Padoc: Enabling Social Networking in Proximity

Adrian Holzer[a], Sven Reber[a], Jonny Quarta[a], Jorge Mazuze[b], Denis Gillet[a]

[a]*Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland*
[b]*Médecins Sans Frontières, 1201 Geneva, Switzerland*

Abstract

Apps supporting social networking in proximity are gaining momentum as they enable to both augment face-to-face interaction with a digital channel (e.g. classroom interaction systems) and augment digital interaction by providing a local real life feeling to it (e.g. nearby friends app in Facebook). Such apps effectively provide a cyber-physical space interweaving digital and face-to-face interaction. Currently such applications are mainly relying on Internet connection to the cloud, which makes them inaccessible in parts of the world with scarce Internet connection. Since many of their interactions happen locally, they could theoretically rely on Mobile Networking in Proximity (MNP), where data could be exchanged among devices without the need to rely on the availability of an Internet connection. Unfortunately, there is a lack of off-the-shelf programing support for MNP. This paper addresses this issue and presents Padoc, a middleware for social networking in proximity that provides multi-hop MNP support when cloud connection is unavailable. Furthermore the paper evaluates three MNP message diffusion strategies and presents Heya a novel classroom interaction app running on iOS devices as a proof-of-concept built on top of Padoc.

Keywords: Social Networking in Proximity, Cyber-Physical Systems, Ad hoc Networking, Middleware

1. Introduction

Far from only serving as ego boosting and gossiping platforms, social media apps have proven to be powerful tools for learning and sharing knowledge, think of YouTube instructional videos, Quora question and answer forums, Stackoverflow coding community, and so forth. Sharing knowledge is particularly important in emergency situations, such as in the 2010 earthquake in Haiti [30], during the 2011 Tsunami in Japan [1], or in the recent 2015 earthquake in Nepal.[1] Spacialized platforms such as Ushahidi are designed to crowdsource social information to produce a crisis map from different sources [23] and mainstream social apps such as Twitter can be mined to produce similarly useful data [9]. From this brief overview it looks like social media applications have taken over the world. Or have they? Unfortunately, their reliance on the cloud makes them still unavailable in much of the world that cannot count on ubiquitous Internet access. For instance, according to United Nations specialized agency for information and communication technologies (ITU) *in least developed countries, only 7% of households have Internet access.*[2] Increasing this number will require massive infrastructure investments [18]. Possibly, wired Internet solutions will never emerge in many places and will be directly replaced by mobile solutions [4]. However, mobile solutions still require infrastructure and impose centralized points of control (the network operators), which implies a single point of failures [33].

1.1. Mobile networking in proximity

A potential solution is to leverage on the networking capabilities of end user devices, such as tablets or mobile phones to enable mobile networking in proximity (MNP), where apps do not need

Email address: adrian.holzer@epfl.ch (Adrian Holzer)

[1] http://www.nytimes.com/2015/04/28/world/asia/google-and-facebook-help-nepal-earthquake-survivors-and-contacts-connect.html?_r=0

[2] http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf

to rely on any fixed infrastructure when the Internet connection is not available. In such settings, phones communicate directly with one another and, as they can relay each other's messages, it is possible for one device to reach devices far beyond its technical transmission range, as long as there is a chain of devices that can act as relays. Such settings can be combined with standard cloud based communication when the network is available to synchronize data. Unfortunately, despite the fact that over a decade of research in ad hoc networking has resulted in a huge amount of literature investigating communication protocols, middleware and other programming support mechanisms, there are currently almost no real world implementations of end-user MNP [6].

1.2. Social networking in proximity

A class of applications that is particularly well suited for MNP is the social networking in proximity apps, also known as co-located social media. Co-located social media apps are gaining momentum as they enable to both augment face-to-face interaction with a digital channel (e.g. classroom interaction systems) and augment digital interaction by providing a local real life feeling to it (e.g. the *nearby friends* app in Facebook). As such, they provide some kind of cyber-physical space where face-to-face interaction interweaves with digital interaction. Since most of the interaction happens between nearby users there is no need for constant Internet connection.

1.3. Contributions

This paper specifically aims at bridging the gap between the theoretical research in ad hoc networks and its practical implementations to support co-located mobile ad hoc social media apps by providing adequate programming support for that purpose. More specifically, we focus on a subclass of such apps, namely classroom interaction apps for classrooms with no access to Internet. Indeed, classroom interaction apps have gained interest in recent years to provide students with channels to express themselves more freely and for teachers to get feedback on the progress of their students, but they are lacking in non-connected settings. The main contributions that we present in this paper can be summarized as follows:

- First, we present Padoc, a novel middleware for commercially available iOS mobile devices designed to support co-located mobile social media apps. Padoc provides an implementation of a location-based multicast addressing service with two message diffusion strategies (push and pull) that can be fine-tuned depending on the application scenario. Furthermore Padoc also provides two message diffusion algorithms for message diffusion. A scoped flooding algorithm and a scoped counter-based scheme that allows to reduce message load. To the best of our knowledge it is the first time such a scheme is evaluated on standard commercial off-the-shelve devices and it is also the first multi-hop library available for such devices.

- Second, we present a performance evaluation of the different message diffusion strategies provided by Padoc using 10 real devices (iPod Touch) in three network topological settings for two messaging patterns. Note that such an evaluation is still challenging as there are no readily available testbeds for ad hoc networking algorithms on actual mobile devices.

- Third, we present Heya, a novel co-located social media application for group interaction in classroom settings with no access to the Internet built on top of Padoc as a proof-of-concept available freely on the Apple AppStore.

- Fourth, we present a preliminary usability evaluation of Heya with national staff of Médecins Sans Frontières in the field in Mozambique.

- Fifth, in order for the contributions of this article to spillover in the community, the latest versions of the middleware[3] as well as the application prototype[4] are freely available under an open source MIT licence.

1.4. Roadmap

The remainder of this paper is structured as follows: Section 2 discusses related work and Section 3 presents a requirement analysis of classroom interaction apps, a type of social networking in proximity. Section 4 presents the Padoc middleware and Section 5 presents its evaluation. Section 6

[3] `https://github.com/react-epfl/padoc-lib`
[4] `https://github.com/react-epfl/padoc-heya`

then presents Heya, the proof of concept application built on top of Padoc along with a pilot usability study in Mozambique with staff from Médecins Sans Frontières. Finally, Section 7 concludes with a discussion of the results and an outlook on future research avenues.

2. Related work

Here we review the main research on middleware mobile networking in proximity and classroom interaction systems and point to open research opportunities.

2.1. Theoretical middleware solutions

Middleware for MNP have been studied extensively in the literature [7, 26, 6]. For instance the authors of [26, 7] surveyed a large range of middleware solutions developed over the last 15 years. They make the harsh observation that hardly any of them are used in the real world. In an overview of research in ad hoc networks, the authors of [6] observe that contrary to sensor network solutions or vehicular network solutions, which are emerging in the real world, there is a lack of real-world people-centric ad hoc networks [6]. The people-centric paradigm is based on the premise that MNP can leverage on mobile devices that people use in their daily lives. This type of network is precisely the type of network on which co-located ad hoc social media apps could be built. So despite theoretical work on MNP, there has hardly been any spillover in the real world [26] and there is currently a lack of solutions that provide programming support for developers to build mainstream MNP. The main issues identified related to building proper people-centric solutions is to move away from adopting traditional system-oriented middleware solutions, and moving towards taking people's needs and constraints into account first [6, 26].

2.2. Practical middleware solutions

One of the main shortcomings of the literature that prevents ad hoc application from becoming mainstream is the fact that most solution are not targeting commercially available mobile devices. Recently, several efforts move in this direction.

BASA. For instance, as part of the European FP7 Societies project[5] which aimed at supporting services bridging community and pervasive computing, researchers recently built the BASA System [32]. BASA provides developers with a system to build mobile ad hoc social media apps on top of Android. BASA's network layer extends the Android limitation that nearby neighbors cannot immediately form communities since there is no ad hoc function that enables this out-of-the box. However, BASA does not address the multi-hop limitation of the platform.

Serval. The Serval Mesh [12, 13] is another example of research project focused on a real world application. Serval is a research project aiming at allowing phone calls in geographically remote areas through an ad hoc network. The app is developed for Android devices, uses the Wi-Fi Direct technology and is available on Google Play. To extend its capabilities, the Serval Project provides a Mesh Extender [25]. The Serval Mesh application is open source and has been tested in different situations, such as with the New Zealand Red Cross in 2012 and 2013. In this situation, they modified the version of the field assessment forms used by the Red Cross teams on the field so that they could use the Serval Mesh to be shared together. The middleware layer of Serval is called Serval DNA[6] and provides a network level Mesh Datagram Protocol (MDP) and an application layer Voice over Mesh Protocol (VoMP). Serval DNA also offers a content distribution protocol called Rhizome. Even though Serval targets commercially available devices, it appears that in order to perform multi-hop communication, the Linux kernel code of the Android platform must be modified. Indeed, it is not possible to create an efficient multi-hop network among devices with the standard Wi-Fi Direct available on off-the-shelf Android devices as *the routing can be implemented only as an application over the transport layer* [8]. In a nutshell, Android offers a Wi-Fi Direct interface for one hop ad hoc networking. Typically a node can create a Wi-fi Direct group. This node becomes the group owner and its neighbors become the clients. Unfortunately, multi-hop communication is hindered by the fact that a node cannot be part of several groups at the same time and the

[5] `http://www.ict-societies.eu`

[6] `http://developer.servalproject.org/dokuwiki/doku.php?id=content:servaldna:Main\%20Page`

fact that setting up connections and closing them is time consuming.

OpenGarden. The most popular ad hoc social media application to date is probably OpenGarden's Firechat.[7] The app is available on the AppStore and Google Play and allows to chat with people nearby without requiring an Internet connection. It was released in March 2014 for iOS as a proof-of-concept of the MultipeerConnectivity framework released shortly before by Apple. The MultipeerConnectivity framework, was Apple's first mainstream library to provide access to ad hoc peer-to-peer messaging. The app became popular thanks to two main events. First, the restrictions on Internet use imposed by the government of Iraq in 2014[8] made people find alternative ways to communicate together, and they discovered FireChat for this purpose, which made the application get known at a larger scale. Second, during the Hong Kong pro-democracy protests [5] that happened in September and October 2014, the authorities shut down the cellphone service in order to prevent people from organizing demonstrations. Since the MultipeerConnectivity framework provided by Apple only offers limited communication features for devices within each other's transmission range, Open Garden is working on providing developers with an SDK[9] to cover these shortcomings. However, at the time of writing, this SDK is not available.

In summary. The programming interfaces currently offered by operating systems for off-the-shelf mobile devices for ad hoc communication are limited to communication in the transmission range. There is currently a lack of programming support that hides the complexity of multi-hop communication behind any communication primitive. Thus more research is needed to convert theoretical findings about multi-hop communication services into practically applicable solutions to provide programming support to developers through adequate interfaces.

3. Classroom interaction apps

The review of middleware solutions suggests that there is still room for further investigation in providing programming support for social networking in proximity. The review also suggests that a way forward is to take a people-centric approach as described in [6] and build support for a set of defined applications that fulfil end user needs and constraints. This is precisely what we aim to achieve by targeting a particular type of social networking in proximity, namely, classroom interaction systems, as they represent a growing topic of interest in a crucial field, namely education (e.g., [15, 3, 16, 21]). These solutions, typically provide a shared messaging space for people to post questions and comments and possibly rate each other's contributions or answer polls created by the tutor. In the classroom context, increasing interactions is appreciated by teachers and students as it allows students to express themselves more frequently and, in cases where the interaction is anonymous, more freely, and it at the same time allows teachers to better understand the issues students face and potentially use the received feedback to improve their instruction in real time.

3.1. Class interaction apps specifics

The specificities of class interaction apps can be broadly described as follows:

Users. In general these apps have two types of users, a tutors (sometimes more than one) and students (between 20-500) with mobile phones with typical Wi-Fi and Bluetooth communication capacities with around a 30 meter communication range. Most students are not expected to be very mobile during the class and tutors are expected to move around the class in a relative slow walking pace.

Proximity. Users are likely to be co-located in a same classroom, or in the field. the surface is likely to be limited, namely below 1500 square meters (which is the size of the largest conference hall in our institution).

Interaction. In these applications any student is expected to post messages and interact with messages from others in real time. The messages can be temporary just for a lecture or an event, but tutors sometimes also want to eventually persist them in order to reflect on the discussion in the class.

[7] https://firech.at

[8] http://www.theguardian.com/technology/2014/jun/24/firechat-updates-as-40000-iraqis-download-mesh-chat-app-to-get-online-in-censored-baghdad

[9] https://opengarden.com/sdk

3.2. Middleware requirements

These specificities, indicate that such interaction could be supported by a communication middleware offering a many-to-many abstraction (since many students are expected to send messages to each other) implemented on a MNP (since interaction is confined to a certain area where all users are located at a point in time) with multi-hop communication (since the area is likely to cover more than the communication range of the devices). Finally, the middleware should allow teachers to persist interactions on the cloud when an Internet connection is available.

4. Padoc – Middleware Solution

Padoc is a novel communication middleware to support multi-hop co-located mobile ad hoc social media application running on standard off-the-shelf iOS mobile devices (iPhones, iPods, iPads).

4.1. Service

Location-based multicast addressing service (LMA) [20], where the many are defined by a multicast group which can contain as little as one node (single receiver) or as many as all nodes in the network (full broadcast) and where both groups and messages can be restricted to a specified distance around a node. LMA is good theoretical candidate for a middleware interface for developer of co-located group interaction apps since (1) interactions can be geographically restricted, (2) LMA supports multi-hop communication, (3) LMA does not make assumptions on node positions and supports node mobility, and (4) LMA supports many-to-many messaging. However, until now LMA has only been implemented for network simulators and no implementation is yet available for off-the-shelf mobile devices. Padoc is the first effort to overcome this limitation. The main service provided to the application developer by the Padoc middleware are the following methods:

```
/* multicasts a message m to members of group located within a range r */
- (void) multicast:(Message*) m to:(NSString*) group within:(NSNumber*) r persisted:(BOOL*) bool;

/* joins a multicast group located within a range r */
- (void) join:(NSString*) group within:(NSNumber*) r persisted:(BOOL*) bool;

/* abandons the multicast group*/
- (void) leaveMulticastGroup:(NSString*) group;
```

Furthermore, it delivers messages to delegate object implementing the following PadocDelegate method:

```
/* callback when a message m is received by padoc */
- (void) padoc:(Padoc*) padoc didReceive:(Message*) m fromCloud:(BOOL bool) ;
```

The padoc object can be instantiated using four different communication strategies (i.e., PadocStrategyPushFlooding, PadocStrategyPushCBS, PadocStrategyPullFlooding, PadocStrategyPullCBS) using the following call:

```
/* instantiation of the padoc middewlare object */
- (id) initWithDelegate:(PadocDelegate*) delegate usingStrategy:(PadocStrategy*) strategy;
```

4.2. Architecture

The architecture of the implementation of the service is depicted in Figure 1. Padoc provides a MNP module and a Cloud module. The MNP module is in charge of disseminating messages in the proximity using ad hoc technology whereas the Cloud module is used to forward information to the cloud when an Internet connection is available and a message requires persistence. Note that this paper focuses on the MNP module. In the MNP module, multicast can be implemented using a pull or push strategy. Since research indicates that none of these strategies is superior for all network topologies and all application scenarios [20], Padoc offers both implementation strategies for multicast. The *Push* strategy can be tuned to use either Scoped Flooding (PadocStrategyPushFlooding) or a Scoped Counter-based Broadcasting Scheme, dubbed PadocStrategyPushCBS, as underlying multi-hop message diffusion protocol. The *Pull* strategy on the other hand is built on to of a Gradiented Routing (GR) protocol for its multi-hop message diffusion. GR uses a warm up phase that can also either rely on Flooding (PadocStrategyPullFlooding) or on CBS (PadocStrategyPullCBS). These services are built on top of a Connection handler which interfaces with Apple's Multipeer Connectivity Framework, a technology providing one-hop ad hoc message diffusion and managing the physical layer encapsulation.

4.2.1. MultipeerConnectivity Framework

The MultipeerConnectivity Framework[10] is a readily usable library provided by Apple in its iOS

[10] https://developer.apple.com/library/prerelease/ios/documentation/MultipeerConnectivity/Reference/MultipeerConnectivityFramework/index.html

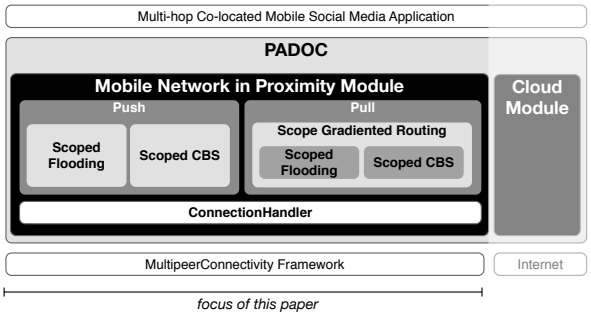

Figure 1: Padoc architecture

SDK since iOS 7. It allows to discover neighboring nodes and communicate with them when the app is active. Neighboring nodes are all nodes located in the transmission range of a node. A communication with a neighbor is called one-hop since it does not require that messages are relayed by intermediate nodes. Note that the ad hoc technology used in this framework is either Wi-Fi or Bluetooth, but is made transparent to users. The limitation of the MultipeerConnectivity framework include, only one hop communication, only a one-to-one messages primitive, connections between neighbors that require manual user confirmation, no background mode. Padoc aims at providing a lightweight layer on top of the Multipeer Connectivity framework to overcome these limitations.

4.2.2. ConnectionHandler

Padoc provides a small wrapper above the MultipeerConnectivity framework called the ConnectionHandler which handles connections with neighbors when their app is in both active or background modes. With the ConnectionHandler, a device automatically detects other devices in its transmission range and establishes a connection without the need for manual intervention from the user. This layer thus allows to loosen the MultipeerConnectivity's constraint, that requires nodes in the network to have their apps in active mode at all time to allow messages to be sent and delivered. This layer simply triggers a task in the background that continues to listen for messages and delivers them to upper layers.

4.2.3. Multi-hop message diffusion

Whereas the Multipeer Connectivity allows to communicate with neighboring nodes, Padoc provides facilities to communicate with a node's extended neighborhood, i.e., all nodes that are reachable through a chain of intermediate nodes that forward the message in what is called a multi-hop communication. Padoc implements three basic multi-hop message diffusion services, namely flooding, counter-based broadcasting and gradiented routing.

Scoped flooding. Scoped Flooding [29] is a very simple one-to-all (broadcast) message diffusion scheme limited to a certain range around the sender.

It works as follows: When Bob receives a message m sent by Alice, Bob increases forwards m unless he has already forwarded it previously or unless he is located at a distance of Alice greater than $m.r$. Note that $m.r$ is the range defined by Alice. Note that the distance is either calculated based on the GPS location of both users if available or it is based on a hop count approximation. note that such approximation can be made on observed values of the transmission range, but are subject to potentially large variations depending on the terrain. This protocol implies that all nodes in a non-partitioned network forward m once. This protocol has the advantage of being very simple and achieve good reliability in a non-partitioned network with little message collisions.

The problem with this algorithm is that as all nodes retransmit a message, the algorithm itself, depending on the number of nodes, the network topology, and the application usage, might cause message collisions in what is referred to as a broadcast storm [29]. The number of forwarders in Flooding can be reduced by requiring each message receiver to roll a dice before deciding to forward m. This reduces the number of forwards predictably (depending on the dice rule), but can result in lower reliability as critical nodes (for example those located in between two network partitions) might decide not to forward m, which implies that some peers may not receive it.

Scoped counter-based broadcasting scheme. The Scoped Counter-based Broadcasting Scheme (CBS) [29] provides a simple, yet effective [10] mechanism to address the broadcast storm issue, by reducing the message forwarding load of a broadcast algorithm while mitigating the reliability issue, while not entirely resolving it. In CBS, when Bob receives m from Alice through the Connection Handler, he sets a timer, and when the timer is up, he only forwards m if he has not received a copy of m from one of his neighbors in the meantime. This

mechanism is particularly effective in reducing the forwarding load in dense networks, where it is most needed.

Scoped gradiented routing. The Scoped Gradiented Routing (GR) protocol is a scoped version of the Six-Shot Multicast [11] and is a proactive route-based one-to-many (multicast) algorithm. It allows to send messages to a subset of nodes in the network. To do so, it goes through a proactive warm up phase in which a routing table is created, and then messages are disseminated to destinations in the routing phase.

The warm up phase works as follows. When Bob joins a multicast group, a message is broadcasted using CBS to the network. This message contains the indication that Bob is interested in receiving messages intended for the group. It also contains an information about the proximity of Bob in terms of hops. So if Carol is located in Bob's neighborhood she will receive a message indicating that Bob is at a 1 hop distance. Alice, who is located 5 hops away from Bob, will receive the indication of that distance. In this phase, either scoped flooding or scoped CBS can be used.

During the routing phase, when Alice sends a message to the group, the algorithm uses the directional acyclic graph (DAG) created in the warm up phase to route messages downhill from Alice who is located at a 5-hop distance toward nodes located lower on the gradient eventually reaching Bob at the bottom of the virtual valley. To do so the algorithm has the following rule. When Alice sends the message to group, it includes its proximity to each of the nodes in the group. Then when her neighbors receive the message they check if they are closer to Bob than Alice (i.e., they have a lower number of hops towards Bob). If they are closer, they set a timer very much like in CBS, and if they did not receive the message from another peer before the timer is elapsed, they replace the distance between Alice and Bob by their own in the message and forward it.

4.2.4. Location-based Multicast Addressing

The multicast service in Padoc offers two implementation strategies, a Push and a Pull strategy similarly to the strategies presented in [20] in the context of Location-based Multicast Addressing or related work (e.g., [22, 31, 19]).

Push Strategy. The Push mutlicast strategy uses either Flooding or CBS as underlying layer. In this scheme, messages are always sent to all nodes in the network. When a node receives a message, it only decides to deliver it if the node is part of the multicast group to which the message is addressed. Thus, when a node joins a group, no messages are exchanged with other nodes.

Pull Strategy. The Pull strategy is based on the GR protocol and thus broadcasts messages to all when a node joins a multicast group and then routes messages only to interested nodes.

5. Middleware evaluation

The goal of the middleware evaluation was to assess the how the Push and Pull strategies compare to each other. For the Push strategy, we measured evaluations for both the implementation using scoped flooding (PushFlooding) and scoped CBS (PushCBS). For the Pull strategy, we only measure the routing phase, which is the same no matter if the warm up strategy uses Flooding or CBS. Note that we do not take into account the cost for the routing table creation. Hereafter we present performance evaluation settings and the results of Padoc evaluations on real iOS devices with MNP conditions (no simulation, nor emulation of location).

5.1. Settings

Hereafter, we detail the settings of the evaluation, namely the type of devices, network topologies, messaging load and measures used.

5.1.1. Devices

The devices used throughout the experiments were 10 5th generation iPod Touch from Apple, 8 of them running iOS 8.2 and the remaining 2 running 8.3. These devices support both Wi-Fi and Bluetooth for communication. For consistency reasons, during the experiments only the Wi-Fi technology has been turned on. Devices had an observed transmission range of approximately 30 meters. The transmission throughput was approximately 100 KB/s.

5.1.2. Network topologies

Devices were placed according to two network topologies covering extreme cases and one topology mimicking a small field class. The network topologies are the result of the actual physical placement

of the devices. For each topology, 12 experiments have been carried out, giving thus a total of 36 different configurations.

Cluster topology. The first network, shown in Figure 2, is a fully connected cluster where all devices were placed in a 2 meter radius so that they were all within a one-hop reach. In the other topologies, since the observed transmission range of the devices was 30 meters, they were placed in such a manner to build the desired topology. Note that with the ConnectionHandler, a device automatically detects other devices in its transmission range and establishes a connection.

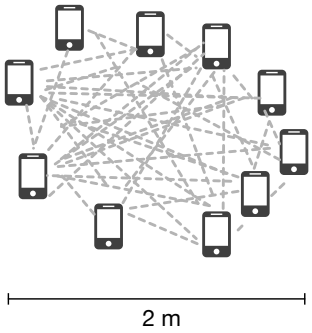

Figure 2: Cluster topology

Line topology. The second topology, shown in Figure 3, corresponds to a straight line where a device is only connected at most with 2 other devices.

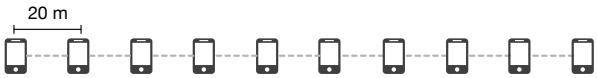

Figure 3: Line topology

Small field class topology. The final topology, shown in Figure 4, mimics a small field classroom with 8 fixed nodes representing students and 2 mobile nodes representing teachers who move through the class at walking speed (approx 1 meter per second). The teachers trajectory is depicted on the figure using arrows. Here only half of the motion is shown. The other half is symmetrical with respect to an imaginary horizontal plane.

5.1.3. Messaging load

In all scenarios, we ran the experiments for 2 minutes and all nodes acted as senders. We varied mes-

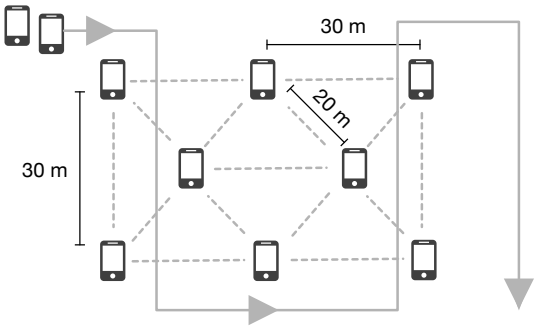

Figure 4: Small field class topology

sage loads along two dimensions, namely the size of the multicast group and the throughput.

Multicast group size. The multicast group size represents the number of nodes interested in delivering a message. For the experiments, we created a dedicated multicast group called padoc-experiment. At the beginning of the experiment a certain number of devices were set to join this group, by calling the join method. We evaluated two extreme cases of multicast group sizes, namely a *full broadcast* setting where all devices joined the multicast group and a *single receiver* setting where only one node joined the multicast group.

Throughput. We evaluated a *mild throughput* setting with 30 messages per minute per sender, and a *high throughput* setting with 120 messages per minute per node. Thus, the number of messages sent in 2 minutes totals 600 in the mild throughput setting and 2400 in the high throughput setting (for the broadcast setting). Note that the messages were sent automatically.

5.1.4. Measures

In order to compare the different implementation strategies and algorithms, we have measured two variables:

Delivery rate. The delivery rate is the ratio between the number of nodes having received a particular packet divided by the number of nodes supposed to receive it. These ratios are then averaged over all packets. This is a measure of effectiveness. A high value indicates more reliability and it should always equal 1 when the retransmission algorithm is reliable.

Retransmission rate. The retransmission ratio of a message is the number of nodes having retransmitted it divided by the number of nodes in the network. The ratios are then averaged over all packets. As a retransmission is costly in terms of energy and bandwidth, the retransmission rate is a measure of efficiency, a low rate indicates more efficiency. It should be noted that this rate also potentially affects delivery rate, since an inefficient retransmission rate can result in message collision, packet losses and thus lower delivery.

5.2. Results

Hereafter, we detail the evaluation results for each topology. The results are presented in Figures 5-7.

5.2.1. Cluster topology results

The retransmission ratio is equal to 1.0 for PushFlooding as expected, since all nodes which receive a message for the first time will forward it. PushCBS is much more efficient as it provides a retransmission ratio of around 0.1 for the mild throughput setting and 0.2 for the high throughput scenario. Finally, the retransmission ratio of Pull is equal to zero as no node is located closer to the recipients than the sender itself. Note that the mean hop count for every configuration is 1.0 as expected, because every node is at a one-hop distance from any other node. In this setting, the delivery rate of PushFlooding slightly higher than the other strategies and is equal to 1.0 in mild throughput conditions and drops to 0.98 and 0.93 in high throughput. The delivery rate of Pull is the lowest in this setting and varies from 1 to 0.91. A delivery rate below 1.0 means that some messages did not reach their destination. This can be due to message collisions or connection interruptions.

5.2.2. Line topology results

The retransmission ratio is equal to 1.0 for both PushFlooding and PushCBS as expected, since all intermediate nodes are required to relay messages. In this topology, the Pull strategy allows to save between 7% and 20% for the full broadcast and between 35% and 37% for the single broadcast scenarios since at least the last nodes of the line can decide not to forward the message. Nevertheless, the delivery rate for the Pull strategy scores worse than the Push strategies in this setting (0.88 to 1.0 compared to 1.0 in all cases).

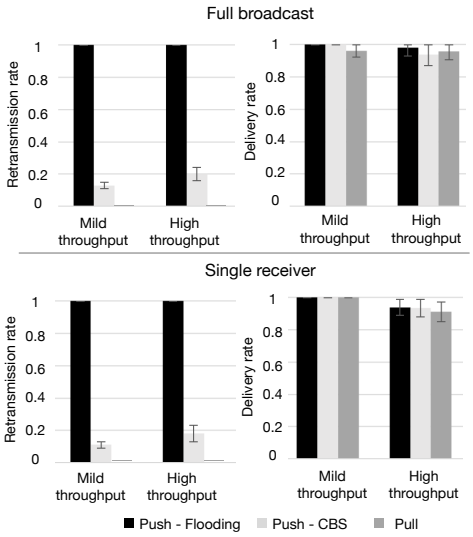

Figure 5: Evaluation results for the Cluster topology

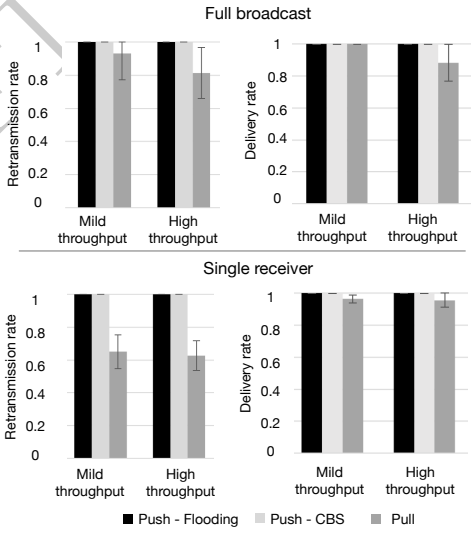

Figure 6: Evaluation results for the Line topology

5.2.3. Small field class topology

For the full broadcast configuration, PushCBS saves between 5% and 9% of messages compared to PushFlooding and Pull saves between 11% and 14%. For the single receiver configuration, Pull is even more efficient and imposes only a retransmission rate of 0.16 to 0.18. In this topology, the delivery rate is the highest with 8 configurations out of 12 scoring above 0.99%.

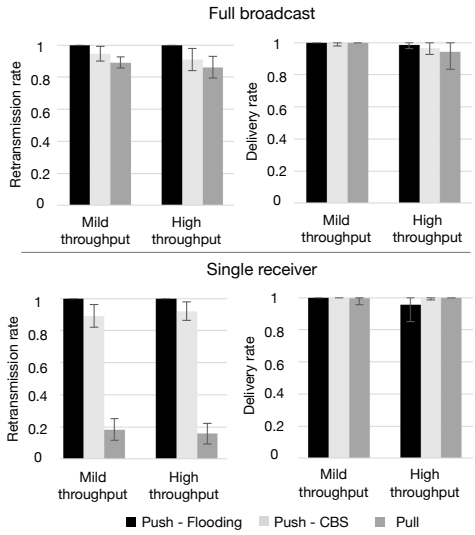

Figure 7: Evaluation results for the small field class topology

5.2.4. Overview

In terms of effectiveness, the different strategies provide good reliability, but can be vulnerable to message collisions in high throughput scenarios. Note that the warm up phase of the Pull strategy is not immune to message loss, which can make its routing phase less reliable than Push based strategies in some cases (especially in broadcast scenarios, where all nodes join the multicast group). To improve effectiveness, an additional layer could be built on top of the routing algorithm to ensure reliability by for example requesting acknowledgement messages.

When comparing PushCBS and PushFlooding, the former can sometimes be slightly less effective (lower delivery ratio), as it does sometimes prevent some nodes from forwarding messages to receivers. For instance in our evaluation we observed a drop of about 10% in distribution ratios in the cluster and small field class topologies when the message throughput was high. In terms of efficiency, when comparing the two Push implementations, PushCBS is more efficient than PushFlooding in most scenarios. This is especially true when the network is highly connected, as in the cluster topology, where we observed a decrease in message load by a factor of 10x.

The Pull strategy is superior to both Push strategies in terms of efficiency because messages are not propagated in parts of the network where no receiver is located. This difference is especially marked when the number of receivers is low. However, the evaluation does not take into account the warm up phase of the Pull strategy. Note, that the cost of the warm up phase depends on the chosen refresh rate and the number of group members and can thus vary substantially. Nevertheless, we can estimate that one warm up phase, will cost the same as one broadcast per receiver. So it will not cost much in efficiency with a low number of receivers, where Pull will remain the best strategy, but it will put a larger burden on efficiency with many receivers. In such cases, the PushCBS will be the more efficient option.

6. Heya – proof-of-concept

Here we present Heya which is a novel classroom interaction app built on top of Padoc as a proof of concept. Heya is based on the SpeakUp app for co-located social media interaction in the classroom [21]. Whereas SpeakUp relies on a client-server architecture suitable in classrooms with access to broadband Internet, Heya works without any external infrastructure and is suitable for field classes that do not have access to any connectivity. Hereafter, we describe the user interface of the app, then we describe and discuss possible implementation architectures of Heya using Padoc.

6.1. Functionality

Heya is an application that allows people to create chat rooms or join existing ones, write posts in those rooms and rate people's posts. Figure 8 shows the user interface of Heya. The image on the top left shows the list of rooms available in the network in proximity. To create a new room, users, typically tutors, can press the plus button, which leads to the screen depicted on the top right of Figure 8. There, users can define the name of a new room which will follow them around. A user can also choose to persist a room. The idea is that when a room is persisted, its content will eventually be uploaded to the cloud. Tutors will then typically be able to reflect on the exchanges that occurred during their session. Students located in the vicinity of the tutor will see the newly created room and will be able to access it by simply pressing the room name in the list of nearby rooms, as depicted in the image at the top left of Figure 8 – no need for a password or registration. As shown in the image at the bottom of Figure 8, they can then see the

existing messages in that room and they can vote on them by pressing the thumb up or thumb down buttons. Users can also post messages anonymously through the input field at the bottom of the screen.

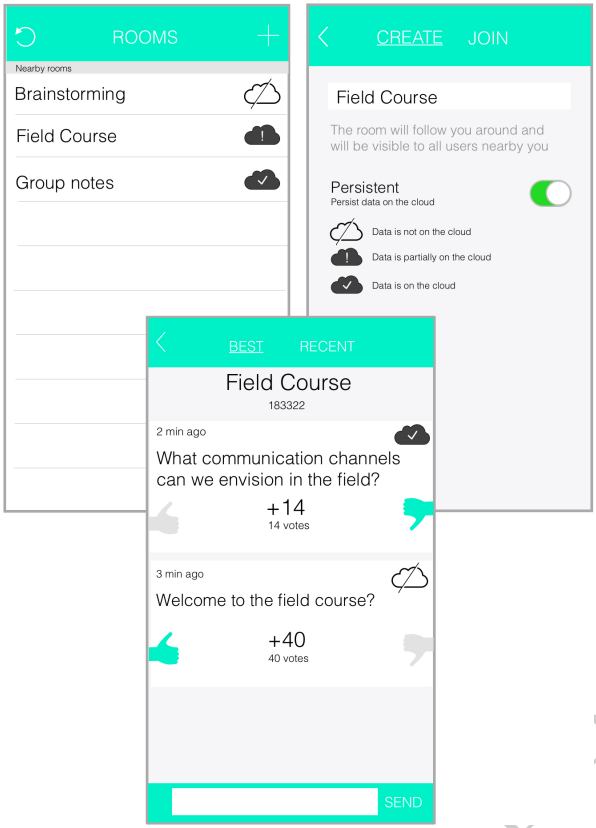

Figure 8: User interface of Heya.

6.2. Implementation using Padoc

There are several possible implementation strategies for Heya using Padoc.

One approach is fully decentralized and delegates computation of the state to all users. This implies potential inconsistencies as certain users might receive some posts before others or even possibly not receive at all some posts.

Another option is to delegate computation of state to one user which then relays the state of the app. This method has the advantage of keeping a coherent state, but has the disadvantage of relying on the presence of a coordinator throughout the interaction.

For the implementation of Heya, we chose this second option as it better mimics the actual application scenario of Heya, which implies a de facto coordinator – the tutor. Note that we chose a PushFlooding strategy for Heya for simplicity.

Creating a room. When a tutor, Alice, creates a room, a roomcreated message containing the new room is sent to the all group, and everyone can add this room. When Alice deletes a room a roomdeleted message containing the room id is sent to the all group. All peers receiving this message will then delete the room. Alice can decide to persist a room or not. In the case she decides to persist it, all calls will set the persisted flag to YES and the middleware will send the room and its content to the cloud when an internet connection is available.

Discovering nearby rooms. When a student, Bob, opens his app, he joins the all group and a group based on Bob's unique peer id bob and a getrooms message is sent to the all group, scoped to 200 meters around Bob[11] and persisted, which means that Bob is both interested in receiving rooms from the cloud and from the ad hoc network in proximity. When Alice or another room creator located within 200 meters, receives such a message, she sends a rooms message to the bob group containing the list of all rooms that she has created. When Bob receives a rooms message, he adds the received rooms to his list of nearby rooms (if they are not already present).

Entering a room. When Bob enters the fieldcourse room created by Alice, a *getroom* message containing the id of the room is sent to the alice group, Alice is then solely in charge of sending the entire list of posts in a room message sent to the bob group. As before, Bob joins the multicast group defined by the fieldcourse room id.

Creating a post. When Bob creates a post in the fieldcourse room, a postcreated message containing his new post is sent to the alice group. When receiving this message, Alice will then send a postcreated message to the fieldcourse group, and other peers will add the post to their list of posts for this room. In the same way, when Bob deletes a post, a postdeleted message containing the post id is sent to the alice group and Alice then forwards the deletion to other peers that will delete the message locally.

[11] Note that this scope is an arbitrary application choice.

Voting on a post. When Bob votes on a post in the fieldcourse room, he sends a postupdated message containing his vote and the post id to the alice group. When receiving this message, Alice will then compute the number of votes and the new score for the room and send postupdated message to the fieldcourse group.

6.3. Tales from the field

In order to evaluate the usability of Heya in the field, we conducted evaluations in the Maputo province in Mozambique in the context of a Médecins Sans Frontières (MSF) mission. We performed three 6-hour hands on sessions with in-depth interviews with one expat NGO workers and 8 national staff of MSF, age 26-56, (mean 41, 1 female). Note that two of the national staff owned iPhones. One of the national staff and the researcher were present in all three sessions, thus five people were present in the first two sessions and four in the last session. Each session started with a 30 minute introduction followed by a free and a guided interactions to test the application's functionalities. In the free interactions, users posted messages and rated them as illustrated in Figure 9. In the guided interaction, the users moved away from each other to test the limit of the transmission range and to test multi-hop communication. Typically ranges up to 120 meters where tested with different network topologies. After the hands on sessions which lasted around 2h30 hours, the users sat down with the researcher for an in depth debriefing with semi guided interview questions and users were also asked to answer to a formal System Usability Scale (SUS) [2] and Attrakdiff 2 [17] questionnaire for a so called quick and dirty usability evaluation. The SUS score of Heya was 80.5 indicates good to excellent usability according to Bangor et al. [2]. This result is inline with the SUS of SpeakUp (83) as evaluated in [21]. The results of the Attrakdiff also show strong positive attitudes towards the applications with maximal values for all three dimensions of the scale (attractiveness, hedonic quality, pragmatic quality). Note that such positive results can also indicate bias due to the convenience sampling method.

Nevertheless, these results were echoed in the in depth interviews. Users did not know about the possibility of ad hoc communication prior the hands on sessions. They were positively surprised by the fact that the app was available on the regular app store. The limits in terms of transmission range did not appear to be an issue for them. One of the main advantages that users saw was the reduction in communication costs. Indeed, many of them cannot afford to pay for unlimited plans, so they use prepaid cards which they use up quickly, which prevents them from being reachable. Furthermore, they also saw the advantage of being able to communicate despite potential government censorship. The government would order the network operators to switch off their service in case of demonstrations for example. Also users, especially those with lower socio-economic status seemed suspicious of the government and were happy to "play the system" by using a parallel communication channel.

Figure 9: Hands on session with Heya in Mozambique

7. Discussion and conclusion

In this paper we have observed that there is a lack of mainstream computing solutions for multi-hop co-located ad hoc social media interactions in settings with scarce connectivity even though such interaction could have important applications, for instance in terms of knowledge sharing and education. To address this issue, we have designed a novel communication middleware called Padoc for off-the-shelf iOS devices. Note, that there is an obvious limitation to using the iOS platform as target since the devices running iOS are top end products and thus even if they are mainstream in developed countries they are much rarer in developing countries where connectivity can be more limited. The rationale behind the choice of iOS resides in the fact that at this stage it is the best candidate for multi-hop communication. Thus we see the iOS library as first proof of concept of multi-hop communication middleware on a commercially available platform. Such a first proof-of-concept allows to build and evaluate end user application that could not

be built otherwise and it is only when such applications are built and deployed that we can get insights about usability and usefulness.

We then carried out real world performance evaluation of the different message diffusion strategies offered in Padoc (i.e., PushFlooding, PushCBS, and Pull) using 10 iPod Touch devices in three different network topologies and several different messaging scenarios. Our results show that these strategies offer good reliability and show that Pull offers the best efficiency, especially in settings with a low number of receivers, where it can reduce the message load by orders of magnitude. However Pull appears to be somewhat less reliable (5%-10%) than the other strategies. PushCBS offers a good alternative for settings with many receivers. The process of gathering performance evaluation from actual mobile devices without the help of location and connection emulators also highlight the logistical difficulty of performing such real life tests given the lack of dedicated testbed tools. Indeed, even though experiments have become more prominent in the community [24], most of the open available testbeds are sensor oriented [14] and few include smartphones (an exception includes SmartSantander [27] which provides a SmartCity testbed) and to the best of our knowledge none of them is targetted at off-the-shelf mobile devices on which people centric social media in proximity would run. Thus, we argue that this is an important research gap, which needs to be addressed in order to support larger scale validation of communication middleware and services for mobile social newtorking in proximity.

In this paper we also presented Heya, a novel colocated classroom interaction system built on top of Padoc as proof-of-concept. As mentionned, the Padoc middleware, with its scoped messaging services is particularly well suited for this application scenario. We believe that Padoc can also be useful for other applications for social networking in proximity, however we do not see it as a magic bullet for any cyber-physical application. We argue that providing a targeted middleware can be a strength rather than a weakness. Indeed, using a people-centric approach to middleware design, as promoted by Conti and Giordano [6] means understanding application usage in a particular context and designing communication accordingly. Such an approach is used by Socievole et al. [28] who use information from online social media apps to predict physical encounters and thus adjust their opportunistic routing protocol accordingly. Future work should further investigate such opportunistic and people-centric middleware design. This is particularly relevant in cyber-physical systems where context is inherently tied to application logic.

Finally, we perform a usability field evaluation of Heya on a mission with MSF, which confirms the interest for such technology in places where communication access cannot be taken for granted, whether by lack of infrastructure, cost or censorship. The small scale of this evaluation, taken with potential biases of the convenience sampling, limits the generalizability of the results and should be taken like a first step of a larger usability study. Nevertheless, it provides a first concrete deployment example on the field of an app for mobile social networking in proximity.

Our evaluation process has however also highlighted the challenges to evaluate both ad hoc communication protocols and applications in real conditions and future work should investigates new methodologies and tools for testing off-the-shelf cyber-physical systems.

Acknowledgement

This research is partly funded by the Hasler Stiftung.

[1] A. Acar and Y. Muraki. Twitter for crisis communication: lessons learned from japan's tsunami disaster. *International Journal of Web Based Communities*, 7(3):392–402, 2011.

[2] A. Bangor, P. T. Kortumb, and J. T. Millerc. An empirical evaluation of the system usability scale. *Int. J. Hum. Comput. Interaction*, 24(6):574–594, 2008.

[3] J. Birnholtz, J. Hancock, and D. Retelny. Tweeting for class: co-construction as a means for engaging students in lectures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 797–800. ACM, 2013.

[4] A. Chaudhuri. Ict for development: solutions seeking problems&quest. *Journal of Information Technology*, 27(4):326–338, 2012.

[5] N. Cohen. Hong kong protests propel firechat phone-to-phone app. *The New York Times*, 2014.

[6] M. Conti and S. Giordano. Mobile ad hoc networking: Milestones, challenges, and new research directions. *IEEE Communications Magazine*, pages 85–96, January 2014.

[7] E. da Silva and L. C. P. Albini. Middleware proposals for mobile ad hoc networks. *Journal of Network and Computer Applications*, 43:103–120, 2014.

[8] M. Danieletto, G. Quer, R. R. Rao, and M. Zorzi. On the exploitation of the android os for the design of a wireless mesh network testbed. In *Military Communications Conference, MILCOM 2013-2013 IEEE*, pages 1032–1038. IEEE, 2013.

[9] H. Gao, G. Barbier, and R. Goolsby. Harnessing the crowdsourcing power of social media for disaster relief. *IEEE Intelligent Systems*, (3):10–14, 2011.

[10] B. Garbinato, A. Holzer, and F. Vessaz. Context-aware broadcasting approaches in mobile ad hoc networks. *Computer Networks*, 54(7):1210–1228, 2010.

[11] B. Garbinato, A. Holzer, and F. Vessaz. Six-shot multicast: A location-aware strategy for efficient message routing in manets. In *Network Computing and Applications (NCA), 2010 9th IEEE International Symposium on*, pages 1–9. IEEE, 2010.

[12] P. Gardner-Stephen. The serval project: Practical wireless ad-hoc mobile telecommunications. *Flinders University, Adelaide, South Australia, Tech. Rep*, 2011.

[13] P. Gardner-Stephen, R. Challans, J. Lakeman, A. Bettison, D. Gardner-Stephen, and M. Lloyd. The serval mesh: A platform for resilient communications in disaster & crisis. In *Global Humanitarian Technology Conference (GHTC), 2013 IEEE*, pages 162–166. IEEE, 2013.

[14] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo. A survey on facilities for experimental internet of things research. *Communications Magazine, IEEE*, 49(11):58–67, 2011.

[15] G. Grotenbreg and S. B. J. Wong. Using Pigeonhole® Live to elicit feedback, questions & reinforce learning during lectures. *CDLT Brief*, 16(2):2–7, Aug 2013.

[16] D. Harry, J. Green, and J. Donath. Backchan.nl: integrating backchannels with physical space. In *CHI'09*, pages 2751–2756. ACM, 2008.

[17] M. Hassenzahl, M. Burmester, and F. Koller. Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität. In *Mensch & Computer*, pages 187–196. 2003.

[18] R. Heeks. Ict4d 2.0: The next phase of applying ict for international development. *Computer*, 41(6):26–33, 2008.

[19] A. Holzer, P. Eugster, and B. Garbinato. Alps–adaptive location-based publish/subscribe. *Computer Networks*, 56(12):2949–2962, 2012.

[20] A. Holzer, P. Eugster, and B. Garbinato. Evaluating implementation strategies for location-based multicast addressing. *Mobile Computing, IEEE Transactions on*, 12(5):855–867, 2013.

[21] A. Holzer, S. Govaerts, A. Vozniuk, B. Kocher, and D. Gillet. Speakup in the classroom: anonymous temporary social media for better interactions. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 1171–1176. ACM, 2014.

[22] R. Meier and V. Cahil. On event-based middleware for location-aware mobile applications. *Software Engineering, IEEE Transactions on*, 36(3):409–430, 2010.

[23] N. Morrow, N. Mock, A. Papendieck, and N. Kocmich. Independent evaluation of the ushahidi haiti project. *Development Information Systems International*, 8:2011, 2011.

[24] G. Z. Papadopoulos, K. Kritsis, A. Gallais, P. Chatzimisios, and T. Noel. Performance evaluation methods in ad hoc and wireless sensor networks: a literature study. *Communications Magazine, IEEE*, 54(1):122–128, 2016.

[25] T. S. Project. Serval mesh extender. `http://developer.servalproject.org/dokuwiki/doku.php?id=content:meshextender:main_page` (accessed in June 2015).

[26] V. Raychoudhury, J. Cao, M. Kumar, and D. Zhang. Middleware for pervasive computing: A survey. *Pervasive and Mobile Computing*, 9(2):177–200, 2013.

[27] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, et al. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61:217–238, 2014.

[28] A. Socievole, E. Yoneki, F. De Rango, and J. Crowcroft. Ml-sor: Message routing using multi-layer social networks in opportunistic communications. *Computer Networks*, 81:201–219, 2015.

[29] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2-3):153–167, 2002.

[30] D. Yates and S. Paquette. Emergency knowledge management and social media technologies: A case study of the 2010 haitian earthquake. *International Journal of Information Management*, 31(1):6–13, 2011.

[31] E. Yoneki and J. Bacon. Distributed multicast grouping for publish/subscribe over mobile ad hoc networks. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 4, pages 2293–2299. IEEE, 2005.

[32] D. Zhang, D. Zhang, H. Xiong, C.-H. Hsu, and A. V. Vasilakos. Basa: Building mobile ad-hoc social networks on top of android. *Network, IEEE*, 28(1):4–9, 2014.

[33] E. Zuckerman. Decentralizing the mobile phone: A second ict4d revolution? *Information Technologies & International Development*, 6(SE):pp–99, 2010.