CS434/534: Topics in Networked (Networking) Systems

Mobile System: Android Component Composition/Inter-process Communication (IPC)

Yang (Richard) Yang
Computer Science Department
Yale University
208A Watson
Email: yry@cs.yale.edu

http://zoo.cs.yale.edu/classes/cs434/
Admin.

- Project planning
  - Start: Apr. 5
    - A meeting w/ instructor; Due: team, potential topics
  - Checkpoint I: Apr. 13 (updated; Thursday)
    - Initial ideas; survey of related work; feedback from instructor/TF
  - Checkpoint II: Apr. 20 (updated)
    - Initial design; feedback from instructor/TF
  - Checkpoint III: Apr. 27 (updated)
    - Design refined; key components details
  - Final report: May 4
    - No more than 6 page write up
Recap: Android Architecture
Recap: Most Basic Android UI App Software Concepts

- Activity

- UI (View/Layout)
  - External definition of views in XML
  - findViewById() to reduce coupling

- Link view events to event handlers
  - set...Listener()

- Other concepts such as fragments which we skip
Recap: Android Execution Model

- One Linux process for each application with a single thread of execution
  - The very important thread is also called the "main" thread or the UI thread

- A key systems complexity of Android programming is to avoid blocking/conflicting with the UI 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);
    }

    ...
Recap: Solutions

- Background thread, with messaging to UI thread on tasks on UI domain
  - Activity.runOnUiThread(Runnable)
  - View.post(Runnable)
  - View.postDelayed(Runnable, long)

- Handler

- AsyncTask
Recap: Post to UI Thread

```java
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            // a potentially time consuming task
            final Bitmap bitmap = processBitMap("image.png");

            mImageView.post(new Runnable() {
                public void run() {
                    mImageView.setImageBitmap(bitmap);
                }
            }); // post
        }
    }).start();
} // run
```
Recap: Android Handler

Background thread sends msg through handler to UI thread, who processes the msg
Handler

public class MyActivity extends Activity {

[...]

// Need handler for callbacks to the UI thread
final Handler mHandler = new Handler();

// Create runnable task to give to UI thread
final Runnable mUpdateResultsTask = new Runnable() {
    public void run() {
        updateResultsInUi();
    }
};

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

[...]
protected void startLongRunningOperation() {

    // Fire off a thread to do some work that we shouldn't do directly in the UI thread
    Thread t = new Thread() {
        public void run() {
            mResults = doSomethingExpensive();
            mHandler.post(mUpdateResultsTask);
        }
    };
    t.start();
}

private void updateResultsInUi() {

    // Back in the UI thread -- update our UI elements based on the data in mResults
    [
    ...

}
Example: 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);
        }
    };
}
```
private Handler mHandler = new Handler(); // UI handler
private Runnable longTask = new Runnable() {

    // processing thread
    public void run() {
        while (notFinished) {
            // doSomething

            mHandler.post(taskToUpdateProgress);
        }

        // mHandler.post(taskToUpdateFinalResult)
    }

    // mHandler.post(taskToUpdateFinalResult)

};

Thread thread = new Thread(longTask);
thread.start();
AsyncTask as Abstraction

private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) { // on some background thread
        int count = urls.length; long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
        }
        return totalSize;
    }
    protected void onProgressUpdate(Integer... progress) { // on UI thread!
        setProgressPercent(progress[0]);
    }
    protected void onPostExecute(Long result) { // on UI thread!
        showDialog("Downloaded "+ result + " bytes");
    }
}

new DownloadFilesTask().execute(url1, url2, url3); // call from UI thread!
Service: Working in Background

Why Android Service:

- Faceless components that typically run in the background
  - the user is not directly interacting with the application).
  - NO GUI, higher priority than inactive Activities
- The system to schedule work for the service, to be run until the service or someone else explicitly stops it.

Note

- A Service is not a separate process. The Service object itself does not imply it is running in its own process; unless otherwise specified, it runs in the same process as the application it is part of.
- A Service is not a thread. It is not a means itself to do work off of the main thread (to avoid Application Not Responding errors).
Outline

- Admin
- Android
  - Platform overview
  - Basic concepts
    - Activity, View, External Resources, Listener
  - Execution model with multiple threads
    - Post to UI thread, Handler, AsyncTask
  - Component composition
Motivation

- We have a much better system if the components can be composed, e.g.,
  - invoke the compose email activity by another application

- Recall:
  - GNURadio composition model
  - TinyOS composition model

- How might Android conduct composition?
Intent

- An intent is an abstract description of an operation to be performed.
  - Indicate operations from your own or others

http://developer.android.com/reference/android/content/Intent.html
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

Yelp

To: MapActivity.class

Map App

class: MapActivity

Only the specified activity receives this message

http://developer.android.com/training/basics/firstapp/starting-activity.html
Explicit Intent: specifies the exact class to run

```java
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);
            }
        });
    }
}
```
Explicit Intent and Manifest

Make sure AndroidManifest.xml announces activities to 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>
</application>
```

Shown in Launcher

Announce class
Exercise: IntentController
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);
    }
//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);
}
Explicit Intent: Start Service

```java
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
  - Platform overview
  - Basic concepts
    - Activity, View, External Resources, Listener
  - Execution model with multiple threads
    - Post to UI thread, Handler, AsyncTask
  - Component composition
    - Intent
    - 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

Yelp

Implicit Intent
Action: VIEW

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);
```
**Implicit Start Activity**

```java
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);
            }
        });
    }
}
```

**See IntentController**
<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();
}
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
  - Platform overview
  - Basic concepts
    - Activity, View, External Resources, Listener
  - Execution model with multiple threads
    - Post to UI thread, Handler, AsyncTask
  - Component composition
    - Intent
    - 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
Content Provider and Intent

- 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.CalendarCALENDAR_URI

- A content consumer declares access requirement
  - <uses-permission android:name="android.permission.READ_CONTACTS"/>
Content Consumer

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.moveToNext()) { // 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

- Platform overview
- Basic concepts
  - Activity, View, External Resources, Listener
- Execution model with multiple threads
  - Post to UI thread, Handler, ASyncTask
- Component composition
  - Intent
  - 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
String action = "edu.yale.cs434.RUN";

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

sendBroadcast(cs434BroadcastIntent);
Intent and Broadcast: Receiver

```xml
<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
}
Summary: Android Gang of Four
Example: A SocialApp
Outline

Admin

Android

- Platform overview
- Basic concepts
  - Activity, View, External Resources, Listener
- Execution model with multiple threads
  - Post to UI thread, Handler, ASyncTask
- Component composition
  - Intent, Explicit intent, Implicit intent, Content provider as target of intent, Broadcast intent
  - Composition (IPC) internal
Intent is a high level abstraction, making it appear that one can call services in another app.
Intent Appearance (Traditional Call)

In the same process
Real Happening

caller

callee

How?

caller

callee

interface

interface
What is the Issue?

- Applications and services may run in separate processes
- Each process has its own address space
- Hence supporting intent need to handle inter-processor communications (IPC), which can introduce significant processing overhead and security holes
IPC Mechanisms

- In GNU/Linux
  - Signal
  - Pipe
  - Socket
  - Semaphore
  - Message queue
  - Shared memory

- In Android
  - Binder: lightweight RPC (Remote Procedure Communication) mechanism
Binder framework provides a mechanisms such that methods on remote objects can be called as if they were local object methods.
“In the Android platform, the binder is used for nearly everything that happens across processes in the core platform. ” – Dianne Hackborn