University of Basel

Filedrop

Project report

Distributed Programming and Internet Architecture
Department of Mathematics and Computer Science

Illia Solohub, Jennifer Jäger, Valdrin Sheremetaj

20/07/2025

Abstract

Filedrop is a tool for sharing files between Android devices that do not support Apple's Airdrop. It uses Bluetooth Low Energy (BLE) to quickly establish a connection between two devices, by performing a "BLE handshake" which exchanges basic information. Then it switches to Wifi-Direct to actually send the files. This will make the file transfer both fast and wireless, without needing an Internet connection.

Table of Contents

1 Introduction

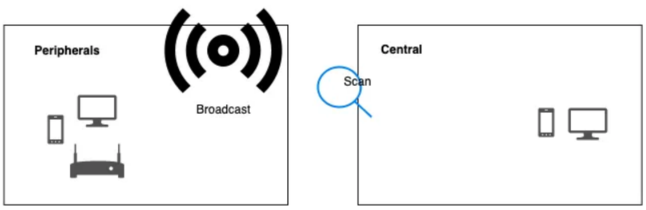
Airdrop is a widely used functionality provided by the Apple ecosystem. It makes it possible to send files between Apple devices seamlessly using a combination of Bluetooth and Wifi-technologies. This peer-to-peer communication takes place automatically in the background, which means that users don't have to think about network configurations or IP addresses. They just select a file and the target device they want to send the file to. This process is not only well-integrated into user experience, but also fast, reliable and does not need an Internet connection. In this project, we tried to mimic a similar file-transferring behavior by using the properties of Bluetooth Low Energy (BLE) and Wifi-Direct on Android Devices. We have a sending and a receiving peer. The sending peer sends a file by scanning for devices over BLE and connects to them afterwards. When it connects to the receiving peer, the receiving peer sends

2
Background

In essence to provide a meaningful explanation of how the project is designed and implemented a reader would need to be acquainted with the two following concepts: Bluetooth Low Energy (BLE) and Wifi-Direct.

2.1 Bluetooth Low Energy (BLE)

Bluetooth Low Energy (Bluetooth LE, colloquially BLE, formerly marketed as Bluetooth Smart) is a wireless personal area network technology designed and marketed by the Bluetooth Special Interest Group (Bluetooth SIG) aimed at novel applications in the healthcare, fitness, beacons, security, and home entertainment industries. The original specification was developed by Nokia in 2006 under the name Wibree, which was integrated into Bluetooth 4.0 in December 2009 as Bluetooth Low Energy. Compared to Classic Bluetooth, Bluetooth Low Energy is intended to provide considerably reduced power consumption and cost while maintaining a similar but smaller communication range. It is independent of classic Bluetooth and has no compatibility, but Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) and LE can coexist. It is primarily useful for Filedrop (and Airdrop as well), because of it's proximity-based discovery. It fits well into the "share a file with someone near you" model, while still keeping a fast discovery speed.
Operating systems including iOS, Android, Windows, MacOs and Linux natively support Bluetooth Low Energy.
How it works under the hood:



We have 2 types of devices: **peripheral** and **scanning**. The **peripheral** devices are advertising their presence with certain time intervals letting the devices that are scanning know about them. After the scanning device has found the advertising device it was looking for we perform a connection and share reasonably small quantities of data. Doing that we are saving energy hence it is called low energy. While doing this they exchange metadata like MAC addresses or Generic Attribute Profile (GATT) containing services and characteristics. A Generic Attribute Profile (GATT) is the protocol used for data exchange once two BLE devices are connected. It can be considered as a database which defines how devices describe and access data over BLE, like services and characteristics. Characteristics are single data points inside services, which define the type of data, what value it has and how we can access them. On the other hand, are services a collection of functionalities, which are identified by a UUID and group together different characteristics. GATT is an important concept, which organizes data into logical groups. The only ”downside” to Bluetooth Low Energy (BLE) is it’s slow transferring speed. Only using BLE to transfer a file is too slow for today’s standards, because of its maximal theoretical output of approximately two mbps, large files are almost unfeasible to send. That’s why after initial discovery and connection, we need a faster way to actually transfer the file. This is possible due to Wifi-Direct.

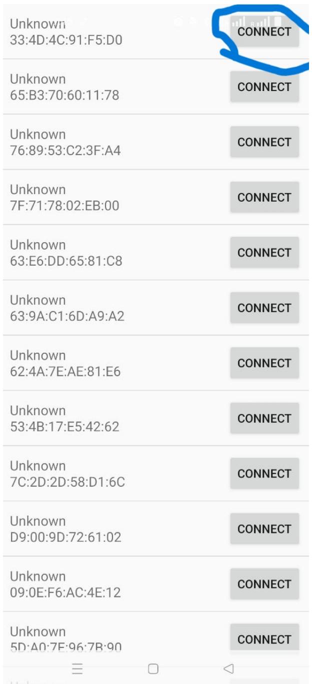
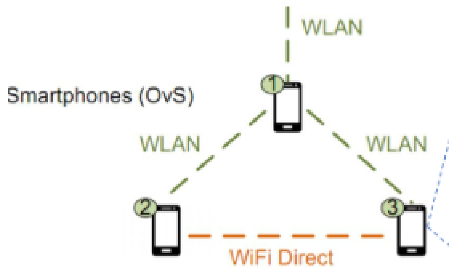


Figure 2.1: BLE Scanning and Connection in Filedrop Application

2.2 Wifi-Direct

Wifi-Direct is a Wifi standard for wireless connections that allows two devices to establish a direct Wifi connection without an intermediary wireless access point, router, or Internet connection. It is a single-hop communication, rather than multi-hop communication like wireless ad hoc networks. Devices using Wifi-Direct can discover each other in high range, establish a connection, and communicate at typical Wifi speeds, which are significantly faster than Bluetooth. Once a BLE handshake is complete and the receiving device has been identified, we initiate the Wifi-Direct phase to handle the actual file transfer. At this point, the receiving device broadcasts its availability using a Wifi Direct service discovery. The sending device listens for these broadcasts and, when it detects the correct service, requests to connect. This initiates the Wifi-Direct group formation process, where one device becomes the Group Owner (essentially acting as a temporary access point), and the other connects as a client. After the connection is established, both devices are on the same local network, even though no router or external Wifi is involved. From here, the sender can initiate a high-speed file transfer, typically over a TCP or UDP socket. This makes Wifi-Direct ideal for high-bandwidth tasks such as sharing of files, like Filedrop.



Although BLE and Wifi-Direct serve different purposes, they work hand in hand in this setup. BLE is lightweight, quick to connect, and energy-efficient and therefore perfect for the initial handshake and service discovery. Wifi-Direct, on the other hand, takes over when high-speed data transfer is needed. By combining the two, we achieve both seamless user experience and efficient performance.

3 Implementation

The app was developed in Android Studio using Kotlin, with a focus on combining BLE and Wifi-Direct in a smooth, modular way. BLE is used for the initial handshake where devices can advertise their name and scan for nearby peripherals. The BLE connection is managed through a small utility class BLEConnectionManager that wraps around Android's GATT callbacks. Once connected, the name from the BLE device is stored and later reused to identify the corresponding peer in the Wifi-Direct phase.

Peer discovery and connections over Wifi-Direct are handled by the WifiDirectManager, which wraps the WifiP2pManager logic and makes it easier to coordinate group creation, discovery, and connection retries. The device that was connected via BLE is matched by name again when scanning for Wifi-Direct peers. It is important to note that inside the code, we hardcoded the Wifi-Direct name. We couldn't figure out how to map the BLE name of the connected peer, to it's actual Wifi-Direct name properly, to ensure a better user experience. The problem is that often times the advertised BLE name is not the same as the Wifi-Direct name, a possible solution to this issue is to implement a shared identifier exchange during the BLE handshake. For example, the BLE advertisement or scanner could include a custom field like a unique device ID that both sides agree on ahead of time. Once a BLE connection is established, the initiating device can request this identifier as part of a GATT characteristic. The receiver can then match this identifier with the name of a discovered Wifi-Direct peer during group formation. But when the advertised Wifi-Direct name is clear, it is possible to just change the name advertised by BLE in PeripheralService with bluetoothAdapter.setName("Wifi-Direct name"). This will work fine, but is not perfect in any means.

Once both devices are connected via Wi-Fi Direct, the actual file transfer is triggered. This is done using a foreground FileTransferService, where we send files over a classic TCP socket. Metadata like filename and MIME type is sent first, followed by the file content. On the receiving side, files are saved directly to the system's Downloads folder using Android's media APIs, while respecting storage.

4
Manual

We will use this chapter to guide users through a Filedropping Mechanism.

4.1 Prerequisites

1. Android Studio must be installed.
2. The project should be built and installed on two physical Android devices (Sender and Receiver).
3. Both devices must have Bluetooth, Wifi and Location enabled.

4.2 Receiver Setup

1. Install and launch the app on the **receiver** device.
2. Press the **Advertise** button to:
   - Start **BLE advertising**.
   - Initialize **Wi-Fi Direct group formation**.
3. The receiver is now waiting for incoming BLE and Wi-Fi Direct connections.

4.3 Sender Setup

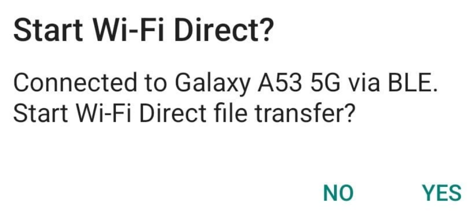
1. Launch the app on the **sender** device.
2. Press the **Start Service** button to begin scanning for nearby BLE devices.
3. The app shows all available BLE, look for the right peer.
   Note: The BLE device name is currently hardcoded (change to right Wifi-Direct Name, typically product name in telephone information)

   ```
   bluetoothAdapter.setName("Galaxy A53 5G")
   ```

4. Once the correct BLE device is found, press the **"Connect"** button to establish a BLE connection.

4.4 Wifi-Direct Prompt

Upon successful BLE connection, a dialog prompts the user to initiate Wifi-Direct:



- Press **"Yes"** to initiate the Wi-Fi Direct connection.
- The sender joins the group formed by the receiver.

4.5 File Transfer

1. Once the Wi-Fi Direct connection is established, the file picker opens on the sender.
2. Select a file to send.
3. The file is transferred over Wi-Fi Direct to the receiver.
4. The receiver saves the file in the `Downloads` folder.

5

Experiments and Evaluation

To validate the effectiveness of our system, we conducted a small series of experiments across multiple Android devices. We tested BLE discovery time, Wifi-Direct connection time, and file transfer reliability across different file sizes. The result correspond to ten tries.

5.1 Test Devices

- Samsung Galaxy A53 5G (Android 13)
- Samsung Galaxy S23 FE (Android 14)
- OnePlus 10 Pro 5G (Android 12)

5.2 Metrics Measured

- Time to discover peer via BLE
- Time to establish Wifi-Direct connection
- File transfer success rate and duration (for 1 MB, 100 MB, and 500 MB files)

5.2.1 Results Summary

Test Case	**Avg. Time (s)**	**Success Rate**	**Notes**
BLE discovery	1.5 – 3.0	100%	Varies slightly with range
Wi-Fi Direct connect	3.0 – 5.0	100%	Occasional retries needed
File transfer (1 MB)	¡1.0	100%	Instantaneous
File transfer (100 MB)	2.5	100%	Stable
File transfer (500 MB)	8.0	80%	Two failures on OnePlus device

Table 5.1: Performance results of BLE + Wifi-Direct system

5.2.2 Discussion

Overall, the system performs reliably across devices and file sizes. BLE discovery is fast and consistent, and Wifi-Direct achieves high throughput during file transfers. The only significant challenge encountered was occasional instability in Wifi-Direct connection initiation, which may be addressed through improved timeout handling or retry logic.

6
Conclusion

In this project, we successfully developed a hybrid file transfer system for Android that leverages **Bluetooth Low Energy (BLE)** for low-power device discovery and pairing, followed by **Wifi-Direct** for high-speed, peer-to-peer file transfer. This approach provides a seamless and efficient user experience without requiring an access point or internet connection. While the system reliably establishes both BLE and Wi-Fi Direct connections, some limitations remain. Particularly, the inability to map the BLE device name to the Wifi-Direct device name required us to hardcode the expected BLE name, which reduces flexibility. Addressing this could significantly improve user experience and system robustness.

Despite this limitation, the combination of BLE and Wifi-Direct proves to be effective for proximity-based file sharing. The two-stage process mirrors the architecture used in commercial systems like Apple's AirDrop, demonstrating the real-world applicability of our design.

Future work may include implementing dynamic name resolution between BLE and Wi-Fi Direct, improving cross-device compatibility, and exploring more robust connection fallback mechanisms.