A Survey on Smartphone Ad Hoc Networks

Matouš Skála
Delft University of Technology
Email: M.Skala@student.tudelft.nl

Abstract—**Over the past few years, smartphones have become powerful devices that come equipped with numerous network connectivity modules. In addition to Bluetooth BR/EDR and Bluetooth Low Energy, most recent devices also come with support for Wi-Fi Direct and Wi-Fi Aware standards to facilitate the interconnection with nearby devices over longer distances and with higher bandwidth.**

In this survey, we discuss the current implementation of various connectivity APIs in Android OS and explore possibilities of deploying a large-scale smartphone ad hoc network. Additionally, we design a proof of concept Android application to demonstrate the feasibility of mesh networking within the constraints imposed by Android OS.

Index Terms—**mesh networks, mobile ad hoc networks**

1. Introduction

The increasing popularity of smartphones and novel wireless networking technologies opens up possibilities to a whole new range of applications that can communicate without the need for the Internet connection. Prospective use cases are ranging from proximity-based social networks, infrastructure-less communication with *Internet of Things (IoT)* devices, student attendance tracking, to communication between attendees during music festivals or protests, where the cellular network is either overloaded or censored [1].

Starting from Android 4.0, *Wi-Fi Direct* [2], also referred to as Wi-Fi peer to peer, is directly supported by the system, allowing device to device Wi-Fi communication without an additional *access point (AP)*. From Android 8.0, the *Wi-Fi Aware* [3] standard, also known as *Neighbor Awareness Networking (NAN)*, has been supported, allowing to automatically form clusters of nearby devices.

Wi-Fi generally offers a higher range of coverage and bandwidth than *Bluetooth*, so it might be more suitable for data-intensive applications such as photo or video sharing. On the other hand, Bluetooth is supported on a wider variety of devices and especially *Bluetooth Low Energy (BLE)* [4] can result in significantly lower battery consumption, thus it is more suitable for transferring small amounts of data.

This survey is structured as follows. In Section 2, we first explore features of all previously mentioned wireless communication protocols. In Section **??**, we discuss their implementation and possible limitations within Android OS. In Section 4, we explore different Android applications taking advantage of mesh networking for censorship resilient messaging. Finally, in Section 5 we present our proof of concept for deploying an ad hoc network with multi-hop routing on Android. Section 6 concludes this work.

2. Wireless Communication Technologies

2.1. Bluetooth

Bluetooth is a wireless data exchange standard developed by Bluetooth SIG[1] and supported by the majority of mobile devices. To connect two devices, one device (a *server*) first needs to make itself discoverable. On Android, the application can request device discoverability for a specified duration, but this action needs to be confirmed by the user in a dialog presented by the system. Then, other devices (*clients*) can start scanning to find MAC addresses of nearby Bluetooth devices.

It is possible to establish a channel between two devices using a *RFCOMM*[2] protocol. First, the server opens a server socket. Clients can then use the discovered MAC address to initiate the connection with the server. This method allows to open a secure channel either between two *paired* devices, or even an insecure one without need of going through the pairing process.

When two devices are linked, one of them takes the role of a *master*, the other one is a *slave*. There can be up to 7 slaves connected to a single master and such a network is called a *piconet*. A device can only be a master of a single piconet. However, since Bluetooth 4.1, it is possible for a slave to connect to multiple masters, or for a master to also act as a slave in another piconet. A network consisting of multiple piconects is referred to as a *scatternet*. [5]

2.2. Bluetooth Low Energy

BLE was first introduced in the Bluetooth 4.0 specification as a power-efficient way to connect with peripherals to exchange small amounts of data. It offers a similar range as Bluetooth with a lower throughput, but a significantly lower power consumption in an idle state, making it suitable for sending short bursts of data.

Like Bluetooth, BLE operates in the 2.4 GHz ISM[3] radio band. There are 40 physical channels, 3 of them are

[1] Special Interest Group
[2] Radio frequency communication
[3] industrial, scientific and medical

used for advertising and 37 are used as data channels. Devices that transmit unidirectional broadcasts on advertising channels are called *advertisers*. Devices that receive data on advertising channels are called *scanners*. Advertisement transmission occurs in advertising events. At the beginning of each event, the advertiser sends an advertising packet. Upon detection of the advertisement, the scanner may send a scan request which is then followed by a scan response from the advertiser. The advertisement packet can be sent in all 3 advertising channels during the same advertising event. Both advertisement and scan response size are limited by 31 bytes, but can be extended to 255 bytes by extended advertisements in Bluetooth 5.0. [5]

If the advertising event is marked as connectable, the advertiser may receive connection requests from *initiators*.

BLE devices can act in two roles: *centrals* and *peripherals*. The peripheral device starts by broadcasting an advertising packet limited to the size of 31 bytes. Central devices can then run a scanning process and filter advertising packets based on the MAC address, service UUID, or other parameters. It is important to note that the MAC received in advertising packets is not the physical MAC address if the devices are not paired, so it cannot be used to create the Bluetooth socket. [6]

Instead, BLE devices usually communicate using the *GATT*[4] server. Both central and peripheral can act as a server or client. The GATT server allows to store short pieces of data known as *attributes* in form of *characteristics* and *descriptors*. Once the client establishes the connection with the server, it can read or write supported attributes.

However, the advertising packet can also be abused to include the physical MAC address as part of the service UUID. Once the receiver extracts the physical MAC address from the packet, it can connect to the Bluetooth socket of the server. This has been up to date probably the most accessible way to connect two devices without user interaction. However, starting Android 8 it is no longer possible to obtain local MAC address programatically, so this solution would now require the user to copy the device MAC address from the system settings.

From Android 10, there is also a possibility to form a *L2CAP*[5] channel between Bluetooth LE devices.

2.3. Wi-Fi Direct

There have been many attempts to enable direct communication between IEEE 802.11 radio devices. The 802.11 standard defines two operating modes. Next to the traditional *infrastructure* mode, there is an *ad-hoc* mode which allows device-to-device communication. However, the ad-hoc mode is not supported by Android OS.

Wi-Fi Direct (also known as Wi-Fi Peer-to-Peer) [7] is a IEEE 802.11 based protocol released by Wi-FI Alliance in 2009 and supported from Android 4.0. With Wi-Fi direct, devices are organized in groups, where one device is the Group Owner (GO) and the rest are Group Members (GM). The roles are not predefined, but are negotiated during the group formation process. Groups are able to support Legacy Clients (LC), which means that even devices without Wi-Fi Direct support can join as group members.

2.3.1. Multi-group communication.

The standard only defines intra-group communication, but it does not restrict a Group Member from participating in multiple groups simultaneously. Moreover, a device can theoretically connect both to Wi-Fi Direct and an infrastructure network by using multiple virtual MAC entities. However, these functionalities are not defined by the standard and thus depend on the implementation. In Android, the GO always has the same hardcoded IP address (192.168.49.1), while GM are assigned a random address from the same subnet (192.168.49.2-254). As a result, the scenario where the device would act in multiple groups at the same time cannot be directly implemented.

There are several workarounds proposed in [8]. The first solution is to use *time sharing* to allow a device to act as a gateway between multiple groups. It requires a device to periodically disconnect and reconnect to different groups and effectively act as a relay passing messages between multiple groups. Another solution takes advantage of *simultaneous connections* using Wi-Fi Direct and the traditional Wi-Fi. The GO advertises its group using a unique Service Set ID (SSID) that can be used by other clients not supporting the framework to join the group. Experiments have shown that it is possible to create a LC/GM gateway node when communicating using multicast UDP datagram sockets. Due to routing-related issues, this solution does not work with traditional datagram or stream sockets. The authors also propose a more efficient *hybrid* protocol that uses multicast sockets as a control channel that triggers gateway configuration change if needed. However, all solutions rely on undocumented behaviors, which means their reliability can vary across devices or Android OS versions.

2.4. Wi-Fi Aware

Wi-Fi Aware, also known as *Neighbor Awareness Networking* (NAN) is a recent networking standard introduced by Wi-Fi Alliance. [9] It works by forming clusters with nearby devices. The discovery process starts when one device (a *publisher*) publishes a discoverable service. Other devices (*subscribers*) who subscribe to the same service will receive a notification once a matching publisher is discovered. After the subscriber discovers a publisher, it can either send a short message or establish a network connection with the device. A device can be both a subscriber and a publisher simultaneously.

2.4.1. Ranging peers.

Devices that are equipped with *Wi-Fi Round Trip Time* (RTT) can also directly measure distance to peers. This information can further be used for location-aware service discovery. Specifically, the discovery process can be restricted to only include services within a particular

[4] Generic Attribute Profile

[5] Logical link control and adaptation protocol

range specified by a minimum and maximum distance in meters.

2.4.2. Wi-Fi Aware on Android. Wi-Fi Aware is supported by Android OS since the version 8.0. However, it is not required by Google for devices to support this standard. During our testing, it was available only on a small subset of potentially capable devices. For example, Google Pixel and Samsung Galaxy S9 do not support Wi-Fi Aware, while they officially support Android later than 8.0. In fact, Google Pixel 2 XL and 4 XL were the only supported devices we were able to test. This makes it impractical to use it as the only communication protocol even for new apps.

To our best knowledge, there have been no attempts to utilize Wi-Fi Aware as a communication layer for mesh networks so far. It has a higher throughput and range then Bluetooth. It also should not suffer from the issues related to Wi-Fi Direct described previously. However, reluctant support from device manufacturers prevents it from being useful for a wide-scale deployment in the near future.

2.5. Nearby Connections API

All previously mentioned technologies can be accessed on Android directly using their corresponding APIs. However, in some cases, the behavior depends on the version of Android OS or even device-specific implementation. Moreover, every API is slightly different, and taking advantage of multiple technologies simultaneously takes considerable effort. For that reason, Google introduced Nearby Connections API in 2015 and its fully-offline version 2.0 in 2018. [10] The API aims to large extent simplify development of Android applications relying on peer to peer communication. It is able to automatically switch between Bluetooth, BLE, Wi-Fi, Wi-Fi Direct, and Wi-Fi Aware depending on the application needs and device capabilities.

It supports multiple strategies for different network topologies, specifically cluster, star, and point to point. The cluster topology is the most general one allowing m-to-n connections. However, in the current version, the cluster stratgy results in lower bandwidth connections as it uses only Bluetooth connectivity.

The library is closed-sourced, but it has been previously reverse-engineered and analyzed by security researchers. Several vulnerabilities have been discovered and disclosed to Google, including impersonation, man-in-the-middle attacks, attacker-induced physical layer switch, DoS, or radio state manipulation. [11]

The library is distributed as part of Google Play Services, which means it is only available on devices that are certified by Google. This also prevents it from being used in countries where access to Google is restricted. This is not compatible with our idea of censorship resilient communication.

3. Routing Protocols

There are several approaches for disseminating content in ad hoc networks. In general, we can divide protocols into two main categories: flooding-based and routing-based.

Technology	Android	Throughput	Range
Bluetooth	2.0+	24 Mbps	$\sim$10 m
BLE	4.3+	0.3 Mbps	$\sim$100 m
BLE GATT	5.0+	0.3 Mbps	$\sim$100 m
BLE L2CAP	10.0+	0.3 Mbps	$\sim$100 m
WiFi Direct	4.0+	250 Mbps	$\sim$200 m
WiFi Aware	8.0+	250 Mbps	$\sim$200 m

Figure 1. The comparison of wireless communication technologies w.r.t. supported Android platform versions, a theoretical throughput, and range

Flooding-based solutions do not perform routing, instead packets are broadcasted by nodes and spread across the whole network. In naive flooding, *broadcast storms* can happen in case nodes broadcast the same packet repeatedly. Therefore, nodes should implement loop prevention by storing a set of recently received messages and only deliver messages that have not been received before. [12] While conceptually simple, flooding incurs high overhead in terms of transmitted messages, especially when only point-to-point communication between two devices is desired.

Routing-based protocols can be classified into tree types: *proactive* (table-driven), *reactive* (on-demand), and *hybrid*.

3.1. Proactive

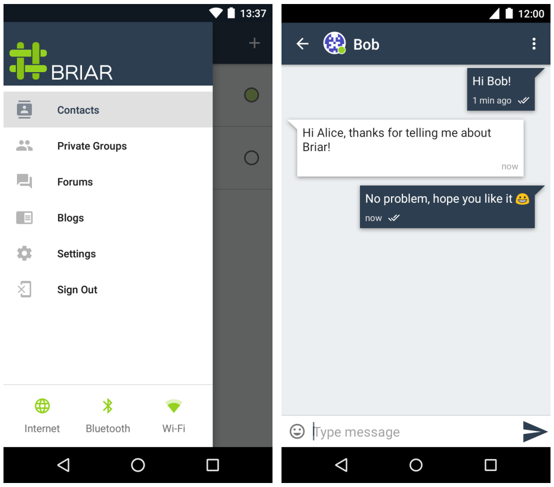
In proactive routing protocols, each node maintains routes to every other node in the network. The routes information is usually stored in routing tables and it is periodically updated as the topology changes. [13] Examples of proactive routing protocols are Destination Sequence Distance Vector (DSDV) [14] or Optimized Link State Routing (OLSR) [15].

3.2. Reactive

In reactive routing, routes are established on demand. When a node needs to send a message to the destination and the route is not known, the *route discovery* mechanism is invoked. This is usually done by broadcasting a route discovery packet through the network. Once the path between the source and the destination is known, data can be transmitted using the selected route. Due to dynamic nature of SPANs, a *route maintenance* mechanism is required to handle route breakage. Some well-known reactive routing protocols are Ad Hoc On-Demand Distance Vector (AODV) [16] or Dynamic Source Routing (DSR) [17].

4. Applications

In this section, we list the state-of-the art applications taking advantage of smartphone ad hoc networks for resilient messaging. We describe their functionality and analyze chosen communication technologies. Finally, we assess their limitations and suggest improvements.



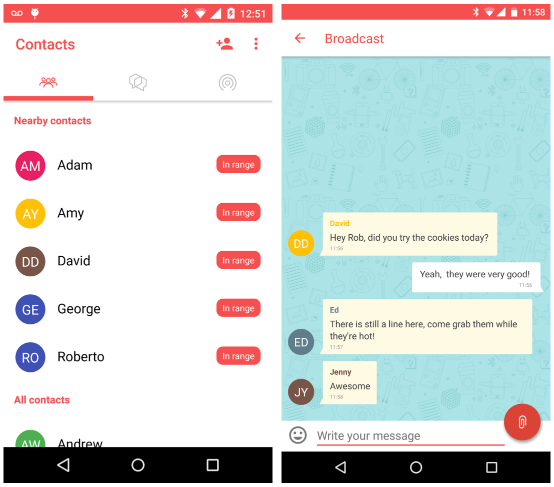
Hi Bob!
1 min ago

Hi Alice, thanks for telling me about Briar!
now

No problem, hope you like it 😀
now

Type message

Figure 2. The menu and conversation detail in Briar [18]



David
Hey Rob, did you try the cookies today?
11:56

Yeah, they were very good!
11:56

Ed
There is still a line here, come grab them while they're hot!
11:57

Jenny
Awesome
11:58

Write your message

Figure 3. The list of nearby contacts and broadcast in Bridgefy [21]

4.1. Briar

Briar [18] is an open-source project that started to support freedom of expression and right to privacy. It enables peer-to-peer encrypted messaging and forums. It is presented as a tool for activists, journalists, and anyone who needs a safe way to communicate.

Before the communication starts, users have to meet in person and scan QR codes from each other's screen. The devices exchange public keys and agree on a shared key using *Bramble QR Code Protocol (BQP)* [19]. This provides strong identities secure against man-in-the-middle attacks.

The device only accepts connections from devices in contacts. However, the user can initiate an introduction between two of her contacts. If both contacts accept the introduction request, then they are able to establish connections without meeting in person.

The communication is built on top *Bramble Transport Protocol (BTP)* and *Bramble Synchronization Protocol (BSP)*, transport and application layer protocols suitable for *delay-tolerant networks*. [20] It does not rely on any central server, but instead allows to synchronize messages using Bluetooth or Wi-Fi. If the Internet is available, it can also connect via the Tor network.

However, the application does not take advantage of mesh networking. It can only communicate with devices it is directly connected to.

4.2. Bridgefy

Bridgefy is an offline messaging app based on Bluetooth connection. It can operate in 4 modes. Person to person mode allows to chat privately with users nearby. A mesh mode allows to route traffic through devices of other users using the app. The broadcast mode allows to broadcast and see messages of all people around, even those not in the contacts. Finally, the online mode allows to send messages anywhere in the world, provided the internet connection is available.

The app is built on top of Bridgefy SDK, a multi-platform SDK for building mesh networking apps. However, both SDK and the app are proprietary. The app requires the user to set up the account and verify using a phone number. The SDK requires internet connection to verify the license. Finally, it requires a permission to access the contacts list. Therefore, while the app has proven to be useful in recent Hong Kong protests [1], it does not work fully offline and raises some privacy concerns.

4.3. FireChat

FireChat is a proprietary offline messaging app built by OpenGarden. It uses Bluetooth and Wi-Fi to communicate with other devices, and takes advantage of multi-hop technology. It allows to send messages and photos, private messages with end-to-end encryption, create private groups, and use hashtags to create public chat rooms. [22]

Similarly to Bridgefy, it requires internet connection for setting up an account. Unfortunately, we were not able to fully test the app at the time of writing this survey due to a bug in the sign up process.

4.4. The Serval Mesh

wifi ad hoc, requires root?

5. Proof of Concept

6. Conclusion

References

[1] J. Koetsier. (2019, Sep.) Hong Kong protestors using mesh messaging app China can't block: Usage up 3685%. Accessed: Sep. 24, 2019. [Online]. Available: https://www.forbes.com/sites/johnkoetsier/2019/09/02/hong-kong-protestors-using-mesh-messaging-app-china-cant-block-usage-up-3685/

[2] Wi-Fi Direct (peer-to-peer or P2P) overview. Accessed: Sep. 24, 2019. [Online]. Available: https://developer.android.com/guide/topics/connectivity/wifip2p

[3] Wi-Fi Aware overview. Accessed: Sep. 24, 2019. [Online]. Available: https://developer.android.com/guide/topics/connectivity/wifi-aware

[4] Bluetooth low energy overview. Accessed: Oct. 26, 2019. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth-le

[5] Bluetooth SIG. (2019, Jan.) Bluetooth core specification version 5.1. Accessed: Oct. 26, 2019. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/

[6] M. Woolley. (2015, Apr.) Bluetooth technology protecting your privacy. Accessed: Oct. 26, 2019. [Online]. Available: https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy/

[7] Wi-Fi Alliance, "Wi-Fi Peer-to-Peer (P2P) technical specification, version 1.7," Jun. 2016. [Online]. Available: https://www.wi-fi.org/file/wi-fi-peer-to-peer-p2p-technical-specification-v17

[8] C. Funai, C. Tapparello, and W. B. Heinzelman, "Supporting multi-hop device-to-device networks through wifi direct multi-group networking," *CoRR*, vol. abs/1601.00028, 2016. [Online]. Available: http://arxiv.org/abs/1601.00028

[9] Wi-Fi Alliance, "Neighbor Awareness Networking specification, version 3.0," Dec. 2018. [Online]. Available: https://www.wi-fi.org/file/neighbor-awareness-networking-specification

[10] R. Nayak, "Announcing Nearby Connections 2.0: fully offline, high bandwidth peer to peer device communication," Accessed: Oct. 30, 2019, Jul. 2017. [Online]. Available: https://android-developers.googleblog.com/2017/07/announcing-nearby-connections-20-fully.html

[11] D. Antonioli, N. O. Tippenhauer, and K. B. Rasmussen, "Nearby threats: Reversing, analyzing, and attacking Google's 'Nearby Connections' on Android," in *NDSS*, 2019.

[12] Thomas Zahn and Greg O'Shea and Antony Rowstron, "An empirical study of flooding in mesh networks," Tech. Rep. MSR-TR-2009-37, Apr. 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/an-empirical-study-of-flooding-in-mesh-networks/

[13] S. Mohseni, R. Hassan, A. Patel, and R. Razali, "Comparative review study of reactive and proactive routing protocols in MANETs," in *4th IEEE International Conference on Digital Ecosystems and Technologies*, April 2010, pp. 304–309.

[14] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, 05 1999.

[15] P. Jacquet, P. Muhlethaler, T. H. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," 02 2001, pp. 62 – 68.

[16] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," United States, 2003.

[17] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Comput.*, vol. 353, 05 1999.

[18] Briar Project. Briar. Accessed: Oct. 26, 2019. [Online]. Available: https://play.google.com/store/apps/details?id=org.briarproject.briar.android

[19] ——. Bramble QR code protocol, version 4. Accessed: Oct. 26, 2019. [Online]. Available: https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BQP.md

[20] ——. A quick overview of the protocol stack. Accessed: Oct. 26, 2019. [Online]. Available: https://code.briarproject.org/briar/briar/wikis/A-Quick-Overview-of-the-Protocol-Stack

[21] Bridgefy. Bridgefy – offline messaging. Accessed: Oct. 26, 2019. [Online]. Available: https://play.google.com/store/apps/details?id=me.bridgefy.main

[22] Open Garden. FireChat. Accessed: Oct. 30, 2019. [Online]. Available: https://www.opengarden.com/firechat/