Solving Consensus in Opportunistic Networks

Abdulkader Benchi, Pascale Launay, Frédéric Guidec

Solving Consensus in Opportunistic Networks

Abdulkader Benchi
IRISA Laboratory
Université de Bretagne-Sud
France
abdulkader.benchi@univ-ubs.fr

Pascale Launay
IRISA Laboratory
Université de Bretagne-Sud
France
pascale.launay@univ-ubs.fr

Frédéric Guidec
IRISA Laboratory
Université de Bretagne-Sud
France
frederic.guidec@univ-ubs.fr

ABSTRACT

Opportunistic networks are partially connected wireless ad hoc networks, in which pairwise unpredicted transient contacts between mobile devices are the only opportunities for these devices to exchange information or services. Ensuring the coordination of multiple parts of a distributed application in such conditions is a challenge. This paper presents a system that can solve consensus problems in an opportunistic network. This system combines an implementation of the One-Third Rule (OTR) algorithm with a communication layer that supports network-wide, content-driven message dissemination based on controlled epidemic routing. Experimental results obtained with a small flotilla of smartphones are also presented, that validate the system and demonstrate that consensus can be solved effectively in an opportunistic network.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*distributed networks, wireless communication*; C.4 [**Performance of Systems**]: Fault tolerance, Measurement techniques, Reliability, availability, and serviceability

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Consensus, opportunistic networking, opportunistic computing

1. INTRODUCTION

Opportunistic networks constitute a category of mobile ad hoc networks (MANETs) in which the sparse or irregular distribution of mobile devices (or *nodes*) yield frequent link disruptions and network partitions [6]. While a MANET is often represented as a *time-varying graph* (TVG [9]) that remains connected at any time, an opportunistic network must rather be depicted as a non-connected TVG, as shown in Figure 1: mobile nodes only come into contact with each other every now and then, depending on the mobility patterns of their respective carriers (which can be human beings, but also vehicles, animals, etc.). When a transient contact occurs between two nodes, this contact has not been planned in advance so it must be exploited opportunistically by these two nodes to exchange information or services [32].

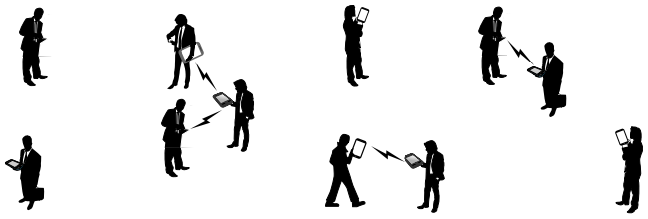

Figure 1: Example of an opportunistic network composed of mobile nodes carried by human beings

Relying exclusively on unpredicted pairwise contacts between mobile nodes does not imply that long-distance communication is not feasible in an opportunistic network. Indeed, each mobile node can serve as a "data mule" for messages that propagate in the network, storing messages in a local cache, and carrying them for a while before they can be forwarded to other nodes. This "store, carry and forward" principle is the foundation principle of Delay/Disruption-Tolerant Networking (DTN [17]): messages can indeed propagate in a partially or intermittently connected environment, but their delivery is not guaranteed and can be delayed by minutes, or even hours or days, as it depends on the wanderings of benevolent mobile carriers.

Opportunistic computing has recently been proposed as a new computing paradigm to build upon opportunistic networking. The general idea is to develop a framework that enables collaborative computing in environments where long disconnections and network partitions are the rule [14]. Several middleware systems have already been proposed to ease the development of distributed applications that are meant to run in opportunistic networks: a middleware system that supports asynchronous messaging based on opportunistic communication is proposed in [3], one that supports the tuple-space model in opportunistic networks is described in [4], and systems that support service provisioning and delivery in opportunistic networks are presented in [15] and [28].

Such middleware systems constitute a first step toward application development targeting opportunistic networks. Yet, in complex distributed applications, multiple processes running on different mobile nodes must be able to agree on a common course of actions. Consensus problems have been studied extensively in the literature, with various system models. Each system model makes specific assumptions

about the underlying communication model (synchronous or asynchronous, based on either reliable or faulty links) and about the processes themselves (reliable, susceptible to crash or to exhibit Byzantine failures). A large majority of papers addressing consensus adopt system models that suit traditional wired networks, such as the Internet. To date only a few papers have considered the consensus problem in wireless ad hoc networks and, to the best of our knowledge, no paper has considered the consensus problem in opportunistic networks yet.

As observed in [7] the models for wired networks are often strongly biased towards node failures to the detriment of link failures. Yet, mobile ad hoc networks –including opportunistic networks– require a model that admits both transient process and link faults, and that considers such faults as benign failures. The Heard-Of (HO) model [13] meets these requirements. This model does not distinguish faulty processes from faulty links, as it focuses on transmission faults (effects) without accounting for the faulty components (causes) [13].

Indeed, in an opportunistic network such as that shown in Figure 1, a wireless link between two neighbor nodes is inherently transient, so when this link disappears (for example because both nodes have moved away from each other) this should not be considered as a "fault": this is the expected consequence of both mobility and limited transmission range in an opportunistic network. Similarly, mobile nodes in an opportunistic network often run on batteries, so a common strategy to preserve their power budget is to turn them off –or put them in suspend mode, or disable their wireless transceiver– frequently. When a node suddenly "disappears" from the network, this should not always be considered as a "fault" (not even a benign one), because this is a perfectly legitimate behavior for this kind of node in this kind of environment.

As mentioned above, delivering messages to remote nodes in an opportunistic network is not guaranteed, for it depends on the wanderings of carriers whose mobility patterns are neither planned nor controlled. Messages can therefore get lost before reaching their destination(s). Two of the consensus algorithms defined in [13] for the HO model can easily tolerate message loss: the Paxos/LastVoting (P/LV) algorithm, and the One-Third Rule (OTR) algorithm. An implementation of the P/LV algorithm for mobile ad hoc networks has been proposed in [7]. This implementation could not run in an opportunistic network, though, as it does not allow mobile nodes to store and carry messages while moving in the network.

In the remainder of this paper we present a middleware system that implements the OTR algorithm, and that can run effectively in an opportunistic network. This system has been fully implemented in Java, and it has been tested in real conditions using a small flotilla of smartphones as mobile nodes. The OTR algorithm is implemented on top of a communication layer that supports network-wide, content-driven message dissemination based on controlled epidemic routing. This combination of the OTR algorithm and epidemic routing is consistent, because epidemic routing is an effective way to disseminate messages in an opportunistic network, where mobility patterns are neither planned nor controlled, and where mobile nodes (with the messages they carry) can disappear for a while from the network. Besides, the OTR algorithm requires n-n communication: in every step each process (or mobile node) must send a message to all other processes (or nodes). With epidemic routing, sending a message to all nodes is not significantly different from sending a message to a single node, so combining the OTR algorithm with epidemic routing makes sense when targeting opportunistic networks.

The remainder of this paper is organized as follows. Related work is presented and discussed in Section 2. The system model we consider in this work is detailed in Section 3. The Heard-Of (HO) model and the One-Third Rule (OTR) algorithm are presented in Section 4. Section 5 presents our implementation of the OTR algorithm, and Section 6 presents experimental results obtained with this implementation. Section 7 concludes the paper.

2. RELATED WORK

Consensus problems have been studied extensively during the last decades, most often with system models that fit the characteristics of traditional wired networks. As a general rule, the papers assume that the network is static and connected, that is, any node can send a message at any time to any other node. Moreover, they also assume that the system can eventually become synchronous, or that it can be augmented with failure detectors, so as to go round the so-called "FLP impossibility result" [18].

Most system models focus on node failures and tend to neglect link failures, though. According to Borran et al. [7] this bias may have its root in the FLP paper [18] (which assumes process crashes and reliable links), but solutions designed for environments where this bias is acceptable should not be used in environments where it is not acceptable.

Mobile ad hoc networks are such environments where reliable links should never be assumed, and for which system models must admit transient link failures. Yet several consensus protocols and algorithms have been proposed for such networks, based on overly optimistic assumptions.

Crash-tolerant broadcast protocols that can be used to solve consensus problems in mobile ad hoc networks are presented in [38]. These protocols assume the existence of oracles that can predict contacts and transmission times between nodes at any point of time. Such an assumption can only be satisfied in very specific mobile networks such as those where the mobility patterns of nodes are planned or controlled explicitly.

A hierarchical consensus protocol involving failure detectors is proposed in [39]. Mobile hosts are distributed into clusters, each cluster being controlled by a clusterhead. The protocol can tolerate faulty nodes, but links are assumed to be reliable.

An algorithm to solve consensus without knowing which nodes –and how many of them– are participating in the consensus is presented in [10], but the system model for this algorithm assumes reliable links and nodes that cannot crash. Variants of this algorithm have later been proposed in [11] and [19]. Both variants can admit faulty nodes, but links are still assumed to be reliable.

An implementation of the Paxos/Last Voting (P/LV) algorithm is proposed in [7]. This round-based algorithm requires the election of a coordinator, which once elected collects the contributions of all other nodes until consensus is reached. Interestingly, unlike the abovementioned solutions the P/LV algorithm can admit both link and node failures, as it relies on the Heard-Of (HO) model that makes no dis-

tinction between both kinds of failures. It assumes end-to-end connectivity in the network, though, which is a reasonable assumption for traditional mobile ad hoc networks, but an unfit one for opportunistic networks.

Byzantine agreement protocols for highly dynamic synchronous networks have been proposed in [2] and [20]. In [2] two randomized round-based protocols are presented, that can achieve almost-everywhere Byzantine agreement with high probability, even under a large number of Byzantine nodes and continuous adversarial churn. The network is represented as a sparse bounded degree expander graph that is assumed to remain connected at any time, although its topology can change arbitrarily from round to round. The protocol presented in [20] creates and maintains an expander overlay of clusters. Each cluster is used to inhibit the behavior of Byzantine nodes, and the overlay ensures communication among clusters. Although these protocols can support Byzantine failures, they can only run in connected networks and could hardly be used in opportunistic networks.

As explained in Section 1, an opportunistic network is a mobile ad hoc network that is at best only partially connected. Such a network can be continuously partitioned, so end-to-end transmission between remote nodes cannot rely exclusively on multiple rounds of short-range wireless transmissions. In fact, a temporaneous multi-hop path between two nodes may never exist during the network's whole lifetime. In such conditions the "store, carry and forward" principle, which is the key principle of Delay/Disruption-Tolerant Networking (DTN), must be used to allow messages to cover long distances by being carried physically by mobile nodes [30, 36]. With this approach mobility is an asset, as it helps bridge the gap between nodes that would otherwise be unable to communicate.

Message routing in opportunistic networks has already justified a fair amount of research. Most of the algorithms proposed rely on more or less constrained variants of the epidemic dissemination scheme, as defined in [37] and [16]: whenever a message is sent, several copies of this message are actually produced so these copies can propagate separately in the network, each copy being carried by a distinct mobile node. This approach is clearly a costly one but it helps preventing message loss, which can occur because of transmission failures (when a message is transmitted wirelessly between two neighbor nodes), or because the carrier of a copy may unexpectedly disappear from the network. Many solutions have been proposed to keep the cost of epidemic routing at a reasonable level, using heuristics that basically aim at reducing the number of carriers for each message, and also sometimes at selecting the "best" carriers for each message. Some solutions rely on probabilistic or semi-probabilistic heuristics [27, 31, 40], or take into account the context of each mobile node [1, 5, 33]. Others assume that mobile nodes are carried by human beings, whose social interactions can be captured and used to drive message forwarding by predicting how people move or meet [29, 34], or by identifying what communities each person belongs to [12, 23, 24].

In this work we make no assumption about the carriers of mobile nodes. These carriers could thus be human beings, but they could as well be vehicles, robots, animals, or any combination of these. The system we define relies on a selective epidemic dissemination model, which in essence can be considered as an effective implementation of the abstract model described in [16].

3. SYSTEM MODEL

3.1 General architecture

The system model we consider consists of a finite set of mobile nodes $\mathcal{V}$. Each node features a short-range wireless interface that allows it to exchange messages in ad hoc mode –that is, without relying on any fixed infrastructure– with nodes in its radio range.

At any time a node is either up or down: a node can disable its wireless transceiver or enter standby mode spontaneously to save battery, or it can be switched off and on alternatively by a user. This behavior is normal and is not considered as a failure. When a node is down, it cannot communicate with its neighbors. When it is switched on again, its previous state is restored, and it initiates a neighbor discovery phase in order to adjust rapidly to its current surroundings.

No assumption is made about the mobility of nodes, or about their spatial distribution. A node can thus be sometimes isolated from the other nodes, when the distance to its closest neighbor exceeds its radio range.

The relations between nodes take place over a time span $\mathcal{T}$. At some time in $\mathcal{T}$ the system model is represented by the static graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ the set of edges. There exists an edge between two nodes u and v if u and v are within mutual radio range, and can thus exchange messages over the wireless medium. Over time, the system model can be represented by a *time-varying graph* (TVG) [9] $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \rho, \psi)$, where $\rho : \mathcal{E} \times \mathcal{T} \rightarrow \{0, 1\}$ is the edge presence function that indicates whether a given edge is available at a given time; and $\psi : \mathcal{V} \times \mathcal{T} \rightarrow \{0, 1\}$ is the node presence function that indicates whether a given node is available at a given time. At some time in $\mathcal{T}$, the underlying graph G of $\mathcal{G}$ is not necessarily connected. In fact, the underlying graphs of $\mathcal{G}$ may all be disconnected over $\mathcal{T}$.

3.2 Communication model

In the TVG formalism, a *journey* in $\mathcal{G}$ is defined by a sequence of couples $\{(e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)\}$, such that $\{e_1, e_2, \ldots e_n\}$ is a walk in G and $\rho(e_i, t_i) = 1$ and $t_{i+1} > t_i$ for all $i < n$ [9]. At time t, a message sent by a node u may eventually be received by a node v if a *journey* $u \rightsquigarrow v$ exists in the TVG $\mathcal{G}$, with a starting date t_1 such that $t_1 \geq t$. In other words, a message sent by u may reach v after being stored, carried, and forwarded successively by several nodes, even if an end-to-end path between u and v never exists in any underlying graph G of $\mathcal{G}$. The time elapsed between two consecutive hops $(t_{i+1} - t_i)$ may range between a few milliseconds (the message being forwarded rapidly between successive neighbor nodes) and minutes or hours (the message being carried for a while before being forwarded to another node).

In our system model, any message sent by a node disseminates in the network according to the epidemic model: when two nodes meet, they can seize this opportunity to create new copies of the messages they are carrying, thus increasing the number of carriers for these messages. This basic interaction scheme takes inspiration from the Autonomous Gossiping (A/G) algorithm [16], which itself defines a selective version of the epidemic routing model proposed in [37].

In our model, each node defines an *interest profile* that determines the kinds of messages it is willing to collect from other nodes, and for which it is therefore willing to serve as a mobile carrier.

The epidemic model increases the probability of message delivery, as it simultaneously exploits all possible journeys that involve nodes whose interest profiles match the kind of message considered. In practice, messages that match an interest profile p can be carried only by a subset of nodes $\mathcal{V}_p \subseteq \mathcal{V}$, which consists of all nodes in $\mathcal{V}$ whose interest profile is p (or larger than p). Possible journeys for copies of a message that matches p are defined in $\mathcal{G}_p = (\mathcal{V}_p, \mathcal{E}, \mathcal{T}, \rho, \psi)$.

3.3 Fault model

The nodes in our system model can be switched off and on at any time. When a node is switched off this is not considered as a "failure", for this is consistent with its normal behavior. Nodes may sometimes crash spontaneously and never recover, though, but this is assumed to be an exceptional event, concerning only a very small number of nodes. Using the TVG formalism, the notation $\psi(v,t) = 0$ is used to indicate that the node v is switched off at time t.

In an opportunistic network, any wireless link between two neighbor nodes is inherently transient. However, journeys between any pair of nodes do not require any temporaneous end-to-end connectivity between these nodes, and can thus rely on successions of transient links. As a consequence, disruptions of wireless links are not considered as "link failures" in our model.

The epidemic routing model cannot guarantee that each message sent in the network eventually gets delivered to all possible recipients. Our system model therefore admits *receive omissions*, and these are considered as *benign faults*.

We do not consider Byzantine faults in this system model: an active node is assumed to behave properly, and to recover appropriately after being switched off and on again. Furthermore, messages transferred wirelessly between neighbor nodes are assumed to be unaltered. More specifically, whenever a message is received from a neighbor node, the integrity of the received copy is checked, and this copy is simply discarded if it has been altered during the transmission. The epidemic model ensures that the receiver will find other opportunities to obtain the message anyway (possibly again from the same neighbor).

3.4 Assumptions

In an opportunistic network, no guarantee can be provided about message delivery ratios, transmission delays, node availability, etc. A few assumptions can however be made to limit uncertainties in our system model.

In the remainder of this paper we call a *session* the process that consists in starting a consensus agreement procedure, and pursuing this procedure until a decision is made. Several sessions may of course progress simultaneously in the network, and some nodes may participate in several consensus sessions simultaneously.

We assume that a consensus session $\mathcal{S}$ involves a subset $\mathcal{V}_\mathcal{S}$ such that $\mathcal{V}_\mathcal{S} \subseteq \mathcal{V}$ (*i.e.*, all nodes in the network are not necessarily involved in $\mathcal{S}$). Nodes in $\mathcal{V}_\mathcal{S}$ are called *participants* for session $\mathcal{S}$. Each of these nodes is expected to provide an initial value for $\mathcal{S}$, and to help run the consensus algorithm until a decision is made. We assume that any node $u \in \mathcal{V}_\mathcal{S}$ knows that it belongs to $\mathcal{V}_\mathcal{S}$, and knows $|\mathcal{V}_\mathcal{S}|$ *(i.e.,* how many nodes participate in $\mathcal{S}$). The definition and creation of the subset $\mathcal{V}_\mathcal{S}$ is application-dependent, and is therefore out of the scope of this paper.

A consensus session $\mathcal{S}$ spans over a time period $\mathcal{T}_\mathcal{S}$. During some periods in $\mathcal{T}_\mathcal{S}$, some nodes in $\mathcal{V}_S$ may disappear temporarily from the network. We assume that these nodes will eventually reappear during $\mathcal{T}_\mathcal{S}$, and resume their activity regarding $\mathcal{S}$. Some nodes in $\mathcal{V}_S$ may also crash and disappear definitively from the network. We assume that the number of crash failures for nodes in $\mathcal{V}_\mathcal{S}$ is unknown but can be bounded, and that the bound is low with respect to $\mathcal{V}_\mathcal{S}$.

Nodes in $\mathcal{V}_\mathcal{S}$ are all expected to serve as mobile carriers for messages pertaining to $\mathcal{S}$. Additionally, any node $u \in \mathcal{V} \setminus \mathcal{V}_\mathcal{S}$ can also serve as a benevolent carrier for messages pertaining to $\mathcal{S}$. Nodes that can carry messages pertaining to $\mathcal{S}$ thus belong to a subset $\mathcal{V}_{\mathcal{C}(\mathcal{S})}$ (carriers for session $\mathcal{S}$) such that $\mathcal{V}_\mathcal{S} \subseteq \mathcal{V}_{\mathcal{C}(\mathcal{S})} \subseteq \mathcal{V}$. These nodes all exhibit an interest profile p that matches messages pertaining to $\mathcal{S}$.

Any message pertaining to $\mathcal{S}$, whatever its sender $u \in \mathcal{V}_\mathcal{S}$, is assumed to disseminate thanks to nodes in $\mathcal{V}_{\mathcal{C}(\mathcal{S})}$, and eventually reach all or some of the nodes in $\mathcal{V}_\mathcal{S}$. The reason why only *some of the nodes* in $\mathcal{V}_\mathcal{S}$ are considered here is a consequence of the epidemic routing model, which cannot guarantee that each message eventually reaches all its possible recipients. The failure ratio is assumed to be low and bounded, though.

Finally, we assume that, for any given node $u \in \mathcal{V}_\mathcal{S}$, there are some periods in $\mathcal{T}_\mathcal{S}$, called "good periods for u" during which all the messages sent by u eventually reach all other nodes in $\mathcal{V}_\mathcal{S}$ (except nodes that crashed during $\mathcal{T}_\mathcal{S}$). This assumption is not required to ensure the termination of the OTR algorithm, but it makes it possible to terminate faster in some cases: as soon as node u decides, then if u is in a "good period" its decision can be transmitted to all other nodes in $\mathcal{V}_\mathcal{S}$. As observed in [7], assuming good periods in an asynchronous system is often more realistic than assuming that the system is partially synchronous.

4. SOLVING CONSENSUS WITH THE OTR ALGORITHM

4.1 Consensus

The consensus problem over a set $\Pi = \{p1, p2, ...p_n\}$ of processes (which in our system model are called *participants* and are assumed to run on distinct nodes) is defined by the following properties:

- Validity: Any decision is the initial value of some participant.
- Agreement: No two participants decide differently.
- Termination: All correct participants[1] eventually decide.

4.2 Overview of the Heard-Of model

A computation in the *Heard-Of* (HO) model [13] evolves in asynchronous communication-closed rounds, without any need for a failure detector. In each round, each process sends a message to all the other processes and then waits

[1]In our system model, a *correct participant* is a participant that does not crash permanently during a consensus session.

to receive similar messages sent in the same round. Late messages pertaining to former rounds are discarded. The features of a specific system are captured by a *communication predicate*, which is expressed in terms of *Heard-Of sets*: $HO(p, r)$ represents the set of processes from which process p "hears of" (i.e., receives some messages) at round r. A consensus problem is solved in the HO model by a *Heard-Of machine* defined by a pair $M = (A, \mathcal{P})$ where A is an algorithm and $\mathcal{P}$ is a communication predicate.

Several consensus algorithms have been expressed in the HO model [13]. These algorithms can hardly tolerate message loss, except for the Paxos/LastVoting (P/LV) algorithm and the One-Third Rule (OTR) algorithm. An implementation of the P/LV algorithm for mobile ad hoc networks has been proposed in [7]. This implementation could not run in an opportunistic network, though, as it is coordinated-based which requires temporaneous end-to-end connectivity between the coordinator and all the other processes.

4.3 Overview of the OTR algorithm

The *One-Third Rule* (OTR) algorithm is a perfect candidate for opportunistic computing. In [13] it is defined with the formalism of the HO model, but similar structure and decision conditions can be observed in other algorithms (*e.g.*, first round in [8]). Each round r in the OTR algorithm consists of two steps (see Algorithm 1): a sending step S_p^r in which process p sends its current contribution (for round r) to the other processes (line 3), followed by a transition step T_p^r in which, provided it has received enough contributions from the other processes (line 5), process p either takes a decision (line 8) or determines its contribution for the next round (line 6) and proceeds to that round (line 7).

Algorithm 1 The *One-Third Rule* algorithm

```
   Initialization:
1: x_p ← v_p                                  {v_p is the initial value of process p}

   Round r:
2: S_p^r:
3: send <x_p> to all other processes
4: T_p^r:
5: if | HO(p,r) | > 2n/3 values then
6:     x_p ← the smallest most often received value
7:     if more than 2n/3 values received are equal to x̄ then
8:         DECIDE(x̄)
```

A node p can tolerate not to receive messages from up to one-third of the other participants in round r, while still being able to decide or proceed to the next round. In practice, in an opportunistic network this may occur because some of the other participants have not reached round r yet (for example because these participants are currently in suspend mode), or because some participants have indeed sent their contributions for round r but these messages have not reached node p yet (and possibly never will).

Moreover, with the OTR algorithm a consensus computation involving n participants can progress from round r to the next if *at least one* participant can receive contributions (pertaining to round r) from more than $\frac{2n}{3}$ other participants. In an opportunistic network where message delivery can sometimes be delayed significantly (at least for some receivers), this property of the OTR algorithm is an asset, for the consensus computation can proceed from one round to the next as soon as one node has received enough contributions to do so.

Conversely, at least $\frac{2n}{3}$ participants must send contributions in each round, since this is a requirement for the algorithm to proceed to the next round or to the final decision.

Based on these observations the assumptions made in our system model (Section 3.4) can be refined so as to account for the specific requirements of the OTR algorithm:

- At each round r, the number of participants sending their contribution in the network should not be smaller than $\frac{2n}{3}$ (which means that about $\frac{n}{3}$ participants can actually "skip" a round without preventing the computation to progress). By extension, the number of crash failures among the participants must be smaller than $\frac{n}{3}$.

- At each round r, message loss should be such that *at least one* node can receive more than $\frac{2n}{3}$ contributions (which means that receive omissions are admitted for most of the participants but one, which should be able to receive enough contributions to proceed to the next round). Using the HO formalism, this assumption can be expressed as:

$$\forall r,\, \exists\, p \in \mathcal{V}_s \; s.t. \; |HO(p, r)| > 2n/3$$

Note that these requirements fit perfectly with the characteristics of an opportunistic network, in which node availability and message delivery cannot be guaranteed. Moreover, the *n-to-n* communication pattern used in the OTR algorithm is satisfied by the epidemic routing model, since with this model sending a message to many or all nodes is not really different from sending a message to a single node.

According to [13] the communication predicate $\mathcal{P}_{(\mathcal{C}_0)\infty}$ associated with the OTR algorithm ensures that the consensus can be solved if there exists a round r_0 where $\mathcal{C}_0$ holds:

$$\exists\, \Pi_0 \; s.t. \; |\Pi_0| > 2n/3,\; \forall p \in \Pi :\; HO(p, r_0) = \Pi_0$$

where Π stands for the set of participants involved in the consensus. In other words, during r_0 all participants must be able to receive contributions from the very same subset of more than $\frac{2n}{3}$ participants (with $n = |\Pi|$), so they can make the same decision. This predicate can be expressed nicely in terms of *HO sets*, and it is sufficient to ensure that consensus is solved. Yet it does not define a necessary condition. If there exists a round r_0 where the contributions collected by participants are such that the smallest most frequent value is the same for all participants, then the consensus can be solved as well, even though all participants may not have received these values from the same contributors. This condition can hardly be expressed in terms of *HO sets,* yet it is weaker than predicate $\mathcal{P}_{(\mathcal{C}_0)\infty}$ and allows more flexibility in the system model.

5. OPPORTUNISTIC IMPLEMENTATION OF THE OTR ALGORITHM

Our system is architectured in two layers: the lower layer can support network-wide, content-driven message dissemination based on controlled epidemic routing, and the upper layer is an implementation of the OTR algorithm that interfaces with the communication layer.

5.1 Opportunistic communication layer

The communication layer implements a content-driven message dissemination model. It can actually be perceived as an effective implementation of the abstract model described in [16]. An overview of this communication layer is provided below. Further details can be found in [21].

Each node periodically broadcasts an announce in order to inform its neighbors (if any) about its presence. When two nodes meet they first exchange their *interest profiles*, that characterize the types of messages each node is willing to receive, and for which it is therefore willing to serve as a mobile carrier. Based on this exchange of profiles, each node can determine accurately which messages could be of interest to its neighbor, and make an offer accordingly. Messages are then effectively exchanged through a succession of query-and-reply cycles.

Each node maintains a local cache to store the messages it carries, so they can be proposed to any new neighbor. In order to prevent network congestion, each message is allowed a specific lifetime. When this lifetime is over, all copies of this message are removed from the caches of mobile nodes, so this message actually stops disseminating in the network. The dissemination of a message can also be canceled explicitly on a mobile node. Once a message is canceled, the node does not propose it to any neighbor anymore, and if conversely a neighbor actually offers to provide this message, this neighbor is notified that it too should cancel the message whose dissemination is not required anymore. This approach is referred to as *network healing* in the literature. It is an effective way to limit the cost of epidemic routing in an opportunistic network, using either *Passive Cure* [25, 22] or *Active Cure* techniques [35].

The communication layer provides a *publish/subscribe* application programming interface (API), presented in Algorithm 2. With this API, the content-driven nature of message selection is based on the notion of group. A group, identified by a group identifier *(grpId)*, is a set of nodes that cooperate in a common task and are thus potentially interested in the same kind of messages. The *interest profile* of a node is a compilation of the ids of all the groups it belongs to. Function *subscribe* (line 2) allows a process to specify that it is interested in receiving the messages for a given group. This function adds the specified group id in the node's interest profile, and consequently the node will try to collect messages addressed to this group from any neighbor it will meet thereafter. Whenever a message is received that matches a group the node belongs to, a *receive* event is triggered (line 3) accordingly.

Algorithm 2 The communication layer API

```
1: Function publish (msgId, grpId, sndId, BODY, [dln])
2: Function subscribe (grpId)
3: Event receive (msgId, grpId, sndId, BODY, [dln])
4: Function cancel (msgId)
5: Function relay (grpId)
```

The communication layer assigns a unique identifier to each node. This identifier can for example be the IMEI on a smartphone, or an auto-configured link-local IPv6 address. Each message must likewise be assigned a unique identifier, which will notably be used by the system to detect duplicates, and thus prevent useless transfers of message copies between neighbor nodes. Sending a message in the network is done with the *publish* function (line 1), that takes as parameters identifiers for the message itself *(msgId)*, for its sender *(sndId)*, and for the group of nodes it is addressed to *(grpId)*. The body of the message is just perceived as a payload by the system.

When a message is published on a node, or received from a neighbor, this message is deposited in the local cache. Afterwards every contact with a new interested neighbor is an opportunity to transfer a copy of the message to that neighbor.

As explained above, each message can optionally be assigned a set lifetime when it is published (last parameter in line 1), and the dissemination of a message can also be canceled explicitly (line 4). Both mechanisms, if used wisely, can help reduce the cost of epidemic routing.

By default a message disseminates by being stored, carried, and forwarded by nodes that have subscribed to the group this message is addressed to. A node can however be configured so as to serve as a benevolent carrier for messages addressed to a group it does not belong to. With the API this is obtained through the *relay* function (line 5), which basically has the same effect as function *subscribe*, except that messages received by a benevolent carrier will not trigger any *receive* event on that node.

5.2 Opportunistic OTR algorithm

Our implementation of the OTR algorithm based on the opportunistic communication layer is shown in Algorithm 3. A consensus session is initiated by calling function *startSession*, taking the group identifier, the number of participants in this group, and the initial value for the local participant as parameters. A subscription is then set for the group (line 8) before starting the first round of the OTR algorithm (line 9).

At each round, the current contribution is published (line 12), and the identifier of the message hence published is recorded in *contribIds* so it can be canceled later (line 13).

When a contribution is received from another participant, the behavior of the receiver depends on whether this contribution pertains to a new round (lines 18-22), to the current round (lines 24-30), or to a former round (line 32). In the first case the receiver cancels messages pertaining to the current round before moving to the new round. In the second case it checks if enough contributions have been received to either make a decision (line 29) or start the next round (line 30). In the third case it simply discards the contribution it has just received, but cancels the corresponding message so as not to take part in its dissemination (line 32).

Note that message cancellation is here used systematically as a means to prevent messages that pertain to former rounds to keep propagating in the network. This form of cross-layering between the OTR algorithm and the opportunistic communication layer helps reduce the cost of epidemic message dissemination.

Function *decide* is called when a decision is made locally (line 29), or when a decision message is received from another participant (line 41). In any case this function is run only once on each node (line 33): all pending messages are canceled (lines 36-37) and the decision is published (line 40) with a unique message identifier that is produced using *grpId* as a seed (line 39). Thus, if several nodes make a decision in the same session, all messages carrying this decision will have the same identifier and will thus be considered as duplicates of the same message by the communication layer.

As explained in Section 3.4, publishing the decision is not required by the OTR algorithm, but it can help terminate a consensus session faster.

Algorithm 3 Opportunistic version of the OTR algorithm

Initialization:
1: $id_p \leftarrow id\ of\ local\ node$ {must be unique in the network}
2: $contrib_p \leftarrow \{\}$ {contributions received for r_p (multi-set)}
3: $contribIds_p \leftarrow \{\}$ {ids of messages received during r_p (set)}

Function *startSession(grpId, nbNodes, v)*:
4: $x_p \leftarrow v$ {initial value for node p}
5: $grpId_p \leftarrow grpId$
6: $nbNodes_p \leftarrow nbNodes$
7: $solved \leftarrow$ **false**
8: **subscribe**$(grpId_p)$
9: **startRound**(1)

Function *startRound(r)*:
10: $r_p \leftarrow r$
11: $msgId \leftarrow$ **genId**() {call id generator}
12: **publish**$(msgId,\ grpId_p,\ id_p,\ CONTRIB(r_p,\ x_p))$
13: $contribIds_p \leftarrow contribIds_p \cup \{msgId\}$

Upon receive $(msgId,\ grpId,\ sndId,\ CONTRIB(r, x))$ **do**
14: **if** *solved* **then**
15: **cancel**$(msgId)$
16: **switch** (r)
17: **case** $(r > r_p)$**:**
18: **for** $id \in contribIds$ **do**
19: **cancel**(id)
20: $contrib_p \leftarrow \{x\}$
21: $contribIds_p \leftarrow \{msgId\}$
22: **startRound**(r)
23: **case** $(r = r_p)$**:**
24: $contrib_p \leftarrow contrib_p \cup \{x\}$
25: $contribIds_p \leftarrow contribIds_p \cup \{msgId\}$
26: **if** $|contrib_p| > 2/3 * nbNodes_p$ **then**
27: $x_p \leftarrow smallest\ most\ often\ received\ value\ in\ contrib_p$
28: **if** *all values are equal to* $\overline{X}$ *in* $contrib_p$ **then**
29: **decide**$(\overline{X})$
30: **startRound**$(r_p + 1)$
31: **case** $(r < r_p)$**:**
32: **cancel**$(msgId)$

Function *decide(v)*:
33: **if** $\neg solved$ **then**
34: $solved \leftarrow$ **true**
35: $x_p \leftarrow v$
36: **for** $id \in contribIds$ **do**
37: **cancel**(id)
38: $contribIds_p \leftarrow \{\}$
39: $msgId \leftarrow$ **genId**$(grpId)$
40: **publish**$(msgId,\ grpId_p,\ id_p,\ DECISION(x_p))$

Upon receive $(msgId,\ grpId,\ sndId,\ DECISION(v))$ **do**
41: **decide**(v)

6. EXPERIMENTAL EVALUATION

Evaluating the performance of a system capable of running in opportunistic networks is a challenge. In the literature, protocols and systems designed for such networks, and more generally for Delay/Disruption-Tolerant Networks, are often evaluated using simulators, and little or no effort is devoted to producing code that can be used in a real setting. Yet, as observed in [26] "rare are the [DTN] protocols that were implemented, tested in real-life and proven to be free of lethal stealthy assumptions". A salient feature of our system is that it has been fully implemented, and validated in real conditions using a small flotilla of smartphones as mobile nodes.

6.1 Experimentation conditions

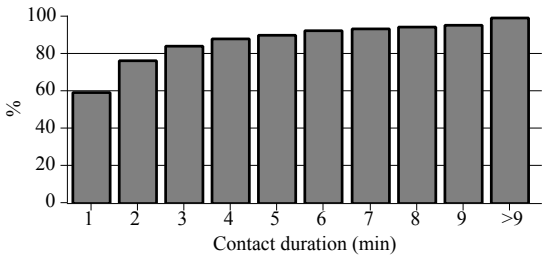

Figure 2: Cumulative distribution of radio contact durations

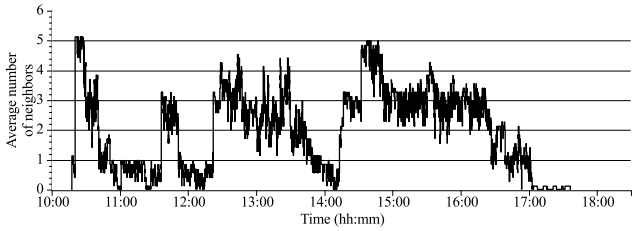

Figure 3: Timeline of the average number of neighbors during the experiment

In order to demonstrate that our system can indeed solve consensus in an opportunistic network, volunteers have been equipped with HTC smartphones, whose Wi-Fi chipsets were configured to operate in ad hoc mode. Each smartphone ran a small Android application (named *iAgree*) based on our system. This simple application allows a user to initiate new consensus sessions, and to join, participate, and display the status of ongoing and past sessions. The volunteers were asked to carry their smartphone while roaming the laboratory building or its surroundings, and to use application *iAgree* every now and then. Trace logs were collected and analyzed after the end of the experiment.

The experiment spanned over 8 hours and involved a small population of 7 volunteers (and as many smartphones). It was mostly meant to serve as a proof of concept. A comprehensive evaluation of the system's effectiveness and efficiency would require a far larger population of mobile nodes, and should ideally span over several days or weeks.

During this experiment the system was configured to give each message a lifetime of 12 hours. No message was therefore removed because of an exhausted lifetime. Moreover, because only 7 smartphones were available for this experiment, none of them was configured to serve as a benevolent carrier: every smartphone was systematically enrolled as a consensus participant.

6.2 Results

3424 radio contacts occurred between smartphones during this 8 hour experiment, with an average contact duration of 162 seconds. The cumulative distribution of contact durations is presented in Figure 2. It can be observed that almost 60% of radio contacts lasted for less than a minute, which confirms the transient nature of the radio contacts established between smartphones.

A timeline of the evolution of the average number of neighbors is presented in Figure 3, and the cumulative distribu-

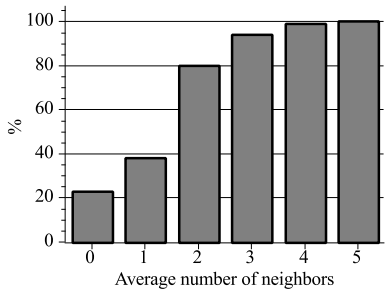

Figure 4: Cumulative distribution of the average number of neighbors

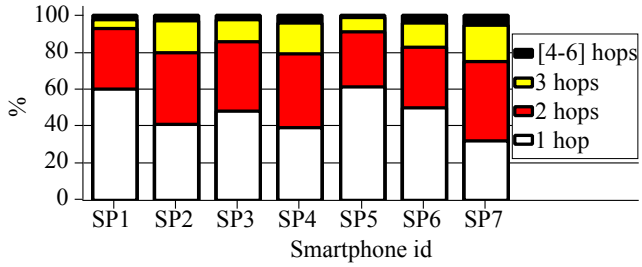

Figure 5: Amount of contributions received in a direct/multi-hop manner per smartphone

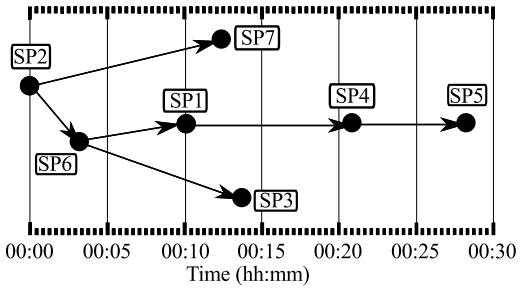

Figure 6: Timeline of the dissemination of a contribution during the experiment

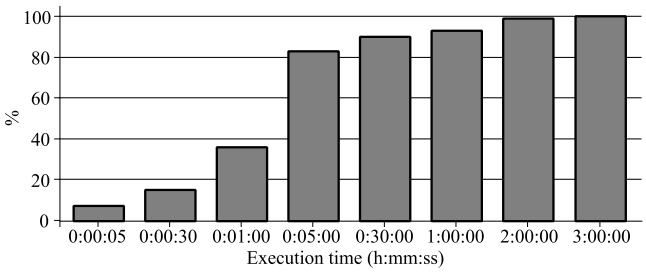

Figure 7: Cumulative distribution of the execution times of sessions

tion of this number is shown in Fig 4. It can be observed that each smartphone had at most one neighbor during about 40% of the experiment's duration, but was actually alone (*i.e.*, with no neighbor) during more than 20% of that time. Moreover, there was no period of time during which all smartphones were connected all together (each smartphone would otherwise have detected 6 neighbors simultaneously). These results confirm the dynamic and disconnected nature of the network, whose topology changed continuously and rapidly during the whole experiment.

151 consensus sessions were initiated by users (either sequentially or concurrently) during the experiment, and running these sessions led to the exchange of 4530 contributions among the smartphones.

Figure 5 shows the distribution of the number of hops required for contributions to reach each smartphone. For example, 50% of the contributions received by smartphone *SP6* were received directly from the sender (1-hop), 33% contributions were received by *SP6* after 2 hops, etc. The smartphones actually received most of the contributions after multi-hop trips during the experiment, a few of these trips requiring up to 6 hops between sender and receiver.

This observation confirms the interest of multi-hop relaying between smartphones, but it is not sufficient to demonstrate the interest of the *store, carry and forward* principle in an opportunistic network such as that formed by the smartphones during the experiment. In order to clarify this point, Figure 6 illustrates how one particular consensus contribution sent by smartphone *SP2* actually disseminated during the experiment. A few minutes after this particular contribution was published (*i.e.* sent to all other nodes) by our system on *SP2,* a radio contact was established with *SP6*, which thus got a copy of the contribution and became a new carrier for this contribution. *SP2* later managed to forward the contribution to *SP7*, while *SP6* forwarded it to *SP1* first, and later to *SP3*. The contribution thus kept disseminating, until it reached the last smartphone *SP5*, almost half an hour after it was initially published on *SP2*.

These results confirm the need to rely on delay/disruption-tolerant multi-hop forwarding to ensure information dissemination in an opportunistic network. Any consensus algorithm requiring temporaneous end-to-end connectivity would be completely ineffective in such a network.

Since consensus contributions had to propagate opportunistically between the smartphones, each consensus session could take a while before a decision was made. Actually, 9% of the sessions could not reach a decision during the experiment, but these sessions were initiated shortly before the end of the experimentation period. Figure 7 presents the cumulative distributed of the execution times for all completed sessions. The execution time of a session is here defined as the time elapsed between the sending of the first contribution by one of the participants, and the last decision made or received by any of the participants. It can be observed that a few sessions were completed in only a few seconds, but most of them required several minutes, and some even took a couple of hours to complete. These results clearly show that consensus solving in an opportunistic network can indeed take a while, but is perfectly feasible provided an appropriate communication model is used.

Figure 8 shows the cumulative distribution of the average number of rounds required to solve consensus in all completed sessions. It can be observed that consensus was obtained in only one round for about 5% of the sessions. Most sessions required a couple of rounds to complete, though, and it is likely that more rounds would have been required with a larger population of consensus participants.

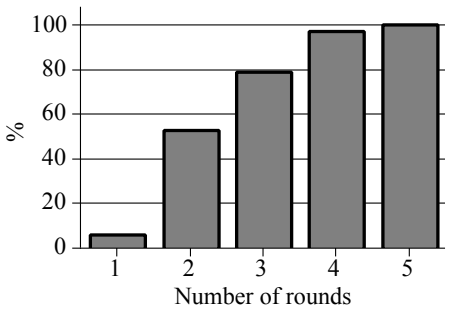

Figure 8: Cumulative distribution of the number of rounds required to solve consensus

All these results globally demonstrate that our system, which combines the OTR algorithm with an opportunistic communication layer, is effective at solving consensus in an opportunistic network. As mentioned above this short experiment was only meant to serve as a proof of concept. Further experiments would be required to observe how this system can perform in far larger networks involving hundreds of mobile nodes, roaming large areas over long time spans. Larger experiments would notably make it possible to observe phenomena that could not be highlighted with a flotilla of only 7 smartphones, such as the effect of relying on benevolent carriers (i.e., mobile nodes that carry messages they are not directly interested in) to carry consensus contributions, or the effect of receive omissions when consensus sessions involve many participants.

7. CONCLUSION

The One-Third Rule (OTR) algorithm is an elegant solution to solve consensus in networks where message loss can occur. Being based on the Heard-Of (HO) model, it is well suited to support transient process and link faults, which makes it an ideal solution to solve consensus in opportunistic networks. In such networks, messages propagate by being carried physically by mobile carriers whose mobility is usually neither planned nor controlled, so there is no guarantee that messages finally get delivered to their destinations.

In this paper we have presented a system that combines an implementation of the One-Third Rule (OTR) algorithm with a communication layer that supports network-wide, content-driven message dissemination based on controlled epidemic routing. Experimental results obtained with a small flotilla of smartphones confirm that this system is effective at solving consensus problems in an opportunistic network. The source code of this system is now distributed under the terms of the GNU General Public License[2].

8. ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their helpful comments and suggestions. They would also like to extend these thanks to Prof. Achour Mostéfaoui for his precious insight into distributed computing issues and solutions.

9. REFERENCES

[2]http://www-casa.irisa.fr/consensus

[1] V. Arnaboldi, M. Conti, and F. Delmastro. CAMEO: A Novel Context-Aware Middleware for Opportunistic Mobile Social Networks. *Pervasive and Mobile Computing*, 2013.

[2] J. Augustine, G. Pandurangan, and P. Robinson. Fast Byzantine Agreement in Dynamic Networks. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 74–83, New York, NY, USA, 2013. ACM.

[3] A. Benchi, F. Guidec, and P. Launay. A Message Service for Opportunistic Computing in Disconnected MANETs. In *12th IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 118–131, Stockholm, Sweden, June 2012. Springer.

[4] A. Benchi, P. Launay, and F. Guidec. A P2P Tuple Space Implementation for Disconnected MANETs. *Peer-to-Peer Networking and Applications*, pages 1–16, Aug. 2013.

[5] C. Boldrini, M. Conti, and A. Passarella. Context and Resource Awareness in Opportunistic Network Data Dissemination. In *The Second IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications*, Newport Beach, CA, USA, June 2008.

[6] C. Boldrini, K. Lee, M. Önen, J. Ott, and E. Pagani. Opportunistic networks. *Computer Communications*, pages 1–4, March 2014.

[7] F. Borran, R. Prakash, and A. Schiper. Extending Paxos/LastVoting with an Adequate Communication Layer for Wireless Ad Hoc Networks. In *2008 Symposium on Reliable Distributed Systems*, page 227–236. IEEE, 2008.

[8] F. V. Brasileiro, F. Greve, A. Mostéfaoui, and M. Raynal. Consensus in One Communication Step. In *Parallel Computing Technologies*, volume 2127 of *LNCS*, pages 42–50. Springer, 2001.

[9] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-Varying Graphs and Dynamic Networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, Apr. 2012.

[10] D. Cavin, Y. Sasson, and A. Schiper. Consensus with Unknown Participants or Fundamental Self-Organization. In *ADHOC-NOW*, volume 3158 of *LNCS*, pages 135–148. Springer, 2004.

[11] D. Cavin, Y. Sasson, and A. Schiper. Reaching Agreement with Unknown Participants in Mobile Self-Organized Networks in Spite of Process Crashes. Technical report, 2005.

[12] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of Human Mobility on Opportunistic Forwarding Algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, jun 2007.

[13] B. Charron-Bost and A. Schiper. The Heard-Of Model: Computing in Distributed Systems With Benign Faults. *Distributed Computing*, 22(1):49–71, 2009.

[14] M. Conti, S. Giordano, M. May, and A. Passarella. From Opportunistic Networks to Opportunistic Computing. *IEEE Communications Magazine*, 48(9):126–139, Sept. 2010.

[15] M. Conti, E. Marzini, D. Mascitti, A. Passarella, and

L. Ricci. Service Selection and Composition in Opportunistic Networks. In *9th IEEE International Wireless Communications and Mobile Computing Conference*, pages 1565–1572. IEEE CS, July 2013.

[16] A. Datta, S. Quarteroni, and K. Aberer. Autonomous Gossiping: a Self-Organizing Epidemic Algorithm for Selective Information Dissemination in Mobile Ad Hoc Networks. In *International Conference on Semantics of a Networked World*, number 3226 in LNCS, pages 126–143, Paris, France, June 2004.

[17] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of The 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 27–34, New York, USA, 2003. ACM.

[18] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of Distributed Consensus With One Faulty Process. *Journal of the ACM*, 32(2):374–382, Apr 1985.

[19] F. Greve and S. Tixeuil. Knowledge Connectivity vs. Synchrony Requirements for Fault-Tolerant Agreement in Unknown Networks. In *Dependable Systems and Networks, 2007. 37th Annual IEEE/IFIP International Conference on*, pages 82–91, June 2007.

[20] R. Guerraoui, F. Huc, and A.-M. Kermarrec. Highly Dynamic Distributed Computing with Byzantine Failures. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 176–183. ACM, 2013.

[21] J. Haillot and F. Guidec. A Protocol for Content-Based Communication in Disconnected Mobile Ad Hoc Networks. *Journal of Mobile Information Systems*, 6(2):123–154, 2010.

[22] K. A. Harras, K. C. Almeroth, and E. M. Belding-Royer. Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks. In *IFIP Networking Conference, Waterloo, Ontario, CANADA*, May 2005.

[23] P. Hui, J. Crowcroft, and E. Yoneki. BUBBLE Rap: Social Based Forwarding in Delay Tolerant Networks. In *9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 241–250, Hong Kong, China, may 2008. ACM.

[24] P. Hui, E. Yoneki, S.-Y. Chan, and J. Crowcroft. Distributed Community Detection in Delay Tolerant Networks. In *Sigcomm Workshop MobiArch*, Kyoto, Japan, aug 2007.

[25] L. jyh Chen, C. hung Yu, C. long Tseng, H. hua Chu, and C. fu Chou. A Content-Centric Framework for Effective Data Dissemination in Opportunistic Networks. *IEEE Journal of Selected Areas in Communications*, 2008.

[26] M. J. Khabbaz, A. Chadi M., and F. Wissam F. Disruption-Tolerant Networking: a Comprehensive Survey on Recent Developments and Persisting Challenges. *IEEE Communications Surveys and Tutorials*, 14(2):607–640, 2012.

[27] A. Lindgren, A. Doria, and O. Schelen. Probabilistic Routing in Intermittently Connected Networks. In *Proceedings of the 1st International Workshop on Service Assurance with Partial and Intermittent Resources*, Fortaleza, Brazil, Aug. 2004.

[28] A. Makke, Y. Mahéo, and N. Le Sommer. Towards Opportunistic Service Provisoning in Intermittently Connected Hybrid Networks. In *4th International Conference on Networking and Distributed Computing*, pages 28–32, Honk Kong, China, Dec. 2013. IEEE CS.

[29] M. Musolesi, B. Hui, C. Mascolo, and J. Crowcroft. Writing on the Clean Slate: Implementing a Socially-Aware Protocol in Haggle. In *IEEE International Workshop on Autonomic and Opportunistic Communications*, Newport Beach, CA, jun 2008.

[30] H. A. Nguyen and S. Giordano. Routing in Opportunistic Networks. *International Journal of Ambient Computing and Intelligence*, 1(3):19–38, 2009.

[31] H. A. Nguyen, S. Giordano, and A. Puiatti. Probabilistic Routing Protocol for Intermittently Connected Mobile Ad hoc Network (PROPICMAN). In *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–6, Helsinky, Finland, June 2007. IEEE CS.

[32] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *IEEE Communications Magazine*, 44(11):134–141, Nov. 2006.

[33] R. F. P. Quelhas. *Improving Opportunistic with Social Context Communications*. PhD thesis, Universidade do Minho, Escola de Engenharia, Oct. 2011.

[34] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: a Networking Architecture Designed Around Mobile Users. In *Proceedings of the 2006 IFIP Conference on Wireless on Demand Network Systems and Services*, Jan. 2006.

[35] J. P. Tower and T. D. Little. A Proposed Scheme for Epidemic Routing With Active Curing for Opportunistic Networks. In *2008 22nd International Workshops on Advanced Information Networking and Applications (AINA Workshops)*, pages 1696–1701. IEEE, 2008.

[36] A. Triviño-Cabrera and S. Cañadas-Hurtado. Survey on Opportunistic Routing in Multihop Wireless Networks. *International Journal of Communication Networks and Information Security*, 3(2), Aug. 2011.

[37] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical report, Duke University, Apr. 2000.

[38] E. W. Vollset and P. D. Ezhilchelvan. Design and Performance-Study of Crash-Tolerant Protocols for Broadcasting and Reaching Consensus in MANETs. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems*, volume 0, page 166–178, Washington, DC, USA, 2005. IEEE Computer Society.

[39] W. Wu, J. Cao, J. Yang, and M. Raynal. Design and Performance Evaluation of Efficient Consensus Protocols for Mobile Ad Hoc Networks. *IEEE Transactions on Computers*, 56(8):1055–1070, 2007.

[40] Z. Zhang. Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges. *IEEE Communications Surveys and Tutorials*, 8(1):24–37, Jan. 2006.