Android Project Report, Comp 422

By: Asma Mehjabeen amehjabeen@luc.edu
     Keerthana Aravamudhan karavamudhan@luc.edu

Date: 9th December, 2013

Topic: Hangman over Wi-Fi Direct

Overview: Interconnect two devices using Wi-Fi direct and allow the users of the devices to play the classic hangman game; and a game session starts. The players take turns in guessing words. The player who initiates the connection goes first and gets to specify a word, which the second player has to guess, and a hint (displayed to player 2). The second player tries to guess the letters of the word and player 1 decides whether the letter is in the word or not. With every wrong guess, a part the man is hanged. The game ends if the second player guessed the word or if the hangman diagram is completed. The total score is displayed at the end of a game session. Any player can end the game session at any time.

Links to Documentation:

Creating P2P Connections with Wi-Fi -

Using Wi-Fi P2P for service discovery -


Software Available at:

Google Drive -
https://drive.google.com/folderview?id=0B_VfnnUGlbOAQ093UIJ1UExCeE0&usp=sharing&usp=sharing&tid=0BwsTRjyLbRNOUGpKUjBOS3J5ejQ

Key Points:

1. Default layout folder contains the layouts for portrait mode. If it needs to be compatible for landscape then include a layout-land folder and add the same xml files with the same name to this folder.
2. drawable-mdpi folder contains the images that are displayed in the portrait mode and
drawable-xhdpi folder contains the images that are displayed in the landscape mode.

3. Wi-Fi P2P can be used to connect to two devices together. The devices should run on
Android version 4.0 or later in order to use Wi-Fi P2P. The devices need not be
connected to a central access point in order to establish a connection but their Wi-Fi
should be turned on at all times.

Report:

1.
   a. To include WiFi Direct in our application we need to add the following permissions in
      the manifest file:
      
      
      <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
      <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
      <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
      <uses-permission android:name="android.permission.INTERNET"/>
      <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

   b. Then the application needs to set up the broadcast receiver that would listen to these
      four intents:
      WIFI_P2P_STATE_CHANGED_ACTION - Indicates whether Wi-Fi P2P is enabled
      WIFI_P2P_PEERS_CHANGED_ACTION - Indicates that the available peer list has
      changed.
      WIFI_P2P_CONNECTION_CHANGED_ACTION - Indicates the state of Wi-Fi P2P
      connectivity has changed.
      WIFI_P2P_THIS_DEVICE_CHANGED_ACTION - Indicates this device's configuration
      details have changed.

   c. In the onCreate() method of our activity we need to call the initialize() method using
      WiFiP2PManager class which returns the channel used for connecting the application to
      the WiFiP2Pframework.

   d. Then make a call to discoverServices() or discoverPeers() based on the need of the
      application.

   e. In order to connect to a peer device, a call to wifiP2Pmanager.connect() should be
      made which takes in the configuration of the peer device as a parameter.
f. For disconnecting, we can make a call to WifiP2Pmanager.disconnect().

g. WiFiP2P also contains other methods. An invitation that was sent to a peer device can be cancelled by calling the WiFiP2PManager.cancelConnect() method which cancels any connection invitations sent to peer devices. WiFiP2PManager.createGroup() can be called in order to set up a connection between multiple devices in which the caller of the method will be the group owner and to remove the group, a call to removeGroup() should be made. Also, if a peer device needs to find out who the current group owner is or any information about the group, then the device can make a call to requestGroupInfo() which returns the group information.

2. We can opt to discover all the devices in range by making a call to discoverPeers() function. But if we intend to develop a game and want to discover only devices that have our application installed, then we need to register a service, advertise it and discover devices broadcasting the same service. Our device still discovers all the other devices in range but the listener(that is set up to listen for peer devices) can be used to filter the devices that the app should display to the user.

3. For playing a game, both the devices should have the application running in the foreground of the application. Since the broadcast receiver was registered in onStart() and unregistered in onPause(), the application doesn't listen to the connection info even if the devices are connected when the activity is not in the foreground. Broadcast receivers can be added in the manifest to listen for intents even if the app is not in the foreground but we couldn't find appropriate <action> to add to <intent-filter> in the manifest for Wi-Fi Direct.

4. Fragments can be added to an application statistically or dynamically. For adding static fragments, include them in the layout file in the <fragment> container. For using dynamic fragments include a <FrameLayout> container and then add your fragments dynamically in the activity by using FragmentTransaction class.

5. Additional threads that are created in the application should be given a handler if they need to update the UI or send messages to the main thread. Only the main application thread should update the UI. Handlers can be used for passing messages between threads.

6. For storing, Android provides many storage options such as storing in external storage or in an SQL Lite database or using Shared Preferences. If you need to store less amount of data then Shared Preferences could be an option where data is stored in key-value pairs and could be private to a specific application.
Future Enhancements:

1. Different communication modules (Bluetooth and SMS) can be implemented and the user could be allowed to choose a method of connection to another device.
2. Game could also support single and multiplayer versions.

Questions, Mysteries, Further Work Needed:

1. Finding if a Service can be implemented that runs in the background and listens for connection intents and starts the app on connection to peer device.

2. Remove the code that calls the new game class from the ConnectionInfoListener and let the user explicitly click a button to start the game when devices are connected. This could solve the problem where the game is not being launched on another device since the application was not running in the foreground.