MMI 2: Mobile Human-Computer Interaction

Übung 5

Prof. Dr. Michael Rohs

michael.rohs@ifi.lmu.de

Mobile Interaction Lab, LMU München

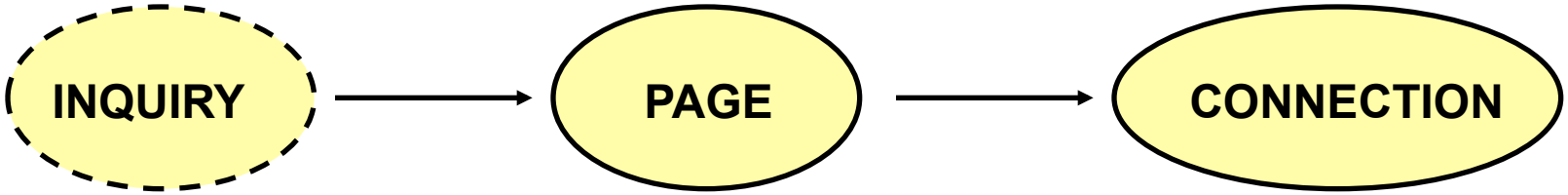
Aufgabe 1: Drahtlose Kommunikation

a) Mit Bluetooth können sowohl asynchrone wie auch synchrone Kommunikationskanäle aufgebaut werden. Wodurch sind diese beiden Kommunikationstypen charakterisiert? In welchen Anwendungsfällen sollten sie jeweils eingesetzt werden?

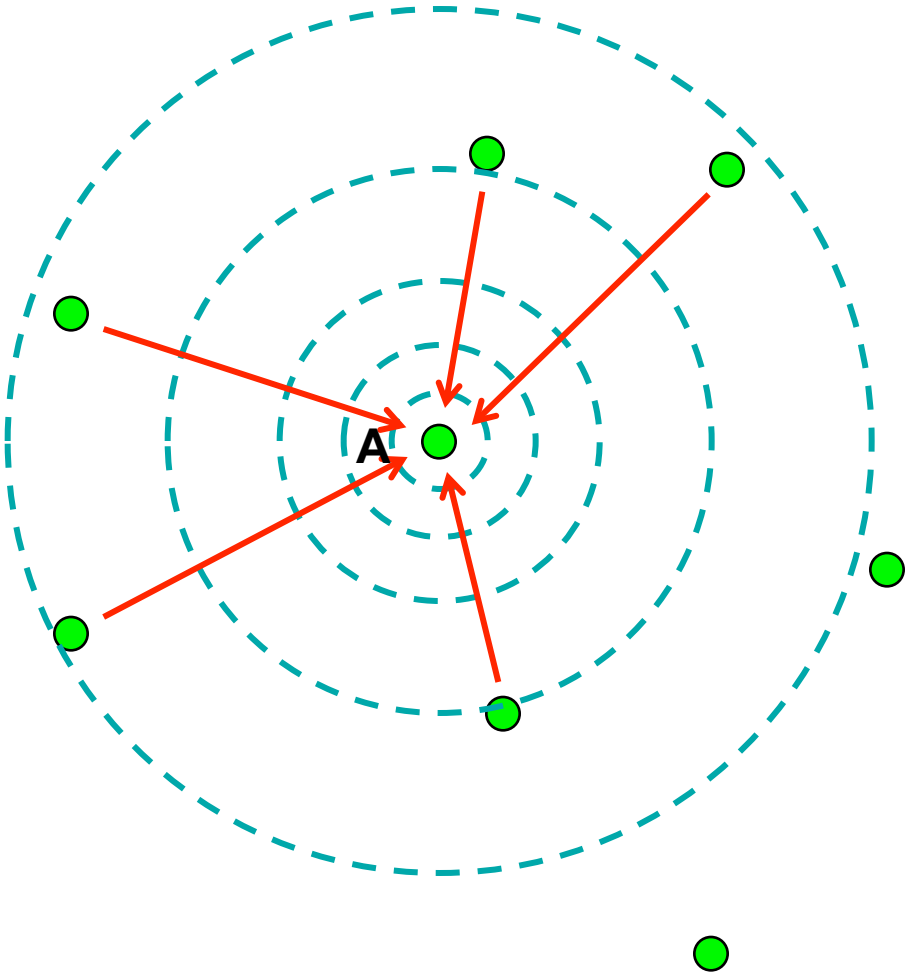
b) Wie lassen sich die auf einem Gerät befindlichen Bluetooth-Dienste ausfindig machen? Was muss bekannt sein, um eine Verbindung zu einem Dienst aufzubauen?

c) Wieso definiert WLAN keinen Mechanismus, um WLAN-Dienste ausfindig zu machen?

Connection Establishment

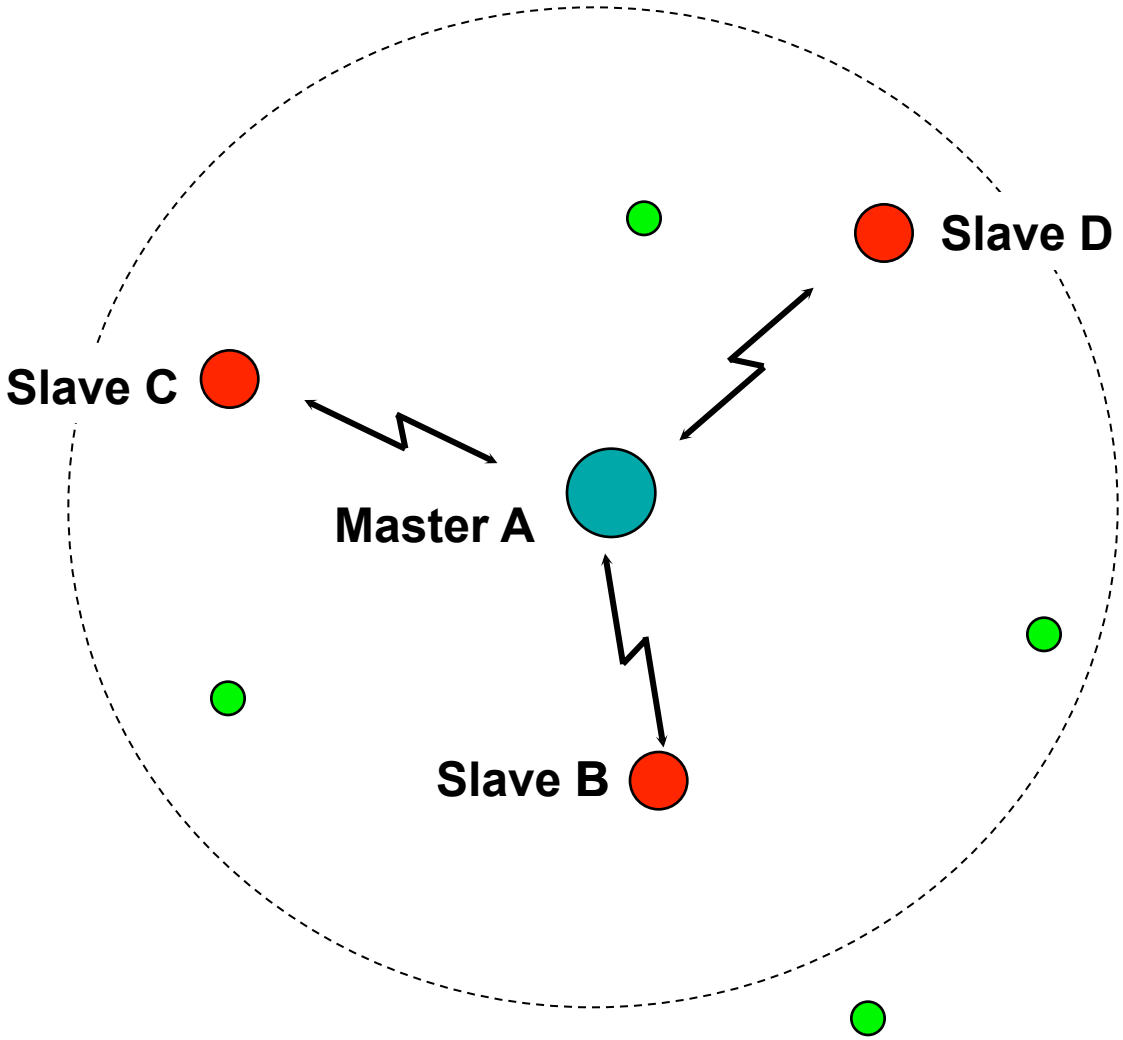


Inquiry: Looking for Nearby Devices



- Responses include:
  - Device Address
  - Class of Device

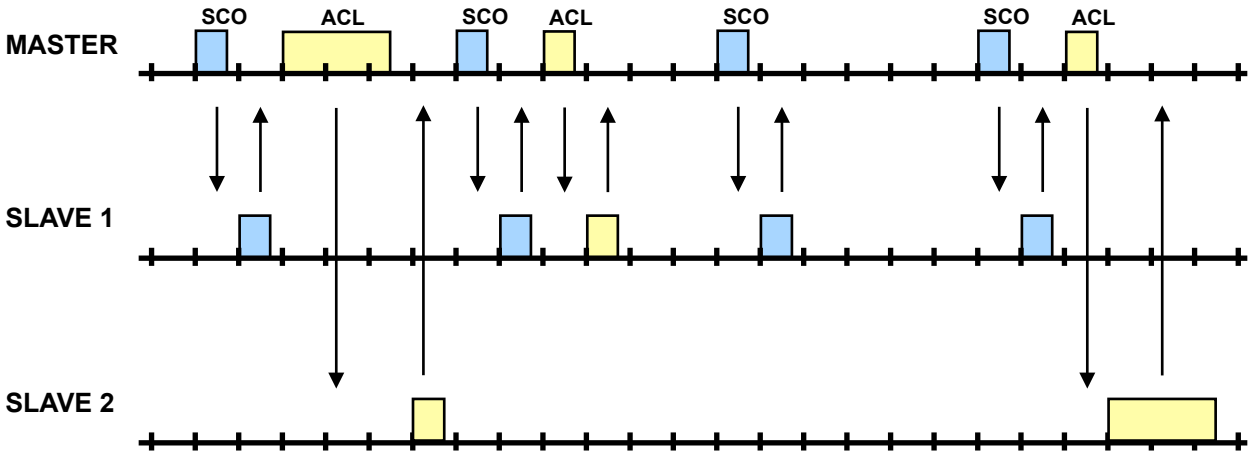
Paging: Establishing a Connection



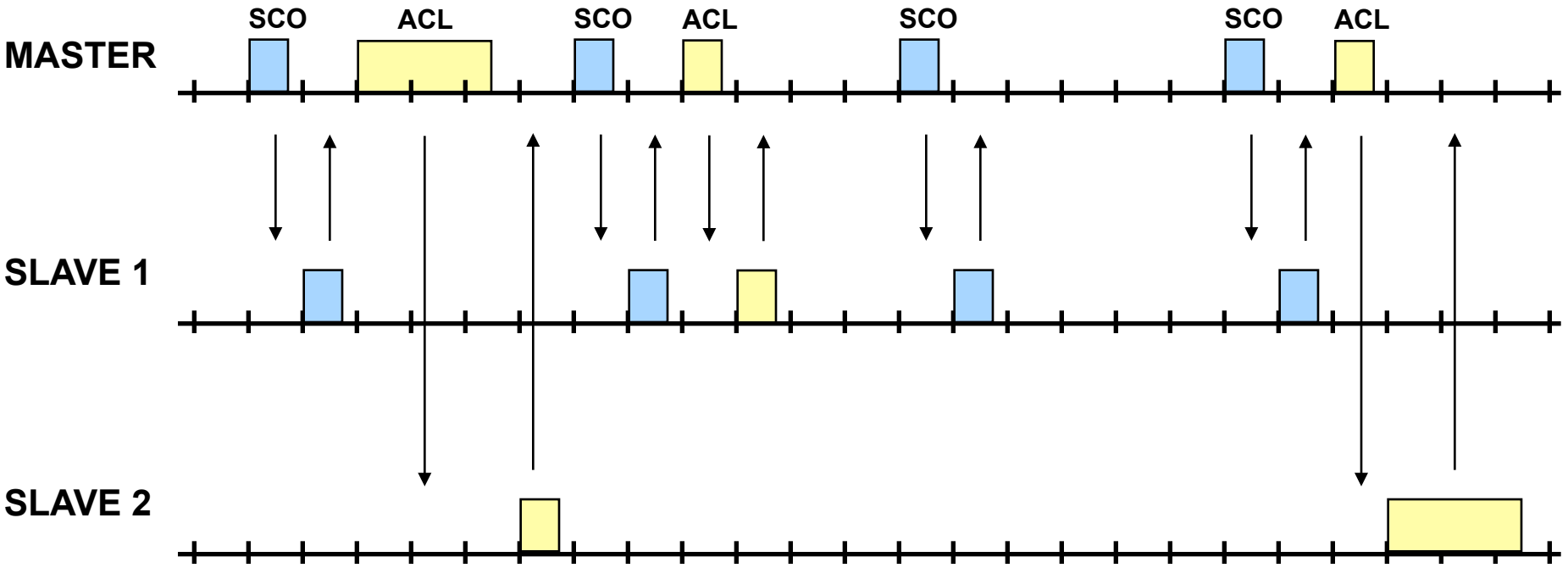
- Done for each device independently
- Paging device becomes master

Baseband Link Types

- Synchronous connection-oriented (SCO) link
  - "Circuit-switched": periodic slot assignment
  - Typically for voice
- Asynchronous connection-less (ACL) link
  - "Packet switching" with acks
  - Variable packet size (1-5 slots)



Mixed Synchronous / Asynchronous Communication



SCO: Synchronous Connection-Oriented link

- uses reserved time slots

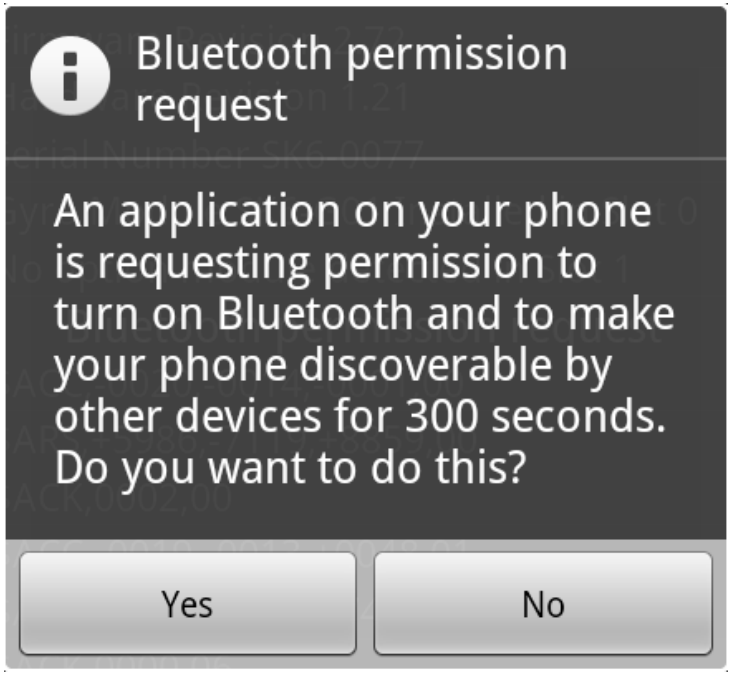
ACL: Asynchronous Connection-Less link

- uses remaining time slots

Make Yourself Discoverable

```
Intent intent = new
    Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
intent.putExtra(
    BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);

startActivity(intent);
```



Discover Other Devices

- Asynchronously, multiple seconds
- Register “Broadcast Receiver” for discovery events

```
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(deviceFoundReceiver, filter);

filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
registerReceiver(deviceFoundReceiver, filter);

filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
registerReceiver(deviceFoundReceiver, filter);


adapter.startDiscovery();
```

Broadcast Receiver for Discovery Events

```
BroadcastReceiver deviceFoundReceiver = new BroadcastReceiver() {
public void onReceive(Context context, Intent intent) {
String a = intent.getAction();
if (BluetoothDevice.ACTION_FOUND.equals(a)) {
      BluetoothDevice device;
      device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
      listAdapter.add(device.getName() + ", " + device.getAddress());
} else if (BluetoothAdapter.ACTION_DISCOVERY_STARTED.equals(a)) {
    listAdapter.add("discovery started");
} else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(a)) {
    listAdapter.add("discovery finished");
} } };
```

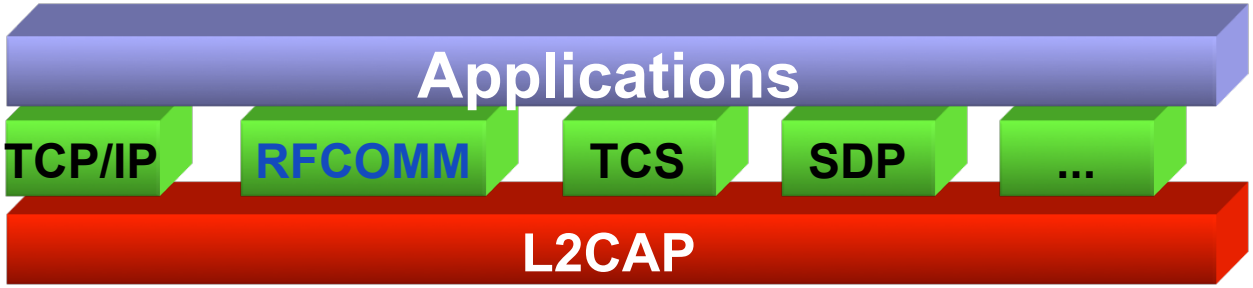
Service Discovery Protocol (SDP)

- Devices may spontaneously join / leave a network
  - Goal: self configure without manual intervention
  - Devices should discover each other, negotiate their needs
- SDP defines an inquiry/response protocol for discovering services
  - Searching for services
  - Browsing services
- SDP has no notification service
  - No automatic notification of new services or services becoming unavailable

SDP: Service Description

- Each service is represented by a service record
- Attributes in the service record describe the service
- Attributes represent
  - Unique identifier
  - Service class information (e.g., “printer” or “audio service”)
  - Access protocol information
  - Human-readable service description
- Attribute values
  - Universally unique identifiers (UUIDs), strings, booleans, integers, URLs, etc.
- 128-bit UUID example (Serial Port Profile, SPP):
  - 00001101-0000-1000-8000-00805F9B34FB

Bluetooth RFCOMM



- Emulates a serial port (RS-232 protocol)
- In-sequence, reliable delivery of serial stream
- Enables cable replacement scenarios
- Allows multiple “channels” between two devices (multiplexing via L2CAP)

Bluetooth RFCOMM in Android

- Server (device A)
  - Publish serial port service in local SDP database
  - Specify UUID for serial port service
  - Specify name for service
  - System assigns RFCOMM channel number
- Client (device B)
  - Client knows device address of server (discovery)
  - Specifies UUID of required service
  - Client adapter requests RFCOMM channel number from server

Blocking I/O

- Bluetooth I/O calls are blocking → separate thread
- Problem: separate thread cannot update GUI → Handlers
- Worker Thread (handler created by main thread):

```
String s = in.readLine(); // from Bluetooth InputStream, blocking
Message msg = handler.obtainMessage(MY_MESSAGE_TYPE, s);
handler.sendMessage(msg);
```

- Main (GUI) Thread:

```
private Handler handler = new Handler() {
    public void handleMessage(Message msg) {
        if (msg.what == MY_MESSAGE_TYPE) {
            String line = (String) msg.obj;
            listAdapter.add(line);
} } };
```

Inter-Thread Communication: Message Queues, Handlers

- Each thread can have a message queue
  - Main thread has message queue
- Each message queue can have zero or more handlers
  - New handler gets attached to message queue of current thread
- Handlers
  - Sending messages to a message queue
    - handler.sendMessage(msg);
  - Handling messages from a message queue
    - **public void** handleMessage(Message msg) { ... }
- Uses
  - Schedule messages for execution at some point of time
  - Enqueue actions for execution by another thread

Aufgabe 2: Spiel für zwei Android-Handys

Ad-hoc mobile gaming. Kommunikation von Touch-Screen-Eingaben und Spiel-Ereignissen:

a) Welche Kommunikationstechnologie bietet sich für dieses Szenario an? Warum?

Aufgabe 2: Spiel für zwei Android-Handys

Ad-hoc mobile gaming. Kommunikation von Touch-Screen-Eingaben und Spiel-Ereignissen:

b) Welcher Kommunikationsaspekt ist im Spielkontext besonders wichtig? Warum?

Aufgabe 2: Spiel für zwei Android-Handys

Ad-hoc mobile gaming. Kommunikation von Touch-Screen-Eingaben und Spiel-Ereignissen:

c) Nach dem Starten des Spiels auf beiden Mobiltelefonen soll automatisch ein bidirektionaler Kommunikationskanal aufgebaut werden. Wie wird die Verbindung hergestellt? Welche Rolle nehmen die Kommunikationspartner jeweils ein? An welchen Stellen wird der Benutzer involviert?

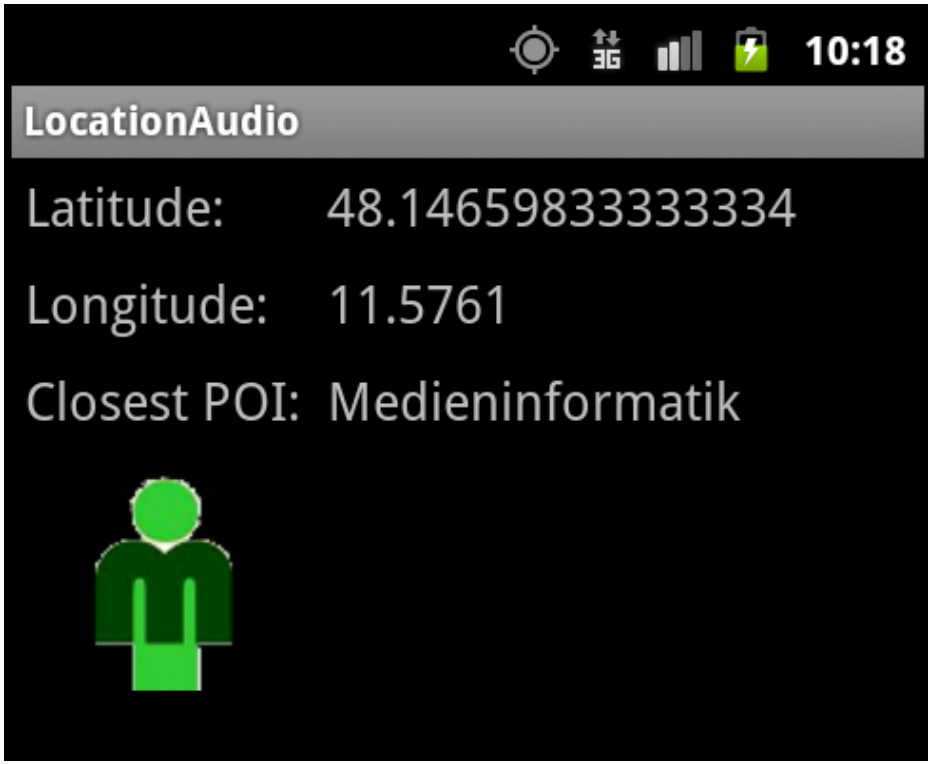
Aufgabe 2: Spiel für zwei Android-Handys

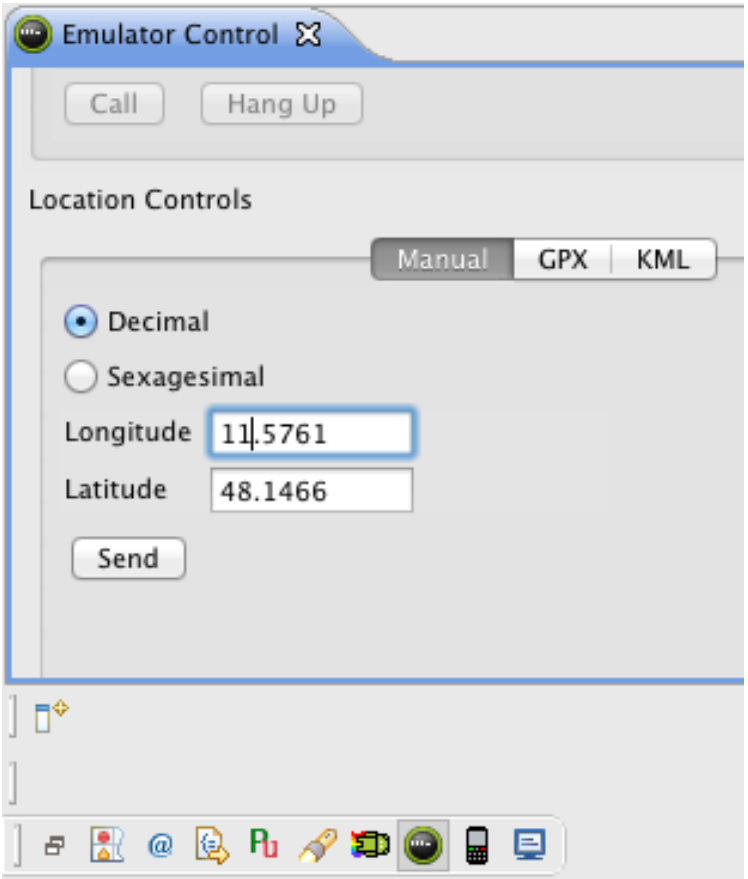
Ad-hoc mobile gaming. Kommunikation von Touch-Screen-Eingaben und Spiel-Ereignissen:

d) Welche Mechanismen bieten sich an, damit blockierende I/O-Aufrufe das Spiel bzw. die GUI nicht blockieren?

Aufgabe 3: Location-Based Audio

Audiodateien an Orten abspielen und Bilder anzeigen





Aufgabe 3: Location-Based Audio

Audiodateien an Orten abspielen und Bilder anzeigen

- Im Umkreis von 400m um den Gasteig (lat, lon = 48.131736, 11.590119) wird klassische Musik gespielt und ein passendes Bild angezeigt.
- Im Umkreis von 750m um den Münchner Zoo (lat, lon = 48.10043, 11.55150) sind Tiere zu hören und ein passendes Bild wird angezeigt.
- Im Umkreis von 200m um das Medieninformatik-Gebäude (lat, lon = 48.146900, 11.576120) ist Ihr Lieblingslied zu hören und ein passendes Bild wird angezeigt.

Aufgabe 3: Location-Based Audio

Audiodateien an Orten abspielen und Bilder anzeigen

Requirements:

- Mindestens eine Audio-Datei ist in der Anwendung enthalten.
- Mindestens eine Audio-Datei wird aus dem Internet geladen.
- Die Bilddateien dürfen alle in der Anwendung enthalten sein.

Aufgabe 3: Location-Based Audio

- Programmgerüst auf Vorlesungs-Webseite
- Audio abspielen: MediaPlayer Klasse
- Emulator: Längen- und Breitengrade eingeben
  - „GPS support“ & „Audio playback support“
  - Evtl. neues device image anlagen
- Handy: „mock locations“ erlauben
  - „Falsche Standorte“ in Einstellungen | Anwendungen | Entwicklung aktivieren
  - Android Market, Suche nach „mock location“

Media APIs

- Package android.media
  - MediaPlayer: Playing audio and video content
  - MediaRecorder: Record audio and video content
- Content sources
  - Internet
  - .apk file (as a resource or as an “asset”)
  - Secure digital (SD) card

Audio Player Example

```
private MediaPlayer mediaPlayer = null;

private void playAudio(String url) throws Exception {
    killMediaPlayer();
    mediaPlayer = new MediaPlayer();
    mediaPlayer.setDataSource(url);
    mediaPlayer.prepare();
    mediaPlayer.start();
}

private void killMediaPlayer() {
    if (mediaPlayer != null) {
        mediaPlayer.stop();
        mediaPlayer.release();
        mediaPlayer = null;
    }
}
```

Playing Local Media Files

- In /res/raw/song.mp3

```
killMediaPlayer();
mediaPlayer = MediaPlayer.create(this, R.raw.song);
mediaPlayer.start();
```

Location Manager Service

- Obtain device's geographical location
- Get notification upon entering a specified location
- Get last location
  - getLastKnownLocation(provider);
  - provider: GPS_PROVIDER, NETWORK_PROVIDER
- Register listener for location updates
  - requestLocationUpdates(provider, minTime, minDistance, listener);
  - provider: GPS_PROVIDER, NETWORK_PROVIDER
  - minTime: minimum time between notifications [ms]
  - minDistance: minimum distance between notifications [m]
  - listener: notified about location updates

Template for Location Updates

```
public class MainActivity extends Activity implements LocationListener {
    LocationManager locationManager = null;
    …
    public void onLocationChanged(Location location) {
        if (location != null) {
            // process location update
        }
    }
    public void onProviderDisabled(String provider) {}
    public void onProviderEnabled(String provider) {}
    public void onStatusChanged(String provider, int status, Bundle ext) {}
}
```

Example: Location Updates

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    LocationManager locMgr = (LocationManager)
            getSystemService(Context.LOCATION_SERVICE);

    locMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER,
            5000, 0, locListener);

    …
}
```

Permissions (in AndroidManifest.xml)

- Permissions for location-based services

```
<uses-permission
  android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
  android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
  android:name="android.permission.ACCESS_MOCK_LOCATION" />
<uses-permission
  android:name="android.permission.INTERNET" />
  …
```

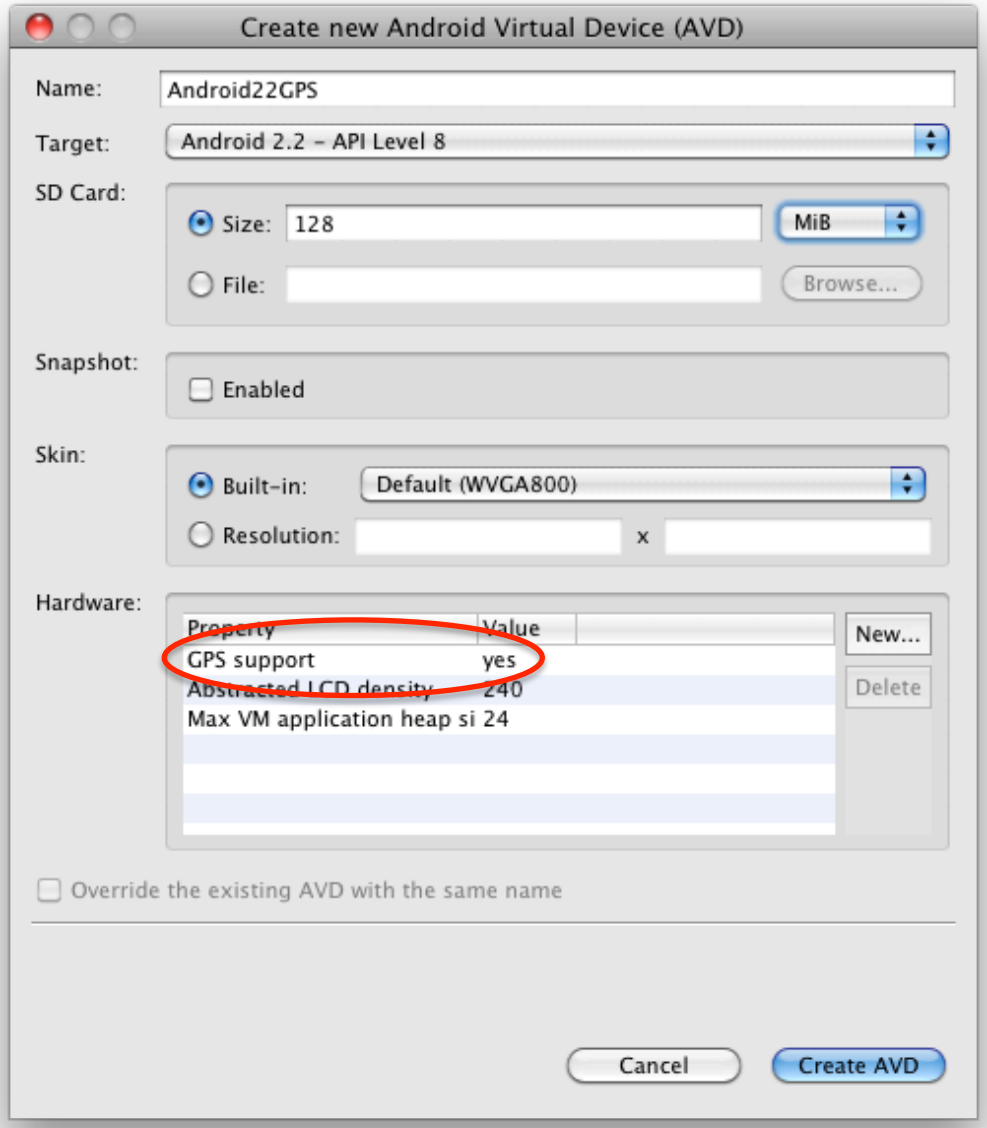
- Child of <application>

```
<uses-library android:name="com.google.android.maps" />
```

- Overview of Android permissions

  developer.android.com/reference/android/Manifest.permission.html

Enabling GPS on the Emulator



Distance Between Geo-Locations

- Distance (in m) between two geolocations

```
float[] results = new float[1];
Location.distanceBetween(lat, lon, poi.latitude, poi.longitude, results);
float distance = results[0];
```

Abgabe

- Plagiate sind verboten und führen zum Ausschluss aus der Veranstaltung!
- Dieses Übungsblatt muss einzeln bearbeitet werden.
- Bestandteile der Abgabe
  - Aufgaben 1 & 2 als PDF-Datei
  - Aufgabe 3 als zip-Datei (Eclipse, Export → Archive file)
- Abgabe bis zum 5.12.2011 um 12:00 Uhr im **neuen** UniWorX Portal (https://uniworx.ifi.lmu.de/) ab.
- Sie sollten Ihre Lösung in der Übung vorstellen können!