Towards Cloudless Co-located Social Media on Android

Adrian Holzer
EPFL
Lausanne, Switzerland
adrian.holzer@epfl.ch

Gabriel De Tiberge
EPFL
Lausanne, Switzerland
gabriel.detiberge@epfl.ch

Denis Gillet
EPFL
Lausanne, Switzerland
denis.gillet@epfl.ch

ABSTRACT
Mobile devices have become ubiquitous. However, they often do not equate with access to communication services. Indeed, many of the 60 million people forcibly displaced worldwide who have access to mobile devices, lack communication services due to lack of infrastructure or lack of Internet credits. In this research-in-progress, we provide the basis to create cloudless co-located social media on Android. To do so, we present a novel middleware solution called PadocAndroid, which allows multihop ad hoc communication between off-the-shelf Android devices. We also present experimental results and a proof-of-concept chat application to illustrate the usage of the middleware.

Keywords
Cloudless communication, ad hoc networking, Android

1. INTRODUCTION
The number of individuals forcibly displaced worldwide is huge (59.5 million in 2014) and growing more rapidly than ever [9]. Many of them now have access to mobile devices for communication. However, in many managed camps located in rural areas [9], low levels of connectivity hinder communication. Furthermore, in camps where communication infrastructure is available there are further economic or policy issues that prevent users from accessing information [8].

Despite theoretical work on ad hoc middleware that could enable cloudless co-located social media to overcome this issue (e.g., [2, 6, 1]), there has hardly been any spillover in the real world [6]. Nevertheless, in recent years several efforts have been made to offer programming support for such service (e.g., BASA, Serval, Padoc or OpenGarden).

BASA [10] 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 support multi-hop communication.

The *Serval Mesh* [3, 4] 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. 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.

OpenGarden is a company which released the most popular ad hoc social media application to date: the Firechat chat app for iOS and Android (`https://firech.at`) Firechat is available as SDK, but only for select organizations.

Padoc [5] is a recent effort to provide a multihop middleware solution for off-the-shelf iOS devices. The fact that Padoc targets iOS devices, which are high end devices, makes it difficult to use in the refugee context.

In this paper, we address these shortcomings through the following contributions:

1. A novel middleware solution called PadocAndroid, which allows multihop ad hoc communication between off-the-shelf Android devices (Section 2).
2. Validation of the PadocAndroid middleware services using 10 standard Android devices (Section 3)
3. A novel chat app, called PadocChat, built on top of PadocAndroid as proof-of-concept (Section 4).
4. Pointers to future research avenues (Section 5).
5. Access to PadocAndroid[1] and to the chat app[2] through an open source MIT licence to encourage reuse.

2. PADOC ANDROID
Padoc Android is java library implementing peer-to-peer, multi-hop connectivity for off-the-shelf Android devices running Android KitKat (4.4) or higher. By off-the-shelf devices, we mean commercially available devices that have not been rooted or jailbroken. The middleware creates a connected multihop ad hoc network with devices in the vicinity.

[1]`https://github.com/react-epfl/padoc-android`
[2]`https://github.com/react-epfl/padoc-android-chat`

Note that no user interaction is required to confirm connections between devices. PadocAndroid offers the following methods for communication:

```
public interface PadocAndroid{
  /*broadcasts a message in the network using the defined
  protocol (FLOOD or CBS)*/
  void broadcast(String message, int PROTOCOL)
  /*sends a message over the network to the specified peer*/
  void send(String message, Address peerAddress);
  /*getAddresses returns an Array of Address objects*/
  Array getAddresses();
}
```

An Address object contains three main fields: the peerID of the destination, the peerID of the next hop, the number of hops to the destination.

2.1 Ad hoc networking on Android

The tools available in Android for direct wireless device-to-device communication are Bluetooth and WiFi-Direct.

WiFi Direct restricts devices to the role of group owner or clients. A group owner can have up to four clients. But no multihop communication is possible. It offers a send and a broadcast service to all group members.

Bluetooth allows devices to act as clients and servers simultaneously, which allows multihop communication in theory. However, our experiments showed that Bluetooth connections start to break when there are more than three devices connected. Contrary to its WiFi counterparts, Bluetooth disconnections are not automatically detected, nor repaired.

PadocAndroid uses a combination of Bluetooth and WiFi-Direct to combine the best of both worlds.

2.2 Creating the network

To discover nearby peers running the specified Padoc-enabled application, Padoc Android relies on WiFi-Direct service discovery. When a peer is discovered a decision is taken depending on the advertised information as well as the current peer state that performed the discovery. If there is more than one peer in the area but none of them is group owner, one of them will become group owner. A group owner then accepts up to four WiFi connections. If there is a group owner and it is accepting WiFi connections, then the peer will attempt to connect through WiFi. If there is a group owner but it has reached its maximum number of clients, the peer will connect to it through Bluetooth and become a WiFi group owner. Thus extending the mesh.

Figure 1 illustrates how devices can be connected with PadocAndroid. The dark nodes are WiFi Direct Group Owners and are connected together through Bluetooth. The light nodes are WiFi direct clients and are connected to Group Owners through WiFi.

2.3 Sending messages to specific peers

In order to send messages to specific peers, a gradient-routing (GR) algorithm is used. In this algorithm each peer maintains a routing table, which contains addresses of all peers in the network. An address indicates which of the neighboring nodes is closest to the destination in terms of hops and how many hops it takes to reach the destination.

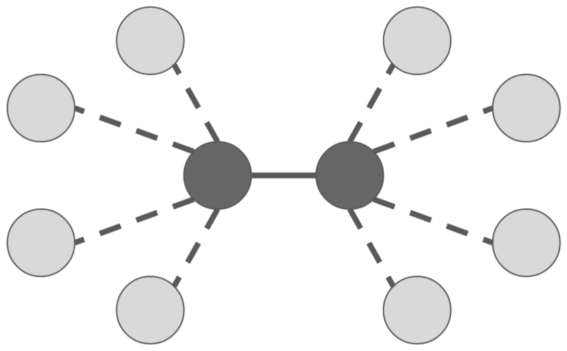


Figure 1: PadocAndroid network connections. Dark nodes are WiFi-Direct group owners connected to light gray clients through WiFi and to other group owners through Bluetooth.

When a new connection is created between two peers, the peers exchange their routing tables and update their entries if there is a new destination or if there is a shorter route to an existing destination. The routing table is also updated in case of topology changes in the network.

When a message is to be sent to a specific destination, GR looks up the address of the destination which indicates to which neighbor the message needs to be sent in order to eventually reach the destination.

2.4 Broadcasting messages to all peers

In order to broadcast messages to all peers in the network, PadocAndroid provides two message diffusion algorithms, namely Flooding and CBS.

2.4.1 Flooding

The flooding algorithm works as follows. When a peer receives a message it forwards it to all peers to which it is connected. If the peer receives copies of the messages again later, they will not be forwarded. This algorithm implies that all nodes in the network will forward the message once, which is not efficient in terms of message load and can lead to message collisions in a so-called broadcast storm [7].

2.4.2 CBS

The CBS (Counter-Based Scheme) algorithm has been proposed to mitigate the broadcast storm issue [7]. Upon receiving a message the peer sets a random timer. When the timer expires, the message is forwarded only if no other copy of the message was received during the waiting time.

2.5 Limitations

The current design limits the size of the network to around 15-20 nodes, due to parallel Bluetooth connections which become quickly unreliable, and due to the fact that the network is fully connected with maintained routing tables. The routing table maintenance cost grows exponentially as the number of peers grow and is not sustainable for larger ad hoc networks. Also, note that the use of heterogeneous technology can result in suboptimal connections where a device can be within WiFi range (used for discovery), but out of Bluetooth range to establish a connection. Nevertheless, this design allows to send messages rapidly through small networks and provides a first step in multihop messaging on off-the-shelf Android devices.

3. EVALUATION

We evaluated PadocAndroid using ten non-rooted, off-the-shelf, Samsung Galaxy J1 devices (depicted in Figure 2), as these represent the low-end devices available in the market. Eight of them running Android 5.1.1 and two Android 4.4.4. These devices are WiFi-Direct enabled and use Bluetooth 4.0. The maximum observed transmission range between two devices was about 60 meters.

Figure 2: Devices used (Samsung Galaxy J1).

For the experiment, the devices were located in the same room and upon initialization they built connections that ended up creating an ad hoc connection along the topology shown on Figure 1.

We compared the performance of the Flooding and CBS in a *broadcast* scenario, where one peer sent messages to all other peers in the network, as well as in a scenario where all peers sent messages to all others. We compared the performance of the GR and CBS in a *unicast* scenario, where all devices sent messages to one specific peer in the network. Note that the CBS algorithm still sent messages to all peers, but the messages would only be relevant to the destination peer.

All evaluations were run for 10 minutes in three throughput conditions which varied the number of messages sent by senders. Low throughput: 300 messages sent (0.5 per second). Medium throughput: 600 messages sent (1 per second). High throughput: 1200 messages sent (2 per second).

The performance metrics measured were the *delivery rate* and their *message load* in different settings. The delivery rate is expressed in a percentage. It indicates the ratio of messages that were successfully delivered. It ranges from 0% to 100%, where the latter indicates that all messages have been delivered. The message load is the total number of message transmissions that occurred in the network during the experiment. A lower message load indicates less network congestion and less resource consumption.

3.1 Broadcast results

The results of the one-to-all broadcast scenario are shown in Figure 3. Note that the message load results are shown as percentage of the Flood message load. The results show that the delivery ratio of both Flooding and CBS are similar and amount to about 90% in all three throughput conditions.

In terms of message load, CBS generates only 40% of the messages generated by Flooding, which is a significant improvement.

In the all-to-all broadcast scenario, the experiments using Flooding showed signs of broadcast storms with high levels of disconnections even in the low throughput condition. CBS on the other hand was able to provides comparable results in terms of delivery and message load. These results convey the fact that CBS is preferable alternative to Flooding even in relatively small network topologies, where broadcast storms have less chance to occur.

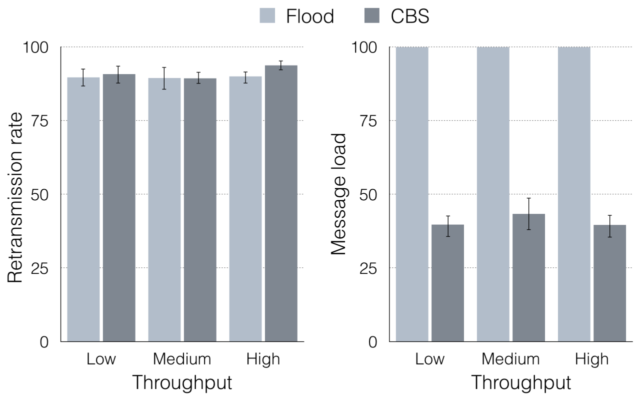


Figure 3: One to all, broadcast results in low, medium, and high throughput conditions. The message load results are shown as percentage of the Flood message load.

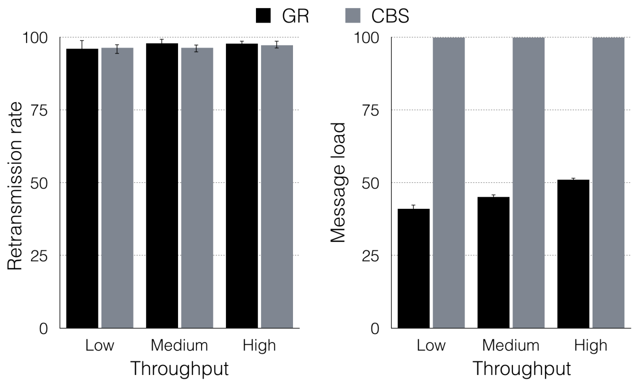


Figure 4: All to one, unicast results in low, medium, and high throughput conditions. The message load results are shown as percentage of the CBS message load.

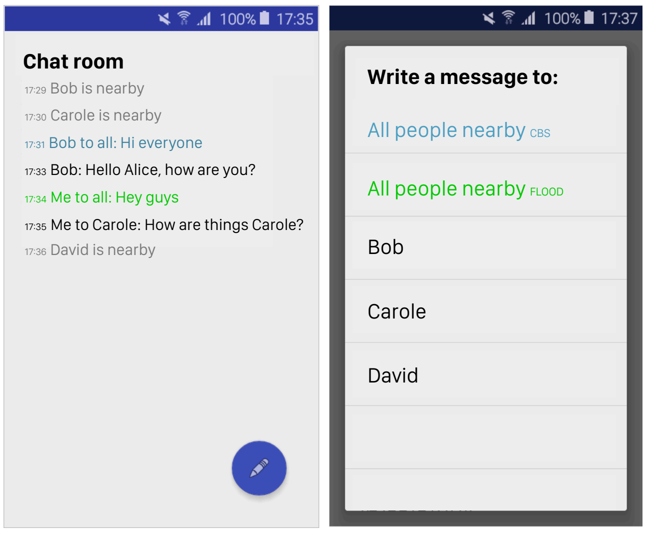
3.2 Unicast results

The results of the all-to-one unicast scenario are shown in Figure 4. Note that the message load results are shown as percentage of the CBS message load. The results show that the delivery ratio of both Flood and CBS are similar and amount to about 97% in all three throughput conditions.

In terms of message load, GR generates only between 40%-50% of the messages generated by CBS, which is a significant improvement. In the current implementation, which targets relatively small network topologies with relatively stable connections due to low mobility (walking speed), the overhead imposed by GR in terms of routing table maintenance is acceptable. However, for larger scales, this overhead might become unsustainable.

4. PADOC CHAT PROOF-OF-CONCEPT

PadocChat is a simple Android cloudless co-located social media chat application, which uses Padoc Android as a proof-of-concept. It demonstrates one possible use of the library in a chat application using a simple layout as shown in Figure 5. This application can typically be used as starting point for developers and researchers.


100% 17:35

Chat room

17:29 Bob is nearby
17:30 Carole is nearby
17:31 Bob to all: Hi everyone
17:33 Bob: Hello Alice, how are you?
17:34 Me to all: Hey guys
17:35 Me to Carole: How are things Carole?
17:36 David is nearby

100% 17:37

Write a message to:

All people nearby CBS
All people nearby FLOOD
Bob
Carole
David

Figure 5: PadocChat screenshots

When the application starts it automatically looks for and connects to other peers who are also running the PadocChat. The application has one single screen, composed of a TextView that shows a list of messages in the chat room and a pencil button (image on the left).

To write a message, a user presses the pencil button and is presented with a list of users in the network (image on the right). The user also has the option to broadcast a message to all people nearby. When a selection is made, a modal message composition input box is shown (not depicted here) that allows users to compose and send the message.

Messages sent to specific users use the GR routing protocol, whereas messages broadcast to all can use either CBS or Flooding, depending on the user selection. Note that such low level protocol selection is not desirable in an end user application, but allows to showcase the different networking protocols available in PadocAndroid.

When a message is received it is added to the user's chat room, which contains both individual and group messages. Furthermore, the chatroom also contains information about newly arrived people in the network.

5. CONCLUSION AND FUTURE WORK

In this research in progress, we presented PadocAndroid, a middleware to support the development of cloudless co-located social media applications. The middleware automatically connects nearby devices in an ad hoc topology and offers unicast and broadcast services. We have validated the effectiveness of these services through our evaluations with 10 real devices in different settings. We have also validated the usage of the middleware, by building PadocChat, a cloudless co-located social media chat applications as proof of concept. Such applications are currently lacking and can prove particularly useful in the humanitarian and development context where either network connection is not available, or where access to the network is too costly or prohibited. We believe that providing developers with adequate middleware can contribute to the creation of more of such application.

In future work we plan to build an opportunistic communication scheme, where peers only connect for short periods with other nodes in their transmission range using Bluetooth. In that way we could overcome the connection limitation of Bluetooth and allow for larger networks. However, this scheme will introduce delays between the time a message is sent and delivered. We believe that for several application scenarios, such delay tolerance is still acceptable. Finally, we plan to investigate hybrid connections, where data is eventually saved on the internet if available.

6. REFERENCES

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

[2] 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.

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

[4] 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 Hum. Tech. Conference (GHTC)*, pages 162–166. IEEE, 2013.

[5] A. Holzer, S. Reber, J. Quarta, J. Mazuze, and D. Gillet. Padoc: Enabling social networking in proximity. *Computer Networks*, page in press, 2016.

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

[7] 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.

[8] UNHCR. Mass communication assessment of syrian refugees in camps in the kurdistan region of iraq. *Assessment Report*, 2014.

[9] UNHCR. Unhcr global trends: Forced displacement in 2015. *Annual Report*, 2016.

[10] 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.