WiFi-based Indoor Navigation

Sebastian Neuser

Cocoa-Heads Siegen

Universität Siegen
Fachbereich 12 - Elektrotechnik und Informatik

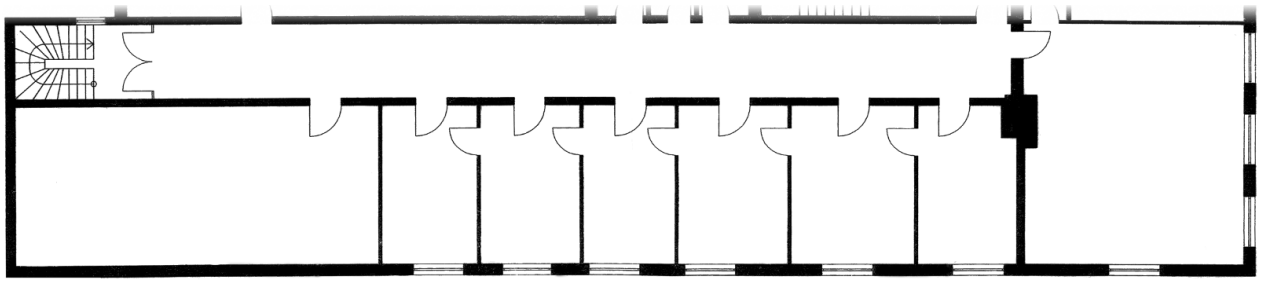
27. Februar 2012

3 Messungen
- Signalstärke
- Mittelwerte

4 Positionsbestimmung
- Vorbereitungen
- Algorithmen

5 Ergebnisse / Fazit
- Ergebnisse
- Fazit

Zielsetzung



- Ortung von Smartphones innerhalb von Gebäuden
- mindestens auf wenige Meter genau
- auf „älteren“ Geräten lauffähig
- kompatibel mit Apples App-Store
- keine Verwendung lizenzierter/unfreier Software
- keine Verwendung teurer Hardware

Problematik

- Ortung in Gebäuden per GPS nicht möglich
- Alternativen: GSM, Bluetooth, WiFi
- erste Idee: Handys scannen WLANs

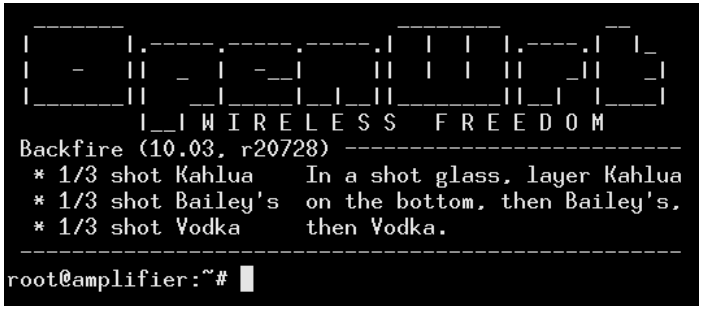
  $\rightsquigarrow$ Problem: App Store Review Guidelines ☹
- neuer Ansatz: Accesspoints suchen nach Smartphones!
- Vergleichswerte: „Fingerprints“

Kismet

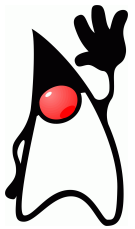
- wireless intrusion detection system
- open source (GPL)
- Plattformen: Linux, Windows, Mac OSX
- passives Scan-Verfahren
- channel hopping
- modularer Aufbau:
  - kismet_drone: Access Point; sammelt WiFi Pakete
  - kismet_server: empfängt und analysiert Pakete von Drohnen
  - kismet_client: empfängt Statusmeldungen und Statistiken vom Server; Benutzerschnittstelle

Messgeräte

- Hardware: TP-Link TL-WR841 ND
- Software: OpenWrt
- Konfigurartion:
  - DHCP
  - SSH
  - kismet_drone



Server I



1. modifizierter Kismet-Server:
   - keine Paket-Statistiken
   - direkte Weiterleitung der Header aller WiFi-Pakete
2. Kern:
   - Implementierung von Kismets Client/Server-Protokoll
     $\rightsquigarrow$ spezieller Kismet-Client
   - Datenbank für die persistente Speicherung von Fingerprints und MAC-Adressen der Clients (XML)
   - Algorithmen zur Berechnung einer Position aus gemessenen Signalstärken
   - Implementierung einer RMI-Schnittstelle

Server II

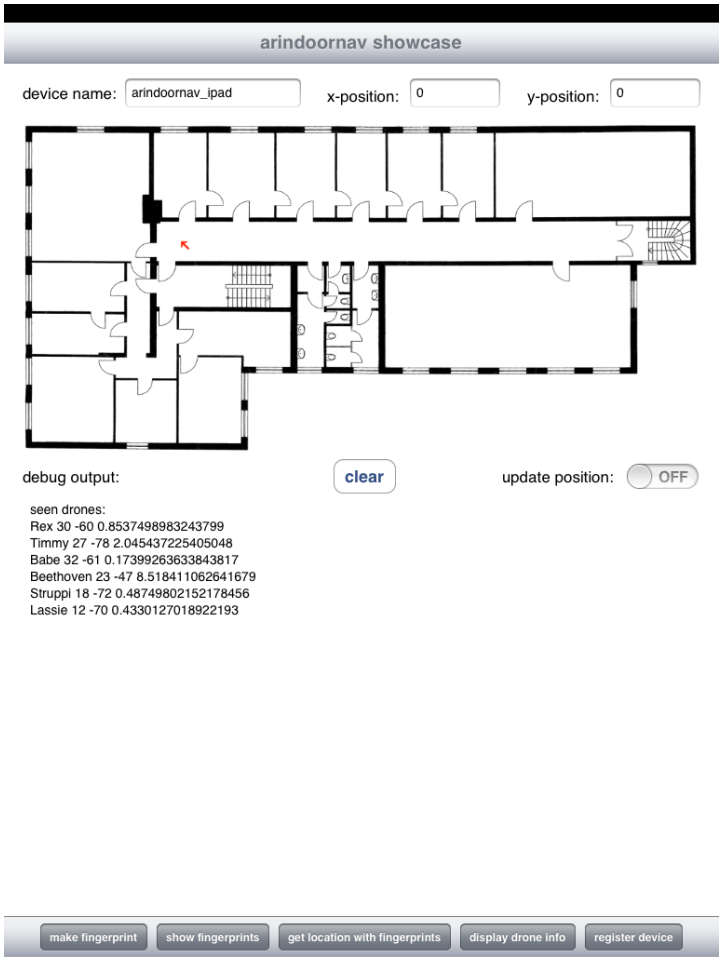
3. Servlet:
   - Container: Tomcat
   - Schnittstelle zu Clients
   - Bereitstellung von „WebServices“:
     - Position bestimmen
     - Fingerprint setzen
     - Fingerprint-Positionen abfragen
     - gemessene Signalstärken eines Clients abfragen
     - neuen Client registrieren
   - Kommunikation mit Kern über Java RMI

Clients

- künstliche Generierung von WiFi Paketen
  - Android: nach WLANs scannen
  - iOS: schwierig, da aktives scannen ja verboten ist
- Kommunikation mit Server über „WebServices“
- Anzeigen der Position auf einer Karte
- diverse Servicefunktionen:
  - Fingerprints setzen/anzeigen
  - Drohnenstatus
  - Gerät beim Server registrieren
  - ⋮

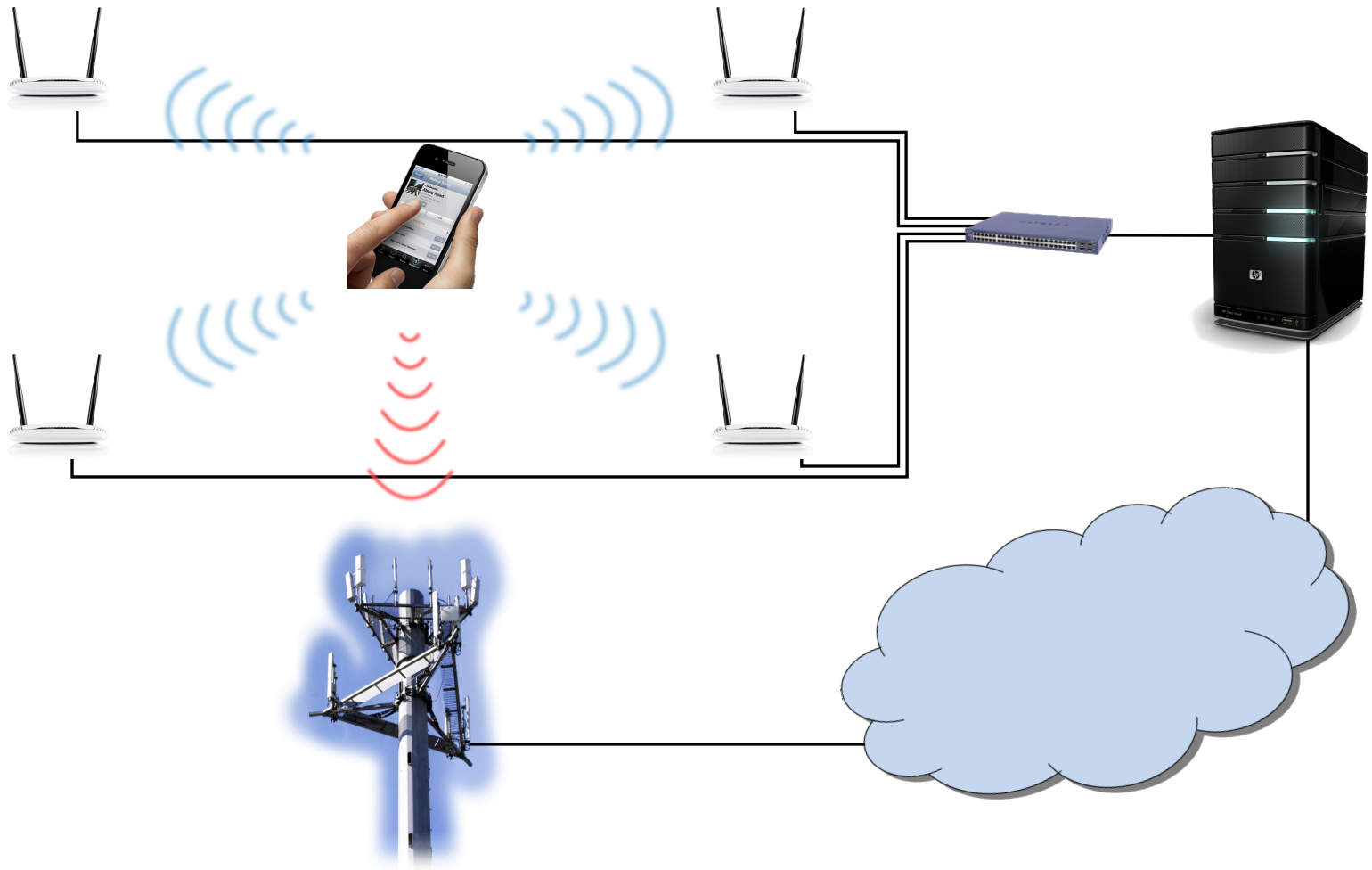
Screenshots

Samsung Galaxy S

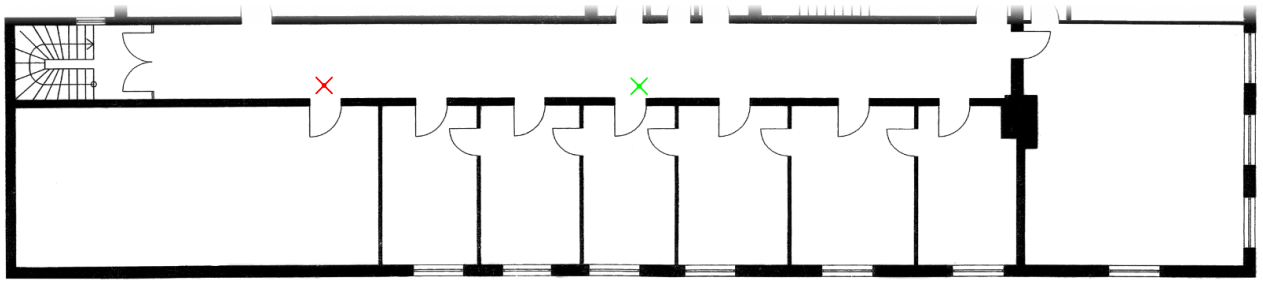


iPad 2

Topologie

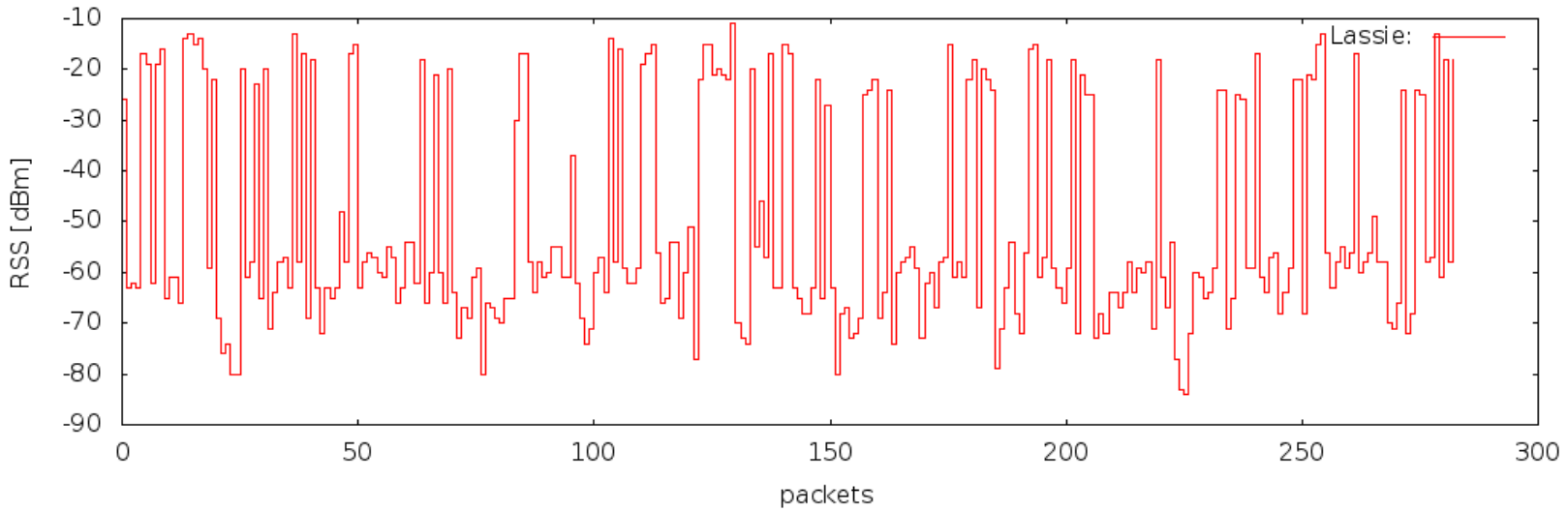


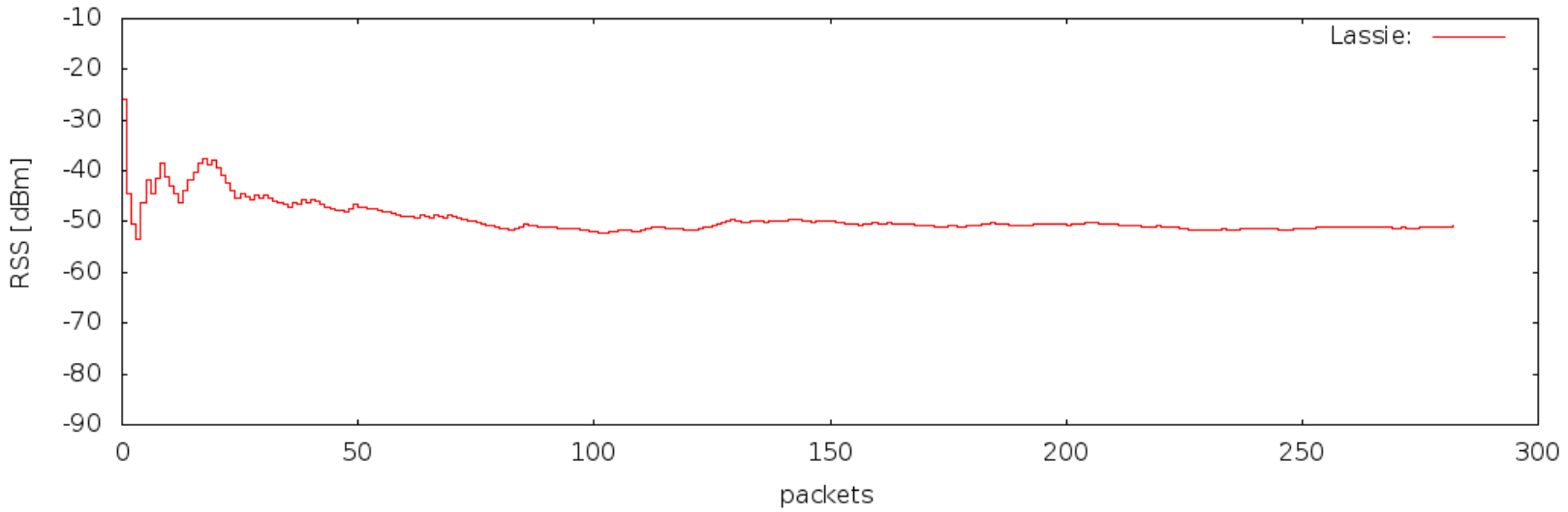
Messbedingungen



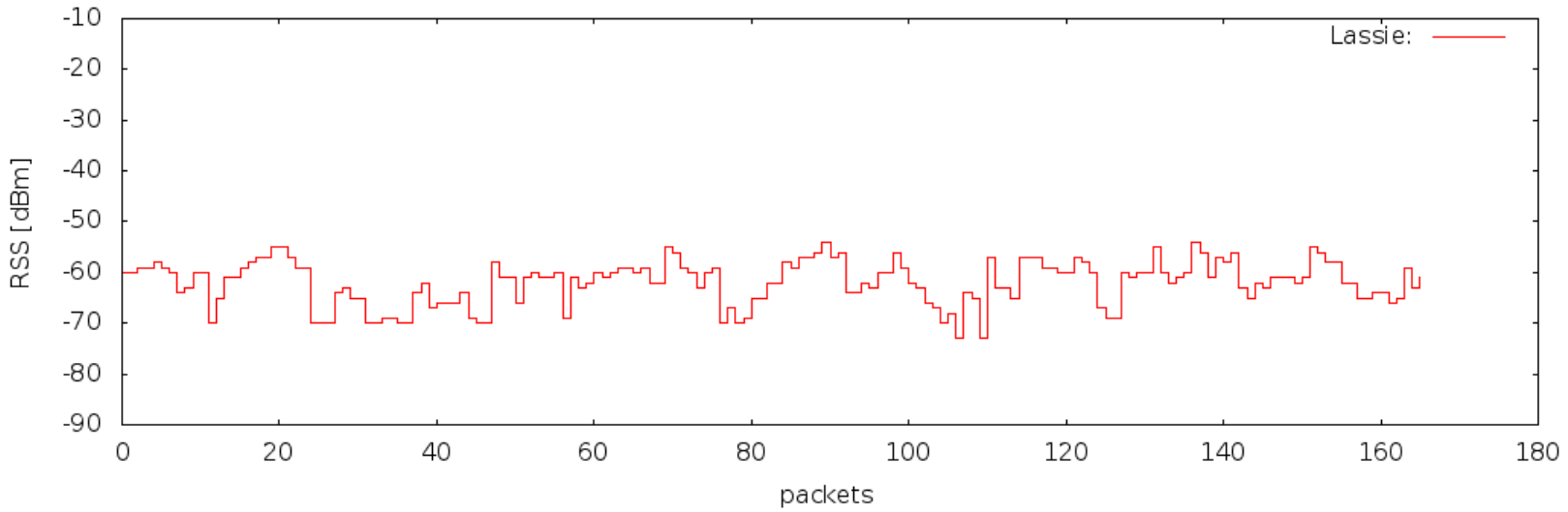
- Aufbau: Drohne und Client in einem Hausflur
- Laufzeit: 30sec
- Client: Samsung Galaxy S
- Software: WLAN-Scan alle 500ms

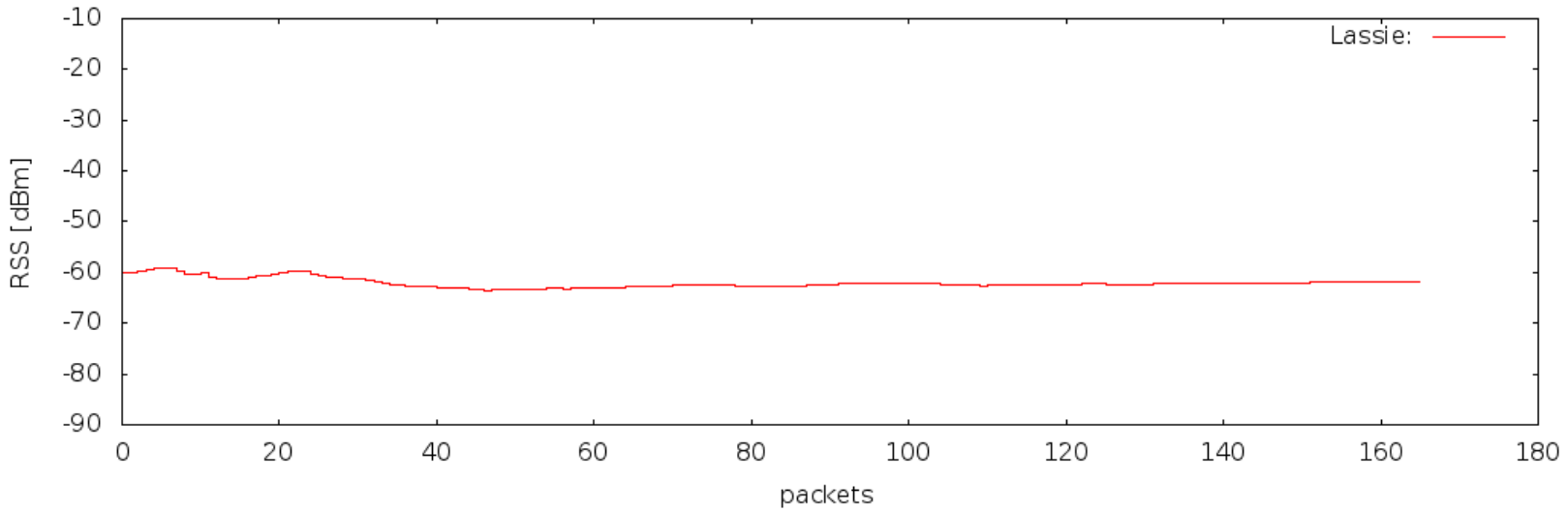
Abstand zur Drohne: wenige cm



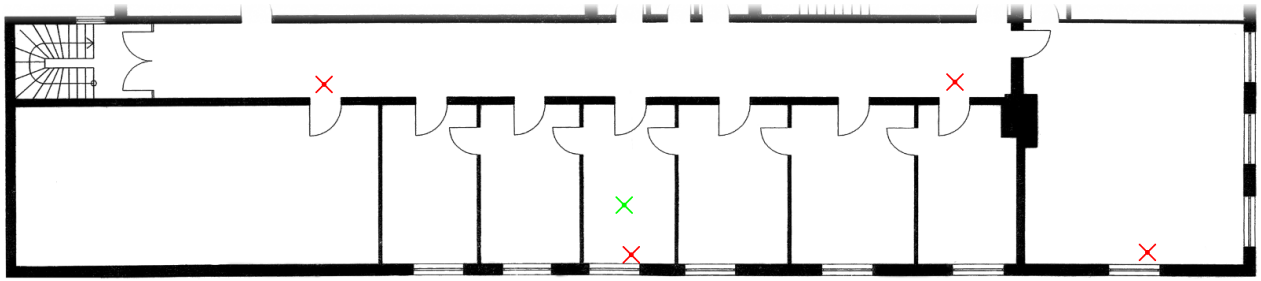


Abstand zur Drohne: 10m



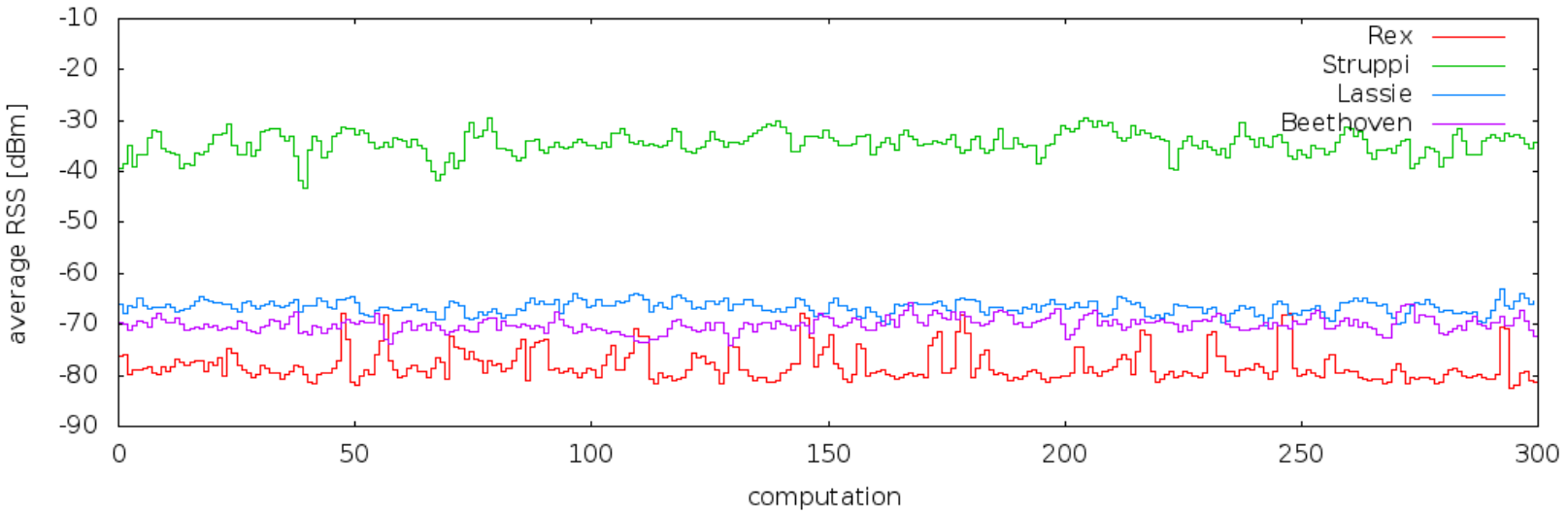


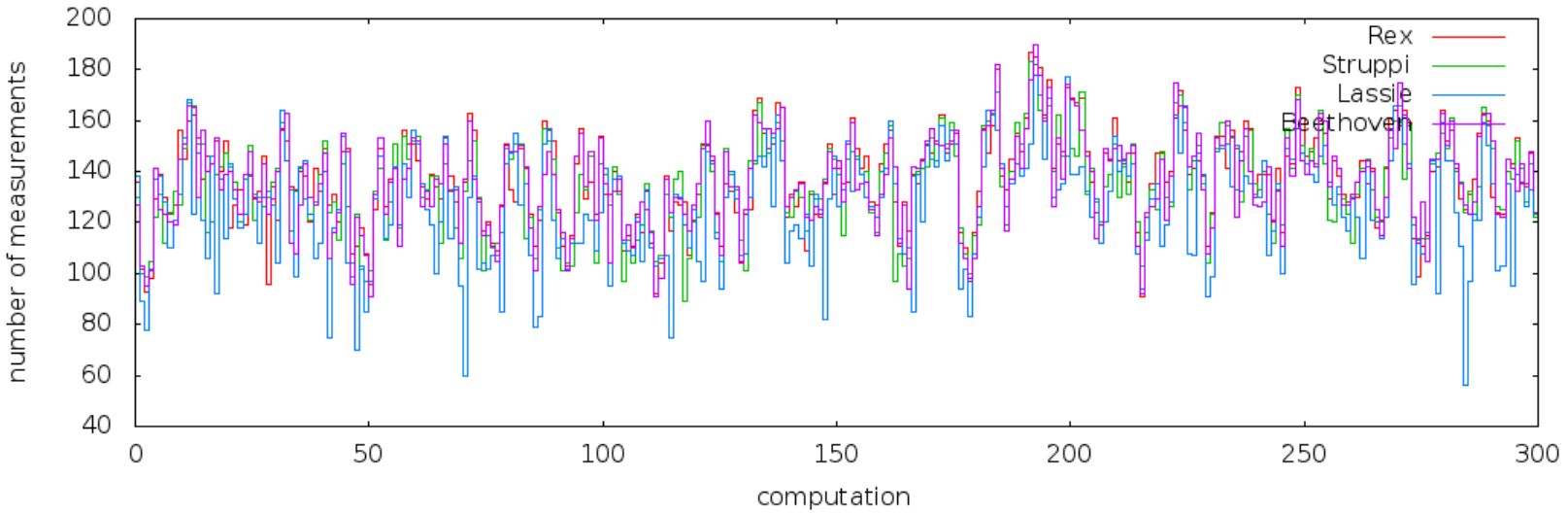
Messbedingungen



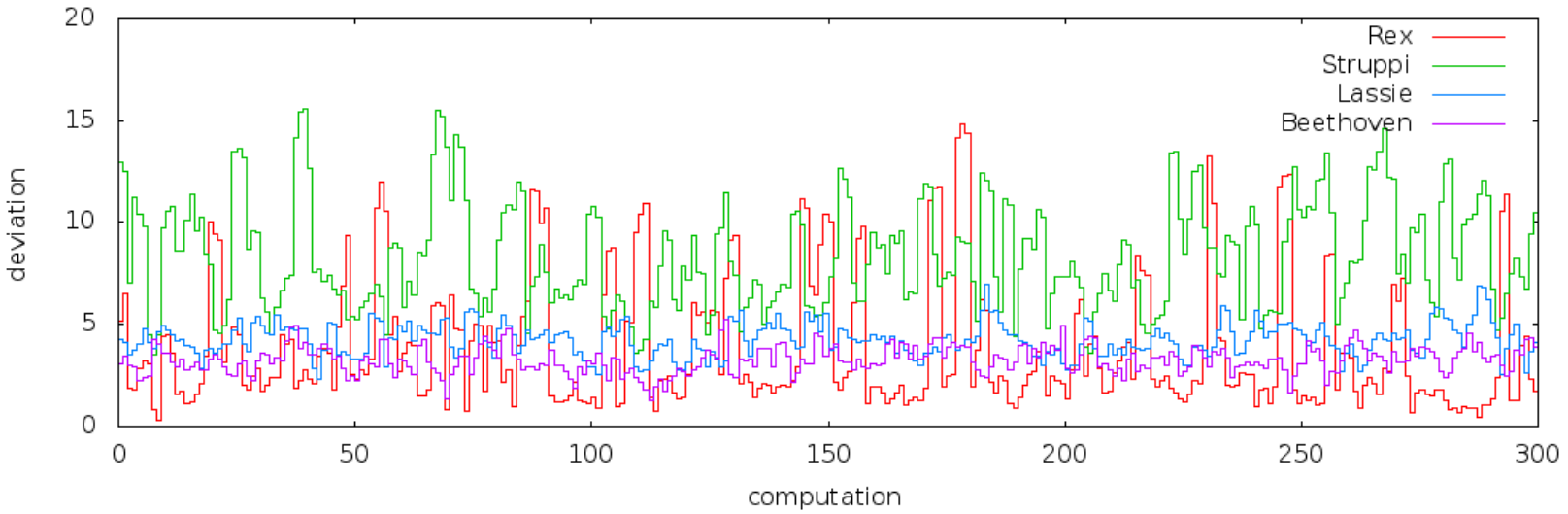
- Aufbau:
  - Drohnen im Gebäude verteilt (gleiche Etage)
  - Client im gleichen Zimmer wie „Struppi“, Abstand 1.5m
  - „Beethoven“ und „Lassie“ im Flur; 10m bzw. 11m entfernt
  - „Rex“ 14m entfernt
- Laufzeit: 20min (15 Berechnungen pro Minute)
- Client: Samsung Galaxy S
- Software: WLAN-Scan alle 500ms

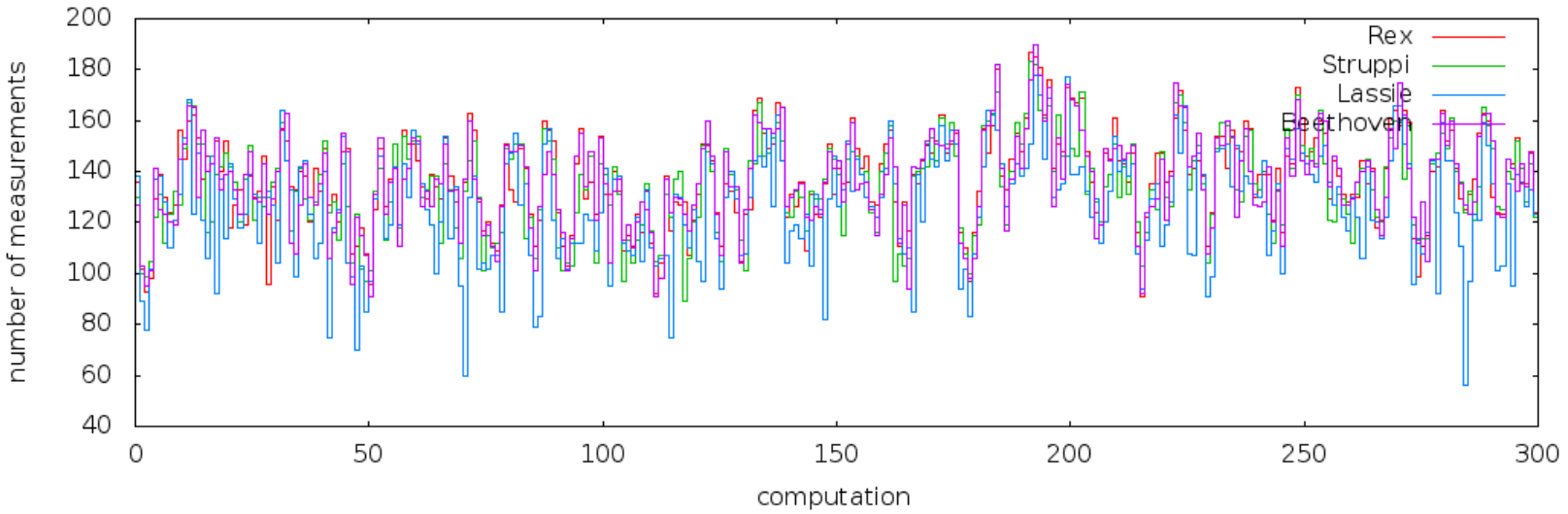
Mittelwert über 10 sec



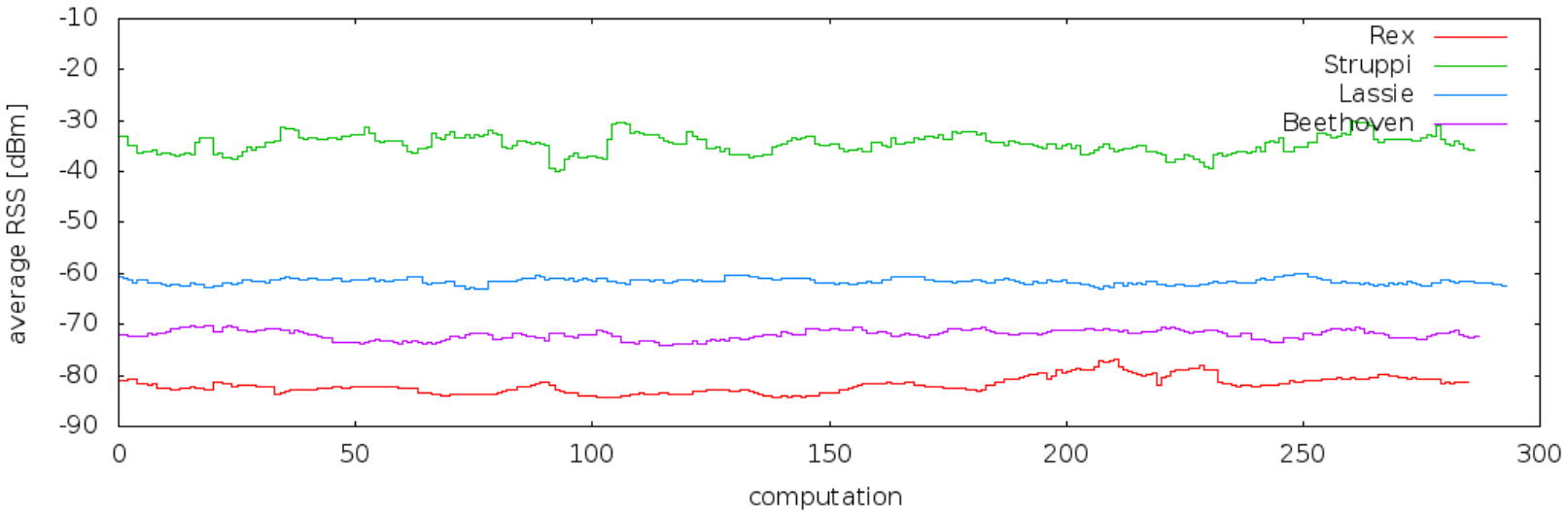


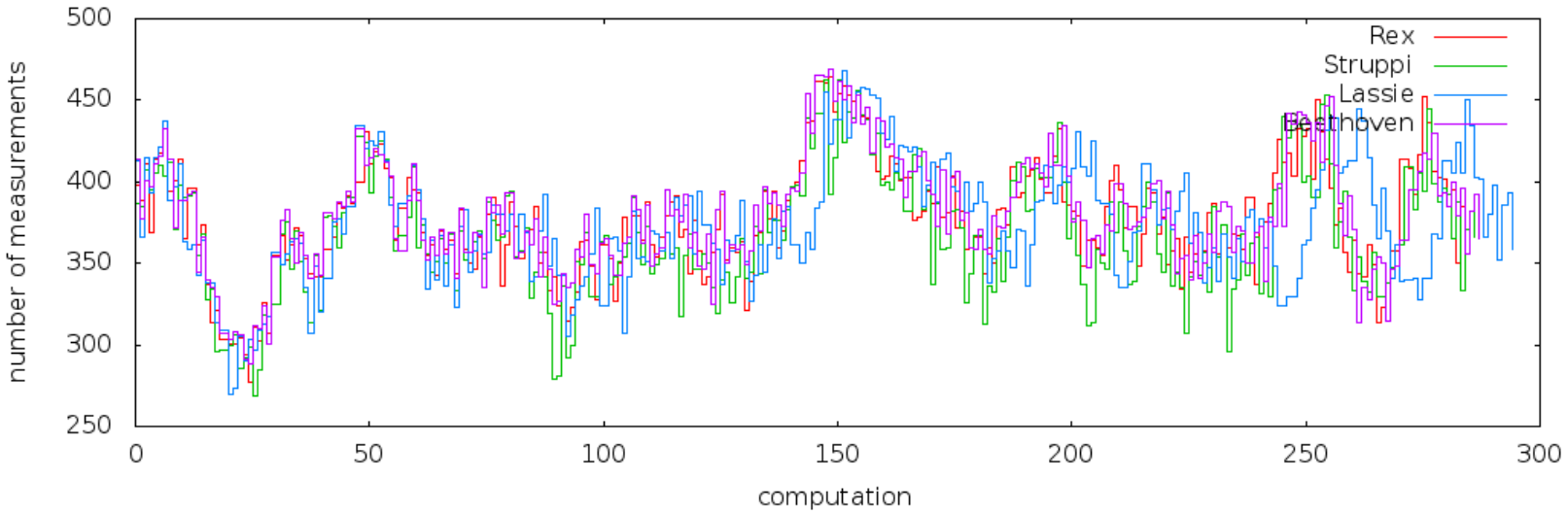
Abweichung über 10 sec



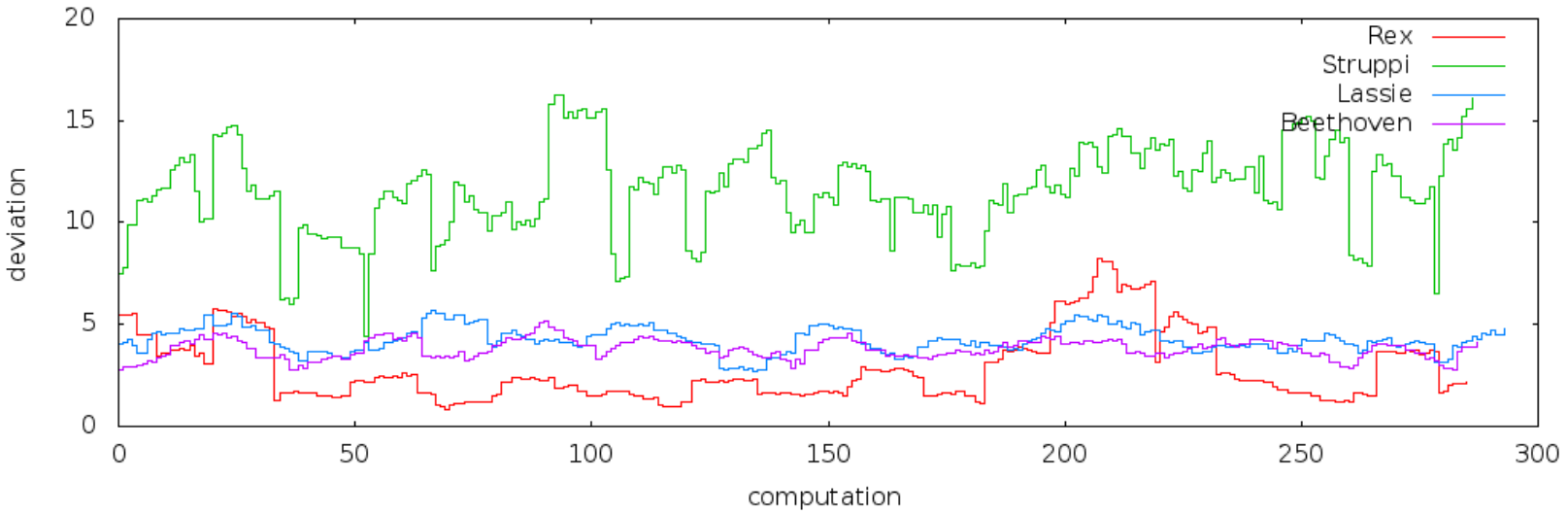


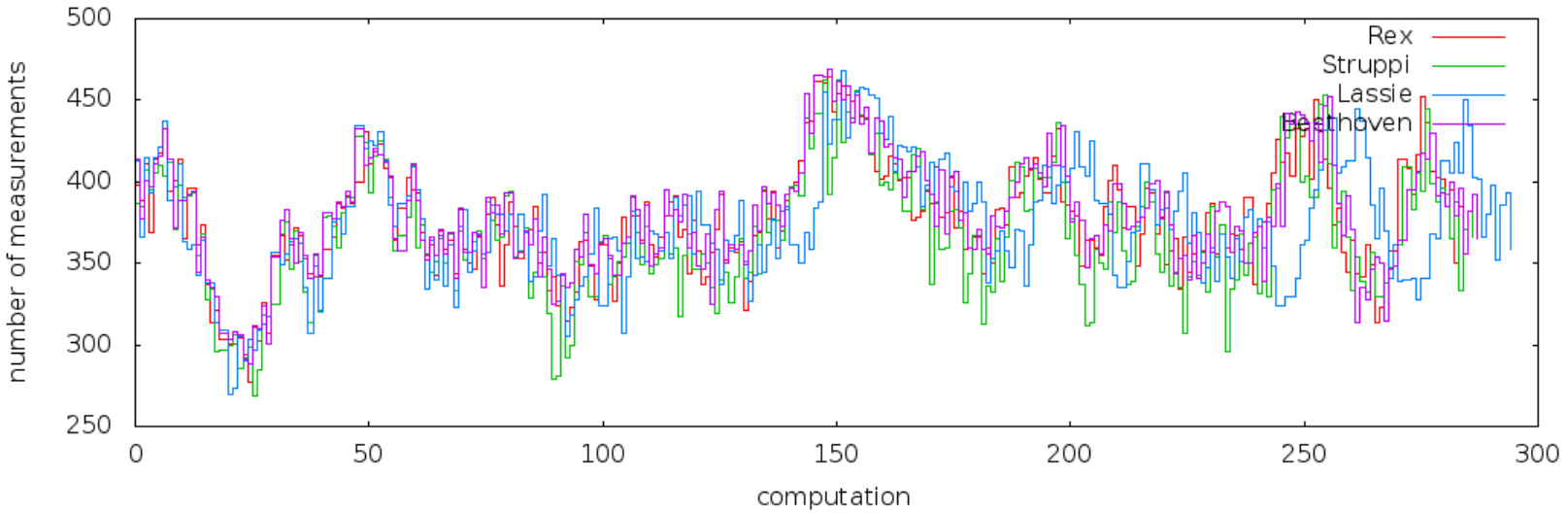
Mittelwert über 30 sec





Abweichung über 30 sec





Vorbereitungen

- Erstellung sogenannter „Fingerprints“ als Referenzwerte
  - gemessene Signalstärken aller Drohnen in Empfangsreichweite
  - Position auf einer (in der App angezeigten) Karte
- Ermittlung eines Skalierungsfaktors (Zuordnung: $px \rightarrow m$)
- Registrieren der MAC-Adresse des Clients

Algorithmen

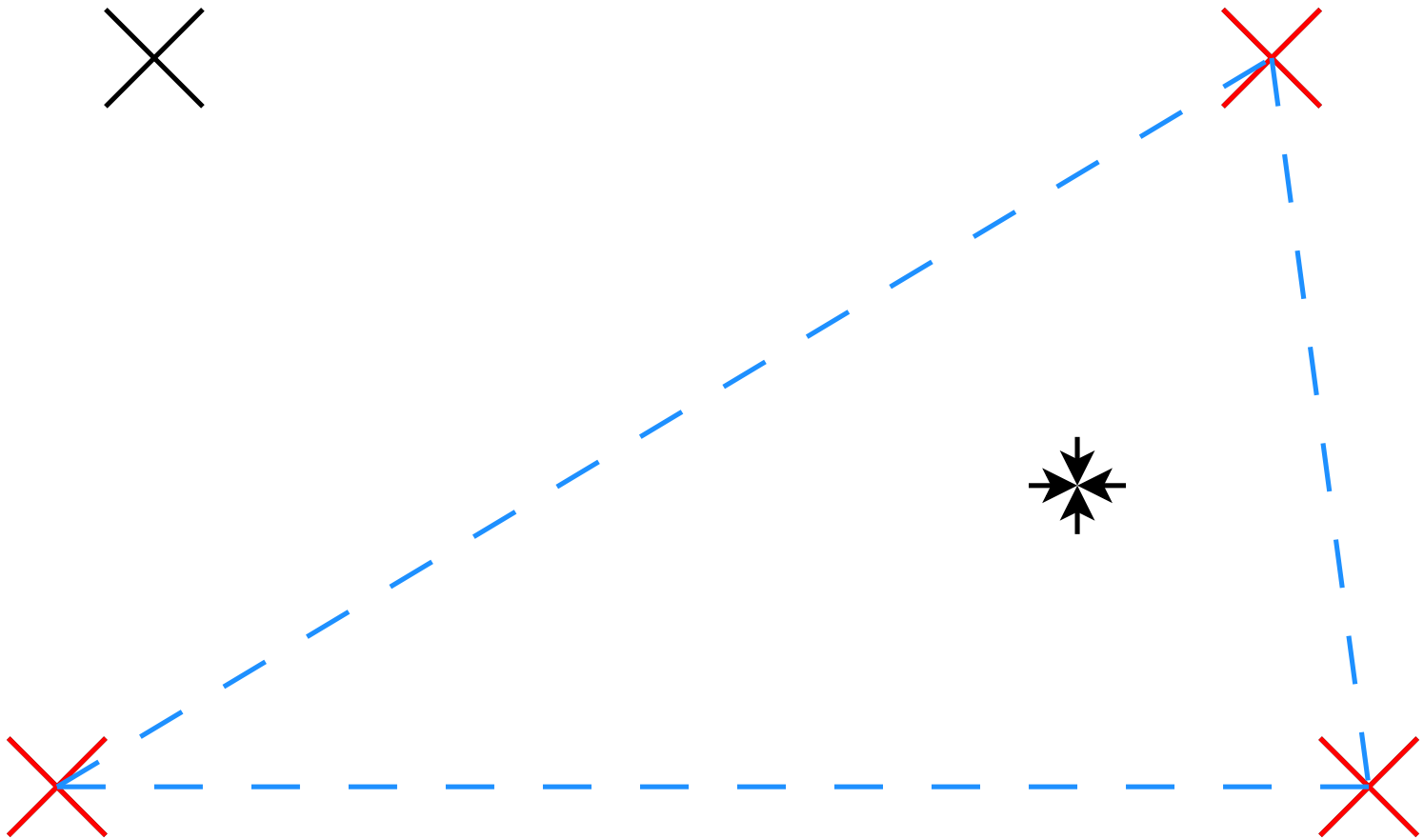
- Mittelwerte der Messdaten der letzten 10-30 Sekunden
- Ermittlung der am besten „passenden“ Fingerprints
- Berechnung von Punkten zwischen jew. zwei Fingerprints anhand der Abweichung der gemessenen Signalstärken
- Schwerpunkt des entstehenden Vielecks

Berechnung I

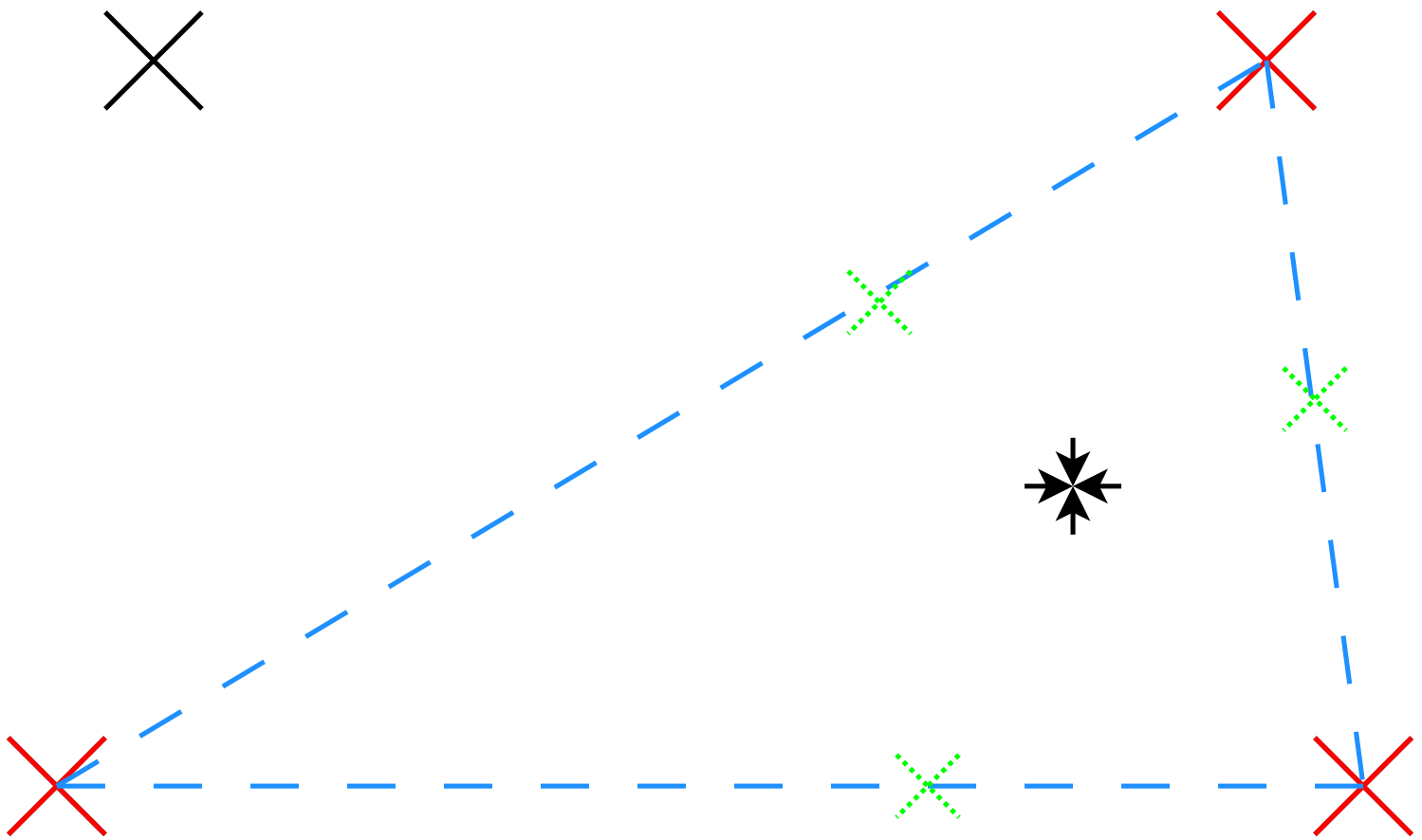




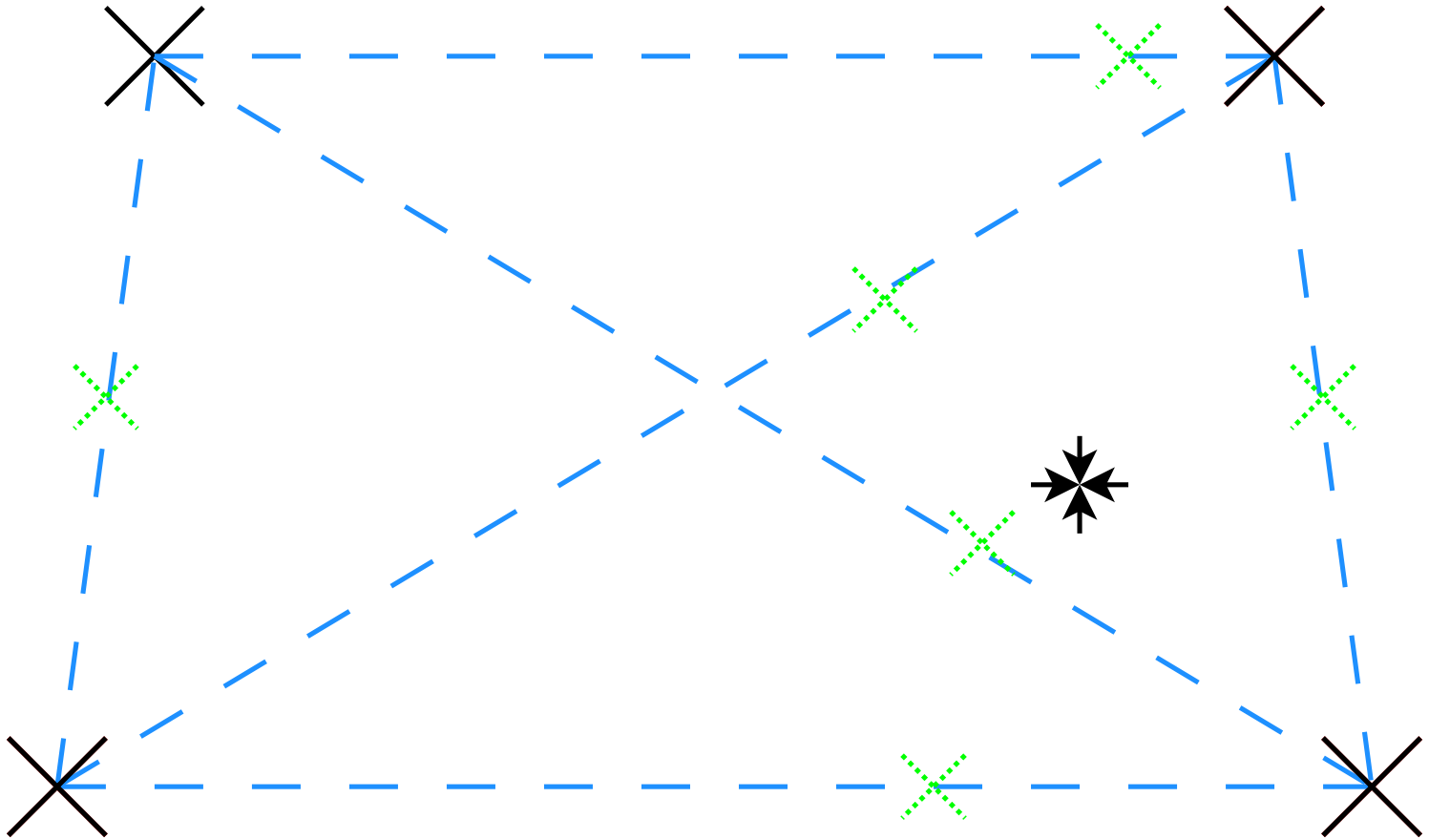
Berechnung II



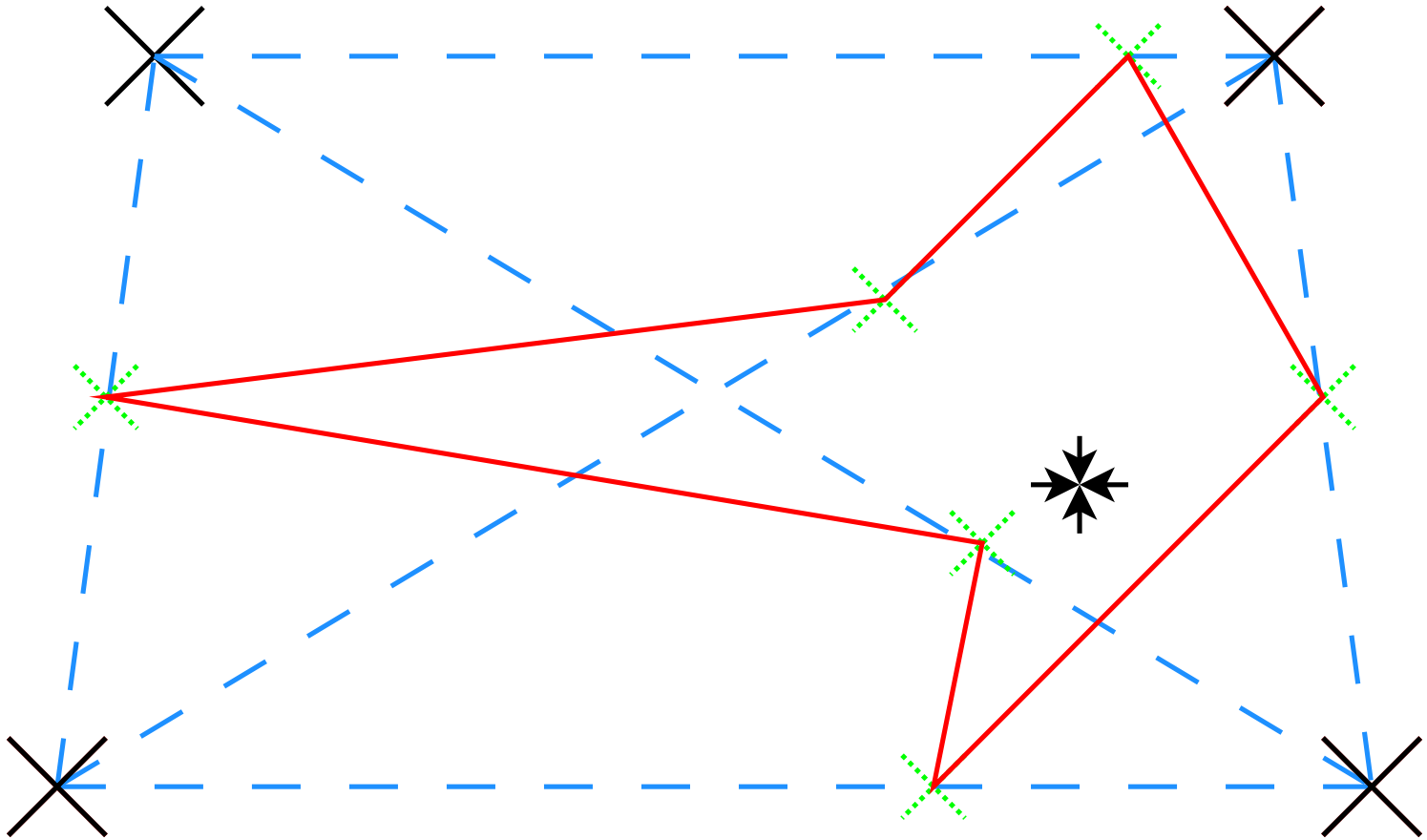
Berechnung III



Berechnung IV



Berechnung V



Ergebnisse

- Topologie:
  - funktionsfähig
  - nur mit wenigen Clients getestet
  - bislang keine Clientauthentifizierung
- Lokalisierungs-Alorithmen:
  - schwierig zu implementieren aufgrund starker Schwankungen der Messwerte
  - momentane Genauigkeit: ca. 5m
  - relativ zuverlässig
- Clients:
  - Android: voll funktionsfähig
  - iOS: Probleme bei der Generierung von WLAN-Paketen

persönliches Fazit

- sehr spannendes und anspruchsvolles Thema
- vielversprechende Zwischenergebnisse
- Apple nervt ☺

Vielen Dank für die Aufmerksamkeit.

Fragen?