A Smartphone-Targeted Opportunistic Computing Environment for Decentralized Web Applications

Lionel Touseau, Yves Mahéo, Camille Noûs

A Smartphone-Targeted Opportunistic Computing Environment for Decentralized Web Applications

Lionel Touseau[†*‡]
[†]CREC, Saint-Cyr Coëtquidan Military Academy
Guer, France
lionel.touseau@st-cyr.terre-net.defense.gouv.fr

Yves Mahéo[*‡]
[*]IRISA, Université Bretagne Sud
Vannes, France
yves.maheo@univ-ubs.fr

Camille Noûs[‡]
[‡]Cogitamus Laboratory, France
camille.nous@cogitamus.fr

***Abstract*—Providing smartphone users with decentralized web applications is an actual trend, but making web applications run in a decentralized though collaborative way in real conditions, with connectivity disruptions, is a challenging task. Opportunistic networking offers a way to be independent from a fixed infrastructure and cope with intermittent connectivity by leveraging both the mobility of nodes and their transient radio contacts. Smartphones' operating systems however remain restrictive as far as device-to-device communication is concerned. In this paper we present an environment that facilitates the development of web applications deployed in opportunistic networks built out of smartphones. We first describe the externalization of the opportunistic communication support to a dedicated handheld device. Then we present the implementation of a programming model called foglet that allows web browsers to share consistent data structures. The feasibility of the approach is shown through experiments performed both on the field and in an emulation platform.**

***Index Terms*—opportunistic networking, mobile browsers, decentralized web, foglet**

I. Introduction

The last decade has witnessed two major trends in software development: software decentralization, and the shift towards web and mobile applications. Individual users have got used to software and data being synchronized and available anywhere, on multiple devices. Highly available software hosted on remote cloud infrastructures has gradually become the prevailing model, thanks to the popularity of web standards and technologies. Users only need a web browser to access a myriad of applications.

Yet, this trend is being more and more challenged. End users are indeed increasingly sensitive to privacy issues and less inclined to entrust their data to digital giants. Recent practices consequently tend to step away from centralized remote software hosted by third-party authorities, and favor self-hosting solutions, decentralized approaches such as blockchain-based systems or local-first software in order to regain ownership and control on the code and their data over private corporations or governments.

Cloud hosted software nevertheless offers some advantages, like collaboration and interoperability, that users would like to keep. While interoperability can be easily achieved using standardized web protocols and cross-platforms web browsers, a decentralized solution should however support multi-user collaboration.

This paper focuses on web applications specifically used in a mobile context. Smartphones have gradually become digital extensions of ourselves that help us cope with a digitizing world (economy, healthcare, education...). Without debating the questionable societal aspects, we have to recognize this trend as an established fact. Studying web-based applications therefore implies to consider thin clients running on mobile devices, i.e., on smartphones' web browsers.

Such applications are usually dependent on an Internet access provided by a network infrastructure. In software design, it is often assumed that the developed system can rely on end-to-end connectivity and a permanent access to the Internet. In practice this assumption is not always true, even more for systems involving mobile devices that depend on wireless network infrastructures (Wi-Fi access points, 4G or 5G networks...). Mobile use of these infrastructures may result in a varying network availability. Unreliable Wi-Fi hotspots, poor or partial network coverage in rural areas, or trying to access a remotely hosted application while traveling, are reasons that prevent users from using their application or that downgrade the quality of their experience.

Web applications do not usually handle very well offline execution, causing connection timeouts or "trying to reconnect" error messages. Even when they do offer an offline mode, the remote server is considered the holder of the primary authoritative copy of data. The local version is not valid until commited to the server.

The work presented in this paper attempts to address the need for decentralization and infrastructure independence, knowing that decentralization in a web environment is not trivial since web technologies are traditionally centralized. The first challenge that we identify consists in enabling decentralized web applications without relying on network infrastructures. In order to meet this challenge our proposition stems from research on Opportunistic Networks (OppNets for short).

The Opportunistic Networking (OppNetting) research area focuses on a particular type of challenged networks : fragmented mobile ad hoc networks where connectivity is prone to disruptions. The opportunistic computing approach consists in leveraging the mobility and the ad hoc communication

capabilities of the nodes to develop applications that cope with network fragmentation. Following the "store, carry and forward" principle, the mobile nodes participating in an OppNet passively carry messages, and forward the stored messages to other nodes in their vicinity when contact opportunities arise. It allows messages to be spread throughout the whole network in a discontinuous manner with somewhat delayed communications.

Nodes in an OppNet must nonetheless provide ad hoc direct node-to-node communication capabilities. Even if off-the-shelf smartphones are devices of choice for mobile web clients due to their ubiquity, they are not so convenient for setting up an OppNet: the most commonly available communication interfaces, namely Wi-Fi and Bluetooth, cannot be used to their fullest, and particularly in an automatic ad hoc mode, due to technical limitations on common smartphone operating systems. This mainly explains why only few opportunistic systems have been actually deployed. In order to achieve an ad hoc network of mobile browsers in line with the opportunistic computing principles, a second challenge consists in circumventing this technological hindrance.

In the light of the above observations, this paper will focus on decentralized web applications distributed across a disruption-tolerant ad hoc network of mobile browsers. Our contribution builds on a work that provides smartphones with opportunistic communication capabilities, bypassing technological limitations, and on a programming model for developing decentralized web applications. We therefore propose a solution for web applications distributed over infrastructure-less networks, implemented on top of our opportunistic communication support for smartphones. This solution called *opportunistic fognet*, that has been validated both in a field experiment and in a larger scale simulation, will be presented following a review of existing works and techniques related to decentralized web applications and opportunistic computing.

II. Related Work

A. Decentralized web applications

Although the World Wide Web has shifted from static pages to dynamic web applications and collaborative user-driven content, it still relies on centralized technologies. Besides, even if the primary goal of a web browser, as its name suggests, is to browse the World Wide Web following hyperlinks, the technological evolution of modern web browsers, and their built-in rendering and Javascript engines, have turned them into web execution platforms. This kind of platform opened the way to the decentralization of web applications, which, in this context, are meant to be applications built on web technologies that run –ideally– solely on web browsers. The local-first initiative [13] emphasizes decentralization and local execution of software as guiding principles. This work led by Ink & Switch research lab does point out the challenge of peer-to-peer (P2P) communication for local-first and decentralized web applications in [23].

Some web technologies like WebRTC or WebSockets which support P2P communications, could be regarded as decentralization enablers allowing web browsers to directly communicate with one another. WebRTC is a peer-to-peer protocol which was primarily designed for audio and video calls and can also be used for P2P application data exchanges. Nevertheless, peer discovery and connection establishment rely on signaling servers and centralized protocols (ICE, STUN, and TURN), thus making WebRTC not applicable to the present work since it depends on a reliable Internet access. The WebSocket protocol provides an efficient communication channel between browser-based peers for carrying application data, but it neither manages peer discovery. Other initiatives support the development of web-based desktop applications, like the Electron framework [21], but they only cover the local execution aspect. In the end, none of these solutions qualifies natively for operating collaboratively off-grid.

Another challenge lies at application data layer. Any decentralized system that involves collaboration between peers must provide some support for data distribution. Although distributed non-relational databases or blockchain-based systems have gained in popularity, they are not tailored for our needs since the class of applications targeted in this article does not aim at storing large volumes of data.

Other research activities in the field of distributed computing, led by the work of Shapiro et al. on Conflict-free Replicated Data Types (CRDTs) [20], have focused on the consistency of distributed data structures. Some of these works have led to web-based implementations of CRDTs, like YJS [16] and an implementation of JSON CRDTs [12] called automerge. These CRDTs frameworks primarily focus on data structure merge operations, and therefore abstract the network layer.

Besides, other works have implemented decentralized web applications using CRDT-like shared data structures. CRATE [15] is a collaborative text editor whose network layer is based on WebRTC and SPRAY random peer sampling protocol for peer discovery. CRATE can be seen as a preliminary work that led to the concept of *foglet* as used in the foglet-core module [6]. These works however rely on WebRTC and signaling, which requires a network infrastructure with Internet access.

B. Opportunistic communication between smartphones

Opportunistic networking (OppNetting) is well suited to dynamic mobile networks that can operate without any network infrastructure. Research in OppNets has now been active for almost two decades [14], addressing off-grid communications in scenarios such as disaster relief, wild life sensing or military communications. Although research has considerably contributed to the definition of forwarding protocols aiming at minimizing the delivery delay and at optimizing the number of exchanged message copies, few real-world experiments have been conducted on effective OppNetting systems at a medium or large scale. One of the main reasons for this is the lack of proper technology support for ad hoc device-to-device communication on widespread handheld devices such as smartphones. So far, Wi-Fi chipsets cannot be used to their fullest due to technical limitations on common mobile operating systems, that hinders their use in ad hoc mode.

A few attempts on OppNets of smartphones have been made, either by *rooting* the operating system [10], using Bluetooth, Wi-Fi Direct [2] or turning some smartphones into access points [22]. But users are usually reluctant to *jail-breaking* their phone, and both Bluetooth or Wi-Fi direct require user confirmation for pairing, which prevents spontaneous interactions. As a consequence, it is an actual challenge to create OppNets with off-the-shelf smartphones and their built-in hardware and system.

Smartphones have often been used for gathering mobility traces [1], [11] that could be reused in OppNetting research investigating disaster relief scenarios. SmartRescue [17] proposed a web-based platform for smartphones to help first crisis responders, but even in a disaster relief scenario it still depends on a cellular network infrastructure, which does not comply with the requirements expressed in this paper.

C. Smartphones transmission capabilities extension solutions

As it is far from sure that device-to-device communication will be effective for all users in cellular networks, even with the coming 5G or 6G, and facing the technology hindrance for ad hoc transmissions with smartphones highlighted in the previous section, a few reasearch projects and commercial products considered the externalization of ad hoc communication.

Both goTenna [5] and bearTooth [3] provide devices that can be paired with a smartphone via BLE (Bluetooth Low Energy). These devices aim at extending the range of smartphones with long distance radio transmissions to cope with infrastructure-less environments. The low throughput limit these solutions to specific applications like short messages, GPS location sharing and PTT voice. Multi-hop mesh networking features are offered by goTenna [4]. The devices act as relays using a proprietary routing protocol, and cannot be used in sparse environments, since they lack the "store, carry and forward" capability of opportunistic networking.

The basic idea behind these solutions is to extend the smartphone with a peripheral device, with which it can interact (typically via a Bluetooth link), and which can be used as another transmission interface for the smartphone. This is an idea we will develop in the solution presented in the following sections.

III. Opportunistic Web Applications

The challenge tackled in the paper consists in building web browser-based applications that are able to operate offline in spite of connectivity disruptions, in a decentralized way. Our proposition is two-fold. We first propose a solution to enable opportunistic communications between mobile web browsers, bridging the gap between the most common web browser-enabled handheld devices, i.e., smartphones, and opportunistic networks, thanks to an external device. This solution acts as a backbone for our second contribution : a network-level Javascript module enabling decentralized web applications based on the *foglet* programming model.

A. Bringing opportunistic networking to smartphones

Our first contribution aims at bringing opportunistic networking features to an off-the-shelf Android smartphone, using a carriable external device. It has led to the definition of an architecture comprising a device called *Ligo*, a Bluetooth gateway application to connect a smartphone to a *Ligo* unit, and a protocol allowing remote access to the features of the middleware in charge of opportunistic networking. This subsection summarizes the key elements of the architecture that can be visualized in Figure 2.

1) Ligo device: In order to circumvent the hindered usability of the Wi-Fi transmission interface of smartphones, the *Ligo* device runs a Linux Debian system that enables the ad hoc mode of the Wi-Fi interface. The *Ligo* prototype ships a Raspberry Pi Zero W in a dedicated casing that also includes a rechargeable Li-Ion battery cell to offer a battery life similar to the one of a smartphone. In addition to its Wi-Fi module, the Pi Zero is equipped with a Bluetooth 4.1 chip. Wi-Fi is dedicated to the opportunistic network while the Bluetooth interface links *Ligo* to the smartphone. A more exhaustive description of *Ligo*'s hardware components can be found in [8].

2) DoDWAN and NAPI protocol: Ligo embeds the DoDWAN middleware [7] to manage opportunistic networking. A beaconing mechanism allows a DoDWAN instance to discover neighbors in its vicinity, sending UDP announcements to an IPv6 multicast group over a Wi-Fi channel. Once peers have discovered each other they can interact via TCP sessions.

DoDWAN is a content-driven dissemination middleware [9] that implements a form of controlled epidemics. It stores messages in a local cache and forward them to other DoDWAN nodes. If a message matches the interest profile of the receiving node, it is delivered to the application layer. Otherwise, it is stored to be potentially forwarded later. For this purpose, DoDWAN provides a publish/subscribe API to applications willing to send and receive messages over a DoDWAN opportunistic network.

A protocol called NAPI (for DoDWAN Network API) has been defined to provide the main features of DoDWAN, namely neighbor discovery and publish/subscribe, to remote applications. The DoDWAN NAPI-WS plugin is a WebSocket implementation of NAPI. It runs a WebSocket endpoint which listens to and processes NAPI commands issued by WS clients (i.e., the application layer). The endpoint uses the established WS channel to deliver messages to that application layer upon reception.

3) Bluetooth tunneling: Lastly, both *Ligo* and the smartphone run a Bluetooth gateway which allows TCP traffic on configured ports to be tunneled over a RFCOMM channel, provided that the two devices have been paired. A Bluetooth gateway app must be installed as an Android APK on the smartphone, and the other one is a Python script running on *Ligo*. The resulting Bluetooth tunnel allows a web client, i.e.,

a web browser on the paired smartphone, to access the NAPI WebSocket endpoint as if it were located on the smartphone.

4) Summary: The resulting architecture provides smartphones with opportunistic communication facilities which can be accessed from a WebSocket client. This feature will be used in the second part of our contribution to build decentralized web applications over opportunistic networks.

B. Opportunistic Foglets

The work presented in this paper also builds upon the *foglet* programming model. It follows on from work on shared data structures in web applications that defined the *foglet* as a piece of software executed by a web browser and gave forth to a network layer implementation for foglets called *foglet-core* [6]. Foglet-core is based on WebRTC and SPRAY random peer-sampling (RPS) protocol, which are not usable in an opportunistic network that does not provide permanent connectivity. Even so, we still consider that the foglet concept is a fitted approach to build on, let aside the WebRTC-based network layer, hence the proposed programming model.

1) Concepts:

Foglet: A foglet is an ephemeral cooperative distributed application ran by a web browser. The code is written in Javascript and may embed web resources (HTML, CSS, images...). Yet, the foglet should be minimalistic and as much self-contained as possible. Foglets are meant to enable applications distributed across browsers to operate offline, and should therefore not depend on third-party servers. This point stands even more if we want to achieve an infrastructure-less web of browsers. The distributed aspect of foglets relies on two main concepts: the foglet network and the data structures shared across that network.

Foglet network: A foglet network, also shortened as *fognet*, represents the network layer of a foglet. It accounts for browser-to-browser communication between instances of a given *foglet*. The fognet interconnects instances of a single foglet into a community. As a result the foglet's life-cycle is bound to the state of the fognet. A foglet is meant to be a short-lived application. Once there is no foglet instance left participating in the fognet, the foglet ceases to exist. A fognet peer provides communication primitives (e.g., broadcast or unicast) to the foglet instance it is associated with, and notifies it when messages are received. It also reflects its awareness of other fognet peers by notifying the foglet instance when other instances join or leave the network.

Shared data structures: Foglets use shared data structures to enable a collaborative and decentralized execution. These structures hold data shared between the foglet instances and are responsible for ensuring concistency across those instances. Ideally, a foglet should be able to manage several consistency models, from strict consistency to various weak consistencies. This paper will present an example of foglet that implements a form of eventual consistency. To this end, data structures are replicated on each foglet instance, and modifications to the data are propagated to the participating nodes using the foglet network.

2) Foglet programming model: Foglets must follow some key principles in respect with their programming model. The sequence diagram on Figure 1 illustrates the main steps of the foglet programming model.

First of all, a foglet must **join** a fognet. The fognet instance must be created beforehand, and be given connection information to the actual network endpoint. Then the foglet initializes its data structures, giving them a reference to the fognet. Upon joining the fognet, the foglet will start listening to **events** produced by this fognet. Each datastructure can therefore receive update messages carried by these events. This event-driven mechanism allows to keep each data structure up to date with its replicates in the fognet. Shared data structure may indeed **broadcast** their state or the performed atomic update operations using the fognet. The content of the update message is specific to each datastructure. Eventually, a foglet instance may decide to **leave** the fognet, thus disconnecting from the actual network and stopping communications with other peers.

Optionnaly, a foglet may also use unicast or multicast communication primitives in order to communicate with a subset of peers. A fognet may also provide information about connected peers when requested, or push this information when a member joins or leaves the network.

Based on this protocol, a programming interface for fognets has been defined. In addition to the above mentionned methods, the Javascript Fognet API also specifies the events that a Fognet implementation should fire. These events can be caught and processed by the foglet and its data structures, thus supporting the event-based interaction model presented above.

3) Opportunistic Fognet implementation: In the targeted infrastructure-less context, implementing a fognet would require a peer discovery mechanism which does not rely on brokering or signaling by a predefined peer. To address this issue, we have developed a fognet implementation tailored for disruption-prone mobile browser networks, that builds upon the OppNetting architecture presented in subsection III-A.

DoDWAN provides peer discovery through its beaconing mechanism, and also ensures that a message that has already been delivered to a node will not be delivered again, even if it is received from multiple sources over time, which results in data structure update messages being only processed once. Our opportunistic fognet Javascript module (shortened as OppFognet) runs a WebSocket client that connects to the NAPI WS endpoint on *Ligo*, with which it exchanges NAPI protocol messages. Thanks to the Bluetooth tunnel, the NAPI WS server acts as the local fognet endpoint. Since the same endpoint of an underlying network can be used by several fognets, the unique identifier of an OppFognet is used as a DoDWAN topic to differentiate traffic inside a DoDWAN OppNet.

Figure 2 depicts the communication channels and protocols that supports our opportunistic foglet network implementation,

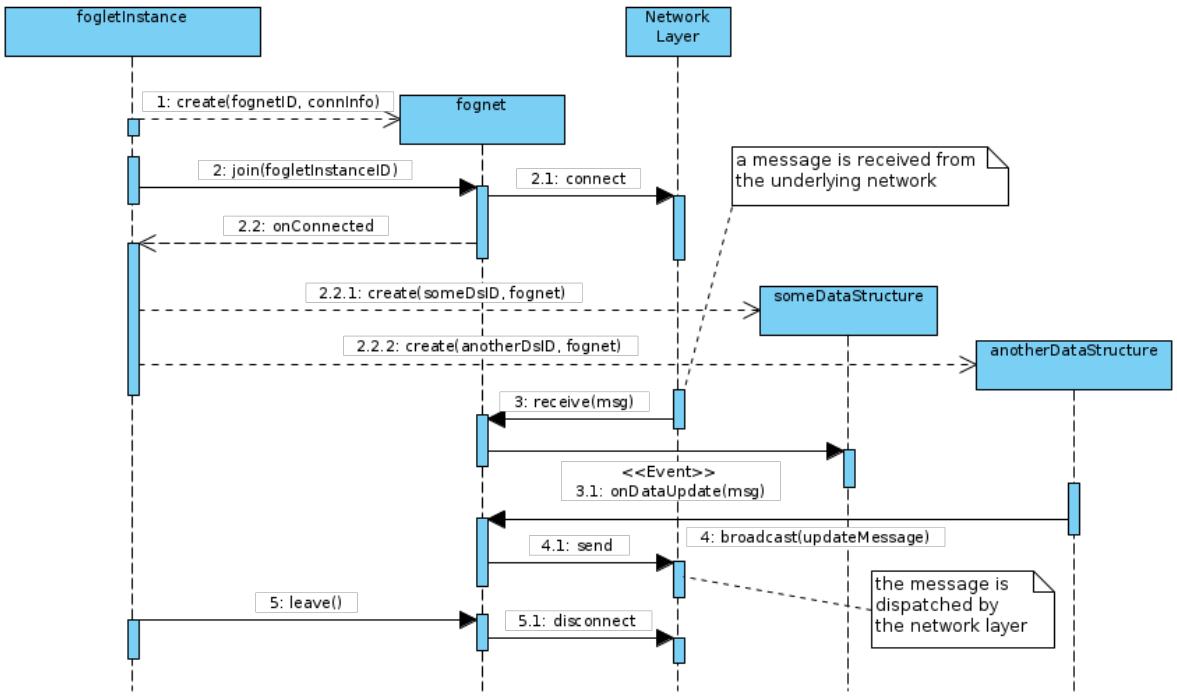

Figure 1. Foglet programming model sequence diagram

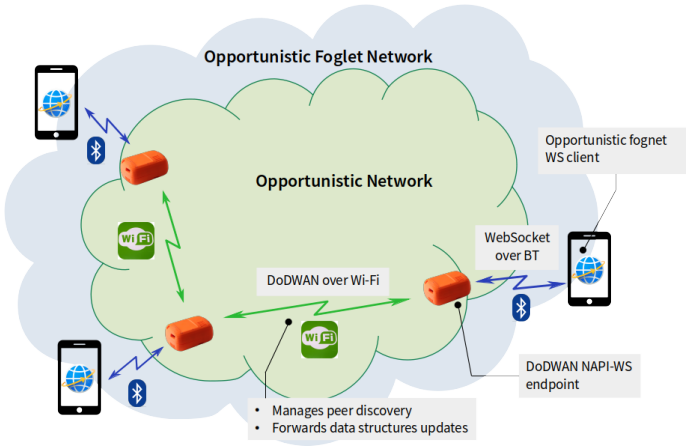

Figure 2. Communication channels and protocols in an opportunistic fognet

respectively NAPI over WebSocket (whose TCP traffic is tunneled in a RFCOMM channel), and the DoDWAN protocol over Wi-Fi in ad hoc mode. This architecture enables the following chain. The update messages related to shared data structures, encapsulated in DoDWAN messages, are received by the DoDWAN middleware running on the *Ligo* device from opportunistically encountered peers. They are then delivered by the NAPI-WS endpoint through the Bluetooth tunnel to the websocket client, which is managed by the opportunistic fognet code, on the smartphone's web browser. In the end, the fognet pushes the received updates to the corresponding data structure replica which is consequently able to process the message and update itself, converging to a consistent state across foglet instances in the fognet.

The Fognet API is implemented as follows. The **join** method establishes a connection to the NAPI WebSocket endpoint. When the connection is confirmed, the fognet sends a NAPI command to DoDWAN in order to subscribe to a topic specific to this fognet. Later, when a message is published on this topic, the fognet is notified through the WebSocket channel. Conversely, the **broadcast** method encapsulates messages such as data updates in the payload of a publish command sent to the NAPI-WS endpoint. The given message is then published on the fognet-specific topic and therefore opportunistically delivered to DoDWAN subscribers, i.e., to other members of this fognet.

4) Foglet provision: Since the smartphone may not be permanently connected to the Internet, we took advantage of the *Ligo* unit to locally serve foglets. *Ligo* runs a web server which acts as a local foglet repository, alongside DoDWAN and its NAPI websocket endpoint, as shown on Figure 3. The server hosts a WebbApp manager application which provides a REST API to add, remove or update foglet web applications in the repository. Foglets can therefore be pre-downloaded to the repository in order to be used later, when offline. The web browser on the smartphone may indeed access the foglet on the local web server (i.e., http://localhost thanks to the Bluetooth tunneling to *Ligo*).

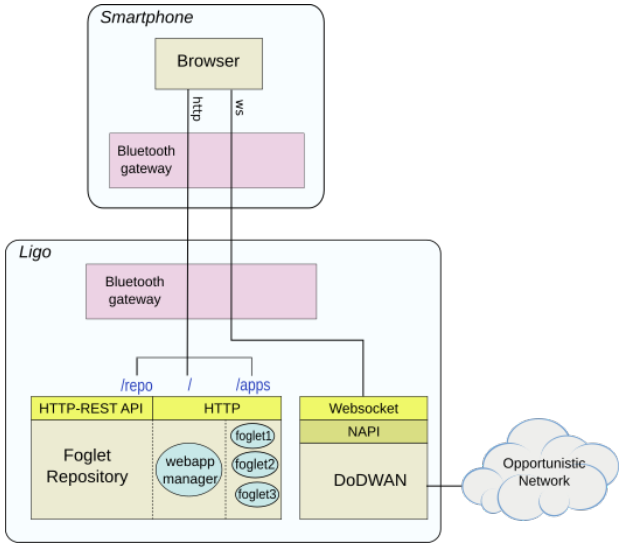

Figure 3. Foglet deployment architecture

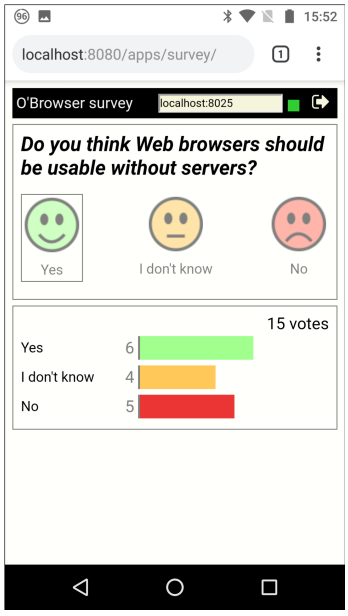

Figure 4. Survey foglet application

C. Survey foglet example

In order to experiment our solution, we have designed a simple foglet : a survey application. The purpose of this foglet is to gather answers from participants, in a decentralized way. This simple survey asks one question to which fognet members can answer by "yes", "no" or "I don't know", as illustrated by Figure 4 that shows a screenshot of the application running on a mobile web browser.

The data structures involved in this simple foglet are three instances of a *Counter* class, one for each possible answer. The developed shared *Counter* listens to events fired by the fognets, as specified in the Fognet API. As participants answer the survey and opportunistically encounter other foglet participants, they see the number of each answer growing (depicted as colored bars in Figure 4). This foglet uses our opportunistic fognet implementation which connects to the local WS endpoint running on Ligo. The consistency model implemented by this opportunistic foglet is the eventual consistency model. Users should therefore observe that the total number of answers converges, as counter update messages are broadcasted through the network. This foglet was the application used in the experiment described hereafter.

IV. Experimental results

A. Field experiment

A field experiment has been conducted to evaluate our solution. The experiment involved ten volunteers carrying their own smartphone and a *Ligo* device. The participants scattered around a small university campus, and once in position, powered on *Ligo* and opened the Bluetooth gateway app on their smartphone. Then, they had to launch a web browser and open the `http://localhost:8080/apps/survey` URL pointing to the survey foglet which was pre-deployed on the *Ligo* unit. They proceeded to answer the survey, thus incrementing one of the foglet's counter. Finally, they could put their phone aside and walk around the campus area for ten more minutes. As participants came in radio range with each other, they could observe the number of answers growing on the foglet's UI.

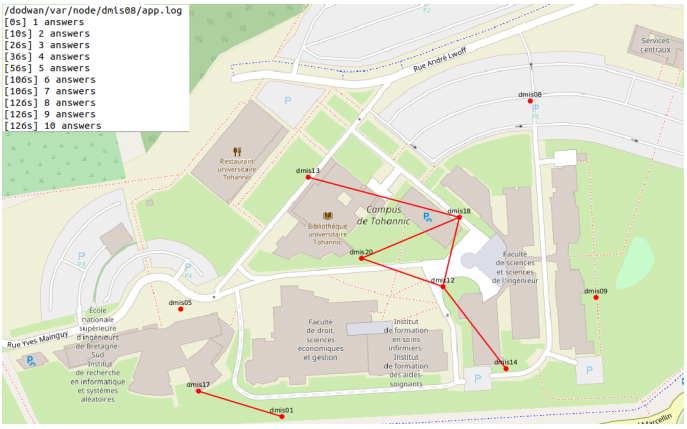

Figure 5. Snapshot of the evolution of contacts between participants of the field experiment

During the experiment, traces of locations, radio contacts and message exchanges have been logged. These traces have been used to replay the scenario afterwards, and display the temporal evolution of nodes and links as shown on Figure 5. Table I summarizes the experiment parameters like the size of the area or the duration of the experiment, and compiles some statistical results such as the number of contacts that occurred, or the observed transmission ranges.

The goal of this experiment was to observe both the proper operation of our architecture in actual conditions, and the convergence of the total number of answers on each foglet instance. Since each node had to broadcast only one answer,

Metrics	Values$^{(*=min/max/avg/stdev\ values)}$
Number of participants	10
Experiment duration	3' (on a total of 10' 15")
Size of campus area	380 m x 200 m
Observed transmission range	0 / 170 / 55 / 27 m $^{(*)}$
Nb of messages sent / received	10 / 90
Contacts between pairs of nodes	61 contacts / 34 pairs
Contact durations	1.8" / 110" / 32" / 24" $^{(*)}$
Time to converge per node	17" / 126" / 73" / 40" $^{(*)}$
Nb of contacts before converging	3 / 10 / 4.9 / 2.1 $^{(*)}$
Nb of distinct encounters to converge	2 / 7 / 3.9 / 1.5 $^{(*)}$

Table I
CAMPUS FIELD EXPERIMENT PARAMETERS AND RESULTS

only a few contacts (five in average) were necessary so that each node receive all ten answers. In this experiment, participants did not start their node at the same time. The startup phase spanned over 100 seconds, and it took overall a little less than three minutes for all instances to converge. We consequently chose to focus on this short period of time, the first three minutes of the experiment, to produce the statistical figures presented in Table I. Among other things, the experiment shows that some nodes only had to establish contact with two distinct neighbors (one-hop contacts) to receive the messages sent by the other seven, which validates the correct operation of the forwarding strategy in our opportunistic system.

Figure 6 shows for each node, from the start of the experiment, the time of answer receptions until all ten answers are received. The convergence delays relative to the startup time of each node are however expressed in Table I. Nodes like dmis17 and dmis20, that were started late, received all survey answers with only a few contacts, since the messages were already disseminated before they joined the OppNet. Conversely dmis08 was the first node in the OppNet, but took up to 126 s to converge since it moved away from others, and had to re-establish contact with nodes carrying the messages from the nodes that joined last.

The convergence observed in this field experiment is highly dependent on the mobility pattern and the radio range. Although the participants tried to spread, the area was modest in size and the range of *Ligo* devices was efficient in the open. These factors explain why foglet instances quickly reached their final state (i.e., the shared counters converged).

B. Discussion and emulation experiment

In order to test our solution at a larger scale in a less controlled environment, we used the LEPTON emulation platform [18]. LEPTON allows opportunistic software developers to run their actual OppNet system with simulated mobility and transmissions. Each LEPTON node runs an instance of the system, with its own applicative behavior. Opportunistic transmissions are intercepted by LEPTON, which transfers messages when nodes are close enough.

For our evaluation, LEPTON nodes had to run a DoDWAN instance and a foglet using our opportunistic foglet network. For this purpose the survey foglet was ported as a NodeJS application, taking advantage of NodeJS being able to run ECMAScript 6 (ES6) compliant code. As a result, our NodeJS-based foglet was able to use the opportunistic foglet implementation and the shared counter developed for the browser version, as is. In our first experiment, each node only produced one update message, so convergence was quickly reached.

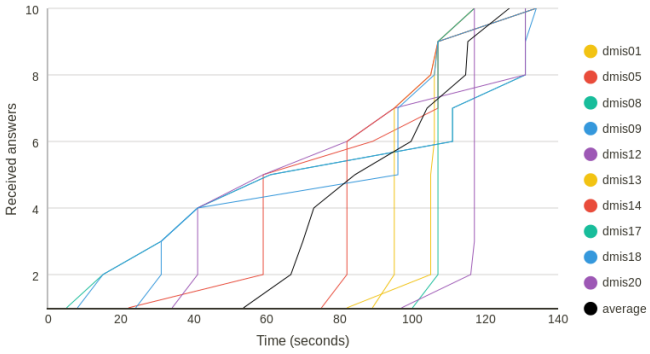

Figure 6. Reception of survey answers along time

In order to simulate a more complex foglet, the developed application was modified to produce more data structure updates in a non-interactive way. Each NodeJS foglet instance manages a single counter which is periodically incremented every x minutes, with x randomly ranging from 3 to 5 minutes. During this 4 hour long experiment, nodes started incrementing the shared counter after 5 minutes, and after 3 hours and a half of activity they stopped incrementing it. Yet, they continued exchanging and applying updates during the last 30 minutes. The number of messages produced per node is given in Table II which summarizes the experiment parameters and results.

Contact data was taken from the Haggle dataset available on CRAWDAD [19]. Cambridge University collected contact and mobility traces of 41 INFOCOM 2005 attendees in Grand Hyatt Miami during almost 3 days to produce this dataset. In the present simulation we only considered a 4 hour long subset involving 34 nodes**.**

In this second experiment, we did not observe a complete convergence in the end since, at best, 28 out of 34 nodes received 99.3% of total updates (i.e., 1796 out of 1808 update messages). Indeed, complete convergence was impossible with this subset of the Haggle dataset as two nodes become isolated towards the end of the scenario. In the simulation, they had time to receive only around 90% of updates. This phenomenon is not uncommon in OppNets where nodes might leave and never come back, or where a subset of nodes may end up isolated from the rest of the fognet. Yet, convergence remains a relevant indicator showing the effectiveness of our OppNet system.

V. CONCLUSION

This paper has presented a solution for decentralized web applications that can exploit opportunistic networks. This solution includes a practical environment that compensates the inadequation of smartphones to opportunistic communication

Metrics	Values$^{(*=min/max/avg/stdev\ values)}$
Number of nodes	34
Experiment duration	3h 59' 58"
Messages sent/received overall	1808 / 58805
Nb of messages sent per node	51 / 55 / 53 / 1.01$^{(*)}$
Nb of messages received per node	1530 / 1744 / 1730 / 44.7$^{(*)}$
Final counter value	1584 / 1796 / 1783 / 44.7$^{(*)}$
Ratio received/total updates	87.6% / 99.3% / 98.6% / 2.5$^{(*)}$

Table II
PARAMETERS OF A SIMULATED EXPERIMENT ON A SUBSET OF HAGGLE WIRELESS CONTACT DATA AND RESULTS

by externalizing this function to an handheld device called *Ligo*, based on a Raspberry Pi Zero. Additional software allows the smartphone's browser to interact with an opportunistic middleware embedded in *Ligo*. The second part of the presented solution consists in the implementation of a programming model, based on *foglets*, that allows the manipulation, in the browser, of data structures whose consistency is maintained across the different nodes of the network. The use of the foglet programming model has been illustrated through a simple survey web application. Experiments, both on the field and in an emulated setting, have shown the feasibility of our approach which could benefit more complex collaborative web applications that need to run without network infrastructure.

In the future, if mobile OS were to facilitate the use of the ad hoc mode on wireless interfaces the *Ligo* architecture could be dispensable, provided that an opportunistic networking middleware is deployed on the smartphone. But even then, *Ligo* saves the phone battery since it is dedicated to opportunistic transmissions, so it is still a good option. All the same, we plan to enhance the performance of the communication between the smartphone and *Ligo*, that suffers at present from a poor exploitation of Bluetooth on the Raspberry Pi Zero.

For the moment, the implementation of the foglet data structures is also not optimal as the operation-based dissemination of updates remains costly. A lead of research would be to adapt recent advances on CRDTs' synchronization algorithms to opportunistic networks.

FUNDING

This work was supported by the French ANR (Agence Nationale de la Recherche) under grant number ANR-16-CE25-0005-02.

REFERENCES

[1] F. Álvarez, L. Almon, P. Lieser, T. Meuser, Y. Dylla, B. Richerzhagen, M. Hollick, and R. Steinmetz, "Conducting a Large-Scale Field Test of a Smartphone-Based Communication Network for Emergency Response," in *Proceedings of the 13th Workshop on Challenged Networks (CHANTS'18)*. New Delhi, India: ACM, 2018, pp. 3–10.

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

[3] "Beartooth. AI Powered Ad Hoc Networks | Advanced off-grid communication," https://beartooth.com, accessed: 2021-05-01.

[4] A. Dusian, R. Ramanathan, W. Ramanathan, C. Servaes, and A. S. Sethi, "VINE: Zero-Control-Packet Routing for Ultra-Low-Capacity Mobile Ad Hoc Networks," in *IEEE Military Communications Conference (MILCOM 2019)*, Norfolk VA, USA, 2019, pp. 521–526.

[5] "goTenna. Extend the edge of connectivity." https://gotenna.com, accessed: 2021-05-01.

[6] A. Grall, T. Minier, and B. Nédelec, "GitHub - foglet-core v5.1.2: Easy use of WebRTC Networks with embedded network management and simple communication primitives," https://github.com/ran3d/foglet-core, accessed: 2021-05-01.

[7] F. Guidec, "DoDWAN: Document Dissemination in Wireless Ad hoc Networks," https://casa-irisa.univ-ubs.fr/dodwan, accessed: 2021-05-01.

[8] F. Guidec, P. Launay, Y. Mahéo, and L. Touseau, "Bringing Opportunistic Networking to Smartphones: a Pragmatic Approach," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2021, to appear.

[9] J. Haillot and F. Guidec, "A Protocol for Content-Based Communication in Disconnected Mobile Ad Hoc Networks," *Journal of Mobile Information Systems*, vol. 6, no. 2, pp. 123–154, 2010.

[10] O. Helgason, S. T. Kouyoumdjieva, L. Pajević, E. A. Yavuz, and G. Karlsson, "A Middleware for Opportunistic Content Distribution," *Computer Networks*, vol. 107-2, pp. 178–193, Oct. 2016.

[11] M. Karimzadeh, Z. Zhao, F. Gerber, and T. Braun, "Mobile Users Location Prediction with Complex Behavior Understanding," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, Chicago, USA, 2018, pp. 323–326.

[12] M. Kleppmann and A. R. Beresford, "A Conflict-Free Replicated JSON Datatype," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2733–2746, 2017.

[13] M. Kleppmann, A. Wiggins, P. van Hardenberg, and M. McGranaghan, "Local-First Software: You Own Your Data, in Spite of the Cloud," in *ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2019)*. Athens, Greece: ACM, 2019, pp. 154–178.

[14] V. F. S. Mota, F. D. Cunha, D. F. Macedo, J. M. S. Nogueira, and A. A. F. Loureiro, "Protocols, Mobility Models and Tools in Opportunistic Networks: A Survey," *Computer Communications*, vol. 48, pp. 5–19, July 2014.

[15] B. Nédelec, P. Molli, and A. Mostefaoui, "CRATE: Writing Stories Together with our Browsers," in *25th World Wide Web Conference*, ACM, Ed., Montréal, Canada, Apr. 2016.

[16] P. Nicolaescu, K. Jahns, M. Derntl, and R. Klamma, "Near Real-Time Peer-to-Peer Shared Editing on Extensible Data Types," in *19th International Conference on Supporting Group Work (GROUP'16)*. Sanibel Island FL, USA: ACM, 2016, pp. 39–49.

[17] J. Radianti, J. Dugdale, J. J. Gonzalez, and O.-C. Granmo, "Smartphone sensing platform for emergency management," in *11th International Conference on Information Systems for Crisis Response and Management (ISCRAM2014)*, University Park, Pennsylvania, USA, May 2014.

[18] A. Sánchez-Carmona, F. Guidec, P. Launay, Y. Mahéo, and S. Robles, "Filling in the missing link between simulation and application in opportunistic networking," *Journal of Systems and Software*, vol. 142, pp. 57–72, Aug. 2018.

[19] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD dataset cambridge/haggle (v. 2006-09-15)," *CRAWDAD Wireless Network Data Archive*, September 2006, downloaded from https://crawdad.org/cambridge/haggle/20060915.

[20] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "A Comprehensive Study of Convergent and Commutative Replicated Data Types," INRIA, Tech. Rep. 7506, Jan. 2011.

[21] The OpenJS Foundation, "Electron | Build cross-platform desktop apps with JavaScript, HTML, and CSS," https://www.electronjs.org/, accessed: 2021-05-01.

[22] S. Trifunovic, M. Kurant, K. A. Hummel, and F. Legendre, "WLAN-Opp: Ad-hoc-less opportunistic networking on smartphones," *Ad Hoc Networks*, vol. 25, Part B, pp. 346–358, Feb. 2015, special isssue on New Research Challenges in Mobile, Opportunistic and Delay-Tolerant Networks.

[23] P. van Hardenberg and M. Kleppmann, "PushPin: towards production-quality peer-to-peer collaboration," in *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data (PaPoC '20)*. Heraklion, Greece: ACM, 2020, pp. 1–10.