CS5530
Mobile/Wireless Systems
Android WiFi Interface

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

Midterm

- Easy? Difficult?
- Time?
  - Final is 2 hours in class

Android WiFi Interface

- On iOS
  - No permission needed
  - interfaceInfo = CNCopyCurrentNetworkInfo(interfaces)

```
{
    BSSID = "6c:f3:7f:ea:44:f1";
    SSID = "UCCS-Wireless";
    SSIDDATA = <55434353 2d576972 656c6573 73>;
}
```

Android WiFi Interface

- Getting WiFi connection information on Android
  1. Request user permission in AndroidManifest.xml
  2. Create a WiFi manager (class WiFiManager)
  3. Get WiFi information from the manager (class WiFiInfo)
  4. Acquire data from WiFiInfo class

Android WiFi Interface

- WiFi access permission
  - AndroidManifest.xml
  - Add
    - Accessing WiFi: <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    - Changing WiFi: <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
  - Access request shows up when user download&install from Google Play Store

Android WiFi Interface

- Create a WiFi manager
  - https://developer.android.com/reference/android/net/wifi/WifiManager.html
  - Provides the primary API for managing all aspects of Wi-Fi connectivity
  - Syntax
    - Context.getSystemService(Context.WIFI_SERVICE)
  - Example
    - WifiManager wifiManager = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);

Android WiFi Interface

- Get WiFi information
  - https://developer.android.com/reference/android/net/wifi/WifiInfo.html
  - WifiInfo wifiInfo = wifiManager.getConnectionInfo();
  - A lot of information accessible

Android WiFi Interface

- Get data from WiFiInfo class
  - wifiInfo.getBSSID()
    - 6c:f3:7f:ea:44:f1 router's MAC address
  - wifiInfo.getMacAddress()
    - f8:a9:d0:81:70:48 phone's MAC address
  - wifiInfo.getSSID()
    - "UCCS-Wireless"
  - wifiInfo.getIpAddress()
    - 1098172032 phone's IP address (int)

Android WiFi Interface

- Get data from WiFiInfo class
  - wifiInfo.getLinkSpeed()
    - 150 (Mbps), indicated by constant wifiInfo.LINK_SPEED_UNITS (string)
  - wifiInfo.getRssi()
    - -64 (dBm)
  - wifiInfo.getSupplicantState()
    - COMPLETED/ASSOCIATED/ASSOCIATING/AUTHENTICATING/DISCONNECTED/SCANNING/INACTIVE
  - wifiInfo.getFrequency()
    - Available at API level 21 and above (5.0 Lollipop)
    - 5220 (MHz) indicated by constant wifiInfo.FREQUENCY_UNITS (string)

Android WiFi Interface

- Get data from WiFiInfo class
  - wifiInfo.getLinkSpeed()
    - 150 (Mbps), indicated by constant wifiInfo.LINK_SPEED_UNITS (string)
  - wifiInfo.getRssi()
    - -64 (dBm)
  - wifiInfo.getSupplicantState()
    - COMPLETED
  - wifiInfo.getFrequency()
    - Available at API level 21 and above
    - Unit: MHz, indicated by constant wifiInfo.FREQUENCY_UNITS (string)

Android WiFi Interface

- How to change min API level
  - Right click app direcotory → “Open Module Settings” → Select app → “Flavors” tabl → Min SDK version
  - Click Ok, then project will be rebuilt

Android WiFi Interface

- How to convert integer IP to string (old method)
  - int ip = wifiInfo.getIpAddress();
    String ipAddress = Formatter.formatIpAddress(ip);
    Log.v(TAG, ipAddress);

Android WiFi Interface

- How to convert integer IP to string (new method)
  - http://stackoverflow.com/questions/16730711/get-my-wifi-ip-address-android/21181887

```
if (ByteOrder.nativeOrder().equals(ByteOrder.LITTLE_ENDIAN))
        ipAddress = Integer.reverseBytes(ipAddress);

byte[] ipByteArray = BigInteger.valueOf(ip).toByteArray();
String ipAddressString;

try { ipAddressString = InetAddress.getByAddress(ipByteArray).getHostAddress();
} catch (UnknownHostException ex) {
Log.e("WIFIIP", "Unable to get host address.");
ipAddressString = null;
}
```

Android Log method

- Print messages to logcat
  - https://developer.android.com/reference/android/util/Log.html
  - Log.d()
    - DEBUG message
  - Log.e()
    - ERROR message
  - Log.i()
    - INFO message

Android Log method

- Print messages to logcat
  - Log.v()
    - VERBOSE message
  - Log.w()
    - WARN message
  - Log.wtf()
    - What a Terrible Failure: Report a condition that should never happen
  - Arguments:
    - Log.v(String tag, String msg)
    - Tag for identifying this message

Android Log method

- Sometimes logcat has no output
  - Known problem
  - Restart Android studio…
- adb logcat

cat announce.txt_

- Projects??!!
- Last class: project demo

Android Get WiFi Connection

- WifiInfo wifiInfo = wifiManager.getConnectionInfo();

```
Log.v(TAG, wifiInfo.getBSSID());
Log.v(TAG, wifiInfo.getMacAddress());
Log.v(TAG, wifiInfo.getSSID());

int ip = wifiInfo.getIpAddress();
String ipAddress = Formatter.formatIpAddress(ip);
Log.v(TAG, ipAddress);

Log.v(TAG, Integer.toString(wifiInfo.getLinkSpeed()) + wifiInfo.LINK_SPEED_UNITS);
Log.v(TAG, Integer.toString(wifiInfo.getRssi()) + " dBm");
Log.v(TAG, wifiInfo.getSupplicantState().toString());

Log.v(TAG, Integer.toString(wifiInfo.getFrequency()) + wifiInfo.FREQUENCY_UNITS);
```

Android Broadcasts (1)

- Android apps can send/receive broadcast messages from the Android system and other Android apps
- Broadcasts are sent when an event of interest occurs
  - **Android system** sends broadcasts, e.g., when the system boots up or the device starts charging
  - **Apps** can also send custom broadcasts, e.g., to notify other apps that some new data has been downloaded

Android Broadcasts (2)

- Android System Broadcasts
  - System automatically sends broadcasts when various system events occur, e.g., when system switches in/out of airplane mode, *WiFi scan complete*
  - Apps can register to receive specific broadcasts
  - System broadcasts are sent to all apps **subscribed** to receive the event

Android Broadcasts (3)

- To initialize a WiFi scan
  - wifiManager.startScan();
- Each broadcast action has a constant field associated with it
  - android.app.action.ACTION_PASSWORD_CHANGED
  - …
  - android.net.wifi.RSSI_CHANGED
  - android.net.wifi.SCAN_RESULTS
  - android.net.wifi.WIFI_STATE_CHANGED

Receiving Broadcasts (1)

- Create an IntentFilter
  - IntentFilter wifiFilter = new IntentFilter (wifiManager.SCAN_RESULTS_AVAILABLE_ACTION);
- Register the receiver
  - by calling registerReceiver(BroadcastReceiver, IntentFilter)
    - E.g., Intent wifiIntent = getApplicationContext().registerReceiver(wifiReceiver, wifiFilter);

Receiving Broadcasts (2)

- Create an instance of BroadcastReceiver
  - BroadcastReceiver wifiReceiver = new BroadcastReceiver() {

```
@Override
public void onReceive(Context context, Intent intent) {
    // define what to do when receive the broadcast
}
};
```

- Stop receiving broadcasts
  - unregisterReceiver()

Receiving WiFi Scan Results (1)

```
WifiManager wifiManager;
public static BroadcastReceiver wifiReceiver;
public static IntentFilter wifiFilter;
public static Intent wifiIntent;

wifiManager = (WifiManager) getApplicationContext()
        .getSystemService(Context.WIFI_SERVICE);

if(wifiManager.getWifiState() == wifiManager.WIFI_STATE_ENABLED) {
    wifiFilter = new IntentFilter(wifiManager.SCAN_RESULTS_AVAILABLE_ACTION);
    wifiIntent = getApplicationContext().registerReceiver(wifiReceiver, wifiFilter);
   wifiManager.startScan();     // needs permission android.permission.CHANGE_WIFI_STATE
    ……
}
```

Receiving WiFi Scan Results (2)

```
wifiReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        context.unregisterReceiver(this);
        List <ScanResult> results = wifiManager.getScanResults();

        for(int i=0; i<results.size(); i++){
            Log.v(TAG, "BSSID: " + results.get(i).BSSID);
            Log.v(TAG, "SSID: " + results.get(i).SSID);
            Log.v(TAG, "capabilities: " + results.get(i).capabilities);
            Log.v(TAG, "frequency: " + results.get(i).frequency);
            Log.v(TAG, "level: " + results.get(i).level);
        }
    }
};
```

Permissions to Get WiFi Scan Results (1)

- Android 5.1 (Lollipop, API level 22) or lower
  - Permission in AndroidManifest.xml
    - <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    - <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />

Permissions to Get WiFi Scan Results (2)

- After Android 6.0 (Marshmallow, API level 23)
  - **AndroidManifest.xml** permission *and* **runtime** permission in code
  - AndroidManifest.xml:
    - <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
      <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
      <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" /> or
      <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  - Why?
    - Knowing nearby WiFi hotspots (same for Bluetooth scans) conceptually is location – a relative location. From the relative location it's becoming easier to get an absolute location
    - https://code.google.com/p/android/issues/detail?id=185370

Permissions on Android (1)

- After Android 6.0, permissions are divided into two categories
  - Normal permissions
    - App needs to access data outside the app's sandbox, but there's very little risk to the user's privacy or the operation of other apps. E.g., permission to use network

Permissions on Android (2)

- Normal permissions
  - If an app declares in manifest that it needs a normal permission, system automatically grants permission at **install time**
  - System does not prompt user to grant normal permissions, and users cannot revoke these permissions (only uninstall)
  - ACCESS_NETWORK_STATE
  - ACCESS_WIFI_STATE
  - BLUETOOTH
  - CHANGE_NETWORK_STATE
  - CHANGE_WIFI_STATE
  - .....

Permissions on Android (1)

- After Android 6.0, permissions are divided into two categories
  - Normal permissions
    - App needs to access data outside the app's sandbox, but there's very little risk to the user's privacy or the operation of other apps. E.g., permission to use network
  - Dangerous permissions
    - App wants data that involve user's private information, or could potentially affect user's stored data or the operation of other apps. E.g., read user's contacts
    - If manifest file lists a dangerous permission, the user has to explicitly give approval to app

Permissions on Android (3)

- Dangerous permissions
  - App requests permissions both in manifest *and* from user at **runtime** (pop-up window like iOS)
  - User can revoke permissions at any time (in Settings > Apps), so app needs to check for permissions every time it runs
  - READ/WRITE_CALENDAR
  - CAMERA
  - READ/WRITE_CONTACTS
  - GET_ACCOUNTS
  - ACCESS_FINE_LOCATION
  - ….

Permissions on Android (4)

- Viewing an app's permissions
  - Settings > Apps. Pick an app and scroll down to see the permissions that the app uses
  - adb shell pm list permissions -s (all permissions)
  - adb shell pm list permissions -d -g (list permissions by group)

Permissions at Run Time (1)

- Check for permissions
  - If app needs a dangerous permission, code must check every time it performs an operation that requires this permission
    - User is always free to revoke the permission
  - // thisActivity is the current activity
    int permissionCheck = ContextCompat.checkSelfPermission(thisActivity,
    Manifest.permission.WRITE_CALENDAR);
    - If app has permission, the method returns PackageManager.PERMISSION_GRANTED
    - If app does not have permission, the method returns PERMISSION_DENIED

Permissions at Run Time (2)

- Request permissions
  - If app needs a dangerous permission that was listed in the manifest, it must ask user to grant the permission

```
if (ContextCompat.checkSelfPermission(thisActivity, Manifest.permission.READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(thisActivity,
                new String[]{Manifest.permission.READ_CONTACTS},
                MY_PERMISSIONS_REQUEST_READ_CONTACTS);
}
```

Permissions to Get WiFi Scan Results

- AndroidManifest.xml
  - <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
- Code
  - String[] PERMS_INITIAL={
    Manifest.permission.ACCESS_FINE_LOCATION,
    };

    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
    != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, MY_PERMISSIONS_REQUEST);
    }