Trust and Reputation Management for Opportunistic Dissemination[☆]

Radu-Ioan Ciobanu[a], Radu-Corneliu Marin[a], Ciprian Dobre[a,*], Valentin Cristea[a]

[a] *Faculty of Automatic Control and Computers*
University Politehnica of Bucharest
313 Splaiul Independentei, Bucharest, Romania

Abstract

Nodes in opportunistic networks need to cooperate to disseminate data. However, employing intermediate nodes for dissemination leads to several security issues. Here, we propose an opportunistic trust and reputation mechanism entitled SAROS, which detects and avoids malicious nodes, i.e. nodes which, upon receiving messages for other interested peers, modify their content in order to spread false information. This can negatively affect the network, by polluting it with spam messages, or dropping messages of interest to the nodes in the network. By detecting and avoiding malicious nodes, SAROS is able to increase the percentage of correct messages that reach their destinations.

Keywords: opportunistic, dissemination, trust, reputation

1. Introduction

Since opportunistic networks (ONs) are totally decentralized, nodes need to cooperate in order to successfully create a publish/subscribe environment that benefits all the nodes in the network. However, employing intermediate nodes for performing dissemination can lead to several issues. In this paper, we

[☆] Supported by national project MobiWay, Project PN-II-PT-PCCA-2013-4-0321.

[*] Corresponding author

Email addresses: radu.ciobanu@cti.pub.ro (Radu-Ioan Ciobanu), radu.marin@cti.pub.ro (Radu-Corneliu Marin), ciprian.dobre@cs.pub.ro (Ciprian Dobre), valentin.cristea@cs.pub.ro (Valentin Cristea)

handle the problem of trust and reputation in opportunistic networks. Namely, we propose a trust and reputation mechanism (entitled SAROS - Socially-Aware Reputation mechanism for Opportunistic diSsemination) which has the purpose of detecting and avoiding malicious nodes. A malicious node is a member of the opportunistic network which, upon receipt of a message to be forwarded to interested nodes, modifies its content in order to spread false information. This can lead to the pollution of the ON with spam messages, and to the loss of messages of interest to the nodes in the network. By detecting and avoiding such malicious nodes, SAROS is able to increase the percentage of correct messages that reach their destinations.

SAROS is implemented as a component of Interest Spaces (presented in more detail in [1] and [2]), which is an interest-based data dissemination framework for opportunistic networks. It is able to disseminate data to interested nodes, by taking advantage of their context information (such as location, interests, social connections, encounter history, etc.). Its advantage is that it offers a unified interface for data dissemination in various situations.

Since opportunistic networks are decentralized and two nodes are only connected upon a contact (i.e. when they are in range), handling trust and reputation is a totally different problem than in regular peer-to-peer networks. There is no central entity that can be used as an authority regarding node reputation, and a node only has information that it has gathered itself, or that it has received from encountered nodes. However, a node cannot know how much to trust the information received from an encountered node, which may very well feed it false data. For example, a malicious node can report other malicious nodes that it collaborates with as trustworthy nodes. Thus, a node can easily form a wrong impression (i.e. it can think that a malicious node is trustworthy) by believing the information provided by malicious nodes, especially if it encounters them often.

However, this is where the advantage of opportunistic networks is shown. Since opportunistic networks are formed of mobile devices, mobility can also act as a benefit. For example, it is much more difficult for a malicious node to follow a “victim” node around to feed it false information than it is for nodes in a regular peer-to-peer network to flood the victim with false information whenever they want. Moreover, opportunistic nodes are generally mobile devices such as smartphones that belong to humans, so interactions are governed by social connections and user mobility. This means that a large part of the encountered nodes are familiar, and most of them are even connected through an online social network. Since connecting on such a social network requires that the nodes know (and implicitly trust, to an extent) each other in real-life, this information can be used to assign pre-set trust values to connected nodes.

In order to have an informed view of the entire network (or as much of it as possible), nodes have to collect data about the behavior of relay nodes, as well as regarding the opinions of other nodes about them. Since an opportunistic node cannot know on the spot whether another node it has relayed its message to actually delivers it to the intended destinations, a different means of confirmation should be employed. Other solutions employ feedback messages that are spread in the network upon a successful delivery. However, such messages tend to flood the network, so a better means might be to employ gossiping. Thus, when a successful delivery occurs, a node can increase its opinion regarding the relay node (or nodes), and gossip this information to other nodes in the network. This way, information is spread through the entire network, and more and more nodes get to see the bigger picture.

Another problem regarding opportunistic networks is that nodes do not have a direct method of deciding whether a received message has not changed during its life in the network (since it may pass through many hops until its destination, some of which can be malicious). Means of encrypting messages have been proposed [3], but they are based on nodes establishing a key in an offline manner, prior to getting on the network, which is not always feasible. Nodes can sign messages with a certificate, but the problem is that there is no central trusted entity that is able to generate and confirm the authenticity of these certificates.

We consider that SAROS is the first opportunistic trust solution that has detection mechanisms for messages that have been tampered with, while also using social information to pre-establish trust. Other social-based solutions simply use social relationships to decide whether a node is trustworthy, without actually analyzing the content carried by that node, while also not giving a chance for delivery to nodes that are not socially connected (which can lead to missing some valid data exchange opportunities). SAROS takes advantage of the inherent design of opportunistic networks to receive a message on multiple paths and decide its correctness.

The rest of this paper is structured as follows. Section 2 presents the state of the art in the area of trust and reputation management for mobile networks. Then, in Section 3 we present an overview of the Interest Spaces framework, discussing its architecture and the composing layers. In Section 4, we propose and present SAROS, and in Section 5 we perform an experimental evaluation of the proposed solution for multiple scenarios. Finally, in Section 6, we present our conclusions.

2. State of the Art

A very thorough survey of security and trust management in ONs is presented in [3]. Among a multitude of security and privacy-related issues, the authors also tackle the problem of managing trust in opportunistic networks, in terms of having confidence that a node that is relayed a message will successfully deliver it towards the intended destination. The authors present existing trust solutions for mobile networks, and split them into several categories, depending on the type of trust establishment: reputation-based trust [4, 5], social trust [6, 7], environmental trust [8], and data-centric trust [9, 10].

EigenTrust [11] is a trust and reputation system where nodes compute and use global trust values for choosing the peers they download data from, in order to avoid and isolate malicious nodes. EigenTrust nodes compute local trust values for their peers based on their transaction history. In order to decide if a node is to be trusted, EigenTrust computes a global trust value for that node, obtained from the local trust values assigned by other peers to that node, weighted by the global reputations of the assigning peers. The authors also introduce the notion of pre-trusted peers, which are the first nodes queried regarding peer reputations, and they are also automatically trusted.

Similarly to EigenTrust, PowerTrust [12] is a reputation system for P2P networks that builds a trust overlay network to model the trust relationship between peers, and is based on the observation that user feedback can be approximated by a power-law distribution. It dynamically selects a small number of power nodes (which have the highest reputations) by using a distributed ranking mechanism. These power nodes are chosen dynamically and constantly updated, so PowerTrust is more robust than EigenTrust towards popular nodes leaving the network or getting infected. Our solution uses some ideas from EigenTrust and PowerTrust. Namely, nodes compute local trust values based on their own experiences with other nodes, but also use gossiping to find out other peers' opinions of interacting nodes, which are used to compute global trust values for a more informed decision. It also uses pre-trusted peers, which are different from node to node, being chosen from the node's social connections.

In [13], the authors propose an ontology-based trust model, where the network nodes' behavior is analyzed based on direct and indirect reputation. It takes advantage of a reputation system to support the decisions that must be made by users when they have to choose whether or not to trust another user during opportunistic encounters. In this situation, trust refers to whether the encountered node is willing to help the requester node. A node A's direct trust in another node B is thus computed based on A's own experiences with B, in terms of information retrieval (i.e. general information users usually share in a communication environment, such as the contact list, files, a task list, the location, etc.) and connection service (i.e. granting someone Internet connection via one of the device network interfaces). Based on the collected data, A performs an average (simple or exponential) over the last experiences with B. On the other hand, A's trust in B is also computed using indirect information obtained from other peers in the network, where other nodes' opinions about B are weighted using A's opinion of said nodes. The Interest Spaces trust component behaves in a similar manner, since a node's global trust is computed not only based on the node's local information, but also based on information collected from other encountered nodes, helping to create a bigger picture.

RADON [4] is a reputation-assisted data forwarding solution for ONs that is based on the notion of positive feedback messages (PFMs). These are special confirmation messages that help the reputation mechanism monitor the behavior of a forwarder. They are used by nodes in the network to assess the reputation of other nodes. RADON evaluates whether an encountered node is a qualified forwarder not only in terms of the probability of running into a destination, but also the encountered node's real forwarding reputation. The algorithm uses both first-hand information (i.e. data obtained by collecting PFMs from the network and analyzing their content), but also second-hand information obtained from other nodes in the network (not only from the current neighbors, but also from previous encounters). Using this information, as well as analyzing the number of times a relay node has encountered a message's destination in the past (the two values being weighted equally), the current node decides whether to forward the message to the encountered peer.

A social-based trust solution is also proposed in [8] and [7], where the social network (with its pre-established friends), its structure, and its dynamics are used to create a subset of trusted nodes in the network. Moreover, nodes that are frequently co-located (the so-called familiars), as well as nodes with common tastes, are employed as the basis of the trust algorithm. This is based on the assumption that, by becoming contacts on a social network such as Facebook or LinkedIn, nodes have prior knowledge of each other. This acts as a contract of understanding between the nodes, and can also be used for logging in the system and obtaining a unique ID.

Social Trust [6] is a trust method that leverages social information to establish trustworthy communication for mobile opportunistic networks. Nodes' trust is social-based, since it is argued that they belong to an opportunistic network composed of people's devices (such as smartphones). Thus, socially connected nodes have an intrinsic trust in each other, since they are likely to interact more often in good conditions. The authors propose two major techniques of establishing trust: Relay-to-Relay and Source-to-Relay. When using the former method, a node that is carrying a message computes the trust in an encountered peer based on the relationship between the two nodes, while the latter method assumes that candidate relays are analyzed based on their relationship with the message's source. For each of the two trust methods, four ways of computing a node's trust are proposed: common interests, common friends, social graph distance, and a combination between common friends and social distance. We have chosen Social Trust for comparison with our solution, because it behaves the best out of all the solutions proposed here.

As can be seen, there are few solutions that attempt to perform what SAROS does, namely detect malicious nodes that tamper with messages in the network. Other solutions focus on establishing trust that a node can deliver a data item to its destination, but we would argue that this is actually the entire point of opportunistic networks (where data routing and dissemination is probabilistic, and nodes have to "trust" that a relayer can take a message closer to interested peers). However, this is not the focus of this paper. Instead, we wish to propose a solution that attempts to increase the probability that the data that reaches a destination node is correct and comes from a trusted source.

3. Interest Spaces

Interest Spaces (presented in more detail in [1] and [2]) is a framework for publish/subscribe-like data dissemination in opportunistic networks. On one hand, nodes are able to generate data (thus being publishers) marked with various tags. On the other hand, nodes can subscribe to topics by specifying that they are interested in certain tags. Since we are dealing with mobile networks where there is no central entity to keep track of subscriptions, nodes are only aware of information about their interests, and those of the nodes they encounter. Thus, solutions that exist in centralized networks cannot be applied in this situation, which is why we have proposed Interest Spaces.

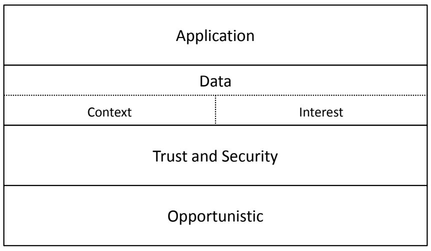


Figure 1: Interest Spaces architecture.

The main purpose of Interest Spaces is to provide nodes in an opportunistic network with information from sources they are subscribed to as quickly and as efficiently as possible. It is a multi-purpose framework that allows nodes to define an interest area and thus keep data objects marked with the corresponding tags (i.e. that the nodes are interested in) close to where they are needed.

There are many real-life situations where a framework such as Interest Spaces would be useful, because it simplifies dissemination by just allowing applications to mark data items with certain tags, letting the framework handle the caching, routing, forwarding, and disseminating. Similarly, applications that need to subscribe to topics simply have to specify the tags they are interested in, and the framework does the rest.

There are four layers in the Interest Spaces framework, as shown in Figure 1. On the highest level there is the application layer, which provides the API for all applications using our framework. It provides the basic calls for publishing a data item marked with at least one tag (interest) and for subscribing to topics.

The second layer of Interest Spaces is the data layer and has two components: interest and context. The interest component handles data publishing and subscriptions based on interests, as a response to application requests. At this level, interests are represented internally, a node's data memory is organized based on interests, and data items are packed with the necessary information. The context component is responsible for collecting context information, both about the current node (such as social connections, location, etc.), as well as about encountered nodes (encounter history, interest history, etc.). The con-

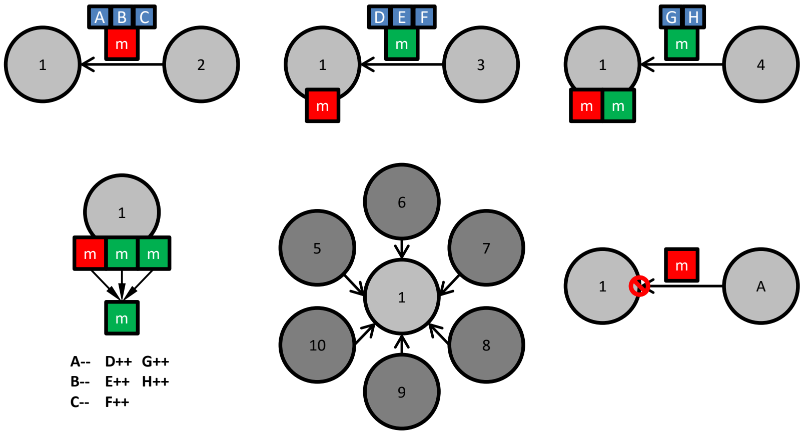


Figure 2: SAROS behavior.

text is used in performing caching and forwarding decisions: nodes decide if they are suitable for carrying data marked with a certain tag based on context information collected opportunistically through mobility and node encounters.

The next layer of the Interest Spaces architecture handles trust and security. This is where SAROS, which is presented in Section 4, is located. At the bottom level of the architecture, there is the opportunistic layer. It handles the communication between nodes, based on the decisions taken at the data and trust and security layers.

4. SAROS

In this section, we propose and present SAROS (Socially-Aware Reputation mechanism for Opportunistic diSsemination), a social-aware reputation mechanism for ONs, which is part of the Interest Spaces framework. We show its basic functionality, the way it handles trust representation, information gossiping, and correct messages selection.

4.1. Functionality

Figure 2 presents the behavior of an opportunistic node employing SAROS for trust and reputation. It is based on local and global trust values, gossiping, and quorum selection to decide which version of a message is correct, as will be described in the next subsections. For now, let us assume that node 1 receives a message m from node 2, which has passed through nodes A, B, and C. Then, the node also receives a different version of the same message, with D, E, and F on the path. Finally, another copy of message m is received, the same version as before, with G and H on the path. Since node 1 now has enough copies of m to decide upon a correct version, it performs the following steps:

1. selects which message version is the correct one (in this case, the one received from nodes 3 and 4, since it represents a majority)
2. sends the correct message to the application layer
3. decreases the trust in the nodes from the path of the wrong message version (A, B, C)
4. increases the trust for the nodes from the correct paths (D, E, F, G, H).

Node 1 can also meet other nodes, which gossip their own trust information, as well as other nodes' trust values. Finally, when node A, which has a bad reputation, attempts to send a message to node 1, it is denied. Furthermore, node 1 will not relay any of its messages to A. All the components of the SAROS solution are presented in details in the following subsections.

4.2. Trust Representation

The representation of trust in SAROS is similar to the idea used by EigenTrust [11]. Namely, a node computes two types of trust values towards each peer: a **local trust**, which is based on the node's own interactions with other participants in the network, and a **global trust**, computed using second-hand information gossiped by other nodes.

Each node stores transaction data for each peer it has interacted with in a separate structure. This structure contains the number of successful and unsuccessful transactions, based on which the node's local trust is computed. The formula for computing the **local trust** s_{ij} of the current node i in node j is as follows:

$$s_{ij} = \frac{suc(i,j)}{suc(i,j) + unsuc(i,j)}$$

Thus, s_{ij} is the percentage of successful transactions out of the total transactions between the two nodes. Note that a transaction between node j and the current node i means that j has been on the path between a message's destination and node i. If there have been no transactions between i and j, then we use an idea similar to the one proposed in the EigenTrust paper. Namely, we assume that there are pre-trusted nodes in the network, which a node automatically puts its trust in unconditionally. Since we are talking about a fluid ever-changing network, which does not necessarily have a beginning and an end, we propose using the connected nodes from the online social networks as the pre-trusted nodes.

If there has been no transaction between nodes i and j and they are not socially connected through any online social networks, the local trust is computed as the minimum between the number of common friends of i and j (normalized with a pre-set threshold), and 1. Thus, not only directly-connected nodes are pre-trusted, but also indirect connections, up to a certain point. In conclusion, the **local trust** s_{ij} of the current node i in node j is computed as follows:

$$s_{ij} = \begin{cases} \frac{suc(i,j)}{suc(i,j)+unsuc(i,j)} & \text{if } suc(i,j) + unsuc(i,j) > 0 \\ 1 & \text{if } i \text{ and } j \text{ are socially connected} \\ min(1, \frac{friends(i,j)}{MAX\_COMMON\_FRIENDS}) & \text{otherwise} \end{cases}$$

However, if a node only computes other nodes' reputations based on its local trust, it may have a skewed view of the network, or incomplete information that can lead to wrong decisions. For example, if node i is not socially connected to node j and they have no common connections, i will just assume that j is untrustworthy and will not deliver any messages to it, even if j is well-intended and can successfully help with dissemination. For this reason, as previously stated, nodes compute reputations using local trust values from all the other nodes in the network (or at least the ones which the current node has received information from).

The easiest way to compute a global reputation would be to average the local trust values of each node, and compute a final value for the node which is analyzed. However, malicious nodes working in collaboration might simply lie about each other's reputation (setting it to the maximum value), which will lead to victim nodes believing that malicious nodes are trustworthy. For this reason, we use a method of computing the global reputation similar to the one employed by EigenTrust [11], by computing a weighted average based on the local trust of the current node in each of the other nodes participating in the computation. Thus, the **global trust** value t_{ik} of a node i in a node k is computed as follows:

$$t_{ik} = \frac{\sum_j s_{ij} s_{jk}}{\sum_j s_{ij}}$$

If node i has not received any information about the trust of a node j regarding node k, then that element is 0 and the local trust in node j is not added to the denominator sum.

The final value of t_{ik} will be a rational number between 0 and 1 which will act as a probability of trust. Therefore, when a node i meets a potential relay j, it will send it messages for delivery and accept its messages with probability t_{ik}. This way, if j is untrustworthy, i's messages will not be delivered to it, so it will not have a chance to modify them and spread them further. Moreover, node i will not help j with its own messages, because they have a very high chance of being spam messages.

4.3. Gossiping

As previously stated, other nodes' local trust values are used in computing the global trust value of a node. In order for this to happen, this information must reach all interested nodes. Since we are dealing with opportunistic networks where nodes can only communicate directly upon a contact, SAROS uses gossiping for exchanging trust data between nodes.

Thus, whenever two nodes meet, they not only exchange their own local trust values, but also the trust values previously received from other nodes. Thus, assuming that a node i contains no local trust information other than its own, and it encounters a node j that also knows the local trust value of node k, i will end up knowing the local trust values of both j and k. Each of these received local trust values has a timestamp, and a node only stores the most recent value. Thus, if i contains a newer value for k's trust than the one j has, both i and j will end up with i's value.

Moreover, in order to avoid malicious nodes spreading false local trust values in the network (in order to bump their own reputation), a node i will update the local trust value of k from j only if i's local trust in j is higher than the local trust in the node that originally gossiped k's local trust value.

4.4. Deciding the Correctness of a Message

Since we have previously established that encryption or the use of certificates are not feasible for use in opportunistic networks, a different method for verifying the correctness of a message must be used by SAROS. We propose employing a quorum-based method, which is suitable for ONs since messages are spread in the network and thus reach a destination multiple times on different paths.

In SAROS, when an interested node receives a message, it does not send it to the application layer directly. Instead, it stores it until several more copies of the same message arrive (assuming messages have unique IDs). When a pre-set number of copies of a message arrive, a quorum algorithm is used to decide which version is the correct one. Thus, each version is counted, and if the most popular version's presence is higher than a pre-set threshold, then it is considered as the correct one and the message is sent to the application layer. For version comparison, each message is hashed when it is delivered to the destination node, and the hashes are compared. This way, differing versions of the same message can be detected, and a malicious node would not be able to influence this operation (i.e. it cannot modify the hash, since it is done by the node receiving the message).

SAROS' use of a quorum method takes advantage of the mobility of nodes in an ON. If a malicious node modifies a message it receives, and forwards it, then a single modified version of the message is sent further upon contact with a node. It is true that this version might be spread in turn by non-suspecting nodes, but so is the original (and correct) version of the message. Moreover, the correct version starts to spread earlier, so it has a higher chance of getting to more nodes than the modified version. Furthermore, the employed dissemination algorithm also attempts to send a message only to nodes that have a high chance of reaching intended destinations, and there is a high chance that the malicious node is not among them. We must also not forget that, once a node's reputation decreases, others will start refusing to carry its messages, so the chance of spreading a modified message increases even further.

A possibility for spreading a false message easier (and to more nodes) is for the malicious nodes to work in cooperation. This means that, when they receive a certain message, they must agree to modify it in the same way. However, given that nodes in ONs cannot communicate unless they are in range, this might prove complicated to implement in real-life.

4.5. Increasing Trust

When a node decides which version of a message is the correct one, it increases the trust of all the nodes on the correct paths, and decreases it for the nodes on the incorrect paths. The main drawback of this approach is that non-malicious nodes might have relayed a modified version of the message without knowing, and will thus get their reputation value decreased without doing anything wrong. However, assuming that the reputation algorithm is good enough, nodes will know not to trust a malicious node, so there is a small likelihood of this happening. As a potential improvement, nodes from all the paths can be analyzed, and only nodes that appear on the incorrect paths should have their reputation decreased. However, malicious nodes can behave badly occasionally, so they can also appear on both correct and incorrect paths, which would make a receiving node increase their reputation. We have chosen the first version, because we believe it is better to receive fewer but more correct messages, as opposed to more but potentially incorrect ones.

This means that, when one of the non-malicious nodes on the path receives a message from one of the malicious nodes, it should already know that the malicious node has a bad reputation, and not forward its messages any further. This will happen after a time, when the algorithm is able to balance itself. It is interesting to analyze how long it takes for the algorithm to balance itself, and how this affects the overall behavior, and we wish to do this as future work.

5. Experimental Results

In this section, we present the performance obtained by SAROS when compared to other similar solutions for multiple scenarios.

5.1. Experimental Setup

We tested SAROS using MobEmu[1] [14], an opportunistic network emulator that is able to replay a mobility trace and apply a desired routing or dissemination algorithm when two nodes meet. We ran SAROS on two traces, Sigcomm 2009 [15] and UPB 2012 [16]. The former was collected using an opportunistic mobile social application entitled MobiClique. The tracing experiment lasted for three days and gathered data from 76 smartphones running MobiClique, which were given to the participants of the Sigcomm 2009 conference in Barcelona. The latter trace was collected using an Android application for a period of 64 days in an academic environment at the University Politehnica of Bucharest, with the participants being 24 students, assistants, and teachers from the faculty. Contact information was obtained through Bluetooth discovery messages and WiFi peer-to-peer interactions. We chose real-life traces instead of synthetic models because they offer a more realistic behavior, and because the traces we used also contain information about the social connections between the nodes, as well as their interests.

[1] `https://github.com/raduciobanu/mobemu`

In all the experiments presented here, data (in the shape of messages) is tagged with topics that nodes are able to subscribe to. When a node is subscribed to a topic, it is interested in any data tagged accordingly that it has not received yet. Every node interested in a certain topic can generate information tagged with it, but not with other topics. Each node that has at least one interest generates 30 messages per day. A node interested in multiple topics is able to generate data for each of them, by choosing randomly. We chose to analyze the hit rate (percentage of messages that reach interested nodes) corroborated with the correctness (percentage of correct messages from the ones that were delivered) for each tested scenario. Thus, we show the correct messages hit rate for every scenario, computed as the hit rate of correct messages reaching their destinations, by multiplying the hit rate with the correctness.

As the data dissemination algorithm over which SAROS is applied, we used Limited Epidemic, which is a limited-memory version of Epidemic [17]. It behaves exactly like the original implementation, except that, when the data memory is full and a new message should be downloaded, the oldest message is deleted from memory. Unless otherwise specified, we tested with data memory sizes of 500, 4500, and 10000 messages. If we assume that a message has 1 MB, then we tested for nodes capable of storing 500 MiB, 4.5 GiB, and 10 GiB worth of messages, which are plausible values given the capabilities of the latest smartphones available on the market.

In order to better highlight the benefits of SAROS, we elected to compare it to an existing trust and reputation solution for opportunistic networks, namely Social Trust [6], since we believe it is the most optimal from all the solutions we previously presented. As shown in Section 2, there are eight possible implementations for Social Trust. For either the relay-to-relay approach or the source-to-relay approach, there are four filters for analyzing the connection between two nodes: social distance, common interests, common friends, and a combination between the social distance and common friends filters. As the Social Trust authors show in their paper, the common friends-based filter outperforms all the other approaches, achieving one of the best cost/success rate trade-offs. For this reason, this is the method that we compare SAROS to. Since the optimum number of common friends depends on the behavior of the network, we compare SAROS to Social Trust with the common friends filter set to values between 2 and 5.

We previously stated that, to be more efficient, malicious nodes in an opportunistic network can not behave badly at all times. Instead, they can sometimes deliver messages correctly, without modifying them, in order to trick other nodes into thinking they are not malicious. Thus, the MobEmu implementation of malicious behavior allows nodes to act badly only a given percentage of the time. We have a test case where we vary this percentage and analyze the results, but unless otherwise specified, the malicious nodes act badly all the time. The default value for the required number of messages for the quorum algorithm is 3, and the default percentage of correct messages is 50% (i.e. a node expects to receive three versions of a message from separate paths, and if two of them are the same, then that is considered the correct version). Also, the $MAX\_COMMON\_FRIENDS$ constant is set to 1, so if a node i has not met a node j and they are not socially-connected, it will compute j's local trust as 1 if the two nodes have at least one common friend, and 0 otherwise. We assume that, upon a contact, nodes can exchange as many messages as they wish. When testing with malicious nodes in the network, unless specified otherwise, these nodes are chosen starting from the ones with the lowest number of social connections, since these nodes are strangers to the others and thus have a higher chance of wanting to hurt them.

5.2. Results

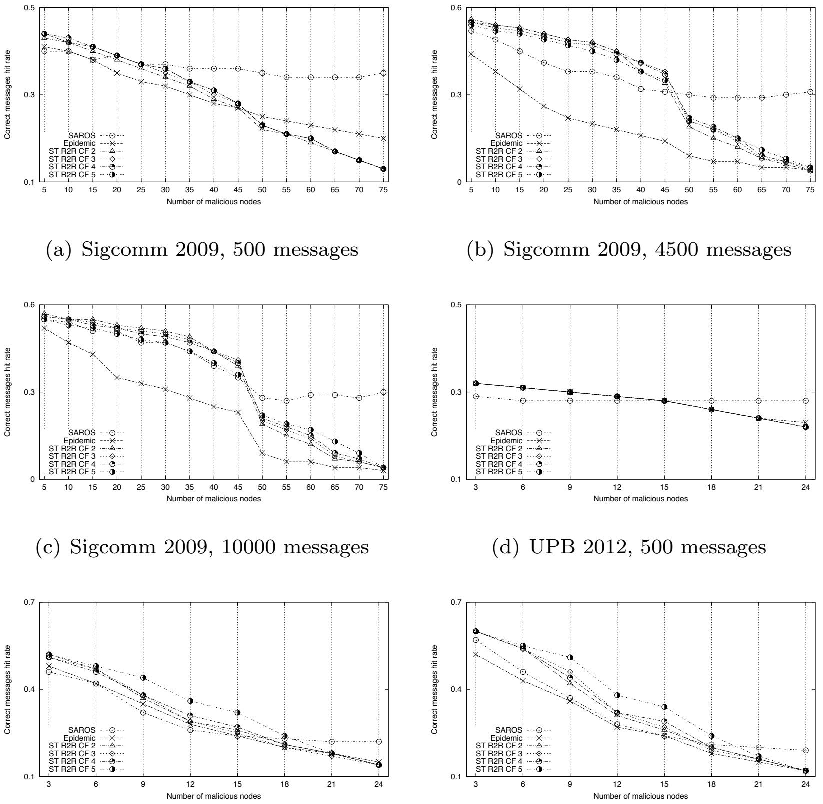
In this subsection, we present multiple scenarios that highlight the benefits of employing a trust and reputation algorithm (and SAROS, in particular) in opportunistic dissemination. We analyze the results in detail and draw conclusions regarding the benefits of SAROS.

5.2.1. Impact of Malicious Nodes on Opportunistic Dissemination

For the first scenario, we want to analyze the impact that non-connected malicious nodes and their number have in an ON. By non-connected we understand that, when a message is relayed to a malicious node, it modifies it randomly, in a different manner than another malicious node. This means that, if two malicious nodes modify the same message, there is a very low chance of them modifying it in the same way (in MobEmu, we perform a random between 0 and the maximum integer value). A test scenario which will be presented later handles the case when malicious nodes act together.

For this scenario, we vary the number of malicious nodes in the network (in increments of five for Sigcomm 2009, from 5 to 75, and in increments of three for UPB 2012, from 3 to 24) and analyze the network's performance in terms of the metrics previously described.

Figure 3 shows the correct message hit rates obtained by the six analyzed algorithms (SAROS, Epidemic, and Social Trust with 2, 3, 4, or 5 common friends) on Sigcomm 2009 and UPB 2012, for data memories of 500, 4500, or 10000 messages, when increasing the number of malicious nodes in the network. Figures 3(a), 3(b), and 3(c) show the results for the Sigcomm 2009 trace. For Epidemic and Social Trust, the hit rate is constant regardless of the amount of malicious nodes in the network, because these algorithms do not look at a message upon its arrival at an interested node. Thus, even if a message that might be correct arrives, it is sent to the application layer and considered delivered. However, these results must be corroborated with the correctness of each algorithm, which shows that, while SAROS' hit rate drops with the increase in the number of malicious nodes, its correctness drops a lot less, especially when comparing to the other solutions. For a small data memory (i.e. 500, in this case), SAROS outperforms Social Trust even for few malicious nodes. It is also important to note that, for this trace, the number of common friends does not affect the Social Trust algorithm too much. A problem with Social Trust in this situation with a small data memory is that it ends up behaving even worse



(a) Sigcomm 2009, 500 messages

(b) Sigcomm 2009, 4500 messages

(c) Sigcomm 2009, 10000 messages

(d) UPB 2012, 500 messages

(e) UPB 2012, 4500 messages

(f) UPB 2012, 10000 messages

Figure 3: Correct messages hit rates when malicious nodes are present.

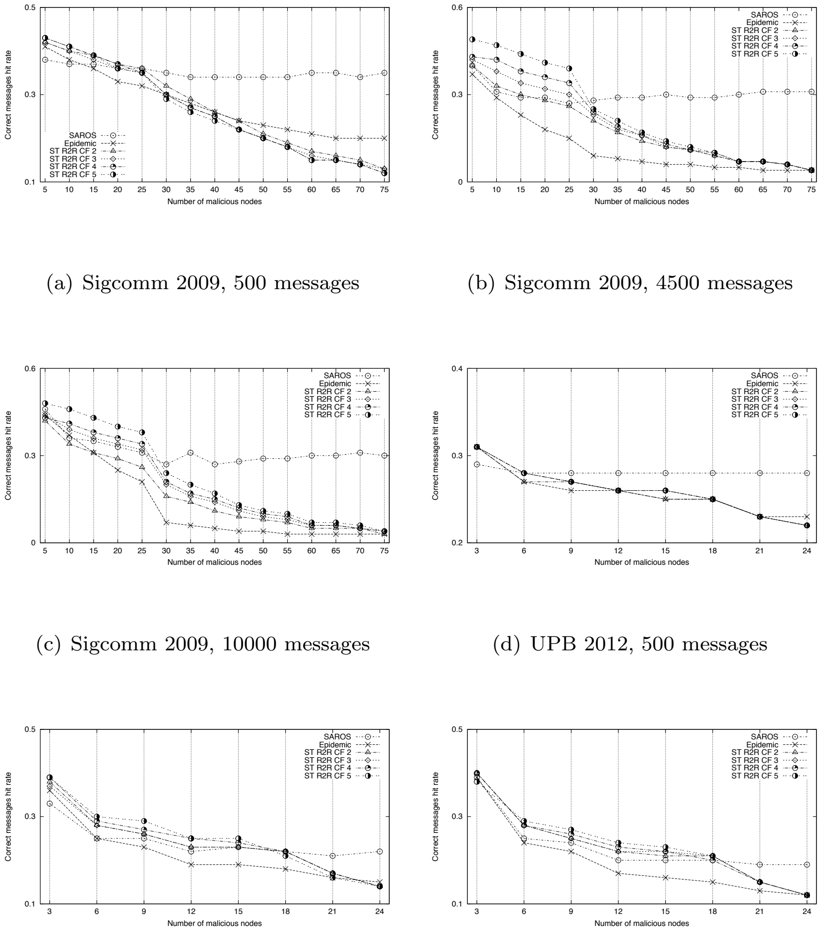
than the case where no trust and reputation methods are used, which makes it unfeasible in networks with many malicious nodes. However, Social Trust does behave better than SAROS when there is a small number of malicious nodes in the network (lower than 45 out of 76 for this trace), for higher data memories. However, the differences in correct messages hit rates are relatively small, no higher than 10%.

These results show that, if a network is known to have many fully malicious nodes, SAROS would be a better solution than Social Trust because it can scale better, while still keeping good results for fewer malicious nodes. In some situations, it is also important that as many correct messages as possible reach their intended targets, even if the total hit rate is lower. In these situations, SAROS would also be a better fit than Social Trust, especially because its correctness rate never goes below 60%, even when almost all the nodes in the network are malicious. It should also be noted that Social Trust is an algorithm that takes advantage of social information, whereas, in the scenarios presented here, SAROS is applied over Epidemic directly. We believe that, if SAROS were applied to Interest Spaces, the results would be better.

We also show the results for this scenario on the UPB 2012 trace in Figures 3(d), 3(e), and 3(f), and it can be seen that, for a trace where nodes are highly connected (the participants being students at the same faculty, more often than not from the same group), a socially-aware solution such as Social Trust outperforms an Epidemic-based implementation of SAROS. However, the results shown for the Sigcomm 2009 trace still hold, namely that SAROS behaves well for scenarios with a high number of malicious nodes. Furthermore, the correctness of SAROS does not drop as badly as that of the Social Trust solution.

5.2.2. Infecting Popular Nodes

In the previous scenario, the malicious nodes were selected starting from the ones with the lowest number of social connections, since those were the strangers that could potentially be malicious, especially under the assumption that we know the people we are connected to in online social networks. However, certain nodes might be infected, in which case they start behaving badly unbeknownst to the node's owner (i.e. the carrier of the infected device). In such situations, algorithms that rely on social connections, such as Social Trust, might be affected. The same is true for SAROS, which uses social connections as pre-trusted nodes. The advantage of SAROS in this situation would be that it only uses the pre-trusted nodes to bootstrap the system. When the quorum algorithm kicks in and starts analyzing messages, the trust values should converge towards the correct ones.



(a) Sigcomm 2009, 500 messages

(b) Sigcomm 2009, 4500 messages

(c) Sigcomm 2009, 10000 messages

(d) UPB 2012, 500 messages

(e) UPB 2012, 4500 messages

(f) UPB 2012, 10000 messages

Figure 4: Correct messages hit rates when popular nodes are infected.

Thus, the only difference between the first scenario and this one is that, in this situation, malicious nodes are chosen from the nodes with the most social connections. Figure 4 shows that, for this situation, SAROS clearly outperforms Social Trust, even for fewer malicious nodes on high data memory situations. Only Social Trust with 5 common friends obtains a better correctness than

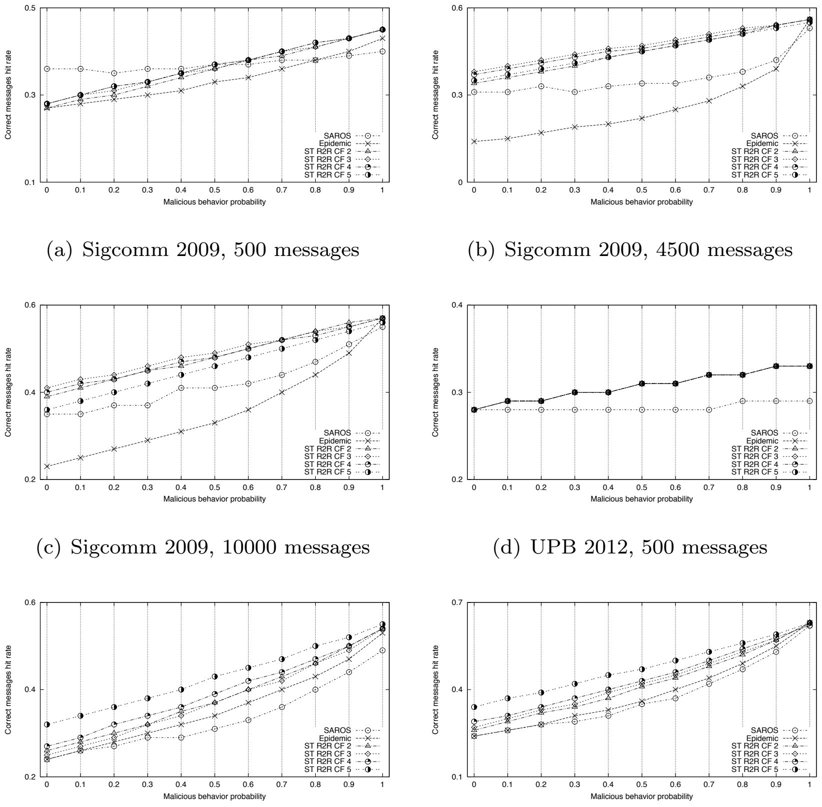
SAROS, and this only happens when less than 35% of the network nodes behave badly. Furthermore, while the correctness of all the other algorithms drops as more malicious nodes are added to the network, SAROS is able to maintain an almost constant value, even increasing it while the number of malicious nodes grows. Similarly to the previous scenario, the correctness of SAROS never goes below 60%. It is also very important to note that, while the correctness of Epidemic and Social Trust decreases much faster with the increase in the number of malicious nodes (i.e. compared to the first scenario), SAROS' correctness is more or less the same for both scenarios. The same is also true for the hit rate of correct messages, as shown in Figure 4. It should also be noted that the best Social Trust version is the one with five common friends.

Similarly to Sigcomm 2009, the results obtained on UPB 2012 (as shown in Figures 4(d), 4(e) and 4(f)) for the current scenario show that SAROS keeps a constant behavior regardless of which nodes are malicious. Thus, the results show that SAROS' correctness is always better than the one obtained by Social Trust or Epidemic, and that the correct messages hit rate is equal to or better than for the other algorithms.

5.2.3. Varying a Node's Malicious Behavior Probability

One possibility for malicious nodes to trick reputation algorithms is to offer good service sometimes. This way, there is a chance that a reputation system might believe that a malicious node behaves normally, and would increase the trust in that node erroneously. For this reason, we varied a malicious node's bad behavior probability from 1 (i.e. always act maliciously) to 0 (i.e. never modify a message) in increments of 0.1. We tested for data memories of 500, 4500, and 10000, with the number of malicious nodes in the network being about 60% of the total number of nodes (i.e. 45 for the Sigcomm 2009 trace and 15 for UPB 2012).

The results for the two traces are shown in Figure 5. It can be seen that, for smaller data memories, all solutions behave similarly on the Sigcomm 2009 trace, with SAROS behaving slightly worse as the bad behavior probability



(a) Sigcomm 2009, 500 messages

(b) Sigcomm 2009, 4500 messages

(c) Sigcomm 2009, 10000 messages

(d) UPB 2012, 500 messages

(e) UPB 2012, 4500 messages

(f) UPB 2012, 10000 messages

Figure 5: Correct messages hit rates when malicious nodes sometimes behave correctly.

increases. However, for higher data memories, the correct messages hit rate for SAROS grows slower than for Social Trust. This most likely happens because the quorum algorithm increases the trust in a malicious node when it offers good service, and it takes longer for this trust to be dropped back upon a wrong transaction. One way to mitigate this would be to assign a higher weight to a negative transaction, so that a malicious node's reputation would drop quicker when it is caught behaving badly. Another solution would be to increase the number of quorum message copies required, since this would increase the probability of the node acting maliciously at one point.

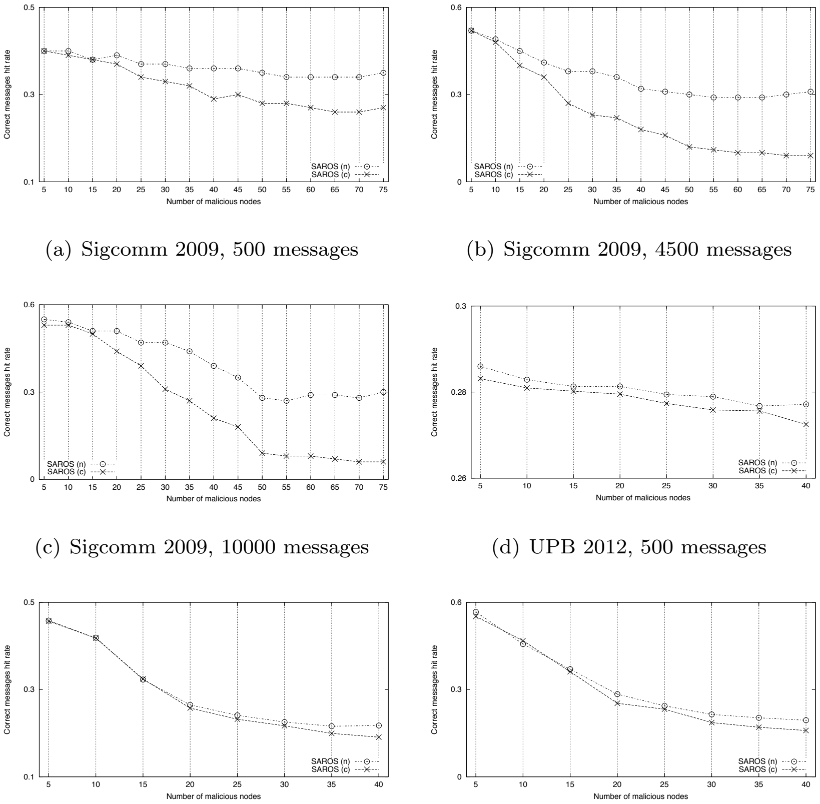
The results for UPB 2012, also presented in Figure 5, show that the same observations hold for this trace as well, only that the correct messages hit rate of SAROS grows a little quicker.

5.2.4. Malicious Node Collaboration

For this scenario, we assume that malicious nodes act in collaboration. Thus, we assume that, whenever a malicious node receives a message and wants to modify it, it will do so with a pre-set new version. All modified messages will therefore look the same, so the quorum algorithm will be easier to trick. All the other testing parameters are set as the ones from the first scenario.

The correct messages hit rates for both traces can be seen in Figure 6. For the Sigcomm 2009 case, when nodes have a small data memory (i.e. 500), the correct messages hit rate when nodes collaborate does not drop very much upon increasing the number of malicious nodes, when compared to the hit rate when nodes do not collaborate (only from 0.35 to 0.27). However, node collaboration can be very efficient when nodes have higher data memories, as seen in Figure 6. On the other hand, the largest drop-off starts at about 20 malicious nodes. Having this amount of nodes in a network of 76 nodes (so approximately 25%) collaborate would be difficult to achieve in real-life, but it is something that should be taken into consideration when implementing an opportunistic network.

However, as also seen in Figure 6, the same cannot be said about malicious node collaboration in the UPB 2012 trace. There, regardless of the size of a node's data memory, the correct messages hit rates when malicious nodes collaborate are very close to the ones obtained when no collaboration occurs. This happens because the trace is dense, and there are many contacts between non-malicious nodes. Thus, a message might already be disseminated before incorrect copies of it appear, so it does not matter that malicious nodes modify all the messages in the same way.



(a) Sigcomm 2009, 500 messages

(b) Sigcomm 2009, 4500 messages

(c) Sigcomm 2009, 10000 messages

(d) UPB 2012, 500 messages

(e) UPB 2012, 4500 messages

(f) UPB 2012, 10000 messages

Figure 6: Correct messages hit rates when malicious nodes collaborate ("n" means no collaboration, "c" means collaboration).

5.2.5. Impact of the Quorum Algorithm

We have also studied the impact that employing the quorum algorithm to select a correct message has on the delivery latency, i.e. the time it takes for a message to be successfully delivered to interested nodes. Since SAROS waits for other versions of a message to arrive, the delivery latency is naturally affected, and Figure 7 shows to what extent. We have performed our analysis on the Sigcomm 2009 trace, where nodes have a data memory of 500 messages, and there are 35 malicious nodes in the network. We have used the lowest delivery

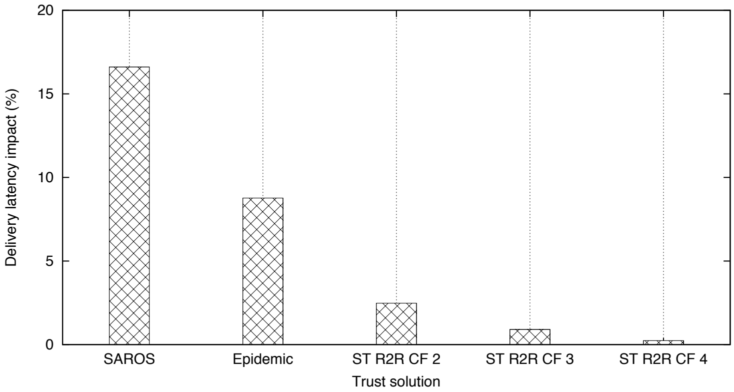


Figure 7: Impact of SAROS on delivery latency.

latency (the one belonging to Social Trust with five common friends) as the baseline, and we show the percentage that the latency is increased with for all the other tested solutions.

As can be seen in Figure 7, SAROS' quorum algorithm increases the delivery latency by 16.61%, the most of all the other solutions. However, this happens on a test case where SAROS outperforms all other solutions in terms of correct messages hit rate with as much as 18%. The results for other test cases are similar to the ones shown in Figure 7. We believe that it is more acceptable to increase the delivery latency if we are guaranteed a higher percentage of correct messages spread in the network, since incorrect messages can lead to malicious data being spread in the network. Although opportunistic networks are delay-tolerant networks, we wish to further investigate new ways of reducing the impact in delivery latency in the future.

5.2.6. Impact of Gossiping

Exchanging trust information through gossiping implies additional data to be exchanged at each contact between two nodes. We attempt to minimize this data by using integer values for storing the trust values. Although both trust parameters can have values between 0 and 1, we represent them as integers between 0 and 100 (e.g. 0.5 is represented as 50). Thus, we only need a single byte for each of the two trust values. Each node stores an array of local and global trust values for each encountered node, so the amount of extra data that has to be sent by a node upon a contact can be no higher than $2 \times N$ bytes, where N is the total number of nodes in the network. Thus, in a network of 100 nodes, only 200 extra bytes need to be exchanged for each contact, which can be achieved by Bluetooth in 0.4 seconds (assuming a Bluetooth transfer speed of 2.1 Mbps), and by WiFi Direct in 0.0003 seconds (for a transfer speed of 250 Mbps). Even for larger networks, the amount of data to be transferred will not be much higher, since a node only sees a subset of all the nodes in the entire network. For performance purposes, a node from a large network can only store trust data for the most recently encountered nodes, to avoid having to exchange too much data.

6. Conclusions

In this paper, we have addressed the issue of trust and reputation in opportunistic networks. We have proposed and presented SAROS, our own approach to trust in ONs, which is part of the Interest Spaces framework for opportunistic dissemination. It attempts to detect and punish malicious nodes, in order to guarantee that a message that is received by a node has not been tampered with. We have analyzed SAROS' behavior thoroughly and have shown that it behaves well in various scenarios. We have tested SAROS while increasing the number of malicious nodes in the ON, infecting popular nodes, making malicious nodes only behave badly sometimes, and making malicious nodes collaborate. We have compared it to existing solutions and have shown that it outperforms them in most cases.

In the future, we wish to continue our work by improving SAROS in terms of malicious nodes collaboration. Furthermore, we wish to decrease the impact that our solution has on the delivery latency.

References

[1] R.-I. Ciobanu, R.-C. Marin, C. Dobre, V. Cristea, C. X. Mavromoustakis, G. Mastorakis, Opportunistic dissemination using context-based data aggregation over interest spaces, in: Communications (ICC), 2015 IEEE International Conference on, 2015, pp. 1219–1225. doi:10.1109/ICC.2015.7248489.

[2] R.-I. Ciobanu, R.-C. Marin, C. Dobre, V. Cristea, Interest Spaces: a Unified Interest-Based Dissemination Framework for Opportunistic Networks, submitted to Computer Networks Journal (September 2015).

[3] Y. Wu, Y. Zhao, M. Riguidel, G. Wang, P. Yi, Security and trust management in opportunistic networks: a survey, Security and Communication Networks 8 (9) (2015) 1812–1827. doi:10.1002/sec.1116.
URL http://dx.doi.org/10.1002/sec.1116

[4] N. Li, S. K. Das, Radon: Reputation-assisted data forwarding in opportunistic networks, in: Proceedings of the Second International Workshop on Mobile Opportunistic Networking, MobiOpp '10, ACM, New York, NY, USA, 2010, pp. 8–14. doi:10.1145/1755743.1755746.
URL http://doi.acm.org/10.1145/1755743.1755746

[5] N. Li, S. K. Das, A trust-based framework for data forwarding in opportunistic networks, Ad Hoc Netw. 11 (4) (2013) 1497–1509. doi:10.1016/j.adhoc.2011.01.018.
URL http://dx.doi.org/10.1016/j.adhoc.2011.01.018

[6] A. Mtibaa, K. A. Harras, Social-based trust in mobile opportunistic networks, in: Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on, 2011, pp. 1–6. doi:10.1109/ICCCN.2011.6006047.

[7] S. Trifunovic, F. Legendre, C. Anastasiades, Social trust in opportunistic

networks, in: INFOCOM IEEE Conference on Computer Communications Workshops , 2010, 2010, pp. 1–6. doi:10.1109/INFCOMW.2010.5466696.

[8] S. Trifunovic, F. Legendre, Trust in opportunistic networks, Computer Engineering and Networks Laboratory (2009) 1–12.

[9] M. Raya, P. Papadimitratos, V. D. Gligor, J.-P. Hubaux, On data-centric trust establishment in ephemeral ad hoc networks, in: INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, 2008. doi:10.1109/INFOCOM.2008.180.

[10] C. Rohner, F. Bjurefors, P. Gunningberg, L. McNamara, E. Nordström, Making the most of your contacts: Transfer ordering in data-centric opportunistic networks, in: Proceedings of the Third ACM International Workshop on Mobile Opportunistic Networks, MobiOpp '12, ACM, New York, NY, USA, 2012, pp. 53–60. doi:10.1145/2159576.2159589.
URL http://doi.acm.org/10.1145/2159576.2159589

[11] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, in: Proceedings of the 12th International Conference on World Wide Web, WWW '03, ACM, New York, NY, USA, 2003, pp. 640–651. doi:10.1145/775152.775242.
URL http://doi.acm.org/10.1145/775152.775242

[12] R. Zhou, K. Hwang, Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing, IEEE Transactions on Parallel and Distributed Systems 18 (4) (2007) 460–473. doi:http://doi.ieeecomputersociety.org/10.1109/TPDS.2007.1015.

[13] M. R. P. Goncalves, E. dos Santos Moreira, L. A. F. Martimiano, Trust management in opportunistic networks, in: Networks (ICN), 2010 Ninth International Conference on, 2010, pp. 209–214. doi:10.1109/ICN.2010.41.

[14] R. I. Ciobanu, C. Dobre, V. Cristea, Social aspects to support opportunistic networks in an academic environment, in: Proceedings of the 11th international conference on Ad-hoc, Mobile, and Wireless Networks, ADHOC-NOW'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 69–82. `doi:10.1007/978-3-642-31638-8\_6`.
URL `http://dx.doi.org/10.1007/978-3-642-31638-8_6`

[15] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, C. Diot, MobiClique: middleware for mobile social networking, in: Proceedings of the 2nd ACM workshop on Online social networks, WOSN '09, ACM, New York, NY, USA, 2009, pp. 49–54. `doi:10.1145/1592665.1592678`.
URL `http://doi.acm.org/10.1145/1592665.1592678`

[16] R.-C. Marin, C. Dobre, F. Xhafa, Exploring Predictability in Mobile Interaction, in: Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on, IEEE, 2012, pp. 133–139. `doi:10.1109/EIDWT.2012.29`.
URL `http://dx.doi.org/10.1109/EIDWT.2012.29`

[17] A. Vahdat, D. Becker, Epidemic Routing for Partially-Connected Ad Hoc Networks, Tech. rep., Duke University (Apr. 2000).
URL `http://issg.cs.duke.edu/epidemic/epidemic.pdf`