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 3, 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.

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

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

2.3. WiFi Direct

2.4. WiFi Aware

3. Implementation in Android OS

3.1. BluetoothManager

3.2. WifiP2pManager

3.3. WifiAwareManager

3.4. Nearby Connections API

4. Applications

4.1. Briar

4.2. Bridgefy

4.3. FireChat

4.4. The Serval Mesh

4.5. B.A.T.M.A.N.

5. Proof of Concept

6. Conclusion

References

[1] J. Koetsier, "Hong Kong protestors using mesh messaging app China can't block: Usage up 3685%," *Forbes*, Sep. 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] (2019, Sep.) Wi-Fi Direct (peer-to-peer or p2p) overview. [Online]. Available: https://developer.android.com/guide/topics/connectivity/wifip2p

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

4. Generic Attribute Profile
5. Logical link control and adaptation protocol

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

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

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