CS434/534: Topics in Networked (Networking) Systems

Mobile Networking System: Network Service Discovery and Composition

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: 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
- UI thread and other thread coordination
  - post, handler, AsyncTask to send tasks to UI thread
Recap: App/Component Composition

- A system is more extensible/flexible if the apps/components can be composed

- Android composition glue: intent
  - An intent is an abstract description of an operation to be performed
    - An intent is defined by action, data, attributes such as category, type (MIME type), component (class name), and extras (key-value store)
  - Intent discovery: Manifest file announces related info

```xml
<application
  <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> ...
```

- Try: `adb shell am start -a android.intent.action.view -d URL`
Recap: App/Component Composition

Intent is used not only to find service, but also pass param, return results

```java
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);
}

// CoolGameA
// return information to calling activity
Intent i = getIntent();
    i.putExtra("returnScore", score); i.putExtra("returnName", user);
    setResult(RESULT_OK, i); finish();
```
Recap: App/Component Composition: Content Provider

- 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"/>`

```java
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);
}
```
Summary: Android Gang of Four
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
    - Intent security
Discussion: Downside of Implicit Intent
Real Example App: iMDb

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 3 others...
Example App

IMDb App

Handles Actions:
\textit{willUpdateShowtimes}, \textit{showtimesNoLocationError}

Showtime Search

Implicit Intent Action:
\textit{willUpdateShowtimes}

Results UI
Vulnerability: Eavedropping

**IMDb App**
- Showtime Search
- Implicit Intent Action:
  - `willUpdateShowtimes`
  - `showtimesNoLocationError`

**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,
showtimesNoLocationError

Malicious Receiver

Action:
showtimesNoLocation Error
Vulnerability: Spoofing
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

- **A good topic as a project**
  - Composition security mechanisms
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
    - Intent security
    - Composition (IPC) internal
Binder History

- Developed under the name OpenBinder by Palm Inc. under the leadership of Dianne Hackborn
- Android Binder is the customized re-implementation of OpenBinder, which provides bindings to functions and data from one execution environment to another.
Binder and IPC in Android

“In the Android platform, the binder is used for nearly everything that happens across processes in the core platform." – Dianne Hackborn

Binder framework provides a mechanism such that methods on remote objects can be called as if they were local object methods.

Q: What does the binder framework need to do?
The most basic function of Binder is data transfer:
- Each invocation is called a transaction, and data is called parcel.
- The procedure of building a parcel is called marshalling an object.
- The procedure of rebuilding an object from a parcel is called unmarshalling an object.

Delivering arguments of method

"flatten"
IBinder Interface and Binder Objects

- **IBinder**

- **Binder**

- **Key method**
  - `transact(int code, Parcel data, Parcel reply, int flags)"
If one process sends data to another process, it is called transaction. The data is called transaction data.

<table>
<thead>
<tr>
<th>Target</th>
<th>Binder Driver Command</th>
<th>Cookie</th>
<th>Sender ID</th>
<th>Data:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Target Command 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Target Command 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Target Command n-1</td>
</tr>
</tbody>
</table>
Binder + Service Manager

- Client
- Service Manager
  - service list
  - Name:Handle
  - Name:Handle
  - Name:Handle
- Server
  - onTransact(…)
  - thread pool

Binder Driver: /dev/binder

$ adb shell service list
Generating Call Directly

```bash
$ adb shell service call phone 1 s16 "123"
Result: Parcel(00000000 '....')
```

```java
interface ITelephony {

    /* Dial a number. This doesn't place the call. It displays *
     * the Dialer screen. */
    void dial(String number);

Source: frameworks/base/telephony/java/com/android/internal/telephony/ITelephony.aidl
```

Phone Application appears in foreground.

parameter "1" → dial()
s16 "123" → String("123")

https://github.com/android/platform_frameworks_base/blob/master/telephony/java/com/android/internal/telephony/ITelephony.aidl
Summary: Binder IPC

- Binders are used to communicate over process boundaries since different processes don't share a common VM context.
  - No more direct access to each others Objects (memory).

- Binders are not ideal for transferring large data streams (like audio/video) since every object has to be converted to (and back from) a Parcel.
Mechanisms of Android to discover and invoke services (activities) from other apps in the device

- Discovery
  - Manifest file
  - ServiceManager

- Execution
  - The Binder framework
Outline

- Admin
- Android
  - Platform overview
  - Basic concepts
    - Activity, View, External Resources, Listener
  - Inter-thread: execution model with multiple threads
    - Post to UI thread, Handler, AsyncTask
  - Inter-process: component composition
    - Intent, Explicit intent, Implicit intent, Content provider as target of intent, Broadcast intent
    - Intent security
    - Composition (IPC) internal using binder
  - Inter-machine: network-wise composition
Discussion

- Issues to address when designing a network-wide composition framework
Network Service Discovery and Execution

- src
- dst
- SD
- service
- location
- request
- result
Outline

- Admin
- Android
  - Platform overview
  - Basic concepts
    - Activity, View, External Resources, Listener
  - Inter-thread: execution model with multiple threads
    - Post to UI thread, Handler, AsyncTask
  - Inter-process: component composition
    - Intent, Explicit intent, Implicit intent, Content provider as target of intent, Broadcast intent, Binder
  - Inter-machine: network-wise composition
    - service discovery
General Service/Naming Discovery Paradigm: Linda

- “Distributed workspace” by David Gelernter in the 80’s at Yale

- Very influential in naming and resource discovery

- Key issues
  - How to name services/resources
  - How to resolve names
The Linda Paradigm

- Naming scheme:
  - arbitrary tuples (heterogeneous-type vectors)

- Name resolution:
  - Nodes write into shared memory
  - Nodes read matching tuples from shared memory
    - exact matching is required for extraction
Linda: Core API

- **out()**: writes tuples to shared space
  - example: `out("abc", 1.5, 12).`
  - result: insert (“abc”, 1.5, 12) into space

- **read()**: retrieves tuple copy matching arg list (blocking)
  - example: `read("abc", ? A, ? B)`
  - result: finds (“abc”, 1.5, 12) and sets local variables
    - $A = 1.5$, $B = 12$. Tuple (“abc”, 1.5, 12) is still resident in space.

- **in()**: retrieves and deletes matching tuple from space (blocking)
  - example: same as above except (“abc”, 1.5, 12) is deleted

- **eval(expression)**: similar to out except that the tuple argument to eval is evaluated
  - example: `eval("ab", -6, abs(-6))` creates tuple (“ab”, -6, 6)
Linda Extension: JavaSpaces

- Industry took Linda principles and made modifications
  - add transactions, leases, events
  - store Java objects instead of tuples
  - a very comprehensive service discovery system

- Definitive book, “JavaSpaces Principles, Patterns, and Practice”
  - 2 of 3 authors got Ph.D.’s from Yale
JavaSpaces - Visual Overview
Outline

- Admin
- Android
  - Platform overview
  - Basic concepts
    - Activity, View, External Resources, Listener
  - Inter-thread: execution model with multiple threads
    - Post to UI thread, Handler, AsyncTask
  - Inter-process: component composition
    - Intent, Explicit intent, Implicit intent, Content provider as target of intent, Broadcast intent, Binder
  - Inter-machine: network-wise composition
    - service discovery
      - Linda, DNS
Today's Internet: DNS

clients

<name, service>

IP address

DNS

routers

servers
DNS: Domain Name System

- **Function**
  - map domain name (e.g. cs.yale.edu) and service type (e.g., email) to IP address (e.g. 128.36.232.5)

- **Domain name:** a hierarchical name space implemented by a distributed database to allow distributed, autonomous administration

```
generic     | Countries
-------------|-----------
int         |           | com
edu         |           | gov
mil         | org       | net
acm         | ieee      | jp
keio        | co        | us
nl          |           |...
```

called a zone
DNS: Naming Scheme

- Each DNS server stores resource records (RR)

| RR format: (name, value, type, ttl) |

- Type=A
  - name is hostname
  - value is IP address

- Type=NS
  - name is domain (e.g. yale.edu)
  - value is canonical name

- Type=CNAME
  - name is an alias name for some “canonical” (the real) name
  - value is canonical name

- Type=MX
  - value is hostname of mail server associated with name
A host is configured with a local DNS server, and queries the server, who may forward the query to the corresponding DNS server.
Problems of Traditional DNS for Service Discovery in Mobile Networks

- Depends on infrastructure: availability of DNS server
  - Service announced through DNS server
  - Limited, fixed service types
Leverage DNS message format, but each node can announce its own services
Example

- Use the `dns-sd` command on Mac as example
  - Advertise (register) an LPR printer on port 515

```bash
dns-sd -R "My Test" _printer._tcp . 515
pdl=application/postscript
```
Realizing DNS-SD without Central DNS Server: mDNS

- Multicast in a small world
  - no central address server
    - each node is a responder
  - link-local addressing
    - send to multicast address: 224.0.0.251
Example

- Use the `dns-sd` command on Mac as example
  - Advertise (register) a web page on local machine
    
    ```bash
    dns-sd -R "My Test" _http._tcp 80 path=/path-to-page.html
    ```
  
  - Browse web pages on local machines
    ```bash
    dns-sd -B _http._tcp
    ```
  
  - See the page in Safari Bonjour
Network Service Discovery in Android

- Based on DNS-SD/mDNS
- Foundation for peer-to-peer/Wi-Fi Direct in Android

See
public void registerService(int port) {
    // Create the NsdServiceInfo object, and populate it.
    NsdServiceInfo serviceInfo = new NsdServiceInfo();

    // The name is subject to change based on conflicts
    // with other services advertised on the same network.
    serviceInfo.setServiceName("NsdChat");
    serviceInfo.setServiceType("nsdchat._tcp");
    serviceInfo.setPort(port);
    ...
}

http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml
public void initializeRegistrationListener() {
    mRegistrationListener = new NsdManager.RegistrationListener() {
        @Override
        public void onServiceRegistered(NsdServiceInfo NsdServiceInfo) {
            // Save the service name. Android may have changed it in order to
            // resolve a conflict, so update the name you initially requested
            // with the name Android actually used.
            mServiceName = NsdServiceInfo.getServiceName();
        }
        @Override
        public void onRegistrationFailed(NsdServiceInfo serviceInfo, int errorCode) {
            // Registration failed! Put debugging code here to determine why.
        }
        @Override
        public void onServiceUnregistered(NsdServiceInfo arg0) {
            // Service has been unregistered. This only happens when you call
            // NsdManager.unregisterService() and pass in this listener.
        }
        @Override
        public void onUnregistrationFailed(NsdServiceInfo serviceInfo, int errorCode) {
            // Unregistration failed. Put debugging code here to determine why.
        }
    };
}
public void registerService(int port) {
    // Create the NsdServiceInfo object, and populate it.
    NsdServiceInfo serviceInfo = new NsdServiceInfo();

    // The name is subject to change based on conflicts
    // with other services advertised on the same network.
    serviceInfo.setServiceName("NsdChat");
    serviceInfo.setServiceType("nsdchat._tcp");
    serviceInfo.setPort(port);

    mNsdManager = Context.getSystemService(Context.NSD_SERVICE);
    mNsdManager.registerService(
        serviceInfo, NsdManager.PROTOCOL_DNS_SD, mRegistrationListener);
}
Service Connection

call `resolveService()`

```java
public void initializeResolveListener() {
    mResolveListener = new NsdManager.ResolveListener() {

        @Override
    public void onResolveFailed(NsdServiceInfo serviceInfo, int errorCode) {
        // Called when the resolve fails. Use the error code to debug.
        Log.e(TAG, "Resolve failed" + errorCode);
    }

    @Override
    public void onServiceResolved(NsdServiceInfo serviceInfo) {
        Log.e(TAG, "Resolve Succeeded. " + serviceInfo);

        if (serviceInfo.getServiceName().equals(mServiceName)) {
            Log.d(TAG, "Same IP.");
            return;
        }
    }

    mService = serviceInfo;
    int port = mService.getPort();
    InetAddress host = mService.getHost();
}
```
Discussion

- Issues of the preceding code