Reaching Consensus in Opportunistic Networks

Radu Drăgan*, Radu-Ioan Ciobanu*, Ciprian Dobre*†
*Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Romania
†National Institute for Research and Development in Informatics, Bucharest, Romania
radu.dragan@stud.acs.upb.ro, radu.ciobanu@cs.pub.ro, ciprian.dobre@cs.pub.ro

***Abstract*—In large systems with many sensors and devices that collect information, the infrastructure costs can reach extremely high values, since all the nodes in the network need to be connected to a server so they can upload their data and have them processed. However, in order to reduce such costs, the sensors and devices can communicate with each other, exchanging data when in close proximity. Thus, an opportunistic network (ON) is created, where devices follow a store-carry-forward paradigm, deciding upon each contact whether the encountered node is the next hop for the data they are carrying. Besides their many advantages, opportunistic networks present several challenges, one of the most disputed being the achievement of consensus between the nodes of the network. The lack of a central entity which could gather information and validate messages leaves this task to the goodwill and collaboration between nodes. Consensus is required in order for the nodes to depend on and trust the data that they receive in the network.**

In this paper, we address the problem of reaching consensus in opportunistic networks. We propose an algorithm for consensus in ONs and show that it is able to correctly select several leaders for each community of devices, which then collaborate between each other to reach consensus. Furthermore, we present two scenarios where opportunistic networks can be employed and consensus is required.

I. Introduction

Since mobile devices can be found practically everywhere nowadays, many interesting applications for mobile networks have been proposed in recent years, leading to advances in the area of crowdsensing. This paradigm is at the basis of some important technologies. However, it is very important to observe that, in certain scenarios and applications (such as smart cities, ambient assisted living, etc.), crowd-based systems cannot function without coordination, which is responsible for leader election, group membership, service discovery, and many other tasks, grouped under the name of "distributed consensus". Finding consensus in a fault-free distributed system is not so difficult to accomplish, but this is not the case for fault-tolerant systems.

In this paper, we deal with one such type of fault-tolerant systems, namely opportunistic networks, composed of mobile devices that are only able to communicate between each other using close-range protocols (such as Wi-Fi Direct or Bluetooth). In an opportunistic network (ON), there is no infrastructure, and nodes communicate with each other without having any knowledge of the overall topology. Furthermore, since ONs are based on the mobility of their composing nodes, no stable paths exist between nodes that wish to communicate. The routing is probabilistic, and various heuristics need to be employed in order to increase the hit rate of the network.

In such networks, since no central governing entity exists, the nodes need to reach their decisions by themselves, through collaboration and altruism. For this reason, mechanisms for ensuring that the data being sent and received are correct need to be employed, so nodes must be able to correctly decide together which information is correct and which is not. Several nodes might generate data regarding the same event, but a part of them might have communication errors or other problems that corrupt the data. If we assume that there is no need for consensus, then the first version of some data that is generated would be the one used by most of the nodes. In case the data is incorrect (or has been tampered with), it will be used as is. Thus, in this paper we focus on the problem of distributed consensus in opportunistic networks. We propose a solution for electing leaders in social communities, and then employ those leaders for obtaining consensus. Through simulations, we show that our algorithm is able to obtain consensus with a good probability and in a timely fashion.

The rest of the paper is structured as follows. In Sect. II, we present related work in the area of consensus in distributed systems, and opportunistic networks in particular. In Sect. III, we propose an algorithm which addresses consensus in ONs, and then we show experimental results in Sect. IV. In Sect. V, we present several scenarios where consensus is needed when employing opportunistic networks, and finally we present our conclusions and future work in Sect. VI.

II. Related Work

Paxos [1] is one of the most remarkable algorithms in the field of finding consensus in fault-tolerant systems. Although it is known as very difficult to comprehend, the Paxos algorithm was adopted for implementing well-known systems like the lock service of Google File System (GFS), named Google Chubby [2], as well as Apache ZooKeeper [3]. Furthermore, the Raft algorithm [4] adopted Paxos in order to provide a managing replicated log system. Since Paxos has always been considered hard to follow, Raft's main guiding purpose was understandability. The majority of these algorithms do not take into account one sensitive issue. They accept node failure (since it can often happen for a component to crash, thus making the entire node unavailable), yet the algorithms do not consider link failures.

The Heard-Of model [5] aims to consider both kinds of failures in finding consensus, being the starting point for some algorithms intended for opportunistic networks. In one of the algorithms proposed in the above-mentioned paper, namely

the One Third Rule, the consensus computation evolves in rounds. A group of nodes is formed at each round, each node first disseminating its contribution, then waiting to receive the contributions of others. When a specific number of contributions is received, the node analyzes all the information received and sets its contribution to the most frequent value met. Then, at the next round, the node disseminates the newly-computed contribution. If the node receives a specific number of contributions with the same value, it makes a decision and then disseminates this decision. When the other nodes receive the decision, they comply with it and reach consensus. The authors claim that consensus can even be resolved in one round, if two thirds of the nodes contribute with the same value. The algorithm was adopted by the authors in [6], where they implemented the algorithm in a system installed on a small fleet of mobile devices belonging to an opportunistic network, demonstrating that the algorithm behaves accurately in a real-life scenario.

A. Consensus in Opportunistic Networks

Although the One Third Rule is an algorithm that, on paper, overcomes some of the challenges of opportunistic networks, there are some limitations regarding its behavior on a larger network. Firstly, in an opportunistic network, nodes are highly mobile: they can enter the network, exit it, or move around at any time. Most of the time, a link between two nodes can be used only for a few seconds. This way, it is really hard to keep communities based on their location (which is something that we try to avoid in our proposed solution). The algorithm offers a failure tolerance mechanism which demands that just one node receives two thirds of the contributions from other nodes, in order to move to the next round. However, it can happen that more than one third of the nodes exit the community, thus rendering it impossible to move forward with the algorithm. Furthermore, being a round-based algorithm usually means that multiple rounds are required in order for the nodes to reach consensus. This approach might ensure consensus with a higher probability (under favorable circumstances), but the time needed for this to happen can be really high. In an opportunistic network scenario, where we generally need information validated as soon as possible, we consider that this approach is not feasible, especially since high delays might be introduced by disconnections.

Another approach for reaching consensus is described in [7], where a 4-phase algorithm is presented, which considers that a leader is already elected, a phase that is vital for the behavior of the algorithm. The election of a single leader in an opportunistic network might not be optimal, considering that the network may be sparse and that some nodes may never have any path to the destination [8]. In this regard, dividing the network into communities is more effective, in which case a leader should be elected for each community (which is a behavior taken by many opportunistic solutions, such as BUBBLE Rap [9]). This way, the leader election can be performed rapidly, thus not affecting the performance of the algorithm. In the first round of the algorithm, all nodes of a community will send their contributions to the leader. When the leader receives proposals from the majority of nodes, it makes a decision based on the values of the proposals. The second round implies the leader sending its decision to all the nodes of the community. In the third round, the nodes receive the decision and then they send an acknowledgment to the leader for receiving the decision. When the leader receives a majority of acknowledgments, it validates the decision and disseminates it to all the nodes of the community. Then, all nodes comply with the decision.

Besides having a fixed number of rounds, the algorithm has some drawbacks. None of the nodes can be considered reliable, since they might exit the community (or even the network) at any point in time. If the leader exits during a consensus session, the algorithm has to start over and elect another leader. Also, having only one leader (even per community) is not always a good choice. Nodes might be malicious, and, if the leader happens to be so, the whole algorithm would be discredited. This makes the leader the single point of failure, issue that we want to overcome in our algorithm.

Our approach is based on decisions made by leaders. Still, we do not restrict the solution to finding only one leader (not even per community). Instead, there may be multiple leaders, each of them making and disseminating decisions. Furthermore, our solution is not round-based, since we consider round-based solutions to have several drawbacks. In systems like opportunistic networks, where messages could take long time before arriving to their destinations, a round might be delayed for long periods. During this time, the structure of the network can change massively, which can badly impact the behavior of the algorithm. The process of electing leaders is further explained in Sect. III.

III. Obtaining Consensus in Opportunistic Networks

A. Leader Election

We consider that the most important attributes of a leader should be the degree of centrality and the trust of other nodes in the leader. A high centrality means that the node has a lot of encounters with other peers, which implies that the leader can gather a lot of information and make the right decision. An informed decision has a high chance to be the right one, even in the presence of malicious nodes (and, the more nodes participate in the decision process, the better will it be, since the influence of malicious nodes that might even work in collaboration will be reduced). The impact of malicious nodes would be smaller if the leader could gather information from more peers.

The trust other nodes have in the leader also has its relevance. If a node is malicious, the other nodes will gossip and will, eventually, find out about it. Then, they can exclude the respective node from their decisions. The malicious nodes should also not be taken into consideration when electing the leader. A malicious leader would disseminate false information in the network. We will demonstrate in Section IV that, even

if some leaders are malicious, nodes could still make the right decision when evaluating a message.

In our previous work [10], we demonstrated that a state where only one leader is elected for the entire network is difficult to reach. Each node will consider its leader the node with the highest product between the centrality of the candidate and the trust the current node has in the candidate. We chose these two metrics because we believe they are the most relevant for our purpose. For computing the centrality, we used the online method presented in the BUBBLE Rap algorithm [9]. The centrality is a measure of how popular a node is, indicating how likely it is for the current node to have many encounters in the future. The trust value is computed similar to SPRINT-SELF [11], being a reflection of how benevolent the node is to actively forward messages to the interested nodes.

The solution we used was to split the network in communities which contain nodes that are close to each other and can easily communicate. We proposed two approaches for electing the leader. The first one, called Direct Election, implies that each node advertises its centrality. Then, all the nodes compute a score for each received candidacy and elect the node with the highest score as the leader. The score used in this approach is $score = w_t \times v_t + w_c \times v_c + w_p \times v_p + w_l \times v_l$, where w_t, w_c, w_p, and w_l are weight values for trust, centrality, probability of meeting the node, and candidacy latency, while v_t, v_c, v_p, and v_l are the values for the preceding metrics. We previously discussed the reasons for employing the first two metrics. In terms of the probability of meeting (which is computed based on the history of contacts and social information, as described in SPRINT [12]), it is important for the current node to be connected to its leader as much as possible in order to be up-to-date with the latest decisions. The candidacy's latency is also a measure of how close the leader is to the node.

The second approach implies that, firstly, communities are formed. Then, all the nodes inside a community need to agree upon a single leader. Communities are formed based on the time nodes spend in contact. Each new node added to a community has to be approved by a given fraction of the composing nodes (dynamically set in order to meet the community needs). Furthermore, in order for a new node added to a community to become the leader, over half of the nodes have to consider it as such, based on the computed leader score. Moreover, the nodes that travel far from their community will be removed so that they will not obstruct the decision-making process. The leader score is similar to the first approach, except there is no candidacy latency.

B. Solving consensus

Once the leaders have been elected, we move towards the final goal of our algorithm: solving consensus. At certain times, the nodes generate messages. These messages are not taken into consideration until they are validated by a leader. In order to do that, a node sends a message toward its leader, using the SPRINT algorithm [12]. We chose SPRINT to accomplish this task because it offers good hit rate values (around 90%), taking into account that it is critical for as many messages as possible to reach the leaders so that the decisions are informed. In addition, it is important for the nodes to elect as leaders nodes located not too far away (in regard to the number of hops). This way, a message from the node to its leader and the other way around can arrive in a timely fashion.

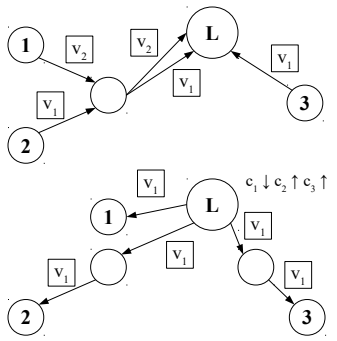


Fig. 1. Consensus algorithm

Thus, when receiving a message intended for itself, a node can know that it is considered leader by the source of the message. Upon receiving each message, the leader recalculates its decision, based on all the messages received regarding a specific data item. If the number of messages received regarding a certain event does not exceed a certain threshold (set based on the scenario where the algorithm is deployed), the decision will not be taken. For the scenarios we present in Sect. IV, we set this threshold to 3, because it is the minimum number that could form a majority and there are some cases where the communities are not larger than 3 nodes.

Figure 1 shows the behavior of three nodes and a leader in our proposed solution. In the first step, the three nodes (marked as 1, 2 and 3) send versions of the same message (observation) to the leader (marked as L) for confirmation. Nodes 2 and 3 send the same version, whereas node 1 sends a different version, but they each do so through opportunistic communication using SPRINT (nodes 1 and 2 using an intermediary node, whereas node 3 directly, since it is in range of the leader). Until all three versions of the same message arrive, the leader does not perform any action. However, once it has received from all three nodes, it can make a decision and select v_1 as the correct version, as shown in the second step from Fig. 1. There, the leader increases the confidence in nodes 2 and 3, while decreasing it for node 1. At the same time, it sends each of the three nodes the correct version.

The existence of malicious nodes is one of the aspects that we have to consider. This is why we developed a malevolence mechanism whose main purpose is the detection and isolation of non-collaborative malicious nodes. For each new contribution received, the leader will assign it a trust level which corresponds to the trust the leader has in the node that generated the contribution. The trust value mentioned here is related to the malevolence mechanism under discussion right now and not to the selfish mechanism mentioned before.

For each value received for the current event, the one with

the highest sum of the trust values will be chosen as the correct one. We also computed the confidence of the decision, which will be the ratio between the trust sum for the correct value and the trust sum for all the values. Afterwards, the nodes that proposed the correct values will be assigned a trust value for the current event equal to the confidence level. Nodes that proposed false values will be assigned a trust value of $1 - confidence\ level$. The leader will keep a list of trust values for a certain node corresponding to all events the node sent contributions for to the leader. When receiving a new contribution, the leader will consider the average of these trust values. Therefore, a node will have its trust value increased by each correct contribution and decreased for each false contribution.

In addition, we set a threshold for the confidence level, so that decisions which do not exceed the threshold will not be disseminated. This way, the leader will wait until it is sure enough regarding the correct value of the event before making the decision. We set this threshold to 60% so that it will be enough for 2 contributions with the same value out of 3, but also 3 contributions with the same value out of 5.

After making the decision, the leader disseminates the decisions to all the nodes that contributed. While waiting for the decision from its leader, a node can also consider decisions made by other leaders regarding the same event. This way, even if a small portion of the leaders are corrupted, most of the nodes should still be capable of finding out the correct value of an event. Having multiple leaders also combats the single point of failure issue of other algorithms. Even if a leader would crash, the nodes would receive decisions from other leaders, so the algorithm would not be affected as badly.

When a node changes its leader, all the contributions sent and not answered will be sent to the new leader. The former leader might have left the network or it is far away from the current node, so there is no point in waiting for its response.

IV. Experimental Results

A. Leader Election

For assessing the algorithm, we used one mobility trace and one synthetic trace, which we simulated using MobEmu[1]. MobEmu is an opportunistic network simulator that allows the user to implement the behavior of a node upon an opportunistic contract. The mobility trace, called Sigcomm [13] took place at a conference in Barcelona and implicated 76 users during 4 days. One important advantage of this trace is the existence of social network information about the users, which enhances the prediction capabilities of our algorithm.

The synthetic trace adopts HCMM (the Home-Cell Community-based Mobility Model) [14], which is based on the caveman model and aims to replicate the movement of the nodes inside the network as accurately as possible. We created a scenario where we attempted to simulate the nodes of an AAL facility, as described in Sect. V. The minimum number of leaders existent in the network was 11 for Sigcomm and 15 for HCMM, so the aim of finding only one leader for the entire network was far from being achieved. In addition, it is hard to believe that one node, however central, would succeed at being responsive to all the nodes of the network, which might get fairly sparse from time to time (especially taking into account the high degree of mobility of nodes and the many disconnections between various partitions of the network).

For the two approaches presented above, we computed the response times of the leaders, which illustrates their responsiveness. The average response times are around 0.3 hours for the HCMM trace, and between 10 and 13 hours for the Sigcomm trace. These results pave the way for more practical applications for opportunistic networks, like consensus, which will be discussed in the following subsection.

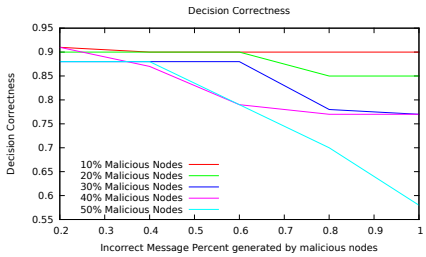
B. Obtaining Consensus

We applied the algorithm presented in Sect. III after electing the leaders and we assessed the algorithm using the two previously mentioned traces: Sigcomm and HCMM. At certain moments of time, all nodes will generate messages regarding a certain event, with different values. We only considered two possible values (true and false) when testing the algorithm, but there is no constraint regarding the number of the possible values of the messages. Then, based on the values received, the leaders have to decide which is the correct value of the message. This way, we computed the percentage of correct decisions made by the leaders, an ability which we consider the most important when it comes to consensus algorithms. Also, we desired to assess the impact that malicious nodes would have on the network, so we injected a specific percentage of malicious nodes that generate false messages.

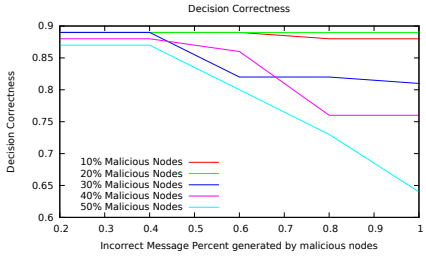
For the malicious nodes percentages, we used the following values: 10%, 20%, 30%, 40%, 50%. We also considered that malicious nodes generate false messages with a given probability, because they may not want to be easily discovered by other nodes. We varied the false messages probability using the following values: 20%, 40%, 60%, 80%, 100%, so that we would form a full picture regarding the influence malicious nodes can have over the network.

Studying the results presented in Fig. 2, we can observe that, by increasing the percent of false messages generated by a malicious node, we will obtain a decrease in the decision correctness. This is the expected behavior, considering that, the more false messages are generated, the higher the probability that some leaders will be tricked by the malicious nodes. There are some abnormalities, like the one presented in Fig. 2(b), when 30% malicious nodes produce more damage than 40% malicious nodes, having, in both cases, a 60% percentage of incorrect messages generated. The results might be influenced, as well, by other factors, like the positioning of the malicious nodes around the leaders and the timing of the incorrect messages generated (the malicious nodes might be the first to send their contributions and have enough of them to trick some leaders from the beginning, having then a greater influence on these leaders than the other nodes). An important thing to notice here is that, if 10% of the nodes are malicious, they do

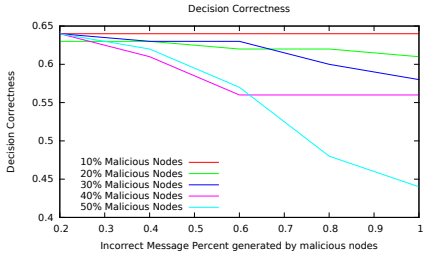
[1]https://github.com/raduciobanu/mobemu



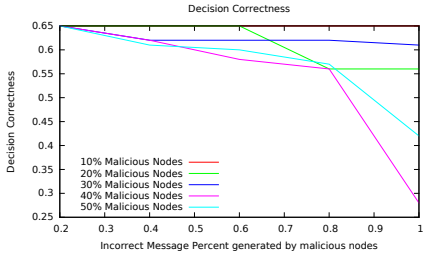
(a) HCMM Community



(b) HCMM Direct



(c) Sigcomm Community

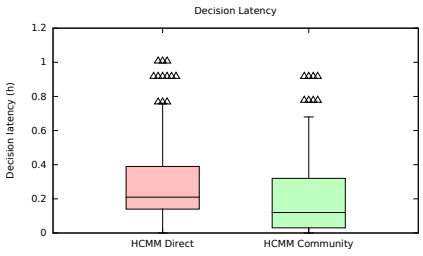


(d) Sigcomm Direct

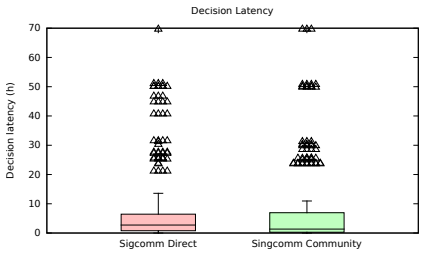
Fig. 2. Correctness rates obtained through consensus.

not influence the network too much (around 90% correctness), and that the smallest values are around 60% for HCMM and around 43% for Sigcomm, for 50% malicious nodes generating only false messages.

Another metric measured in our tests is the average decision latency, because a decision delivered too late might not be useful anymore, so we should focus on delivering the decisions as soon as possible, while keeping the reasoning informed. The results illustrated in Fig. 3 show an average detection latency of around 0.2 hours (12 minutes) for HCMM and around 2-3 hours for Sigcomm, which are good values when comparing to the contact and inter-contact times for each trace. These results give us hope towards the numerous applications that the consensus algorithm could be employed for in an ON.



(a) HCMM



(b) Sigcomm

Fig. 3. Decision latencies.

V. Discussion

There are various situation where opportunistic networks can be (or have already been) employed, including (but not limited to) disaster management, smart cities, sensor networks, floating content, mobile advertising, crowd management, Internet access in limited conditions, wildlife tracking, ambient assisted living, distributed social networks, data offloading, or mobile cloud computing. In this section, we would like to discuss two of the use cases presented above where consensus is required when employing ONs.

In an AAL facility, there are a multitude of sensors that collect data regarding the patients, and then send them to the Cloud for processing. However, it would be expensive if all sensors were equipped with Wi-Fi or mobile broadband capabilities, so their data are sent opportunistically (through close-range protocols) to mobile nodes which are carried by the residents or by the nurses and doctors. The mobile nodes receive the data collected by the sensors when they are in close range, and then, through mobility, move them further towards some access points that have the possibility of connecting to the facility's server or to the Cloud. In such a scenario, it is vital that the nodes decide whether the information that is being eventually uploaded to the processing service is actually correct. Several carriers might collect the same data from some sensors, but a part of them might have communication errors or other problems that corrupt the data. Another situation where consensus is required in an AAL scenario occurs when there are multiple sensors that collect similar data about the same resident of the AAL facility. For example, there are cameras and motion sensors in several rooms, each of them recording information about the residents that are in the respective rooms at a moment in time. Naturally, the sensors should collect similar data for the same resident, but there

may appear situations where, due to various reasons (such as malfunctioning, wrong configuration, etc.), this is not the case. In this situation, the network needs to be able to correlate information from several sensors, and select the most accurate one. The same situation also arises when internal sensors collect some data leading to a certain conclusion, whereas external sensors would reach a different conclusion.

In a smart city scenario, opportunistic communication can be used for creating a network of vehicles that exchange information about traffic events through a mobile application. Using this approach, the mobile phones would not need any Internet connection, but will instead communicate through opportunistic networking, taking advantage of the lower battery consumption demands of Wi-Fi compared to 3G/4G, while at the same time offering higher transmission speeds. The nodes would be able to generate information regarding traffic events such as gridlocks, accidents and so on, somewhat similarly to Waze's functionality. However, unlike Waze, the app in this scenario will not need to connect to a server in order to disseminate the information. However, how can we know that this information is correct? There might be some nodes who could benefit from disseminating false information, and this is why we have to isolate these nodes and validate the generated data. Still, this is non-trivial, considering that we deal with an ON where there is no central entity that can generate, aggregate and confirm the information.

VI. Conclusions and Future Work

In this paper, we have addressed the problem of obtaining consensus in opportunistic networks. We have proposed a solution based on selecting multiple leaders for the various communities formed between opportunistic nodes, and having those leaders work together in order to reach consensus. This is a very important topic in mobile networking, because the network needs to be certain that the data is correct, especially if we are dealing with critical information. We have presented an analysis of leader election (the first step required for implementing a consensus solution), and then showed experimental results for the consensus algorithm. In the future, we wish to test the implementation of the algorithm in a real-life scenario, using an Android application. We also plan on developing a simulation model and testing more metrics, while comparing our solution to other implementations (since these do not necessarily exist for opportunistic networks, we will have to adapt the existing algorithms).

Given the results obtained so far, we strongly believe that our algorithm will behave well in real-life opportunistic network situations, and we look forward to validating these assertions by running the algorithm and achieving the expected results. The consensus problem is one of the most debated issues of ONs, given the challenges presented by this kind of network, and we are optimistic about our algorithm being, if not a solution for this problem, at least a starting point for further development in this area.

Acknowledgment

The research presented in this paper is supported by University Politehnica of Bucharest, through the "Excellence Research Grants" Program, UPB - GEX 2017. Identifier: UPB-GEX2017, Ctr. No. AU 11.17.02/2017, and by project Traffic and Data Offloading in Mobile Networks – TTOff, H2020 as part of Measuring Mobile Broadband Networks in Europe.

References

[1] A. Ailijiang, A. Charapko, and M. Demirbas, "Consensus in the cloud: Paxos systems demystified," in *2016 25th International Conference on Computer Communication and Networks*, ser. ICCCN. IEEE, 2016.

[2] M. Burrows, "The chubby lock service for loosely-coupled distributed systems," in *Proc. of the 7th Symposium on Operating Systems Design and Implementation*, ser. OSDI '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 335–350. [Online]. Available: http://dl.acm.org/citation.cfm?id=1298455.1298487

[3] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems," in *Proc. of the 2010 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIXATC'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 11–11. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855840.1855851

[4] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIX ATC'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 305–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=2643634.2643666

[5] B. Charron-Bost and A. Schiper, "The Heard-Of model: computing in distributed systems with benign faults," *Distributed Computing*, vol. 22, no. 1, pp. 49–71, 2009. [Online]. Available: http://dx.doi.org/10.1007/s00446-009-0084-6

[6] A. Benchi, P. Launay, and F. Guidec, "Solving consensus in opportunistic networks," in *Proc. of the 2015 International Conference on Distributed Computing and Networking*, ser. ICDCN '15. New York, NY, USA: ACM, 2015, pp. 1:1–1:10. [Online]. Available: http://doi.acm.org/10.1145/2684464.2684479

[7] F. Borran, R. Prakash, and A. Schiper, "Extending paxos/lastvoting with an adequate communication layer for wireless ad hoc networks," in *Proc. of the 2008 Symposium on Reliable Distributed Systems*, ser. SRDS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 227–236. [Online]. Available: http://dx.doi.org/10.1109/SRDS.2008.21

[8] R. Ciobanu, C. Dobre, V. Cristea, F. Pop, and F. Xhafa, "SPRINT-SELF: social-based routing and selfish node detection in opportunistic networks," *Mobile Information Systems*, vol. 2015, pp. 596 204:1–596 204:12, 2015. [Online]. Available: https://doi.org/10.1155/2015/596204

[9] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: social-based forwarding in delay tolerant networks," in *Proc. of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '08. New York, USA: ACM, 2008, pp. 241–250. [Online]. Available: http://doi.acm.org/10.1145/1374618.1374652

[10] R. Dragan, R.-I. Ciobanu, and C. Dobre, "Leader election in opportunistic networks," presented at 16th International Symposium on Parallel and Distributed Computing (ISPDC), 2017.

[11] R. I. Ciobanu, C. Dobre, V. Cristea, F. Pop, and F. Xhafa, "Sprint-self: Social-based routing and selfish node detection in opportunistic networks," *Mobile Information Systems*, vol. 2015, 2015.

[12] R. I. Ciobanu, C. Dobre, and V. Cristea, "SPRINT: Social prediction-based opportunistic routing," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. IEEE, Jun. 2013, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/wowmom.2013.6583442

[13] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot, "Mobiclique: middleware for mobile social networking," in *Proc. of the 2nd ACM workshop on Online social networks*. ACM, 2009, pp. 49–54.

[14] C. Boldrini and A. Passarella, "Hcmm: Modelling spatial and temporal properties of human mobility driven by users social relationships," *Computer Communications*, vol. 33, no. 9, pp. 1056–1074, 2010.