Mobile Software Development
Framework: Android IPC;
Intro to Mobile Push Notification

10/16/2012

Y. Richard Yang
Outline

- Admin
- Android Inter-process communications
- Mobile push notification
Admin.

- Schedule for the rest of semester
Recap: Event Handler Execution

- Event handler (EH) executed by the main/UI thread’s Looper
- Slow EH blocks event processing
- Guidelines
  - Notify user
  - Incremental update
  - Short/non-blocking handler, real processing offloaded to other thread
public class LoadingScreen extends Activity implements Runnable {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.loading);

        // start a new thread to load
        Thread thread = new Thread(this);
        thread.start();
    }

    public void run() {
        longRunningTask();
        setContentView(R.layout.main);
    }

    ...
}

Conflicts with UI thread
Recap: Android Handler

Background thread sends msg through handler to UI thread, who processes the msg.
Recap: Fixing LoadingScreen

```java
public class LoadingScreen extends Activity implements Runnable {
    private Handler mHandler = new Handler(); // UI handler

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.loading);
        // start a new thread to load
        Thread thread = new Thread(this);
        thread.start();
    }

    public void run() {
        longTask();
        mHandler.post(mSetFinalViewTask);
    }

    private Runnable mSetFinalViewTask = new Runnable() {
        public void run() {
            setContentView(R.layout.main);
        }
    };
}
```
Recap: Inter-Process Communications (IPC)

- Objective: reuse existing data and services among Android components
Recap: Inter-Process Communications (IPC)

- Component
  - `startActivity()`
  - `startActivityForResult()`

- Component
  - `startService()`
  - `bindService()`

- Component
  - `broadcastIntent()`

Activity

Service

Broadcast Receiver
Recap: Intent Data Structure

- Primary pieces of info in an Intent
  - **Action**: The general action to be performed
    - `ACTION_VIEW`, `ACTION_DIAL`, `ACTION_EDIT`, ...
    - Your own definition of strings
  - **Data**: a URI
    - `tel:123`
    - `content://contacts/people/1`
    - `http://zoo.cs.yale.edu/classes/cs434`
    - `hotel://name/Omni_New_Haven`

- Other attributes
  - **Category**
  - **Type** (MIME type)
  - **Component** (class name)
  - **Extras** (key-value store)
Explicit Intent

Only the specified activity receives this message

http://developer.android.com/training/basics/firstapp/starting-activity.html
Make sure `AndroidManifest.xml` announces activities that can be started

```xml
<application
    android:icon="@drawable/icon"
    android:label="@string/app_name" >

<activity
    android:name=".IntentController"
    android:label="IntentController" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity
    android:name=".TipCal"
    android:label="TipCal" >
</activity>
```
public class IntentController extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.intentcontroller);

        // launch tip cal button
        Button tipBtn = (Button) findViewById(R.id.tipButton);
        tipBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent tipIntent = new Intent(IntentController.this, TipCal.class);
                startActivity(tipIntent);
            }
        });
    }
}
private void startGame() {
    Intent launchGame = new Intent(this, CoolGameA.class);

    // passing information to launched activity
    launchGame.putExtra("userName", userName);
    launchGame.putExtra("userScore", userScore);

    startActivityForResult(launchGame, PLAY_GAME);
}"
public class CoolGameA extends Activity {
    private TextView tv2; int previousScore, score; String user;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.game);

        tv2 = (TextView) findViewById(R.id.game_text);

        //Get the intent that started this activity to fetch passed info
        Intent i = getIntent();

        //returns [] if not initialized by calling activity
        user = i.getStringExtra("userName");

        //returns -1 if not initialized by calling activity
        previousScore = i.getIntExtra("userScore", -1);

        tv2.setText(user + ":" + previousScore);
        doSessionWithInput(user, previousScore);
    }
StartActivity for Result: Callee

```java
//change values for an example of return
score = previousScore - 41;

//setup button listener
Button startButton = (Button) findViewById(R.id.end_game);
startButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
        //return information to calling activity
        Intent i = getIntent();
        i.putExtra("returnScore", score);
        i.putExtra("returnName", user);
        setResult(RESULT_OK, i);
        finish();
    }
});
```
private void startGame() {
    Intent launchGame = new Intent(this, CoolGameA.class);

    // passing information to launched activity
    launchGame.putExtra("userName", userName);
    launchGame.putExtra("userScore", userScore);

    startActivityForResult(launchGame, PLAY_GAME);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLAY_GAME && resultCode == RESULT_OK) {
        userName = data.getExtras().getString("returnName");
        userScore = data.getExtras().getInt("returnScore");

        // show it has changed
        tv.setText(userName + ":" + userScore);
    }
    super.onActivityResult(requestCode, resultCode, data);
}
public class PlayMusicService extends Service {

    public void onCreate() {
        super.onCreate();
    }

    public int onStartCommand(Intent intent, int flags, int startId) {
        play_music();
        return 1;
    }

    private void play_music() {
        while (true) {
            play_music_note(currentIndex);
            currentIndex++;
        }
    } // end of play_music

http://developer.android.com/guide/components/services.html
Discussion

- Problem of explicit intent
Outline

- Admin
- Android Inter-process communications
  - Intent data structure
  - Explicit intent
  - Implicit intent
Intent Resolution: Implicit Intent

- Intent does not specify exact class to run
  - Info in the Intent used by the system to determine the best component, at run time, to handle the intent
Implicit Intents

Implicit Intent
Action: VIEW

Yelp

Handles Action: VIEW

Browser A
Implicit Intents

Implicit Intent Action: VIEW

Yelp

Handles Action: VIEW

Browser A

Handles Action: VIEW

Browser B
**Intent Filter**

- **Problem:** how to know what an Activity/Service can handle?
- **Solution:** Activities/Services/Receivers declare what they can/want to receive in Intent filter

```xml
<intent-filter ...>
   <action android:name="com.example.project.SHOW_CURRENT" />
   <action android:name="com.example.project.SHOW_RECENT" />
   <action android:name="com.example.project.SHOW_PENDING" />
   ...  
</intent-filter>
```

```xml
<intent-filter ...>
   <category android:name="android.intent.category.DEFAULT" />
   <category android:name="android.intent.category.BROWSABLE" />
   ...  
</intent-filter>
```

```xml
<intent-filter ...>
   <data android:type="video/mpeg" android:scheme="http" ... />
   <data android:type="audio/mpeg" android:scheme="http" ... />
   ...  
</intent-filter>
```
Intent Filter: Example

- AndroidManifest.xml file for com.android.browser

```xml
<intent-filter>
  <action android:name="android.intent.action.VIEW"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <scheme android:name="http"/>
  <scheme android:name="https"/>
  <scheme android:name="file"/>
</intent-filter>
```

```java
String action = "android.intent.action.VIEW";
Uri data = Uri.parse("http://www.google.com");
Intent myIntent = new Intent(action, data);
startActivity(myIntent);
```
public class IntentController extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.intentcontroller);

        // launch dial button
        Button dialBtn = (Button) findViewById(R.id.dialButton);
        dialBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String action = "android.intent.action.DIAL";
                String phno = "tel:4326400";
                Uri data = Uri.parse(phno);
                Intent dialIntent = new Intent(action, data);
                startActivity(dialIntent);
            }
        });
    }
}
<activity android:name=".Booking" android:label="Booking">
    <intent-filter>
        <action android:name="com.hotelapp.ACTION_BOOK" />
        <data android:scheme="hotel" android:host="name"/>
    </intent-filter>
</activity>
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Intent intent = getIntent(); // why am I called
    String action = intent.getAction();
    Uri data = intent.getData();

    String hotelName = data.getPath();

    // do the booking
    setResult(RESULT_OK);
    finish();
}
A Design Template: Invoker

```java
String action = "com.hotelapp.ACTION_BOOK";
String hotel = "hotel://name/" + selectedHotel;
Uri data = Uri.parse(hotel);
Intent bookingIntent = new Intent(action, data);
startActivityForResults(bookingIntent, requestCode);
```
Outline

- Admin
- Android Inter-process communications
  - Intent data structure
  - Explicit intent
  - Implicit intent
  - Content provider as target of intent
Content Provider

- Enable uniformed API for sharing data across applications
  - E.g., Address book, photo gallery
- Each provider can expose its data as a simple table on a database model

<table>
<thead>
<tr>
<th>_ID</th>
<th>NUMBER</th>
<th>NUMBER_KEY</th>
<th>LABEL</th>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>(425) 555 6677</td>
<td>425 555 6677</td>
<td>Kirkland office</td>
<td>Bully Pulpit</td>
<td>TYPE_WORK</td>
</tr>
<tr>
<td>44</td>
<td>(212) 555-1234</td>
<td>212 555 1234</td>
<td>NY apartment</td>
<td>Alan Vain</td>
<td>TYPE_HOME</td>
</tr>
<tr>
<td>45</td>
<td>(212) 555-6657</td>
<td>212 555 6657</td>
<td>Downtown office</td>
<td>Alan Vain</td>
<td>TYPE_MOBILE</td>
</tr>
<tr>
<td>53</td>
<td>201.555.4433</td>
<td>201 555 4433</td>
<td>Love Nest</td>
<td>Rex Cars</td>
<td>TYPE_HOME</td>
</tr>
</tbody>
</table>

- Query, delete, update, and insert rows
Each content provider exposes a public URI that uniquely identifies its data set:

- android.provider.Contacts.Phones.CONTENT_URI
- android.provider.Contacts.Photos.CONTENT_URI
- android.provider.CallLog.Calls.CONTENT_URI
- android.provider.Calendar.CONTENT_URI

A content consumer declares access requirement

- `<uses-permission android:name="android.permission.READ_CONTACTS"/>`
private void pickContact() {
    // Create an intent to "pick" a contact, as defined by the content provider URI
    Intent intent = new Intent(Intent.ACTION_PICK, Contacts.CONTENT_URI);
    startActivityForResult(intent, PICK_CONTACT_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // If the request went well (OK) and the request was PICK_CONTACT_REQUEST
    if (resultCode == Activity.RESULT_OK && requestCode == PICK_CONTACT_REQUEST) {
        // Perform a query to the contact's content provider for the contact's name
        Cursor cursor = getContentResolver().query(data.getData(),
                                                      new String[]{Contacts.DISPLAY_NAME}, null, null, null);
        if (cursor.moveToFirst()) { // True if the cursor is not empty
            int columnIndex = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
            String name = cursor.getString(columnIndex);
            // Do something with the selected contact's name...
        }
    }
}
Outline

- Admin
- Android Inter-process communications
  - Intent data structure
  - Explicit intent
  - Implicit intent
  - Content provider as target of intent
  - Broadcast intent
Broadcast Intents

- Multiple components may be interested in an event/update
  - e.g., system event such as an incoming phone call, battery level low, network cell changes
  - receives notification by declaring a broadcast receiver
Intent and Broadcast: Sender

String action = "edu.yale.cs434.RUN";

Intent cs434BroadcastIntent =
    new Intent(action);
    cs434BroadcastIntent.putExtra("message",
        "Wake up.");

sendBroadcast(cs434BroadcastIntent);

Example: IntentController
Intent and Broadcast: Receiver

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

<receiver android:name="MyPhoneReceiver">
    <intent-filter>
        <action android:name="android.intent.action.PHONE_STATE"/>
    </intent-filter>
</receiver>

<receiver android:name=".CS434BroadcastReceiver" android:enabled="true">
    <intent-filter>
        <action android:name="edu.yale.cs434.RUN" />
    </intent-filter>
</receiver>
```
public class CS434BroadcastReceiver extends BroadcastReceiver {
    public static final String CUSTOM_INTENT = "edu.yale.cs434.RUN";

    // Display an alert that we've received a message.
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(CUSTOM_INTENT)) {
            String message = (String)intent.getExtras().get("message");
            CharSequence text = "Got intent " + CUSTOM_INTENT + " with " + message;
            int duration = Toast.LENGTH_SHORT;

            Toast mToast = Toast.makeText(context, text, duration);
            mToast.show();
        } // end of if
    } // end of onReceive
}
public class MyPhoneReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle extras = intent.getExtras();
        if (extras != null) {
            String state = extras.getString(TelephonyManager.EXTRA_STATE);
            if (state.equals(TelephonyManager.EXTRA_STATE_RINGING)) {
                String phoneNumber = extras.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
                Toast.makeText(context, "Incoming number: " + phoneNumber, Toast.LENGTH_LONG).show();
            } // end of if
        } // end of if
    } // end of onReceive
Discussion: Downside of Implicit Intent
Real Example App: iMDb

![iMDb Showtimes app screenshot]

**Date & Location**
- Thursday, June 23

**Current Location**
- Current Location

**New This Week**

- **Bad Teacher (2011)**
  - Rated R, 1 hr 32 mins, 6.3/10
  - Showtimes from Century Richmond Hilltop 16, Century San Francisco Centre 9 and XD, and 1 other...

- **Cars 2 (2011)**
  - Rated G, 1 hr 53 mins, 6.9/10
  - Showtimes from AMC Bay Street 16, AMC Bay Street 16 and 8 others...

[Image of the iMDb app showing showtimes for two movies: Bad Teacher and Cars 2.]
Example App

IMDb App

Handles Actions:
willUpdateShowtimes,
showtimesNoLocationError

Showtime Search

Implicit Intent
Action:
willUpdateShowtimes

Results UI
Vulnerability: Eavedropping

IMDb App

Showtime Search

Implicit Intent Action: 

willUpdateShowtimes

Eavesdropping App

Handles Action:

willUpdateShowtimes,

showtimesNoLocationError

Sending Implicit Intents makes communication public
Vulnerability: Spoofing

Malicious Injection App

Malicious Component

Action: showtimesNoLocationError

IMDb App

Handles Action: willUpdateShowtimes, showtimesNoLocationError

Results UI

Receiving Implicit Intents makes the component public
Vulnerability: Man-in-the-Middle

IMDb App

Handles Action:

- willUpdateShowtimes,
- showtimesNoLocation Error

Man-in-the-Middle App

Handles Action:

- willUpdateShowtimes,
- showtimesNoLocation Error

Action:

- willUpdateShowtimes

Showtime Search

Results UI

Malicious Receiver

Action:

- showtimesNoLocation Error
Vulnerability: Spoofing

New This Week

- **Bad Teacher (2011)**
  - Rated R, 1 hr 32 mins, 6.3/10
  - Showtimes from Century Richmond Hilltop 16, Century San Francisco Centre 9 and XD, and 1 other...

- **Cars 2 (2011)**
  - Rated G, 1 hr 53 mins, 6.9/10
  - Showtimes from AMC Bay Street 16, AMC Bay Street 16, and 8 others

Please specify a location
No showtimes were found for the selected date and location.
Vulnerability: Permission re-Delegation

Permission re-delegation occurs when an application without a permission gains additional privileges through another application.
Permission System

Malware

Deputy

toggleWifi()
toggleWifi()
Permission Redelegation

- Malware
  - pressButton(0)

- Deputy
  - toggleWifi()

- Permission System
  - API
Permission Redelegation: Reality Check

- Analyzed manifests of 5 system applications
- Built attacks using 5 of the 16 system apps
- Found 15 attacks in the 5 applications
More Examples of Attack

DeskClock:
- Start an internal service
- Tell it to infinitely vibrate with a WAKE_LOCK on

Phone:
- Trigger the “phone call answered” message receiver
- Phone call will be silenced, vibrate cancelled

More details see schedule page links
More Info on IPC

- Intent is a high level abstraction

- For more details on implementation of Intent, see a set of slides on binder
Progress So Far

- **Issue (responsiveness):** slow UI is a sin
  - Solution: event listener gives tasks to background thread(s)
  - Issue: Background threads may need to update UI
  - Solution: Handler/AsyncTask so that one thread can delegate tasks to another thread

- **Issue (composability):** reusability is highly desired
  - Solution: Intent
Accessing Data in the Cloud

Challenge: How do you keep data on a device fresh?
Progress So Far

- **Issue (responsiveness): slow UI is a sin**
  - Solution: event listener gives tasks to background thread(s)
  - Issue: Background threads may need to update UI
  - Solution: Handler/AsyncTask so that one thread can delegate tasks to another thread

- **Issue (composability): reusability is highly desired**
  - Solution: Intent
Solution 1: Polling

- Simple to implement

- Device periodically asks server for new data/update

- Appropriate for content that changes constantly
  - Stock Quotes, News Headlines

- Problems?
Impact of Polling on Battery

- Baseline: ~5-8 mA
- Network: ~180-200 mA
  - Tx more expensive than Rx
- Assume radio stays on for 10 sec.
  - Energy per poll: ~0.50 mAh
  - 5 min frequency: ~144 mAh / day
- Droid 2 total battery: 1400 mAh

Source: Android development team at Google
Solution 2: Push Notification

- Design issue: Who to push to client device?
  - Option 1: each app does it individually
  - Option 2: a shared push service
Push Service

- A single **persistent** connection from device to a cloud push service provider
- Multiple application providers push to the service provider
- Service provider pushes to a device using the persistent connection

- Two examples
  - Apple Push Notification Service (APNS)
  - Google Cloud Messaging (GCM)