SENSE: A Collaborative Selfish Node Detection and Incentive Mechanism for Opportunistic Networks[☆]

Radu-Ioan Ciobanu[a], Ciprian Dobre[a,*], Mihai Dascălu[a], Ștefan Trăușan-Matu[a], Valentin Cristea[a]

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

Abstract

Generally, routing algorithms for opportunistic networks rely on the fact that nodes are willing to accept data from other nodes under any circumstances, but this is not the case. Selfish nodes might not want to participate in the routing process for various reasons, such as low resources (e.g. battery, memory, CPU, network bandwidth), fear of malicious data from unknown users, or even lack of interest in helping nodes from other communities. Therefore, these types of nodes should be detected and avoided in the routing process. Moreover, incentive mechanisms that reward nodes when they actively take part in the network and punish them when they don't, should be employed where possible. In this paper, we propose SENSE, a novel social-based collaborative content and context-based selfish node detection algorithm with an incentive mechanism, which aims to reduce the issues of having selfish nodes in an opportunistic network. Since local information may not be sufficient to reach an informed decision, nodes running SENSE collaborate through gos-

[☆]This work was supported by project "ERRIC -Empowering Romanian Research on Intelligent Information Technologies/FP7-REGPOT-2010-1", ID: 264207. The work has been cofounded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557. The paper has benefited from the collaborative research efforts of the EU Green-Net group.

[*]Principal corresponding author

Email addresses: `radu.ciobanu@cti.pub.ro` (Radu-Ioan Ciobanu), `ciprian.dobre@cs.pub.ro` (Ciprian Dobre), `mihai.dascalu@cs.pub.ro` (Mihai Dascălu), `stefan.trausan@cs.pub.ro` (Ștefan Trăușan-Matu), `valentin.cristea@cs.pub.ro` (Valentin Cristea)

siping, for the final goal of detecting selfish nodes, punishing and avoiding them. We compare our approach to an existing algorithm (IRONMAN) and show that it behaves better in terms of network performance and detection accuracy. Moreover, we show that SENSE behaves well even when we simulate the existence of devices with batteries that get depleted, thus rendering them inactive for given time periods.

Keywords: collaborative, networking, opportunistic, selfishness, social

1. Introduction

The emergence of mobile devices has helped create the premises for various new means of communication and interaction, particularly in the area of Delay-Tolerant Networks (DTNs). Opportunistic networks are a recently proposed type of such networks, which are established in environments where human-carried mobile devices act as network nodes and are able to exchange data while in proximity. Opportunistic networks are based on the store-carry-and-forward paradigm [1], where a node stores a message, carries it around until it encounters its destination or another node more likely to deliver the message to the destination, and then forwards it.

The assumption generally made by various opportunistic routing algorithms is that nodes are willing to participate in the routing process at all times, but in a real-life scenario this isn't necessarily the case. Some selfish nodes may choose not to take part in the routing process. A node may be selfish for node A and unselfish for node B, meaning that it can decide to help node B with its routing (if, for example, B belongs to its own community), but not necessarily node A. There are several reasons why a node may be selfish. For example, it might be low on resources (battery life, memory, CPU) at a certain point and would like to save them for future use. Another reason for selfishness might be the fear of malicious data from unknown users, or even the lack of interest in helping nodes from other communities. We refer to these nodes as selfish, although they are not actually selfish in the true sense of the word, as all the reasons previously stated are valid reasons for not participating in data routing. However, because from the point of view of other nodes they are selfish (since they do not aid in the routing process), we will keep refrering to them as selfish nodes.

Thus, selfish nodes should be detected and avoided in the routing process. Furthermore, incentive mechanisms that reward nodes when they actively take part in the network and punish them when they don't, should be employed. In this paper, we propose SENSE, a novel social-based collaborative content and context-based selfish node detection algorithm with an incentive mechanism that aims to reduce the impact of selfish nodes in an opportunistic network. Since information collected locally by each node may not be sufficient to reach an informed decision, nodes running SENSE collaborate through gossiping. After informing each other of their observations, nodes reach decisions individually based on their local and received information. Unlike other collaborative detection algorithms, nodes don't have to rely on other nodes doing their computations. Moreover, perceived altruism values for other nodes are not binary, so a node is not considered totally selfish or altruistic. Instead, the altruism is a value between 0 and 1, and the decision to use a node for routing can be done at a higher level.

SENSE bases its analysis on the current context, such as social knowledge or information about the device's battery. Social information is used because nodes tend to interact more and be more altruistic towards members of their own community [2], while the battery status helps decide if a node was being selfish due to the fact that it was low on resources. Furthermore, SENSE is also content-based since it analyzes every message a node has sent and makes decisions based on its type (such as source and destination communities, message priority, age, hop count, etc.).

Since humans constitute the network, human altruistic behavior is an important factor in the feasibility of an opportunistic network. Authors of [2] first presented a systematic study of the impact of altruism on communication, particularly on opportunistic networks. Based on this study, IRONMAN [3] is currently a well-known implementation of an incentive mechanism for opportunistic networks that uses pre-existing social information to detect and punish selfish nodes, incentivising them to participate in the network. In order to provide more insight to our approach, we compare it to IRONMAN and show that it behaves better in terms of network performance and detection accuracy. We also show that SENSE behaves well when we simulate the existence of battery-powered devices that become inactive for certain time periods (e.g. when they have to recharge). We chose IRONMAN for comparing our proposed solution to, because it has been shown to have good results compared to previous existing mechanisms.

The following section presents notions about altruism modeling in practice and describes other similar solutions for selfish node detection in opportunistic networks, as well as several incentive mechanisms. Section 3 presents some simple improvements that can be brought to the existing IRONMAN algorithm; it also proposes SENSE, a novel social-based collaborative content and context-based selfish node detection algorithm with an incentive mechanism. Section 4 compares the results obtained by IRONMAN and SENSE in three different scenarios, while Sect. 5 presents conclusions and future work.

2. Related Work

2.1. Altruism Modeling

In order to design an opportunistic network, an altruism model that specifies how selfish nodes are spread in the network and how they behave towards the nodes they encounter should be used. In [2] and [4], several such models are presented: percentage of selfishness, uniform, normal, geometric, degree-biased or community-biased distributions. In the percentage of selfishness model, nodes may be totally selfish or totally altruistic, and there is a given percentage of selfish nodes in the network. However, this model is unrealistic, since a node generally isn't fully selfish or fully altruistic.

The uniform distribution model assumes that the altruism values for nodes in an opportunistic network are uniformly distributed, while the normal distribution model uses a Gaussian distribution, restricting and normalizing the range of values using the 5% and 95% values of the cumulative distribution function [4]. The advantages of these two models are that they are popularly encountered in nature and are relatively easy to implement.

When employing a geometric distribution, altruism values are computed for node pairs, each of them following a distribution in which the probability decreases with the social hop-distance [2]. The degree-biased distribution is based on the assumption that nodes become more popular and have many social connections because the owners of the devices are willing to help other people, while the community-biased model assumes that people in a community have greater incentives to carry messages for other members of the same community. In this case, altruism is modeled using an intra and an inter-community altruism level, with the first value being higher.

For our purpose, the most suitable model would be the community-biased distribution model, since we are dealing with an opportunistic network where the nodes are mobile devices carried by humans that interact based on social relationships. However, altruism values should also be distributed inside a community (i.e. not all nodes in the same community should have the same altruistic values towards each other), using a uniform or normal distribution.

The altruism model can be complemented with a battery consumption model [5] that assumes a node starts with a certain battery level that decreases in time. There are also several situations in which the battery consumption model can be integrated with the altruism model. For example, a node can behave normally according to its preset altruism until the battery level reaches a certain threshold. When this happens, the node doesn't exchange data until the battery is recharged. We propose a more complex method to compute a new altruism value, based on the original altruism a_i and the current battery level:

$$b_i = a_i * e^{\frac{E}{E_{max}} - 1}$$

where b_i is the battery-based altruism of node i given by the chosen model, E is the current residual capacity of the battery, E_{max} is the maximum capacity of the battery, and e is Euler's number. We chose this function because we needed an exponential decrease in altruism with regard to the percentage of battery available. Additionally, a threshold can also be enforced to the altruism model proposed above, so that when the battery level is below a certain value, the altruism automatically becomes 0. We chose an exponential function because we believe that a node nearing the critical level of battery depletion should take advantage of the fact that it still has some battery left and attempt to make itself known as unselfish. If the altruism drops linearly, other nodes might consider it selfish even when it has an acceptable battery level (e.g. 30%), which would lead to the node not being helped by others after it has recharged its battery.

Regarding the behavior of a node when it is selfish, another node can send it messages, but the sender can't know if those messages were received or if they are ever going to be delivered. This is inherent in mobile networks based on WiFi or Bluetooth.

2.2. Selfishness Detection Algorithms and Incentive Mechanisms

Although it has been shown that opportunistic networks are robust towards altruism distribution [2], detecting the selfish nodes and avoiding them (or making them unselfish) can lower the unnecessary loss of resources or the delays that might appear. Therefore, several methods for the detection of selfish nodes in DTNs have been proposed in previous works.

The selfish node detection mechanism for DTNs and Mobile Ad Hoc Networks (MANETs) described in [6] uses a collaborative watchdog approach to detect selfish nodes and spread this information in the network. There are three states regarding the perceived altruism of another node in the network: either no information exists, the node is known as selfish, or the node is known as altruistic. If a node A has no information about node B and a "selfish" or "unselfish" notification is received from another node, then A's perceived state of B is set to the received value. If A sees B as either selfish or altruistic and receives the opposite value, then it resets B's perceived value to the no-information state. The main disadvantages of this approach are that it assumes that a node can either be fully altruistic or fully selfish, and that a node's perceived state can fluctuate heavily if contradictory information comes from different nodes.

We propose an approach that uses values between 0 and 1 for a node's altruism since it is more realistic, and computes perceived altruism values based on both context (social knowledge, battery level) and content (computations are performed by analyzing each message). Our approach is somewhat similar to [7], where gossiping is used by nodes to spread their interpretation of the monitoring level, for a faster detection of selfish nodes in the network. Another method [8] splits selfish nodes into *free riders*, *black holes* and *novas*, and uses message path analysis to separate them from other nodes.

However, simply detecting selfish nodes may not be enough to improve the performance of a network [6]. An incentive mechanism may, for example, not accept messages from nodes considered selfish, thus forcing them to participate if they want their messages delivered. Such a mechanism is IRONMAN [3], which uses pre-existing social network information to detect and punish selfish nodes, incentivising them to participate in the network. Each node stores a perceived altruism (or trust) value for other nodes, that is initialized based on the social network layout: if the nodes are socially connected, this value is higher than for non-community nodes. When a node A meets a node B, it checks its encounter history to see if B has ever created a message for A that has been relayed to another node C. If this is the case, and A has encountered C after B had given it the message but A didn't receive the message, then C is considered selfish, and A's perceived altruism of C is decreased. Whenever a node A receives a message from a node B which is not the source of the message, A's perceived altruism of B is increased.

Apart from detecting selfish nodes, IRONMAN also uses incentives to make nodes behave better. Therefore, whenever a node B is considered selfish by A (its perceived altruism is below a given threshold), it is notified, and A won't accept any messages from it (but will keep on forwarding its own messages to B). Therefore, a selfish node might end up not being able to send its messages, unless it becomes altruistic. Like IRONMAN, SENSE also takes advantage of social information about the nodes, only it doesn't use it solely at the bootstrapping phase, but also during the algorithm's run time. We compare our proposed solution to IRONMAN and highlight the advantages it brings in Sect. 4.

3. Selfish Nodes in Opportunistic Networks

3.1. IRONMAN improvements

The IRONMAN algorithm for detecting and discouraging selfishness in opportunistic networks [3] uses perceived altruism ratings for encountered nodes, in order to decide if they are selfish or not. These ratings are computed locally based on the analysis of the history of contacts whenever two nodes meet, but the local values are exchanged with other nodes at every encounter, in order to inform them if a node is selfish. The local values are combined with the foreign values from other nodes through an addition. However, we believe that certain issues may appear because of the way the ratings are computed, and we propose an alternative to correct this.

For example, let's assume we have a node C that is selfish, and two other nodes A and B that have not encountered C yet, so they know nothing about it. In [3], the default perceived altruism value for an unknown node is equal to the threshold that separates selfish from altruistic values. If nodes A and B meet each other several times before any of them encounters C, then every time they get in range, the perceived altruism of C will double, although none of the nodes knows C. If, for example, A and B start with an altruism of 50 for C (with the altruism threshold also being equal to 50), after five encounters, the rating will end up being 800 for both nodes. This can easily happen when A and B don't encounter other nodes that have seen C. When finally A or B meet C or another node that has seen C, it will take some time before the perceived altruism value decreases under the threshold, so messages may be sent to C, which will not forward them. This may lead to data loss or to increased delays in data delivery.

In light of the previous example, we propose using an average instead of a sum when updating the altruism value of a node upon an encounter, because it will keep the values around the same point, while becoming sensible to newly received information. Therefore, we propose using the average of the local and foreign altruism values when computing the new perceived altruism rate of a node. This version of IRONMAN will be referred to as IRONMAN-AVE for the rest of the paper. The second alternative assumes that locally-collected information is more reliable, so it favors the local perceived altruism value. Therefore, the new altruism rate is computed as a weighted average with a weight of 0.9 for the local value and 0.1 for the foreign one (this version of IRONMAN will be referred to as IRONMAN-WEI). The weight values 0.9 and 0.1 were selected after multiple experimental iterations and seem to provide the most suitable and reliable solution.

3.2. The SENSE Algorithm

SENSE assumes that each node has a unique ID that can be used to identify it within the entire network. This ID may be set by a global service whenever a node enters the network, or (since opportunistic networks are generally fully decentralized) it can be the IMEI code for phones and tablets, the MAC address for other devices, or any other uniquely identifying number. For privacy concerns, we propose that if the latter approach is employed, the IMEI code or MAC address should be hashed in order to obtain the actual ID. Aside from the ID, a node stores social information, i.e. the relationships it has with other participants in the network. Since it has been previously proven that nodes in opportunistic networks tend to interact more with members of their own social community than with other nodes [9, 10], we consider this information to be helpful in designing our selfishness detection algorithm. Furthermore, as we have shown in Sect. 2.1, nodes tend to be less selfish towards their own communities, so having knowledge about a node's social connections might help in deciding whether it was being selfish or if it couldn't deliver a message due to other reasons (such as insufficient space in the data memory). Therefore, a node contains information about its own community as a list of nodes it has social relationships with. For the sake of simplicity, we do not take into consideration the strength of a social relationship, so two nodes are either socially connected or not. The community can either be taken from various social networks such as Facebook or Google+, but when this information is not available, we use a distributed community detection algorithm such as k-CLIQUE [11]. Also, the battery level is another contextual information that a node has access to at an encounter, and that it may use in the altruism computation process.

Aside from information regarding its ID, community, and battery level, each SENSE node has its data memory split into four sections. The role of these sections is to control the number of messages stored for forwarding, as well as the amount of contact history each node is aware of. A message may have multiple copies in the memory, but they only occupy the space of a single message, since a counter is used to specify the number of copies. The data memory is therefore split as follows:

1. G - messages generated by the node itself. There is a separate list for these messages because they should be kept at the generating node for a longer period of time than messages carried for other nodes.
2. C - messages that the node stores, carries and then finally forwards (or drops, if the memory is full) for other nodes.
3. O - past forwards. It contains information regarding past message forward operations performed either by the current node, or by other nodes. Therefore, the following information is stored: ID and community of both nodes that participated in the forward (sender and receiver), time of the encounter, encountered node's battery level when the contact occurred, and metadata about the message that has been exchanged between the sender and the receiver (source and destination node IDs, priority, etc.).
4. I - past receives. It contains information regarding past message receive operations performed either by the current node, or by other nodes, and the stored data is similar to the one from O. Both I, as well as O, are updated whenever a new data exchange takes place (e.g. if node A sends a message to node B, an entry will be added in A's O and one in B's I).

When two nodes A and B running SENSE meet, they perform a series of steps (also presented in Alg. 1). Firstly, each node checks if its battery level is above a certain threshold, and if it's not, then that node will not participate in the data exchange. Secondly, each node computes an altruism value for the other node, as specified in Sect. 2.1 and, based on that value, decides if it will forward data for the other node. If the two nodes decide that they are unselfish towards one another, at the next step they exchange the I and O lists of past data transfers. When a node receives one of these lists, it updates its own list with the newly received information. This way, a node can have a more informed view of the behavior of various nodes in the network, through gossiping. We consider this to be an improvement over IRONMAN, since node A isn't simply told that node C has a certain degree of altruism based on the computations performed by B, but it is allowed to make the decision itself based on information gathered from encountered nodes. This is an advantage for our algorithm, as node C may be selfish towards node B because they are not socially connected, and unselfish with regard to node A because they belong to the same community. When this situation happens at IRONMAN, if node B decides that C is selfish and notifies A of this before A ever encounters node C, then A will end up considering C as being selfish, although that may not be the case. Since the sizes of O and I are limited, there may not be enough room for the entire history of contacts to be stored; therefore a node only keeps the most recent information in its own contact history memory.

After two nodes decide to be altruistic towards one another and they finish exchanging knowledge about past encounters, each of them advertises its own specific information, such as battery level and metadata about the messages it carries (which includes source and destination IDs). Based on the lists of past encounters I and O, each node computes a perceived altruism value for the other node with regard to the messages stored in its own data memory (in other words, it computes how willing the encountered node is to forward a certain type of message). If this value is within certain limits, the communication continues and the desired algorithm is applied. If (for example) node B's computed altruism is not within the given limits for any of A's stored messages, then it is considered selfish by node A, so A doesn't accept messages from B. Node A then notifies B that it considers it selfish, so B won't end up considering node A selfish (because this would lead to all nodes in the network becoming selfish eventually, since nobody would accept messages from anybody else). This also functions as an incentive mechanism, because if a node wants its messages to be routed by other nodes, it shouldn't be selfish towards them. Therefore, every time a node is notified that it is selfish in regard to a certain message, it increases its altruism value. If there is a social connection between the node considered selfish and the source of the message, then the inter-community altruism is increased. If the two nodes aren't socially connected, the intra-community altruism value grows.

The formula for computing perceived altruism values for a node N and a message m based on the lists of past forwards (O) and receives (I) is:

$$altruism(N, m) = \sum_{o \in O, i \in I, o.m = i.m}^{N.id = o.d, N.id = i.s} type(m, o.m) * thr(o.b)$$

In the previous formula, a past encounter x has a field $x.m$ which specifies the message that was sent or received, $x.s$ is the source of the transfer, $x.d$ is the destination and $x.b$ is the battery level of the source. *type* is a function that returns 1 if the types of the two messages received as parameters are the same (in terms of communities, priorities, etc.), and 0 otherwise, while *thr* returns 1 if the value received as parameter is higher than a preset threshold, and 0 if it's not. Basically, the altruism computation function counts how many messages of the same type as m have been forwarded with the help of node N, when N's battery was at an acceptable level. In other words, we exclude the cases in which a node N didn't forward a message when it had (for example) 2% battery left, since we wouldn't expect it to do that anyway. The result of the formula is normalized with the total number of message exchanges, so the final value of the perceived altruism is between 0 and 1.

Because the altruism computation takes into account information such as the battery life and social relationships, the algorithm can be described as *context-based*. Furthermore, as the perceived altruism value is computed with regards to the type of every message in a node's data memory, our proposed selfish detection algorithm is also *content-based*.

Algorithm 1 The SENSE algorithm when node A encounters node B

```
function ENCOUNTER($A$, $B$)
    exchange($A$, $B$, $I$, $O$)
    update($I$, $O$)
    for all messages $m$ in $B.G$ and $B.C$ do
        $altruism$ = compute_altruism($A$, $B$, $m$)
        if $altruism > altruism\_threshold$ then
            routing_algorithm($A$, $B$, $m$)
        else
            notify_selfish($B$, $m$)
        end if
    end for
end function

function NOTIFY_SELFISH($B$, $m$)
    if $B$ is in the same community as $m.source$ then
        increase $B.intra$
    else
        increase $B.inter$
    end if
end function
```

4. Evaluation

This section presents an evaluation of SENSE in various situations. We compare its results with the ones obtained when running IRONMAN and its two modified versions proposed in Sect. 3.1.

4.1. Test Setup

We tested both with human mobility traces, as well as using a synthetic mobility model, both of which are presented in this subsection.

4.1.1. Mobility Traces

All tests were performed on the MobEmu emulator [12] which parses human mobility traces and then applies an opportunistic routing and selfish node detection algorithm at every encounter between two nodes. We used three mobility traces, publicly available in the CRAWDAD archives [13]. UPB 2011 [10] and UPB 2012 [14] are two traces taken in an academic environment at the University Politehnica of Bucharest, where the participants were students and teachers. UPB 2011 includes data collected for a period of 25 days by 22 participants, while UPB 2012 had a duration of 64 days and involved 66 participants. St. Andrews [15] is a real-world mobility trace taken on the premises of the University of St. Andrews and its surroundings. It was taken by the authors of the IRONMAN algorithm and was also used within their tests. The trace lasted for 79 days and involved 27 participants that used T-mote Invent devices. The main advantage of these three traces is that they also include social information about participating nodes, in the form of a graph of social connections. As shown in Sect. 3.2, this information is necessary to our algorithm.

4.1.2. Mobility Model

Besides the mobility traces, we also tested SENSE and IRONMAN using the Home-Cell Community-based Mobility Model (HCMM) [16], a synthetic mobility model that tries to replicate the behavior of nodes in an opportunistic network. It is based on the caveman model [17], which assumes that nodes have home communities, where they spend most of their time, and acquainted communities, which they visit more rarely. Nodes in HCMM are driven not only by the social relationships between them, but also by the attractions of physical locations. According to this model, the attraction of an external cell is computed based on the relationships with nodes that have their home in that cell. When a node reaches a cell that is not its own home community cell, it stays there with a probability p_e, and returns to its home cell with the probability $1 - p_e$.

When testing with HCMM, we tried to simulate an academic environment. Therefore, we split the physical space into a 400x400-meter grid, with 10x10-meter cells, in order to simulate the campus of a university. The speed of the nodes was chosen between 1.25 and 1.5 meters per second, which is the average human speed, while the transmission radius of the nodes was 10 meters, which is the regular Bluetooth range. There were 33 nodes in the network, split into seven communities. The duration of the experiment was three days, and we used the HCMM community grouping to create the social network used for routing decisions.

4.1.3. Opportunistic Routing Algorithm

We performed our tests using Spray-and-Wait [18], an opportunistic routing algorithm where each message has a fixed number of copies (chosen based on the number of encounters in the network). At every encounter, a node sends half of its copies of a message to the encountered node, if the latter doesn't already contain it. When a single copy of the message is left at a node, it is only delivered directly to the destination. As stated in Sect. 3.2, when a node has multiple copies of a message, it simply keeps a counter that specifies the amount of copies available, so a message takes up a single slot in the data memory, regardless of the number of copies present.

4.1.4. Testing Parameters

When testing, we assumed we were in an environment with devices that have limited resources, therefore the data memory of a node was limited. We tested with multiple values, ranging from 20 to 4500. When a node has to receive a message, but doesn't have enough room in its memory, it discards the oldest message and replaces it with the new one. The size of the two history lists O and I was set empirically to 1000.

For all experiments, we tried to model the generation of messages so as to resemble a real-life academic environment. Thus, every weekday, each node from the trace generates 30 messages with destinations chosen based on its social relationships using a Zipf distribution with an exponent of 1. Therefore, a node has a higher chance of sending a message to another member of its community than to other nodes. Inside the community, the destinations are chosen randomly. The time of the day when the messages are sent is randomly chosen inside the two-hour interval when the most contacts occur for each trace. We chose a Zipf distribution of messages because it has been shown that data requests and sends follow power law distributions [19].

For the IRONMAN implementation, we used the description in [3] and the parameter values shown there. Thus, the default perceived altruism value for nodes in the social network was 100, and for unknown nodes it was 50. The trust threshold was also 50, as well as the behavioral constant added or subtracted when a node is found to be altruistic or selfish. These values are the same for IRONMAN-AVE and IRONMAN-WEI. Since we use a community-biased distribution as an altruism model, a node has two levels of altruism (one for nodes in its community, and one for nodes outside it), both between 0 and 1. Therefore, when a node is informed by another that it is considered selfish, it increases both these values by 0.1. This is done similarly for our algorithm, except that only one of the two altruism values is increased by 0.1, as shown in Sect. 3.2. Both the inter and the intra-community altruism values were distributed normally in the network with a mean of 0.4 for inter-community and 0.6 for intra-community. We opted for two different values in order to better model real-life scenarios, whereas the new mean values for the distributions were obtained as deviations of trust in terms of incremental steps (0.1) from the standard mean (0.5).

4.2. Testing Scenarios

Taking into consideration the previous setup, we performed several simulations and analyzed various metrics that highlight the benefits brought by SENSE.

4.2.1. Hit rate, latency, delivery cost, and hop count

For the first scenario, we looked at four performance metrics very important in opportunistic networks and how they are affected by selfishness, for the three mobility traces and HCMM. The *hit rate* is the ratio between successfully delivered messages and the total number of generated messages. The *delivery latency* is defined as the time passed between the generation of a message and its eventual delivery to the destination. The *delivery cost*, defined as the ratio between the total number of messages exchanged and the number of generated messages, shows the congestion of the network. The *hop count* is the number of nodes that carried a message until it reached the destination on the shortest path, and should also be as low as possible in order to avoid node congestion.

Trace	Participants	Max Copies
UPB 2011	22	10
UPB 2012	53	30
St. Andrews	27	6
HCMM	33	16

Table 1: Testing parameters

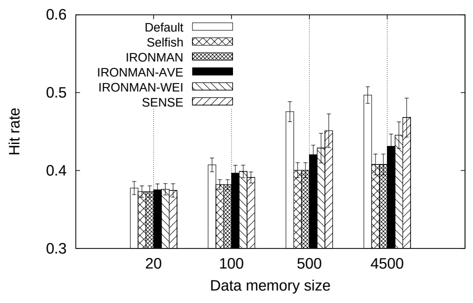
We ran the default Spray-and-Wait algorithm with and without selfish nodes (we called these the "default" and "selfish" cases), after which we applied the four selfish node detection and incentive algorithms: IRONMAN with its two versions, and SENSE. The total number of opportunistic network participants and the number of maximum message copies allowed for the Spray-and-Wait algorithm are shown in Table 1. For the UPB 2011 trace, the SENSE threshold was 5, while for the other traces and for HCMM it was set to 1 (this threshold was normalized with the total number of message exchanges).

4.2.2. Community-biased detection accuracy

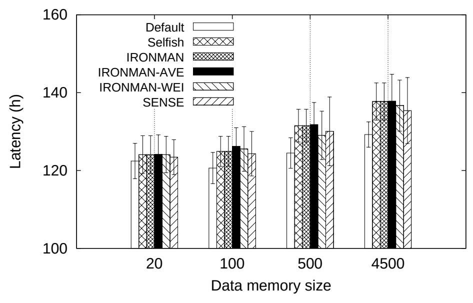
Normally, the detection accuracy is the proportion of selfish nodes that were correctly detected as selfish, but since we employ a community-biased altruism model, nodes aren't fully altruistic or fully selfish. Instead, as we are dealing with values between 0 and 1, we define the community-biased detection accuracy as the percentage of nodes that end up with an altruism value of 1 thanks to the incentive mechanisms used. These are the nodes that have been recognized by most nodes in the network as being selfish and thus avoided until their altruism levels increase and they start opportunistically carrying data for other nodes. The scenario was performed on the UPB 2012 trace.

4.2.3. Battery-aware scenario

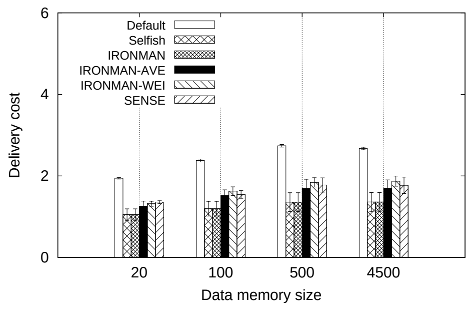
For the third scenario, we added a battery consumption model to the emulator, that assumes a node starts with a certain battery level which decreases in time (in our case, the battery lasts 24 hours and takes two hours to recharge). When the battery is below a certain threshold, the device doesn't participate in the routing process. We tested on the UPB 2012 trace with the same parameters as above, using the four metrics presented previously, with battery threshold values ranging from 10% to 25%. In this scenario, the



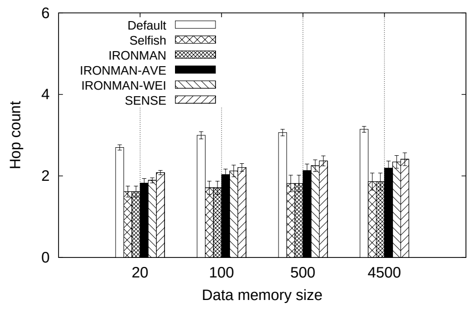
(a) Hit rate.



(b) Average latency.



(c) Delivery cost.



(d) Average hop count.

Figure 1: UPB 2011 trace.

residual capacity of the battery can decrease in time based on the following simple formula presented in [5]:

$$E = E - \sum_{d} I_d * \Delta t * V_{nominal}$$

4.3. Results for Opportunistic Network Metrics

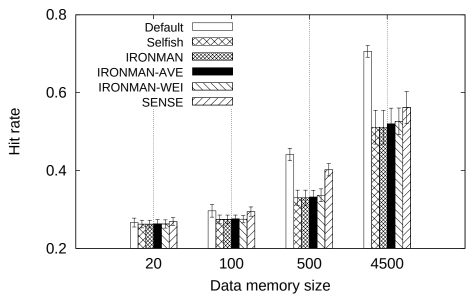
This subsection is focused on presenting the results obtained when running the first scenario, shown in Fig. 1 through 4.

Firstly, it can be seen from Fig. 1(a) that, for the UPB 2011 trace, adding selfishness to an opportunistic network leads to an important drop in hit rate, even for small data memory sizes (for a data memory of 100 messages, the difference between running with or without selfishness is 2.53%, while for a memory of 4500, it is as high as 8.92%). This happens because messages are sent to selfish nodes that end up dropping or never delivering them, and this is why a selfish node detection and an incentive mechanism is necessary.

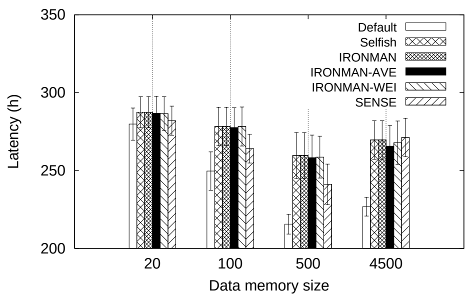
Selfish nodes must be avoided or convinced to become unselfish, in order to improve the hit rate. Figure 1(a) shows that using IRONMAN without any changes doesn't bring much improvement at all; the reasons are the ones described in Sect. 3.1, namely that selfish nodes are hard to detect because their perceived altruism value at other nodes grows very quickly. However, using IRONMAN-AVE and IRONMAN-WEI, we obtain a higher hit rate than the original IRONMAN, regardless of the size of the data memory. Moreover, the hit rate obtained for IRONMAN-AVE and IRONMAN-WEI is also higher than the one resulted when there is no selfish node detection algorithm employed. For this particular trace, the weighted method brings better results, which means that a node is able to gather sufficient information locally, and makes an informed decision based only on the data collected from other nodes (i.e. it doesn't need other nodes to compute its perceived values for it). Finally, for data memory sizes higher than 100, SENSE outperforms any version of IRONMAN and gets close to the hit rate obtained when no selfish nodes are present. We should also note that the hit rate improves when the size of the data memory increases, because older messages are deleted after a much longer time and the chance of replacing them before delivery becomes lower.

In Fig. 1(b), it can be observed that latency increases when there are selfish nodes in the network because giving messages to selfish nodes leads to their loss or to large delays in their arrival at the destination. For example, let's assume that a selfish node A carries a message destined for node B. When the two nodes first meet, A doesn't deliver the message because it is selfish. B may or may not infer that A is selfish, but the important part is that A doesn't deliver the message when it is supposed to, only much later (at a future contact, if any) or not at all. If the nodes hadn't been selfish, A would have delivered the message the first time it encountered B. Using IRONMAN doesn't decrease the latency, except for IRONMAN-WEI, which (as has been explained above) seems to be suitable for this trace. However, SENSE sees a drop in delivery latency of about 2 hours when compared to the selfish case. Although opportunistic networks are delay-tolerant, we believe that the latency should be decreased where possible, if we want a real-life implementation of such networks.

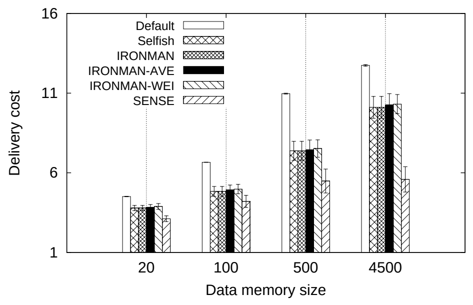
The delivery cost and hop count are also affected by the introduction of selfish nodes in the opportunistic network, as can be seen in Fig. 1(c) and Fig. 1(d). However, they decrease (which in this case is a positive change), but this only happens because the hit rate also drops. The messages that aren't



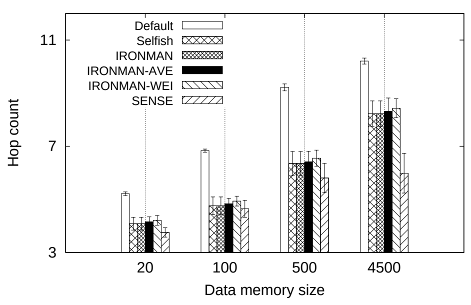
(a) Hit rate.



(b) Average latency.



(c) Delivery cost.

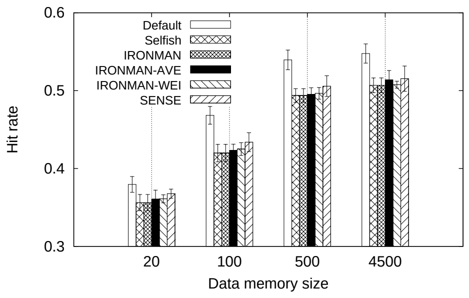


(d) Average hop count.

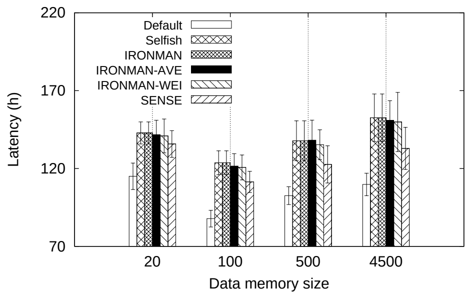
Figure 2: UPB 2012 trace.

delivered any more because of selfish nodes were probably ones with high delays, that moved a lot across the network, so they increased the average values. On closer inspection of the results for SENSE in comparison to the selfish case, we can see that the achieved gain in hit rate is higher than the loss in hop count and delivery latency, which further argues that our solution is appropriate for this scenario.

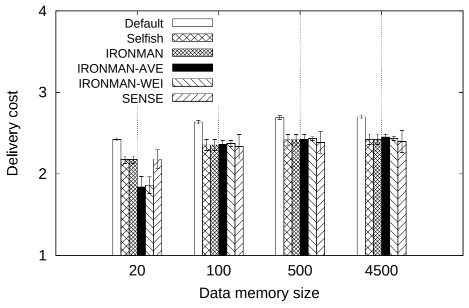
Since the UPB 2012 trace is similar to UPB 2011 (except that it lasted longer and had more users), the results obtained for this scenario are similar for these two traces, and this can be seen in Fig. 2. In terms of hit rate, IRONMAN brings an insignificant improvement when compared to the selfish case, while our two modified versions perform slightly better (especially IRONMAN-WEI). Our proposed algorithm helps improve the hit rate of a selfish node and outperforms all IRONMAN versions. Furthermore, for small data memories (20 messages), it even yields a better hit rate than the default case. This happens because, for the default case, nodes send messages



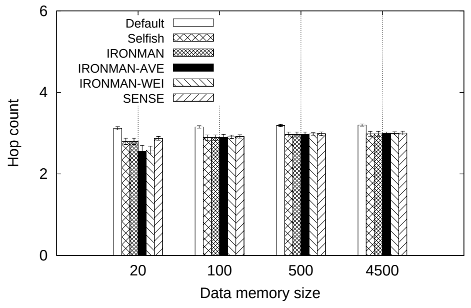
(a) Hit rate.



(b) Average latency.



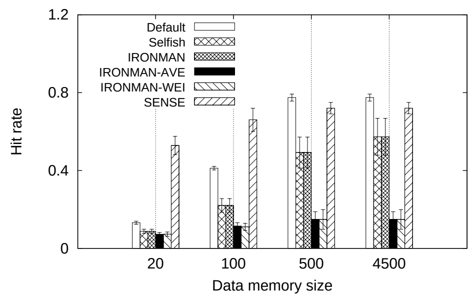
(c) Delivery cost.



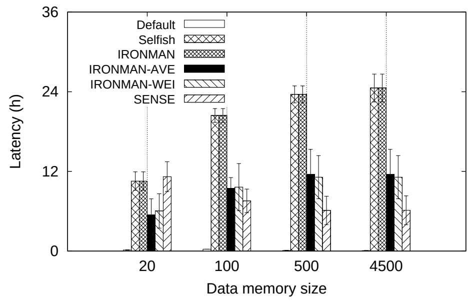
(d) Average hop count.

Figure 3: St. Andrews trace.

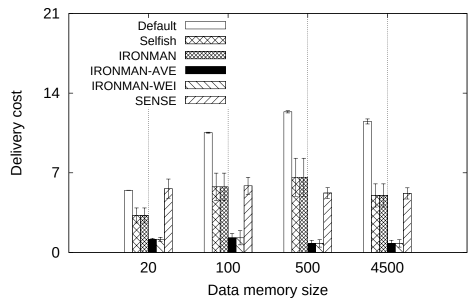
to any encountered nodes and (since the number of message copies is limited) may end up depleting them, while the nodes that receive them don't ever encounter the destinations. When employing our algorithm, messages aren't sent to just any node, but to nodes that have previously delivered similar messages, so there is a higher chance of them repeating this. The latency results are also similar to the ones obtained for UPB 2011. They show that SENSE has lower latencies than the selfish case and all three IRONMAN versions (the original, IRONMAN-AVE and IRONMAN-WEI), but still higher than for a network with unselfish nodes. However, the most important results are seen when analyzing the delivery cost in Fig. 2(c) and hop count in Fig. 2(d). Whereas at the UPB 2011 trace, our algorithm only yielded better values than the default scenario, for UPB 2012 the delivery cost and hop count for SENSE are better than for all other cases. This happens because messages are forwarded only to nodes that have previously forwarded similar messages.



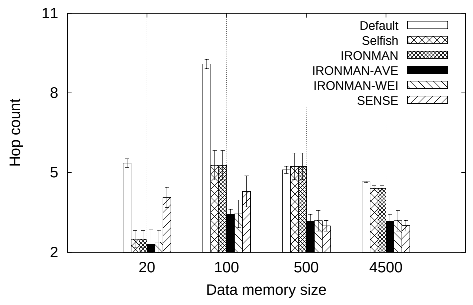
(a) Hit rate.



(b) Average latency.



(c) Delivery cost.

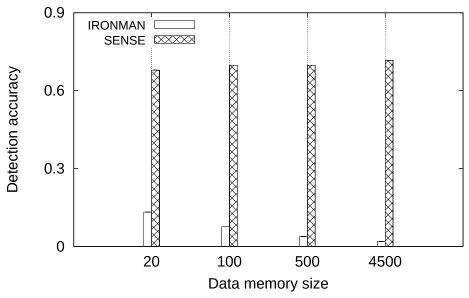


(d) Average hop count.

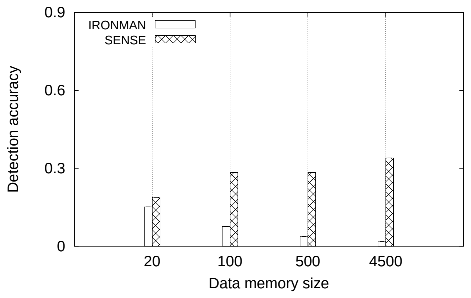
Figure 4: HCMM.

The results obtained when testing with the St. Andrews trace (shown in Fig. 3) are also very similar to what we obtained for UPB 2011 and UPB 2012. Thus, our algorithm obtains a better hit rate than the selfish and IRONMAN cases, a higher latency than the default case, but lower than all other cases, and a hop count and delivery cost lower than the ones from the default scenario and very close to what was obtained for the selfish case.

Fig. 4 shows the results of the first test scenario when simulating an opportunistic network using HCMM. SENSE performs very well in terms of hit rate: for a data memory of 20 and 100 messages, it yields much higher hit rates than even the default test case, with an improvement of 39.71% for a data memory of 20 messages, and 24.89% for 100 messages. This happens because we are dealing with many contacts and the copies of a node's messages get consumed really quickly in the default case, so the node only remains with one copy of each, that it can only deliver to the message's destination. For higher volume data memories, this no longer happens because nodes can



(a) Inter-community accuracy.



(b) Intra-community accuracy.

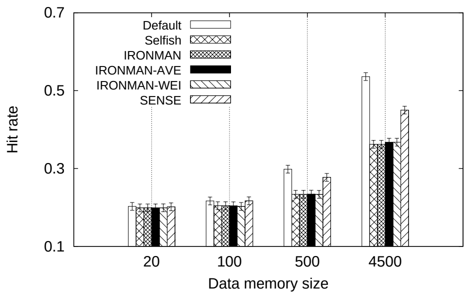
Figure 5: Community-biased detection accuracy.

store a larger amount of messages. Figure 4(a) also shows that, for HCMM, the original IRONMAN is better than our two versions. The latency charts show that having selfish nodes increases the latency, but SENSE can help decrease it. Finally, for delivery cost and hop count, IRONMAN-AVE and IRONMAN-WEI perform the best, but this is because they have very low hit rates (as seen in Fig. 4(a)). SENSE yields much better values for hop count and delivery cost, and obtains hit rates close to that of the default test case.

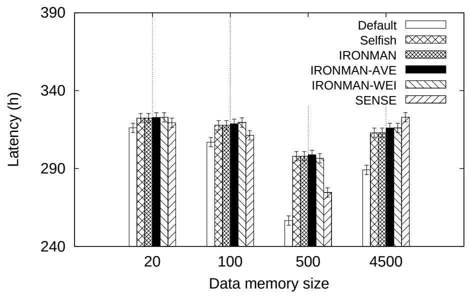
4.4. Results for Community-Biased Detection Accuracy

Figure 5 shows the community-biased detection accuracy for selfish nodes inside or outside the community. For nodes not belonging to the community, SENSE has a much better accuracy than IRONMAN: it is 5 times better for a data memory of 20 messages, and 37 times better for 4500 messages (the difference being 69.81%). This suggests that SENSE detects more nodes that are selfish towards non-community nodes than IRONMAN, and convinces them through incentives to become more altruistic. Our solution also performs better regarding intra-community nodes, but the differences are not so large (with a maximum improvement of 32.07%, which is 17 times better than IRONMAN).

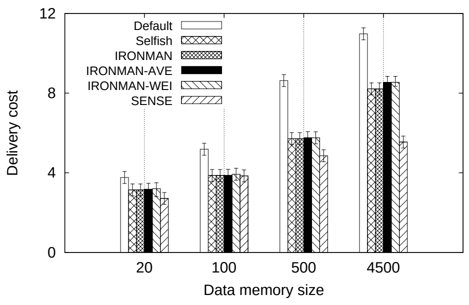
The high detection accuracy for SENSE is due to the analysis of encounter history, which is both context, as well as content-based. We not only base our decisions on the social relationships between the sending and receiving nodes or battery life, but we also compute perceived altruism values repeatedly, for each type of message.



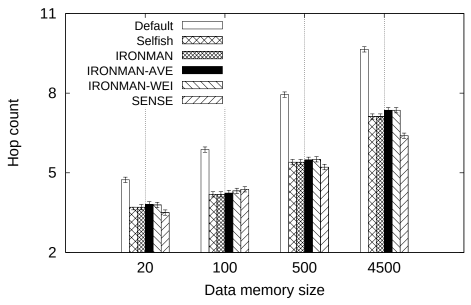
(a) Hit rate.



(b) Average latency.



(c) Delivery cost.



(d) Average hop count.

Figure 6: Battery-aware scenario.

4.5. Results for a Battery-Aware Scenario

Finally, Fig. 6 presents the hit rate, latency, delivery cost and hop count when employing a battery model while testing with the UPB 2012 trace. We highlight the improvements brought by using context-awareness in the decisions that SENSE makes. Since the battery-aware scenario was performed for the UPB 2012 trace, Fig. 6 looks similar to Fig. 2. However, it can be seen that, while the difference in hit rate between the default case and the selfish case is similar for the two scenarios (about 20% for a data memory of 4500), the improvement brought by our algorithm is almost double for the battery-aware scenario (10% vs. 5%). This means that our algorithm handles well the situation where nodes have depleting battery and can become selfish when the battery level is too low. Moreover, the improvements brought by any of the three versions of IRONMAN on the battery-aware scenario are close to 0, while on the previous UPB 2012 tests, there was a slight improvement obtained.

Regarding the other three parameters, the charts from Fig. 6 resemble the ones from UPB 2012, with the main difference being that the latencies are higher for the battery-aware scenario because nodes don't participate in the network for small intervals of time when they charge their battery. The hop count and delivery cost are also lower in the battery-aware scenario.

The reason why SENSE performs well is that it tries to avoid nodes with low battery by not sending them messages to be forwarded. This happens because, if we would send a message to a low-battery node, first of all the carrier node would lose some copies of its messages (as per the Spray-and-Wait algorithm), and secondly, the message received by the low-battery node would take longer to be delivered, since the node would be inactive while recharging. Moreover, SENSE has the advantage over IRONMAN that it doesn't consider low-battery nodes selfish, which would lead to them being avoided by other nodes. It just avoids them while they have a low battery level, but continues to use them for forwarding when they are charged.

5. Conclusions

We have presented SENSE, a novel social-based collaborative content and context-based selfish node detection algorithm with an incentive mechanism, which aims to reduce the issues of having selfish nodes in an opportunistic network. Our approach uses gossiping and context information to make informed decisions regarding the altruism of nodes in the network, on one hand, and incentive mechanisms to make selfish nodes become altruistic, on the other. Our solution takes advantage of social relationship knowledge regarding the nodes in the network to decide if a node is selfish towards its own community. Furthermore, it also makes its decisions based on content, since a node's perceived altruism level is computed with respect to every message in a node's data memory.

In terms of validation, we have tested SENSE on three scenarios and compared it to IRONMAN. We have shown that it outperforms IRONMAN in regards to hit rate and latency, and even that it fares better than the default case in terms of hop count and delivery cost. This happens because SENSE sends messages in a selective manner, only to nodes that have already successfully delivered messages of that type. Furthermore, we have also shown that SENSE has better detection accuracy than IRONMAN. In the end, we ran a scenario where a battery life model was used, and showed that our proposed algorithm significantly increases the hit rate.

As future work, we plan to test SENSE in other situations that do no necessarily resemble an academic environment. Furthermore, we wish to analyze its results while using different routing algorithms. An interesting research topic that derives from selfish node detection is how to find malicious nodes that behave unexpectedly on purpose, and how can these be avoided in order to ensure privacy and security.

References

[1] C. Boldrini, M. Conti, A. Passarella, in: Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems, MSWiM '08, ACM, New York, NY, USA, 2008, pp. 203–210.

[2] P. Hui, K. Xu, V. Li, J. Crowcroft, V. Latora, P. Lio, in: INFOCOM Workshops 2009, IEEE, pp. 1–6.

[3] G. Bigwood, T. Henderson, in: SocialCom/PASSAT, IEEE, 2011, pp. 65–72.

[4] K. Xu, P. Hui, V. O. K.Li, J. Crowcroft, V. Latora, P. Lio, in: Proceedings of the first international conference on Ubiquitous and future networks, ICUFN'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 153–158.

[5] L. M. Feeney, D. Willkomm, in: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, SIMUTools '10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2010, pp. 20:1–20:4.

[6] E. Hernández-Orallo, M. D. Serrat Olmos, J.-C. Cano, C. T. Calafate, P. Manzoni, in: Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems, MSWiM '12, ACM, New York, NY, USA, 2012, pp. 159–166.

[7] A. Lavinia, C. Dobre, F. Pop, V. Cristea, in: Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on, pp. 482 –489.

[8] Q. Zhou, J. Ying, M. Wu, in: Network Infrastructure and Digital Content, 2010 2nd IEEE International Conference on, pp. 835 –838.

[9] S. Okasha, The British Journal for the Philosophy of Science 56 (2005) 703–725.

[10] R. I. Ciobanu, C. Dobre, V. Cristea, 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.

[11] P. Hui, E. Yoneki, S. Y. Chan, J. Crowcroft, in: Proc. of 2nd ACM/IEEE inter. workshop on Mobility in the evolving internet architecture, MobiArch '07, ACM, New York, NY, USA, 2007, pp. 7:1–7:8.

[12] R. Ciobanu, C. Dobre, V. Cristea, D. Al-Jumeily, in: Parallel and Distributed Computing (ISPDC), 2012 11th International Symposium on, pp. 251 –258.

[13] CRAWDAD, http://crawdad.cs.dartmouth.edu/, 2012.

[14] R.-C. Marin, C. Dobre, F. Xhafa, in: Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on, IEEE, 2012, pp. 133–139.

[15] G. Bigwood, D. Rehunathan, M. Bateman, T. Henderson, S. Bhatti, in: Proceedings of the 2008 IEEE International Conference on Wireless & Mobile Computing, Networking & Communication, IEEE Computer Society, Washington, DC, USA, 2008, pp. 484–489.

[16] C. Boldrini, A. Passarella, Comput. Commun. 33 (2010) 1056–1074.

[17] J. Wu, D. J. Watts, SIGMOD Rec. 31 (2002) 74–75.

[18] T. Spyropoulos, K. Psounis, C. S. Raghavendra, in: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, WDTN '05, ACM, New York, NY, USA, 2005, pp. 252–259.

[19] L. A. Adamic, B. A. Huberman, SCIENCE 287 (2000) 2115.