A FRAMEWORK FOR P2P NETWORKING OF SMART DEVICES USING WI-FI DIRECT

Shrikant B. Maid, Prashant B. Pawar, Amit P. Mishra, Gaurav V. Jorvekar

Shrikant B. Maid BE IT,Department of IT, Sanjivanee college of Engineering,Kopargaon, Maharashtra , India
Prashant B. Pawar BE IT, Department of IT, Sanjivanee college of Engineering,Kopargaon, Maharashtra, India
Amit P. Mishra BE IT, Department of IT, Sanjivanee college of Engineering,Kopargaon, Maharashtra, India
Gaurav V. Jorvekar BE IT ,Department of IT, Sanjivanee college of Engineering,Kopargaon, Maharashtra, India

ABSTRACT

The use of smart portable devices has become very popular in the society nowadays. Many of these devices are equipped with sensors that can provide a wealth of information about the surroundings once their readings are aggregated. These capabilities have fueled interest in employing these devices in emerging unconventional applications such as crowd sourcing and vehicular networking, where sensors data is shared for better situational awareness, managing road congestion, coordinating disaster recovery, criminals hunting, etc. Internetworking of collocated devices in these applications enables aggregation of sensor readings and decreasing the demand on the carrier's communication resources, which translate to lower cost, better bandwidth utilization and increased user participation. Wi-Fi Direct is one of the most promising peer-to-peer technologies that can enable such local internetworking of devices. Wi-Fi Direct transceivers are available on almost all new smart devices. However, the software support for Wi-Fi Direct is quite limited. This paper highlights limitations in Android based platforms and presents a framework for supporting formation and management of groups of communicating devices. The proposed framework is validated through the implementation of a chat application over multiple Android based devices.

Keyword : - *Android, Group management, Peer-to-peer networks, Service discovery, Smart devices, Wi-Fi direct.*

1. INTRODUCTION

Smart devices like iPhone and Android powered phones and tablets are spreading very fast. These devices now contain a bunch of sensors ready to be used like accelerometers, gyroscopes, compasses, and GPS. Such sensors open the doors for many applications that can utilize them to better the individual and society. Hazard detection and reporting on the road and crowd-sourcing are sample of these applications [1]. Peer-to-Peer (P2P) data dissemination in such applications can be achieved by many techniques such as Bluetooth and Wi-Fi. Bluetooth is limited in its range, so it is not suitable for long range communications between peers. Wi-Fi, on the other hand, offers two possible solutions to connect multiple peers. The first option is use the Ad-hoc mode that employs the Independent Basic Service Set (IBSS) concept to connect multiple peers without infrastructure. However, the Ad-hoc mode in Wi-Fi does not support long-range communication and does not enable the security features that are available for the infrastructure mode. The other option is to use Wi-Fi Direct.
Wi-Fi Direct [2] (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 the Wi-Fi infrastructure mode. It also enables forming groups for data exchange without the need for intermediate access points. 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 (GMs). 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 [3]. 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 GMs. For the last case, the persistent mode, the devices saves 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. 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. 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. Another problem is that 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 IP address of its own Wi-Fi Direct interface to share with peers. The same applies when a device tries to retrieve its MAC. Thus, a way of finding local addresses is required first before attempting to distribute the list of IPs. 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 a specific class of service that the group members offer.

In this paper, we propose an efficient and lightweight framework for peer-to-peer networking of Android smart devices over Wi-Fi Direct. The framework includes a connection setup protocol and a group management protocol. The former enables different devices with the same set of services to form a Wi-Fi Direct group by utilizing the Wi-Fi Direct service discovery APIs provided by Android. The group management protocol overcomes the limitations in the implementation of Wi-Fi Direct in Android and allows all group members to communicate with each other in a peer-to peer fashion. To achieve this, the GO collects from all connected devices their IP/MAC addresses and then distributes them within the group. We have developed our own functions that allow a device to obtain its IP/MAC addresses before sending them to the GO. Moreover, the proposed framework automatically handles the topology changes by notifying the group members about the addition of any new member to the group and the departure of any of the existing members from the group. Upon receiving these notifications, devices react accordingly and connect to/disconnect from the device they are notified about. The proposed framework is implemented and validated by implementing a group chatting application. The overhead imposed by our protocols is also calculated.

2. LITERATURE SURVEY

Motta and Pasquale [1] were among the first to point out the potential of Wi-Fi Direct in the implementation of mobile P2P systems. They highlighted some applications that can benefit from P2P networking over Wi-Fi Direct such as text messaging, dissemination of traffic information, dissemination of emergency data, photo/video sharing during an event, and last–mile connectivity. The authors also explained how to use a new middleware for P2P networking based on JXTA, which employs distributed hash tables (DHT) to search for peers. Their aim was to implement the proposed middleware in Android once the Wi-Fi Direct API becomes available. However, no progress has been reported in the literature on the implementation of such middleware.

In [4], Thomas et al. discussed the implementation of their Smart Phone Ad-Hoc Networks (SPAN) protocol on Android. The ultimate goal is to allow smart phones to create mesh networks. They have also promoted the concept of off-grid communications, where peers are able to talk to each other without the need for a cellular connection. Since forming an ad-hoc network is not supported by Android, the command line utility “iwconfig” is used to configure the connection. Because not all Android devices support “iwconfig”, the kernel is modified for some of these devices to allow the usage of the command. Nonetheless, such an approach for P2P networking is not suitable for contemporary users, as it requires the device to be at least rooted to use the “iwconfig” command. Rooting and/or modifying the kernel are not trivial jobs for average users to do. This makes this method unsuitable. In addition, the ad-hoc mode in Wi-Fi has range, speed, and security limitations. Our work, in the other hand, does not require any modification to existing system as it uses Wi-Fi Direct, which is supported by the current Android APIs and does not have the limitations that the Wi-Fi ad-hoc mode suffers. Although we had to overcome some limitation in the Android API, no rooting or kernel modification is done. Thus, any application that utilizes our framework could work correctly in stock version of Android.

Conti et al. [5] explored the possibility of creating opportunistic networks over Wi-Fi Direct by studying the latency in forming a group at the link layer. They experimented with different number of nexus devices, ranging from two to six. No group management protocol is proposed. Their work is considered an extension to Camps-Mur et al. [3] who performed real experiments using only two devices. The experiments studied the performance in the standard, autonomous, and persistent modes of group formation. The results showed that attempting to connect to a group is greatly affected by the mode of the group formation. It was concluded that connecting to autonomous group is faster than connecting to a persistent group, mainly because the devices in the persistent group try to quickly connect to the group owner before it is ready, and thus some of request-to-join messages get discarded. This work gave real measurements for on how fast devices could form a group, as well as confirmed the Wi-Fi Direct’s suitability for peer-to-peer systems. They planned to use Wi-Fi Direct in their middleware CAMEO [6], although they did not explain how such integration would be performed.

Peer management for iTrust over Wi-Fi Direct [7] is an attempt to port the iTrust protocol over SMS system to Wi-Fi Direct. iTrust is a peer-to-peer publication, search and retrieval system that enables peers to construct a mobile ad-hoc network for decentralized information sharing. A peer management protocol is proposed for adding new group members. However, no specific details were provided for how a peer departure from a group is handled. In addition, the service discovery feature in Wi-Fi Direct is not exploited to limit membership to only peers with capable services. Furthermore, unlike our proposed framework, their protocol does not specify how to handle topology changes.

Park et al. [8] proposed DirectSpace, a framework for collaboration between devices. The main goal is to provide means for sharing workspaces between users over Wi-Fi Direct. The framework is composed of two services, a connection service, and a collaboration service. The former handles the peer discovery and the connection/disconnection operations. The collaborative service is used for resource sharing and group management. To obtain a list of all peers in a group, the address resolution protocol (ARP) is used to translate MAC addresses into IP addresses. However, it is not clear how the group owner distributes this list to the members in the group. Moreover, ARP Tables are not reliable, as these tables are flushed periodically. Like iTrust, DirectSpace does not benefit from the service discovery API in Wi-Fi Direct.

Other work related to P2P networking using SIP is presented in [9], where content sharing is accomplished by an overlay network. SIP is carried out in a decentralized manner. This approach also suffers from the lack of service discovery.

3. WI-FI DIRECT P2P NETWORKING FRAMEWORK

To enable peer-to-peer 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. First, we define the scope of the P2P that our proposed protocols opt to cover.

1.1 Design Goals and Scope of P2P Support

As discussed earlier, the software support for Wi-Fi Direct in Android needs major expansion to enable P2P networking. In this paper, we opt to make some progress toward that goal. Among the current shortcomings are the lack of accessibility to MAC and IP addresses for the device and the lack of a group management protocol that can handle topology changes and dynamic group membership. To elaborate, in Android, by default, the members of a Wi-Fi Direct group know the IP address of the group owner. This way, each member can open a socket connection with the group owner to exchange data. However, for a true peer-to-peer system, each device in a group should know the IP addresses of all other members in such a group. Thus, a way of distributing the list of peers’ IP addresses to every member is needed. Our proposed framework overcomes such a limitation. The connection establishment protocol allows only the devices with the same set of services to connect together. The group management protocol allows treating the Wi-Fi Direct topology, which is by convention a star network, as a mesh network. The protocol does so by providing a way for distributing the peers' IP addresses, facilitating transport layer connections and managing addition and removal of peers. However, not all shortcomings are addressed in this paper. Specifically, the Android implementation of Wi-Fi Direct does not allow a device to be associated with more than one group at the same time. Wi-Fi Direct specification states that supporting multiple group membership for a device is possible but it is an optional feature. Android omission of this feature limits the extension of the peer-to-peer system beyond the range of the group. Allowing multi-group communications requires substantial modifications to the code base of the Android framework. Such modifications are beyond the scope of this paper and are part of our future work.

1.2 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 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. The proposed protocol states that each device should add a uniqueID and the availability status to the record. A device calculates its uniqueID once, by concatenating two random numbers, and stores it for future connections. The probability of have a completely unique number is very high, as the devices generating these random are not tied to the same clock and also are not running at the same exact instance of time. This uniqueID is used to differentiate between devices in case they have the same name. It is also used by the group management protocol for retrieving peer records. The reason for relying on the uniqueID in this phase is that the MAC address of the device cannot be retrieved until the connection is already established.
An optional username can be included in the service record. When included, this username will be used by peers as a more friendly identifier. In case of omitting the user name field, the device name is to be used. Upon receiving a service discovery announcement from a peer, the device retrieves the information in it. The device then checks for the service type that this peer provides. If both of this device and the connecting peer agree on the same service type, the device continuous processing the information, otherwise it is discarded. The service record is then retrieved and stored. If a username is included the device uses it when displaying the found peer, otherwise the peer's device name is used. The availability status allows the user to know when to attempt the connection with the device. Figure 1 Summaries the connection establishment protocol.

1.3 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 connections: 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 GMs. 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(a).
- *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 exchange 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 2(b).

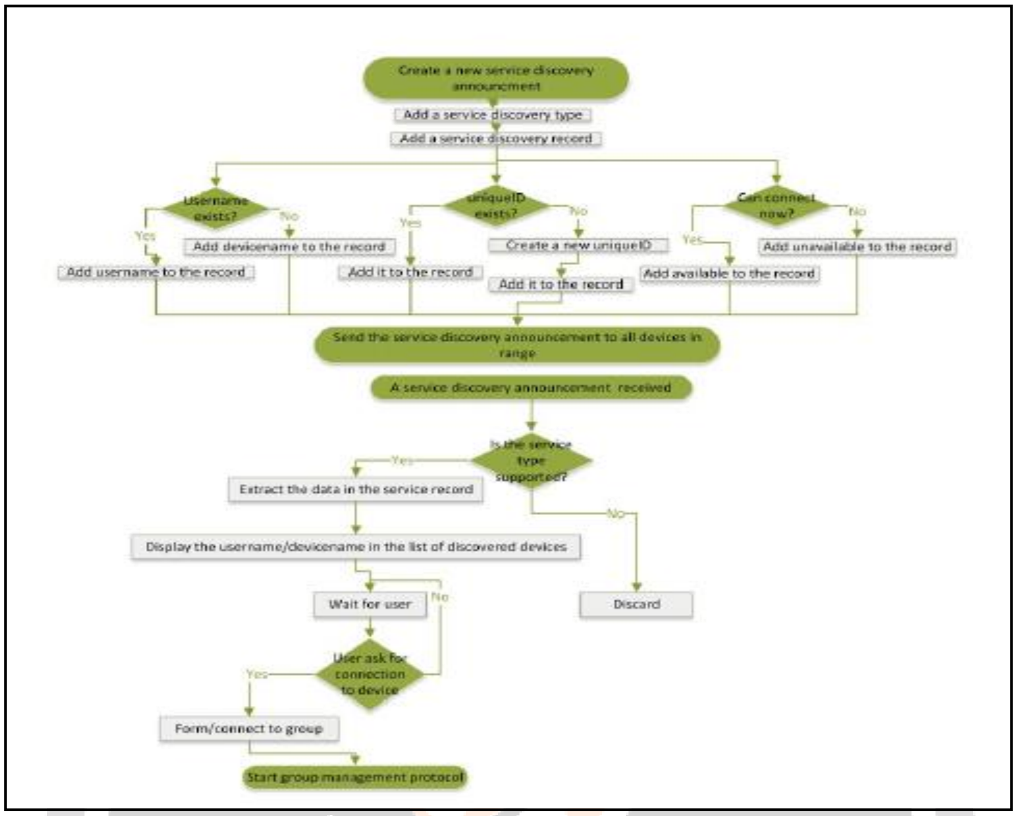

Create a new service discovery announcment
Add a service discovery type
Add a service discovery record
username exists?
Yes
No
Add devicename to the record
Add username to the record
uniqueID exists?
Yes
No
Add it to the record
Create a new uniqueID
Add it to the record
Can connect now?
Yes
No
Add unavailable to the record
Add available to the record
Send the service discovery announcement to all devices in range
A service discovery announcement received
Is the service type supported?
Yes
Extract the data in the service record
No
Display the username/devicename in the list of discovered devices
Wait for user
No
Discard
User ask for connection to device
Yes
Form/connect to group
Start group management protocol

Fig -1: A flowchart for the connection establishment protocol.

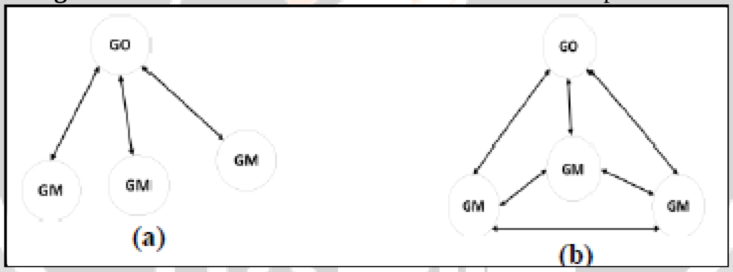


Fig -2: (a) The topology for the management sockets connection; (b) The topology for the data exchange sockets connection.

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 to the GO through the opened management sockets. The message is a comma separated string composed of the concatenation of the uniqueID, devicename/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.

- *Duplicate 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 exists. 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 second by all devices if the peer information is not heard again. If the GO receives a heartbeat message from the peer, it resets the TTL value for that peer to γ. If a 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 peer's 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 the removed peer until it reached zero. In that case the GM disconnects from any data or management sockets associated with that peer, and removes that peers completely from its peer's 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 for the GO in their peers 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 3.

4. IMPLEMENTATION AND VALIDATION

To validate the proposed P2P networking framework, we have implemented the connection establishment and group management protocols by developing an Android P2P application for group chatting. The implementation of our framework allowed only the devices that run this application to connect to each other, multiple peers to chat together, and the seamless handling of addition or removal of peers. From a user perspective, any chat message posted by a device is sent to all other devices in the group. This application is written in AndroidStudio using Android SDK and is made available at the Google Play Store [10]. The application works with Android API level 16 in order to be able to run the service discovery mechanism used in the connection setup protocol (the first part of the framework). The application is composed of the following components NearByPeers Class, Main Activity, SocketManager Class, Chat Fragment, and Settings Activity. Table 1 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 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 chat

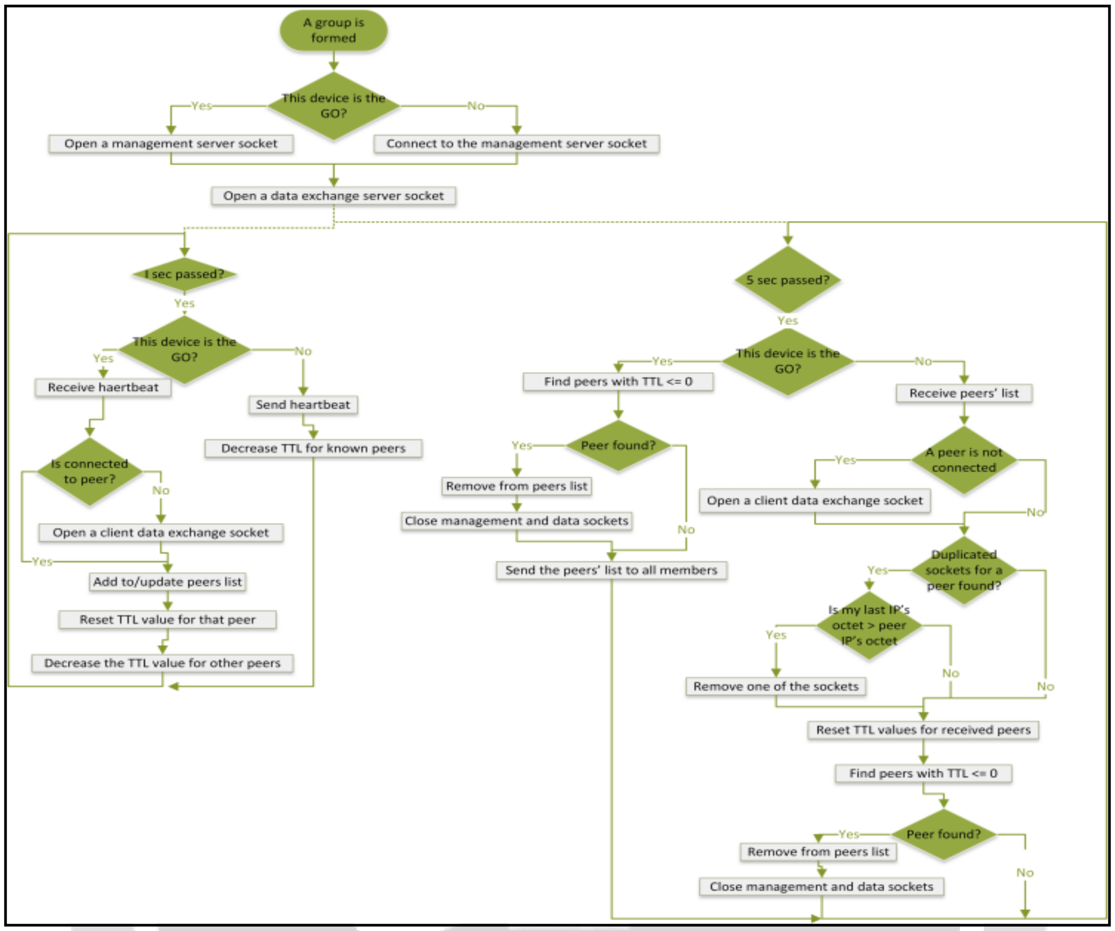
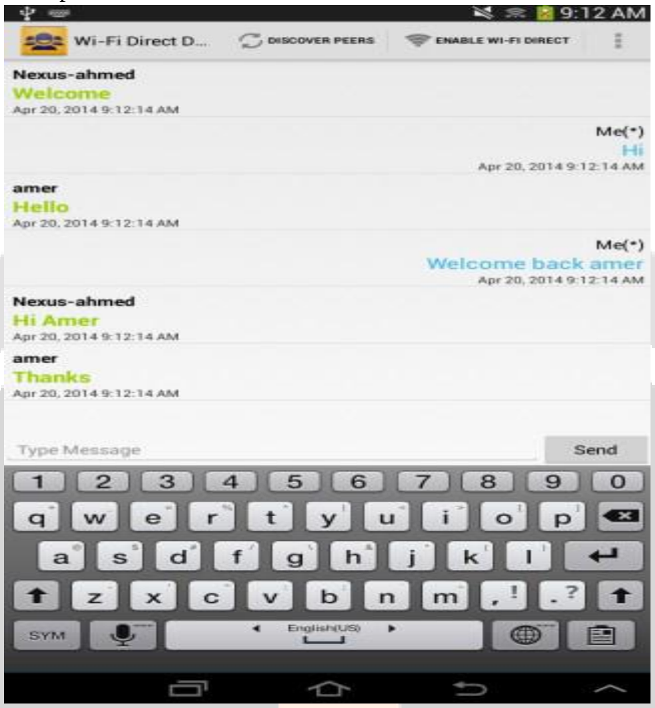

A group is formed
This device is the GO?
Yes
No
Open a management server socket
Connect to the management server socket
Open a data exchange server socket
1 sec passed?
Yes
This device is the GO?
Yes
No
Receive haertbeat
Send heartbeat
Decrease TTL for known peers
Is connected to peer?
No
Open a client data exchange socket
Yes
Add to/update peers list
Reset TTL value for that peer
Decrease the TTL value for other peers
5 sec passed?
Yes
This device is the GO?
Yes
No
Find peers with TTL <= 0
Receive peers' list
Peer found?
Yes
No
Remove from peers list
Close management and data sockets
Send the peers' list to all members
A peer is not connected
Yes
No
Open a client data exchange socket
Duplicated sockets for a peer found?
Yes
No
Is my last IP's octet > peer IP's octet
Yes
No
Remove one of the sockets
Reset TTL values for received peers
Find peers with TTL <= 0
Peer found?
Yes
No
Remove from peers list
Close management and data sockets

Fig -3: The flowchart for the group management protocol.

Table 1: Application components and their description

Component	Description
NearByPeers Class	- 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 second/ peers list (GO) every 5 seconds. - Periodically decrease the TTL values for peers every second. - Periodically prune peers every 5 seconds. - 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
SocketManager 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
Chat Fragment	- Handles user text input and sends the chat message to all peers. - Display chat messages from other peers.
Setting Activity	- Allow the user to change the announced username

messages 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 connections to each other. The result was that every device could send a chat message to the other two. Next, the forth device was connected successfully and the data exchange proceeded normally. Figure 4 shows a screenshot where three devices were chatting together. 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.



Nexus-ahmed
Welcome
Apr 20, 2014 9:12:14 AM

Me(*)
Hi
Apr 20, 2014 9:12:14 AM

amer
Hello
Apr 20, 2014 9:12:14 AM

Me(*)
Welcome back amer
Apr 20, 2014 9:12:14 AM

Nexus-ahmed
Hi Amer
Apr 20, 2014 9:12:14 AM

amer
Thanks
Apr 20, 2014 9:12:14 AM

Fig -4: A screenshot while three devices are chatting together .

5. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of the proposed framework. Mainly, we focus on the group management protocol. The goal of the evaluation is to capture the protocol overhead, and assess the ability of the protocol to adapt to topology changes. The assumptions used in this section are shown in Table 2. All the calculations performed next are based on the worst-case scenarios.

Table 2: The assumptions used in this section

Parameter	Value
Maximum length of the heartbeat message	N bits
Maximum number of members in the group, including GO	M
Maximum length of peers' list message	N*M bits

5.1 Protocol Overhead

The GO sends a peer list message every β seconds and receives a heartbeat message every α seconds from every member. A GM sends a heartbeat message every α seconds and receives a peers' list message every β seconds.

Total number of message (sent and received) by the GO = $\frac{(M-1)}{\alpha} + \frac{1}{\beta}$ messages per second

Total number of message (sent and received) at each GM = $\frac{1}{\alpha} + \frac{1}{\beta}$ messages per second

The bandwidth consumed by the protocol is the number of bits transmitted and received per second by both GO and GM. Thus, Bandwidth consumption by the GO $= (\frac{(M-1)}{\alpha} X\ N) + (\frac{1}{\beta}\ X\ N X\ M) = (\frac{M-1}{\alpha} + \frac{M}{\beta})$ X N *bps*

Bandwidth consumption at each GM $= \left(\frac{1}{\alpha} X\ N\right) + \left(\frac{1}{\beta}\ X\ N\ X\ M\right) = \left(\frac{1}{\alpha} + \frac{M}{\beta}\right) X\ N\ bps$

If we assume that N, M, α, and β equals 500, 20, 1, and 5 respectively, the total bandwidth consumption at the GO = 11.5 Kbps and the total bandwidth consumption at each GM = 2.5 Kbps. Assuming that the bandwidth for Wi-Fi Direct is 54 Mbps, we can notice that the overhead of the protocol is negligible.

5.2 Topology changes

In this section, we assess how the protocol adapts to topology changes. In case of connecting a new member to the group, the GO starts hearing the heartbeat from that member after α second. This means that after α second of connecting to the group, the GO and the new GM can start data exchange with each other. Other members in the group knows about the addition of the new member once they receive, from the GO, the peers' list message that comes after at most β seconds (additional α seconds should be taken into consideration, as the GO has to wait α seconds before receiving the heartbeat from the new GM). That means that the previous members and the new GM can start data exchange, in the worst case, after α + β seconds. Note that, adding more than one peer at once is handled just as the case of handling the addition of one peer. Let us now consider the case of removing a member from the group. Both the GO and the GMs start decreasing the TTL value for the disconnected peer every second, as they are not hearing from it any more. The GO and GMs perform pruning of peers every β seconds which removes any peer with TTL less than zero. As long as the TTL value for the peer is more than zero, the GO continues to include it in the peers' list message. This causes the GMs to reset their TTL values for the disconnecting peer to γ as they still hear about it. After γ seconds, the GO finds that the peer is already gone, so it stops including it in the peer list. When the GMs receive the revised peers' list message from the GO, they will have a TTL value of γ – β for the disconnected peer (as they already started decreasing the value after receiving the last message for GO). It takes γ – β seconds more before the TTL value reaches zero and the GMs remove the disconnected peer from the list of their nearby peers. Thus, after nearly 2γ – β seconds (γ seconds for the GO to notice the disconnection and γ – β more for the GMs), the removal of a peer will be reflected at all peers. It is worth noting that, the disconnection of many peers at once is handled in the same amount of time mentioned above for one peer. The protocol also can handle the case when a peer is not able to communicate with others due to temporary problem, like interference or jamming. In such a case the GO will not hear the heartbeat message, thus it starts decreasing the TTL value for the mentioned peer. The GMs also will Decrease the TTL value for that peer. If the peer is able to communicate again within γ – 1 seconds, the GO will reset its TTL value. The GMs, in that case, will also reset the TTL value within β seconds of the GO re-initialization of the TTL value for the peer. Therefore, if the channel is jammed for a period less than γ seconds, the group will be able to continue its operation. If we assumed the values 1, 5, 30 for α, β, and γ, respectively, adding new peer takes 1 and 6 seconds for the GO and GMs to handle, respectively. Removing a peer takes 30 and 55 seconds for the GO and GMs respectively.

4. CONCLUSIONS

In this paper, we have proposed a new framework for enabling peer-to-peer 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. Our group management protocol allows treating the Wi-Fi Direct topology, which is by convention a star network, 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 framework is implemented and validated through a group chat application. The proposed framework is extendable and can be applied to any type of applications including audio/video streaming, dissemination of traffic information, broadcasting emergency announcement, and last–mile connectivity. We plan to modify the Android implementation of Wi-Fi Direct to allow a device to be associated with more than one group. In addition, our group protocol assumes full connectivity, i.e., each member can directly reach every other member on the group. In the future, we like to relax such a requirement.

6. REFERENCES

[1]. R. Motta and J. Pasquale, "Wireless P2P: Problem or opportunity," Proc. of the Second International Conference on Advances in P2P Systems, Florence, Italy, Oct 2010.
[2]. Wi-Fi Alliance, "P2P Technical Group, Wi-Fi Peer-to-Peer (P2P) Technical Specification v1.0," December 2009.
[3]. D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device to device communications with wifi direct: overview and experimentation," *IEEE Wireless Communications Magazine*, Vol. 20, No. 3, pp. 96 – 104, 2013.
[4]. J. Thomas and J. Robble, "Off grid communication with Android: Meshing the mobile world," *Proc. of the IEEE Conf. on Technologies for Homeland Security (HST'12)*, Waltham, MA, Nov 2012.
[5]. M. Conti, F. Delmastro, G. Minutiello, and R. Paris, "Experimenting opportunistic networks with WiFi Direct," *Proc. of the 6th Wireless Days Conference*, Valencia, Spain, Nov 2013.
[6]. V. Arnaboldi, M. Conti, and F. Delmastro, "Implementation of cameo: A context-aware middleware for opportunistic mobile social networks," *Proc. of the IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Lucca, Italy, June 2011.
[7]. I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang, "Peer management for iTrust over Wi-Fi Direct," *Proc. of the Int'l Symp. on Wireless Personal Multimedia Comm.*, Atlantic City, NJ, Jun. 2013.
[8]. Jong-Eun Park, Jongmoon Park, and Myung-Joon Lee, "DirectSpace: A Collaborative Framework for Supporting Group Workspaces over Wi-Fi Direct," *Proc. of the 4th Int'l Conf. on Mobile, Ubiquitous, and Intelligent Computing (MUSIC 2013)*, Gwangju, Korea, Sept 2013.
[9]. Tuan Nguyen Duong; Ngoc-Thanh Dinh; YoungHan Kim, "Content sharing using P2PSIP protocol in Wi-Fi direct networks," *Proc. of the 4th Int'l Conf. on Communications and Electronics (ICCE 2012)*, Aug. 2012.
[10].Ahmed Amer Shahin, "WiFi Direct group chat," https://play.google.com/store/apps/details?id=esnetlab.apps.android.wifidirect.discovery, April 2014.