# Experimentation with MANETs of Smartphones

Eduardo Soares, Pedro Brandão, Rui Prior, Ana Aguiar
Instituto de Telecomunicações, Universidade do Porto
{esoares, pbrandao, rprior}@dcc.fc.up.pt, anaa@fe.up.pt

***Abstract*—Mobile AdHoc NETworks (MANETs) have been identified as a key emerging technology for scenarios in which IEEE 802.11 or cellular communications are either infeasible, inefficient, or cost-ineffective. Smartphones are the most adequate network nodes in many of these scenarios, but it is not straightforward to build a network with them. We extensively survey existing possibilities to build applications on top of ad-hoc smartphone networks for experimentation purposes, and introduce a taxonomy to classify them. We present AdHocDroid, an Android package that creates an IP-level MANET of (rooted) Android smartphones.AdHocDroid supports standard TCP/IP applications, providing real smartphone IEEE 802.11 MANET and the capability to easily change the routing protocol. We validate the MANET with off-the-shelf applications and experimental performance evaluation, including network metrics and battery discharge rate.**

***Keywords*—Ad-hoc networks, MANET, mesh networks, Android**

## I. Introduction

Although Internet connectivity is nearly ubiquitous, there are many situations in which using infrastructureless communication is better than an IEEE 802.11 hotspot or cellular communication, because the latter are either infeasible, inefficient, or cost ineffective. For example, in remote areas, e.g. forests, ocean, or in catastrophe scenarios [1], [2], there is simply no infrastructure to provide connectivity. Or in social upraise scenarios, in which the infrastructure cannot be trusted, as the use of Open Garden[1] shows. A more leisurely example is low latency gaming [3], [4] or sharing a file with acquaintances. Another application scenario could be group communication in mass events, like conferences or concerts [5], in which infrastructure may be unable to support all communication demand. For all these reasons, wireless ad-hoc networking was identified as a major emerging technology at the "Internet on the Move" workshop [6] and by Conti et al. [7].

Recently, application frameworks that leverage WiFi Direct or cooperation between IEEE 802.11 and Bluetooth (BT) appeared. But those solutions do not create an IP-level network, require overlay routing for multi-hop communication, and applications need to use a specific framework Application Programming Interfaces (APIs). IP-level multi-hop networking makes the difference where communication with other IP enabled devices like laptops is wanted. Moreover, it is completely transparent to applications, which just use the sockets API.

In this paper, we review work on MANETs of smartphones, and survey solutions that claim to provide ad-hoc connectivity for smartphones (section II). We then introduce AdHocDroid to turn smartphones into nodes of an IP-level mobile ad-hoc network. We describe how to set up an IP-level 802.11 ad-hoc network with multi-hop capability on smartphones running the Android Operating System (OS), and share lessons learned (section III).Then, we introduce a taxonomy of MANET features and use it to characterize the surveyed solutions and AdHocDroid (section IV). Finally, we experimentally validate AdHocDroid running applications on the network, and evaluating network performance (throughput and latency) and battery consumption (section V).

[1] http://opengarden.com/about

## II. Related Work

A survey of experimental work with MANETs [8] shows that real-world experiments can bring to light significantly different behaviours than MANETs simulation or even emulation. The survey describes 5 static experiments and 8 with mobility, but only 2 testbeds (APE and ORBIT), which use laptops running the Linux OS with 802.11 dongles. The survey highlights the importance of real world-experimentation and describes toolsets. Recently, Papadopoulos et al. [9] highlighted the benefits of experimentation for the deployment of ad-hoc networks, and identify reproducibility as a caveat of the methodology. In this sense, we expect to contribute to the acceleration of experimentation with ad-hoc networks of smartphones by providing a tool that simplifies setting up such a network. The next sub-sections describe protocols and frameworks that aim at providing MANETs.

### A. 802.11 Support

The latest revision of the IEEE 802.11 [10] supports two different modes that can be used for ad-hoc networking: Independent Basic Service Set (IBSS) mode and 802.11s.

*1) IBSS Mode:* In this mode, all nodes play similar roles, and any node can communicate directly with any other node within the network, defined as the set of nodes in IBSS mode sharing the same Service Set Identifier (SSID) that are within its radio range.

The IBSS mode itself, however, does not provide multi-hop capabilities. In IBSS-based ad-hoc networks, these functions must be performed by an additional protocol, like OLSR [11] usually at the network layer. Since connectivity is provided at the link layer, IP-based applications work without any modification in an IBSS-based network. Interoperation with non-Android systems works out-of-the-box for the single-hop case, and requires a routing protocol for multi-hop.

*2) 802.11s:* More recently, mesh networking support has been introduced in 802.11 through the 802.11s amendment, now incorporated in the standard [10]. 802.11s defines the Mesh Basic Service Set (MBSS) that provides a wireless Distribution System (DS) based on meshing at the link layer. 802.11s defines a standard path metric and path selection protocol, Hybrid

Wireless Mesh Protocol (HWMP), though others can be used as long as all stations in the mesh agree.

In Android, vendor-provided system images do not usually support 802.11s in the kernel. This means that the use of 802.11s, even on devices with supported chipsets, is limited to those using third party, customized Android versions, and we are not aware of any that currently supports 802.11s.

### B. WiFi Direct

WiFi Direct is a technical specification [12] of the WiFi Alliance that leverages existing standards to provide a convenient way for securely connecting devices without installed infrastructure, enhanced with features like peer and service discovery. It is based on the *infrastructure BSS* mode of 802.11. One of the devices, selected through negotiation, will become the group owner (Group Owner (GO)) and act as an Access Point (AP). This has the advantage of allowing legacy clients to connect to the GO. The GO incorporates a Dynamic Host Configuration Protocol (DHCP) server for providing IP addresses to the Client nodes.

A significant disadvantage of WiFi Direct is that if the GO leaves, the group is torn down and a new group must be established from scratch. While these limitations are irrelevant in simple situations like a printer letting computers and other devices connect, they make WiFi Direct unsuitable as a basis for multi-hop networking.

### C. Open Garden

Open Garden is a software for Internet connection sharing on mobile devices using a mesh of BT or WiFi Direct links. It also allows communication between devices across multiple hops as long as the application uses OpenGarden's proprietary forwarding software. From the scarce documentation and our own tests, we concluded that no IP-level connectivity that might be used by other applications is provided.

Open Garden works by creating a Virtual Private Network (VPN) to a BT paired device also running the application. The other device terminates the VPN tunnel and either forwards the request to another node or redirects the message to the local application that registered for it (most commonly FireChat)[2].

When a device has Internet connection it can forward the requests received. This, again, after the Open Garden software interprets and re-routes the data packets.

### D. Serval Project

The Serval project [13] provides a free and open-source software to allow mobile phones to communicate in the absence of phone towers and other infrastructure, targeting disaster situations and remote communities. The Serval Mesh applicationprovides voice calls, text messaging and file sharing directly over IEEE 802.11 links between mobile devices. It can be used for peer-to-peer communication through an 802.11 AP or in an ad-hoc multi-hop topology without infrastructure support. The MANET is implemented over 802.11 in IBSS mode, using an *in-house* ad-hoc routing protocol.

[2]We were unable to verify the level of node identification used. From our experiments, it seemed that this was carried in a proprietary message.

The current application on the Google Play Store includes the Serval Mesh that provides the above functionality including the project's routing protocol. Currently the development is being driven for mobile phones and Android is the one currently supported with applications. The specificities of the protocols outlined above make the Serval approach unusable by applications that are not aware of their API and sub-system. This provides little to no flexibility as a MANET test-bed.

## III. AdHocDroid

AdHocDroid is an Android application that makes the necessary changes in the device to effortlessly create a MANET in one step. The application sets up the IBSS network, enabling ad-hoc mode on the wireless card, offers the possibility to choose the network name, and configures the IP address, network mask and gateway for the device. All parameters have default values, e.g. the IP address is chosen according to [14]. The application also allows an easy way to import and run different routing protocols, and using tools to monitor and evaluate the state of the network.

We where able to successfully use our application enabling multi-hop connectivity on the Gigabyte Gsmart G1305 with Android 2.3 (CyanogenMod 7), and Samsung Nexus S with Android 4.3 (CyanogenMod 10.2.1) smartphones, and on the Samsung Galaxy Tab 10.1 tablet with Android 3.2. We also tested it on LG Nexus 4, LG Nexus 5 and Motorola Moto G (2013), but due to driver or chipset issues the Ad-Hoc mode did not function correctly. We have also tested the smartphone ad-hoc network with a first responder monitoring application with success [15].

### A. Architecture

The application follows a modular architecture as shown in Fig. 1. There are three clearly defined modules that map to the elements of the application, the network configuration (NetConfig), the routing protocols (Routing) and tools (Tools).

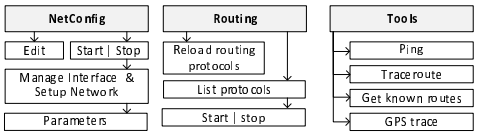


Figure 1. AdHocDroid application architecture

For the network configuration all parameters can be adjusted. The start/stop button executes the network stack setup procedure. First it turns off the network interface via the Android API, which avoids any other application making changes to the configurations of the networks. It then adds or removes the IBSS network, and finishes turning on the network interface in order to load the new network, and applies to it the network parameters (IP address, network mask, gateway).

For multi-hop connectivity, we bundle the application with an OLSR [11] routing daemon, but it is easy to import, start and stop other routing protocols. This can be done without recompiling or changing AdHocDroid, by creating a zip file with bash scripts for starting and stopping the routing protocol, and the executable binaries with the implementation of the protocol cross-compiled to the architecture of the device. Using

this routing protocol, we verified that it is possible for terminals running other OS's to join the created MANET. This was tested with a laptop running a Linux distribution (Ubuntu) with OLSR and two smartphones compatible with AdHocDroid.

AdHocDroid has additional tools that we have found useful when carrying out experiments in the field, like providing information on the routing table, and execute ping or traceroute commands. It is also possible to log the smartphone GPS co-ordinates, e.g. to map connectivity using geographic positions.

### B. Lessons Learned

During development and testing, we came across problems with the Android APIs and the diversity of devices, which we report in this section.

The Android API to configure network information, namely IP address, gateway and network mask, was deprecated in Android HoneyComb (3.0). To surpass this problem we used reflection through undocumented and internal APIs, and wrapped this in a library[3].

The Android API does not enable the creation of an ad-hoc network. Thus, the library directly edits the system configuration file, usually `wpa_supplicant.conf` in Linux distributions, located in `/data/misc/wifi/` in Android. However, we found that it has other names in some devices. For example, in Samsung Galaxy Tab 10.1 it was named `bcm_supp.conf` and was located in `/data/wifi/`.

Sometimes, after adding a new IBSS network, the phone would prefer a previously saved 802.11 network instead of connecting to the new network. To fix this, the library edits `wpa_supplicant.conf` to show only IBSS networks to the OS when we want to connect to ad-hoc networks.

Even after the network was set up, some devices (LG Nexus 5 and Motorola Moto G (2013)) were unable to connect to the MANET. We assume that the WiFi chipset drivers did not implement this mode, since the Android OS did not present any limitation and messages of issues in the driver were present in the Android WiFi state machine.

Some other devices (LG Nexus 4) are able to connect to the network in IBSS mode, but we found non-compliant behaviour. The first device to connect would act as an AP, and other devices would from then on use this AP to route traffic. If the first device (acting as AP) left or went out of range, the rest of devices in the network where unable to communicate.

In summary, IBSS mode in Android devices is problematic mainly due to drivers or chipsets not implementing the required functionality. Nevertheless, we tested and were/are able to run AdHocDroid repeatedly and consistently on Gigabyte Gsmart G1305, Samsung Nexus S and Samsung Galaxy Tab 10.1.

## IV. Is It Really a Wireless MANET?

Many applications erroneously claim to provide mobile ad-hoc networking. In our view provisioning a MANET requires answering all the following questions with yes:

- is communication possible without connectivity to the Internet? (**No Internet Needed**)
- is multi-hop communication possible? (**Multi-hop**)
- can any application take advantage of the provided connectivity through a regular socket API, thus not requiring adaptation/re-writing? (**Any App**)
- does not need additional wireless technology to provide communication, e.g.: using IEEE 802.11 needs also BT? (**No other Wireless**)
- can we use other OS's to communicate with the MANET? E.g.: Android device can we communicate with a PC running another OS? (**Other Systems**)

We analysed the technologies addressed in section II according to this definition, and summarise the results in table I.

The main problem that almost every proposal faces is the support in different systems. This in some cases may be a matter of adoption (802.11s and AdHocDroid) while in others it involves "heavier" development from the proponents themselves. Some points are critical for a MANET testbed namely supporting multi-hop and providing the regular socket interface for applications. In our view, this makes Open Garden, Serval and WiFi Direct not fit the "wireless MANET" name.

Table I. MANET NETWORK SOLUTIONS CHECK-LIST. *yes* IS DENOTED BY ✓, *no* BY ✕ AND ◆ INDICATES *partially* OR *with some adaptations*

| Proposal | No Internet Needed | Multi-hop | Any App | No other Wireless | Other Systems |
|---|---|---|---|---|---|
| 802.11s (native) | ✓ | ✓ | ✓ | ✓ | ◆ |
| Open Garden | ✓ | ◆ | ✕ | ✕ | ✕ |
| Serval | ✓ | ✓ | ✕ | ✓ | ✕ |
| WiFi Direct | ✓ | ✕ | ✕ | ✓ | ✓ |
| AdHoc-Droid | ✓ | ✓ | ✓ | ✓ | ◆ |

As is summarized in table I, only AdHocDroid and 802.11s truly provide all the features for a MANET. As we mentioned our aim is to advance the state-of-the-art in MANET experimentation by providing a framework to easily test MANET network protocols. In the future, 802.11s, when it becomes adopted in chipsets and their drivers, will provide an easy establishment of a multi-hop MANET. However, as multi-hop is provided at driver level, it can be more difficult to test routing protocols than with our solution.

## V. Experimental Results

We performed a series of experiments to test AdHocDroid, measuring throughput, delay and battery consumption and comparing them with values obtained in a standard infrastructured scenario with an AP. The experiments were performed using three Samsung Nexus S smartphones running Android 4.3 with CyanogenMod 10.2.1, in single (SH) and multihop (MH) configurations, as shown in Fig. 2.

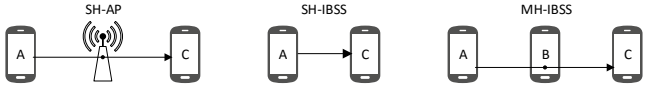


Figure 2. Test scenarios

We measured the throughput of the different nodes in the scenarios shown in Fig. 2. We generated traffic from node A to node C using iPerf (version 2.0.5), sending UDP packets at the maximum 802.11a/g PHY rate (54 Mbits/s), thus overloading the channel and collecting throughput once per second. As the

[3]https://github.com/eSoares/Android-IP-Manager

experiments were done while measuring battery discharge, they were done until one of the nodes had the battery depleted. We executed six of these batches.

The results are shown in Fig. 3. As expected, throughput is comparable in the multihop IBSS and the infrastructured cases, and larger (more than double) in the single hop IBSS case. This is due to the absence of forwarding at an intermediate node (AP or router), implying lower medium contention.

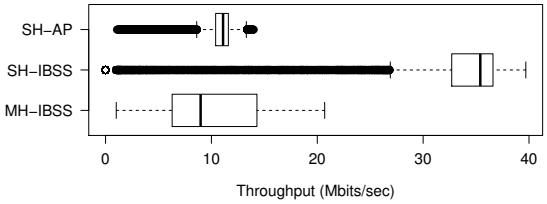


Figure 3. Throughput: AP vs. IBSS single hop vs. IBSS multi-hop

We evaluated the impact on a smartphone's battery discharge rate of being used as MANET node. We measured the battery when simply setting the devices to IBSS mode, without network traffic. Compared to an infrastructure scenario through an AP, there is a significant increase in battery consumption, the median discharge rate increases from 3.29% to 10.23% per hour, and remains at 10.25% when the routing protocol is added (note that this is a stationary scenario with only three nodes).

We also measured the discharge rate of the different nodes in the same scenarios. Because the discharge rate is a function of the amount of traffic being sent/received, overloading the channel, as iPerf does, ensures that we are observing worst-case battery discharge rates. We used the same setting as in the throughput experiments. In these, we additionally collected the time interval for each percent point drop in battery until the first node has the battery depleted. We plot the distribution of the discharge rate between consecutive points in Fig. 4.

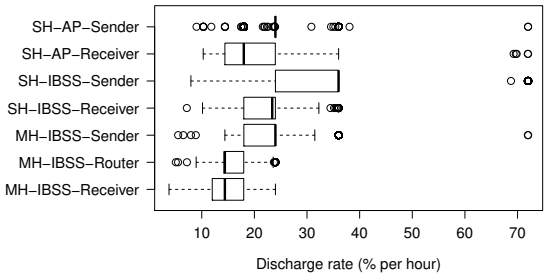


Figure 4. Discharge rate: AP vs. IBSS single hop vs. IBSS multi-hop

Battery consumption in multihop IBSS is comparable to the infrastructured case for both sender and receiver. In single hop IBSS, the consumption is higher since the throughput is also higher due to the lower medium contention. Consumption at the sender is higher than at the receiver because it is trying to send at a higher rate than is actually possible. The router in the multihop IBSS scenario discharges at a similar rate as the receiver, confirming that the discharge rate is determined by the time that the network interface card is busy. We double checked this by sending at rate lower than saturation (6 Mbps) and observing similar battery discharge rates in all nodes.

We also tried standard IP peer-to-peer applications from the Google App Store. They worked mostly as expected, apart some issues with multicast DNS used by one.

## VI. Conclusions

Motivated by a wide range of application scenarios for MANETs of smartphones proposed in the literature, we survey the existing solutions to actually enable IP-level ad-hoc networking on Android smartphones. We describe a software to enable ad-hoc networking on Android devices and shared learned lessons, so that anyone in the community can easily build a MANET test-bed with multihop capability that applications can access through the sockets API. We thus expect to contribute to moving the envisioned applications one step closer to reality.

A full version of this short paper is available in arXiv[4].

**Acknowledgements:** Research funded by Project VR2Market - Ref. CMUP-ERI/FIA/0031/2013.

## References

[1] P. Mitra and C. Poellabauer, "Emergency response in smartphone-based Mobile Ad-Hoc Networks," in *IEEE Int. Conference on Communications (ICC)*. IEEE, Jun. 2012, pp. 6091–6095.

[2] D. Srikrishna and R. Krishnamoorthy, "SocialMesh: Can networks of meshed smartphones ensure public access to twitter during an attack?" *IEEE Communications Magazine*, vol. 50, no. 6, pp. 99–105, Jun. 2012.

[3] T.-Y. Huang, C.-M. Lin, J.-R. Jiang, W. T. Ooi, M. Abdallah, and K. Boussetta, "SYMA: A Synchronous Multihop Architecture for Wireless Ad Hoc Multiplayer Games," in *IEEE 17th Int. Conference on Parallel and Distributed Systems*, Dec. 2011.

[4] A. Le, L. Keller, C. Fragouli, and A. Markopoulou, "MicroPlay: a networking framework for local multiplayer games," in *Proc. of the 1st ACM Int. Workshop on Mobile gaming*, New York, USA, Aug. 2012.

[5] O. Turkes, H. Scholten, and P. J. Havinga, "BLESSED with Opportunistic Beacons: A Lightweight Data Dissemination Model for Smart Mobile Ad-Hoc Networks," in *Proc. of the 10th ACM MobiCom Workshop CHANTS '15*. ACM Press, Sep. 2015.

[6] A. Sathiaseelan and J. Crowcroft, "Internet on the move: Challenges and solutions," *SIGCOMM CCR*, vol. 43, no. 1, Jan. 2012.

[7] M. Conti and S. Giordano, "Mobile ad hoc networking: milestones, challenges, and new research directions," *IEEE Comms Magazine*, Jan. 2014.

[8] W. Kiess and M. Mauve, "A survey on real-world implementations of mobile ad-hoc networks," *Ad Hoc Networks*, vol. 5, no. 3, Apr. 2007.

[9] 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," *IEEE Comms Magazine*, vol. 54, no. 1, Jan. 2016.

[10] IEEE Association, "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2012*, pp. 1–2793, Mar. 2012.

[11] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," IETF, RFC 3626, Oct. 2003.

[12] Wi-Fi Alliance Technical Committee P2P TG, "Wi-Fi Peer-to-Peer (P2P) Technical Specification v1.2," WiFi Alliance, Tech. Rep., 2010.

[13] P. Gardner-Stephen and S. Palaniswamy, "Serval mesh software-WiFi multi model management," in *Proc. of the 1st Int. Conference on Wireless Technologies for Humanitarian Relief - ACWR '11*, NY, USA, Dec. 2011.

[14] S. Cheshire, B. Aboba, and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses," IETF, RFC 3927, May 2005.

[15] A. Aguiar, E. Soares, P. Brandão, T. Magalhães, J. M. Fernandes, and I. Oliveira, "Demo: Wireless IP Mesh on Android for Fire-fighter Monitoring," in *Proc. of the 9th ACM MobiCom workshop CHANTS '14*, New York, USA, Sep. 2014.

[4] https://arxiv.org/abs/1702.04249