Collaborative Selfish Node Detection with an Incentive Mechanism for Opportunistic Networks

Radu-Ioan Ciobanu, Ciprian Dobre, Mihai Dascalu, Stefan Trausan-Matu, Valentin Cristea
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
Bucharest, Romania
Emails: radu.ciobanu@cti.pub.ro, {ciprian.dobre, mihai.dascalu, stefan.trausan, valentin.cristea}@cs.pub.ro

***Abstract*—Selfish nodes in an opportunistic network are nodes that do not want to participate in the routing process for various reasons, such as low resources (e.g. battery, memory, CPU, network or bandwidth), fear of malicious data from unknown users, or even lack of interest in helping nodes from other communities. This may lead to messages sent in the network being delayed or even lost. Therefore, these types of nodes should be detected and avoided. Moreover, incentive mechanisms that reward nodes when they actively take part in the network (i.e. relay messages for other nodes) and punish them when they do not, should be employed where possible. In this paper, we propose a novel collaborative content and context-based selfish node detection algorithm and an incentive mechanism which aim to reduce the issues of selfish nodes in an opportunistic network. Since local information may not be sufficient to reach an informed decision, nodes running our algorithm collaborate through gossiping, for the final goal of detecting selfish nodes, later on to punish and avoid them. We compare our approach to an algorithm entitled IRONMAN and show that it behaves better in terms of network performance and detection accuracy.**

I. Introduction

The emergence of mobile devices has helped to 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 that can exchange data while in proximity. Opportunistic networks are based on the store-carry-and-forward paradigm, 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 opportunistic routing algorithms is that nodes are willing to participate in the routing process at all times, but in a real-life scenario this assumption isn't necessarily true. A node may be selfish for node A and unselfish for node B. There are several reasons why a node may be selfish. For example, it might be low on resources (battery life, memory, CPU, network or bandwidth) 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. The existence of selfish nodes in an opportunistic network might lead to messages having high delays or never being delivered at all. Thus, selfish nodes should be detected and avoided when routing. Furthermore, incentive mechanisms should reward nodes when they actively take part in the network and punish them when they do not.

In this paper, we propose a novel social-based collaborative content and context-based selfish node detection algorithm and an incentive mechanism which aim 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 our algorithm collaborate through gossiping. After informing each other with their observations, each node individually reaches a decision based on the received information. Unlike other collaborative detection algorithms such as [1] or [2], nodes do not have to rely on other nodes doing their computations. Moreover, perceived altruism values for other nodes are not binary. Instead, we use fuzzy values when analyzing a node's altruism.

Our algorithm also bases its analysis on the current context, by using social knowledge, as well as information about the device's battery. Social information is used because nodes tend to interact more and be more altruistic towards other members of their own community, while the battery status helps decide if a node was being selfish due to the fact that it was low on battery. Furthermore, the algorithm proposed in this paper is also content-based since it analyzes every message a node has sent and makes decisions based on their type. Messages that are sent to selfish nodes may end up being dropped or not delivered, and this is why a selfish node detection and incentive mechanism is necessary. Our solution provides such a mechanism that is able not only to avoid selfish nodes, but also to increase the performance of the network by doing so (in terms of various metrics such as hit rate, delivery latency, etc.). In order to provide more insight into our approach, we compare it to an existing algorithm entitled IRONMAN [2] and show that it behaves better in terms of network performance and detection accuracy.

Section II presents related work. Section III proposes a novel social-based collaborative content and context-based selfish node detection algorithm and an incentive mechanism. Section IV compares the results obtained by IRONMAN and our algorithm, while Section V presents our conclusions.

II. Related Work

This section presents notions about altruism modeling in practice and describes similar solutions for selfish node detection and incentive mechanisms in opportunistic networks.

A. 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 [3] 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 either be totally selfish or totally altruistic, and there is a given percentage of selfish nodes in the network. However, this model is not realistic, since a node isn't generally totally selfish or totally altruistic. The uniform distribution model assumes that the altruism value of the opportunistic network is uniformly distributed, while the normal distribution model uses a normal distribution, restricting and normalizing the range of values using the 5% and 95% values of the CDF [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. 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.

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. Regarding the behavior of a node when it is selfish, another node can send it messages, but it 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.

B. Selfishness Detection and Incentive Mechanisms

Although it has been shown that opportunistic networks are robust towards altruism distribution [3], detecting and avoiding selfish nodes (or making them unselfish) can lower the unnecessary loss of resources or the delays that may appear. Therefore, several methods for the detection of selfish nodes in DTNs have been proposed in the past.

The selfish node detection mechanism for Mobile Ad Hoc Networks (MANETs) and DTNs described in [1] uses a collaborative watchdog approach to detect selfish nodes and spread this information in the network. In such an approach, if one node has previously detected a selfish node, it transmits this information to encountered nodes. However, this method has the main disadvantage that it assumes that a node can either be fully altruistic or fully selfish. Therefore, the perceived state of a node can fluctuate heavily if contradictory information comes from different sources.

We propose an approach that uses fuzzy values 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 per message). Our approach is somewhat similar to [5], where gossiping is used by nodes to spread their interpretation of the monitoring level in order to have a faster detection of selfish nodes in the network. Another proposed method [6] 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. 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 [2] that 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 regular 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 trust in C is decreased. Whenever a node A receives a message from a node B which is not the source of the message, A's trust in 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 trust score is below a given threshold), it is notified, and A won't send it any messages. Moreover, it won't accept any messages from B either, so a selfish node might end up not being able to send its messages, unless it becomes altruistic. We compare our proposed solution to IRONMAN and highlight the advantages it brings. Unlike IRONMAN, our approach doesn't assume that a node is either selfish or altruistic, but it computes a fuzzy value that can be interpreted in multiple ways. Moreover, we also take advantage of social information about the nodes, but do not use it solely at the bootstrapping phase, but also during the algorithm's runtime. We have chosen to compare our solution to IRONMAN because it has been shown to have good results compared to previous existing mechanisms.

III. Detecting and Handling Selfish Nodes in Opportunistic Networks

In our algorithm, each node has a unique ID that can be used to identify it within the entire network. Aside from it, 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 others [7], [8], we consider this information to be helpful in designing our selfishness detection algorithm. Furthermore, as we have shown in Section II-A, nodes tend to be less selfish towards their own communities, so having knowledge about a node's social connection might help us 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, or low battery). Therefore, a node contains information about its own community as a list of nodes it has social relationships with. 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 [9], which is an algorithm that detects an opportunistic network node's community on the fly based on the duration and number of encounters with other nodes. Also, the battery level is another contextual information that a node has access to and may use in the altruism computation process.

Each node running our algorithm also contains a data memory split into four sections. Firstly, there is the list of messages that the node has generated in the course of time. Secondly, each node has a list of messages that it stores, carries and then finally forwards (or drops, if the memory is full) for other nodes. Additionally, each node has another two sections of data memory that contain information regarding past transfers: a list of past forwards O and a list of past receives I. O 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, TTL, priority, etc.). The list of past receives I 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 of these lists are updated whenever a new data exchange takes place and they only store the most recent information.

When two nodes A and B running our algorithm meet, they perform a series of steps. Firstly, each node computes an altruism value towards the other node, as specified in Section II-A and, based on that value, decides if it will help 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. Since the sizes of I and O are limited, only the most recent transfers are kept (regardless of whether they belonged to the current node, or to a different one). 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.

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 thresholds, the communication continues and the desired opportunistic routing or dissemination 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 send it messages for routing and doesn't accept messages from B, either. 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 selfish node and the source of the message, the inter-community altruism is increased. Otherwise, the intra-community altruism value grows.

The formula for computing altruism values for a node N and a message m based on the list of past forwards O and on the list of past receives I is the following:

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

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 the case. Thus, the 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.

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 content of every message in a node's data memory, our proposed selfish detection algorithm is also *content-based*.

IV. Evaluation

This section presents an evaluation of our selfish detection and incentive algorithm in various situations. We compare its results with the ones obtained when running IRONMAN [2].

A. Test Setup and Scenarios

All tests were performed on the MobEmu emulator [10] which parses human mobility traces or runs a mobility model simulator, and then applies an opportunistic routing and selfish node detection algorithm. We used one mobility trace (UPB 2012) and one synthetic mobility model (HCMM). UPB 2012 [11] is one of a series of two mobility traces taken in an academic environment at the University Politehnica of Bucharest (the other one being UPB 2011 [8]), where the participants were students and teachers at the faculty. It includes data collected for a period of 64 days by 66 participants, during faculty working hours (8 AM - 8 PM). We also tested our algorithm using HCMM [12], a synthetic mobility model that tries to replicate the behavior of nodes in an opportunistic network. It assumes that nodes are driven not only by the social relationships between them, but also by the attractions of physical locations. HCMM is based on the caveman model and assumes that each node is attracted to its home cell according to the social attraction exerted on that node by all nodes that are part of its community. We chose the UPB 2012 trace because we were interested in applying our algorithm to an academic environment, where contacts happen often, due to the large number of participants in a relatively small space. HCMM was deployed in order to simulate a similar environment, in case the collected trace was incomplete or non-representative of the actual academic environment. Therefore, the HCMM simulations will have a lot more contacts than the trace.

The opportunistic routing algorithm we used in our tests was Spray-and-Wait [13], 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. However, 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, in order to limit the overhead of the gossiping stage. An entry in one of these two lists may contain up to 10 fields. Assuming they are represented as integers, it means that an entry in O or I takes up about 80 bytes. Thus, the maximum size of a history list is approximately 8 kilobytes. Compared to the sizes of regular messages that are being sent between nodes (which can reach an order of tens of megabytes), the history lists are insignificant. Since the speed of Bluetooth 2.0 is around 3 megabits per second, this means that an entire history list can be sent in 20 milliseconds. Since it has been shown that the average contact duration in an academic environment can be as much as 30 minutes [8], it is clear that the overhead of exchanging history lists is insignificant.

For every test case, we tried to model the generation of messages so as to resemble a real-life environment as closely as possible. 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 particular trace.

For the IRONMAN implementation, we used the description in [2] and the parameters 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. 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. When a node is informed by another that it is considered selfish, it increases both these values by 0.1 (this value was chosen because we have observed empirically that it is the most suitable in terms of results obtained). This is done similarly for our algorithm, except that only one of the two altruism values is increased by 0.1, depending on the relationship between the current and the encountered nodes (i.e. if they are in the same community, the intra-community value is increased). 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 this discrepancy in order to best model real-life scenarios, whereas the new mean values for the altruism distributions were obtained as deviations of trust in terms of incremental steps of 0.1 from the standard mean of 0.5.

Taking into consideration the previously described scenarios, we performed several simulations and analyzed various metrics that highlight the benefits brought by our proposed approach.

1) Hit rate, latency, delivery cost, and hop count: For all sets of tests, we looked at four performance metrics very important in opportunistic networks and how they are affected by selfishness. The *hit rate* is the ratio between messages delivered successfully and the total number of generated messages. The *delivery latency* is defined as the time between the generation of a message and its 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 network congestion. 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. The tests were performed on UPB 2012 and on HCMM. We ran the default Spray-and-Wait algorithm with and without any selfish nodes (we called these the "default" and "selfish" cases), after which we applied IRONMAN and our proposed algorithm. For the UPB 2012 trace, there were 53 active devices in the network, with 30 maximum message copies allowed for the Spray-and-Wait algorithm, and a threshold of 1 for our algorithm. For the test scenario using 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 scenario was three days, and we used the HCMM community grouping to create the social network used for routing decisions. The maximum number of allowed Spray-and-Wait copies was set to 16 and our altruism threshold was set to 1.

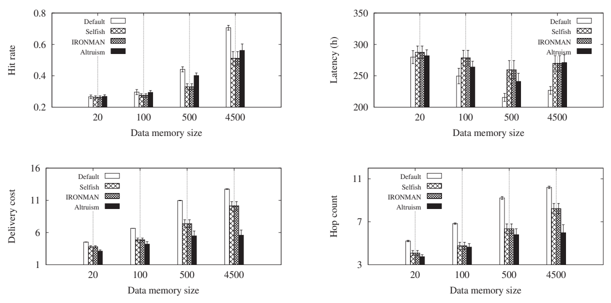


Fig. 1. Selfish node detection results for the UPB 2012 trace.

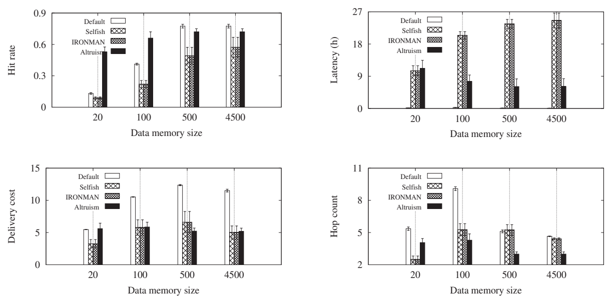


Fig. 2. Selfish node detection results for the HCMM model.

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 fuzzy 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. Community-biased detection accuracy experiments were performed on the UPB 2012 trace.

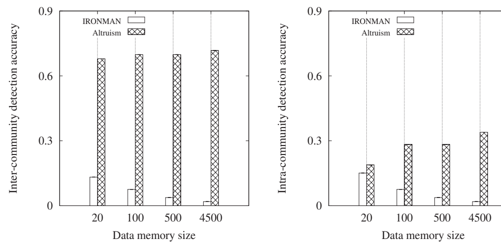
B. Results for Opportunistic Network Metrics

Firstly, it can be seen from Figure 1 that, for the UPB 2012 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 2.21%, while for a memory of 4500, it is as high as 19.51%). 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 incentive mechanism is necessary. Selfish nodes must be avoided or convinced to become unselfish, in order for the hit rate to improve. Figure 1 shows that using IRONMAN doesn't bring much improvement at all; the reason is that selfish nodes are hard to detect because their perceived altruism value by other nodes grows very quickly (i.e. nodes see then as being very altruistic). The algorithm we proposed in Section III outperforms IRONMAN for all data memory sizes and gets relatively close to the hit rate obtained when no selfish nodes are present. Furthermore, for small data memories (in this case, of 20 messages), it even yields a better hit rate than the default case. This happens because, for the default case, nodes send messages to any encountered nodes and (since the number of message copies is limited) may end up depleting them, while the nodes that receive them do not 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 doing it again. It is also important to note that the hit rate improves when increasing the size of the data memory, because older messages are deleted after a much longer time and the chance of replacing them becomes lower.

Also in Figure 1, 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 seem to improve the latency, which means that selfish nodes aren't always avoided. However, our algorithm sees an improvement in delivery latency of as much as 18 hours when compared to the selfish case. Although opportunistic networks are delay-tolerant, we believe that the latency should be decreased where possible.

The delivery cost and hop count are also affected by the introduction of selfish nodes in the opportunistic network, and this is where we achieve the most important results, as can be seen in Figure 1. Not only are the hop count and delivery cost values obtained by our algorithm lower (i.e. better) than the ones from IRONMAN, they are better than what is obtained at the default and selfish scenarios. This is due to the reason given above, that messages are forwarded only to nodes that have previously forwarded similar messages, not to any encountered node. The improvement over the default scenario may happen only because the hit rate also drops, which means that the messages that aren't 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. However, our hit rates are better than the ones from the selfish scenario, and still the hop count and delivery cost are lower for all test cases, which shows that our algorithm is very efficient in terms of network and node congestion.

Additionally, Figure 2 shows the results obtained when running on HCMM. It can be easily seen that our algorithm 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. The explanation is that we are dealing with many contacts and the copies of a node's messages get consumed quickly on 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 data memories, this no longer happens because nodes can store more messages. The latency results correspond to what we have seen earlier: having selfish nodes increases the latency, but our algorithm can help decrease it. Finally, although our



(a) Inter-community (b) Intra-community

Fig. 3. Community-biased detection accuracy.

algorithm obtains hit rates close to that of the original test case, it yields much better values for hop count and delivery cost.

C. Results for Community-Biased Detection Accuracy

Figure 3 depicts the community-biased detection accuracy for selfish nodes inside or outside the community. It can be seen that, for nodes outside the community, our algorithm has a much better accuracy than IRONMAN, the difference for a data memory of 4500 being 69.81%. This suggests that our algorithm detects more nodes that are selfish towards non-connected nodes than IRONMAN, and convinces them through incentives to become more altruistic. Our algorithm also performs better regarding intra-community nodes, but the differences are not so large. The high detection accuracy for our algorithm is obtained thanks to the fact that our analysis of encounter history is both context, as well as content-based. It is also important to observe that, for both types of metrics, the detection accuracy grows with the data memory size for our approach, while for IRONMAN it decreases.

V. Conclusions

We have presented a novel social-based collaborative content and context-based selfish node detection algorithm, including 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, on one hand, and incentive mechanisms to make selfish nodes become altruistic, on the other. It takes advantage of social 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 of a node.

In terms of validation, we have tested our algorithm on a social trace and a mobility model and compared it to IRONMAN [2]. 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 our solution sends messages in a selective manner, only to nodes that have already successfully delivered messages of that type. Furthermore, we have also shown that our solution has better detection accuracy than IRONMAN.

Acknowledgment

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.

References

[1] E. Hernández-Orallo, M. D. Serrat Olmos, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Evaluation of collaborative selfish node detection in manets and dtns," in *Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, ser. MSWiM '12. New York, NY, USA: ACM, 2012, pp. 159–166. [Online]. Available: http://doi.acm.org/10.1145/2387238.2387266

[2] G. Bigwood and T. Henderson, "Ironman: Using social networks to add incentives and reputation to opportunistic networks," in *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*, oct. 2011, pp. 65 –72.

[3] P. Hui, K. Xu, V. Li, J. Crowcroft, V. Latora, and P. Lio, "Selfishness, altruism and message spreading in mobile social networks," in *INFOCOM Workshops 2009, IEEE*, april 2009, pp. 1–6.

[4] K. Xu, P. Hui, V. O. K.Li, J. Crowcroft, V. Latora, and P. Lio, "Impact of altruism on opportunistic communications," in *Proceedings of the first international conference on Ubiquitous and future networks*, ser. ICUFN'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 153–158. [Online]. Available: http://dl.acm.org/citation.cfm?id=1671729.1671759

[5] A. Lavinia, C. Dobre, F. Pop, and V. Cristea, "A failure detection system for large scale distributed systems," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on*, feb. 2010, pp. 482 –489.

[6] Q. Zhou, J. Ying, and M. Wu, "A detection method for uncooperative nodes in opportunistic networks," in *Network Infrastructure and Digital Content, 2010 2nd IEEE International Conference on*, sept. 2010, pp. 835 –838.

[7] S. Okasha, "Altruism, Group Selection and Correlated Interaction," *The British Journal for the Philosophy of Science*, vol. 56, no. 4, pp. 703–725, Dec. 2005. [Online]. Available: http://dx.doi.org/10.1093/bjps/axi143

[8] R. I. Ciobanu, C. Dobre, and 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*, ser. ADHOC-NOW'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 69–82. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31638-8_6

[9] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft, "Distributed community detection in delay tolerant networks," in *Proc. of 2nd ACM/IEEE inter. workshop on Mobility in the evolving internet architecture*, ser. MobiArch '07. New York, NY, USA: ACM, 2007, pp. 7:1–7:8. [Online]. Available: http://doi.acm.org/10.1145/1366919.1366929

[10] R. Ciobanu, C. Dobre, V. Cristea, and D. Al-Jumeily, "Social aspects for opportunistic communication," in *Parallel and Distributed Computing (ISPDC), 2012 11th International Symposium on*, june 2012, pp. 251 –258.

[11] R.-C. Marin, C. Dobre, and 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. [Online]. Available: http://dx.doi.org/10.1109/EIDWT.2012.29

[12] C. Boldrini and A. Passarella, "HCMM: Modelling spatial and temporal properties of human mobility driven by users' social relationships," *Comput. Commun.*, vol. 33, pp. 1056–1074, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2010.01.013

[13] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143