Give2Get: Forwarding in Social Mobile Wireless Networks of Selfish Individuals

Alessandro Mei, *Member*, *IEEE*, and Julinda Stefa

Abstract—In this paper, we present two forwarding protocols for mobile wireless networks of selfish individuals. We assume that all the nodes are selfish and show formally that both protocols are strategy proof, that is, no individual has an interest to deviate. Extensive simulations with real traces show that our protocols introduce an extremely small overhead in terms of delay, while the techniques we introduce to force faithful behavior have the positive and quite surprising side effect to improve performance by reducing the number of replicas and the storage requirements. We test our protocols also in the presence of a natural variation of the notion of selfishness—nodes that are selfish with outsiders and faithful with people from the same community. Even in this case, our protocols are shown to be very efficient in detecting possible misbehavior.

Index Terms—Delay tolerant networks, pocket switched networks, social mobility, selfishness, forwarding protocols.

1 Introduction

In the last few years, the diffusion of mobile personal devices exploded. Smartphones are used by people—not only technology geeks—to communicate, to use applications once run only by desktops, and to organize their life. Typically, these devices can communicate with each other over short distances by using wireless technologies such as bluetooth. In this way, a new kind of network emerges where nodes are carried by people and links appear and disappear as people move and get in contact. These networks, also known as Pocket Switched Networks (PSN [1], [2]), can be key technology to provide innovative services to the users without the need of any fixed infrastructure. Pocket Switched Networks are usually disconnected, are characterized by social-based mobility and heterogeneous contact rate. Examples of such networks include people at work places, students on university campuses, and citizens in metropolitan areas. Possible application scenarios of PSNs include bringing connectivity to rural areas and enabling services such as e-governance, citizen journalism etc.

The problem of designing efficient forwarding protocols for PSNs has attracted the attention of many researchers. In forwarding protocols, messages are routed from source to destination thanks to intermediate relays. One fundamental and natural question, especially in this setting, is why nodes should accept to use their own energy and bandwidth just to carry other people's messages. Indeed, the protocols in the literature break down immediately if you do not assume that all the nodes cooperate in an altruistic manner. We show this phenomenon, which is intuitive indeed, by a few experiments on Epidemic Forwarding [3] and Delegation Forwarding [4], two important protocols in the literature.

• *The authors are with the Department of Computer Science, Sapienza University of Rome, Via Salaria 113, 00198 Rome, Italy. E-mail: {mei, stefa}@di.uniroma1.it.*

Manuscript received 7 Mar. 2011; revised 19 Nov. 2011; accepted 19 Mar. 2012; published online 23 Mar. 2012.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2011-03-0038. Digital Object Identifier no. 10.1109/TDSC.2012.37.

In this paper, we introduce Give2Get (G2G) Epidemic Forwarding and Give2Get Delegation Forwarding, which are, to the best of our knowledge, the first protocols for packet forwarding in a social mobile setting that leverage on the social aspects of the network to tolerate selfish behavior: friend nodes meet with high frequency. This helps us showing formally that no rational node has any incentive to deviate. In other words, our two protocols are strategy proof, i.e., the strategies of following the protocols are Nash Equilibria. However, for simplicity we will use the words protocol and strategy (to follow that protocol) interchangeably. In the paper, we describe our methodology and the main steps, the mechanisms, and the ideas that we have used to build the complete proof.

Lastly, we perform a large set of experiments to check the performance of G2G Epidemic Forwarding and G2G Delegation Forwarding. Quite surprisingly, we discover that some of the mechanisms that we introduce to make these protocols Nash equilibria are also useful to control the number of replicas in the network and push the messages quickly and cheaply far from the community where they have been generated. As a result, G2G Epidemic Forwarding and G2G Delegation Forwarding, besides providing robustness in a network where every node is selfish, have nearly the same delay and success rate of their original alter egos, and have a considerably lower cost in terms of number of replicas (around 20 percent less). Moreover, we also perform a detailed study of the memory load required by all these protocols. To the best of our knowledge, this is the first time such a detailed study is performed. We measure the cost generated by each protocol by computing the average storage requirements to forward one message. Our experimental results show that our G2G protocols, aside providing tolerance to selfish behavior, require considerably less storage than their vanilla alter-egos in almost all cases, even including the overhead due to signatures and other information used by the G2G mechanisms.

The paper is organized as follows: Section 2 reports on the literature in the area; Section 3 defines our system model; Sections 4 and 5 present G2G Epidemic Forwarding

and G2G Delegation Forwanding along with proofs that the two protocols are Nash Equilibria; in Section 6 we first illustrate a large set of experiments that show the dramatic effect of selfish behavior on vanilla Epidemic and Delegation Forwarding; we show that our protocols detect efficiently and quickly possible deviations; we then describe the excellent performance of our protocols—similar success rate and delay of earlier protocols (that do not tolerate selfish behavior) and considerably less requirements in terms of both number of replicas and storage. Lastly, Section 7 concludes the paper.

2 Related Work

A lot of work has been done in building efficient forwarding protocols for Pocket Switched Networks. Many of the protocols in the literature use in sophisticated ways the properties of human mobility [4], [5], [6], [7], [8], [9]. All of them rely on the altruistic cooperation among the nodes.

The problem of building mechanism and protocols that can tolerate selfish behavior is an important and modern issue in the design of networking protocols and distributed systems. See, as an important example, the work in [10], [11]. Earlier work has been done to mitigate the impact of selfish behavior in mobile ad hoc networks as well. The solutions can be classified into two main approaches: *reputation-based schemes* [12], [13], [14], [15] and *credit based schemes* [16], [17], [18], [19]. In the former schemes, nodes collectively detect misbehaving members and propagate declarations of misbehavior throughout the network. Eventually, other nodes will avoid routes through selfish members. In credit-based approaches, nodes pay and get paid for providing service to others. Digital cash system is implemented in order to encourage correct behavior among nodes. In [20], a combination of the two schemes is presented. All these solutions assume the use of public key cryptography for authentication of messages. Regardless of the performance of these schemes on ad hoc networks, none of them is designed for social mobile networks. Indeed, no previous work is neither designed with a social mobile scenario in mind, nor exploits the social nature of the network or the properties of the movement that such social nature generates.

Recently, Buttyán et al. [21] introduced a barter-based cooperation system to increase message delivery rate in opportunistic networks. The authors assume that altruistic static nodes scattered on the network area generate messages downloadable by interested network members in physical proximity. When two nodes meet, they exchange the list of the messages in their buffers and each node decides to download from the other node only the messages of its interest. Then, the nodes start downloading messages till they move out each other's communication range. A game-theoretical model helps the authors prove that the approach foster cooperation. They support their findings with extensive simulations done with the restricted random waypoint (RRW) model and the Simulation of Urban MObility SUMO [22]. Though it introduces a novel technique of stimulation of cooperation, their work is oriented to a gossip-like service, where messages are created from special nodes and many other nodes are interested in downloading them, which is a natural incentive for the distribution.

COFFEE [23] is an interesting mechanism to enforce cooperation among nodes in wireless networks. However, the solution relies on fixed and known-in-advance routes among nodes. Indeed, it uses the DSR routing protocol, which is shown to be inapplicable in a dynamic, delay tolerant setting such as pocket switched networks. In [24], [25], the authors study the impact of different degrees of cooperation among the nodes on the performance of Epidemic Forwarding [3], Binary Spray&Wait [26], and the Two-Hop relaying algorithm [27]. A recent work [28] presents a routing mechanism built upon the willingness (declared by each individual) to forward other individuals' messages. In [29], the authors study for the first time the impact of different distributions of altruism on the throughput and the delay of mobile social communication systems. They show that, when forwarding algorithms that use multiple paths are considered, social mobile networks are robust to different distributions of altruism of nodes. To the best of our knowledge their work is the first study aimed to explore altruistic/selfish behavior in these types of networks and encourages for further work in this direction.

3 The System Model

3.1 System and Node Properties

In our system model, every node is selfish. This is a realistic scenario, if people can get the same level of service without using part of their battery or part of their wireless uptime or memory without any consequence, they will. And as soon as the first user finds a way to get more (or the same) while paying less, and publishes the patch of the system software, everybody will download the patch and use it. So, it is reasonable to assume that, if some of the nodes deviate selfishly, after a while everybody will.

We assume that there are no byzantine nodes in the network. We also assume that selfish nodes do not collude. All the nodes in the system are interested in receiving and sending messages, in other words, all the nodes are interested in staying in the system. Nodes are loosely time synchronized. Loose time synchronization (i.e., to effectively limit the clock-drift to an external reference [30]) is very easy to get, if a precision in the order of the second is enough, like in our protocol. We assume that every control message of our protocols is labeled with a time stamp, though it does not appear in the protocols to keep the presentation clean. The clock is used to check the time-outs, and the time stamp is used when reporting misbehavior to the authority.

Lastly, nodes are capable of making use of public key cryptography—this capability will be used to sign messages and to make sender to destination encryption. Therefore, we assume that every node has a public key and the corresponding private key.

3.2 The System Authority and Key Revocation

In our system nodes that join and leave are handled by a central authority. The authority handles new nodes joining the network in a standard way: It identifies the new node and it signs the new node's certificate (or the master public key is handed out to the node in case of an identity-based public key system). More authorities can coexist, as long as they exchange information on nodes that enter and exit the system

in real time. To communicate with the nodes, we assume that the authority can use the cellular infrastructure or wireless technology like, e.g., GSM. This technology is very expensive compared with Bluetooth communication used by our forwarding protocols. However, it is used very sparingly: Just when a node is revoked from the system using methods like in [31], [32]. We can reasonably expect that this event is rare. The cellular network can also have some delay, usually due to nodes that are temporarily out of coverage—this is not an issue, even if a revocation due to misbehavior is late, it does not lose its power as a deterrent. Moreover, nodes can pretend to have run out of battery or to be out of coverage just to prevent communication with the authority. These problems can be dealt with quite easily (for example, by forcing the nodes to keep all the cryptographic proofs of their behavior until they are up and with cellular network coverage). However, in the following, we will assume that the cellular network covers the whole network and that nodes are always up and running. This assumption is not fundamental (we can work out a solution even without it), it is not far from reality, and it considerably simplifies presentation. Node failures are dealt with as if a node switches of the device for a certain amount of time. It is considered to be a legitimate behavior in the system, even though, as we will prove, no node will chose to do so deliberatively for its quality of service will drop during such time.

It is known that public key cryptography is more expensive than symmetric cryptography. However, modern cryptography techniques, like those based on elliptic curves, provide short signatures (a secure signature based on elliptic curves is just 320 bits long), and cheaper and cheaper computation [33], which is shown to be adequate even for sensors. The same is true for identity-based cryptosystems [34]. In addition to this, the delay tolerant nature of the PSNs (there are less delay constrains) gives nodes the time to generate and verify signatures. Moreover, in our study we are addressing a network of smartphones or PDAs, which are not-so-small devices. Modern smartphones can run sophisticated applications, like decoders of streaming videos, 3D games, web browsers that can open SSL sessions, and others. For these devices, a signature per message can be considered a relatively low overhead. In the rest of this paper, we will use $H()$ to denote a hash function, and $\langle m \rangle_A$ to denote a message m signed by node A.

4 Give2Get Epidemic Forwarding

In Epidemic Forwarding [3], every contact is used as an opportunity to forward messages. When node A meets node B, and A has a message that B does not have, the message is relayed to node B. Epidemic forwarding with unlimited buffer is often used as a benchmark, it is easy to see that it is impossible to get smaller delay, or higher success rate. However, the overhead in terms of number of copies of the same message is very high. Put simply, many of the forwarding protocols in the literature on Pocket Switched Networks have the goal of reducing drastically the overhead without affecting much the delay and the success rate of Epidemic Forwarding.

However, Epidemic Forwarding does not tolerate a scenario in which users can make selfish choices. Indeed, selfish nodes simply drop every message they receive (except those destined to themselves!). We will call *message droppers* the nodes that implement this simple form of deviation. In this section, we show how to build a version of Epidemic Forwarding, called Give2Get Epidemic Forwarding, that works in a system where every node is selfish. We will see that G2G Epidemic Forwarding is a Nash equilibrium, that is, no selfish node has a better choice than following the protocol truthfully. In this way, we protect the network against message droppers and against any other rational deviation. Most of the ideas and techniques that we develop in this section will be used in the more sophisticated protocols we introduce later in this paper.

$$A \xrightarrow{\langle \text{RELAY\_RQST}, H(m), T_{AB} \rangle_A} B \quad (1)$$

$$A \xleftarrow{\langle \text{RELAY\_OK}, H(m) \rangle_B} B \quad (2)$$

$$A \xrightarrow{\langle \text{RELAY}, H(m), E_k(m) \rangle_A} B \quad (3)$$

$$A \xleftarrow{\text{POR}(m,A,B) \equiv \langle \text{POR}, H(m), A, B \rangle_B} B \quad (4)$$

$$A \xrightarrow{\langle \text{KEY}, H(m), k \rangle_A} B \quad (5)$$

Fig. 1. Protocol of the relay phase (in case node B does not have the message).

G2G Epidemic Forwarding consists of three phases: Message generation, relay, and test. The idea behind each phase is as follows: 1) during message generation the message is modified so that a relay candidate has no interest in not accepting it; 2) the relay phase "forces" nodes to collect the so-called *proof of relay* to show to previous relays (or source), during the test phase, 3) that they have correctly behaved with the message—this is to make it impossible to relays to drop messages. The details of each phase will be given in the remaining of the section.

Message generation executes when one node creates a message to send. Suppose that node S has a message to send to node D. The message is built according to the following form: $m = \langle D,\ E_{PK_D}(S, msg\_id,\ \text{body}) \rangle_S$. Key PK_D is the public key of the destination D. Note that it is a precise design choice to hide the sender of the message to every possible relay except the destination. We will see later why this is important.

4.1 G2G Epidemic Forwarding: The Relay Phase

Once the message is generated, the sender S tries to relay it to the first *two* nodes it meets. When node S meets node B, node S starts a session with the possible relay by negotiating a cryptographic session key with node B. This is easily and locally done by using the certificates of the two nodes, signed by the trusted authority. In this way, both identities are authenticated. From this point on, every communication during the session is encrypted with a symmetric algorithm like AES and the session key (to keep the notation clean, this encryption is not shown in the protocols). Node S starts the relay phase by asking node B if it has already handled a message with hash $H(m)$ (see Fig. 1, where the role of S is described as done by node A step 1). Token T_{AB} is a cryptographic proof that node A has completed with the test phase the last interaction with

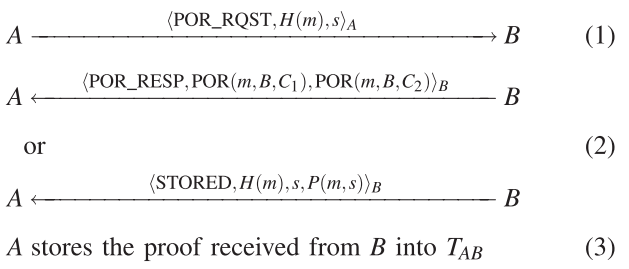


Fig. 2. Protocol of the test phase.

node B. In the formal proof, the token will be key to show that also intermediate relays—and not just the sender—has an interest to perform the test phase. More details on how the token T_{AB} is being computed will be given in the description of the test phase (Session 4.2). However, in case this is the first interaction, the token can be empty. In case node B has never seen this message, the relay phase goes on (step 2), otherwise node B informs S that it should not be chosen as a relay. Note that node B would not lie, since it still does not know the content of the message, its destination, and, in particular, if node B itself is the destination. In other words, if B deviates and executes a modified version of the protocol in which it declines offers of being a relay without knowing the destination of the message, it won't receive any message, against its own interest. Node S generates a random key k, and sends message m to B, encrypted with key k (step 3). Then, node B sends a *proof of relay* to node S which in turn, lastly, sends key k to B, who now knows whether it is the destination of the message or just a relay. Note that the relay phase is only started for messages that have not expired yet. Indeed, after message time-out (that we will denote with Δ in the rest of the paper), nodes are allowed to discard everything about expired messages.

4.2 G2G Epidemic Forwarding: The Test Phase

Once it realizes that it is a relay for message m, node B follows the same protocol as done by node A, the previous relay. That is, find two other nodes and relay the message to these two nodes by executing the relay phase as shown in Fig. 1. By doing so, it can collect two proofs of relay (PORs) that it will be asked to show, when meeting node A again, during the test phase. If node B is not able either to show the two proofs or to prove to have the message still in its memory (as detailed at Step 2 in Fig. 2), then node A can send by the cellular network a *proof of misbehavior* (PoM) to the authority. The proof of misbehavior consists of the proof of relay $\langle \text{POR}, H(m), A, B\rangle_B$, signed by node B. The authority, in turn, will revoke node B, if node B is not in the position to prove that A is wrong by showing the two proofs of relay or by proving to have the message still in its memory to the authority (the protocol between the authority and node B is simple and similar to the test phase in Fig. 2). It is important to realize, however, that under our assumptions misbehavior never happens. Indeed, all the nodes are rational—they will not deviate from the protocol since it is against their own interest as we prove more formally later in this paper. In particular, nodes have no interest in sending fake proofs of misbehavior against other nodes to the authority since it is expensive and it is not going to have any effect.

Only when two proofs are collected the message can be discarded from B's memory. After a time-out Δ, B can stop looking for relays and can discard every information regarding the message. In turn, node A can discard the token T_{AB}, in case a test phase between the two nodes is executed. Time-out Δ plays the role of the message time to leave (TTL) in Epidemic Forwarding. Therefore, it should be chosen in such a way that the success rate is high enough. Our experiments show that the delay of G2G Epidemic Forwarding is very close to the delay of Epidemic Forwarding, and so Δ can be chosen as in its original alter ago without affecting the success rate.

The test phase is started by node A (see Fig. 2), when meeting node B before time-out Δ. During the test phase, node A challenges node B: Either it has two proofs of relay, or it still stores the message. In case node B has two proofs of relay, it can reply with the two proofs. The challenge is a simple cryptographic protocol in which node A generates a random seed s and asks node B to send back the value of a hard to compute puzzle of message m and seed s. The puzzle (denoted with $P(m, s)$ in the figure) is a so-called *cost-function*: efficiently verifiable, but expensive to compute. Example of cost-functions are the Interactive Hashcash function [35] used to mitigate sybil and DOS attacks to servers, or the Rivest et al. time-lock puzzle [36] famous for mitigating distributed DOS attacks. By choosing a cost-function that is hard enough to compute, node B is encouraged to relay the message and get the two proofs of relay in all cases in which the probability of meeting node A again is not negligible (below a fixed and small probability). Note that B does not know the seed beforehand, it must be storing the message unless it has found two relays. So, while it is a legitimate part of the strategy to keep the message and not to relay it, it is rational to relay it unless the meetings with the previous relay are very infrequent. In this way, we can make the event that B chooses to store the message rare in such a way that success probability is virtually unaffected. Note that in many cases A cannot check whether B has correctly computed the puzzle: Node A may have already forwarded m to 2 relays (one is B itself) and thus legitimately dropped the message afterwards. However, if A is the source, A is interested in storing the message and in checking that B has behaved correctly by verifying the puzzle result. Since B does not know whether A is the source of m or not, node B has to behave otherwise it can be removed from the system.

Lastly, if A (relay or source) does not start the test phase as required by the protocol, node B is forced to keep proofs of relay in his buffer till time-out Δ expires, although it could get rid of them and free the memory if A started the test phase. Thus, the token T_{AB} will not be sent to A, and without it, node B will punish A and not relay any message whose destination is node A until time-out Δ expires. If A does not start the test phase this is very important to get node A perform the test phase. In addition, it is *not* possible for B to fool A by forging any of the two proofs, since they are signed by the two relays. In any case, node A will store the proof received by node B as token T_{AB} until expiration of time-out Δ.

4.3 G2G Epidemic Forwarding Is a Nash Equilibrium

We formally show that G2G Epidemic Forwarding is a Nash equilibrium by defining a set of *players*, a set of possible *strategies* $\mathcal{S}$, and a *payoff function* $f : \mathcal{S}^n \mapsto \mathbb{R}^n$. In our case, the set of players consists of the n nodes of the network, and the set $\mathcal{S}$ of possible strategies contains all the possible protocols that the nodes might implement. When each player i chooses strategy $s_i \in \mathcal{S}$, the payoff function maps the *strategy profile* $s = (s_1, \ldots, s_n)$ to $f(s) = (f_1(s), \ldots, f_n(s))$, where $f_i(s)$ is the payoff of player i. Note that the payoff of each player depends on the strategies chosen by all the players. A strategy profile s is a *Nash equilibrium* if no player can do better by changing *unilaterally* his strategy. Formally, a strategy profile s is a Nash equilibrium if for all players i and for all strategy profiles s' such that s' and s differ only in position i, $f_i(s') \leq f_i(s)$. When we say "protocol s^* is a Nash equilibrium," what we formally mean is that strategy profile $(s^*, \ldots, s^*)$ is a Nash equilibrium.

Since strategies (that is, protocols) are not limited in length, set $\mathcal{S}$ contains an infinite number of elements. To handle the proof, we organize it in the following way: Assume that our goal is to prove that protocol π is a Nash equilibrium and that π consists of r steps. We partition the set $\mathcal{S}$ of strategies into disjoint subsets $\mathcal{P}_1, \ldots, \mathcal{P}_{r+1}$, where $\mathcal{P}_j$ contains all the protocols that are equal to π in the first $j-1$ steps and then start to deviate. Clearly, $\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_{r+1}$ is equal to $\mathcal{S}$ and $\mathcal{P}_{r+1} = \{\pi\}$. Fixed player i and step j, if for all strategy profiles $s' \in \mathcal{P}_j$ such that s' and $(\pi, \ldots, \pi)$ differ only in position i it holds $f_i(s') \leq f_i(\pi)$, then we say that player i *executes step j of protocol π truthfully*. Indeed, in this case player i has no incentive to deviate at step j. Therefore, we will prove that protocol π is a Nash equilibrium by showing that for all players i and for all $j = 1, \ldots, r$, player i executes step j of protocol π truthfully. Note that, while our protocol do not use randomization, the user can switch to randomized protocol steps like "do C_1 with probability p, otherwise do C_2." Therefore, also randomized deviations are taken into account (deviate with probability p). In the description of the protocols and in the proofs, keep in mind that deviations can be randomized and performed with some probability.

Lastly, we define the payoff function. One of the driving forces in the system is that every node has the ultimate interest of being part of the system and to get a service of good quality. At the same time, every node is selfish and has a rational tendency to save energy and memory. We can measure energy cost in joules and memory cost in bytes seconds (clearly, using one KByte of memory for one second or for one year does not have the same cost). Therefore, we can define f in a very general way as a function $f(s) = (f_1(s), \ldots, f_n(s))$ of the strategy profile such that for all i $f_i(s)$ is strictly positive and

1. $f_i(s) = 0$ if player i, as a consequence of strategy profile s, has a nonnegligible probability, say at least 50 percent, of not being able to send or receive messages with the same performance of the original protocol.
2. otherwise, $f_i(s)$ is decreasing when either the expected value of energy cost or the expected value of memory cost required by the protocol increase.

Our goal is to show that, if π is G2G Epidemic Forwarding, then the resulting strategy profile $(\pi, \ldots, \pi)$ is a Nash equilibrium. That is, no node has an incentive to unilaterally deviate from the protocol. The proof is quite technical and long, we organize it into two lemmas. We will consider player i, and show that, if player i deviates by executing a protocol $\pi' \neq \pi$, then $f_i(\pi') \leq f_i(\pi)$, that is, player i does not do better by deviating (either it is not able any more to send and receive messages with the same performance or it uses more energy or memory).

Lemma 1. *A rational node will follow all the steps of the relay phase truthfully.*

Proof. Assume that π is G2G Epidemic Forwarding. Strategy π is composed by eight steps—the five steps in Fig. 1 and the three steps in Fig. 2. Assume that $s = (\pi, \ldots, \pi)$, $\pi' \neq \pi$, and strategy profile s' is equal to s except in position i, where the entry is π'. This is the formal way to say that in strategy profile s' player i has unilaterally chosen to deviate from protocol π (our G2G Epidemic Forwarding) to protocol π'. We show that, if $\pi' \in \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_5$, then $f_i(s') \leq f_i(s)$ (in other words, if a node deviates from the relay phase then its payoff is reduced). Recall that $\mathcal{P}_j$ contains all the protocols that are equal to π in the first $j-1$ steps and that deviate starting from step j. We organize the proof in cases, from $\pi' \in \mathcal{P}_1$ to $\pi' \in \mathcal{P}_5$. These cases collectively cover all possible ways to deviate from the relay phase.

Let $\pi' \in \mathcal{P}_1$. That is, player i deviates from step 1 of the relay phase (see Fig. 1, where player i has the role of node A). If player i deviates when it is the source of the message, then player i is not able to deliver its own messages any more. Therefore, payoff $f_i(s') = 0 \leq f_i(s)$ and we are done. On the other hand, if node A is an intermediate relay, node A has a nonnegligible probability to meet the previous relay and that it will be asked to show two proofs or relay (later on, in our experimental results, we will see that the probability of detection is higher than 90 percent, therefore the probability of meeting again the previous relay for at least one message is also more than 90 percent) or to show that he still has the message and perform the puzzle P and thus consuming more energy, in expectation. Note, however, that a strategy where A deliberately choses to not relay the message and keep it in memory at the cost of having to compute the heavy puzzle each time it meets the previous relay is perfectly legitimate. However, since the puzzle P is a cost-function [35], [36] computationally expensive to compute, the energy consumed is higher than the energy saved by storing the message without relaying it. Thus, we get $f_i(s') \leq f_i(s)$. Even so, there might be rare cases in which A knows beforehand that, with high probability, it will not meet the previous relay soon enough. Thus, A might chose to not forward the message to B and keep it in his memory: The risk of having to compute the heavy puzzle is very low, and A knows it, i.e., in these rare cases $f_i(s') \geq f_i(s)$. However, as our experiments show, these events are very rare in social mobile networks, in which most the meetings are typically between people that know each

other or have something in common to each other which makes them frequent the same places over and over again (e.g., go to the same gym, take the same bus to go to work in the morning, etc.). Thus, this strategy adopted by node A very rarely (only for previous relays which he knows he will not see soon), does not impact neither the delivery ratio nor the other performance metrics of the G2G protocol.

We get again $f_i(s') \leq f_i(s)$ if the node simply drops the message an so it is not able to perform any of the two operations, indeed it is discarded from the system and his payoff drops to zero. Moreover, node A is interested in getting rid of the message as soon as possible, since a message typically uses much more memory than the proofs of relays and it does not want to be in the position of performing the heavy puzzle P in step 2 of the test phase. Therefore, we have shown that, if $\pi' \in \mathcal{P}_1$, then $f_i(s') \leq f_i(s)$. In other words, player i executes step 1 of the protocol truthfully.

Let $\pi' \in \mathcal{P}_2$. That is, player i deviates from step 2 of the relay phase (see Fig. 1, where player i has the role of node B). Let's consider node B in the relay phase. Once node B is asked to be a relay of a particular message m, it does not know who is the destination of m. If player i deviates from the protocol and declines to be a relay (either it does no respond or does not send RELAY_OK), it will never receive any message, since it will discover both message content and message destination only at step 5. Consequently, $f_i(s') = 0$.

The case $\pi' \in \mathcal{P}_3$ can be dealt exactly as the case $\pi' \in \mathcal{P}_1$, and the case $\pi' \in \mathcal{P}_4$ as the case $\pi' \in \mathcal{P}_2$.

Lastly, player i will execute step 5 truthfully. Let $\pi' \in \mathcal{P}_5$. Indeed, the message sent in step 5 is very short and, if the key is not sent, node B freezes the session with node A until it gets the key. In case A deviates to a protocol in which step 5 is not executed, it won't be able to send and to receive any message through node B and, after many frozen sessions, it won't be able to send or receive any message at all (or with a limited performance), in such a way to make $f_i(s')$ drop to zero. □

Let's consider the test phase.

Lemma 2. *A rational node will follow all the steps of the test phase truthfully.*

Proof. Again, assume that $s = (\pi, \ldots, \pi)$, $\pi' \neq \pi$, and strategy profile s' is equal to s except in position i, where the entry is π'. Just like in Lemma 1, we show that if $\pi' \in \mathcal{P}_6 \cup \cdots \cup \mathcal{P}_8$, then $f_i(s') \leq f_i(s)$.

Let $\pi' \in \mathcal{P}_6$. That is, node A does not start the test phase. If this is the case, then node B will never relay node A's messages and therefore the quality of service experienced by A is reduced. That is, $f_i(s') \leq f_i(s)$. Now, let $\pi' \in \mathcal{P}_7$. That is, player i deviates from step 2 of the test phase (see Fig. 1, where player i has the role of node B). Again, we can show that $f_i(s') \leq f_i(s)$. Indeed, the message sent in step 2 of the test phase is very short and, if the proof is not sent, node A freezes the session with node B until it gets the proof. In case B deviates to a protocol in which step 2 is not executed, it won't be able to send and to receive any message through node A in such a way to make $f_i(s')$ drop to zero. Finally, node B will not risk avoid computing the heavy puzzle and send to A a fake value: If A still has the message or is the sender (who always keeps the message till it expires), B will get busted and reported by A, thus, thrown out of the system.

Lastly, we can see that node A will perform step 3 of the test phase (Fig. 1) truthfully. Indeed, if A does not store the proof received from B into token T_{AB}, then A is not able to start a new relay phase with node B. □

Note that it is not a rational strategy to shut off the radio every time node B meets node A in such a way to avoid the test phase. Indeed, in this case node B will not receive other messages destined to itself that node A or other nodes might have during that interval of time. Therefore, node B would experience a reduced quality of the service that makes its payoff drop. Moreover, note that session interruption due to node mobility is not a problem for none of the phases of the protocol. Indeed, in such case the session will resume from where it was interrupted as soon as the nodes get in touch again with each other.

To summarize, if a node deviates from G2G Epidemic Forwarding (deterministically or following a probability distribution) then its payoff is reduced. Since the detection happens right away (as we will see from our experiments), rational users will follow the protocol truthfully.

Theorem 1. *G2G Epidemic Forwarding is a Nash equilibrium.*

4.4 Coalitions

Consistent with a very large part of the literature, in this paper we assume that nodes do not collude. This is a common assumption, coordination has a cost and in many practical settings people just do not trust each other enough to form a coalition. However, if a few nodes have a strong will to cheat the system, they could deviate in a coordinated manner from our protocols and do better. While this attack is out of the scope of this paper, we would like to point out a few countermeasures that can be worked out to mitigate its effect.

Assume that a set $\mathcal{C}$ of nodes, the *coalition*, deviate to a protocol in which the nodes mark in some secret way the body of messages coming from members of the coalition, exchange fake proofs of relay when handling messages outside of the coalition, never execute the test-phase within the coalition, and drop messages that are not originated or destined to members of the coalition. Of course, they stick to the protocol when in session with users outside the coalition. To stop this one and similar attacks, we can put in place two mechanism: *Random checks of conformity* and *rewarding traitors*.

Random checks of conformity. The sender of a message, with small probability p, sends the proof of relays got from first relay B on the paths toward destination to the authority. The authority, at a random time before the time-out, checks whether node B has collected two proofs of relay or it is still storing the message. Then, the authority can follow the message at the next relays and check whether the message has not disappeared. The sender has an interest to perform this check, and node B does not know if the previous relay is the sender or not. This check can be very costly since the authority uses the expensive cellular network infrastructure.

However, since the authority can impose a very stiff penalty, like the eviction from the system, this mechanism can be a good deterrent even with an extremely small probability p, in such a way that the cost can be reasonably low.

Rewarding traitors. A simple observation is that any member of the coalition can prove that another node is a member as well. Suppose that node $A \in \mathcal{C}$, without actually exchanging message, gets a fake proof of relay from node $B \in \mathcal{C}$. Now, node A has a proof that can nail node B. If node A gives away the fake proof of relay signed by node B to the authority, then the authority can ask node B for a proof of storing the message (or two proofs of relays) and node B will not be able to respond. If the authority rewards node A in such a way that the benefit of betraying is larger than the benefit of being part of the coalition, this attack can be prevented efficiently.

These mechanisms, among others, can be used to extend our protocols in such a way to mitigate the possible presence of coalitions or to limit the possible harm they can make, including the protocols described in the next section. However, the full development of these mechanisms is out of the scope of this paper. Therefore, we stick to the classical assumption that nodes do not collude and proceed with more sophisticated, social-aware forwarding protocols.

5 Give2Get Delegation Forwarding

Delegation Forwarding [4] is a class of protocols that have been shown to perform very well. In Delegation Forwarding, every node is associated with a *forwarding quality*, that may depend on the destination of the message at stake. When a message is generated, it is associated with the forwarding quality of the sender. Then, the message is forwarded from node to node, creating a new replica of the message at each step, according to the following protocol: When a relay node A gets in contact with a possible further relay B, node A checks whether the forwarding quality of B is higher than the forwarding quality of the message. If this is the case, node A creates a replica of the message, labels both messages with the forwarding quality of node B, and forwards one of the two replicas to B. Otherwise, the message is not forwarded.

Delegation Forwarding, in many of its flavors, has been shown to reduce considerably the cost of forwarding (that is, the number of replicas), without reducing considerably success rate and delay. However, just like Epidemic Forwarding, it is far from being a Nash equilibrium. A selfish node can easily send messages and receive messages without taking care of relaying any other message. It is also easy to see that it is not enough to translate all the techniques used in G2G Epidemic Forwarding in order to get a version of Delegation Forwarding that is a Nash equilibrium.

Simply speaking, the techniques we developed to build G2G Epidemic Forwarding can be used to stop message droppers in G2G Delegation Forwarding, but are not enough to make it a Nash Equilibrium. Indeed, selfish nodes have many other rational ways to deviate in these more sophisticated protocols. First, nodes can lie on their forwarding quality. They can claim that their quality is zero and get their messages served without participating actively. We will call these nodes *liars*. Not only that, selfish nodes can change the forwarding quality of the message to zero, in such a way to get rid of the message soon—they would be able to relay it to the first two nodes they meet. We will call these nodes *cheaters*. Of course, cheaters are less vicious than liars, in our setting. However, we will show how to build a version of Delegation Forwarding that is a Nash equilibrium. Just like what we did with G2G Epidemic Forwarding, our approach is not to add patches against liars and cheaters or incentives for altruistic nodes, our approach is to design a protocol such that, step by step, it can formally be shown that every rational player in the protocol cannot but follow the protocol truthfully. In this way, we protect our system against liars, cheaters, and any other possible way to deviate rationally.

Here, we consider Delegation Destination Frequency and Delegation Destination Last Contact (DDLC) [4].

Delegation Destination Frequency. Node A forwards message m to node B if node B has contacted $m's$ destination more frequently than any other node that the copy of the message m carried by A has seen so far.

Delegation Destination Last Contact. Node A forwards message m to node B if node B has contacted $m's$ destination more recently than any other node that the copy of the message m carried by A has seen so far.

In the above definitions the forwarding quality q_{AB} is, respectively, the number of encounters between node A and node B, and the time of the last encounter between node A and node B. Since encounters are committed by both nodes with a commonly agreed time, it is easy to see that $q_{AB} = q_{BA}$ for every pair of nodes A and B.

G2G Delegation Forwarding builds upon all the techniques that we have developed for G2G Epidemic Forwarding. First, in G2G Delegation Forwarding the quality of the message is changed only when forwarded (we will see later why). G2G Delegation Forwarding consists of four phases: Message generation, relay, test by the sender, and test by the destination. Message generation is just like message generation and in G2G Epidemic Forwarding. Thus, in the next sections, we will describe only the phases that are substantially different from G2G Epidemic Forwarding. As in G2G Epidemic, the relay and the test phases are based on the idea of making nodes collect proofs of relay and to check relays about their behavior with the message. The test by destination phase is to protect the system against cheaters: *very low* forwarding qualities declared by nodes are "embedded" in the message by the source, and, when eventually the destination is reached, are double-checked (forwarding qualities are symmetric) by the destination. In the remaining, we give details on the phases.

5.1 G2G Delegation Forwarding: The Relay Phase and the Test by the Destination Phase

Fig. 3 shows the protocol of the relay phase. Just like G2G Epidemic Forwarding, node A has an interest to start this phase, since it has to collect the proof of relay for the message. In step 1, node A asks B what is its forwarding quality to destination D (denoted by q_{BD}). Note that forwarding quality q_{BD} must be *provable*. That is, it must be computed based on signed interactions. This is why nodes A and B exchange signed messages carrying q_{AB} and q_{BA} at every session. Note that, if $q_{AB} \neq q_{BA}$, then one of the

$$A \xrightarrow{\langle \text{FQ\_RQST}, H(m), D', T_{AB}, q_{AB}\rangle_A} B \quad (1)$$

$$A \xleftarrow{\langle \text{FQ\_RESP}, B, D', q_{BD'}, q_{BA}\rangle_B} B \quad (2)$$

$$A \xrightarrow{\langle \text{RELAY}, H(m), q_m, E_k(m)\rangle_A} B \quad (3)$$

$$A \xleftarrow{\langle \text{POR}, H(m), A, B, D', q_m, q_{BD'}\rangle_B} B \quad (4)$$

$$A \xrightarrow{\langle \text{KEY}, H(m), k\rangle_A} B \quad (5)$$

Fig. 3. G2G Delegation Forwarding: Protocol of the relay phase.

two players is cheating, and the loyal player can prove it based on the quality exchanged in the previous session. Again, in the request node A includes a token that certifies that the last interaction has completed with the test phase. Node B replies with its forwarding quality (we will see later why B has no interest in lying). When the destination of m is different from B, D' is the actual destination D; when the destination of m is B, D' is chosen as a random node different from B. This mechanism has the goal of making it impossible to B to know whether it is the destination of the message or not before taking the message and giving the proof of relay. Therefore, just like in G2G Epidemic Forwarding, node B will follow all the relay protocol with the hope of being the actual destination of the message. Note that in G2G Delegation Forwarding the proof of relay contains much more information, including the forwarding quality toward D claimed by node B and the forwarding quality of the message at that point in time.

Node A forwards the message to two other nodes. In the case when node A is also the sender of the message, A stores the signed message $\langle \text{FQ\_RESP}, B, D, q_{BD}\rangle_B$ for the nodes B that failed to be good relays for the message, that is $q_{BD} < q_m$ (recall that $D = D'$ except the case in which B is message's destination). As soon as node A finds a good relay, the last two signed qualities of such failed relays are embedded into the message toward D. If the destination D receives the message, it will be able to check if q_{BD} is correct or not (this is the *test by destination* phase). Indeed, q_{BD} should be equal to q_{DB}. Since nodes B and C do not know whether A is the sender or not, they will not lie about their forwarding quality since there is a nonnegligible probability to be tested by the destination (in the experiments we will see that this is exactly the case). When D detects that node B is a liar, he sends the proof to the authority by using the cellular network infrastructure. Indeed, node D can prove that the correct quality is different based on the quality signed by node B himself during their last interaction. The authority does not need to contact node B and can proceed with its revocation from the system.

In order to make this mechanism work, the forwarding quality q_{BD} is not the *current* quality, it is the quality computed *in the last completed time frame*. Every node keeps three versions of the forwarding quality, the current and the two forwarding qualities computed in the previous two completed time frames. In this way, B and D have a consistent notion of forwarding quality. Of course, the time frame has to be set in such a way that, with high probability, the message delay falls within one of the last two completed time frames, so that the destination has the necessary information to detect liars. As a consequence of this set of techniques, no relay will lie about their forwarding quality—we will see in the experiments that, in case of deviation, the probability of being removed from the system is actually very high.

5.2 G2G Delegation Forwarding: The Test Phase

The test by the sender is executed by the relay node as in G2G Epidemic Forwarding. Assume that node B has received the message from the node A. When B gets in contact with A again, node B is tested and, just like in G2G Epidemic Forwarding, it gives the two proofs from relays C_1 and $C_2\langle \text{POR}, H(m), B, C_1, D, q^1_m, q_{C_1D}\rangle_{C_1}$ and $\langle \text{POR}, H(m), B, C_2, D, q^2_m, q_{C_2D}\rangle_{C_2}$ to node S. In this way, it is guaranteed that it is not rational to become a message dropper. More than that, this phase is also important to check that B is not a cheater, that is it has not reduced the message quality q_m to get rid of the message quickly. Indeed, A can check whether

$$q_{BD} = q^1_m < q_{C_1D} = q^2_m < q_{C_2D}.$$

The second equality in this equation is true since the quality of the messages is changed only when forwarded. Since we are sure that nodes do not lie, than we know that q_{BD}, q_{C_1D}, and q_{C_2D} are sound. Therefore, also $q^1_m = q_{BD}$ and consequently node B has not selfishly modified the forwarding quality of the message to convince B to take it. Similarly, we also know that $q^2_m = q_{C_1D}$ and so node B has not cheated with node C_2 as well. Note that it is *not* possible for B to forge fake proofs or fake forwarding quality declarations of another node C. Indeed, as in G2G Epidemic, proofs and forwarding qualities are signed.

Again (as in G2G Epidemic), node A will store the proofs received by node B as token T_{AB} until expiration of time-out Δ. Lastly, if A does not start the test phase as required by the protocol, node B will not relay any message whose destination is node A until time-out Δ expires.

To summarize, by using the techniques developed for G2G Epidemic Forwarding and specific techniques for G2G Delegation Forwarding, we can get the following result:

Theorem 2 G2G Delegation Forwarding. *is a Nash equilibrium.*

6 Experiments

6.1 Selfishness and Selfishness with Outsiders

In social environments, it is natural to consider two different ways of being selfish. The first is just selfishness—nodes that can deviate from the protocol with the goal of maximizing their personal interest. The second is *selfishness with outsiders*—nodes that can deviate from the protocol for their personal interest only when this does not damage people from the same community. This notion is natural since it comes from our personal experience: Some people can tend to be truthful with those they care about, and selfish with outsiders. Formally, it is just vanilla selfishness with a different objective function. However, it is useful to define it as an independent notion. To implement selfishness with outsiders, we use the *k-clique* algorithm [37] (also used in [5]) for community detection on each data trace. Nodes that are selfish with outsiders deviate from the protocol only in sessions with nodes from other communities.

TABLE 1
The Three Experimental Data Sets

Data set	Cambridge 06	Infocom 05	Infocom 06
Duration (days)	11	3	3
Devices #	36	41	78
Contacts #	10,873	22,459	191,336
Avg. Contacts/pair/day	0.345	4.6	6.7

6.2 The Data Set

For our experiments we have used three traces collected during experiments done with real devices carried by people. We will refer to these traces as *Infocom 05*, *Infocom 06*, and *Cambridge 06*. Characteristics of these data sets such as intercontact and contact distribution have been observed in several previous works [1], [38], [2]. *Cambridge 06* [39] is a data set collected by distributing Intel iMotes to students of Cambridge University. The iMotes were programmed to log contacts of all visible mobile devices and a number of stationary nodes (that we do not use in this paper). The number of nodes is 36. This data set covers 11 days. *Infocom 05* [40] is a data set collected during the Infocom 2005 student workshop. The number of devices is 41. This experiment covers approximately three days. *Infocom 06* [41] is similar to Infocom 05 except that the scale is larger, with 78 participants from the Infocom 2006 conference. Further details on the real traces we use in this paper are shown in Table 1.

6.3 Experiments on Detecting Deviations

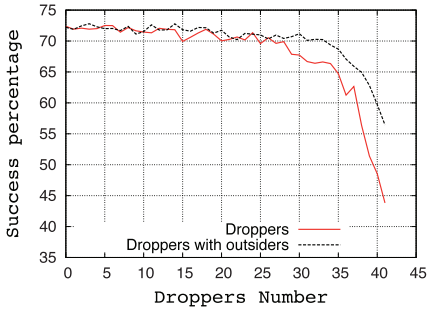
Here, we study the impact of selfish behavior on Epidemic Forwarding. Then, we turn to G2G Epidemic Forwarding and test how good is our protocol in detecting possible deviations.

6.3.1 *Impact of Selfish Behavior on Epidemic Forwarding and Detection of Deviations in G2G Epidemic Forwarding*

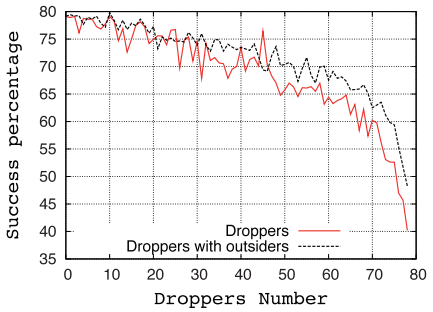
For each trace and forwarding protocol we use the same assumptions and the same way of generating traffic to be routed as in [4]: A set of messages is generated with sources and destinations chosen uniformly at random, and generation times from a Poisson process averaging one message every 4 seconds. We isolated 3-hour periods for each data trace. Each simulation runs therefore 3 hours. To avoid end effects no messages were generated in the last hour of each trace. The nodes are assumed to have infinite buffers. We set the message TTL value to be the smallest one that maximizes the success rate of Epidemic Forwarding in each scenario, in the sense that the performance is the same as if messages never expire. These values are, respectively, 30 minutes for the two Infocom scenarios and 35 minutes for the Cambridge one.

Fig. 4 shows how the success rate[1] of Epidemic Forwarding is affected by the presence of message droppers. In our experiments these nodes drop messages right after the end of the relay phase (see Fig. 1). As you can see, the performance of the protocol drops to around 50 percent, which is unacceptably low. Basically, when all the nodes are droppers, the only hope for success is that the sender gets personally in contact with the destination. It is not much different when we consider selfishness with outsiders (i.e., nodes who drop messages right after the end of the relay phase with an outsider). In the experiments we have used the same notion of community as in [5].

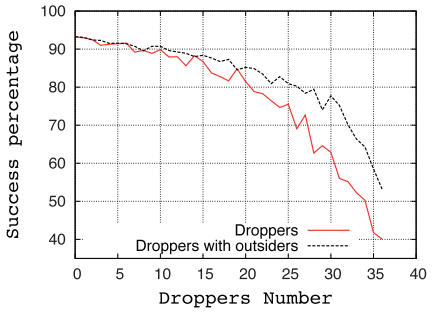
1. The success rate is under 100 percent when the number of misbehaviors is 0 because of the intermittent connectivity in the network.



(a) Infocom05



(b) Infocom06



(c) Cambridge06

Fig. 4. Effect of message droppers on Epidemic Forwarding.

Next, we focus on the detection of message droppers. As we already mentioned, Δ denotes the time interval within which messages, proofs of relay, tokens (and everything else that G2G Epidemic requires) is kept in the memory of relay nodes. After this time interval nodes can safely discard everything related to the message. Thus, in order to get best delivery rate, Δ is bounded from below by the optimal TTL. On the other side, Δ has to be large enough to allow droppers to be detected. Our experiments show that it is already safe to set $\Delta = \text{TTL}$ in every scenario.

According to our experiments, shown in Table 2, detection probability is more than 94 percent in the selfish case and more than 91 percent in the selfish with outsiders case. Deviations are detected very quickly (on the order of minutes) and the time does not depend on the number of the nodes that deviate (see Fig. 5). So, we can safely say that our assumption, that nodes are tested with nonnegligible probability, is sound. For more details on the rate and time of detection see Table 2.

TABLE 2
Performance of G2G Epidemic on the Real Traces

	Infocom 05		Infocom 06		Cambridge 06	
	Detec. Rate	Avg detec. time	Detec. Rate	Avg detec. time	Detec. Rate	Avg detec. time
Droppers	94.7%	12.1	95.8%	17.2	93.4%	21
Droppers with outsiders	91.3%	13.3	93.2%	18.1	91.6%	27.3

Detection time (in minutes) is measured starting from the moment in which a given dropper discards the message.

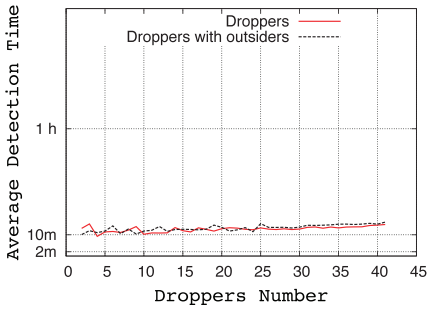
6.3.2 Detection of Deviations in G2G Delegation Forwarding

Here, we consider message droppers, liars, and cheaters. Note that it is not rational to be a cheater in Delegation Forwarding—if a node labels a message with forwarding quality zero, then it will have to relay it with higher probability, doing more work. It is different in G2G Delegation Forwarding, where cheaters can find two relays much more quickly, discard the message, and save storage space. Recall that *no* deviation is rational in G2G Delegation Forwarding—it is a Nash equilibrium—and these experiments on detection are important just to make sure that our assumption that every node has a nonnegligible probability of being tested during the test phase is sound. Here, we present the results for Delegation Destination Last Contact. Delegation Destination Frequency, as far as detection of deviations is concerned, behaves in a very similar way. In all our experiments we set Δ to be 50 minutes in the Infocom scenarios and 80 minutes in the Cambridge scenario. In the experiments, droppers are again those who drop the message immediately, liars are those who always report a node forwarding quality equal to 0, while cheaters are those who bring down the quality of the message to be relayed (in order to get rid of it as soon as possible). We ran a set of experiments on Delegation Forwarding. Figs. 6 and 7 show the results, that clearly indicate that both droppers and liars have a big impact on the success rate, both in the case of selfishness and in the case of selfishness with outsiders. Then, we have run experiments to see how reliably these deviations are detected by the protocols in all the three traces, Infocom 05, Infocom 06, and Cambridge 06. According to our results, droppers and liars are detected with a probability that exceeds 80 percent, whereas cheaters are detected with a probability that exceeds 60 percent in all scenarios. The detection rate remains high even in the case of selfishness with outsiders. Table 3 shows all the results. Note that the detection probabilities are a bit lower for the Cambridge 06 scenario, which is in accordance with its lower contact frequency with respect to the Infocom scenarios (see Table 1). In all cases, this is enough to say that the probability of being detected is not negligible.

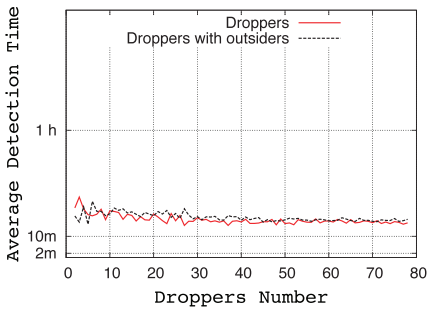
Then, our question is how fast is the detection. In Fig. 8 we can see that in all scenarios droppers are detected sooner than liars that are detected sooner than cheaters and that the speed of detection does not depend on the number of deviating nodes. Moreover, detailed experimental results shown in Table 3 indicate that nodes cannot hope to deviate and remain in the system for long time. G2G Delegation Forwarding is clearly very fast in the detection in all scenarios. Note that these experiments are just to prove that the probability of detection is very high in our system. Therefore, our mathematical assumptions are realistic and the threat of being removed from the system is actually a deterrent. As a consequence, no node deviate and there is no need for the authority to revoke (by using, e.g., GSM) any node. This is why the cellular infrastructure is not considered in our experiments.

6.4 Experiments on the Performance: Number of Replicas and Delay

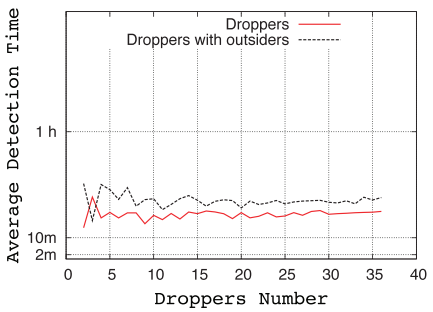
Initially, our goal was to show that adding all the mechanisms needed to make the protocols Nash equilibria does not reduce the *success rate* and the *delay*, and does not increase the *cost* in



(a) Infocom05

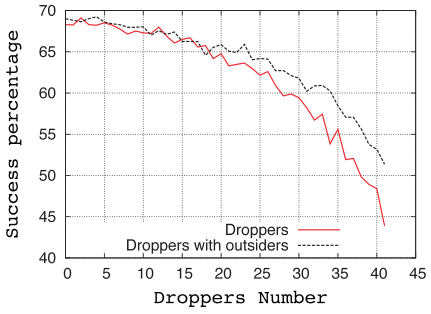


(b) Infocom06

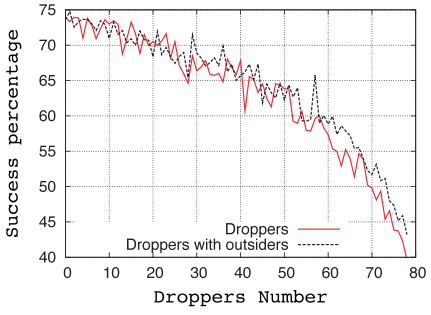


(c) Cambridge06

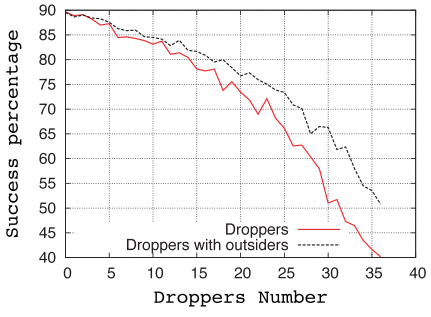
Fig. 5. Dependence of droppers detection time by the number of droppers in G2G Epidemic Forwarding. Detection time (in minutes) is measured starting from the moment in which a given dropper discards the message.



(a) Droppers in Infocom05

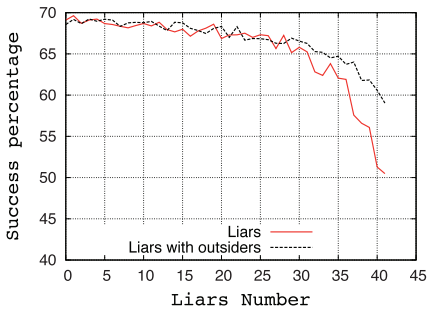


(b) Droppers in Infocom06

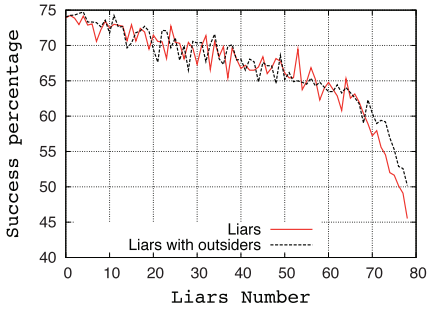


(c) Droppers in Cambridge06

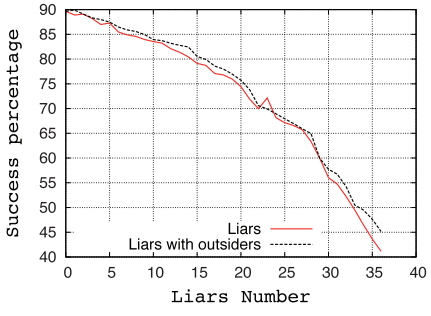
Fig. 6. Effect of message droppers on Delegation Forwarding.



(a) Liars in Infocom05



(b) Liars in Infocom06



(c) Liars in Cambridge06

Fig. 7. Effect of message liars on Delegation Forwarding.

terms of number of replicas. During the protocols design, we realized that the mechanism used to reduce the number of relays to two, besides being fundamental to show that the protocol is a Nash equilibrium, has the interesting property that the message more cheaply flows far from the community that generated it. Cheaply in the sense that fewer replicas need to be generated to reach destinations that are far from the sender in terms of community. Fig. 9 summarizes a long set of experiments on success rate, delay, and cost. Indeed, the results might seem surprising—G2G Epidemic Forwarding is much better than Epidemic Forwarding in terms of cost (the improvement is around 20 percent), and G2G Delegation Forwarding is considerably better than Delegation Forwarding, again in terms of cost, while the performance in terms of delay and success rate very close to the original, more expensive, protocols. Note that the experimental setting that we have chosen is considered to be standard in the literature.

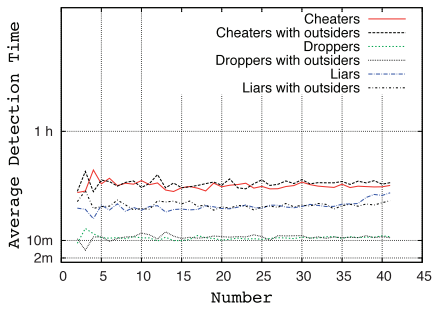
6.5 Experiments on the Performance: Storage and Communication Overhead

Another natural question is whether the storage overhead of G2G protocols is high. Indeed, G2G protocols introduce

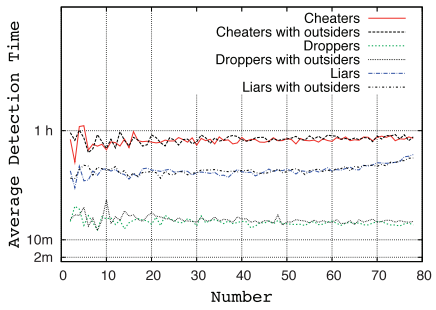
TABLE 3
Performance of G2G Delegation on the Real Traces

	Infocom 05		Infocom 06		Cambridge 06	
	Detec. Rate	Avg detec. time	Detec. Rate	Avg detec. time	Detec. Rate	Avg detec. time
Droppers	88%	12	89%	19	86%	21
Liars	67%	26	71%	28	65%	52
Cheaters	83%	35	85%	37	84%	64
Droppers with outsiders	87%	15	86%	21	84%	23
Liars with outsiders	64%	28	68%	29	62%	54
Cheaters with outsiders	83%	37	84%	39	81%	68

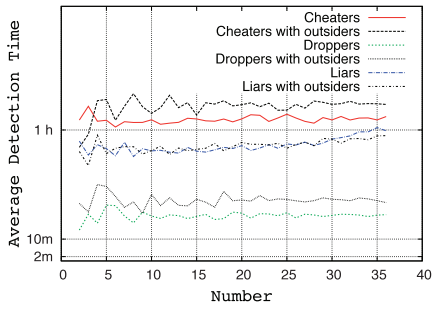
Detection time (in minutes) is measured starting from the moment in which a given dropper discards the message.



(a) Infocom05



(b) Infocom06



(c) Cambridge06

Fig. 8. Dependence of detection time from the number of selfish individuals in G2G Delegation Forwarding. Detection time (in minutes) is measured starting from the moment in which a given dropper discards the message.

PORs to be stored with the messages. For this reason, we perform a detailed analysis of the memory required by each of the protocols. Clearly, we also include the overhead due to the G2G mechanisms. To the best of our knowledge, this is the first such analysis in the literature for Pocket Switched Networks.

The overhead depends on the size of the messages in the system. In our study, we consider two types of messages: Short messages (SMS or Tweets) and long messages (MMS). We assume that each message is composed by the header and the message body, and that the header is 32 bytes long. We assume that short messages have a body of 140 bytes. (This is actually the length of a regular SMS, 160 ASCII characters encoded into 140 bytes, 7 bits per character; while tweets have a limit of 140 UTF-8 characters, which is again 140 bytes.) Long messages have a body of 600 kbyte. (The length of a standard MMS according to AT&T.) We use SHA-256 as the hash function, therefore hashes are 32 bytes long. Signatures are computed by using elliptic curve cryptography [33], therefore signatures are 320 bits long. The metric we use for storage is kilobyte hour (kBh). If you take Epidemic Forwarding, the analysis is easy. When a message is received by a relay node, the message is stored until the TTL lapses. In the case of long messages in a scenario like Infocom 06, where the TTL is 30 min, the cost incurred by the node to store the message body is 600 kbyte times 30 min, that is 300 kBh. This metric is intuitive, it is clearly cheaper to store one MMS for a minute versus storing the same MMS for a hour.

The analysis for G2G protocols is more complicated. Take, as an example, G2G Epidemic Forwarding. When a message is received by a relay node, the storage used is just the message—header and body. When the first relay is found, the node has to store a POR along with the message. When the second relay is found, the node has to store a second POR, but it can discard the message. The PORs can be discarded when the node is tested by the previous relay, but a token must be stored by the tester according to the protocol. The size of the POR depends on the protocol: In G2G Epidemic Forwarding, we can see from Fig. 1 that a POR is composed by the hash of the message and by the IDs of the two relays, together with a signature. In G2G Delegation Forwarding the POR is somewhat longer, since it stores other information as described in Fig. 3.

We have performed a complete analysis of the storage requirement of all the protocols discussed in this paper by

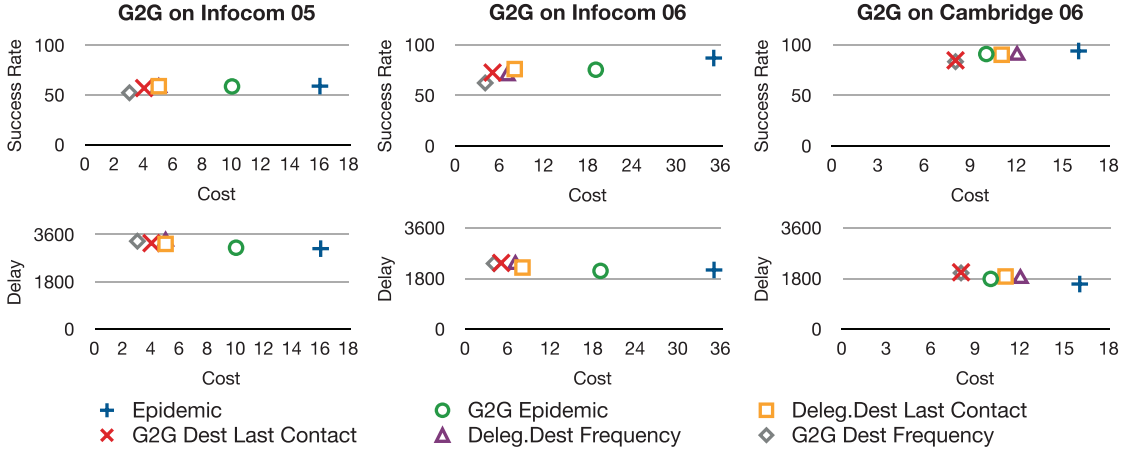
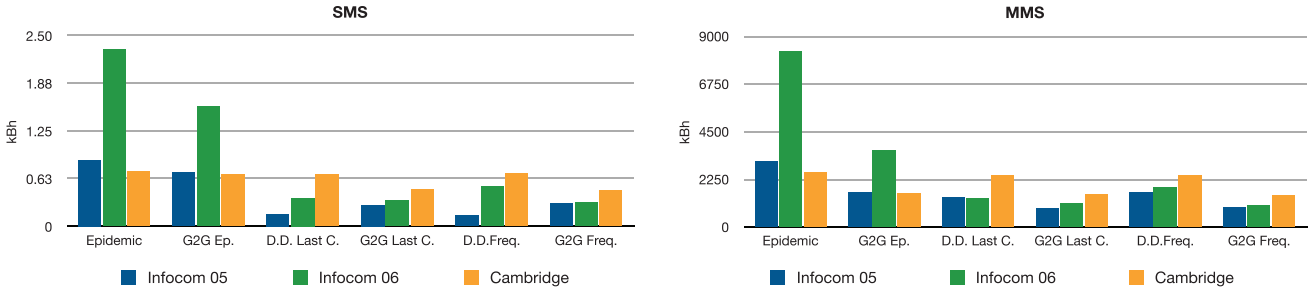


Fig. 9. Performance of G2G Epidemic Forwarding and G2G Delegation Forwarding compared with Epidemic Forwarding and Delegation Forwarding.



(a) Memory occupation for short messages

(b) Memory occupation for long messages

Fig. 10. In the figure "D.D.Last C." refers to Delegation Destination Last Contact and "D.D.Freq" to Delegation Destination Frequency. The G2G prefix indicates the respective Give to Get version of each protocol.

using the same traces we have used for the other experiments. The results are shown in Figs. 10a and 10b for short messages and the long messages, respectively. Quite surprisingly, the memory occupation of the G2G protocols, even including the overhead induced by the PORs that we have used to make them Nash Equilibria, is almost always smaller than their vanilla alter egos. If we consider short messages, only in the specific scenario of Infocom 06 and protocol DDLC the overhead is about 10 percent; in all the other cases the storage overhead of G2G mechanisms ranges from -5 to -32 percent (that is, G2G mechanisms actually reduce storage requirements!). If we consider long messages, in Infocom 06 and DDLS the overhead of G2G mechanisms is about -17 percent, while in all other cases it ranges from -37 to -55 percent. The reason is that in G2G protocols the nodes, to optimize their own payoff function, discard the message as soon as the first two relays are found. This is more than enough to balance the overhead due to storing proofs of relays and other additional information.

Lastly, since the G2G mechanisms makes nodes exchange more information (e.g., proofs of relay), one might think that they incur high-communication overhead. However, the execution of a Relay+Test phase induces a relatively small (422 Bytes for G2G Delegation and 386 Bytes for G2G Epidemic) overhead per message, independent from the message payload. Moreover, since G2G mechanisms reduce the number of replicas distributed in the network, the actual communication overhead is low. In Table 4 we show the average (per message) information incurred to be exchanged in the network during forwarding. We can see that, when short messages like SMS are taken into consideration, the overhead of G2G protocols is actually reasonably low compared to their vanilla alter egos. While for longer messages like MMS, G2G protocols have actually a smaller overhead to their vanilla alter egos due to the reduced number of replicas. Because of space limitations we only show results with the Infocom 05 scenario. However, the other scenarios present similar results.

TABLE 4
Average Information Transmitted per Message in the Network

	Epidemic		Delegation	
	Vanilla	G2G	Vanilla	G2G
SMS	3 KB	4.59 KB	1.56 KB	2.1 KB
MMS	9001 KB	5400 KB	2400.3 KB KB	1200 KB

7 Conclusions

In this paper, we have presented G2G Epidemic Forwarding and G2G Delegation Forwarding, the first protocols for message forwarding that work under the assumption that all the nodes in the network are selfish. We formally show that the G2G protocols are Nash equilibria. Quite surprisingly, G2G protocols also outperforms their alter egos in terms of cost, while being almost as good in terms of success rate and delay.

Acknowledgments

A. Mei is supported by a Marie Curie Outgoing International Fellowship funded by the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement n. 253461.

References

[1] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket Switched Networks and Human Mobility in Conference Environments," *Proc. ACM SIGCOMM Workshop Delay-Tolerant Networking (WDTN '05)*, 2005.

[2] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms," *Proc. IEEE INFOCOM '06*, 2006.

[3] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," Technical Report CS-200006, Duke Univ., 2000.

[4] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation Forwarding," *Proc. ACM MobiHoc '08*, 2008.

[5] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble Rap: Social-Based Forwarding in Delay Tolerant Networks," *Proc. ACM MobiHoc '08*, 2008.

[6] E.M. Daly and M. Haahr, "Social Network Analysis for Routing in Disconnected Delay-Tolerant Manets," *Proc. ACM MobiHoc '07*, 2007.

[7] F. Li and J. Wu, "LocalCom: A Community-Based Epidemic Forwarding Scheme in Disruption-Tolerant Networks," *Proc. IEEE CS Sixth Ann. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks (SECON '09)*, 2009.

[8] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in Delay Tolerant Networks: A Social Network Perspective," *Proc. ACM MobiHoc '09*, 2009.

[9] A. Mei, G. Morabito, P. Santi, and J. Stefa, "Social-aware Stateless Forwarding in Pocket Switched Networks," *Proc. IEEE INFOCOM '11*, 2011.

[10] A.S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth, "Bar Fault Tolerance for Cooperative Services," *Proc. 20th ACM Symp. Operating Systems Principles (SOSP '05)*, 2005.

[11] H.C. Li, A. Clement, E.L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "Bar Gossip," *Proc. Seventh Symp. Operating Systems Design and Implementation (OSDI '06)*, 2006.

[12] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proc. MobiCom '00,* 2000.

[13] S. Buchegger and J.-Y.L. Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes Fairness in Dynamic Ad-Hoc NeTworks," *Proc. IEEE/ACM MobiHoc '02,* June 2002.

[14] K. Balakrishnan, J. Deng, and V. Varshney, "Twoack: Preventing Selfishness in Mobile Ad Hoc Networks," *Proc. IEEE Wireless Comm. and Networking Conf.,* vol. 4, Mar. 2005.

[15] P. Michiardi and R. Molva, "Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," *Proc. IFIP TC6/TC11 Sixth Joint Working Conf. Comm. and Multimedia Security,* 2002.

[16] L. Buttyán and J.-P. Hubaux, "Enforcing Service Availability in Mobile Ad-Hoc Wans," *Proc. ACM MobiHoc '00,* 2000.

[17] J.-P. Hubaux, T. Gross, J.-Y. LeBoudec, and M. Vetterli, "Towards Self-Organized Mobile Ad Hoc Networks: The Terminodes Project," *IEEE Comm. Magazine,* vol. 39, no. 1, pp. 118-124, Jan. 2001.

[18] M. Jakobsson, J.-P. Hubaux, and L. Buttyán, "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," *Proc. Int'l Conf. Financial Cryptography '03,* Jan. 2003.

[19] M. Onen, A. Shikfa, and R. Molva, "Optimistic Fair Exchange for Secure Forwarding," *Proc. Fourth Ann. Int'l Conf. Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous '07),* 2007.

[20] H. Miranda and L. Rodrigues, "Preventing Selfishness in Open Mobile Ad Hoc Networks," *Proc. Seventh CaberNet Radicals Workshop,* Oct. 2002.

[21] L. Buttyán, L. Dóra, M. Félegyházi, and I. Vajda, "Barter Trade Improves Message Delivery in Opportunistic Networks," *Ad Hoc Networks,* vol. 8, no. 1, pp. 1-14, 2010.

[22] "SUMO-Simulation of Urban Mobility," http://sumo.sourceforge.net/, 2012.

[23] C. Song and Q. Zhang, "Coffee: A Context-free Protocol for Stimulating Data Forwarding in Wireless Ad Hoc Networks," *Proc. IEEE CS Sixth Ann. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks (SECON '09),* 2009.

[24] A. Panagakis, A. Vaios, and I. Stavrakakis, "On the Effects of Cooperation in DTNs," *Proc. Second Int'l Conf. Comm. Systems Software and Middleware (COMSWARE '07),* pp. 1-6, 2007.

[25] G. Resta and P. Santi, "The Effects of Node Cooperation Level on Routing Performance in Delay Tolerant Networks" *Proc. IEEE CS Sixth Ann. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks (SECON '09),* 2009.

[26] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," *Proc. ACM SIGCOMM Workshops,* Aug. 2005.

[27] M. Grossglauser and D. Tse, "Mobility Increases the Capacity of Ad Hoc Wireless Networks," *IEEE/ACM Trans. Networking,* vol. 10, no. 4, pp. 477-486, Aug. 2002.

[28] Q. Li, S. Zhu, and G. Cao, "Routing in Socially Selfish Delay Tolerant Networks," *Proc. IEEE INFOCOM '10,* 2010.

[29] P. Hui, K. Xu, V. Li, J. Crowcroft, V. Latora, and P. Lio, "Selfishness, Altruism and Message Spreading in Mobile Social Networks," *Proc. First IEEE Int'l Workshop Network Science for Comm. Networks (NetSciCom '09),* Apr. 2009.

[30] D. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Trans. Comm.,* vol. 39, no. 10, pp. 1482-1493, Oct. 1991.

[31] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in Distributed Systems: Theory and Practice," *ACM Trans. Computers Systems,* vol. 10, pp. 265-310, 1992.

[32] E. Frank and M. Buer, "Key Revocation in a Mobile Device," US Patent Number 7860486, 2010.

[33] A. Liu and P. Ning, "Tinyecc: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," *Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN '08),* 2008.

[34] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," *Proc. CRYPTO '84 Advances in Cryptology,* pp. 47-53, 1984.

[35] A. Back, "Hashcash—A Denial of Service Counter-measure," Whitepaper, technical report, 2002.

[36] R. Rivest, A. Shamir, and D. Wagner, "Time-Lock Puzzles and Timed-Release Crypto," technical report, Cambridge, MA, 1996.

[37] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society," *Nature,* vol. 435, no. 7043, pp. 814-818, 2005.

[38] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft, "Opportunistic Content Distribution in an Urban Setting," *Proc. SIGCOMM Workshop Challenged Networks (CHANTS '06),* 2006.

[39] J. Leguay, A. Lindgren, J. Scott, T. Riedman, J. Crowcroft, and P. Hui, "CRAWDAD Trace upmc/content/imote/cambridge (v. 2006-11-17)," Downloaded from http://crawdad.cs.dartmouth.edu/upmc/content/imote/cambridge, 2006.

[40] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD Trace Cambridge/haggle/imote/infocom (v. 2006-01-31)," Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote/infocom, Jan. 2006.

[41] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD Trace Cambridge/haggle/imote (v. 2009-05-29)," Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote, 2009.

Alessandro Mei received the laurea degree in computer science summa cum laude from the University of Pisa, Italy, in 1994. He continued working toward the PhD degree at the Department of Mathematics of the University of Trento, Italy, and as a visiting scholar at the Department of EE-Systems of the University of Southern California during 1998 and part of 1999. He received the PhD degree in mathematics at the University of Trento in 1999. After a postdoctoral position at the University of Trento, in 2001 he joined the faculty of the Computer Science, Department at Sapienza, University of Rome, Italy. His main research interests include computer system security and parallel, distributed, and networked systems. He was presented with the Best Paper Award of the 16th IEEE International Parallel and Distributed Processing Symposium in 2002, the EE-Systems Outstanding Research Paper Award of the University of Southern California for 2000, and the Outstanding Paper Award of the Fifth IEEE/ACM International Conference High Performance Computing in 1998. He is a member of the ACM and the IEEE, a past associate editor of the IEEE Transactions on Computers (2005-2009), and the general chair of IEEE IPDPS 2009, Rome, Italy. During 2010-2011 he was on leave at the CSE Department, University of California at San Diego, supported by a Marie Curie fellowship.

Julinda Stefa received the Laurea degree in computer science, summa cum laude, and the PhD degree in computer science from Sapienza University of Rome, respectively, in July 2006 and February 2010. She is a postdoc at the Computer Science Department of Sapienza University of Rome, Italy. In 2005 she joined Google Zurich for three months as an engineering intern. She was a visiting scholar at the CS Department of UNC-Chapel Hill, North Carolina, from November 2008 to April 2009, and a Research Intern at Microsoft Research, Cambridge, United Kingdom, from January to April in 2011. Her research interests include computer systems and network security, parallel and distributed systems, and analysis and modeling of social mobile wireless networks. She has published in some of the topmost Conferences and journals like IEEE INFOCOM, ACM MobiHoc, IEEE ICDCS, and *IEEE Transaction on Computers*, and is involved as reviewer and in technical program committees of several Conferences and workshops in the field. She is winner of a research grant from Working Capital PNI and offered by Telecom Italia (30 winners out of 2,138).