



BÀI 4:

www.poly.edu.vn **MẠNG**

ANDROID NÂNG CAO

ĐỊNH VỊ NGƯỜI DÙNG - QUẢN LÝ KẾT NỐI

- ⊙ Kết thúc bài học này bạn có khả năng
 - ⊙ Định vị người dùng
 - ⊙ Quản lý kết nối mạng




Phần I: Định vị người dùng

-  Giới thiệu Location Service

-  Quy trình xác định vị trí người dùng

Phần II: Quản lý kết nối mạng

-  Kiểm tra kết nối mạng của thiết bị

-  Quản lý sử dụng mạng





BÀI 4:
MẠNG

**ĐỊNH VỊ NGƯỜI DÙNG -
QUẢN LÝ KẾT NỐI**

PHẦN 1: ĐỊNH VỊ NGƯỜI DÙNG

Dịch vụ định vị vị trí của người dùng (Location Service)

- Sử dụng GPS và Android Network Location Provider để lấy thông tin vị trí người dùng
- GPS cung cấp vị trí chính xác hơn, nhưng chỉ hoạt động ở ngoài trời và mất nhiều thời gian để trả lại kết quả
- GPS tiêu tốn pin hơn
- Android Network Location Provider sử dụng các cột thu phát sóng (cell tower) và WiFi để xác định vị trí
- Android Network Location Provider hoạt động tốt trong nhà và ngoài trời, tiêu tốn ít pin hơn và trả lại kết quả nhanh hơn GPS
- Có thể sử dụng đồng thời GPS và Android Network Location Provider

Thách thức khi định vị vị trí người dùng

- Khó khăn khi chọn GPS, Cell-ID hay Wifi để xác định vị trí người dùng
- Do người dùng di chuyển liên tục nên phải tính toán lại vị trí người dùng thường xuyên
- Tính chính xác vị trí người dùng khác nhau theo thời gian



Gửi yêu cầu cập nhật vị trí

- Gọi phương thức **requestLocationsUpdates()** của Location Manager
- Truyền tham số **LocationListener** cho requestLocationsUpdates()
- LocationListener phải miêu tả một số phương thức callback mà Location Manager sẽ gọi khi vị trí người dùng thay đổi hoặc trạng thái của service thay đổi

Gửi yêu cầu cập nhật vị trí

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // Called when a new location is found by the network location provider.
        //makeUseOfNewLocation(location);
    }

    public void onStatusChanged(String provider, int status, Bundle extras) {}

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {}
};

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
```

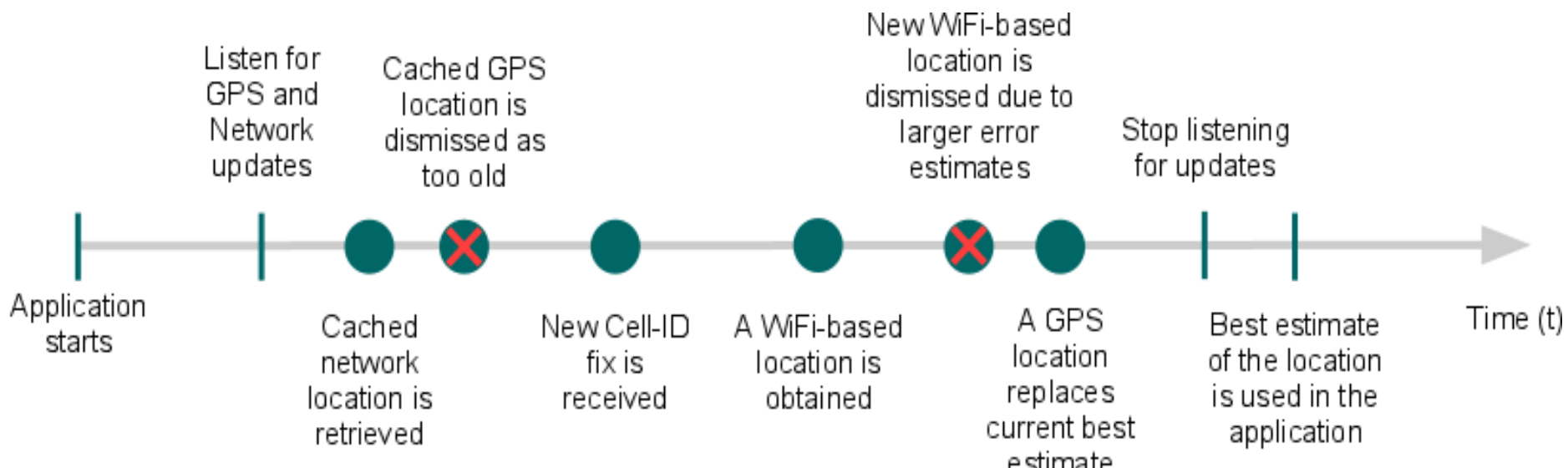

Yêu cầu User Permission

- Nếu cập nhật vị trí từ GPS Provider, thay thế NETWORK_PROVIDER bằng GPS_PROVIDER
- Khi sử dụng NETWORK_PROVIDER, phải thêm quyền ACCESS_COARSE_LOCATION vào file Android Manifest
- Nếu sử dụng cả hai NETWORK_PROVIDER và GPS_PROVIDER, chỉ cần thêm quyền ACCESS_FINE_LOCATION

```
<manifest ... >  
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />  
    ...  
</manifest>
```

Quy trình xác định vị trí người dùng

- Bước 1: Khởi tạo ứng dụng
- Bước 2: Sau đó, bắt đầu lắng nghe từ provider
- Bước 3: Đánh giá và tìm ra vị trí tốt nhất ở thời điểm hiện tại bằng cách loại bỏ vị trí mới nhưng kém chính xác hơn
- Bước 4: Dừng lắng nghe cập nhật vị trí
- Bước 5: Nhận thông tin đánh giá vị trí



Xác định khi nào bắt đầu lắng nghe

- Bắt đầu lắng nghe cập nhật vị trí bằng cách gọi `requestLocationUpdates()`



```
String locationProvider = LocationManager.NETWORK_PROVIDER;  
// Or, use GPS location data:  
// String locationProvider = LocationManager.GPS_PROVIDER;  
  
locationManager.requestLocationUpdates(locationProvider, 0, 0, locationListener);
```

Cache location

- Nhận thông tin cache location bằng cách gọi `getLastKnownLocation(String)`



```
String locationProvider = LocationManager.NETWORK_PROVIDER;  
// Or use LocationManager.GPS_PROVIDER  
  
Location lastKnownLocation = locationManager.getLastKnownLocation(locationProvider);
```

Cung cấp dữ liệu giả về vị trí người dùng trên Emulator

- Có thể dễ dàng kiểm thử ứng dụng sử dụng location service trên thiết bị thật
- Trên emulator, để cung cấp cho ứng dụng dữ liệu giả về vị trí người dùng (phải sử dụng GPS location data để test trên emulator)
 - Sử dụng DDMS





DEMO

Ứng dụng Android sử dụng Location
Service





BÀI 4:
MẠNG

**ĐỊNH VỊ NGƯỜI DÙNG -
QUẢN LÝ KẾT NỐI**

PHẦN II: QUẢN LÝ KẾT NỐI MẠNG

Kiểm tra kết nối mạng của thiết bị

- Một thiết bị có thể có nhiều kiểu kết nối mạng
- Hai kiểu kết nối mạng chủ yếu là kết nối WiFi hoặc 3G
- WiFi thường có tốc độ nhanh hơn, 3G thì thường chậm hơn
- Trước khi bạn thực hiện các thao tác mạng, bạn nên kiểm tra trạng thái kết nối mạng
- Nếu ứng dụng của bạn không có kết nối mạng, bạn nên thông báo với người dùng



Kiểm tra kết nối mạng của thiết bị

- Hai lớp sau được dùng để kiểm tra kết nối mạng
 - **ConnectivityManager**: Truy vấn trạng thái kết nối mạng, đồng thời thông báo người dùng khi có thay đổi về kết nối mạng của thiết bị
 - **NetworkInfo**: miêu tả trạng thái của mạng ứng với một kiểu cụ thể (WiFi hoặc 3G)



Kiểm tra kết nối mạng của thiết bị

- Sử dụng phương thức **getNetworkInfo** của lớp **NetworkInfo** để kiểm tra kết nối mạng
- Lớp ConnectivityManager có các kiểu **TYPE_WIFI** (mạng WiFi) hoặc **TYPE_MOBILE** (mạng 3G)
- Phương thức isConnected của NetworkInfo để kiểm tra kết nối mạng WIFI, 3G,.. của thiết bị. Phương thức trả lại giá trị true hoặc false
- Khi kiểm tra kết nối mạng trên Emulator, bạn sử dụng phím F8 để tắt và bật 3G trên Emulator
- Phải bổ sung quyền **android.permission.INTERNET** và **android.permission.ACCESS_NETWORK_STATE** trong AndroidManifest.xml để ứng dụng có thể thực hiện các kết nối INTERNET

Kiểm tra kết nối mạng của thiết bị

```
public void checkNetworkStatus(View v)
{
    ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
    boolean isWifiConn = networkInfo.isConnected();
    networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
    boolean isMobileConn = networkInfo.isConnected();
    if (isWifiConn)
    {
        Toast.makeText(getApplicationContext(), "Thiết bị đã kết nối Wifi", Toast.LENGTH_LONG)
            .show();
    }
    if (isMobileConn)
    {
        Toast.makeText(getApplicationContext(), "Thiết bị đã kết nối 3G", Toast.LENGTH_LONG)
            .show();
    }
}
```



DEMO

Kiểm tra kết nối mạng Wifi, 3G của thiết
bị Android



Quản lý sử dụng mạng

- Bạn có thể xây dựng một preference activity cho phép người dùng điều khiển sử dụng tài nguyên cho ứng dụng.
- Ví dụ: Bạn cho phép người dùng upload video chỉ khi thiết bị kết nối mạng Wifi
- Bạn có thể khai báo intent filter cho hành động **ACTION_MANAGE_NETWORK_USAGE** (từ Android 4.0) để miêu tả rằng ứng dụng của bạn sẽ định nghĩa một Activity cho phép điều khiển sử dụng mạng
- **ACTION_MANAGE_NETWORK_USAGE** hiển thị setting để quản lý dung lượng mạng của ứng dụng cụ thể
- Bạn nên khai báo intent filter cho setting activity

Quản lý sử dụng mạng

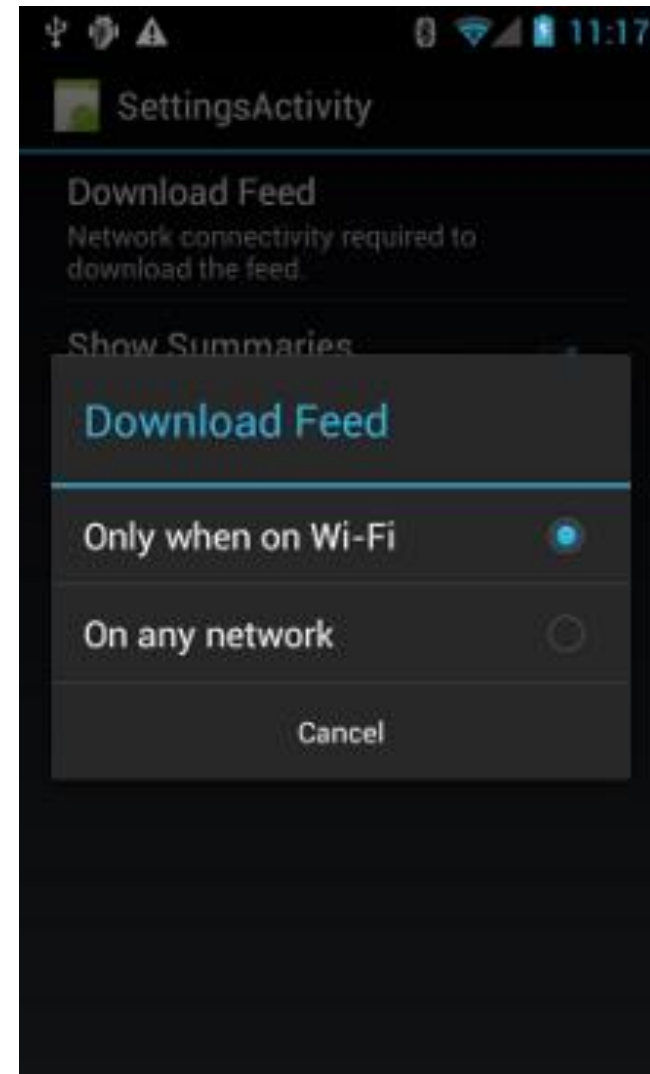
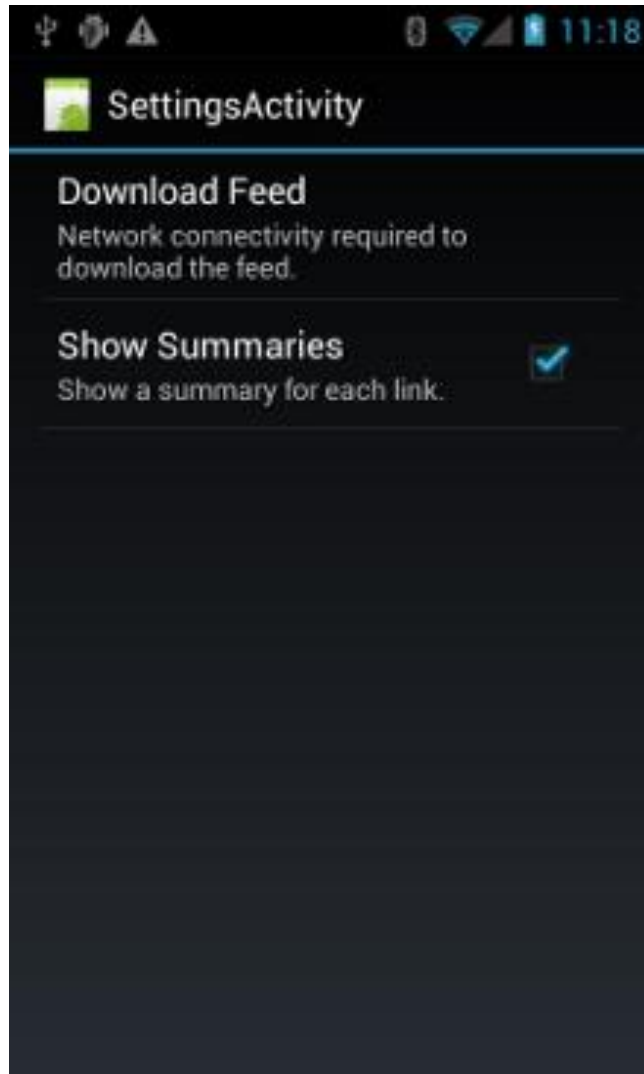
- Trong ví dụ sau, lớp SettingActivity hiển thị giao diện preference cho phép người dùng chọn khi nào download feed từ Sta

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.networkusage"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="4"
        android:targetSdkVersion="16" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name="com.example.android.networkusage.NetworkActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:label="SettingsActivity" android:name=".SettingsActivity">
            <intent-filter>
                <action android:name="android.intent.action.MANAGE_NETWORK_USAGE" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Xây dựng Preference Activity

- Ví dụ, chúng ta sẽ hiển thị màn hình preference cho phép người dùng thiết lập các thông tin sau:
 - Hiển thị thông tin tóm tắt của mỗi XML feed entry hoặc chỉ hiển thị link của mỗi entry
 - Cho phép tải XML feed từ bất kỳ mạng nào hoặc chỉ cho phép tải khi có WiFi





DEMO

Ví dụ NetworkActivity



Đáp ứng thay đổi Preference

- Khi người dùng thay đổi preference trong màn hình setting, ta phải thay đổi hành vi của ứng dụng
- Trong ví dụ code sau, ứng dụng sẽ kiểm tra preference setting trong onStart(), ví dụ kiểm tra nếu setting là WiFi và thiết bị có kết nối Wifi, ứng dụng sẽ tải feed và refresh kết quả

```
public class NetworkActivity extends Activity {
    public static final String WIFI = "Wi-Fi";
    public static final String ANY = "Any";
    private static final String URL =
        "http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest";

    // Whether there is a Wi-Fi connection.
    private static boolean wifiConnected = false;
    // Whether there is a mobile connection.
    private static boolean mobileConnected = false;
    // Whether the display should be refreshed.
    public static boolean refreshDisplay = true;

    // The user's current network preference setting.
    public static String sPref = null;

    // The BroadcastReceiver that tracks network connectivity changes.
    private NetworkReceiver receiver = new NetworkReceiver();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Register BroadcastReceiver to track connection changes.
        IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
        receiver = new NetworkReceiver();
        this.registerReceiver(receiver, filter);
    }
}
```

```
// Refreshes the display if the network connection and the
// pref settings allow it.
@Override
public void onStart() {
    super.onStart();

    // Gets the user's network preference settings
    SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);

    // Retrieves a string value for the preferences. The second parameter
    // is the default value to use if a preference value is not found.
    sPref = sharedPrefs.getString("listPref", "Wi-Fi");

    updateConnectedFlags();

    // Only loads the page if refreshDisplay is true. Otherwise, keeps previous
    // display. For example, if the user has set "Wi-Fi only" in prefs and the
    // device loses its Wi-Fi connection midway through the user using the app,
    // you don't want to refresh the display--this would force the display of
    // an error page instead of stackoverflow.com content.
    if (refreshDisplay) {
        loadPage();
    }
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (receiver != null) {
        this.unregisterReceiver(receiver);
    }
}
```

```
// Checks the network connection and sets the wifiConnected and mobileConnected
// variables accordingly.
private void updateConnectedFlags() {
    ConnectivityManager connMgr =
        (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

    NetworkInfo activeInfo = connMgr.getActiveNetworkInfo();
    if (activeInfo != null && activeInfo.isConnected()) {
        wifiConnected = activeInfo.getType() == ConnectivityManager.TYPE_WIFI;
        mobileConnected = activeInfo.getType() == ConnectivityManager.TYPE_MOBILE;
    } else {
        wifiConnected = false;
        mobileConnected = false;
    }
}

// Uses AsyncTask subclass to download the XML feed from stackoverflow.com.
// This avoids UI lock up. To prevent network operations from
// causing a delay that results in a poor user experience, always perform
// network operations on a separate thread from the UI.
private void loadPage() {
    if (((sPref.equals(ANY)) && (wifiConnected || mobileConnected))
        || ((sPref.equals(WIFI)) && (wifiConnected))) {
        // AsyncTask subclass
        new DownloadXmlTask().execute(URL);
    } else {
        showErrorPage();
    }
}
```

Nhận biết sự thay đổi kết nối mạng

- NetworkReceiver là lớp con của BroadcastReceiver. Khi thay đổi kết nối mạng của thiết bị, NetworkReceiver ngăn cản action `CONNECTIVITY_ACTION`, xác định trạng thái của mạng và thiết lập cờ `wifiConnected` và `mobileConnected` là `true` hoặc `false`
- Thiết lập Broadcast Receiver có thể gây tốn tài nguyên hệ thống nếu Broadcast Receiver thực hiện một số lời gọi không cần thiết
- Trong ví dụ NetworkExample, NetworkReceiver được đăng ký trong `onCreate()` và hủy trong `onDestroy()`.
- Khi bạn đăng ký `<receiver>` trong Manifest, Broadcast Receiver có thể được gọi bất kỳ khi nào, kể cả khi nó không chạy trong một thời gian dài

Nhận biết sự thay đổi kết nối mạng

- Bằng cách đăng ký Broadcast Receiver trong main activity, bạn sẽ đảm bảo Broadcast Receiver sẽ không được gọi khi người dùng rời ứng dụng
- Nếu bạn khai báo `<receiver>` trong AndroidManifest và biết chắc khi nào dùng nó, bạn có thể sử dụng `setComponentEnabledSetting()` để enable và disable khi cần thiết

```
public class NetworkReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        ConnectivityManager connMgr =
            (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        // Checks the user prefs and the network connection. Based on the result, decides
        // whether
        // to refresh the display or keep the current display.
        // If the userpref is Wi-Fi only, checks to see if the device has a Wi-Fi connection.
        if (WIFI.equals(sPref) && networkInfo != null
            && networkInfo.getType() == ConnectivityManager.TYPE_WIFI) {
            // If device has its Wi-Fi connection, sets refreshDisplay
            // to true. This causes the display to be refreshed when the user
            // returns to the app.
            refreshDisplay = true;
            Toast.makeText(context, R.string.wifi_connected, Toast.LENGTH_SHORT).show();
            // If the setting is ANY network and there is a network connection
            // (which by process of elimination would be mobile), sets refreshDisplay to true.
        } else if (ANY.equals(sPref) && networkInfo != null) {
            refreshDisplay = true;

            // Otherwise, the app can't download content--either because there is no network
            // connection (mobile or Wi-Fi), or because the pref setting is WIFI, and there
            // is no Wi-Fi connection.
            // Sets refreshDisplay to false.
        } else {
            refreshDisplay = false;
            Toast.makeText(context, R.string.lost_connection, Toast.LENGTH_SHORT).show();
        }
    }
}
```



Phần I: Định vị người dùng

 Giới thiệu Location Service

 Quy trình xác định vị trí người dùng

Phần II: Quản lý kết nối mạng

 Kiểm tra kết nối mạng của thiết bị

 Quản lý sử dụng mạng





Cảm ơn