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*—Making web applications run in a decentralized though collaborative way, 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. This paper presents an environment that facilitates the development of web applications deployed in opportunistic networks built out of smartphones. In order to overcome the technology hindrance regarding smartphones' ad hoc communication capabilities, our contribution externalizes opportunistic networking to a dedicated device. On top of this support for smartphones, we propose an implementation of a programming model called foglet that allows web browsers to share consistent data structures. This implementation helps achieving decentralized applications distributed across mobile web browsers without relying on any network infrastructure.**

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

I. Introduction

Individual users have gotten used to software and data being synchronized and available anywhere, on multiple devices, thanks to the popularity of web standards and technologies.

Yet, this trend is being more and more challenged as end users are increasingly sensitive to privacy issues. Recent practices consequently tend to step away from centralized remote software hosted by third-party authorities, and favor decentralized approaches such as local-first software in order to regain ownership and control on the code and their data over private corporations or governments.

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. 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 wireless network infrastructures like Wi-Fi access points or broadband cellular networks. Yet, mobile use of these infrastructures may result in a varying network availability which is not usually very well handled by web applications.

The local-first initiative [6] led by Ink & Switch research lab, which emphasizes decentralization and local execution of software as guiding principles, does point out the challenge of peer-to-peer (P2P) communication for decentralized web applications. 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. None of these solutions however qualifies natively for operating collaboratively off-grid. While WebRTC depends on a reliable Internet access since it relies on signaling servers and centralized protocols (ICE, STUN, and TURN) for peer discovery and connection establishment, the WebSocket protocol provides an efficient communication channel between browser-based peers but does not manage peer discovery.

Another challenge lies at application data layer. Any decentralized system that involves collaboration between peers must provide some support for data distribution. Research activities in the field of distributed computing, led by the work on Conflict-free Replicated Data Types (CRDTs) [9], have focused on the consistency of distributed data structures. Some of these works have led to web-based implementations of CRDTs [5] which 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 [8] is a collaborative text editor whose network layer is based on WebRTC and SPRAY random peer sampling protocol. CRATE can be seen as a preliminary work that led to the concept of *foglet* that is used in this paper. The work on CRATE however requires a reliable network infrastructure and is consequently not usable as is.

In order to enable decentralized web applications without relying on network infrastructures, our proposition stems from research on Opportunistic Networks. The Opportunistic Networking (OppNetting) research area [7] addresses fragmented mobile ad hoc networks where connectivity is prone to disruptions in scenarios such as disaster relief, wild life sensing or military communications. 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 oppor-

tunities arise. It thus allows messages to be spread throughout the whole network.

Yet, 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. Wi-Fi and Bluetooth interfaces cannot be used to their fullest due to technical limitations on common smartphone operating systems.

A few attempts on OppNets of smartphones have been made, either by *rooting* the operating system [4], using Bluetooth, Wi-Fi Direct [1] or turning some smartphones into access points [10]. 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.

A few projects considered the externalization of ad hoc communication like goTenna or bearTooth which propose to extend the smartphone's range with a long-range transmission device connected via Bluetooth. Even though goTenna offers multi-hop mesh networking features, 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.

A similar approach has been used in [2], except that the device called *Ligo* provides the smartphone with opportunistic communication capabilities. *Ligo* and the smartphone are linked by their Bluetooth interfaces, while *Ligo*'s Wi-Fi interface is dedicated to opportunistic networking. The *Ligo* unit runs an instance of the DoDWAN OppNetting middleware. DoDWAN is a content-driven dissemination middleware [3] that implements a form of controlled epidemics. It stores messages in a local cache and forward them to other DoDWAN nodes. A protocol called DoDWAN-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 work presented in this paper relies on *Ligo*'s architecture and on the DoDWAN NAPI-WS plugin which is a WebSocket implementation of NAPI.

We propose a programming model and a support for web applications distributed over infrastructure-less networks, implemented on top of this opportunistic communication support for smartphones. This solution called *opportunistic fognet*, that has been validated in a field experiment, will be presented in the next section.

II. 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 builds upon the work on *Ligo* that enables opportunistic communications between mobile web browsers, bridging the gap between smartphones and opportunistic networks. Since *Ligo* addresses one of the challenges presented in the introduction, this section addresses the decentralization issue pertaining to web applications. It presents a network-level Javascript module enabling decentralized web applications over opportunistic networks, based on the *foglet* programming model.

A. Foglets

The *foglet* programming model 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, as presented in the previous section. Our contribution builds upon this *foglet* approach.

1) Foglet-related 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, ...). Foglets are meant to enable applications distributed across browsers to operate offline, and should therefore not depend on third-parties. This point stands even more if we want to achieve an infrastructure-less web of browsers. The distributed aspect of foglets relies on the following 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*. Since 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 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 are responsible for ensuring concistency across those instances. The data structures presented in this paper implement 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 steps in respect with their 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. Upon joining the fognet, the foglet will start listening to **events** produced by this fognet. Each data structure can therefore receive update messages carried by these events, in order to keep each data structure up to date with its replicates in the fognet. Shared data structures may indeed **broadcast** their state or the performed atomic update operations using the fognet. Eventually, a foglet instance may decide to **leave** the fognet, thus disconnecting from the actual network and stopping communications with other 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.

B. 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 *Ligo* OppNetting architecture.

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, 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.

Figure 1 depicts the architecture as well as the communication channels and protocols that supports our opportunistic foglet network implementation, 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. Conversely, data structures broadcast their updates using the fognet and the same communication channels : DoDWAN publish commands are sent to the NAPI-WS endpoint on Ligo through the WS channel.

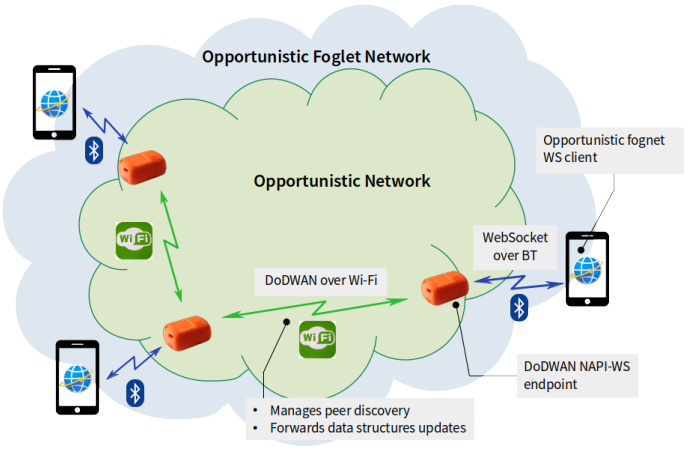

Figure 1. Communication channels and protocols in an opportunistic fognet

III. Experimental results

A. 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. The foglet asks one question to which fognet members can answer by "yes", "no" or "I don't know". The data structures involved in this simple foglet are three instances of a *Counter* class, one for each possible answer. The shared counters listen to events fired by the fognet, as specified in the Fognet API. As participants answer the survey and opportunistically encounter other foglet peers, they see the number of each answer growing. Users should 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.

B. 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.

During the experiment, traces of locations, radio contacts and message exchanges have been logged. They have been used to display the temporal evolution of nodes and links as shown on Figure 2. 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 or exchanged messages.

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, only a few contacts (five in average) were necessary so that each node receive all ten answers. 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

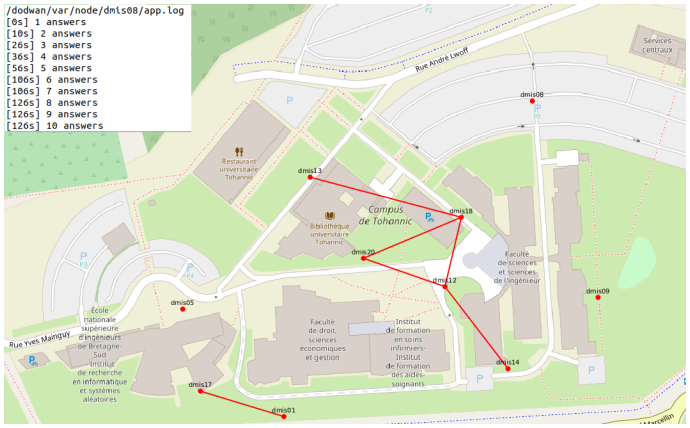

Figure 2. Snapshot of the evolution of contacts between participants of the field experiment

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

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 3 shows for each node, from the start of the experiment, the time of answer receptions until all ten answers are received. 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).

IV. Conclusion

This paper has presented a solution for decentralized mobile web applications that can exploit opportunistic networks. The use of the foglet programming model that enables rapid prototyping of easy-to-write web applications, has been illustrated

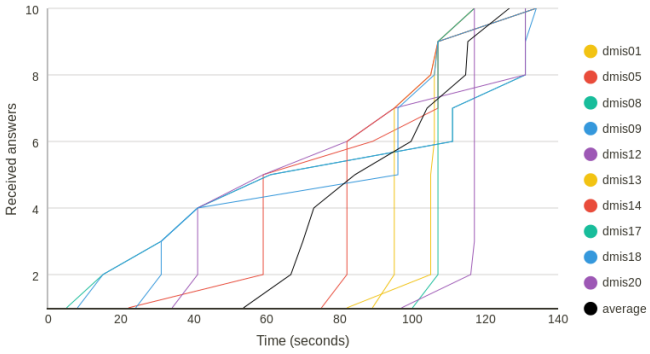

Figure 3. Reception of survey answers along time

through a simple survey web application. Experiments conducted on the field have shown the feasibility of our approach which could benefit more complex ephemeral collaborative web applications that need to run without network infrastructure, such as census applications for scientific wildlife campaigns, or in disaster relief scenarios.

Funding

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

References

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

[2] 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)*. Virtual event: IEEE, Jul. 2021, pp. 574–579.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.