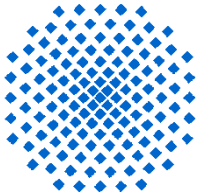


Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Mobile Computing Lab
Assignment 3

Positioning

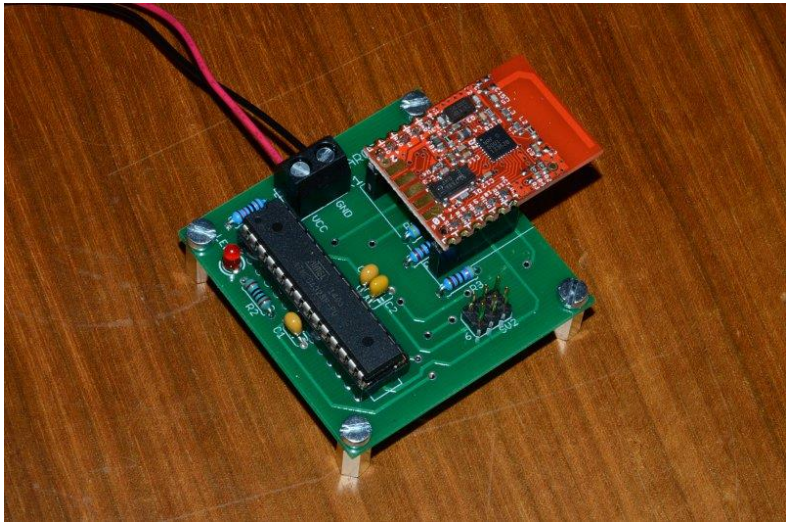
Frank Dürr, Saravana Murthy, Ahmad Slo, Zohaib Riaz

Outline

- Task 1: Reading Google Eddystone beacons with Android
- Task 2: GPS positioning with Android

Task 1: Reading Eddystone Frames with Android

- **Setup:** Eddystone Beacon in the lab
  - Faros beacon: https://github.com/duerrfk/Faros
  - Broadcasts UID, URL, TLM frames alternating every second
- **Task:** Receive and decode UID, URL, TLM frames with Google Android and display the following information in an Android activity:
  - Beacon ID
  - URL
  - Voltage
  - Temperature
  - Distance to beacon in meter (optimize estimation/calibration)



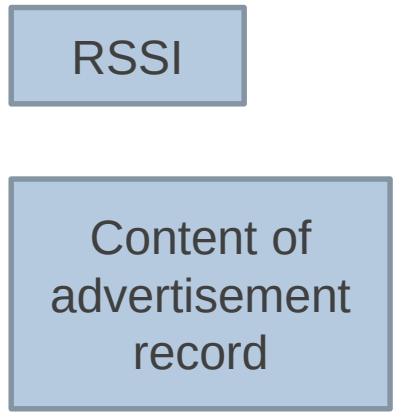
Hint for the App

- Use the onLeScan method of the LeScanCallback

```
new BluetoothAdapter.LeScanCallback() {
        @Override
        public void onLeScan(
                final BluetoothDevice device,
                int rssi,
                byte[] scanRecord) {

                /* add your code here */

        }
};
```



Outline

- Task 1: Reading Google Eddystone beacons with Android
- **Task 2: GPS positioning with Android**

Goals

- Sense location of a real mobile device
  - Using the Android location manager & GPS
- Log location trace to SD-card
  - Using GPX file format
- Display logged track on your PC
  - In Google Earth or any other tool you like

Background: Android

We need some **technical knowledge of Android** to solve this task

- Last time, we provided the Android essentials:
  - Android Studio
  - Android Virtual Devices
  - Debugging (Breakpoints, logging)
  - Activities (Lifecycle & GUI)
- Now, we need in addition:
  - Android services
  - Android location manager
  - File handling in Android

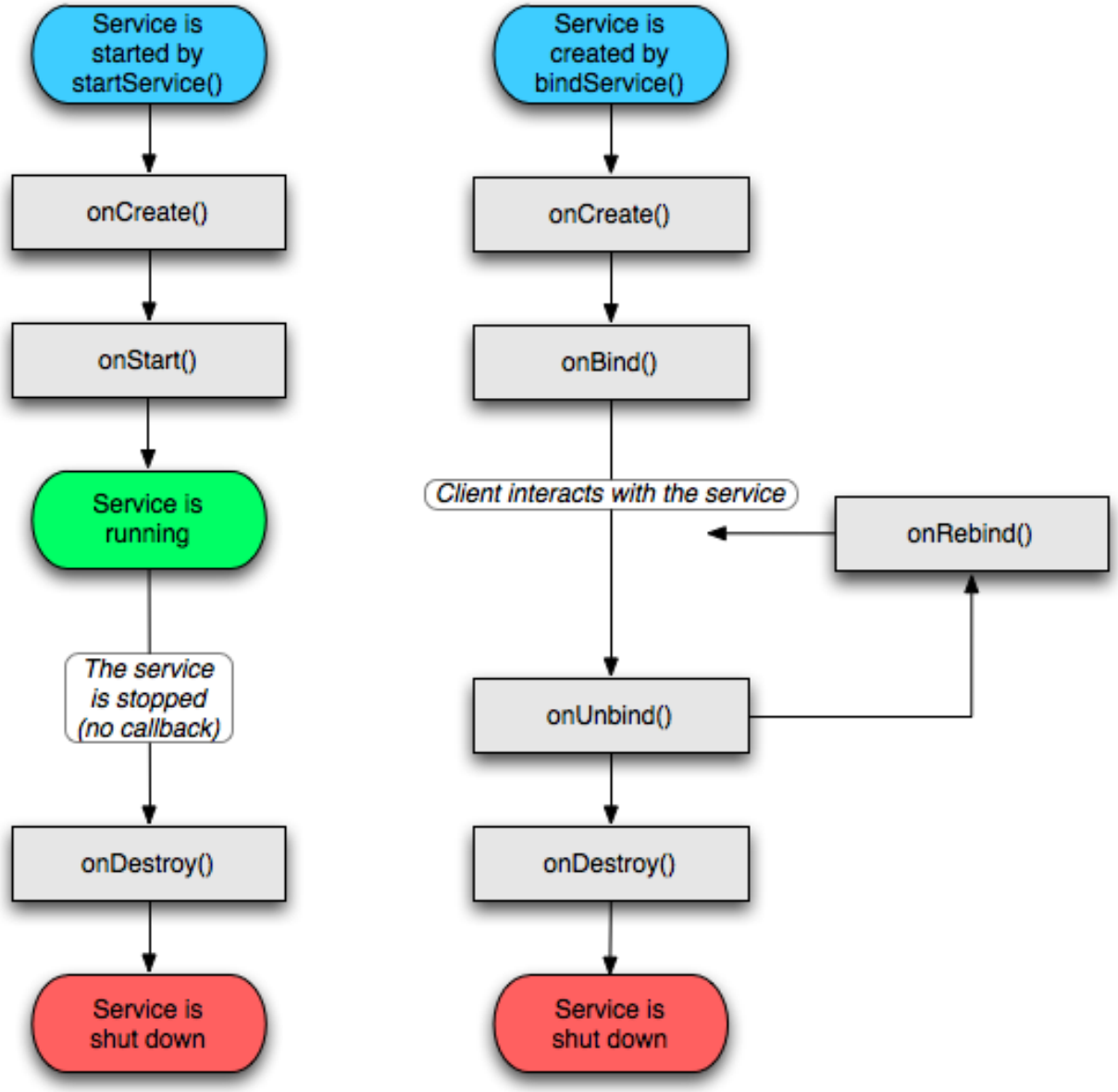
Android Services

- Support long-running background task
- No GUI
- Activities can communicate with service (RPC)
- Examples:
  - Playing music in the background
  - Fetching/Sending e-mails in the background
  - Logging locations

Service Lifecycle

Service start mechanisms:

- Explicitly using `startService()` & `stopService()`
- Implicitly by binding an activity to the service ("using" the service)
  - After binding, activity invokes (remote) service methods (RPC)
- Both mechanisms can be mixed!



Starting Services

- **Explicitly** by specifying exact class to be run:

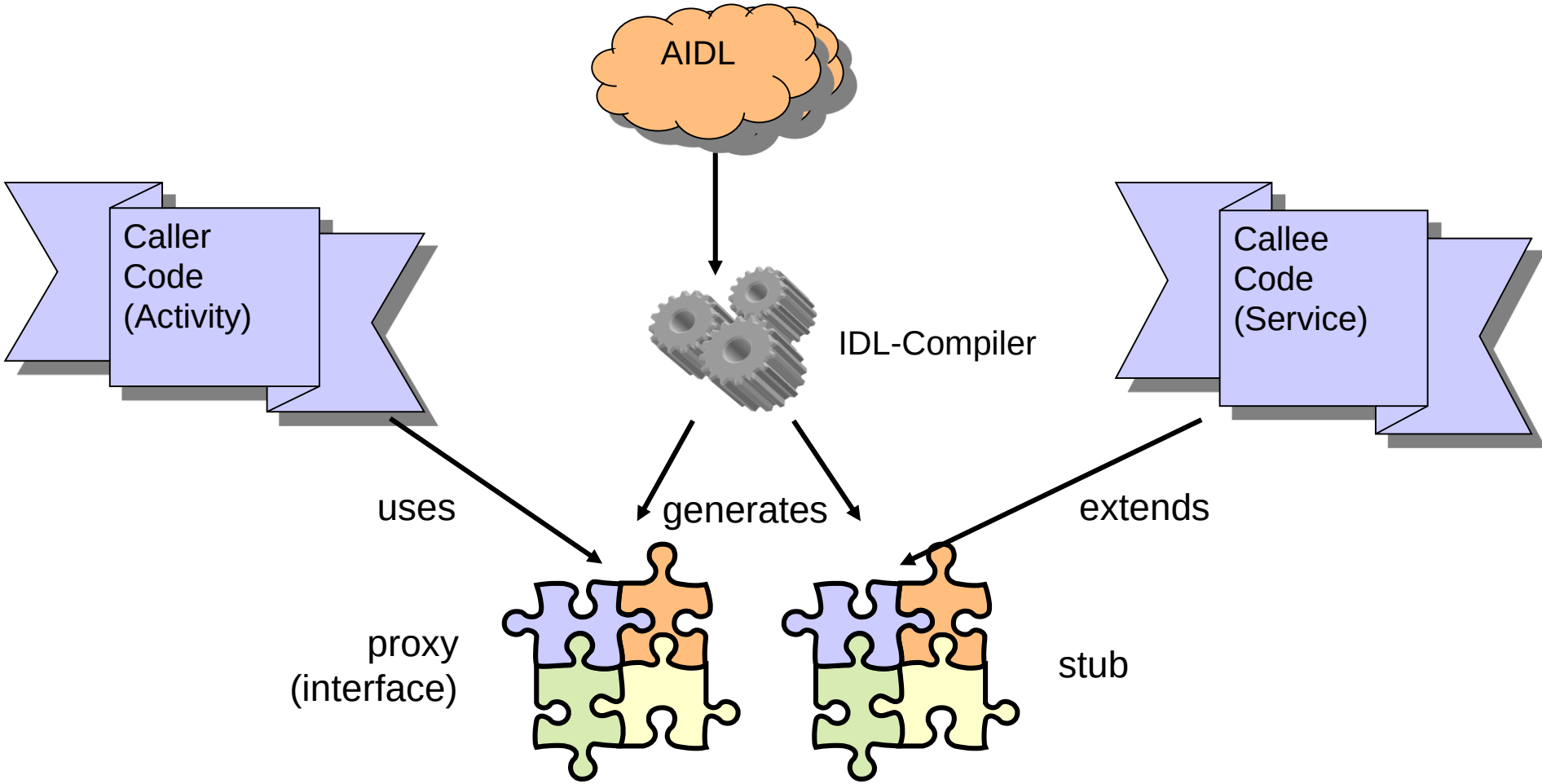
```
Intent i = new Intent(context, MyService.class);
startService(i);
```

- **Implicitly** by binding to service:

```
Intent i = new Intent(this, MyService.class);
bindService(i, this, 0);
```

Inter-process Communication (RPC)

- Activity calls method of service in local process
- Code is executed in remote process



Android RPC Example (1)

Service interface specified with Android Interface Definition Language (AIDL):

```
package de.uni_stuttgart.calculatorservice;

interface ICalculatorService {
  double add(double x, double y);
}
```

Android RPC Example (2)

- Android IDL compiler generates stub class
- Stub must be extended by service implementation:

```
class CalculatorService extends Service {
  ...
  private class CalculatorServiceImpl extends
      ICalculatorService.Stub {
      public double add(double x, double y) {
          return x+y;
      }
  }
  ...
```

Android RPC Example (3)

1. Activity (Client) binds to service:

```
Intent i = new Intent(this, CalculatorService.class);
bindService(i, this, 0);
```

2. Activity gets proxy object when service connection is established (callback)

```
public void onServiceConnected(ComponentName name,
      IBinder service) {
      ICalculatorService calcServiceProxy =
      ICalculatorService.Stub.asInterface(service);
}
```

3. Activity calls remote method

```
double result = calcServiceProxy.add(5,6);
```

4. Activity unbinds service

```
unbindService(this);
```

Location Manager (1)

- **Location Manager** provides applications with location information
  - Longitude, latitude, height
- Android implements two **location providers**:
  - **GPS:**
    - satellite-based
    - only outdoors
    - high precision & high energy consumption
  - **Network:**
    - positions of WiFi access points and cell towers
    - indoors & outdoors
    - less precise & smaller energy consumption

Location Manager (2)

Application can **query** last known position or register for **notifications** on location updates:

```
LocationManager locationmanager = (LocationManager)
     getSystemService(Context.LOCATION_SERVICE);
locationmanager.requestLocationUpdates(
     LocationManager.GPS_PROVIDER, // use GPS device
     interval,    // hint for notification interval
     minDistance, // hint for minimum position distance
     subscriber); // callback receiver
locationmanager.removeUpdates(subscriber);
```

Location Manager (3)

Location event subscriber implements callback interface **`android.location.LocationListener`**:

```
public void onLocationChanged(Location location) {
     double longitude = location.getLongitude();
}
public void onProviderDisabled(String provider) {}
public void onProviderEnabled(String provider) {}
public void onStatusChanged(String provider, int
status, Bundle extras) {}
```

Location Manager (4)

Position information is **privacy sensitive data**!

- Application must declare that it uses location information in manifest:

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

- User can deny installing App

File I/O (1)

- Similar to standard Java file IO
  - See Oracles's Java tutorial: http://docs.oracle.com/javase/tutorial/essential/io/index.html
- Special to Android:
  - By default, files are private to an application
    - Open private files with
      `Context.openFileInput(filename)`
      `Context.openFileOutput(filename, mode)`
    - To make files reachable from other apps, check:
      `FileProvider class`
    - Need to set permissions to access SD card in manifest:
      `<uses-permission android:name=`
      `"android.permission.WRITE_EXTERNAL_STORAGE" />`

File I/O (2)

- Accessing files on SD-card:

```
File sdDir = Environment.getExternalStorageDirectory();
File myFile = new File(sdDir, "foo.txt");
...
```

- To use SD-card with emulator, you have to create an SD-card image file and tell the emulator about it:
  - SD card size can be specified during creating of Android Virtual Device (AVD)
- Reading file from SD-card image:

```
adb pull sdcard/myfile.txt
```

Recommended Reading

- Android services:
  http://developer.android.com/guide/components/services.html
- Location information in Android:

  http://developer.android.com/guide/topics/location/index.html

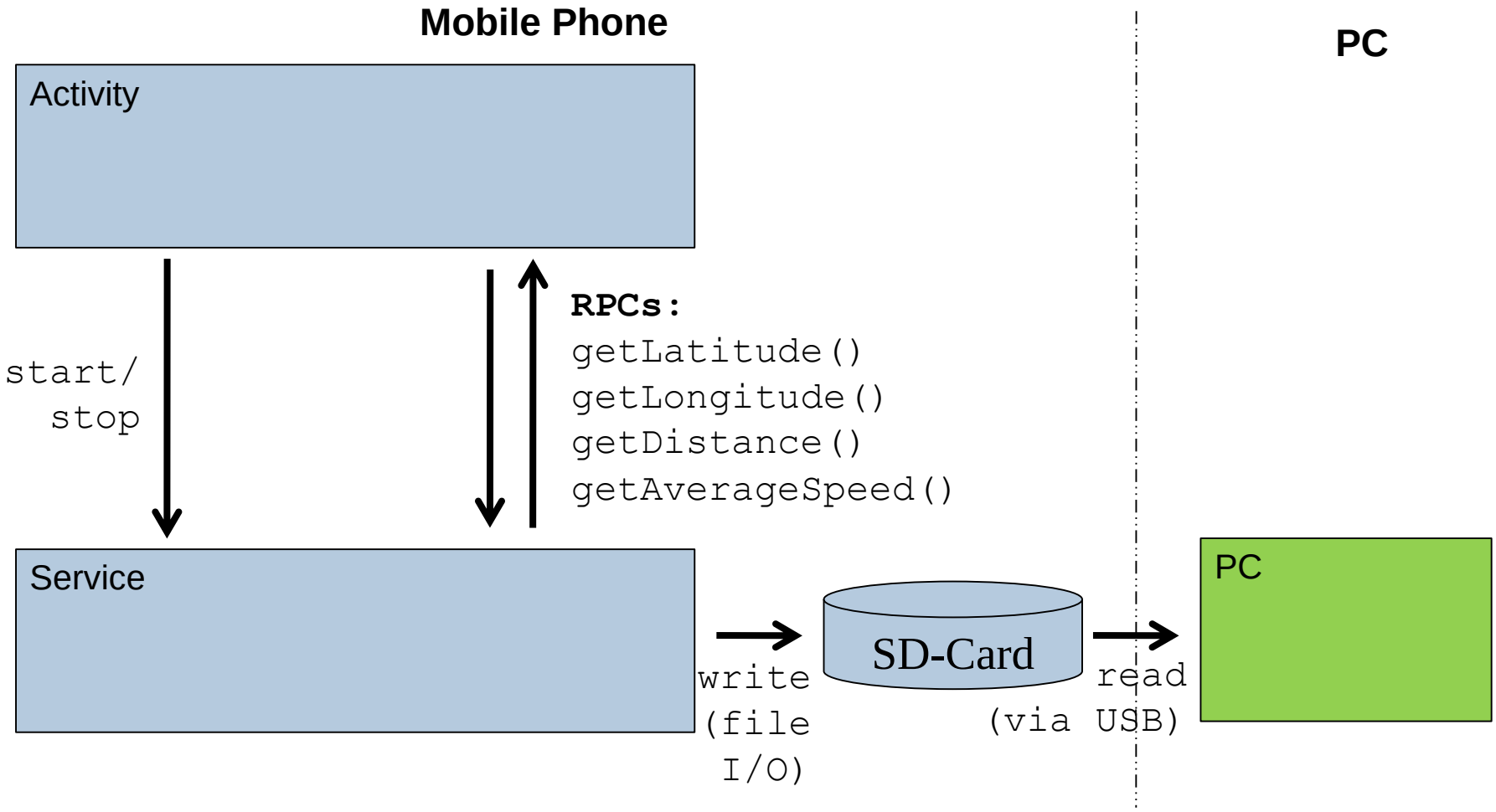
Task 2

- Sense positions using location manager
  - Use GPS
  - Subscribe to location updates
  - Define reasonable values for distance and time thresholds for updates
- Log positions to file on SD card
  - Using GPX file format
- Test your App with emulator and real device
- Display trace
  - In Google Earth or any other tool you like

Logging done in background (service).

Logging must continue even if no activity is displayed.

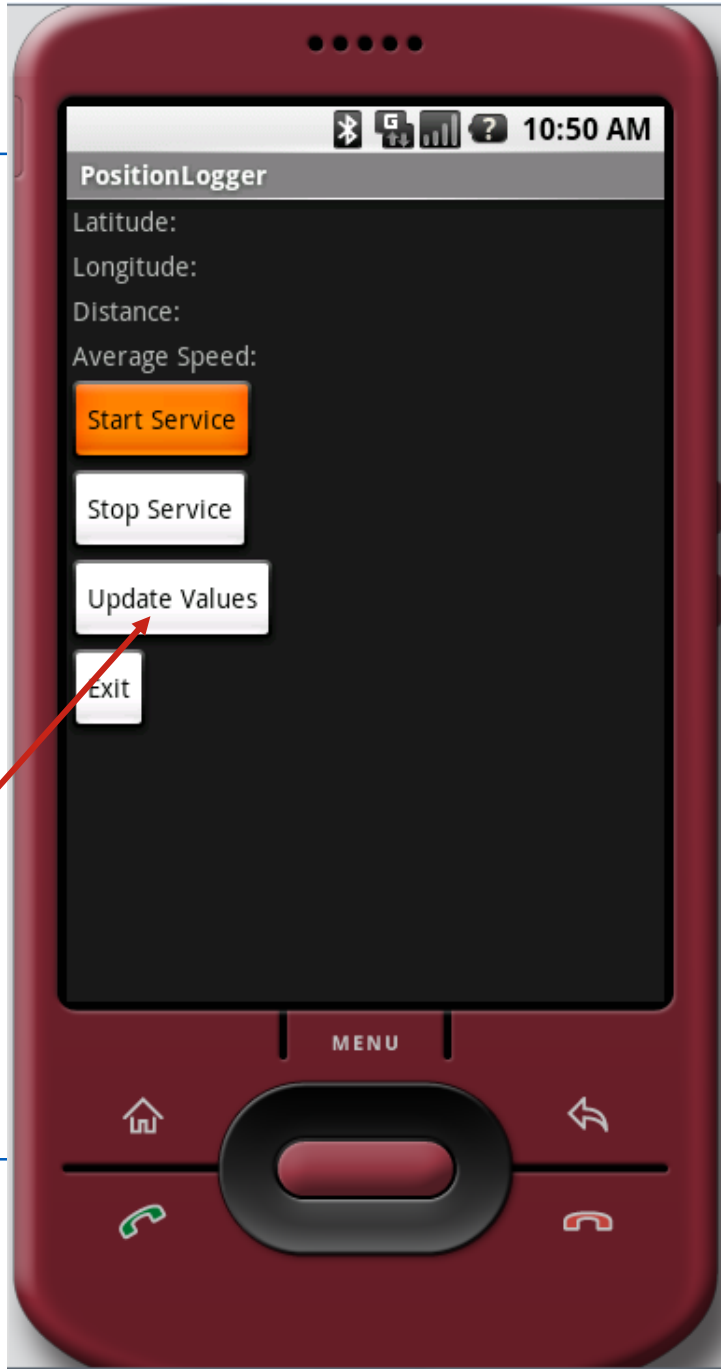
Architecture



The Activity

- **Controlling service**
  - Starting
  - Stopping
  - Querying location information
- **Displaying location information**
  - Current position
  - Distance travelled
  - Average speed

Click updates the above location information



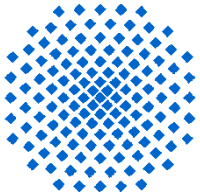
The Service

- **Logs GPS position information**
  - Write log file onto SD-card
    - New logs override old log files
  - Logging starts when service is started and stops when service is stopped via GUI
  - Logger service must keep running even if GUI (Activity) is finished
- Provides **RPC interface** to the client/activity:
  - getLongitude: current position
  - getLatitude: current position
  - getDistance: travelled distance since service has been started
  - getAverageSpeed: averageSpeed

Testing

- First, test with **Android Virtual Device**
  - Playback provided NEMA/GPX file

- Testing with **real hardware**
  - Can use your own Android device
  - Or use our phones: Samsung Galaxy Nexus (Android 4.3)
    - Ask the tutors for access
    - Don’t take devices with you (test around computer science building)
    - Only WiFi communication (EDUROAM)



Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Organizational Information

Submission & Next Meeting

- Post questions on the ILIAS board

- You have **2+1 weeks time** to work on this assignment until the final date of submission!
- Next assignment and demonstration of your results scheduled for **Wednesday, June 20, 2018**
  - Room 0.153 at 3:45 pm

- **Submit via Ilias**
  - **Source code** of you evaluation results
  - Group submission!

Questions?