Bluetooth

Programming the Android Platform

BlueTooth

- Wireless networking technology
  - Limited broadcast range (< 10 meters)
- Example uses include
  - Connecting headsets to phones, keyboard/mouse to desktop computer, medical devices to controllers, etc.

Bluetooth (cont.)

- Some key classes:
- BluetoothAdapter - The local Bluetooth radio
- BluetoothDevice - A remote Bluetooth device
- BluetoothServerSocket - An open server socket that listens for incoming requests
- BluetoothSocket - Communication endpoint

Bluetooth Permissions

- android.permission.BLUETOOTH
  - Allow connections & data transfer
- android.permission.BLUETOOTH_ADMIN
  - Turn on Bluetooth radio & discover local devices

Bluetooth Usage

- Set up Bluetooth
- Find devices already paired or available in the local area
- Connect devices via a BluetoothSocket
- Transfer data

BluetoothAdapter

- Get BluetoothAdapter instance with
  - BluetoothAdapter.getDefaultAdapter();
    - Returns null if Bluetooth is not available
- Check whether Bluethooth is enabled with
  - BluetoothAdapter.isEnabled()
    - Returns true if Bluetooth is on; false otherwise

BluetoothAdapter (cont.)

- Can use built-in Intent to enable Bluetooth
  - BluetoothAdapter.ACTION_REQUEST_ENABLE
- Presents users a dialog asking whether they give permission for Bluetooth to be enabled

Get BluetoothAdapter

```
public class DataTransferActivity extends Activity {
 ...
 public void onCreate(Bundle savedInstanceState) {
  ...
  mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
  if (mBluetoothAdapter == null) {
   // No Bluetooth support
   finish();
  }
  ...
```

Enable Bluetooth

```
...
if (!mBluetoothAdapter.isEnabled()) {
  Intent enableBluetoothIntent =
        new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
  startActivityForResult(
                        enableBluetoothIntent,REQUEST_ENABLE_BT);
}
...
```

Finding Devices

- Paired devices are devices that have already consented to interact with the local device
  - BluetoothAdapter.getBondedDevices();
- Can discover new devices with
  - BluetoothAdapter.startDiscovery();
  - Remote devices must be discoverable
    - Must go through pairing process
- Can stop discovery process with
  - BluetoothAdapter.cancelDiscovery()

Find & Select Device

```
private void selectServer() {
 Set<BluetoothDevice> pairedDevices =
                                    mBluetoothAdapter.getBondedDevices();
 ArrayList<String> pairedDeviceStrings = new ArrayList<String>();
 if (pairedDevices.size() > 0) {
  for (BluetoothDevice device : pairedDevices) {
   pairedDeviceStrings.add(device.getName() + "\n" + device.getAddress());
  }
 }
 Intent showDevicesIntent = new Intent(this, ShowDevices.class);
 showDevicesIntent.putStringArrayListExtra("devices", pairedDeviceStrings);
 startActivityForResult(showDevicesIntent, SELECT_SERVER);
}
```

Find & Select Device (cont.)

```
public class ShowDevices extends ListActivity {
 …
 protected void onCreate(Bundle savedInstanceState) {
  …
  // search for more devices
  mBluetoothAdapter.startDiscovery();
  …
   // user selects one device
  BluetoothDevice device =
                    mBluetoothAdapter.getRemoteDevice(/* mac addr String */);
  Intent data = new Intent();
  data.putExtra(BluetoothDevice.EXTRA_DEVICE, device);
  setResult(RESULT_OK, data);
  finish();
  …
```

Device Discovery

- Intent broadcast on device discovery
  - BluetoothDevice.ACTION_FOUND
- Device info contained in BluetoothDevice instance
  - intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE)

Device Discovery (cont.)

```
final BroadcastReceiver mReceiver = new BroadcastReceiver() {
  public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (BluetoothDevice.ACTION_FOUND.equals(action)) {
      BluetoothDevice device = intent
                        .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
      // update display
    }
  }
};
```

Connecting Devices

- Client & server need the same UUID
  - 128-bit Universally Unique Identifier
  - Many freely-available generators exist
- Client creates

BluetoothServerSocket

- Server creates BluetoothServerSocket with
  - BluetoothAdapter
    .listenUsingRfcommWithServiceRecord(name, uuid)
- Listen for incoming connection with BluetoothServerSocket.accept()
  - Returns a BluetoothSocket on connection

BluetoothServerSocket

```
class AcceptThread extends Thread {
 public AcceptThread(Handler handler) {
   …
   try {
     mServerSocket = mBluetoothAdapter
          .listenUsingRfcommWithServiceRecord("Bluetooth Demo",
                                              DataTransferActivity.APP_UUID);
   } catch (IOException e) {}
 }
 …
```

BluetoothServerSocket (cont.)

```
…
public void run() {
  while (true) {
    try {
      mBluetoothSocket = mServerSocket.accept();
      manageConnectedSocket();
      mServerSocket.close();
       break;
    } catch (IOException e1) { … }
  }
}
…
```

BluetoothSocket

- Client creates a BluetoothSocket with
  - BluetoothDevice
    .createRfcommSocketToServiceRecord(uuid)
- Make connection on BluetoothSocket with
  - BluetoothSocket.connect()

BluetoothSocket (cont.)

```
public class ConnectThread extends Thread {
 …
 public ConnectThread(String deviceID, Handler handler) {
  mDevice = mBluetoothAdapter.getRemoteDevice(deviceID);
  try {
    mBluetoothSocket = mDevice
     .createRfcommSocketToServiceRecord(DataTransferActivity.APP_UUID);
   } catch (IOException e) { … }
 }
 …
```

BluetoothSocket (cont.)

```
public void run() {
  mBluetoothAdapter.cancelDiscovery();
  try {
    mBluetoothSocket.connect();
    manageConnectedSocket();
  } catch (IOException connectException) { … }
}
```

Data Transfer

```
private void manageConnectedSocket() {
  ConnectionThread conn =
     new ConnectionThread(mBluetoothSocket, mHandler);
  mHandler.obtainMessage(
     DataTransferActivity.SOCKET_CONNECTED, conn)
                                                  .sendToTarget();
  conn.start();
}
```

Data Transfer (cont.)

```
public class ConnectionThread extends Thread {
 …
 ConnectionThread(BluetoothSocket socket, Handler handler){
  super();
  mBluetoothSocket = socket;
  mHandler = handler;
  try {
   mInStream = mBluetoothSocket.getInputStream();
   mOutStream = mBluetoothSocket.getOutputStream();
  } catch (IOException e) { … }
}
…
```

Data Transfer (cont.)

```
...
public void run() {
  byte[] buffer = new byte[1024];
  int bytes;
  while (true) {
    try {
      bytes = mInStream.read(buffer);
      String data = new String(buffer, 0, bytes);
      mHandler.obtainMessage(
            DataTransferActivity.DATA_RECEIVED,data).sendToTarget();
    } catch (IOException e) { break;  }
  }
}
...
```

Data Transfer (cont.)

```
  …
  public void write(byte[] bytes) {
    try {
      mOutStream.write(bytes);
    } catch (IOException e) { … }
  }
}
```

Data Transfer (cont.)

```
// Handler in DataTransferActivity
 public Handler mHandler = new Handler() {
  public void handleMessage(Message msg) {
   switch (msg.what) {
    case SOCKET_CONNECTED: {
     mBluetoothConnection = (ConnectionThread) msg.obj;
     if (!mServerMode)
      mBluetoothConnection.write("this is a message".getBytes());
     break;
    }
    case DATA_RECEIVED: {
     data = (String) msg.obj;
     tv.setText(data);
     if (mServerMode)
      mBluetoothConnection.write(data.getBytes());
  }
 …
```

Lab Assignment

Source Code Examples

- BluetoothSetupAndTransferData