Communications and Network, 2021, 13, 143-158
https://www.scirp.org/journal/cn
ISSN Online: 1947-3826
ISSN Print: 1949-2421

A Framework for Multi-Hop Ad-Hoc Networking over Wi-Fi Direct with Android Smart Devices

Rémy Maxime Mbala[1,2*], Jean Michel Nlong[1,2], Jean-Robert Kala Kamdjoug[3]

[1]Department of Mathematics and Computer Science, Faculty of Science, The University of Ngaoundéré, Ngaoundéré, Cameroon
[2]Laboratory of Mathematics, Computer Science and Applications, Faculty of Science, The University of Ngaoundéré, Ngaoundéré, Cameroon
[3]GRIAGES, Catholic University of Central Africa, Yaoundé, Cameroon
Email: *rmbala@univ-ndere.cm

How to cite this paper: Mbala, R.M., Nlong, J.M. and Kamdjoug, J.-R.K. (2021) A Framework for Multi-Hop Ad-Hoc Networking over Wi-Fi Direct with Android Smart Devices. *Communications and Network*, **13**, 143-158.
https://doi.org/10.4236/cn.2021.134011

Received: July 14, 2021
Accepted: November 27, 2021
Published: November 30, 2021

Abstract

The wide diffusion of mobile devices that natively support ad hoc communication technologies has led to several protocols for enabling and optimizing Mobile Ad Hoc Networks (MANETs). Nevertheless, the actual utilization of MANETs in real life seems limited due to the lack of protocols for the automatic creation and evolution of ad hoc networks. Recently, a novel P2P protocol named Wi-Fi Direct has been proposed and standardized by the Wi-Fi Alliance to facilitate nearby devices' interconnection. Wi-Fi Direct provides high-performance direct communication among devices, includes different energy management mechanisms, and is now available in most Android mobile devices. However, the current implementation of Wi-Fi Direct on Android has several limitations, making the Wi-Fi Direct network only be a one-hop ad-hoc network. This paper aims to develop a new framework for multi-hop ad hoc networking using Wi-Fi Direct in Android smart devices. The framework includes a connection establishment protocol and a group management protocol. Simulations validate the proposed framework on the OMNeT++ simulator. We analyzed the framework by varying transmission range, number of hops, and buffer size. The results indicate that the framework provides an eventual 100% packet delivery for different transmission ranges and hop count values. The buffer size has enough space for all packets. However, as buffer size decreases, the packet delivery decreases proportionally.

Keywords

Wi-Fi Direct, Android, Smart Devices, Mobile Ad Hoc Network, Framework, Connection Protocol, Multi-Hop, Service Discovery

1. Introduction

Facilitated by the high popularity of portable mobile devices, MANET is considered a promising emergency communication solution during catastrophic natural disasters when existing communication systems are compromised [1]. Recently, MANET argues to have the new potentiality to play a crucial role in LTE offloading systems. Recent studies reveal that LTE is looking at the interoperability with other off-grid communication technologies by introducing the concept of network-assisted D2D communication [2] [3] [4]. In the vein, the cellular interface would jump-start the D2D link between suitable devices by handling the discovery and authentication phases, thus serving as broker party [5] [6].

Generally, mobile devices have three tools to make off-grid communications: Blue-tooth, Wi-Fi Ad-hoc mode, and Wi-Fi Direct, but each of them has its limitations. Bluetooth has a slow data transfer rate at about 3 Mbps which is significantly lower than Wi-Fi-based wireless communications with a speed of 54 Mbps or higher. Another limitation of Bluetooth regards the communication distance. The range of Blue-tooth is about 10 meters, and as a comparison, Wi-Fi supports a range of up to 46 meters indoor and 92 meters outdoors.

Wi-Fi Ad-Hoc mode inherits basic Wi-Fi features. The Smart Phone Ad-hoc Net-work (SPAN) project designed by Homeland Security System Engineering and Development Institute (HS SEDI) uses MANET over Wi-Fi Ad-hoc mode to target natural disasters or terrorist incidents. Here, the existing network infrastructure is overloaded, destroyed, or compromised [7]. Even though MANET's successful implementation over Wi-Fi Ad-hoc mode, the disadvantage of Wi-Fi Ad-hoc mode is inevitable. First of all, only selected mobile devices in the market have Wi-Fi Ad-hoc mode support. The device support limitation could decrease the peer-to-peer discovery probability and thus restrict the MANET popularity. Furthermore, Wi-Fi Ad-hoc mode is yet not user-friendly. It often requires rooting a device and making modifications in the operating system, kernel or drivers. This rooting process is complex for most smartphone users [8].

Wi-Fi Direct is also an extension of Wi-Fi but more user-friendly than Wi-Fi Ad-hoc mode. The Wi-Fi Alliance has now certified more than 550 products for Wi-Fi Direct [8]. Android provides an open-source API for developers to access Wi-Fi Direct on non-rooted smartphones. However, Wi-Fi Direct is restrained to its Group Owner-Client topology so that the expansion from a static device-to-device network to a dynamic MANET is complex. Currently, the technical literature proposes some solutions to overcome this constraint. [9] introduced a multi-group, interconnected logical topology that overcomes P2P inter-group data transfer limitations by exploiting transport-layer tunneling. However, the idea is behind many assumptions, and the network topology is not dynamic. Some devices are assigned particular roles to maintain the network backbone's stability, which violates dynamic MANET where the network backbone node is not a requirement. [8] introduced a method to achieve multi-hop communication among open-source, non-rooted An-droid devices using Wi-Fi Di-

rect technology. In this solution, all devices become Group-Owner when there is no data transmission; if a device is trying to initiate data transmission, it must first remove its GO status and connect to the target device as a Wi-Fi P2P client. In this way, with the complete connection made, sending data is now possible, and after the target device had received data, it disconnects from the group and removes the client status. It becomes a group owner again and gets ready for the next transmission cycle. However, this network topology has two critical challenges: the unstable peer discovery caused by the switching of GO and client role in the P2P network and the possible communication collision while flooding the proactive routing message.

In this article, we focus on overcoming Wi-Fi Direct network restriction and introducing a new framework for MANET using Wi-Fi Direct on Android devices. Our contribution is manifold:

- We first illustrate how mobile devices can connect and form a Wi-Fi P2P network based on Wi-Fi Direct, including detailed information exchanging, group establish requirements and P2P group limitations.
- Secondly, we present our lightweight framework for ad hoc networking of Android smart devices over Wi-Fi Direct. We explain how to form a novel dynamic multi-hop ad hoc network using our framework that overcomes the traditional Wi-Fi Direct limitations.
- Thirdly, we use a proactive routing in this network, allowing us to enable bi-directional and dynamic data transfer to any device in the network, which is impossible in regular Wi-Fi Direct groups.
- Finally, we demonstrate this multi-hop network and show that it can successfully transfer data through this network.

The rest of the paper contains four sections. Section 2 provides an overview of Wi-Fi Direct, including the limitations of its API on Android OS. Section 3 introduces the framework for MANET network using Wi-Fi Direct to overcome Wi-Fi P2P shortcomings. In this section, we also analyze Routing. Section 4 presents the implementation and validation of the proposed framework. Finally, section 5 concludes the contribution.

2. Overview of Wi-Fi Direct

2.1. Group Formation

Wi-Fi Direct [10] (sometimes called Wi-Fi P2P) is one of the most promising peer-to-peer technologies for smart devices, which allows data exchange over long ranges with speeds higher than Bluetooth. It is geared to support the same speed and range of that of Wi-Fi infrastructure mode. It also enables forming groups or data exchange without the need for intermediate access point. Typically, one of the Wi-Fi Direct capable devices acts as a software access point to the rest of the devices in the group; this device is called the group owner (GO). The other devices associate with the group owner and become group members (GM) also called Clients. The selection of a group owner depends on the type of

the group formation. A group can be formed using one of the three different modes: standard, autonomous, and persistent [2]. In the standard mode, the devices involved in setting up the group negotiate which one becomes the GO. To do the negotiation process, each device states its desire to become a group owner by broadcasting an integer value called the GO intent. This value ranges from 15 to 0 where a high value reflects increased interested in serving as GO. The device with the highest intent value wins the negotiation and becomes the group owner. In case of a tie, a tiebreaker bit is used. In the autonomous mode, a device creates a group and declares itself as a GO. Other devices can connect to this group as Clients. For the last case, the persistent mode, the devices save the GO of the previous group for future usage. Once the same devices start a group again, the previous GO takes ownership of the group. Wi-Fi Direct is suitable for exchanging data in areas with no cellular towers or access points. In recent years, Wi-Fi Direct has become available on most new smart devices.

2.2. Wi-Fi Direct Limitations on Android API

Android is one of the most popular and widely spread operating systems for portable smart devices. Most Android phones with Ice Cream Sandwich (API Level 14) or higher versions are capable of Wi-Fi Direct communication. However, the APIs for Wi-Fi Direct only provide basic support for connecting multiple peers in one P2P group. With the current Android APIs, a peer-to-peer system in the sense that every device can communicate with others is not possible. Each device can only belong to one group and if it wants to transmit message to another device that is not in the group, it has to either join the same group the target device belongs or it has to establish a new group with the target device. This limitation makes the Wi-Fi Direct network could only be a one-hop ad-hoc network with the GO being the gateway device. Another problem is that actually, there are no library routines for informing every device in the group about the IP addresses of the other group members. Thus, a method of distributing IP addresses is required to allow the devices to operate in a peer-to-peer mode. Although the Android APIs makes the IP address of the GO easily accessible to every device in the group, no API exists that allow a device to obtain the API address of its own Wi-Fi Direct interface to share with peers. Thus, a way of finding local addresses is also required.

Despite the limitations, Android still have good Wi-Fi Direct APIs that can be exploited by a group management protocol to form a peer-to-peer system. Moreover, Android has support for Wi-Fi Direct service discovery (since API Level 16), which allows a device to announce the services it offers and discover the services provided by others, even before attempting to form a group. In fact, nearby devices could have different services. Thus, for a correct peer-to-peer implementation, there is a need to connect only the peers with the same set of services together. Using Wi-Fi Direct service discovery protocol, it is possible to define a set of service classes and to limit the device enrollment in a group based

on specific class of service that the group members offer.

3. Wi-Fi Direct Ad Hoc Networking Framework

To enable Ad hoc networking over Wi-Fi Direct in Android we propose a connection establishment protocol and a group management protocol. The proposed protocols are described in detail in the balance of this section.

3.1. Connection Establishment Protocol

When many devices are in the range of each other, many group announcements are made and a device becomes overwhelmed. The goal of our connection establishment protocol is to allow the devices to define their supported service type and to filter nearby devices based on such service type. Thus, this protocol takes care of allowing only the devices that provide the same service to connect together and increase the efficiency of group management. Basically, before attempting to form any Wi-Fi Direct group or connecting to an existing group, a device has to announce its supported services. The Android APIs for Wi-Fi Direct service discovery has an option to include a service record along with the service type that the device can provide. The service record is used for exchanging additional data, which is used in the connection establishment.

Our new group formation procedure consists of eliminating the 3-way handshake of the Group Owner Negotiation (GON) procedure. It allows to elect the Group Owner based on other parameters in addition to the Group Owner Intent. It also gives the Group Owner, once elected, the opportunity to belong to another group as Client and to interconnect several groups.

In this procedure, we include all the information required for group formation in the packets defined and supported by Wi-Fi Direct: The Probe Request (PbReq) and the Probe Response (PbRes). Our method consists in calculating a Score function and inserting it, as well as the list of discovered equipment (with their Scores), in the P2P Information Element (IE) [10] available in PbReq and PbRes.

When a device receives a PbReq or a PbRes from another device, it can simply determine which is more capable of being a Group Owner. The equipment with the highest Score can start an autonomous group independently, thus becoming the Group Owner of the new group created and invite the other equipment discovered and present in its list to join its group as a Client. The Group Owner equipment in a group can also be invited to connect as a Customer in another group and thus allow the interconnection of the two groups. **Figure 1** shows the flowchart of our algorithm for the group formation.

This method offers the possibility for P2P equipment to have an idea of the capacities of discovered neighbors and to interconnect P2P groups two by two. **Figure 2** shows the initial network topology after the algorithm has been executed. In this **Figure 2**, the devices which receive the arrows are the Clients of the groups and those which initiate the arrows are Group Owners. For example,

device C is Group Owner in Group 1 and Client in Group 2, whose Group Owner device is H.

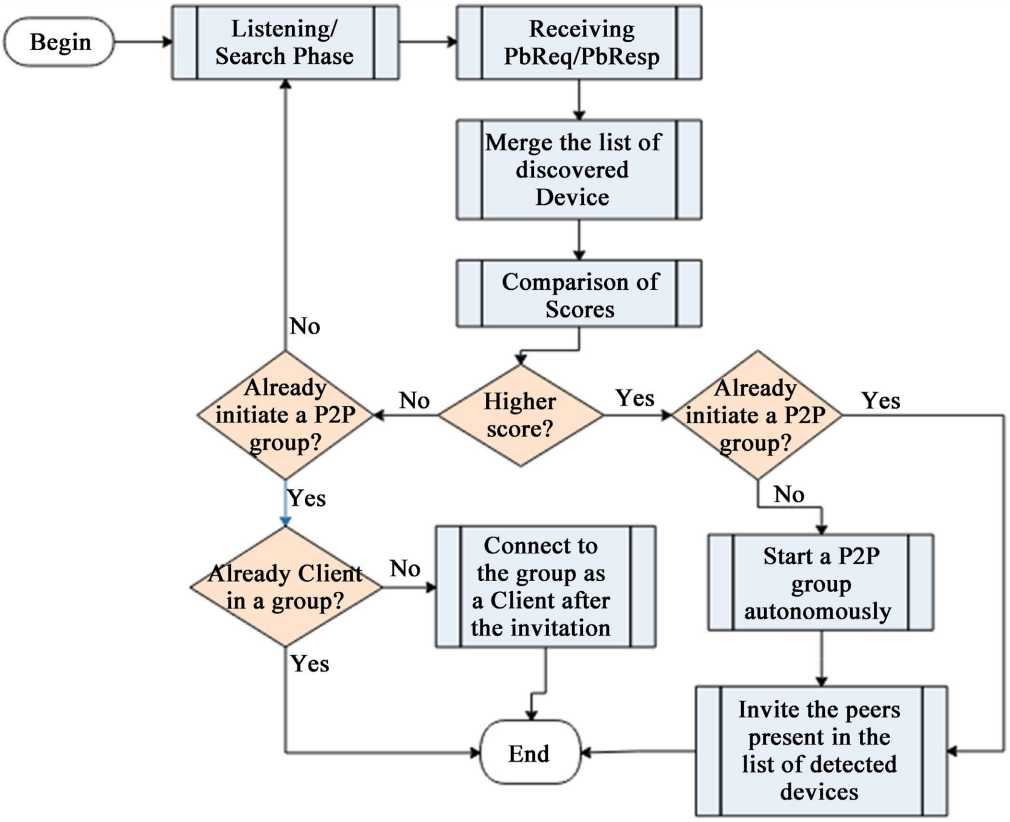

Figure 1. Flowchart of connection establishment.

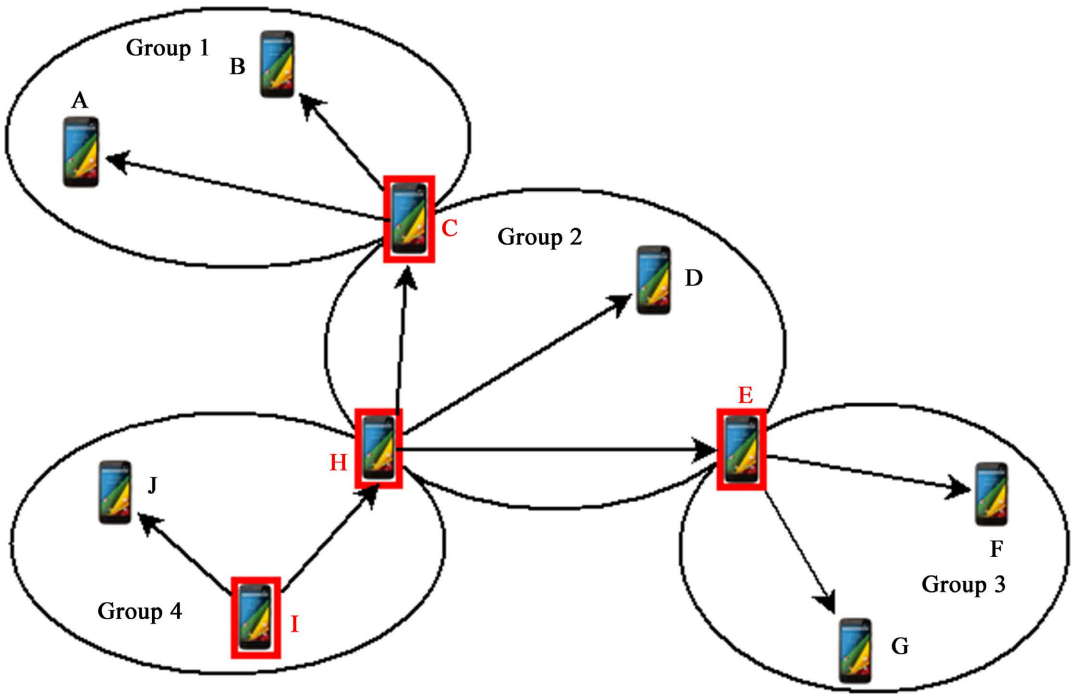

Figure 2. Initial network topology.

By eliminating the Group Owner Negotiation (GON) stage, we improve group formation time. Our method offers the possibility of choosing the best Group Owner compared to the capacities of neighboring devices discovered, which is not possible with the current API of Wi-Fi Direct. Another advantage of our method is that it allows the Group Owner to build the list of its neighbors with their corresponding Score, which is very useful for group management and topology control.

Calculation of the Score function

The Score function is calculated by the devices before starting to build the network. It is evaluated by each device and depends on several parameters:

- ***Battery information* (*B*):** by adopting a similar method as in [11], where the information retained is the state, the level of charge and the capacity of the battery. Equipment with a high charge level and capacity would make a good candidate to be a Group Owner.

$$B = a_1 \times E + a_2 \times \frac{L}{1000} + a_3 \times \frac{C}{4000} \quad [11] \tag{1}$$

where E defines the state of the battery: 0 or 1 (when the percentage of charge is still sufficient), L the charge level being in the range [1, 100] and C the battery capacity. Capacity is divided by 4000 mAh, reflecting an average capacity of a number of commercially available devices. These battery information's can be obtained from the Android API using the *batteryManager* package. The constants a_1, a_2 and a_3 being fixed at 0.34, 0.33 and 0.33 respectively [11].

- ***Degree* (*D*):** which defines the difference between the number of devices actually discovered (d), present in the list of devices discovered and the number of devices actually supported (M).

$$D = |d - M| \tag{2}$$

The number of devices supported is given by the Wi-Fi hotspot (Tethering).

- ***Intent* (*I*):** the Group Owner intent used by Wi-Fi Direct.

So, the final expression of the *Score* function is given by:

$$Score = c_1 \times B + c_2 \times \frac{D}{8} + c_3 \times \frac{I}{10} \text{ with } \sum_{i=1}^{3} c_i = 1 \tag{3}$$

In our case, the weighting factors c_1, c_2 and c_3 are also set to 0.34, 0.33 and 0.33 respectively.

3.2. Group Management Protocol

Once the connections are established, based on service matches, and a group is formed, the group management protocol forms peer-to-peer links among the group members at the level of the transport layer. The following explains the operation of the group management protocol.

Socket connection

The group management protocol uses two layers of socket connections one for management purposes and the other for data exchange purposes.

- ***Management sockets***: As the name indicates, the management sockets are for managing the group. The GO uses dedicated server sockets for receiving peers' information and sending the list of peers to all group members (GM). Upon creating the Wi-Fi Direct group, the GO opens a server socket and binds it to a predefined management port. Then every GM tries to connect to such a socket in the GO. On every successful socket connection, the GO adds the connected socket to a list of opened management sockets and opens another instance of the server socket. The final topology of the connections at the management level is a star topology that originates at the GO as shown in Figure 2.
- ***Data exchange sockets***: in addition to previous socket connections, every device in the group including the GO opens another set of server sockets for data exchanges purposes and binds them to a predefined data exchange port. Afterwards, each device waits for other peers in the group to connect to the opened socket (given that all devices will eventually know the IP address of each other as discussed later). When a connection from a peer is accepted, a list of all opened data sockets is updated to reflect the new connection. The final topology of the connections at the data exchange level is a mesh topology as shown in Figure 3.

Group Owner as the master for management

The group management protocol is centralized and performed by the GO. The GO receives heartbeat messages from GMs, updates its local peers' list as necessary, and sends the current list of peers to the other devices. The following are the detailed operations:

- ***Heartbeat messages***: the heartbeat messages are used to announce that a GM is connected or still alive. Each GM sends a heartbeat message every α second

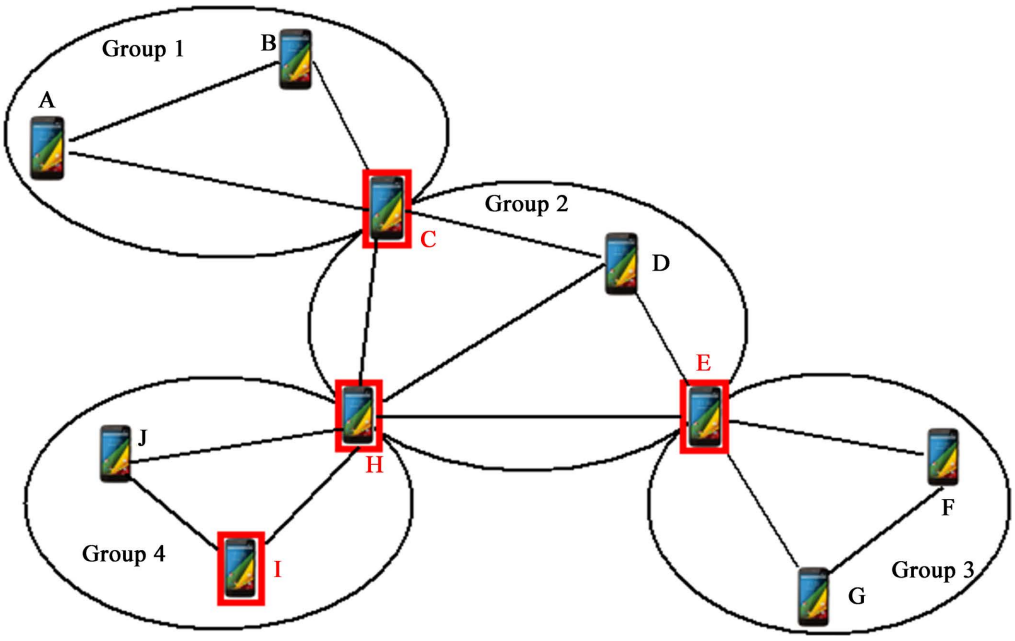

Figure 3. The topology for data exchange sockets connection.

to the GO through the opened management sockets. The message is a comma separated string composed of the concatenation of the BSSID (Basic Service Set IDentifier) of the group, device name/username, MAC address, and IP address. Upon receiving the heartbeat message, the GO stores the peer information in the current peers' list. If the GM is already known, the GO only updates the stored values.

- ***Current Peers list***: The GO notifies its GMs about peers by sending a message every β seconds that contains the list of all known peers, where β is a multiple of α to allow the GO to collect the data of more than one new GM before sending the peers list message. The list is just a semicolon-concatenated string that is composed of the heartbeat messages received from the GMs plus the information of the GO itself. Upon receiving the list from the GO, each GM stores it in (or updates) its peers list data structure. It then attempts to open client socket connections for data exchange with all peers in the list (including the GO), using the IP addresses given in the list. If a peer is already connected, no need to connect it again. A list of all socket connections in the device is updated accordingly in order to reflect how many data exchange sockets are open.
- ***Duplicated socket connections removal***: as the GMs attempts to connect to other peers, as soon as the list from the GO is received, there is a possibility that duplicated data exchange sockets are opened between two peers. To prevent this, we added a random wait time before a peer attempts connecting to another. Thus, before connecting to a peer, a GM checks if a socket connection with that peer already exits. While this decreases the possibility of duplicate connections, there is still a possibility for this problem to happen. To eliminate this problem completely, we have added a new procedure that allows removing any redundant connections. This procedure runs after a peer opens connections with all other peers. Where, each peer iterates along all sockets in the list of the data exchange sockets, finds any duplicated socket connection (using the IP address associated with the socket), and removes it. To avoid removing sockets from both ends, we use the last octet value in the IP address of the device to force only one socket connection removal. Each device when iterating through the list, it removes the duplicated socket connection only if the last octet in its own IP address is higher than the last octet of the IP address of the peer associated with the socket. After running this procedure, each device will keep only one data socket connection with other peers.
- ***Pruning peers***: to tell whether a peer is alive or not, a time-to-live (TTL) value is associated with each peer in the stored peer list. The TTL value is initialized to γ (where γ is a multiple of β to allow the GMs to perform pruning and addition of peers at the same step). This value is decreased every top of time by all devices if the peer information is not heard again. If the GO receives the heartbeat message from the peer, it resets the TTL value for that peer

to γ. If the GM sees the peer again in the list transmitted by the GO, it resets the value to γ again. Once the GO determines that a certain peers' TTL value has reached zero, it assumes that the peer has departed the group. The GO then disconnects any data and management sockets opened previously with that peer. The peer is also removed from data structure of the list of peers. In the next time the GO transmits a peers' list message to its members, the removed peer will not be there. A GM continues to decrease the TTL value for a removed peer until reached to zero. In that case the GM disconnects from any data or management sockets associated with that peer, and removes that peer completely from its peers' list data structure.

- ***Restarting after GO failure***: if one of the GM fails or disconnects from the group, the GM will take the required action to tell other members that this peer is not available any more. However, in case of GO failure, the devices would not hear normal peers' list message. They will start to decrease the TTL of the GO in their peer data structure. Once the TTL value for the GO reached zero, they all disconnect from the GO. In this case, they detect the removal of GO and they flush any peer's data structure and start over again.

A flowchart that shows the complete steps for the proposed group management protocol for both GO and GM is shown in Figure 4.

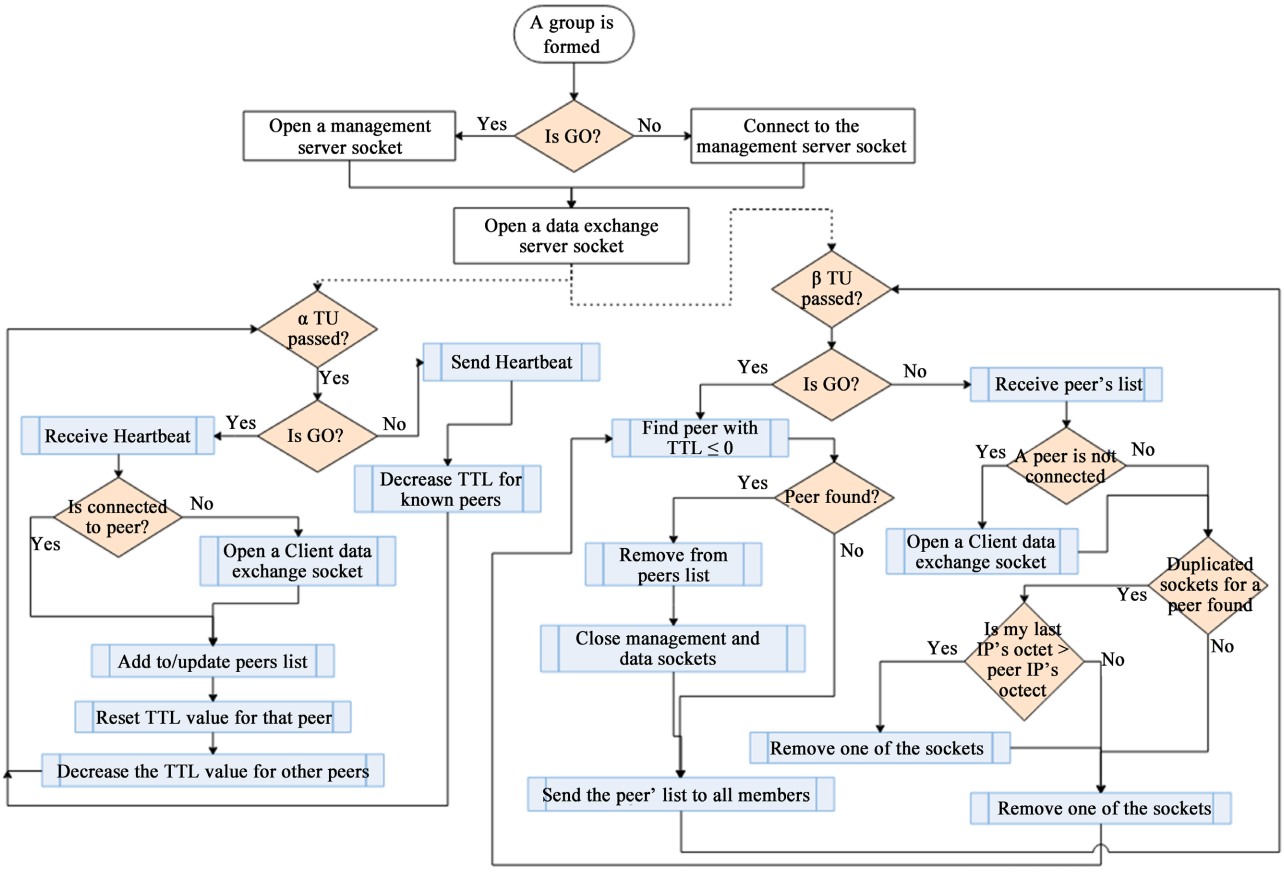

Figure 4. The flowchart for the group management protocol.

4. Implementation and Validation

To validate the proposed framework, we run several simulations using OMNeT++ simulator and verify its basic functionalities. [12] has implemented the main procedures of Wi-Fi Direct such as discovery, negotiation and group formation in the INET framework of OMNeT++. We modified this first implementation by adding the features of our framework. We set-up several scenarios, to evaluate packet delivery as a function of time delay while varying transmission range, hop count, and buffer size.

4.1. Simulation Setup

We perform the simulations over an area of 1500 × 300 m^2. All the simulations are averaged over 100 runs with each simulation running for 200 s. Simulations are performed with 50 nodes. We fixed α, β and γ to 1 ms, 5 ms and 30 ms respectively. We perform simulations with a packet size of 1024 bytes. We use the OMNeT++ IPTrafGen application to generate CBR (Constant Bit Rate) traffic. The 802.11 g MAC is the link layer over the range propagation loss model to limit the transmission ranges of nodes. The transmission range of the nodes is set as 100 m for evaluation. The mobility model used is steady-state random waypoint with random velocities from 0.01 - 20.0 m/s. A summary of all simulation parameters is shown in Table 1.

4.2. Simulation Analysis

We considered three scenarios for evaluating the performance of our framework. In the first scenario, the transmission range is varied from 10 m to 250 m. In the second scenario, the number of hops is varied from 1 hop to 8 hops. In the third scenario, the buffer length is varied from 10 packets to 2000 packets. In all scenarios, the percentage of packets delivered is plotted as a function of packet delivery latency.

Varying transmission range

In this scenario, the transmission range is varied with five different values:

Table 1. Simulation parameters.

Parameters	*Value*
Simulation area	1500 m × 300 m
Number of runs	100
Total simulation time	200 s
Mobility model	Random waypoint
Node speed	0 - 20 m/s
Packet size	1024 bytes
Number of packets	1980 packets/simulation
Link layer	Wi-Fi g 54 Mbps
Propagation loss model	Range

{10, 25, 50, 100, 250} meters. With all of these transmission range values, we get eventually, 100% packet delivery. However, as transmission range increases, the packet delivery latency increases. This is because when the transmission range is high (e.g., 250 m), the connectivity becomes highly intermittent, which causes the network to have a higher number of disconnected components consisting of a small number of nodes. On the other hand, as the transmission range decreases, connections are fast enough and less disturbed. As a result, the packet delivery latency is much shorter for 10 m than 250 m, as shown in Figure 5.

Varying Hop count

In this scenario, the hop count is varied with five different values: {1, 2, 3, 4, 8} hops. Similar to the varying transmission range, packet delivery is 100% for all hop count values. However, as hop count increases, the packet delivery latency increases as shown in Figure 6. This is because packets with a high hop count take longer time in the network to reach the destination and can be dropped faster than packets with a lower hop count, especially if the network is congested. For example, packets with a count value of 8 hops are not delivered until the nodes that can connect the source and receiver nodes are not within direct communication range of each other, which takes longer time because the nodes move randomly. However, with a count value of 1 hop, the packets can be transmitted directly which will take less time.

Varying Buffer length

In this scenario, the buffer length is varied with eight different values: {10, 20, 50, 100, 200, 500, 1000, 2000} packets. The results are shown in Figure 7. In the scenario with 2000 buffer length, the packet delivery is 100%. This is because there are 1980 packets in each simulation, the length 2000 emulates an infinite buffer. However, for the other values of buffer lengths, the packet delivery

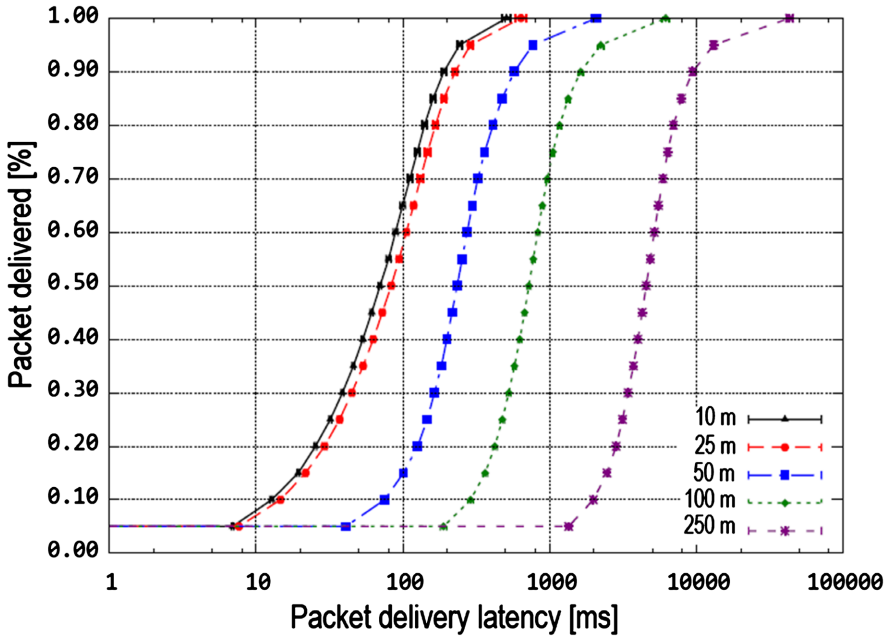

Figure 5. CDF for varying transmission range.

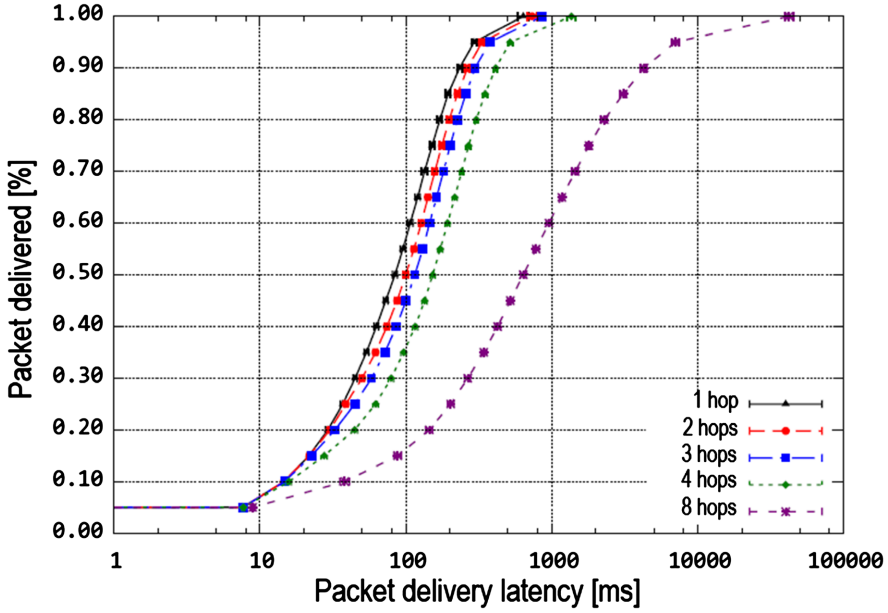

Figure 6. CDF for varying hop count.

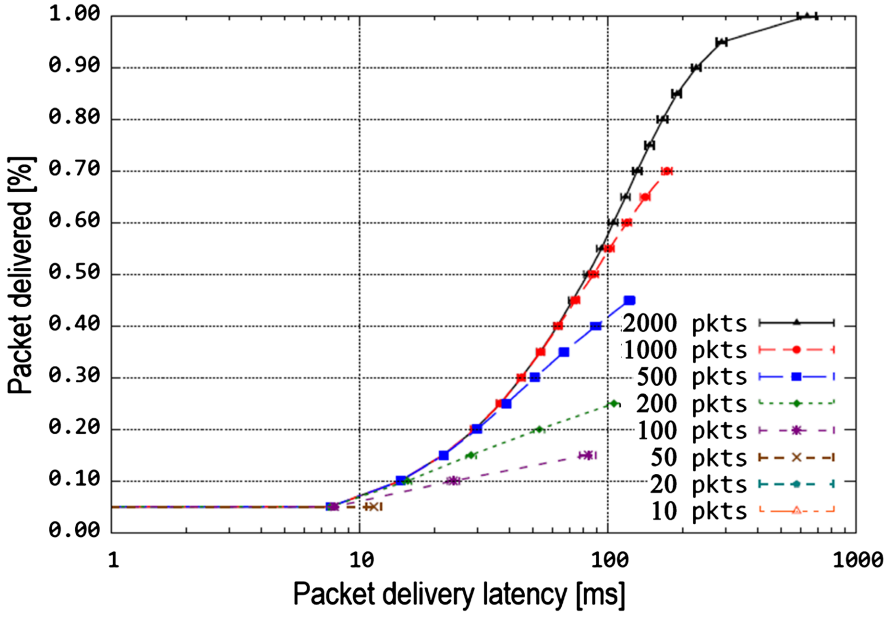

Figure 7. CDF for varying buffer size.

decreases proportionally with buffer length. This is because packets are dropped when node buffers reach their capacity. This shows that the buffer capacity is a critical parameter and it should be set large enough to contain all packet expected in the network to obtain 100% packet delivery in this scenario.

4.3. Implementation on Real Devices

To validate our proposed ad hoc networking framework, we have also implemented the connection establishment and group management protocols by developing an Android P2P application for file transferring. The implementation

of our framework allowed only the devices that run this application to connect to each other, multiple peers to exchange file together, and the seamless handling of addition or removal of peers. From a user perspective, any file transferred by a device is sent to all other devices in the network. This application is written in AndroidStudio using Android SDK and is made available at the Google Play Store [13]. The application woks with Android API level 16 in order to be able to run the service discovery mechanism used in the connection setup protocol. The application is composed of the following components NearByPeers class, Main Activity, SocketManager Class, File Manager Fragment, Multi-hop Manager Class and Settings Activity. Table 2 shows a brief description for what each component does.

This application is tested on four Android devices (two Samsung Galaxy Tab 2 7.0 tablets and two Nexus 4 phones). In this test, we fixed α, β, and γ to 1, 5, and

Table 2. Application components and their description.

Components	*Description*
NearByPeers	• Attributes: Management Sockets List, Data Sockets List, and Peer Record • Methods: Add new peer, connect to peer, prune peers, and remove duplicated sockets
Main Activity	• Performs the service discovery announcement. • Finds nearby devices and adds them to a list. • Connects to a device, opens/connects to the required management sockets, and opens/connects to data exchange sockets. • Periodically sends peer info (GM) every millisecond/ peers list (GO) every 5 milliseconds. • Periodically decrease the TTL values for peers every millisecond. • Periodically prune peers every 5 milliseconds. • Handling messages from management sockets and add/update peers list, prune peers, remove duplicated socket connections, and reset TTL values accordingly. • Opens a data exchange connection if not already connected.
Socket Manager Class	• Handles incoming data from the socket • Notifies the main activity about the data and its type (Management/data) • Handles outcoming data to the socket
File Manager Fragment	• Registers incoming or outgoing Application Message intents • Checks where exactly the file is located in the device and returns an appropriate response • Sends a file request to peers
Multi-hop Manager	• Stores and forwards messages • Have an intent register which registers the following intents: new Connection (when we have a new connection where in all the messages are forwarded; Message Received (when we have an incoming message and outgoing Application Message (when we receives an outgoing application message that needs to be forwarded to all peers.
Setting Activity	• Allow the user to change the announced username

30 respectively. We started the application in two devices first and connected them together. Each of the two devices was able to discover the other, and to display its name in the list of discovered devices. The management sockets were created and connected. The heartbeat messages and the peers' list messages flaw as expected. The GM was able to connect to the data sockets of the GO and vice versa, which means that the NearByPeers object was populated with the current peers. Sending and receiving file was going normally. We then started the application in the third device. Afterwards, we performed the connection setup protocol in the new device, which was able to discover the GO of the current group. After that, we connected the new device with the GO. The required socket connections (management and data) between the new device and the GO were successfully opened. The device was successfully added to the list of peers. The GO was receiving two heartbeat messages from the two members connected to it. The old and the new GM were able to open data socket connection to each other. The result was that every device could send a file to the other two. Next, the fourth device was connected successfully and the data exchange proceeded normally. Finally, we disconnected one of the devices from the group to see how the other devices react and observed that they successfully removed the departing device from their NearByPeers object and closed opened sockets for that device.

5. Conclusion

Wi-Fi Direct is one of the best ways to establish MANET among mobile devices because of its long communication range, high data and high popularity. This paper presents a new framework for enabling multi-hop ad hoc networking over Wi-Fi Direct in Android. The main components of the framework are a connection establishment protocol and a group management protocol. The connection establishment protocol allows only devices with the same set of services to connect together and permits GOs to interconnect two different groups. Our group management protocol allows treating the Wi-Fi Direct topology, which is by convention a star network [9], as a mesh network. The protocol does so by providing a mean of distributing peer IP addresses, facilitating transport layer connections and managing addition and removal of peers from the group. The proposed framework is validated by simulations on OMNeT++ simulator. We analyzed the framework by varying transmission range, number of hops, and buffer size. The results indicate that the framework provides an eventual 100% packet delivery for different transmission ranges and hop count values as long as the buffer size has enough space for all packets. However, as buffer size decreases, the packet delivery decreases proportionally. In addition, our group protocol assumes full connectivity, *i.e.*, each member can directly reach every other member on the group.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Lien, Y.-N., Chi, L.-C. and Huang, C.-C. (2010) A Multi-Hop Walkie-Talkie-Like Emergency Communication System for Catastrophic Natural Disasters. 39th *International Conference on Parallel Processing Workshop*, San Diego, CA, 13-16 September 2010, 527-532. https://doi.org/10.1109/ICPPW.2010.77

[2] Camps-Mur, D., Garcia-Saavedra, A. and Serrano, P. (2013) Device-to-Device Communications with Wi-Fi Direct: Overview and Experimentation. *IEEE Wireless Communications*, **20**, 96-104. https://doi.org/10.1109/MWC.2013.6549288

[3] Shen, W.L., Hong, W.S., Cao, X.H., Yin, B., Shila, D.M. and Cheng, Y. (2014) Secure Key Establishment for Device-to-Device Communications. *IEEE Global Communications Conference*, Austin, TX, 8-12 December 2014, 336-340. https://doi.org/10.1109/GLOCOM.2014.7036830

[4] Shen, W.L., Yin, B., Cao, X.H., Cai, L.X. and Cheng, Y. (2019) Secure Device-to-Device Communications over Wi-Fi Direct. *IEEE Network*, **30**, 4-9. https://doi.org/10.1109/MNET.2016.7579020

[5] Asadi, A. and Mancuso, V. (2017) Wi-Fi Direct and LTE D2D in Action. *IFIB Wireless Day* (*WD*), 1-8.

[6] Andreev, S., Pyattaev, A., Johnsson, K., Galinina, O. and Koucheryavy, Y. (2017) Cellular Traffic Offloading onto Network Assisted Device-to-Device Connections. *IEEE Communications Magazine*, **52**, 20-31. https://doi.org/10.1109/MCOM.2014.6807943

[7] Thomas, J., Robble, J. and Modly, N. (2018) Off Grid Communications with Android Meshing the Mobile World. *IEEE Conference on Technologies for Homeland Security* (*HTS*), 401-405.

[8] Liu, K.C., Shen, W.L., Yin, B., Cao, X.H., Cai, L.X. and Cheng, Y. (2019) Development of Mobile Ad-Hoc Network over Wi-Fi Direct with Off-the-Self Android Phones. *IEEE International Conference on Communications* (*IC*), 1-6.

[9] Casetti, C., Chiasserini, C.F., Pelle, L.C., Del Valle, C., Duan, Y.F. and Giaccone, P. (2015) Content-Centric Routing in Wi-Fi Direct Multi-Group Networks. *IEEE* 16th *International Symposium on a World of Wireless, Mobile and Multimedia Networks* (*WoWMoM*), Boston, MA, 14-17 June 2015, 1-9. https://doi.org/10.1109/WoWMoM.2015.7158136

[10] Wi-Fi Alliance. (2016) P2P Technical Group. Wi-Fi Peer-to-Peer (P2P) Technical Specification V1.3, Pages 1-187.

[11] Shahin, A.A. and Younis, M. (2015) Efficient Multi-Group Formation and Communication Protocol for Wi-Fi Direct. *IEEE* 40th *Conference on Local Computer Networks* (*LCN*), Clearwater Beach, FL, 26-29 October 2015, 233-236. https://doi.org/10.1109/LCN.2015.7366314

[12] Iskounen, S., Nguyen, T.M.T. and Monnet, S. (2016) Wi-Fi Direct Simulation for Inet in Omnet++. arXiv preprint arXiv: 1609.04604.

[13] Mbala, R.M. (2021) Wi-Fi Direct Group File Transfer. https://play.google.com/store/apps/details?id=esnetlab.apps.android.wifidirect.discovery