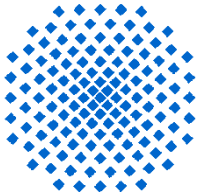


Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

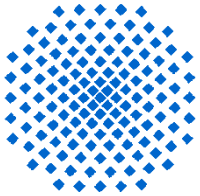
Mobile Computing Lab
Assignment 2

Bluetooth Low Energy (BLE)

Frank Dürr, Christoph Dibak

Outline

- Background: Bluetooth Low Energy (BLE)
- Task 1: Android BLE App: Weather App
- Task 2: Android BLE App: Fan Control App
- Task 3: Android BLE App: iBeacon Proximity
- Task 4: Android BLE App: Trilateration with iBeacons
- Organizational issues



Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Bluetooth Low Energy

Motivation

The Internet of Things: Everything connected

Wireless sensors and actuators will be everywhere

- “Quantified Self”: monitor everything about your life
  - Fitness trackers, blood pressure, glucometers
- Environmental and urban monitoring
  - Air quality, noise level, temperature
- Home automation
- Smart watches, wearables
- Proximity sensors (iBeacon)

➔ Low energy consumption is key!

Motivation

The Internet of Things: Everything connected

Wireless sensors and actuators will be everywhere

- “Quantified Self”: monitor everything about your life
  - Fitness trackers, blood pressure, glucometers
- Environmental and urban monitoring
  - Air quality, noise level, temperature
- Home automation
- Smart watches, wearables
- Proximity sensors (iBeacon)

➔ Low energy consumption is key!

Bluetooth Low Energy

- 2.4 GHz wireless communication technology
- Low range
  - ~ 10 meters
- Ultra low energy consumption
  - Run from coin cells for months or years
  - No need for chargers
- Low cost
  - Less than 1$
- Low latency
  - Connect and acknowledge data within 3 ms
  - Can send data without connection
- High data rate not a goal
  - Standard Bluetooth more bandwidth- and energy-efficient for high data rates

Achieving Low Energy Consumption

- **Minimize duty cycles**
  - μA in sleep mode vs. mA in active mode
  - Active only every 7.5 ms to 4 s (connection interval)

- **Fast connection setup**
  - Bluetooth uses frequency hopping on channels
  - BLE only uses 3 channels for advertising: radio on for 1.2 ms
    - Standard Bluetooth uses 16 to 32 channels: radio on for 22.5 ms
  - Only 3 ms between connecting and acknowledgement of packet
    - Standard Bluetooth might take up to 100 ms for connection setup
  - BLE can also broadcast data without any connection setup

Device Roles

- Devices supporting connections:
  - **Peripheral**
    - Only one connection to one central
  - **Central**
    - Possibly multiple connections to different peripherals
    - Initiates connection to peripheral

- Devices not supporting connections:
  - **Broadcaster:** only sender
  - **Observer:** only receiver

Generic Attribute Profile (GATT): Profiles, Services, Characteristics (1)

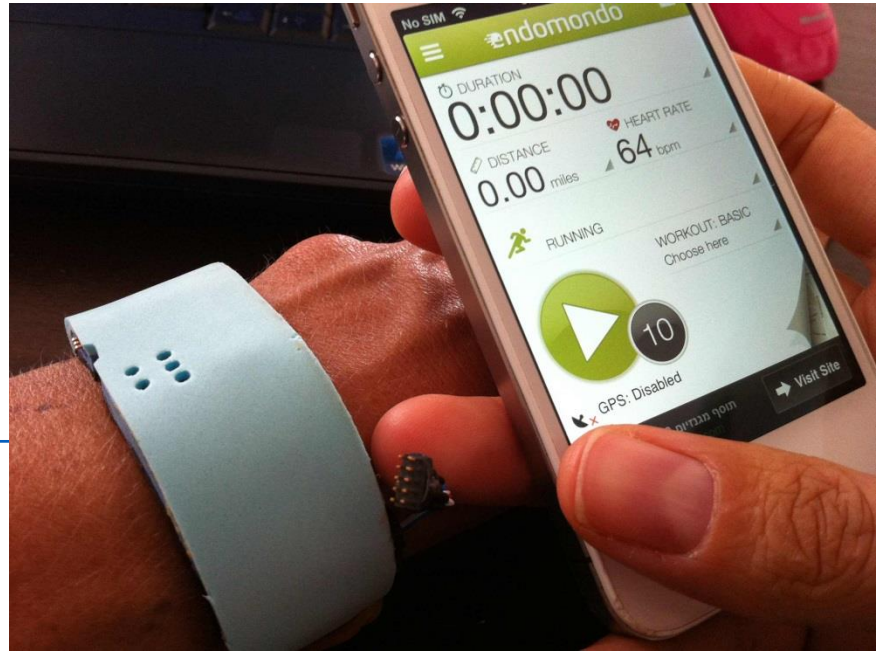
- **Generic Attribute Profile (GATT):** Describes how GATT **servers** can provide small pieces of data to GATT **clients**

- **Profile:** use case
  - Includes services to implement use case
  - Example: heart rate profile
    - "This profile enables a Collector device to connect and interact with a Heart Rate Sensor for use in fitness applications." [https://developer.bluetooth.org]
    - Used services: org.bluetooth.service.heart_rate

Generic Attribute Profile (GATT): Profiles, Services, Characteristics (2)

- **Service:** Collection of data (characteristics) and behavior
  - Which characteristics are provided?
  - Which operations are supported on characteristics?
    - read, write, notify (see next slides)

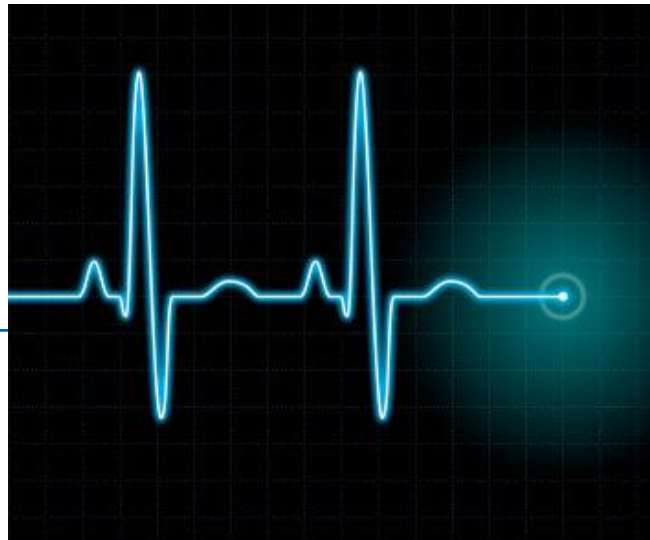
- **Example:** heart rate service (org.bluetooth.service.heart_rate)
  - Characteristic: heart rate measurement
    - Supported operation: indication
  - Characteristic: body sensor location
    - Supported operation: read



Generic Attribute Profile (GATT): Profiles, Services, Characteristics (3)

- **Characteristic:** Data value
  - Data structure declaring fields and defining data layout
  - Descriptors describing value

- **Example:** heart rate measurement (org.bluetooth.characteristic.heart_rate_measurement)
  - Flags (8 bits)
    - Bit 0: 0 = heart rate defined as uint8; 1 = heart rate defined as uint16
    - etc.
  - Heart rate measurement (uint8 or uint16)
  - etc.



Standard and Custom Services and Characteristics

- BLE defines sets of standard …
  - … profiles:
    - https://developer.bluetooth.org/gatt/profiles/Pages/ProfilesHome.aspx
  - … services:
    - https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx
  - … characteristics:
    - https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicsHome.aspx
- Everyone can define custom profiles, services, characteristics
  - … you will use two custom services of IPVS ☺

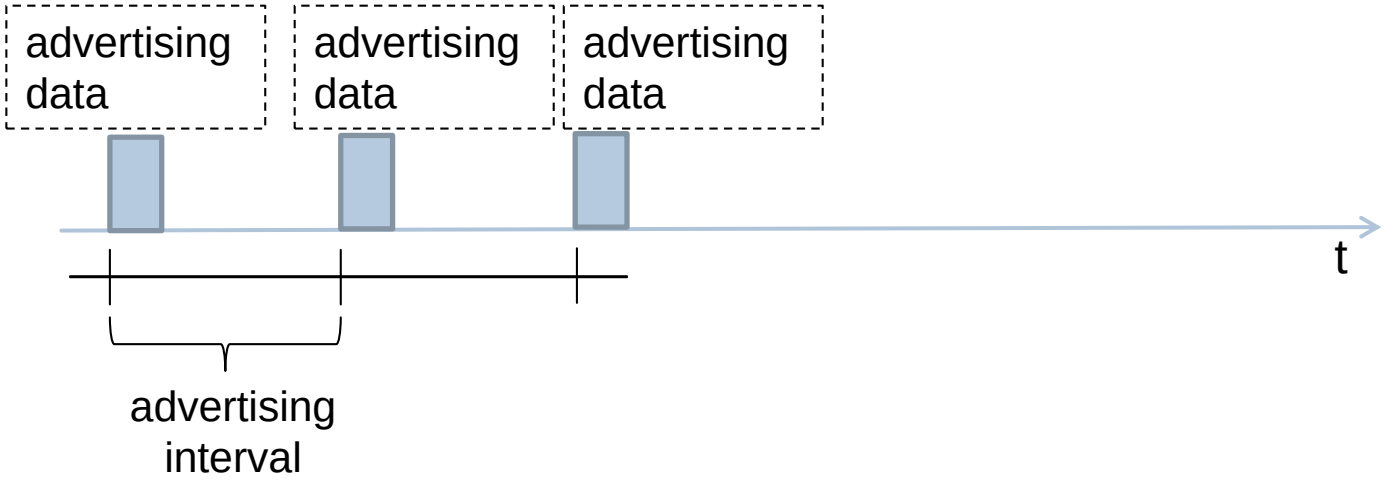
Unique Identifiers

Services and characteristics are identified by **globally unique identifiers**

- **16 bit and 32 bit UUID for standard services and characteristics**
  - Mapped to 128 bit UUIDs:
    - $UUID_{128bit} = UUID_{16bit} * 2^{96} + BaseUUID$
    - $UUID_{128bit} = UUID_{32bit} * 2^{96} + BaseUUID$
    - BaseUUID = 00000000-0000-1000-8000-00805F9B34FB
- **128 bit UUID for custom services and characteristics**
  - Created independently without coordination
    - Unix tool: `uuidgen`
    - Tons of websites
  - Must use values outside reserved range!
    - Use values smaller than BaseUUID

Advertisements

- Peripherals constantly send advertisements to let centrals in range know that they exist
  - Advertising intervals: 20 ms to 10 s
  - Payload: up to 31 bytes
    - e.g. name of the service



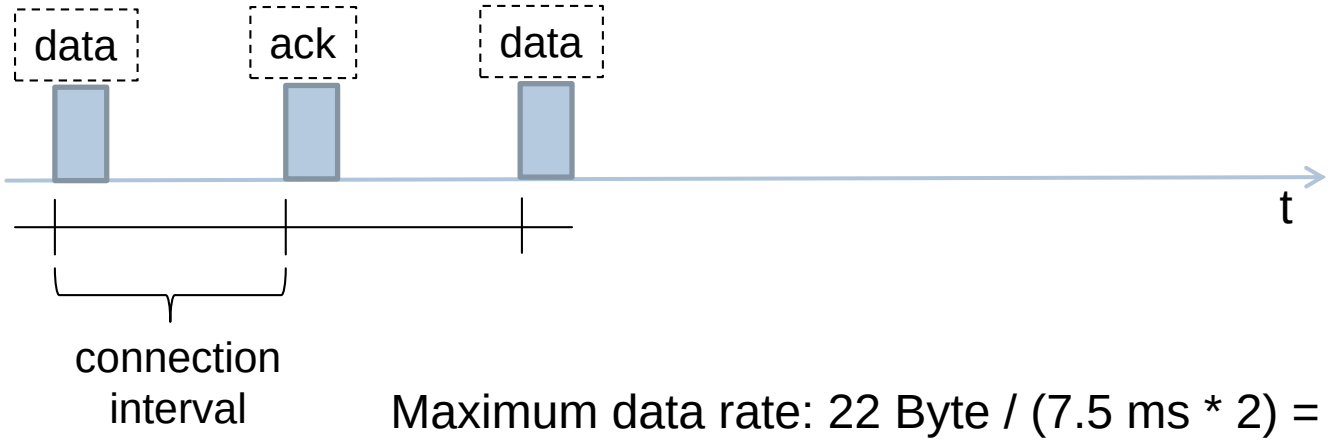
Transferring Data between Client and Server (1)

Possible **operations on characteristics**:

- **read** data from server
  - Acknowledged
  - Payload size: 22 Byte
- **write** data to server
  - Acknowledged
  - Payload size: 20 Byte
- **write without response**
  - Unacknowledged
  - Payload size: 20 Byte
- **notification** (no ACK) and **indication** (ACK) from server to client
  - Payload size: 20 Byte

Transferring Data between Client and Server (2)

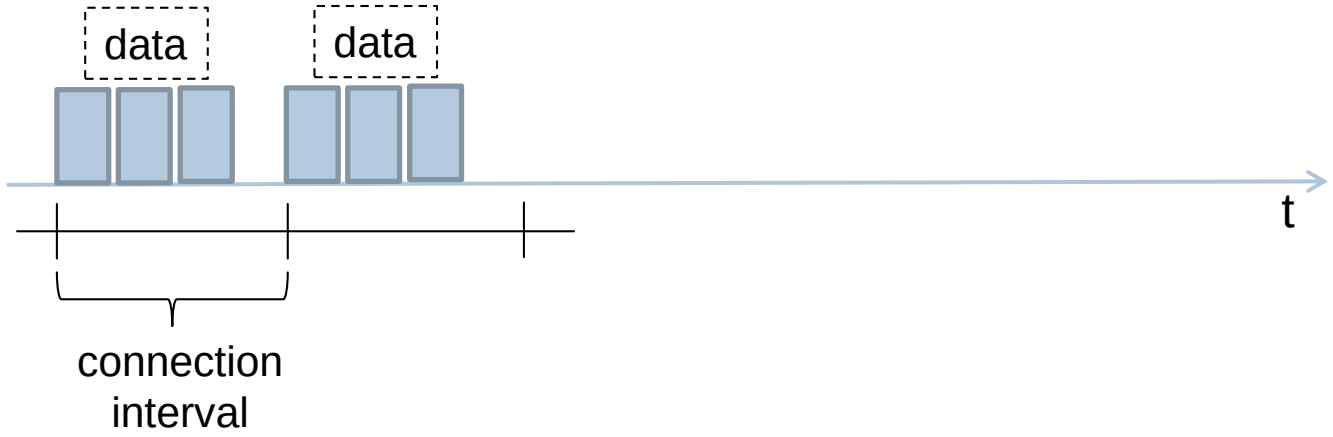
- Packets are sent in **connection intervals**
  - 7.5 ms – 4 s
  - Deep sleep (radio off) between intervals
- Acknowledged operations wait for ack before sending next packet
  - Only one packet (data or ack) per interval
  - Two intervals required for data & ack for one read or write operation



Maximum data rate: 22 Byte / (7.5 ms * 2) = 11.7 kbps

Transferring Data between Client and Server (3)

- Unacknowledged operations can send several packets in one interval
  - Number of packets depends on send buffer size of peripheral



- Maximum data rate assuming realistic send buffer size of 8 packets:
  - 20 Byte * 8 / 7.5 ms = 170.67 kbps

Let‘s get practical: BLE in Android

BLE in Android – device discovery

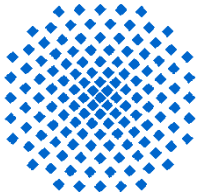
- `BluetoothManager` class manages BLE
- Methods for scanning devices
  - `startLeScan(callback)` or `startLeScan(UUID [], callback)`
  - callback is instance of `LeScanCallback`
  - Second method to specify Array of UUIDS to scan for
- If device is found, the `onLeScan(..)` of `callback` instance will be called
  - RSSI is given as parameter
- Stop scan with `stopLeScan(callback)`

BLE in Android - GATT

- Connect to GATT server: call `device.connectGatt(..)`
  - `device` provided as parameter to `onLeScan(..)`
  - needs callback instance as parameter
  - returns `BluetoothGatt` instance
- Following methods can be implemented in callback instance (among others):
  - `onConnectionStateChange`: called on connect/disconnect
  - `onServicesDiscovered`: called when service, characteristics, descirptors have been updated
  - `onCharactersiticRead`: result of read operation
  - `onCharactersiticChanged`: used for notifications
- Requesting notifications using `setCharcteristicNotification(..)` on the `BluetoothGatt` instance

Recommended Reading

- Android application fundamentals:
  http://developer.android.com/guide/topics/fundamentals.html
- Location information in Android:
  http://developer.android.com/guide/topics/location/index.html
- User interfaces:
  http://developer.android.com/guide/topics/ui/index.html
- HelloWorld example:
  http://developer.android.com/guide/tutorials/hello-world.html
- BLE:
  http://developer.android.com/guide/topics/connectivity/bluetooth-le.html



Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

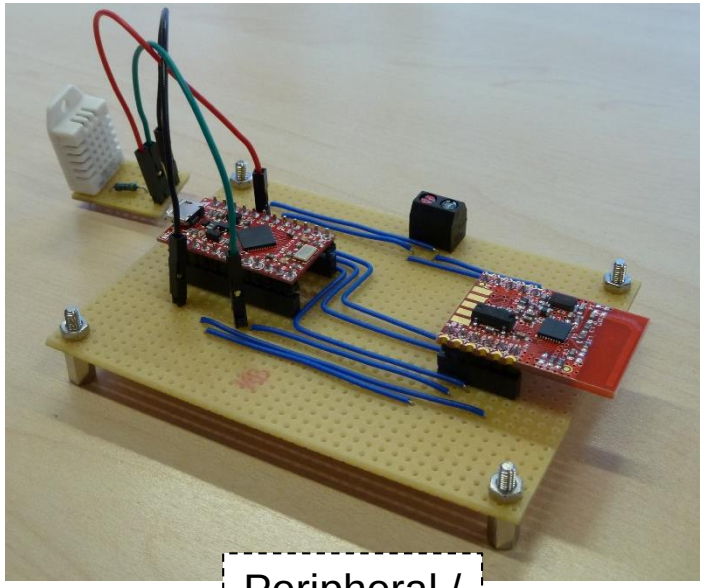
Universitätsstraße 38
D-70569 Stuttgart

Task 1
Android BLE App: Weather App

Task

Implement an **Android App for retrieving weather data from an BLE sensor**

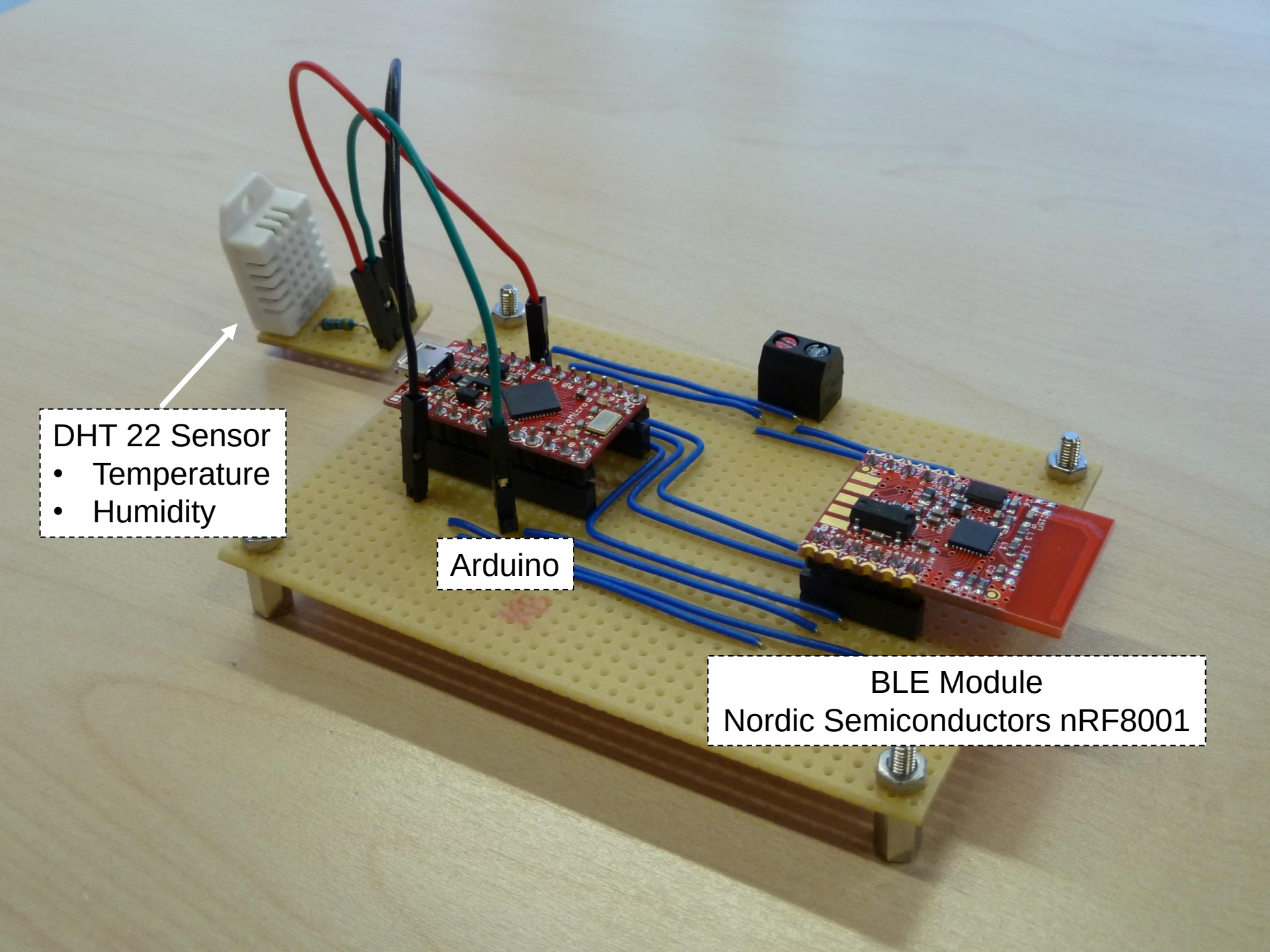
- Peripheral (sensor + Arduino + BLE radio) is provided by us
- You need to implement the **central** role on Android smartphones



Peripheral / Server

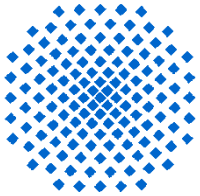
BLE

Central / Client


DHT 22 Sensor
• Temperature
• Humidity
Arduino
BLE Module
Nordic Semiconductors nRF8001

BLE Weather Service

- **Service UUID:** 0000-0002-0000-0000-FDFD-FDFD-FDFD-FDFD
- **Characteristics:**
  - Temperature Measurement
    - Standard BLE characteristic
    - https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicViewer.aspx?u=org.bluetooth.characteristic.temperature_measurement.xml
  - Humidity
    - Standard BLE characteristic
    - https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicViewer.aspx?u=org.bluetooth.characteristic.humidity.xml
- **Supported operations:** Both characteristics support read and notify
  - Your App should implement functions for querying (reading) and subscribing to notifications



Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

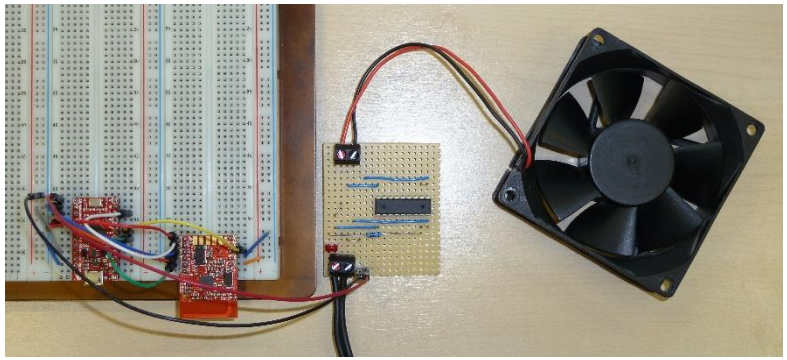
Universitätsstraße 38
D-70569 Stuttgart

Task 2:
Android BLE App: Fan Control App

Task

Implement an **Android App for controlling the speed of a fan / light-intensity of an LED**

- Peripheral (Fan/LED + Arduino + BLE radio) is provided by us
- You need to implement the **central** role on Android smartphones

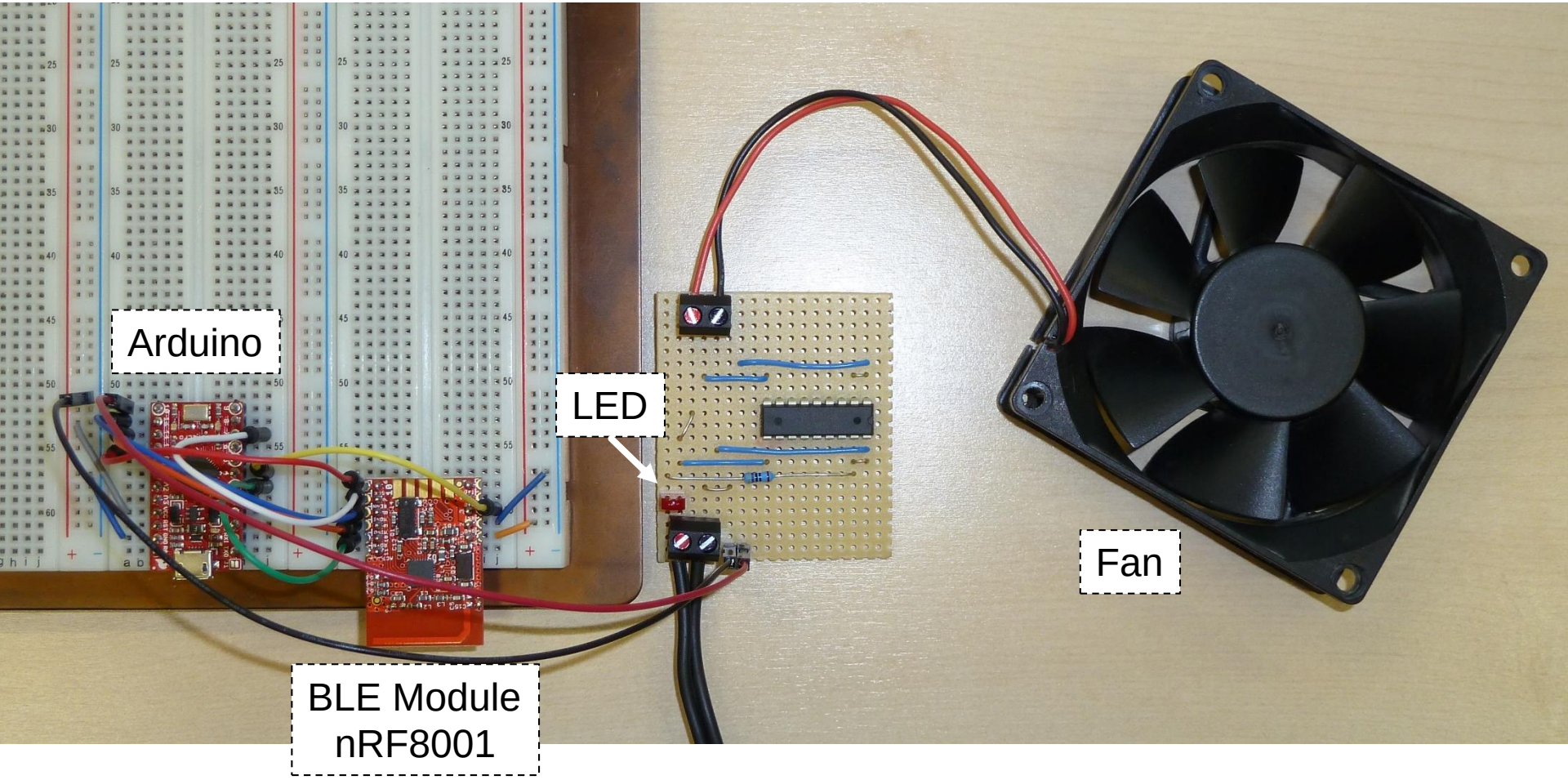


Peripheral / Server

BLE

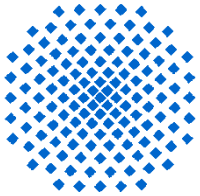
Central / Client

Peripheral / Central



BLE Fan Control Service

- **Service UUID:** 0000-0001-0000-0000-FDFD-FDFD-FDFD-FDFD
- **Characteristics:**
  - Intensity
    - UUID: 1000-0001-0000-0000-FDFD-FDFD-FDFD-FDFD
    - Format: uint16 (0 min intensity, 65535 max intensity)
    - Exponent: 0
    - Unit: none
- **Supported operations:** Write



Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

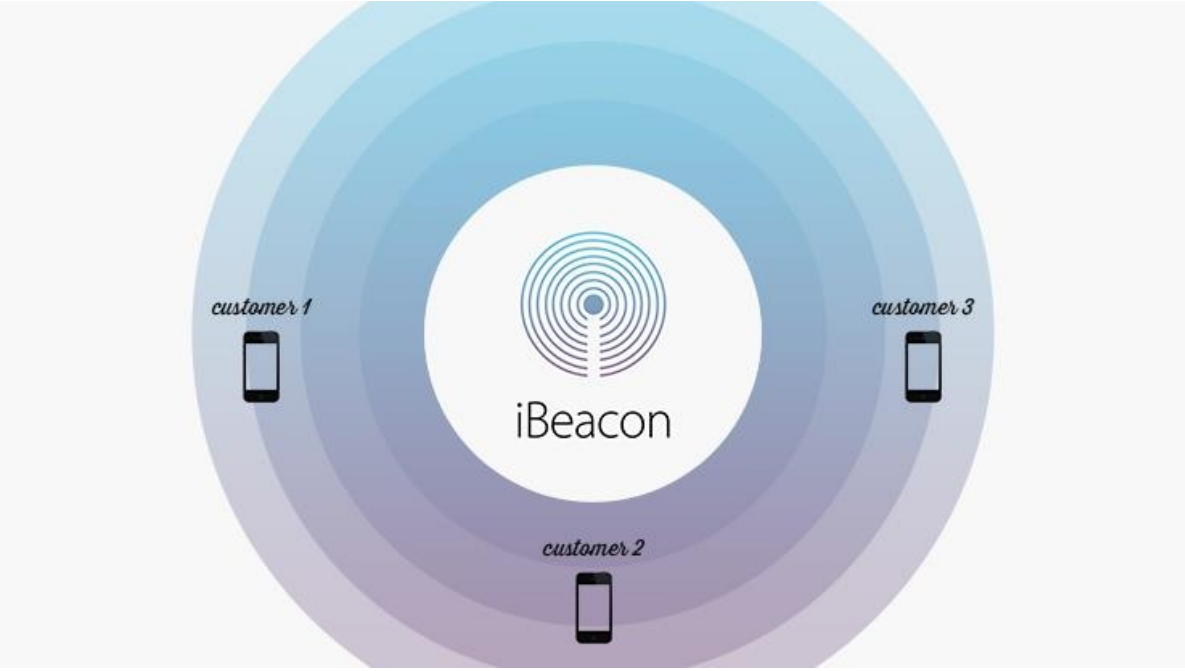
Task 3:
Android BLE App: iBeacon Proximity

iBeacon – Proximity Service

- Provide location-based information
  - Location Service
  - Indoor Navigation
  - Location-dependent marketing
  - Payments at the point of sale

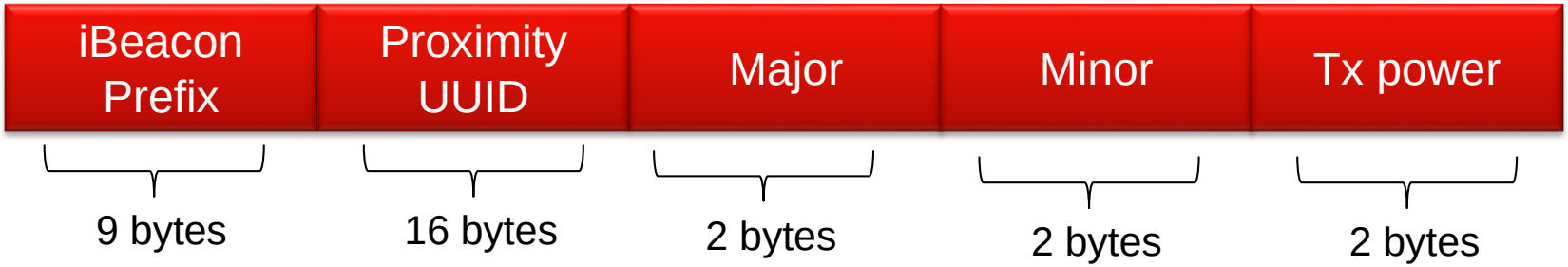
iBeacon Basics

- Beacons are small, cheap Bluetooth transmitters
  - Only send advertisements, NO connection!
- Receiver (typically smartphones) scan for beacons in their proximity



iBeacon payload

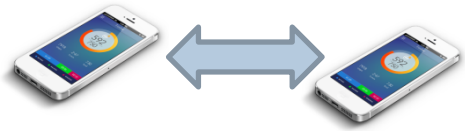
- iBeacons use payload of advertisements
  - iBeacon Prefix – 9 fixed bytes
  - Proximity UUID – distinguishes individual iBeacon services
  - Major – group related iBeacons
  - Minor – identify individual iBeacons
  - TX Power - **2's complement of actual TX power**
    - **Actual TX Power:** signal strength measured at 1 meter from the iBeacon



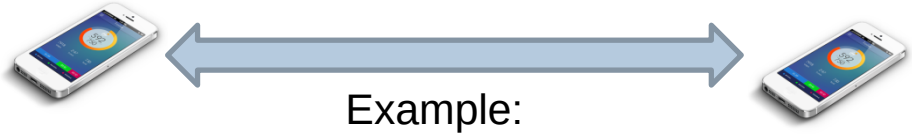
Received Signal Strength Indicator RSSI

- Measurement of the power present in a received radio signal
  - Often expressed in dBm

- Varies with distance/environment
  - Devices are close
    - Better quality and speed of communication
    - Closer to zero
  - Devices are further away
    - Lower quality and speed of communication
    - Further below zero



Example:
0.1 m
-35 dBm



Example:
1 m -59 dBm

Approximated Distance based on RSSI

- Common approximation:
  - $RSSI(d) = -(10 * n) * \log_{10}(d) + A$

  - d – distance between communication devices
  - RSSI(d) – RSSI at distance d
  - n – Pathloss exponent (free space at approx. 2)
  - A – reference received signal strength at 1 m (TX power)!

Task

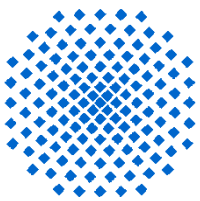
- Implement an application for Android that finds iBeacons
  - Should display all relevant information (prefix, UUID, Minor, Major, TX power)

- Calibrate TX power (on smartphone, not on peripheral!)
  - Until now, default value sent by peripheral
  - Plot distance error for several distances!
    - (several measurements)

Hints

- Android Bluetooth LE:
  - http://developer.android.com/guide/topics/connectivity/bluetooth-le.html
  - Requires Android 4.3 (API Level 18)!

- 4 beacons available:
  - Raspberry Pis + BLE
- 4 developer smartphones
  - Ask Christoph or Frank for access



Universität Stuttgart

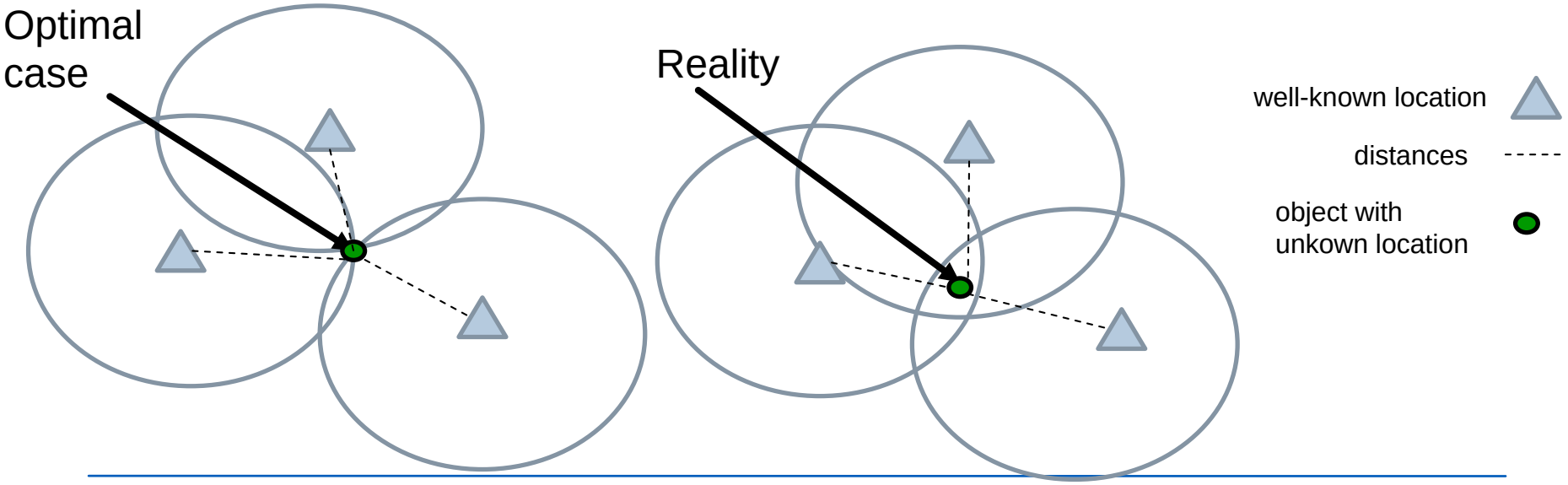
Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Task 4:
Android BLE App: Trilateration with iBeacons

Trilateration

- Find location based on distance to three well-known locations in environment
- Basic idea:
  - Intersection point of three circles is actual location
  - However: high uncertainty of locations when based on RSSI!



Task

- Implement Trilateration for Android
  - Distances based on measured RSSI of iBeacons (see Task 3)
  - Plot distance error for several locations

- Basic idea:
  - Fixed, well-known position of 3 iBeacons
    - E.g., fixed layout on a desk

    ➔ One edge of desk is origin of coordinate system
  - Measured distance as radii of circles centered at iBeacons

Submission & Next Meeting

- Questions and Answers session for this assignment:
  Thursday, May 21st, 2015
  - Send (bigger) questions by e-mail to {duerr,dibak}@ipvs.uni-stuttgart.de or post them on ILIAS at least one day before the meeting
- You have 4 **weeks time** (excluding holidays) to work on this assignment until the final date of submission!
  - Demonstration of your results scheduled for
    Thursday June 11th 2015
  - Same time, same place
- **Submit via Ilias** at least the night before the demonstration meeting
  - **Source code** and presentation (**slides**) of you evaluation results
  - Group submission!

Questions?