Composable Distributed Android Applications

Dale Pedzinski
Dept of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX USA
dpedzinski@pedz-tech.com

James Autry
Dept of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX USA
jtautry@utexas.edu

***Abstract*— The purpose of our paper is to explore the composition of distributed applications on the Android operating system. The paper provides an overview of the operating system and its networking capabilities. In addition, discusses the Wi-Fi Peer-to-peer (P2P) module to handle connectivity of devices in a distributed system. Finally, to bring these concepts to life an application is built to the confirm the viability of such an application.**

Keywords—Android, Distributed systems, Wi-Fi Direct, Peer-to-peer

I. Introduction

The growing presence of mobile devices sparked our interested in creating an application that would allow ad-hoc networking across multiple devices. We decided we wanted to created something that would be able to join a network of devices without the need for setup files or physical connection because it doesn't seem practical in today's world of virtual and wireless devices to have either of these. However, we had to select the correct platform given the three-week timeframe and with neither of us experts in mobile development, we decided to tackle a platform that would utilize our Java programming skills. With this in mind, we selected the Android platform since it is primary Java and seems to have a lower learning curve than IOS. The paper briefly goes over some important contextual knowledge about Android then narrates our journey of creating a WiFi P2P application. However, throughout the journey, we explore opportunities for improvements, lessons learned and how a lot of these concepts are a real-world example of how classroom concepts are productized. If you are curious, the application in its entirety is available at this repository on GitHub: https://github.com/pedzindm1/distrSystems/tree/master/Term%20Project.

II. Android

A. Overview

At its core, Android is an open-source, Linux-based mobile operating system. Android's install base includes not only mobile phones, but devices such as e-readers, appliances, automobiles, and wearables. It is developed and maintained by Google, and regular updates are released as open-source to the Android Open Source Project (AOSP) [1], [2]. Original equipment manufacturers (OEM), such as Sony, HP, Amazon, NVIDIA, and Samsung, are then free to fork the codebase and develop customized, proprietary operating systems for their devices license free [9].

B. Availability and Market Share

Android is the most popular mobile operating system in the world. According to one Gartner press release, Android devices sold in the third quarter of 2016 accounted for 87.8% of the market share (compared to 11.5% for Apple's iOS devices) [8]. The estimated total install base for the Android operating system is 2.66 billion devices, with iOS a distant second at 589 million [10]. The open nature of the operating system, combined with its prevalence in the market, make Android a popular platform for application development.

C. Platform and API Versioning

The major releases of the Android platform are referred to as the platform version and are typically codenamed with the name of a dessert, with the first letter incremented for each name. The current platform version is 8.0 and is codenamed "Oreo" [4], [7]. Additionally, each platform version supports a specific application programming interface (API) level that lets developers know which APIs are supported. For each application, developers can specify both a minimum supported API level and the target API level. The API level is also referred to as a software development kit (SDK) version [7].

Fig. 1 shows the percentage of devices running each version of the Android platform [4]. The API levels highlighted (17 through 26) represent 96.8% of all devices.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.2%
4.2.x		17	3.1%
4.3		18	0.9%
4.4	KitKat	19	13.8%
5.0	Lollipop	21	6.4%
5.1		22	20.8%
6.0	Marshmallow	23	30.9%
7.0	Nougat	24	17.6%
7.1		25	3.0%
8.0	Oreo	26	0.3%

Fig. 1. Platform version data collected in November 2017 [4]

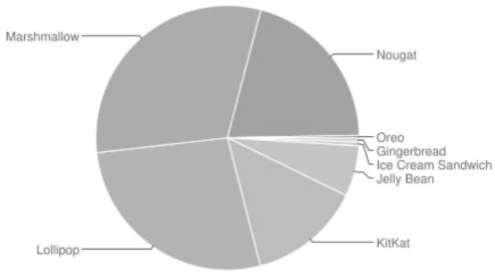

Fig. 2. Graphical representation of data from Fig. 1 [4]

III. Networking

A. Overview

Android supports a variety of technologies that allow devices to communicate with one another. For the purposes of our application, we are primarily concerned with those that allow data transfer among devices on an ad hoc basis without the need for an access point, such as Bluetooth, Near Field Communication (NFC), Wi-Fi Aware, and Wi-Fi peer-to-peer (P2P) [12]. We eliminated Bluetooth as an option due to the complexity of setting up connections via pairing and the limited range [13]. Additionally, the ad hoc network created via Bluetooth, called a piconet, is limited to only eight devices [17]. NFC connectivity was not a viable option because of the requirement that the devices be within a few centimeters of one another to connect, or that they be attached to an underlying network of NFC transceivers [16], [18]. Wi-Fi Aware is a very promising protocol that is more reliable than Wi-Fi P2P and eliminates some of the complexity involved with the discovery of devices and setup of the ad hoc network [14]. However, it is only supported in API level 26 and above, representing less than 1% of available devices [4], [14].

B. Wi-Fi P2P (Wi-Fi Direct)

Ultimately, we chose to utilize Android's Wi-Fi P2P API for setting up the ad hoc network. Wi-Fi P2P, known more generally as Wi-Fi Direct, is broadly supported among the Android code base [15]. In a traditional Wi-Fi setup, the network is created and maintained by an access point to which clients then connect. The functionality of the different types of nodes on the network is static, i.e. a client cannot function as an access point and vice versa. With Wi-Fi Direct, this functionality is transient, whereby a device can function as both a client *and* an access point and can in fact perform both roles simultaneously [5].

C. Equations

The first step in the creation of an ad hoc network with Wi-Fi Direct is for the P2P Group to be created. The P2P Group functions just like a regular Wi-Fi network, wherein the device that is performing the AP role is called the P2P Group Owner, referred to in this paper as group owner, and subsequent devices connecting to the network are P2P Clients, referred to as clients for this paper[5]. IP addresses are assigned to the clients by the group owner which serves as a Dynamic Host Configuration Protocol (DHCP) server [5]. A major limitation of Wi-Fi Direct is that it does not natively provide a mechanism for the group owner to be reassigned in the case of failure, i.e. the network is no longer available and must be re-established for communication between clients to occur [5].

IV. The Application

A. Overview

Building upon the Wi-Fi Direct Protocol, we explored ideas about how to visualize the connected data of the system across multiple Android devices, then it came to us in the form of application metadata installed on devices. Each user has their own list of applications installed on their device, so it would be beneficial to create an application, that relies on Wi-Fi P2P, to see which applications within a network are most popular. This is the general direction that guided us to the application but we wanted to incorporate concepts from class like Socket programming and Fault Detection into the application to make it both beneficial to the casual Android user and as well as learning exercise [20].

B. Compatibility

When we started building this application, we quickly realized that we wanted it be very inclusive of all Android versions so our minimal Android version required to run SDK 16, which explained earlier, allows close to 99% of devices to be our user base. Looking back on this decision, it became both a hindrance and a blessing because most of the new Wi-Fi protocols like Wi-Fi Aware are only supported in the newer API SDKs. But in terms of viability of the application and project, we learned that the Emulator, that allows a mobile device to be mocked on your computer, doesn't support networking activities like Wi-Fi P2P. This was fine because we

happened to find two Android devices running SDK 21 and 22 to build the application on. These devices became our debugging platform, but due to the lack of documentation around Wifi P2P, we discovered that the Wi-Fi service is required to be on for most of the P2P actions to work correctly. Additionally, the development of this application was very slow going because every run of the software application required a full deployment to the Android device, then as the network communication started it required a full deployment to both Android devices.

C. Application Operation

Despite these limitations, we continued to build an application that utilizes several native Android components, such as the Package Manager, Wi- FiP2PManager and the Local Storage option. Fig. 3 is a screenshot of the output resulting from the combination of the Package Manager and the Local Storage application. This is the first step in the process of getting and building a list of the most common applications across the system.

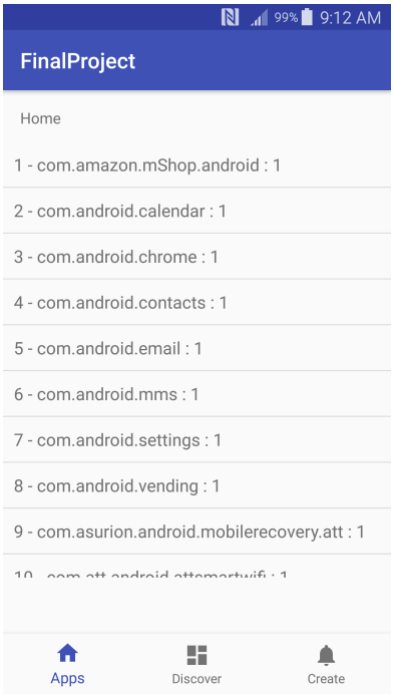

Fig 3. Screenshot of the application names captured from the application.

Fig. 4 describes the operations in the Android application. The first chevron describes the operations around obtaining the application list. The next down, describes the process around Wi-Fi P2P and connecting the system together. The last one describes the connection of the devices and the manipulation of the application data to sort and combine the data for display.

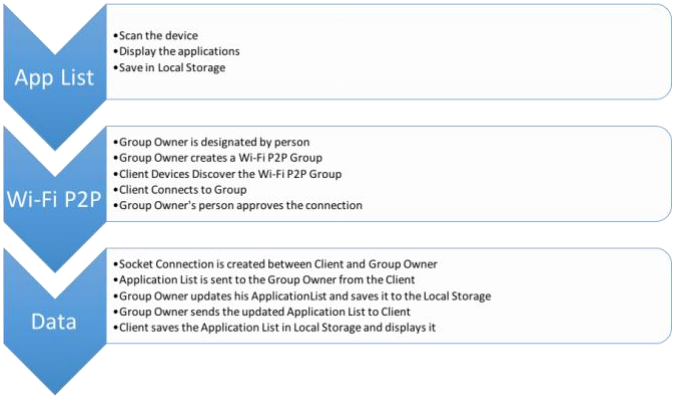

Fig. 4. Application operation

D. Wi-Fi P2P Implementation

In an attempt to jump to the meat of the application, we are going to skim over how we obtain of the application described in fig. 4 and go into how Wi-Fi P2P works within the application. Android's example application for getting started on Wi-Fi P2P was used as a starting point for our work here [19]. This tutorial examines the different components, described in detail below, but as mentioned before, the tutorial and the code examples were very outdated and lacked the context necessary to be useful for the casual developer so a lot of trial and error was done to figure out the necessary order of operations within the application.

To start off, there are four main pieces that work in unison to make this work. They are the broadcast receiver, channel, Wi-FiP2PManager and the Intent filters, which together, allow the device to be notified about changes in the network, create a Wi-Fi P2P and discover/connect to a Wi-Fi P2P Group.

To continue along the process, the following Intent Filters are the first components to initiate the Wi-Fi Network. There are a lot of Intent Filters within Android that allow the notification and callback of certain events into this part of the code. For example, the WI-FI_P2P_STATE_CHANGED_ACTION is triggered on the initial load of the application to determine, if the dependent Wi-Fi service, is enabled and if the service is toggled on/off throughout the lifecycle of the application.

Intent Filters used in the application:

- WI-FI_P2P_STATE_CHANGED_ACTION - This intent filter is triggered on initial load of the Android application and on any action that effects the Wi-Fi service or if the Wi-Fi service status changes
- WI-FI_P2P_PEERS_CHANGED_ACTION – This intent filter is triggered when the Wi-Fi Manager calls the method call request or discover peers within the application. This means either the client is searching for a group to join or the group owners is searching for a list of clients within it's network.
- WI-FI_P2P_CONNECTION_CHANGED_ACTION – This intent filter is triggered when the Wi-FiManager

creates or deletes a connection between a client and the group owner.

E. Group Discovery

There are two parts to groups with Wi-Fi P2P. The first is the client part and the second is the group owner part. But before we start creating groups, let's go over the requirements that must be satisfied before the Wi-Fi P2P group can be created. 1) A device can't be a part of an existing Wi-Fi P2P group. 2) A device must have Wi-Fi enabled throughout the process.

Another key note is that the client cannot connect to a Wi-Fi P2P group unless it is created by a group owner device first. Now, let's examine the order of execution to create a group by the group owner.

To a create a group, the group owner device would execute this method:

```
mManager.createGroup()…
```

This method call would register that device as the group owner and start broadcasting this device as a connectable endpoint for client devices to connect too.

The client devices have a bit more code to examine to connect to the new formed group:

```
mManager.discoverPeers()…
```

The discoverPeers method call discovers the newly formed group within the BroadcastReceiver class and trips the WI-FI_P2P_PEERS_CHANGED_ACTION Intent filter. This filter, then allows the client to discover the peers that the client can connect too, which leads us to the next operation below.

```
mManager.requestPeers(mChannel, new Wi-FiP2pManager.PeerListListener() {
    @Override
    public void onPeersAvailable(Wi-FiP2pDeviceList peers) {
```

The above method requests the list of peers so the application can display the list of options to the user so they can decide which group to connect too. Once the desired group is selected, then the client executes the last method below to connect to the group.

```
mManager.connect()…
```

The connect method initiates the connection to the group and asks for permission from the Group Owner's person to confirm before the final connection is established. See fig. 5 below for an example of the confirmation. This is one of the drawbacks of Wi-Fi P2P as there is no programmatic way to establish a connection without a human.

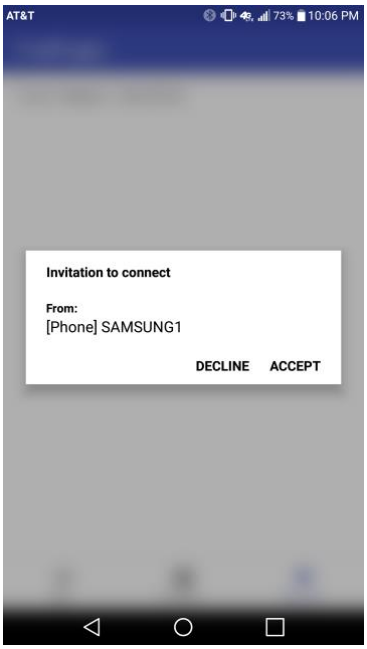

Fig. 5. Group Owner confirmation

F. Transferring of Applications

Once the connection is established by the steps above, the group owner and client are able to exchange IP addresses programmatically and establish a TCP Socket Connection, as described in class, to exchange the Application List asynchronously [20]. This was critical in our problem as we wanted the ability to connect without the need for configuration files or input parameters. The Socket Connection is a critical component because this is how the data is being sent around the system via the group owner.

G. Failures

One of the major drawbacks of Wi-Fi P2P is how it does not natively handle any failovers and at it's core is setup as a centralized network with a very obvious single point of failure in the group owner. This is similar to router devices used to broadcast internet within a house, if that router is disconnected or fails, the internet network is destroyed. This analog is exactly what happens when you stop or interrupt the group owner when it has client's connected. A future project would be to design how we could if failover gracefully to a new group owner and dynamically reconnect the clients, if that happens.

V. Conclusion

Overall, the Wi-Fi P2P Group is a quick way to establish an ad-hoc connection with several devices to exchange data quickly. However, we ran into several roadblocks in our journey, like the compatibility issues of connecting LG devices and Samsung devices and how to safely failover the system if a failure happens. The application built and described above is a simple way to prove that a distributed network can be achieved

without the need for setup files or inputs. Additionally, it shows that the application although basic could be utilized for a production application that allows groups of friends to compare like applications.

REFERENCES

[1] Android (operating system), https://en.wikipedia.org/wiki/Android_(operating_system)

[2] O. C. Novac, M. Novac, C. Gordan, T. Berczes and G. Bujdosó, "Comparative study of Google Android, Apple iOS and Microsoft Windows Phone mobile operating systems," *2017 14th International Conference on Engineering of Modern Electric Systems (EMES)*, Oradea, 2017, pp. 154-159.

[3] Android, the world's most popular mobile platform, https://developer.android.com/about/index.html

[4] Dashboards, https://developer.android.com/about/dashboards/index.html

[5] D. Camps-Mur, A. Garcia-Saavedra and P. Serrano, "Device-to-device communications with Wi-Fi Direct: overview and experimentation," in *IEEE Wireless Communications*, vol. 20, no. 3, pp. 96-104, June 2013.

[6] U. Demir, C. Tapparello and W. Heinzelman, "Maintaining Connectivity in Ad Hoc Networks Through WiFi Direct," *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Orlando, FL, USA, 2017, pp. 308-312.

[7] Device Compatibility, https://developer.android.com/guide/practices/compatibility.html

[8] Gartner Says Chinese Smartphone Vendors Were Only Vendors in the Global Top Five to Increase Sales in the Third Quarter of 2016, https://www.gartner.com/newsroom/id/3516317

[9] Is Android an Open Source Operating System, or Not?, http://www.droidviews.com/android-an-open-source-operating-system-or-not/

[10] Installed base of smartphones by operating system from 2015 to 2017 (in million units), https://www.statista.com/statistics/385001/smartphone-worldwide-installed-base-operating-systems/

[11] Run Apps on the Android Emulator, https://developer.android.com/studio/run/emulator.html

[12] Connectivity, https://developer.android.com/guide/topics/connectivity/index.html

[13] Bluetooth, https://developer.android.com/guide/topics/connectivity/bluetooth.html

[14] Wi-Fi Aware, https://developer.android.com/guide/topics/connectivity/wifi-aware.html

[15] Wi-Fi Peer-to-Peer, https://developer.android.com/guide/topics/connectivity/wifip2p.html

[16] Near Field Communication, https://developer.android.com/guide/topics/connectivity/nfc/index.html

[17] Piconet, https://en.wikipedia.org/wiki/Piconet

[18] Bin Da, Haihua Yu, Wei Wang, Linju Yang and Ke Liao, "Area Restricted Ad-hoc Network system based on Near Field Communication technology," *2015 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, Bandung, 2015, pp. 120-124.

[19] Creating P2P Connections with Wi-Fi, https://developer.android.com/training/connect-devices-wirelessly/wifi-direct.html

[20] Garg, Vijay K. *Elements of Distributed Computing*. Wiley-Interscience, 2006.