Lec #15: Telephony and SMS
Objective

• Telephony

• SMS
  ➤ Using Intents to send SMS and MMS messages
  ➤ Using the SMS Manager to send SMS Messages
  ➤ Handling incoming SMS messages

• Presentation

  - Calling All Cars: Cell Phone Networks and the Future of Traffic
    • Presenter: Jeremiah Dunn
  
  - The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring
    • Presenter: Timothy Werner
SMS and MMS
Overview

• SMS sends short text messages between mobile phones.
  ➤ Supports sending both text messages and data messages

• MMS (multimedia messaging service) messages have allowed users to send and receive messages that include multimedia attachments such as photos, videos, and audio.

• Using the **SMSManager**, you can replace the native SMS application to send text messages, react to incoming texts, or use SMS as a data transport layer.

• Use the **SEND** and **SEND_TO** actions in Intents to send both SMS and MMS messages using a messaging application installed on the device.
Sending SMS/MMS thru Native App

• Use Intent with Intent.ACTION_SENDTO action:
  ➢ Specify a target number using sms: schema notation as the Intent data.
  ➢ Include the message you want to send within the Intent payload using an sms_body extra.

```java
Intent smsIntent = new Intent(Intent.ACTION_SENDTO, Uri.parse("sms:55512345"));
smsIntent.putExtra("sms_body", "Press send to send me");
startActivity(smsIntent);
```
Sending SMS/MMS thru Native App

- You can also attach files (effectively creating an MMS message) to your messages
  - Add an `Intent.EXTRA_STREAM` with the URI of the resource to attach.
  - Set the Intent `type` to the `mime-type` of the attached resource.
  - Use `ACTION_SEND` and include the target phone number as an address extra

```java
// Get the URI of a piece of media to attach.
Uri attached_Uri = Uri.parse("content://media/external/images/media/1");

// Create a new MMS intent
Intent mmsIntent = new Intent(Intent.ACTION_SEND, attached_Uri);
mmsIntent.putExtra("sms_body", "Please see the attached image");
mmsIntent.putExtra("address", "07912355432");
mmsIntent.putExtra(Intent.EXTRA_STREAM, attached_Uri);
mmsIntent.setType("image/png");
startActivity(mmsIntent);
```
Sending SMS Manually

- SMS messaging in Android is handled by the `SmsManager`.
  
  ```java
  SmsManager smsManager = SmsManager.getDefault();
  ```

- Specify the `SEND_SMS` uses-permission.
  
  ```xml
  <uses-permission android:name="android.permission.SEND_SMS"/>
  ```

- Use `sendTextMessage` from the SMS Manager, passing in the address (phone number) of your recipient and the text message you want to send,
  
  ```java
  String sendTo = "5551234";
  String myMessage = "Android supports programmatic SMS messaging!";
  smsManager.sendTextMessage(sendTo, null, myMessage, null, null);
  ```

**SMS Manager Reference:**
Tracking and Confirming SMS Delivery

• The final two parameters in `sendTextMessage` let you specify Intents to track the transmission and delivery.

• Implement and register corresponding Broadcast Receivers that listen for the actions you specify when creating the Pending Intents you pass in `sendTextMessage`.

• Intent parameter, `sentIntent`, is fired when the message either is successfully sent or fails to send.
  ➢ Activity.RESULT_OK
  ➢ SmsManager.RESULT_ERROR_GENERIC_FAILURE
  ➢ SmsManager.RESULT_ERROR_RADIO_OFF
  ➢ SmsManager.RESULT_ERROR_NULL_PDU

• The second Intent parameter, `deliveryIntent`, is fired only after the destination recipient receives your SMS message.
SMS delivery monitoring pattern

String SENT_SMS_ACTION = "SENT_SMS_ACTION";
String DELIVERED_SMS_ACTION = "DELIVERED_SMS_ACTION";

// Create the sentIntent parameter
Intent sentIntent = new Intent(SENT_SMS_ACTION);
PendingIntent sentPI = PendingIntent.getBroadcast(getActivityContext(),
0, sentIntent, 0);

// Create the deliveryIntent parameter
Intent deliveryIntent = new Intent(DELIVERED_SMS_ACTION);
PendingIntent deliverPI = PendingIntent.getBroadcast(getActivityContext(),
0, deliveryIntent, 0);

// Register the Broadcast Receivers
registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context _context, Intent _intent) {
    switch (getResultCode()) {
        case Activity.RESULT_OK:
            [... send success actions ...]; break;
        case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
            [... generic failure actions ...]; break;
    }
SMS delivery monitoring pattern

case SmsManager.RESULT_ERROR_RADIO_OFF:
    [. . . radio off failure actions . . . ]; break;
case SmsManager.RESULT_ERROR_NULL_PDU:
    [. . . null PDU failure actions . . . ]; break;
}

registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context _context, Intent _intent)
    {
        [. . . SMS delivered actions . . . ]
    }
},
    new IntentFilter(SENT_SMS_ACTION));

// Send the message
smsManager.sendTextMessage(sendTo, null, myMessage, sentPI, deliverPI);
Large SMS Messages

• SMS text messages are normally limited to 160 characters.
• Longer messages need to be broken into a series of smaller parts.
  ➤ **divideMessage** method accepts a string as an input and breaks it into an Array List of messages
  ➤ use the **sendMultipartTextMessage** method on the SMS Manager to transmit the array of messages
  ➤ The **sentIntent** and **deliveryIntent** parameters in the 
    **sendMultipartTextMessage** method are Array Lists that is used to specify different Pending Intents to fire for each message part.

```java
ArrayList<String> messageArray = smsManager.divideMessage(myMessage);
ArrayList<PendingIntent> sentIntents = new ArrayList<PendingIntent>();
for (int i = 0; i < messageArray.size(); i++)
    sentIntents.add(sentPI);

smsManager.sendMultipartTextMessage(sendTo, null,
    messageArray, sentIntents, null);
```
Handling Incoming SMS Messages

• With received SMS, new broadcast Intent is fired with the “android.provider.Telephony.SMS_RECEIVED” action.

• Specify the RECEIVE_SMS manifest permission.

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

• Use the pdu extras key to extract an array of SMS PDUs each of which represents an SMS message

• Call SmsMessage.createFromPdu to convert each PDU byte array into an SMS Message object

```java
Bundle bundle = intent.getExtras();
if (bundle != null) {
    Object[] pdus = (Object[]) bundle.get("pdus");
    SmsMessage[] messages = new SmsMessage[pdus.length];
    for (int i = 0; i < pdus.length; i++)
        messages[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
}
```
Example of Incoming SMS Messages

- Register the Broadcast Receiver using an Intent Filter that listens for the `android.provider.Telephony.SMS_RECEIVED` action String

```java
final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
IntentFilter filter = new IntentFilter(SMS_RECEIVED);
BroadcastReceiver receiver = new IncomingSMSReceiver(); //defined below
registerReceiver(receiver, filter);
```

- Broadcast Receiver implementation whose `onReceive` handler checks incoming SMS texts that start with the string `@echo`, and then sends the same text back to the number that sent it.

```java
public class IncomingSMSReceiver extends BroadcastReceiver {
    private static final String queryString = "@echo";
    private static final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
```
Example of Incoming SMS Messages

```java
public void onReceive(Context _context, Intent _intent) {
    if (_intent.getAction().equals(SMS_RECEIVED)) {
        SmsManager sms = SmsManager.getDefault;
        Bundle bundle = _intent.getExtras();
        if (bundle != null) {
            Object[] pdus = (Object[]) bundle.get("pdus");
            SmsMessage[] messages = new SmsMessage[pdus.length];
            for (int i = 0; i < pdus.length; i++)
                messages[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
            for (SmsMessage message : messages) {
                String msg = message.getMessageBody();
                String to = message.getOriginatingAddress();
                if (msg.toLowerCase().startsWith(queryString)) {
                    String out = msg.substring(queryString.length());
                    sms.sendTextMessage(to, null, out, null, null);
                }
            }
        }
    }
}
```
Simulating Incoming SMS Messages/Calls

- Use the Android debug tools to simulate incoming SMS messages or calls from arbitrary numbers.

![Emulator Control Interface](image.png)
Questions?
To DO

• Examples –
  • Emergency Responder SMS Example
  • Automating the Emergency Responder
  • Example Link: http://www.cs.odu.edu/~cs495/materials/Lec-15_SMS_Examples.pdf

• Nokia Mobile Data Challenge 2012: