CMSC 628: Introduction to Mobile Computing
Basics of Android programming: Activity, Intents, Services, BroadcastReceiver

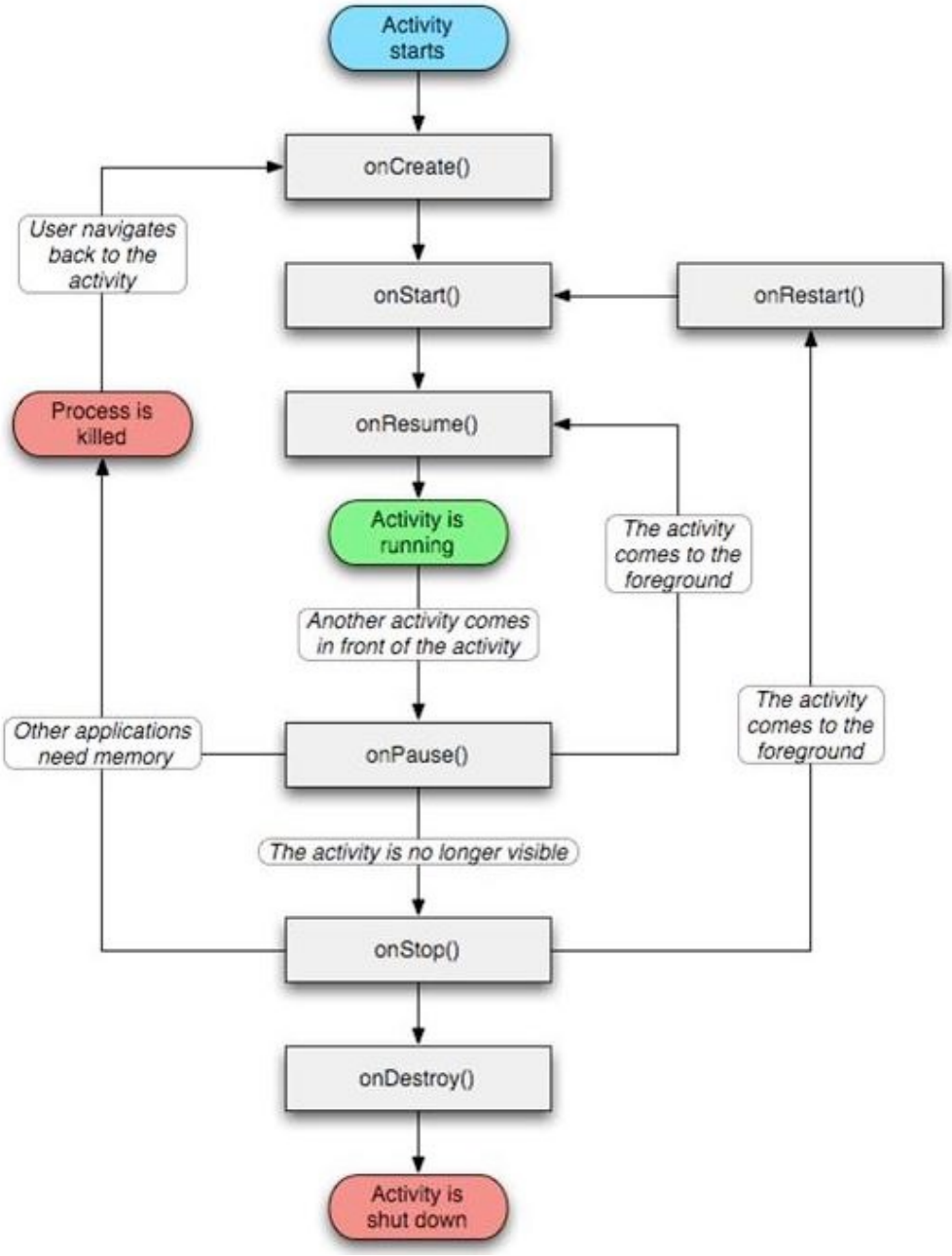
Nilanjan Banerjee

University of Maryland
Baltimore County
nilanb@umbc.edu
http://csee.umbc.edu/~nilanb/

Activities

- Typically corresponds to one UI screen
- But they can be
  - Faceless
  - A floating window
  - Return a value
- Typically a complex application will have multiple activites
  - E.g., email application
    - Activity 1: log in page
    - Activity 2: displaying a set of email
    - Transfer data between activities
      - Usually form a bundle and pass it around (we will talk about in detail)

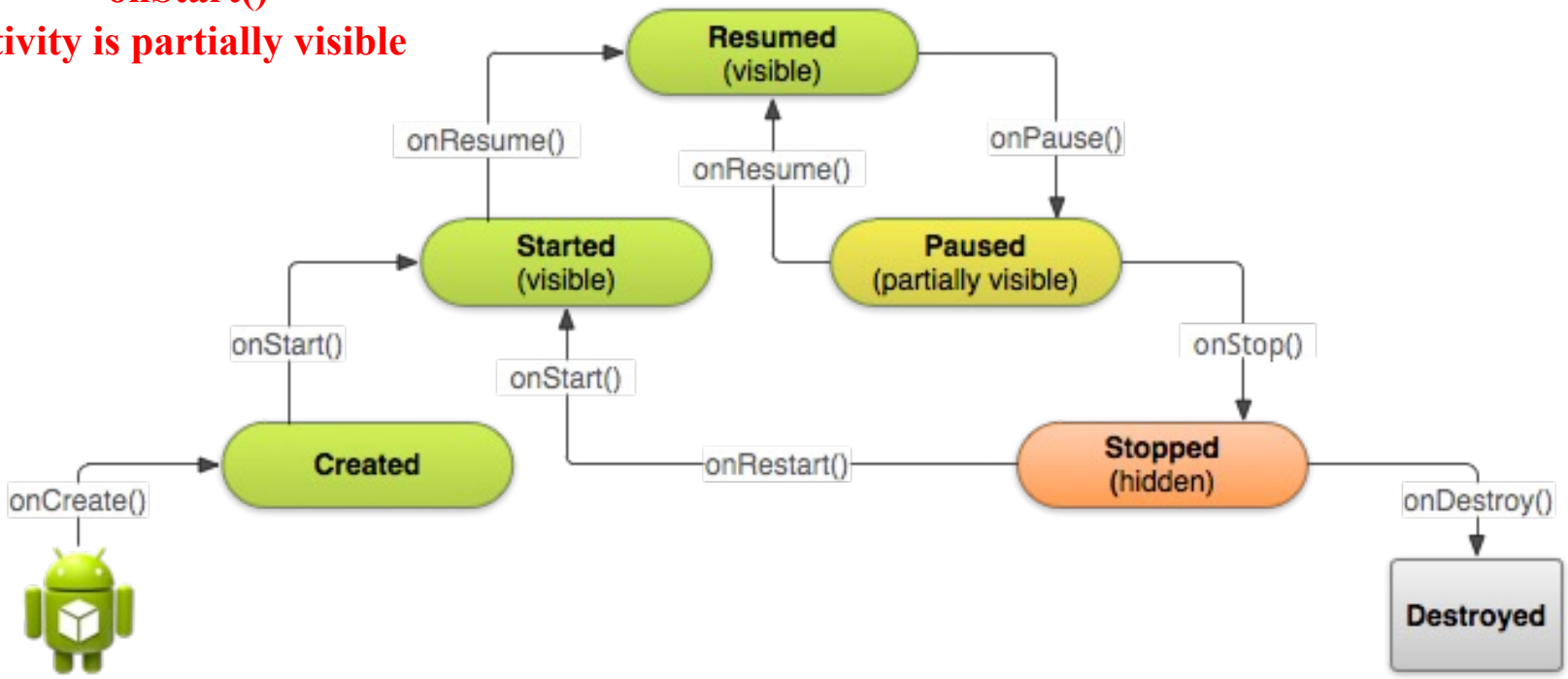
Lets delve deeper into an Activity and its lifecycle



Starting an activity

- Activity starts
  - onCreate(), onStart(), onResume() are called in succession

onStart()
Activity is partially visible

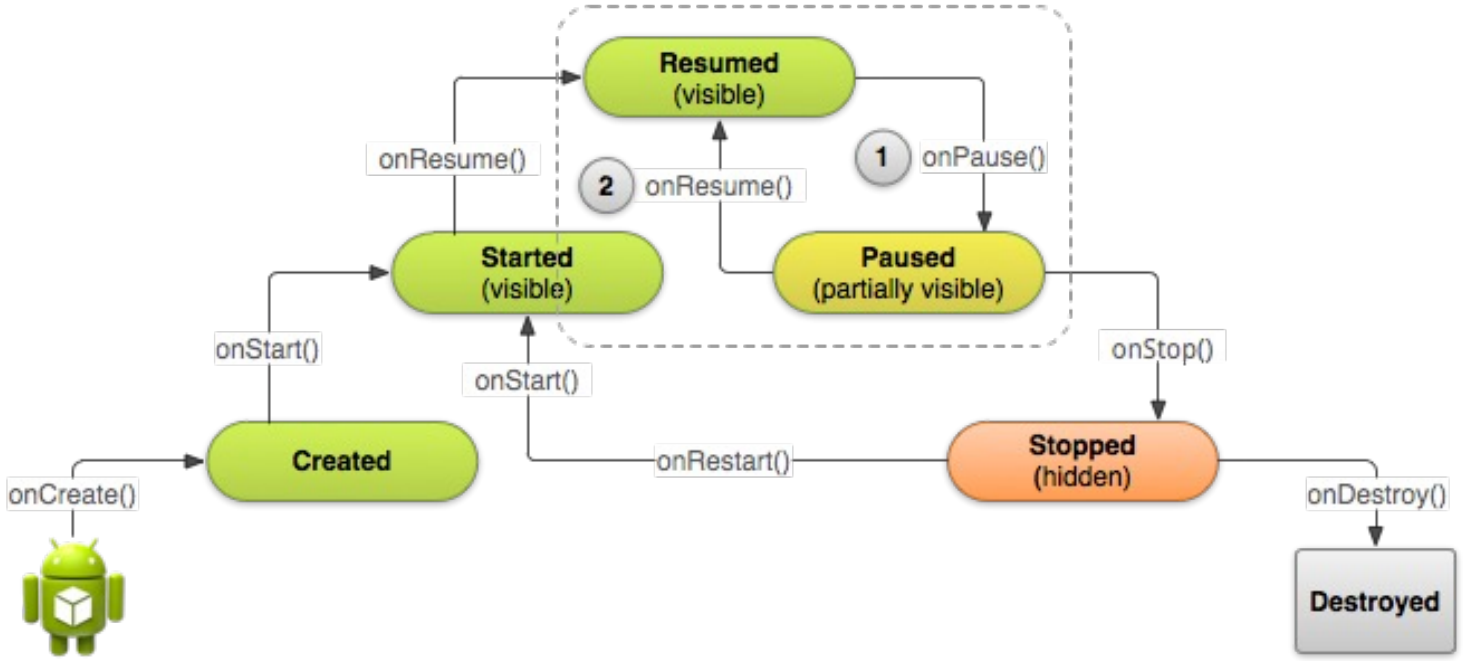


Ack: Android Development website

Pausing and Resuming an Activity

- How is pausing defined?
  - If an activity is partially visible in the background
  - Dialog boxes open up
  - Method called is onPause()

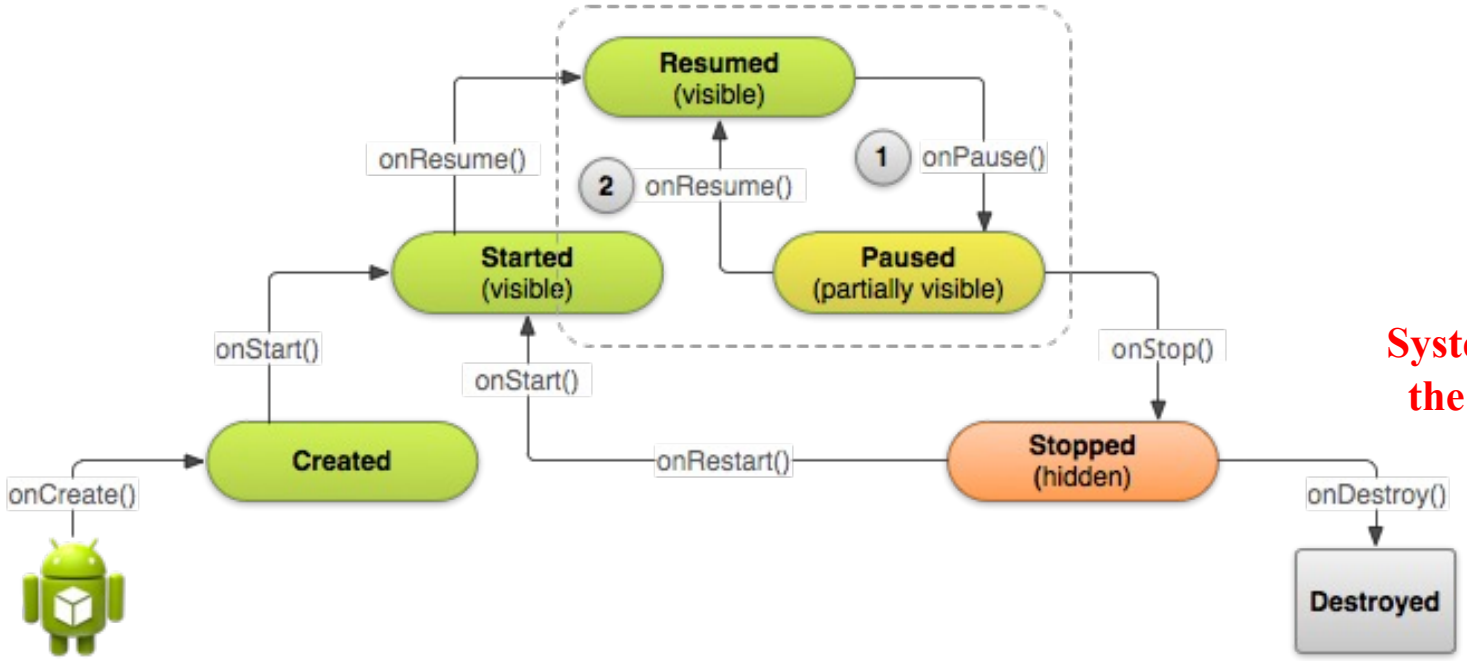
What should you do before an activity pauses
save state of the application?
release resources



Stopping and Starting an Activity

- How is stopping an activity defined?
  - If the activity is not visible
  - Press back or home button
  - Start a new activity or receive a phone call

What should you do before an activity stops
save state of the application?
release resources

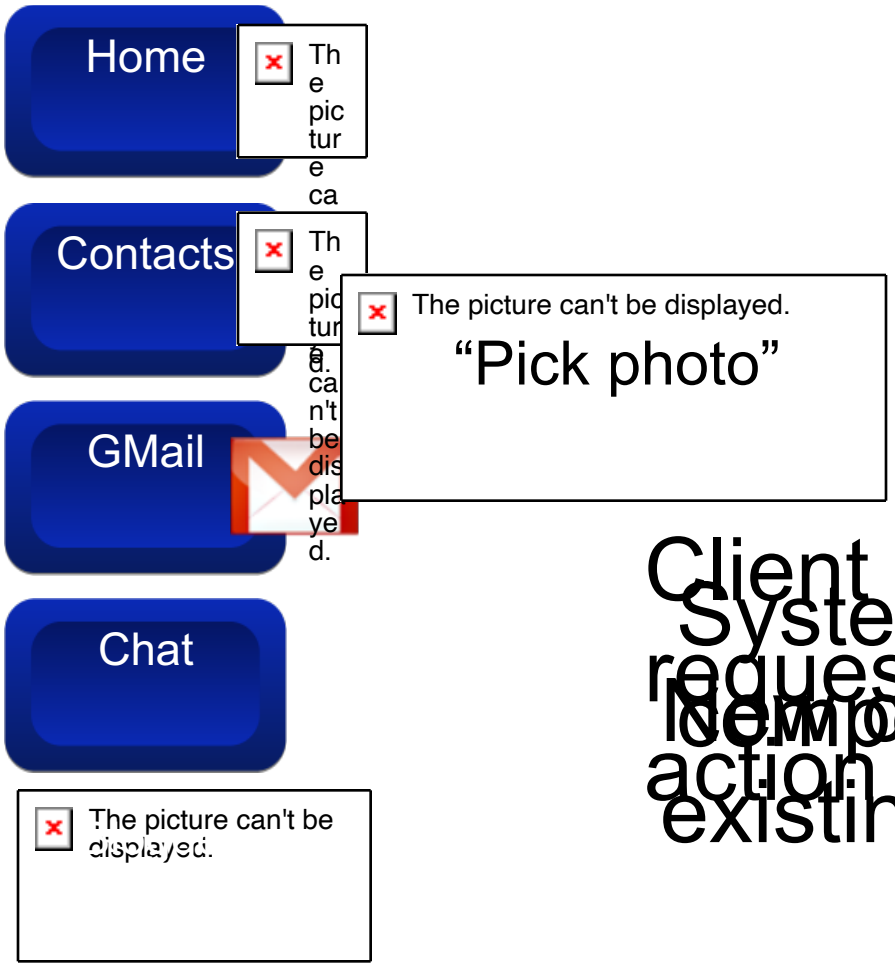


System destroys the activity

Intents

- A description of what you want done… something like a verb
  - E.g. Intent of a music player is to PLAY
- Intents are of two types – Implicit and Explicit
- Explicit
  - Application states what it needs
- Implicit
  - System decides for you which application/component can best respond to the Intent.
  - You just specify what the Intent is.

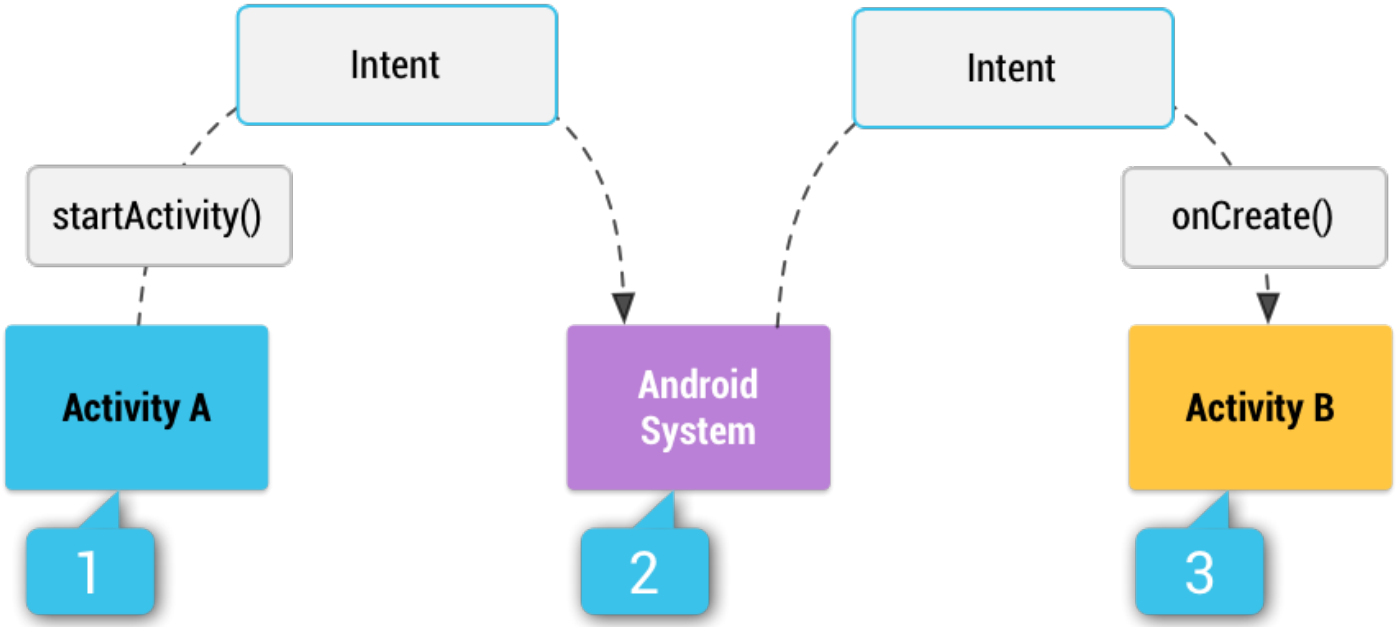
Implicit Intents



Client component makes a request for a specific action

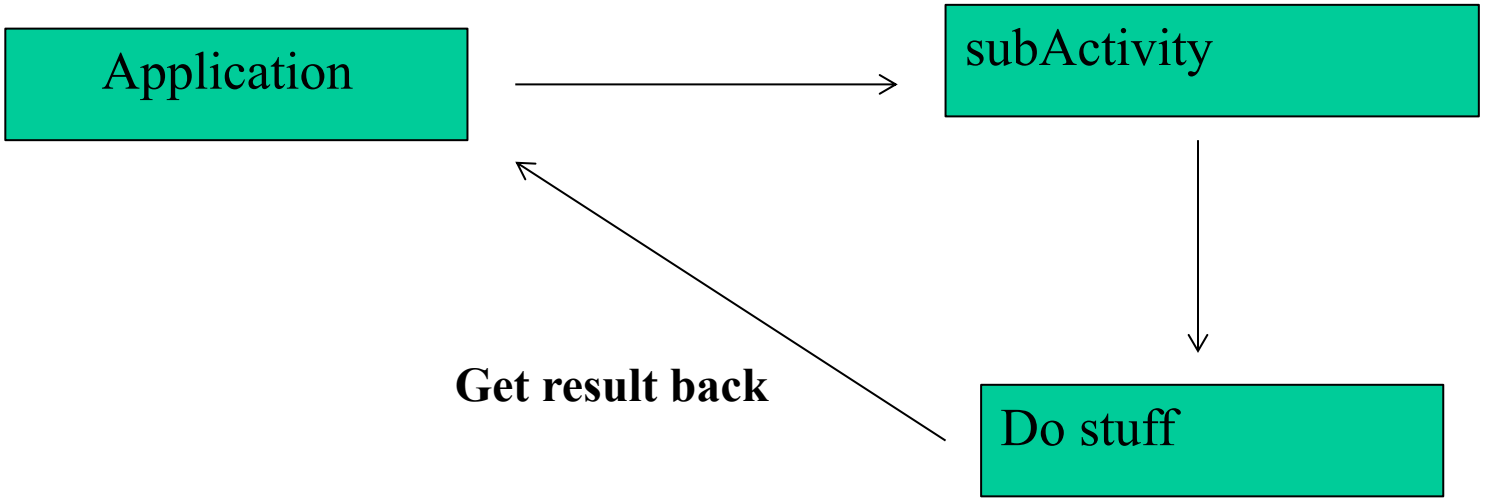
System picks best component for that action

What are the applications of Intents?



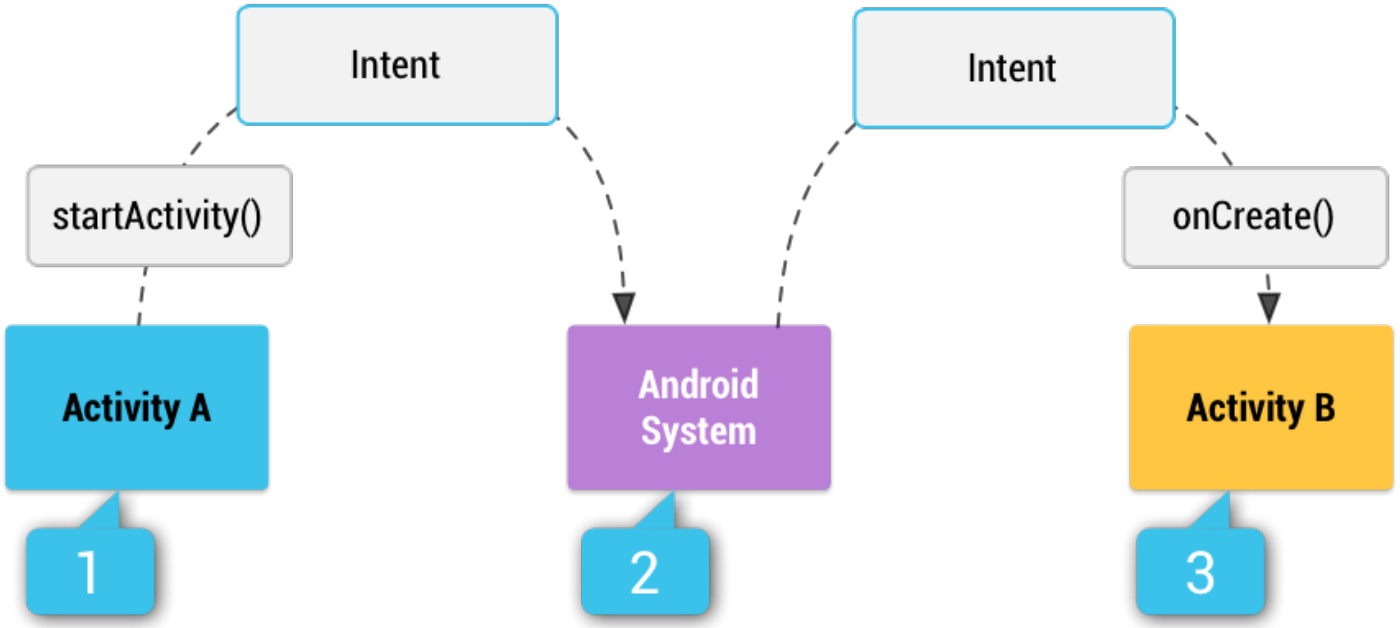
- Start a second activity from a first activity
  - Second activity might be explicitly defined in your application

In some cases you also need results back from an activity

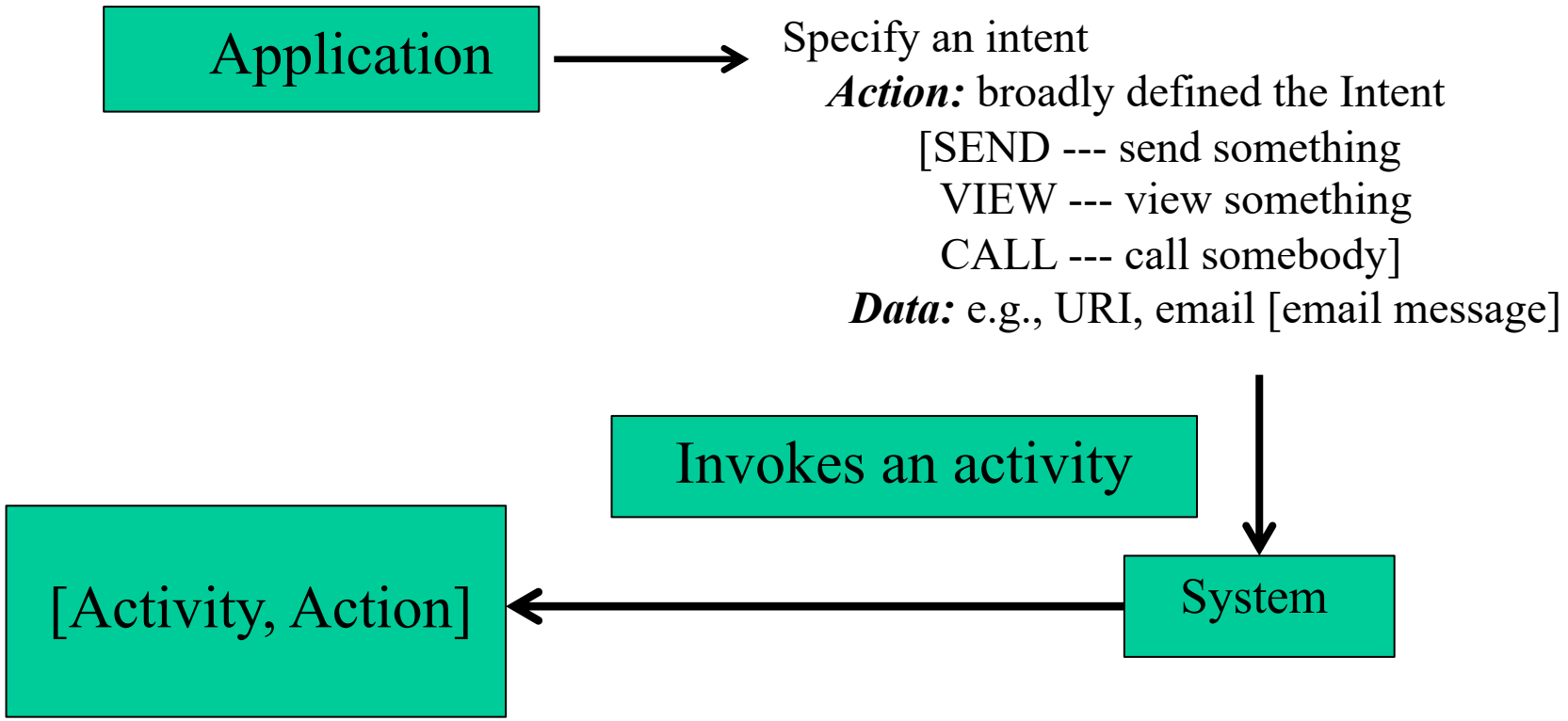


What are the applications of Intents?

- Start a second activity from a first activity
  - Second activity might be explicitly defined in your application
  - ***Second activity might be a system activity/application already available on the Android system***



How is this implemented using Intents



Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));

What does the application's manifest look like that accepts the Intent?

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

What are the applications of Intents?

- Start a second activity from a first activity
  - Second activity might be explicitly defined in your application
  - ***Second activity might be a system activity/application already available on the Android system***
- Starting a new Service from an Activity
- Delivering a broadcast

What is and is not an Android service?

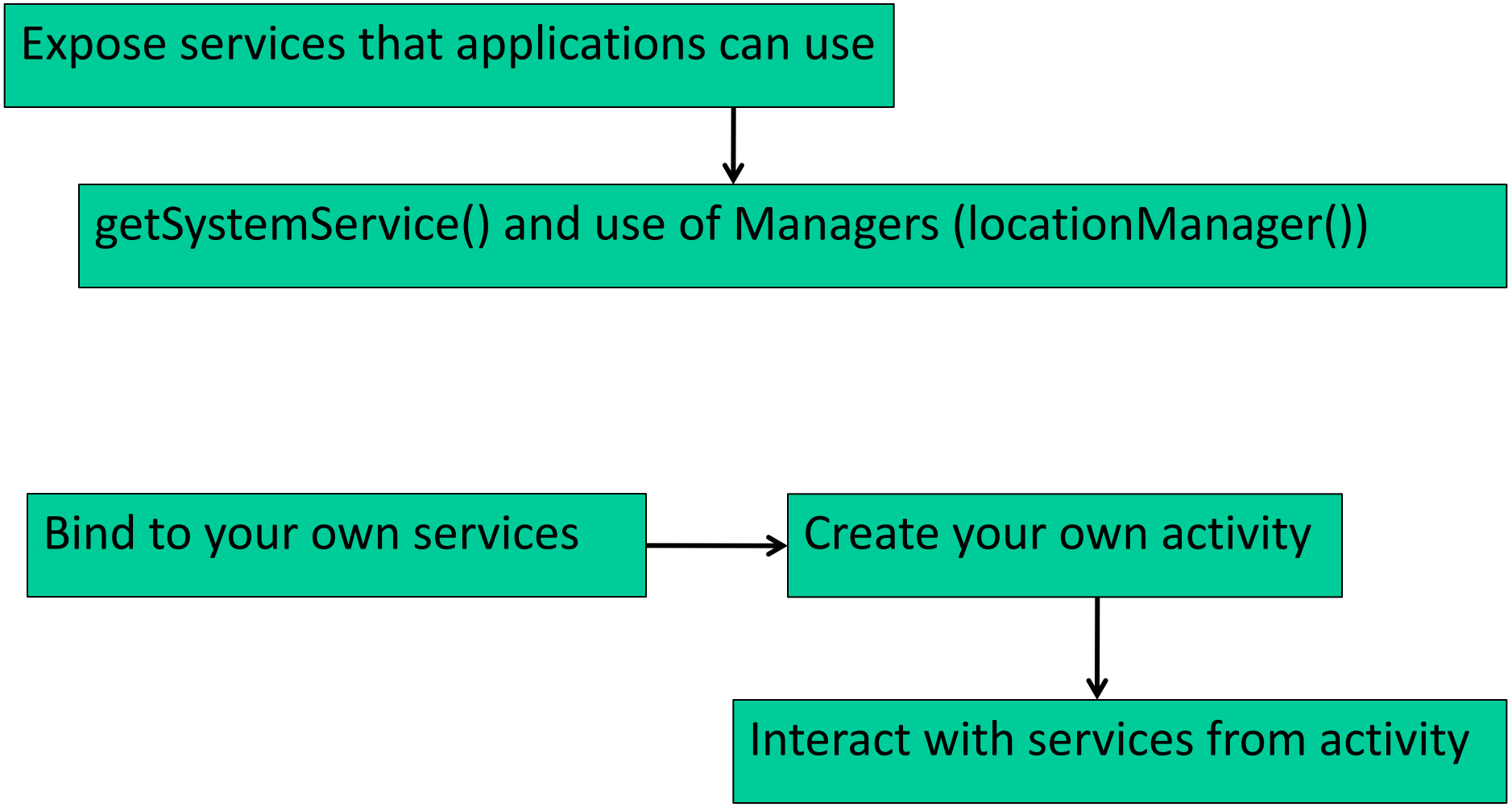
A service is **not** a separate process

A service is **not** a separate thread!

A facility for an application to tell the system about something it wants to do be doing in the background (**Context.startService()**)

A facility for an application to expose some functionality that other applications can use (**Context.bindService()**)

Uses of Android Services



Defining and consuming your own service

- An activity can start a service
  - startService() --- starts the service from an activity
  - stopService() --- stops the service from the activity
- An activity interacts with a service using bindService()
  - Requires a ServiceConnection which allows to connect to a service and which returns a Ibinder object
  - Lifecycle flow: onCreate() ---- onStartCommand()
  - **Return value of onStartCommand() START_STICKY – explicitly stopped or started, START_NOT_STICKY --- end automatically after the onStartCommand()**
  - Another implementation using a **Messenger**

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
 <activity android:name=".ServiceConsumer"    android:label="@string/app_name">
  <intent-filter>
       <action android:name="android.intent.action.MAIN" />
       <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
 </activity>
 <service android:name="WordService"></service>
</application>
```

Defining and consuming your own service

```
public class WordService extends Service
{
	private Timer timer = new Timer();
	private static final long UPDATE_INTERVAL = 5000;
	private final IBinder mBinder = new MyBinder();
	private ArrayList<String> list = new ArrayList<String>();
	private String[] fixedList = { "Linux", "Android", "iPhone", "vogella.de","helpful", "stuff" };
	private int index = 0;

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

	private void pollForUpdates()
	{
		timer.scheduleAtFixedRate(new TimerTask()
			{
				@Override
				public void run()
					{
						if (list.size() >= 6)
						    {list.remove(0);}
						list.add(fixedList[index++]);
						if (index >= fixedList.length) {index = 0;}}},
			0, UPDATE_INTERVAL);
			Log.i(getClass().getSimpleName(), "Timer started.");
	}
			@Overridepublic
			void onDestroy() {super.onDestroy()}
```

Defining and consuming your own service

```
public class WordService extends Service
{

        if (timer != null)
                {
                        timer.cancel();
                }
                Log.i(getClass().getSimpleName(), "Timer stopped.");
      }
    // We return the binder class upon a call of bindService
                @Override
                        public IBinder onBind(Intent arg0)
                           {
                                   return mBinder;
                             }

                        public class MyBinder extends Binder
                           {
                               WordService getService()
                                   {
                                           return WordService.this;
                                   }
                        }

                public List<String> getWordList()
                        {
                                   return list;
                        }
}
```

Defining and consuming your own service

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"android:orientation="vertical"
android:layout_width="fill_parent"android:layout_height="fill_parent">

  <Button android:text="Reload Data" android:id="@+id/button1"android:layout_width="wrap_content"
android:layout_height="wrap_content"android:onClick="showServiceData">
   </Button>

  <ListView android:id="@+id/list" android:layout_width="match_parent" android:layout_height="match_parent">
  </ListView>

</LinearLayout>
```

```
public class ServiceConsumer extends Activity {
     private WordService s;
     private ArrayList<String> values;
    /** Called when the activity is first created. */
    @Overridepublic
    void  onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.main);
                    doBindService();values = new ArrayList<String>();
                    adapter = new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, values);
                    ListView list = (ListView) findViewById(R.id.list);list.setAdapter(adapter);

}
```

Defining and consuming your own service

```
private ServiceConnection mConnection = new ServiceConnection()
 {
      public void onServiceConnected(ComponentName className, IBinder binder)
           {
                s = ((WordService.MyBinder) binder).getService();
                Toast.makeText(ServiceConsumer.this, "Connected",Toast.LENGTH_SHORT).show();
           }

     public void onServiceDisconnected(ComponentName className)
                 {
                           s = null;
                }
                };


     private ArrayAdapter<String> adapter;

     void doBindService()
                {
                           bindService(new Intent(this, WordService.class),
                mConnection,Context.BIND_AUTO_CREATE);
                }

     public void showServiceData(View view)
               {
                           if (s != null) {List<String> wordList = s.getWordList();
                           values.clear();
                           values.addAll(wordList);
                           adapter.notifyDataSetChanged();
                }
     }
}
```

Broadcast Receiver

- Registered as a receiver in an Android application via AndroidManifest.xml
- Service sends out Intent using sendBroadcast() method while the BroadcastReceiver defines a method called onReceive() to receive the Intent
- Example: Application which listens for changes in the phone state

```
<manifest>
  <application>
   <receiver android:name="MyPhoneReceiver">
       <intent-filter>
           <action android:name="android.intent.action.PHONE_STATE> </action>
       </intent-filter>
   </receiver>
   <uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
  </application>
</manifest>
```

Broadcast Receiver: Phone state example

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
mport android.os.Bundle;
import android.telephony.TelephonyManager;
import android.util.Log;


public class MyPhoneReceiver extends BroadcastReceiver
{
 @Overridepublic
 void onReceive(Context context, Intent intent)
  {
                Bundle extras = intent.getExtras();
                if (extras != null)
                    {
                       String state = extras.getString(TelephonyManager.EXTRA_STATE);
                       Log.w("DEBUG", state);
                       if (state.equals(TelephonyManager.EXTRA_STATE_RINGING))
                                {
                                   String phoneNumber = extras.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
                                   Log.w("DEBUG", phoneNumber);
                                }
                     }
   }
}
```

Pending Intent

- Token that an application gives another application which allows the other application to use permissions of your application to execute code
  - Other application could be Notification Manager, Alarm manager etc.

- Example of a pending Intent and BroadcastReceiver.

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"android:orientation="vertical"
android:layout_width="fill_parent"android:layout_height="fill_parent">

<EditText android:layout_height="wrap_content" android:id="@+id/time"android:layout_width="wrap_content"
android:hint="Number of seconds"android:inputType="numberDecimal">>
</EditText>

<Button android:text="Start Counter" android:id="@+id/ok"android:onClick="startAlert"
android:layout_width="wrap_content"android:layout_height="wrap_content">
</Button>

</LinearLayout>
```

System Services and Broadcast Receiver

```
import android.content.BroadcastReceiver;
import android.content.Context;I
mport android.content.Intent;
import android.os.Vibrator;
import android.widget.Toast;

public class MyBroadcastReceiver extends BroadcastReceiver
{
@Override
public void onReceive(Context context, Intent intent)
{
  Toast.makeText(context, "Don't panik but your time is up!!!!.",Toast.LENGTH_LONG).show();// Vibrate the mobile phone
   Vibrator vibrator = (Vibrator) context.getSystemService(Context.VIBRATOR_SERVICE);
   vibrator.vibrate(2000);
  }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"package=“com.android.alarm"
android:versionCode="1"android:versionName="1.0">
<uses-sdk android:minSdkVersion="9" />
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".AlarmActivity" android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<receiver android:name="MyBroadcastReceiver"></receiver></application>
<uses-permission android:name="android.permission.VIBRATE"></uses-permission>
</manifest>
```

System Services and Broadcast Receiver

```
import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class AlarmActivity extends Activity {/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
}

public void startAlert(View view)
 {
    EditText text = (EditText) findViewById(R.id.time);
    int i = Integer.parseInt(text.getText().toString());
    Intent intent = new Intent(this, MyBroadcastReceiver.class);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(this.getApplicationContext(), 234324243, intent, 0);

    AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
    alarmManager.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis()+ (i * 1000), pendingIntent);
    Toast.makeText(this, "Alarm set in " + i + " seconds",Toast.LENGTH_LONG).show();
 }

}
```

CMSC 628: Introduction to Mobile Computing
Basics of Android programming: Activity, Intents, Services, BroadcastReceivers

Nilanjan Banerjee

University of Maryland
Baltimore County
nilanb@umbc.edu
http://csee.umbc.edu/~nilanb/