2010 International Conference on Distributed Computing Systems

Give2Get: Forwarding in Social Mobile Wireless Networks of Selfish Individuals

Alessandro Mei and Julinda Stefa
Department of Computer Science
Sapienza University of Rome, Italy
Email: {mei, stefa}@di.uniroma1.it

***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 Nash equilibria, 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 side-effect to improve performance by reducing the number of message considerably (more than 20%). 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.**

I. Introduction

Nowadays people walk around carrying all sorts of devices such as cellphones, PDAs, laptops, etc. Typically, these devices are able to communicate with each other in short distances by using communication technologies such as blue-tooth. These networks, also known as Pocket Switched Networks (PSN [1], [2]), can be a key technology to provide innovative services to the users without the need of any fixed infrastructure. Pocket Switched Networks fall in the class of the Delay Tolerant Networks (DTNs). In DTNs, messages can multi-hop from source to destination by using the forwarding opportunities given by the contacts between the nodes. These networks are usually disconnected, are characterized by social-based mobility and heterogeneous contact rate. Examples of such networks include people in working places, students in university campuses, and citizens in metropolitan areas.

The problem of designing a forwarding protocol for Pocket Switched Networks has attracted the attention of many researchers. In most cases, the protocols in the literature break down immediately if you assume that all the nodes in the network are selfish. 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. Epidemic Forwarding is often used as a benchmark, since it is easy to see that, by generating as many replicas as possible in the network, it has optimal delay and success rate. Delegation Forwarding, on the other hand, is one of the best protocols in the literature with its excellent trade-off of cost, delay, and success rate.

In this paper, we introduce Give2Get Epidemic Forwarding and Give2Get Delegation Forwarding, which are, to the best of our knowledge, the first protocols for packet forwarding in a social setting such as the Pocket Switched Network, that cope with the social aspects of the network to tolerate selfish behavior. We reach this goal by showing formally that no rational node has any incentive to deviate. In other words, our two protocols are Nash equilibria. In the paper we describe our methodology and the main steps, the mechanisms, and the idea that we have used to build the complete proof.

Lastly, we perform a large set of experiments to check performance of G2G Epidemic Forwarding and G2G Delegation Forwarding. Quite surprisingly, we discover that some of the mechanisms that we introduced 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 ego, and a considerably lower cost in terms of number of replicas (around 20% less).

The paper is organised as follows: Section II reports on the literature in the area, Section III defines our system model, Sections IV and V present G2G Epidemic Forwarding, the dramatic effect of selfish behavior on vanilla Epidemic Forwarding, and the capability of our protocol to detect possible deviations; Sections VI and VII builds upon the mechanisms developed for G2G Epidemic Forwarding to design the much more sophisticated G2G Delegation Forwarding; lastly, Section VIII describe our experiments on the performance of the G2G protocols and Section IX concludes the paper.

II. Related Work

A lot of work has been done in building efficient forwarding protocols for Pocket Switched Networks. Many of these protocols use in clever and sophisticated ways the properties of human mobility [4]–[6], [20], [21]. All of them rely on the altruistic cooperation among nodes, which, in this settings where nodes are independent individuals cannot be given as granted. Therefore, 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

1063-6927/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDCS.2010.36

distributed systems. See, as an important example, the work in [7], [8].

Previous work has been done in studying techniques of selfish mitigation for mobile ad-hoc networks. The solutions can be classified in two main approaches: *reputation based schemes* [9]–[12] and *credit based schemes* [13]–[15]. In the former, nodes collectively detect misbehaving members and propagate declaration of misbehaving throughout the network. Eventually this propagation leads to other nodes avoiding 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 [16], 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.

Recently in [17] the authors introduce a barter-based co-operation system that aims to increase message delivery in opportunistic networks. The authors assume that altruistic static nodes scattered on the network area generate messages down-loadable from interested network members that pass by. 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 from the subset of the messages to which it is interested. Then the nodes start downloading one message per node at time slot, till they move out each other's communication range. The game-theoretical model developed helps the authors prove that the approach foster cooperation among the nodes. They support their findings with extensive simulations done with the restricted random waypoint model RRW model and the Simulation of Urban MObility SUMO [18]. Though it introduces a novel technique of cooperation stimulation, their work is oriented to a gossip-like service within the network, where messages are created from special nodes and have no specific destination. Moreover, the setting is different and the solution does not consider social aspects of pocket switched networks.

In a very recent work [22] the authors build a routing mechanism based on the willingness (declared by each individual) to forward other individuals' messages.

In [19], the authors study for the first time the impact of different distributions of altruism on the throughput and delay of mobile social communication system. 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.

III. The system model

In our system model, every node is selfish. This is a realistic scenario, if people can get the same level of service without consuming 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) by paying less, and publish 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 will 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 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 timeouts, and the time-stamp is used when reporting misbehavior to the other network members.

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. 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 160 bits long), and cheaper and cheaper computation [23], which is shown to be adequate even for sensors. Moreover, in our study we are addressing a network of smart-phones or PDAs, which are not-so-small devices. Modern smart-phones 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. Therefore, we assume that every node has a public key and the corresponding private key. The public key is signed by an authority that is trusted by every node in the system. Anyhow the authority is *never* used actively in the protocols, thus, as far as our protocols are concerned, it may remain off-line all the time.

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.

IV. Give2Get Epidemic Forwarding

In Epidemic Forwarding [3], every contact is used as an opportunity to forward messages. If node A meets node B, and A has a message that B does not have, the message is relayed to node B. Epidemic forwarding 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 of the network 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 would simply drop every message they receive (except those destined to themselves!). In this section, we will show how

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

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

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

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

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

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

to build a version of epidemic forwarding, called Give2Get Epidemic Forwarding, that works in a system in which 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. 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.

G2G Epidemic Forwarding consists of three phases: Message generation, relay, and test. Message generation executes when one node has a message to send to some other node in the system. 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 it is important.

A. G2G Epidemic Forwarding: The relay phase

Once the message is generated, the sender S tries to relay it to the first *two* (at least) nodes it meets. Assume that 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 a 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 Figure 1, where the role of S is described as done by node A step 1). 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 execute 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.

B. G2G Epidemic Forwarding: The test phase

Node B, once it realizes that it is a relay for message m, will follow the same protocol as done by node S—find two other nodes and relay the message to these two nodes by executing the relay phase as shown in Figure 1. By doing so, it can collect two proofs of relay that it will be asked to show, when meeting node S again, during the test phase. If node B is not able either to show the two proofs or to prove to have still the message in its memory, then node A can broadcast a *proof of misbehavior* (PoM) to the whole network that, in turn, will remove node B if node B is not in the position to prove that A is wrong. The proof of misbehavior consists of the proof of relay $\langle \text{POR}, H(m), A, B\rangle_B$, which is signed by node B.

Only when two proofs are collected the message can be discarded from B's memory. After a timeout Δ_1, B can stop looking for relays, and after an additional timeout Δ_2, node B can discard every information regarding the message. Timeout Δ_1 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 hat the delay of G2G Epidemic Forwarding is very close to the delay of Epidemic Forwarding, and so Δ_1 can be chosen as in its original alter ago without affecting the success rate.

The difference $\Delta_2 - \Delta_1$ bounds the time during which S can test B, and indicates how much longer B has to keep memory of the message. Thus, the shorter this difference, the better in terms of memory usage. On the other hand, timeout Δ_2 should be chosen in such a way that, with non-negligible probability, nodes B meets node S again before Δ_2 expires. We have to trade-off detection rate for efficiency. In our setting, here we can use the "good properties" of social networks: If S and B meet, then it is likely that they will meet again in the near future (within Δ_2 in our case). Indeed, it has been shown in previous work [1], [2], [5] that in social mobile network nodes tend to form clusters of members that meet often in time. Our experiments in the following section fully support this claim. Simply by setting $\Delta_2 = 2\Delta_1$ the detection rate is very high (more than 90% of misbehaving nodes are detected). This result implicitly reveals that re-encounters between pairs of nodes happen soon enough with high probability. Note that during the interval (Δ_1, Δ_2) the nodes do not act as relays anymore. According to what happened before time Δ_1 nodes keep trace of the message/PORs required in the test phase. This might be the message itself (no relays or only one relay have been found till Δ_1) or the two PORs (the message was relayed to 2 other nodes before Δ_1). Note that the POR requires just the same overhead of a message signature. Thus, in the worst case, nodes keep a copy of the message for time $\Delta_2 - \Delta_1$ longer than in Epidemic Forwarding. On the other hand, speaking in terms of total replicas of messages generated in the network we have a gain in terms of cost. Indeed, differently from vanilla Epidemic, in G2G Epidemic nodes forward message copies to *at most* two other relays. While this is a design choice to make the protocol a Nash equilibrium,

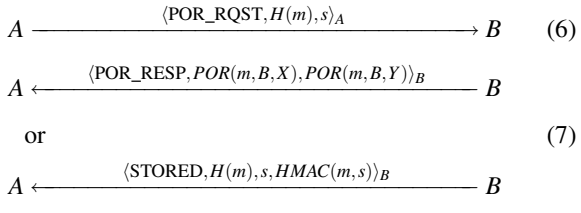


Fig. 2. Protocol of the test phase.

our experiments show that this reduces the number of replicas of some 20%.

The test phase is started by node S (see Figure 2, where, again, the role of S is described as done by node A), when meeting node B, after timeout Δ_1 has expired. During the test phase, node S 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 replay with the two proofs. The challenge is a simple cryptographic protocol in which node S generates a random seed s and asks node B to send a keyed-Message Authentication Code *HMAC* on message m. The particular HMAC used in this protocol should be designed in such a way to be heavy to compute, since we want to incentive node B to relay the message and get the two proofs of relay. Since B does not know the seed beforehand, it must be storing the message unless it has found two relays. Note that it is *not* possible for B to fool S by forging any of the two proofs, since they are signed by the two relays. Note also that the test phase is started only by the source of the message, not by intermediate relays. This is very important to get a Nash equilibrium: only the sender has the interest of checking. As a positive side-effect, the heavy *HMAC* is virtually never executed if no node deviates from the protocol—it is extremely unlikely that the first two relays are not able to find two other nodes that have never seen the message.

C. G2G Epidemic Forwarding is a Nash equilibrium

We formally show that G2G Epidemic Forwarding is a Nash equilibrium by defining a set of *players*, in our case the n nodes in the network, a set of possible *strategies* $\mathscr{S}$, and a *payoff function* $f : \mathscr{S}^n \mapsto \mathbb{R}^n$. When each player i chooses strategy $s_i \in \mathscr{S}$, 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.

In our case, the set $\mathscr{S}$ of possible strategies contains all the possible protocols that the node might implement. Since strategies (that is, protocols) are not limited in length, set $\mathscr{S}$ contains an infinite number of elements. To handle this complexity, we organize our proof 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 $\mathscr{S}$ of strategies into subsets $\mathscr{P}_1, \ldots, \mathscr{P}_{r+1}$, where $\mathscr{P}_j$ contains all the protocols that are equal to π in the first $j-1$ steps and then start to deviate. Clearly, $\mathscr{P}_1 \cup \cdots \cup \mathscr{P}_{r+1}$ is equal to $\mathscr{S}$ and $\mathscr{P}_{r+1} = \{\pi\}$. Fixed player i and step j, if for all strategy profiles $s' \in \mathscr{P}_j$ 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.

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 receive 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 of the strategy profile such that $f_i(s)$ is strictly positive, is decreasing when either the expected value of energy cost or the expected value of memory cost required by the protocol increase, and drops to zero if player i, as a consequence of strategy profile s, has a non-negligible probability, say at least 50%, of not being able to send or receive messages with the same performance of the original protocol.

Now 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 keep it reasonably short by going through the most important steps without hiding critical details. We will consider player i, and show that, if it deviates from the protocol 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.

Assume that $s = (\pi, \ldots, \pi)$, $\pi' \in \mathscr{P}_1$, 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) by moving to protocol π'. Since $\pi' \in \mathscr{P}_1$, player i deviates from step 1 of the relay phase (see Figure 1, where player i has the role of node A). If player i deviates when he is the source of the message, then player i is not able to deliver its own messages and its payoff $f_i(s') = 0 \leq f_i(s)$. It is trickier to understand why $f_i(s') \leq f_(s)$ when player i has the role of node A and it is not the sender of the message. Note that node A does not know the sender of the message, so, there is a non-negligible probability that it is one of the two first relays from the sender and that it will be asked to show two proofs or relay (later, in our experimental results, we will see that the probability of detection is higher than 90%, therefore the probability of being one of the first two relays for at least one message is also more than 90%) or

to show that he still has the message and perform an heavy HMAC and thus consuming more energy, in expectation. If the heavy HMAC is properly chosen in such a way that the energy consumed is higher than the energy saved by storing the message without relaying it, we get $f_i(s') \leq f_i(s)$. 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 none of the two operations, indeed it is discarded from the system and his payoff drops to zero. Note that node A will realize whether it is one of the first two relays only after the timeout Δ_1 has expired. 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, lastly, it does not want to be in the position of performing the heavy *HMAC* in step 7 of the test phase. Therefore, we have shown that, if $\pi' \in \mathscr{P}_1$, then $f_i(s') \leq f_i(s)$. In other words, node A executes step 1 of the protocol truthfully. With the same argument as for step 1, we can easily show that node A executes steps 3 truthfully.

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 node B deviates from the protocol and declines to be a relay, it will never receive any message, since it will discover the destination of the message only at step 5. So, node B will execute truthfully steps 2 and 4.

Lastly, node A will also execute step 5 truthfully. Indeed, the message is extremely 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, in such a way to make $f_i(s')$ drop to zero. Hence, we have got the following result.

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

Let's consider the test phase. As described, only the sender will start the phase, and it will otherwise the other nodes will start dropping its messages. On the other hand, every relay cannot tell whether the previous relay is the sender or not (the sender is encrypted in the message), so they cannot tell whether they will be tested or not. Since nodes do not take the risk of being removed from the network they will get two proofs of relays or keep storing the message. More formally, if node B drops the messages, experiments show that with probability higher than 90% they will be removed from the network and therefore the payoff is zero. 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. Note that the sender won't reveal to be the sender before the timeout Δ_1.

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

To summarize, if a node deviates from G2G Epidemic Forwarding then its payoff is reduced.

Theorem 1: G2G Epidemic Forwarding is a Nash equilibrium.

V. G2G Epidemic Forwarding: Experiments with real traces on detecting deviations

In this section, we report on the results of some experiments with real data, with the goal of understanding what is the impact of selfish behavior in Epidemic Forwarding. Then, we turn to G2G Epidemic Forwarding and test how good is our protocol in detecting possible deviations. In the case of Epidemic Forwarding, we consider *message droppers*—nodes that use the system to send and receive messages and that just drop every message they happen to relay. We will see that message droppers can make the performance of Epidemic Forwarding drop quickly, and we will also see that G2G Epidemic Forwarding detects this kind of deviation right away.

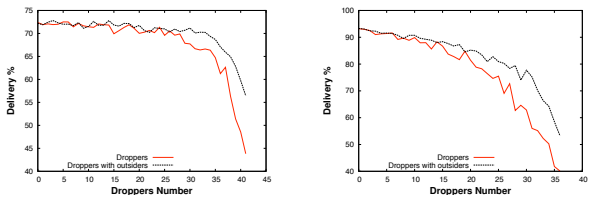
A. Selfishness and selfishness with outsiders

In a social environment, 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 form 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 [24] (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.

B. The data set

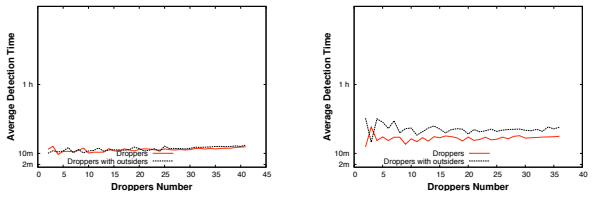
For our experiments we have used two traces collected during experiments done with real devices carried by people. We will refer to these traces as *Infocom 05* and *Cambridge 06*. Characteristics of these data sets such as inter-contact and contact distribution have been observed in several previous works [1], [2], [25].

- In *Cambridge 06* [26] the authors used Intel iMotes to collect the data. The iMotes were distributed to students of the University of Cambridge and were programmed to log contacts of all visible mobile devices. Also, a number of stationary nodes were deployed in various locations around the city of Cambridge UK. The data of the stationary iMotes will not be used in this paper. The number of mobile devices used is 36 (plus 18 stationary devices). This data set covers 11 days.
- In *Infocom 05* [27] the same devices as in *Cambridge* were distributed to students attending the Infocom 2005 student workshop. The number of devices is 41. This experiment covers approximately 3 days.



(a) Infocom05 (b) Cambridge06

Fig. 3. Effect of message droppers on Epidemic Forwarding



(a) Infocom05 (b) Cambridge06

Fig. 4. Dependence of droppers detection time from the number of droppers in G2G Epidemic Forwarding. Detection time is considered after the expiring of TTL value Δ_1 of message.

C. 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 per 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 for Infocom 05 and Cambridge 06 are respectively 30 and 35 minutes.

We first focus on the effect of selfish behavior on Epidemic Forwarding. Figure 3 shows how success rate[1] is affected by the presence of message droppers. In our experiments these nodes drop messages right after the end of the relay phase (see Figure 1). As you can see, the performance of the protocol, under our assumption that every node is selfish, drops to around 50%, 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].

Next, we focus on the detection of message droppers. Anyhow, before showing detection results let us make some considerations on the setting of timeouts Δ_1 and Δ_2. As we already mentioned, Δ_1 is bounded from below by the optimal TTL of the messages, in order to get best delivery rate. Thus, we set it to be 30 minutes in Infocom 05, and 35 minutes in Cambridge 06. From our discussion in the previous section, we set $\Delta_2 = 2\Delta_1$ thus 60 and 70 minutes respectively.

According to our experiments detection probability is 94.7% in the selfish case and 91.3% in the selfish with outsiders case. Moreover, 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 Figure 4). More specifically, in the Infocom 05 scenario the detection takes 12.1 and 13.3 minutes in average, respectively for the vanilla selfishness and for the selfish with outsiders case. Whereas in the Cambridge 06 case it takes 21 and 27.3 minutes in average, respectively for the vanilla selfishness andfor the selfish with outsiders case. All the detection timing is considered after Δ_1 expires. Note that in both scenarios these values are much below the respective $\Delta_2 - \Delta_1$ timeout.

VI. 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 case, node A creates a replica of the message, label 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 prevent message dropping by those who take the message. However, selfish nodes has 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 nobody can do much about this, these nodes would 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

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

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 following the protocol truthfully. In this way, we protect our system against liars, cheaters, and any other possible way to deviate rationally.

In this paper, we consider Delegation Destination Frequency and Delegation Destination Last Contact [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 is respectively the number of encounters with the destination (Destination Frequency), and the time of the last encounter with the destination (Destination Last Contact). 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 messages 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. We will describe only the phases that are substantially different from G2G Epidemic Forwarding. Message generation is just like message generation and in G2G Epidemic Forwarding. Again, the complete proof is long and technical. In the following sections we show the key elements, without hiding important details.

A. G2G Delegation Forwarding: The relay phase and the test by the destination phase

Figure 6 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 8, node A asks B what is its forwarding quality to D'. 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 towards D claimed by node B and the forwarding quality of the message at that point in time.

$$A \xrightarrow{\langle \text{FQ\_RQST}, H(m), D' \rangle_A} B \quad (8)$$

$$A \xleftarrow{\langle \text{FQ\_RESP}, B, D', f_{BD} \rangle_B} B \quad (9)$$

$$A \xrightarrow{\langle \text{RELAY}, H(m), f_m, E_k(m) \rangle_A} B \quad (10)$$

$$A \xleftarrow{\langle \text{POR}, H(m), A, B, D', f_m, f_{BD} \rangle_B} B \quad (11)$$

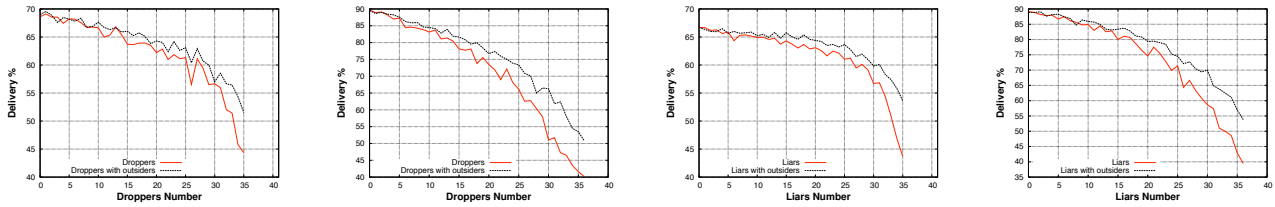
$$A \xrightarrow{\langle \text{KEY}, H(m), k \rangle_A} B \quad (12)$$

Fig. 6. G2G Delegation Forwarding: Protocol of the relay phase.

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 $\langle \text{FQ\_RESP}, B, D, f_{BD} \rangle_B$ for the nodes B that failed to be good relays for the message, that is $f_{BD} < f_m$. As soon as node A finds a good relay, the last two signed qualities of such failed relays are embedded into the message towards D. If the destination D receives the message, it will be able to check if f_{BD} is correct or not (this is the *test by destination* phase). Indeed, f_{BD} should be equal to f_{DB}. Since nodes B and C does not know whether A is the sender or not, they will not lie about their forwarding quality since there is a non-negligible 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 will add B and the PoM of B (the fake f_{BD} value signed by B), to his black list. From that moment on D will be allowed to discard any message that involves B. As in the G2G Epidemic case, D will also broadcast the couple (B, PoM) to the network so that other members of the network exclude B from future message paths generated from or destined to D. Because all network nodes are selfish, they will behave exactly like D when receiving the broadcast message. Thus, B will soon be out of the system.

Note that, in our setting, we don't really need to introduce mechanisms to make this proof checkable by the authority, or by other network members: Node D has no interest in lying. However, simple techniques can be introduced to make it impossible for D to remove faithful nodes. For example, in case of Delegation Destination Last Contact, if the nodes exchange a signed message (with a time-stamp, as usual) at every contact, this message would be a proof of misbehaving against B. Similar techniques can be introduce for Delegation Destination Frequency.

In order to make this mechanism work, the forwarding quality f_{BD} is not the *current* quality, it is the quality computed *in the last completed timeframe*. Every node keeps three versions of the forwarding quality, the current and the two forwarding qualities computed in the previous two completed timeframes. In this way, B and D has a consistent notion of forwarding quality. Of course, the timeframe has to be set in such a way that, with high probability, the message delay falls within one of the last two completed timeframes, so that the destination has the necessary information to detect liars. As a consequence of this set of techniques, no relay will lie about



(a) Droppers in Infocom05 (b) Droppers in Cambridge06 (c) Liars in Infocom05 (d) Liars in Cambridge06

Fig. 5. Effect of message droppers and liars on Delegation Forwarding

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.

B. G2G Delegation Forwarding: The test by the sender phase

The test by the sender is executed only by the sender of the message. Assume that node A is not the sender, and that it has received the message from the sender S. When A gets in contact with S again, after timeout Δ_1 (defined as in G2G Epidemic Forwarding), node A is tested and, just like in G2G Epidemic Forwarding, it gives the two proofs $\langle \text{POR}, H(m), A, B, D, f_m^1, f_{BD} \rangle_B$ and $\langle \text{POR}, H(m), A, C, D, f_m^2, f_{CD} \rangle_B$ to node S. In this way, it is guaranteed that it is not rational to became a message dropper. More than that, this phase is also important to check that A is not a cheater, that is it has not reduced f_m to get rid of the message quickly. Indeed, S can check whether

$$f_{AD} = f_m^1 < f_{BD} = f_m^2 < f_{CD}.$$

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

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.

VII. G2G Delegation Forwarding: Experiments with Real Traces on Detecting Deviations

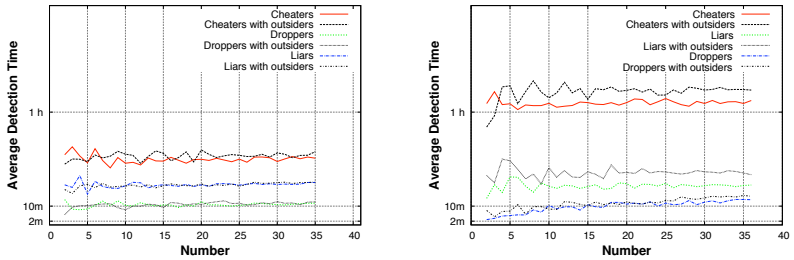
In this section we will consider message droppers, liars, and cheaters. Note that it is not rational to be a cheater in vanilla Delegation Forwarding—if a node labels the message with forwarding quality zero, than it will have to relay it with higher probability, doing more work. Since we are interested only in *rational* deviations, we will see what is the impact of droppers and liars for Delegation Forwarding, and how fast and reliably G2G Delegation Forwarding is able to detect droppers, liars, and cheaters as well. Recall that *no* deviation is rational in G2G Delegation Forwarding—it is a Nash equilibrium—and these experiments on the detection probability are important just to make sure that our assumption that every node has a non-negligible 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 timeout Δ_1 to be 45 minutes in the Infocom 05 case and 75 minutes in the Cambridge 06 case. Again, as in Section V these TTL values are the smallest ones that maximize the performance of the vanilla Delegation protocols. Following the same reasoning as in the G2G Epidemic settings, Δ_2 is set to be $2\Delta_1$ for every scenario. The timeframe after which nodes update the old values of forwarding qualities is set in every experiment to be 34 minutes.

In our experimental setting droppers are again those who drop a message received right after a relay phase, liars are those who report a forwarding quality equal to 0 any time they're asked to, whereas cheaters are those who lower the quality rate within a message to be relayed (in order to get rid of it as soon as possible).

In all our experimental results G2G Delegation Last Contact and G2G Delegation Frequency perform the same. Thus for the sake of space saving here we show only results related to G2G Delegation Last Contact. We run a first set of experiments to see what is the impact of these deviations on Delegation Forwarding. Figure 5 shows 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. Second, we have run a large set of experiments to see how reliably these deviations are detected by the protocols in both traces, Infocom 05 and Cambridge 06. According to our results, droppers and liars are detected with a probability that exceeds 80%, whereas cheaters' detection probability exceeds 60% in both scenarios. The detection rate remains high even in the case of selfishness with outsiders. Table I shows all the detailed results. Note that the detection probabilities are a bit lower for the Cambridge 06 scenario, which is in accordance with it's lower frequent contact rate with respect to Infocom 05. In all cases, this is much more than enough to say that the probability of being detected is not negligible. Recall that, in our model, users have the interest of being part of the system, and that they are not willing to risk

	Infocom 05		Cambridge 06	
	Detection Rate	Avg detection time (minutes)	Detection Rate	Avg detection time (minutes)
Droppers	88%	12	86%	21
Liars	67%	26	65%	52
Cheaters	83%	35	84%	64
Droppers with outsiders	87%	15	84%	23
Liars with outsiders	64%	28	62%	54
Cheaters with outsiders	83%	37	81%	68

TABLE I
PERFORMANCE OF G2G DELEGATION ON THE REAL TRACES. DETECTION TIME IS CONSIDERED AFTER THE EXPIRING OF TTL VALUE Δ_1 OF MESSAGE.



(a) Infocom05 (b) Cambridge06

Fig. 7. Dependence of detection time from the number of selfish individuals in G2G Delegation Forwarding. Detection time is considered after the expiring of TTL value Δ_1 of message

(even with small probability) to be removed from the network.

Then, our question is how fast is the detection. In Figure 7 is plotted the detection time versus the number of selfish members present in the network. As can be seen from the figure, the detection time does not depend on such number. In both scenarios droppers are detected sooner than liars that are detected sooner than cheaters. Moreover, detailed experimental results shown in Table I 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 both scenarios. Again note that the time needed is longer in Cambridge 06, which has less frequent contact rate.

VIII. EXPERIMENTS ON THE PERFORMANCE OF G2G EPIDEMIC AND G2G DELEGATION FORWARDING

In this section we are interested in evaluating the performance of G2G Epidemic Forwarding and G2G Delegation Forwarding compared with their original alter egos. The experimental setting is the same that has been used throughout the whole paper, and described in Section V. We are interested in the following metrics: *memory* (amount of memory overhead of the protocol), *success rate*, *delay*, and *cost* in terms of number of replicas of the same message in the network. We start from memory. Indeed, we can easily check that the memory used by the G2G version of these protocols is within a constant factor from their original counterpart. It is enough to go through the protocols step by step.

Initially, a reasonable goal was to show that adding all the mechanisms and functionalities needed to make the protocols Nash equilibria does not reduce the performance considerably. During the protocols design, we realized that we could hope for more. 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. Figure 8 summarizes a long set of experiments on success rate, delay, and cost. Indeed, looking at the results, something that might seem surprising is happening—G2G Epidemic Forwarding is much better than Epidemic Forwarding in terms of cost, and G2G Delegation Forwarding is considerably better than Delegation Forwarding, again in terms of cost. Note that the experimental setting that we have chosen is considered to be standard in the literature.

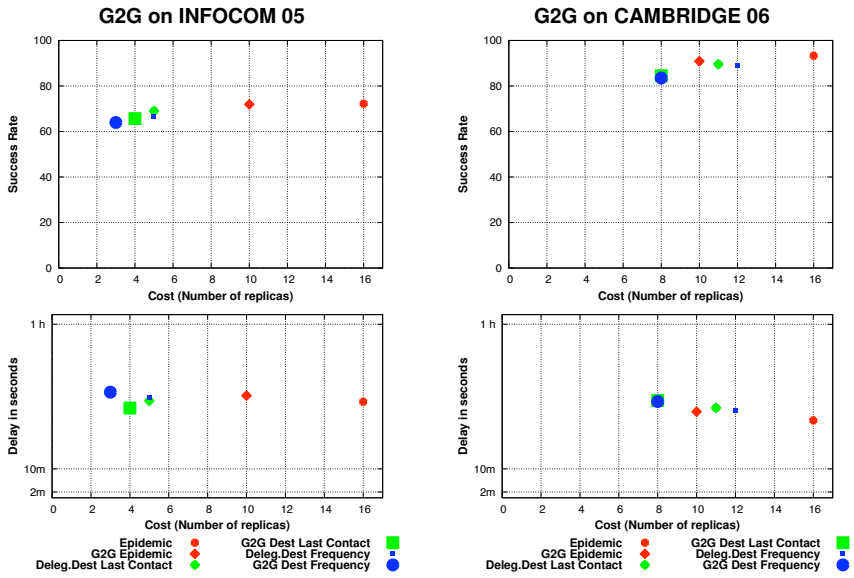
To summarize the results of our experiments, G2G protocols show an excellent performance in terms of cost, even compared with their alter egos that are not Nash equilibrium, decreasing considerably (more than 20%) the number of replicas generated in the system, while their performance in terms of delay and success rate are very close to the original protocols.

IX. 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.

REFERENCES

[1] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference envi-



(a) Infocom05 (b) Cambridge06

Fig. 8. Performance of G2G Epidemic Forwarding and G2G Delegation Forwarding compared with Epidemic Forwarding and Delegation Forwarding

ronments," in *WDTN '05: Proc. of the ACM SIGCOMM workshop on Delay-tolerant networking*. ACM Press, 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," in *INFOCOM '06. Proc. of the 25th IEEE International Conference on Computer Communications*, 2006.

[3] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep. CS-200006, 2000.

[4] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *MobiHoc '08: Proc. of the 9th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2008.

[5] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *MobiHoc '08: Proc. of the 9th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2008.

[6] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *MobiHoc '07: Proc. of the 8th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2007.

[7] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth, "Bar fault tolerance for cooperative services," in *In SOSP05 20th ACM Symposium on Operating Systems Principles*. ACM, 2005.

[8] H. C. Li, , A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "Bar gossip," in *Proc. of the 7th Symposium on Operating System Design and Implementation (OSDI '06)*, 2006.

[9] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *MobiCom '00: Proc. of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000.

[10] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the CONFIDANT protocol: Cooperation Of Nodes Fairness In Dynamic Ad-hoc NeTworks," in *MobiHoc 02: Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing*, June 2002.

[11] K. Balakrishnan, J. Deng, and V. Varshney, "Twoack: preventing selfishness in mobile ad hoc networks," in *Wir. Comm. and Net. Conf., 2005 IEEE*, vol. 4, March 2005.

[12] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*. Kluwer, B.V., 2002.

[13] L. Buttyán and J.-P. Hubaux, "Enforcing service availability in mobile ad-hoc wans," in *MobiHoc '00: Proc. of the 1st ACM international symposium on Mobile ad hoc networking & computing*. IEEE Press, 2000.

[14] J.-P. Hubaux, T. Gross, J.-Y. LeBoudec, and M. Vetterli, "Towards self-organized mobile ad hoc networks: The terminodes project," *IEEE Communications Magazine*, vol. 39, no. 1, 2001.

[15] M. Jakobsson, J.-P. Hubaux, and L. Buttyán, "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," in *Proceedings of Financial Crypto 2003*, January 2003.

[16] H. Miranda and L. Rodrigues, "Preventing selfishness in open mobile ad hoc networks," in *Proc. of the Seventh CaberNet Radicals Workshop*, October 2002.

[17] 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.

[18] "SUMO-Simulation of Urban MObility," http://sumo.sourceforge.net/.

[19] P. Hui, K. Xu, V. Li, J. Crowcroft, V. Latora, and P. Lio, "Selshness, altruism and message spreading in mobile social networks," in *Proc. of First IEEE International Workshop on Network Science For Communication Networks (NetSciCom09)*, April 2009.

[20] F. Li and J. Wu, "LocalCom: a community-based epidemic forwarding scheme in disruption-tolerant networks," in *SECON'09: Proc. of the 6th Annual IEEE communications society conference on Sensor, Mesh and Ad Hoc Comm. and Net.*, 2009.

[21] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: a social network perspective," in *MobiHoc '09: Proc. of the tenth ACM international symposium on Mobile ad hoc networking and computing*, 2009.

[22] Q. Li, S. Zhu, and G. Cao, "Routing in socially selfish delay tolerant networks," in *To appear in INFOCOM 2010*.

[23] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proc. of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track*, 2008.

[24] 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, 2005.

[25] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft, "Opportunistic content distribution in an urban setting," in *CHANTS '06: Proc. of the SIGCOMM workshop on Challenged networks*. ACM, 2006.

[26] J. Leguay, A. Lindgren, J. Scott, T. Riedman, J. Crowcroft, and P. Hui, "CRAWDAD trace upmc/content/imote/cambridge (v. 2006–11–17)."

[27] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD trace cambridge/haggle/imote/infocom (v. 2006–01–31),"