C. Raghavendra, Spray and Wait: an Efficient Routing Scheme for Intermittently Connected Mobile Networks, in: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, ACM, 2005, p. 259.