Technische Universität Darmstadt

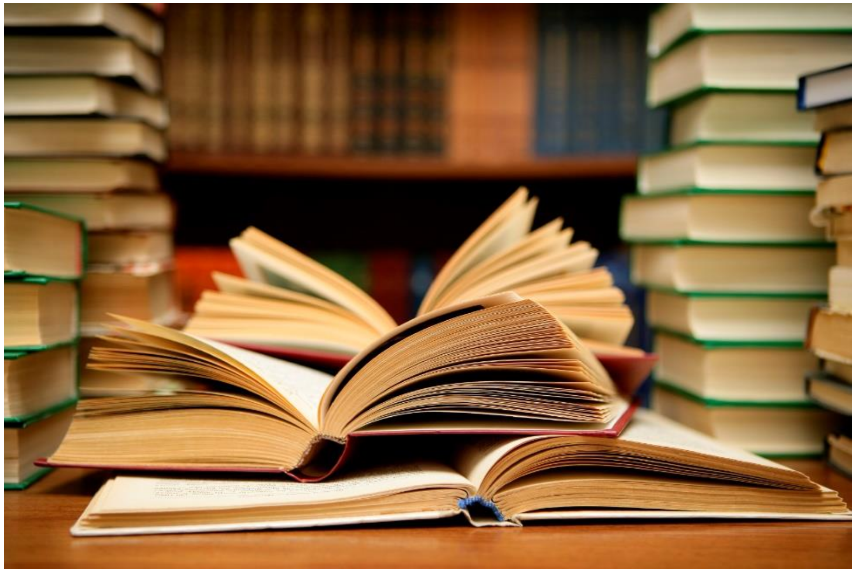
Telecooperation Lab
Prof. Dr. Max Mühlhäuser

TK3: Ubiquitous Computing

Chapter 1: Devices & Sensing
Part 3: People-centric Sensing
Lecturer: Dr. Immanuel Schweizer

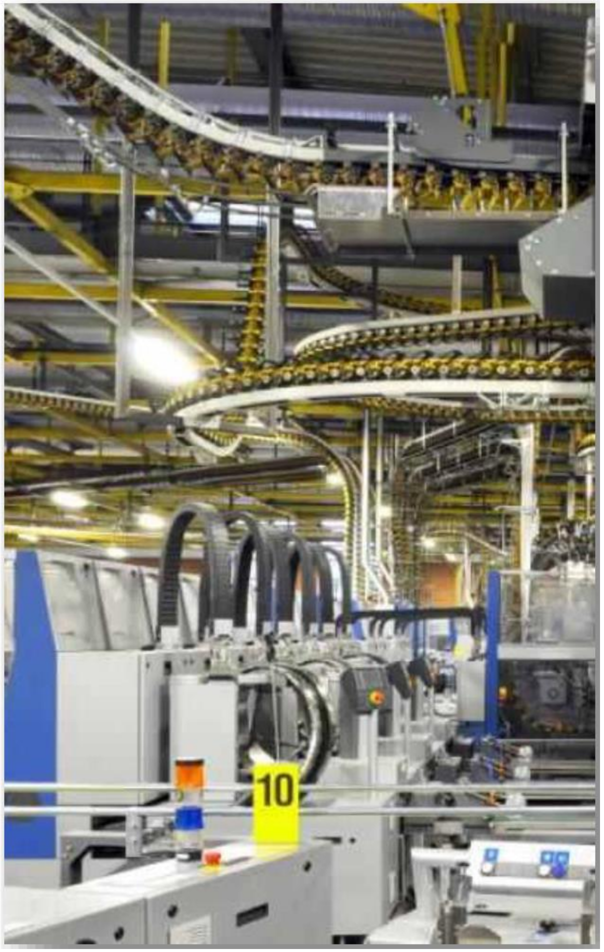
Sources

- http://www.nii.ac.jp/userimg/nii-borcea-lect4.pdf
- http://tech.brookes.ac.uk/frmitchell/u08971/lectures/week-7-sensor-slides
- https://www.uni-weimar.de/medien/wiki/images/Zeitmaschinen-smartphonesensors.pdf
- http://slideplayer.com/slide/3098588/



Recap

- Embedded Systems
- Sensor Nodes
- Wireless Sensor Networks
- Cyper-Physical Systems



CPS & UC

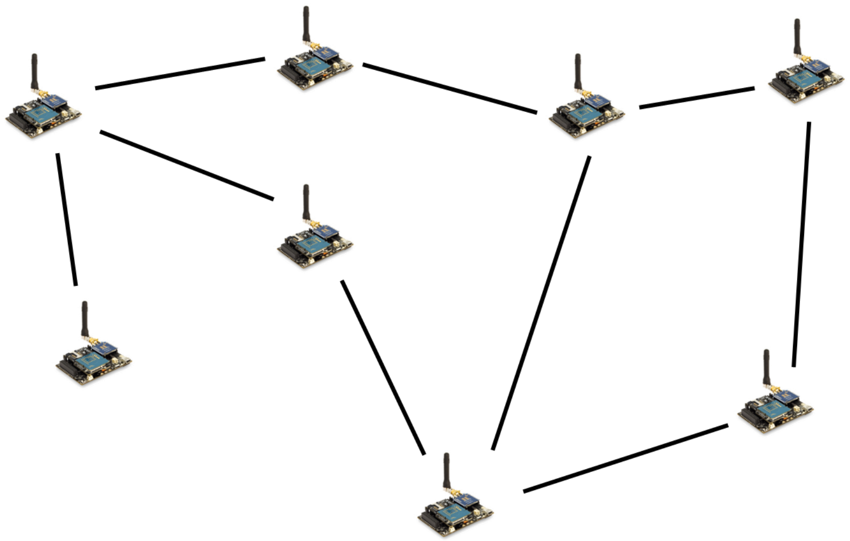
TECHNISCHE UNIVERSITÄT DARMSTADT

- Fast reaction times
  - Real-time constraints
- Humans out of the loop
- Safety, security

- Same building blocks
- Enabled by connectivity & smaller, more powerful devices

Recap: Characteristics

- Small-scale, short lived, mostly-static
- Mobility not an issue
- Application-specific
- Very energy-constrained
- Multi-hop wireless
- Humans out of the loop



Now what?

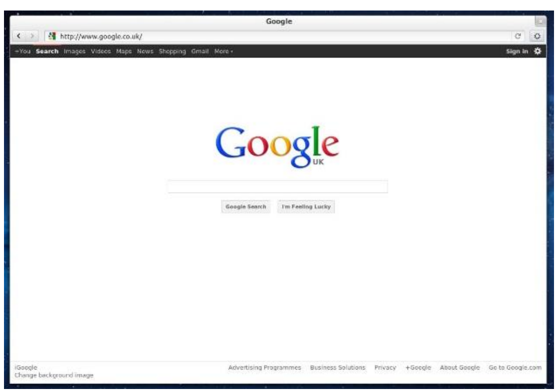
People-centric Sensing

- Sensing where the users are
- Cost of the device is paid by the **user**
- **Energy** and **maintenance** done by the user
- **Ubiquitous** devices packed with sensors
  - Mobile phones
  - Cars
- **Communication**
- Mobility
  - Coverage problem
- **Non-controllable**
  - Do I **pay** users?
- **Incentive** to waste time and energy
- **Personal** (gain) vs. **community** (gain)

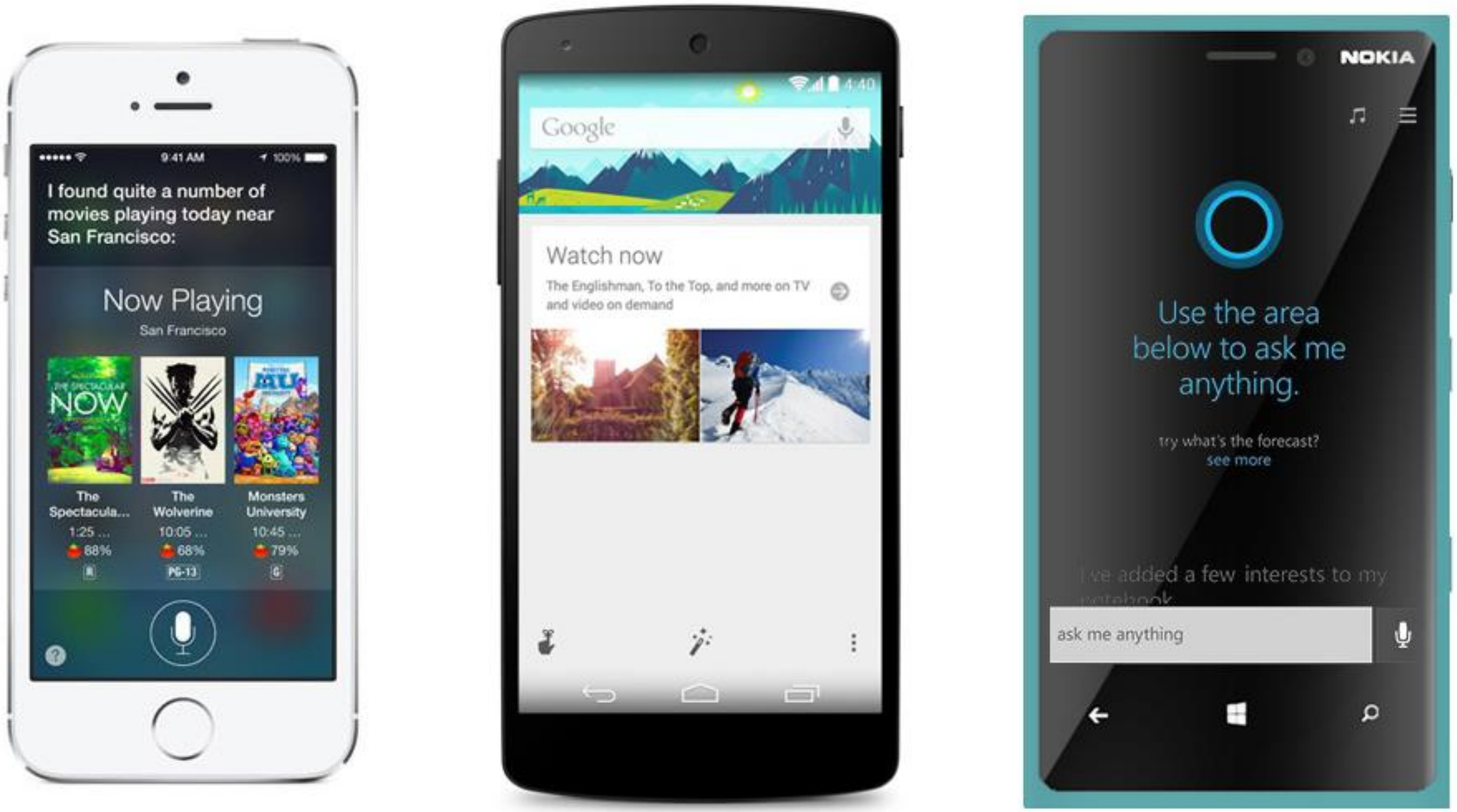
Applications

- Personal Sensing
  - Focus on individual benefit
  - E.g., health apps
- Peer Sensing
  - Small group of users
  - E.g., urban games
- Utility Sensing
  - Provides utility for community
  - E.g., Noise mapping

Personal Sensing

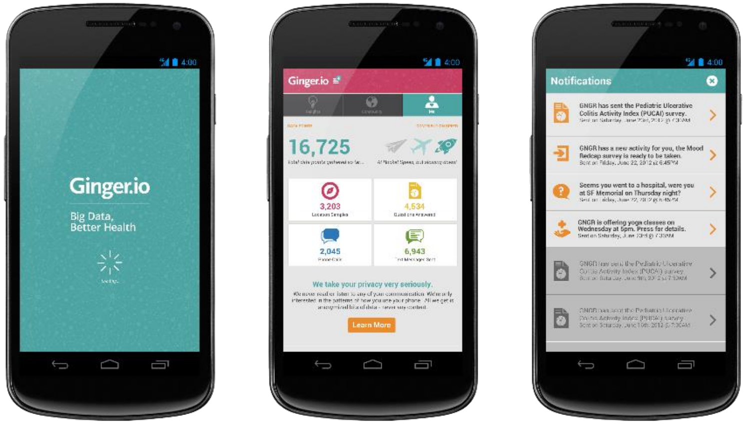


Personal Assistance

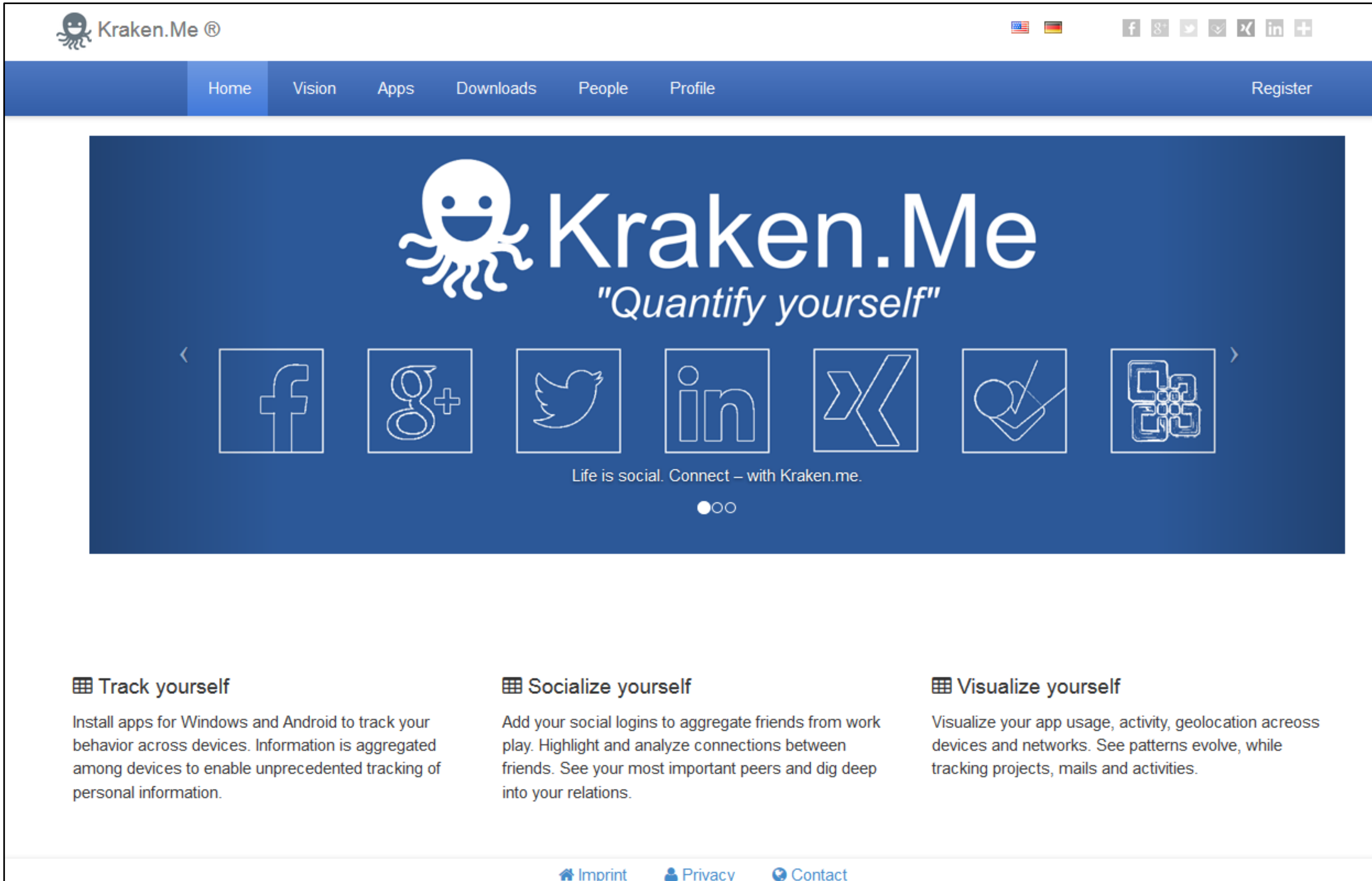


Platforms & Projects

- DeviceAnalyzer
  - Detailed device status every 5 minutes
- Funf
  - Rapid prototyping
- Ginger.io
  - Health insights from mobile data
- AWARE
  - Powerful instrumentation toolkit

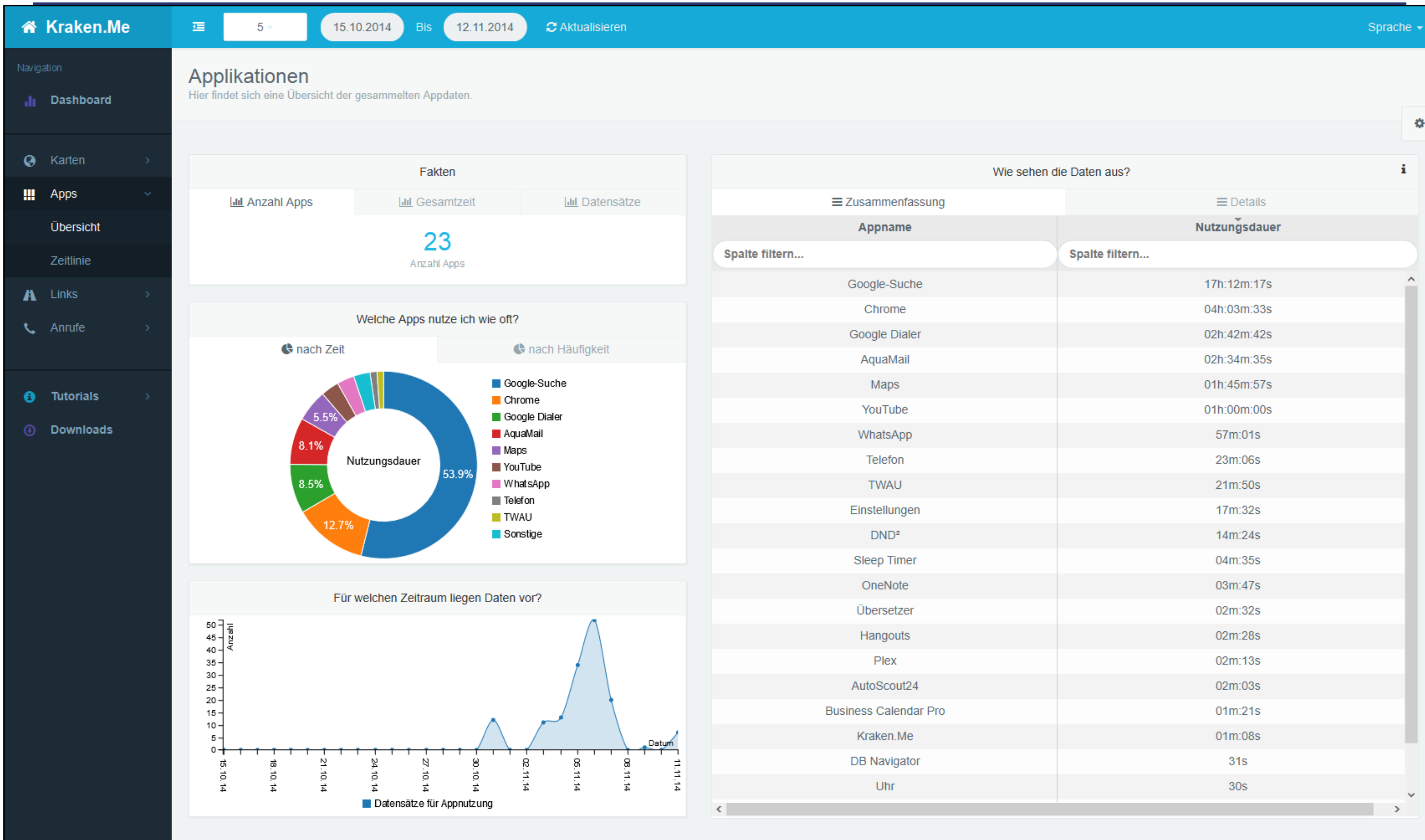


Kraken.me


Kraken.Me ®
Home
Vision
Apps
Downloads
People
Profile
Register
Kraken.Me
"Quantify yourself"
Life is social. Connect – with Kraken.me.
Track yourself
Install apps for Windows and Android to track your behavior across devices. Information is aggregated among devices to enable unprecedented tracking of personal information.
Socialize yourself
Add your social logins to aggregate friends from work play. Highlight and analyze connections between friends. See your most important peers and dig deep into your relations.
Visualize yourself
Visualize your app usage, activity, geolocation acreoss devices and networks. See patterns evolve, while tracking projects, mails and activities.
Imprint
Privacy
Contact

Kraken.me – Applications

TECHNISCHE UNIVERSITÄT DARMSTADT



User Awareness

- What did I do yesterday?
- Which documents did I edit?
- How much did I walk / run / etc.?
- How long did I use my smartphone / desktop?
- Which websites did I browse?

- Does awareness change behavior?

Peer Sensing

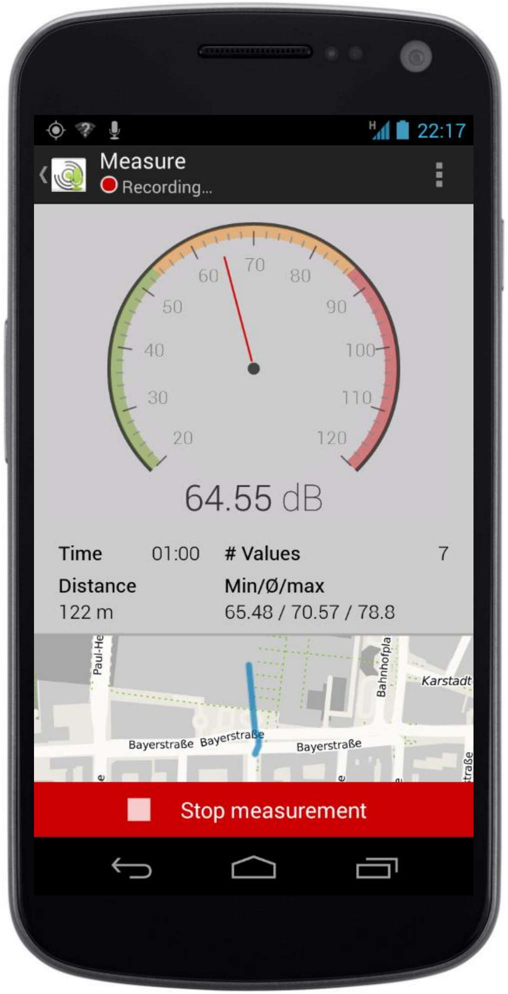
- Gaming
  - AR game – Ingress
- Peers defined by members of the virtual world
- Shared personal sensing
  - Community running platforms

Utility Sensing

- Provide community benefit
- Environmental data
  - Forest fires, parking spots, traffic monitoring
- Documenting public events, social media
- Health research

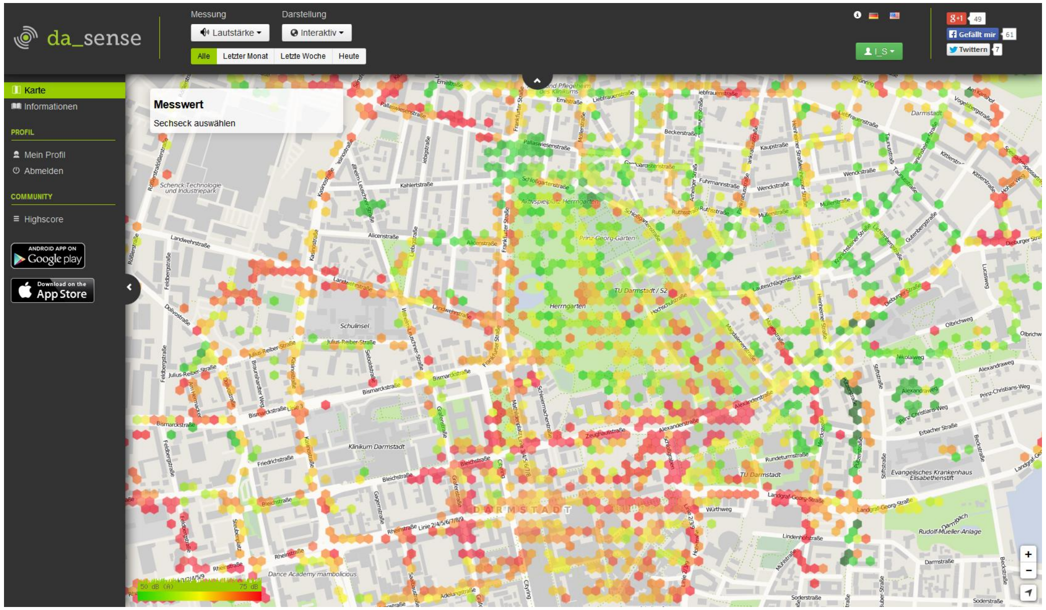
Utility Sensing

TECHNISCHE UNIVERSITÄT DARMSTADT



Noisemap

- Noise pollution
  - dB(A)



Characteristics

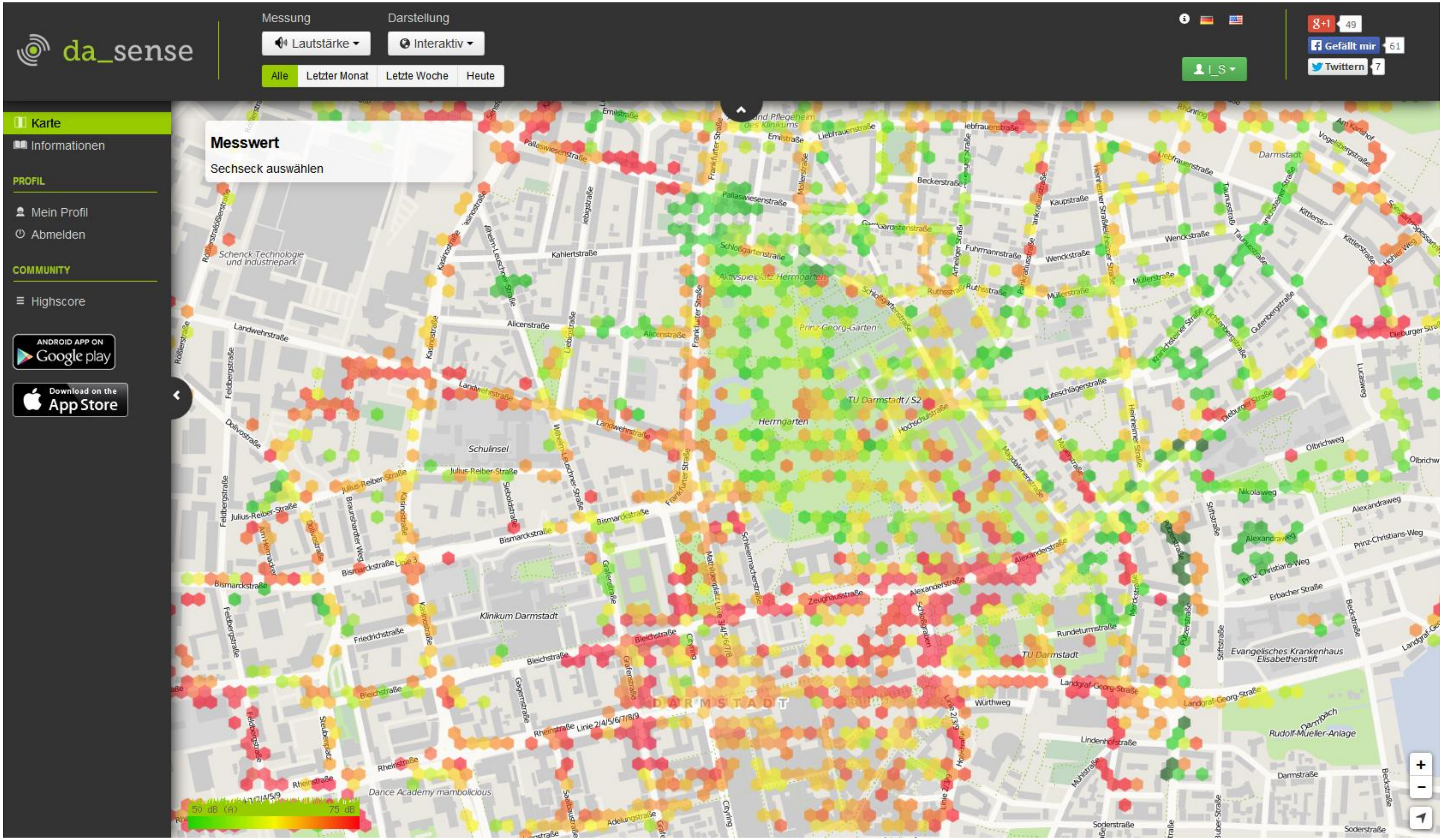
- Small-scale, short lived, mostly static
  Large, long lived, mobile
- Mobility not an issue
- Application-specific
- Multi-hop wireless
- Very energy-constrained
- Humans out of the loop

Scale


Globe
Critical Infrastructure
Municipal command and control center
Cloud and services
Home energy management
Smart grid
Responsive store
Hospital optimization
Communication network optimization
Intelligent hospital
Intelligent highway
Intelligent digital signage
City
Building / Organization
Team
Individual

Large-scale Incidents

City-scale


da_sense
Messung
Lautstärke
Darstellung
Interaktiv
Alle
Letzter Monat
Letzte Woche
Heute
I_S
49
Gefällt mir
61
Twittern
7
Karte
Informationen
PROFIL
Mein Profil
Abmelden
COMMUNITY
Highscore
ANDROID APP ON Google play
Download on the App Store
Messwert
Sechseck auswählen
Herrngarten
Prinz-Georg-Garten
TU Darmstadt
Klinikum Darmstadt
Schulinsel
DARMSTADT
50 dB (A)
75 dB

Characteristics

- Small-scale, long-lived, mostly-static
- Large-scale, short-lived, mobile
- Mobility not an issue
- Mobility-by-design
- Application-specific
- Very energy-constrained
- Multi-hop wireless
- Humans out of the loop

Mobility-by-design

- Unpredictable mobility
  - There are patterns
- Communication issues
  - Handoffs etc.
- Coverage
  - Trade-off between spatial and temporal

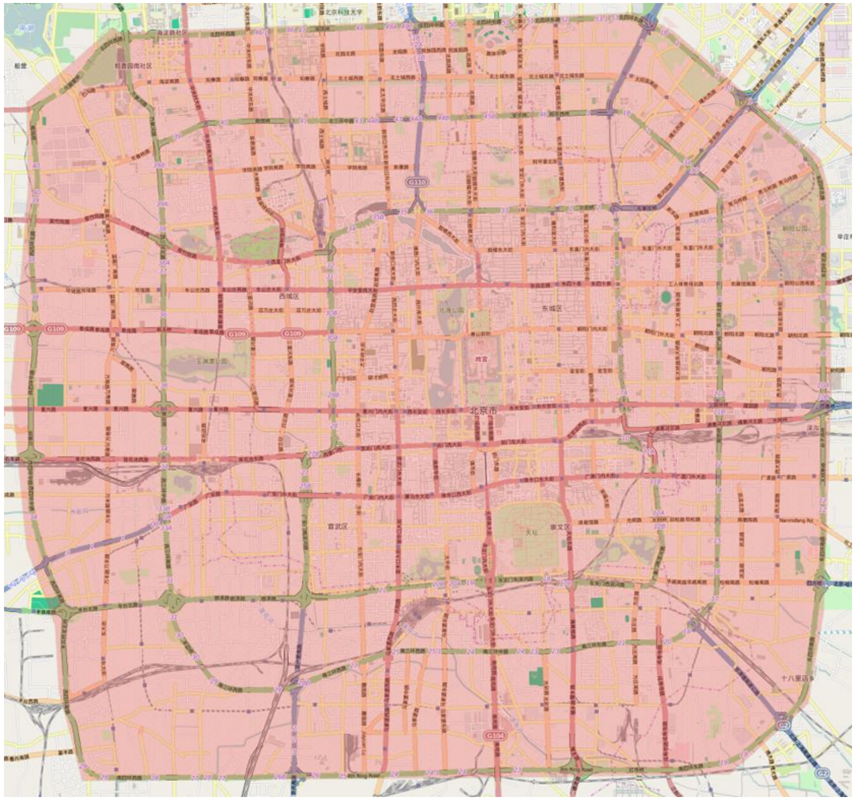
Coverage problem

- Definition: the measurements of quality of service that can be provided by a particular sensor network

- Area coverage: Cover a given area
- Point coverage: Cover a set of (static) targets
- Detectability: Determine the maximal support / breach paths that traverse a sensor field

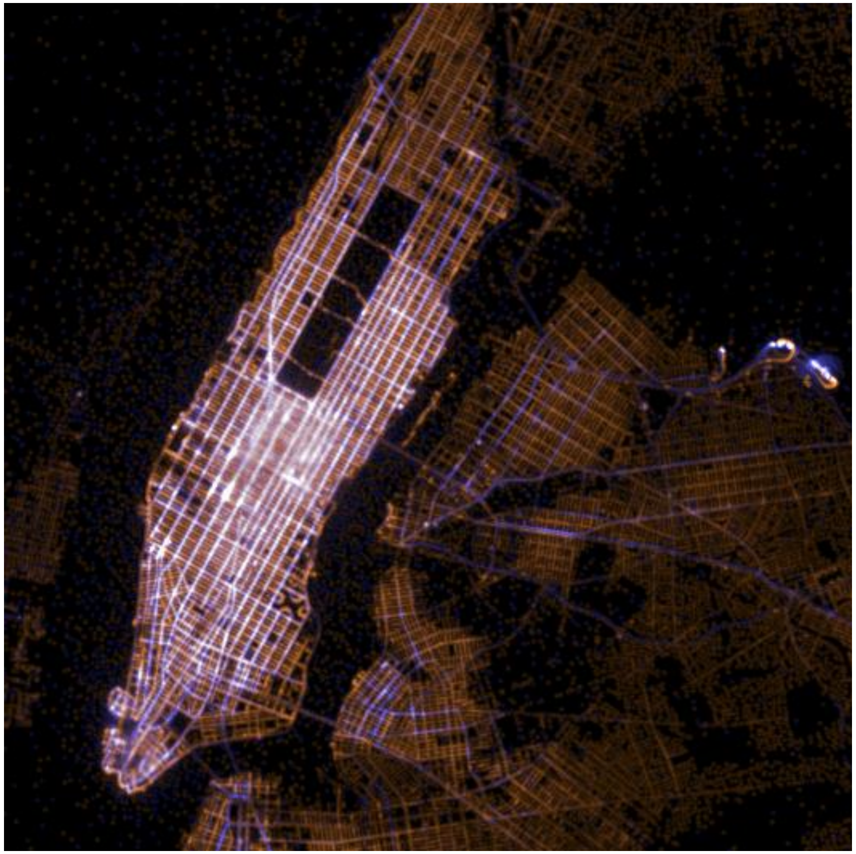
Coverage with Mobility

- Trade-off
  - Spatial and temporal resolution
- Pick the best users
  - Maybe limited budget
  - Query Optimization
- Temporal component
  - Users leave the system



Coverage with Mobility

- Study in NYC
  - Average time between visiting two parking spots
  - Rush hour: 1.5 minutes (10%)
- Study in Beijing
  - ~17 sensors = 90% spatial coverage

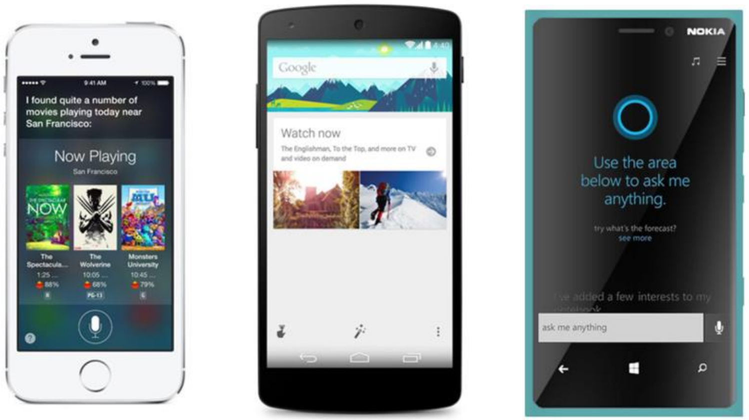


Characteristics

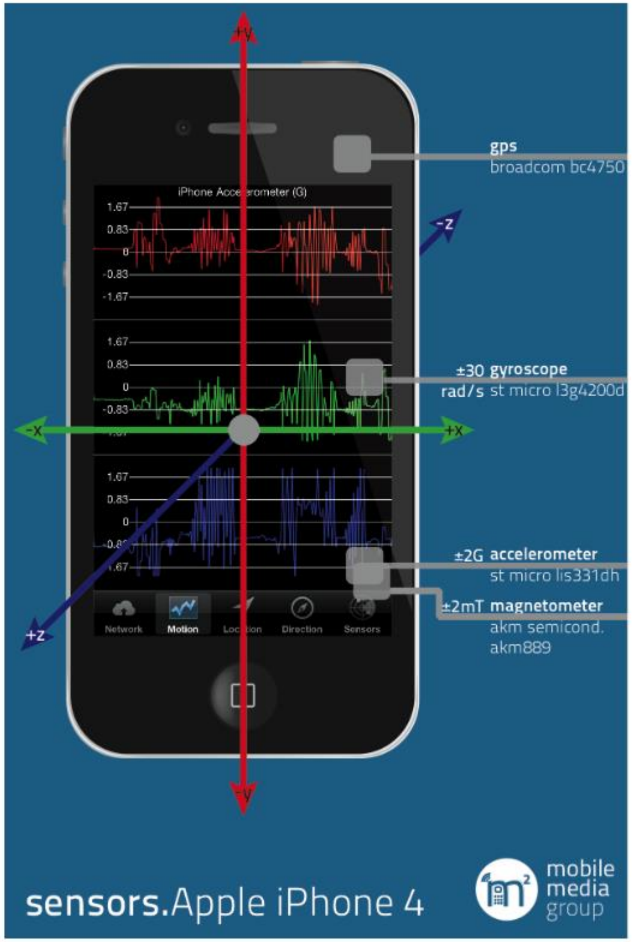
- Small-scale, short-lived, / Large-scale, long-lived, mostly-static / mostly-mobile
- Mobility-not-an-issue / Mobility-by-design
- Application-specific / Application-agnostic
- Very energy-constrained
- Multi-hop wireless
- Humans out of the loop

Application-agnostic

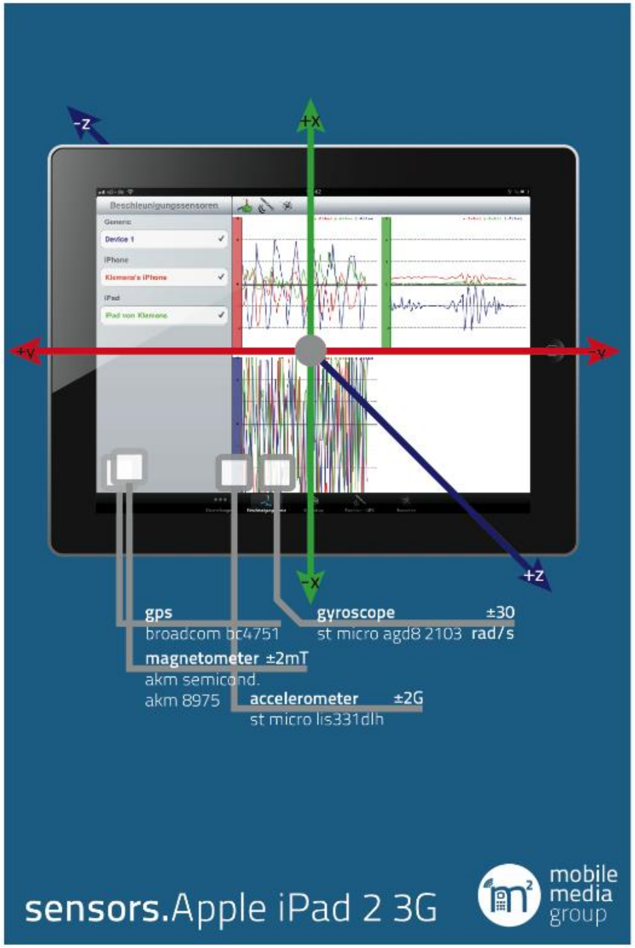
- Multi-purpose Operating systems
  - Android, iOS, Windows Phone
- App Store delivery model
  - Easy deployment and update of applications
  - But: Thousands of other apps
- Multiple build-in sensors



Sensors



+y
gps
broadcom bc4750
-z
±30 gyroscope
rad/s st micro l3g4200d
-x
+x
±2G accelerometer
st micro lis331dh
±2mT magnetometer
akm semicond.
akm889
+z
-y
sensors.Apple iPhone 4
mobile
media
group



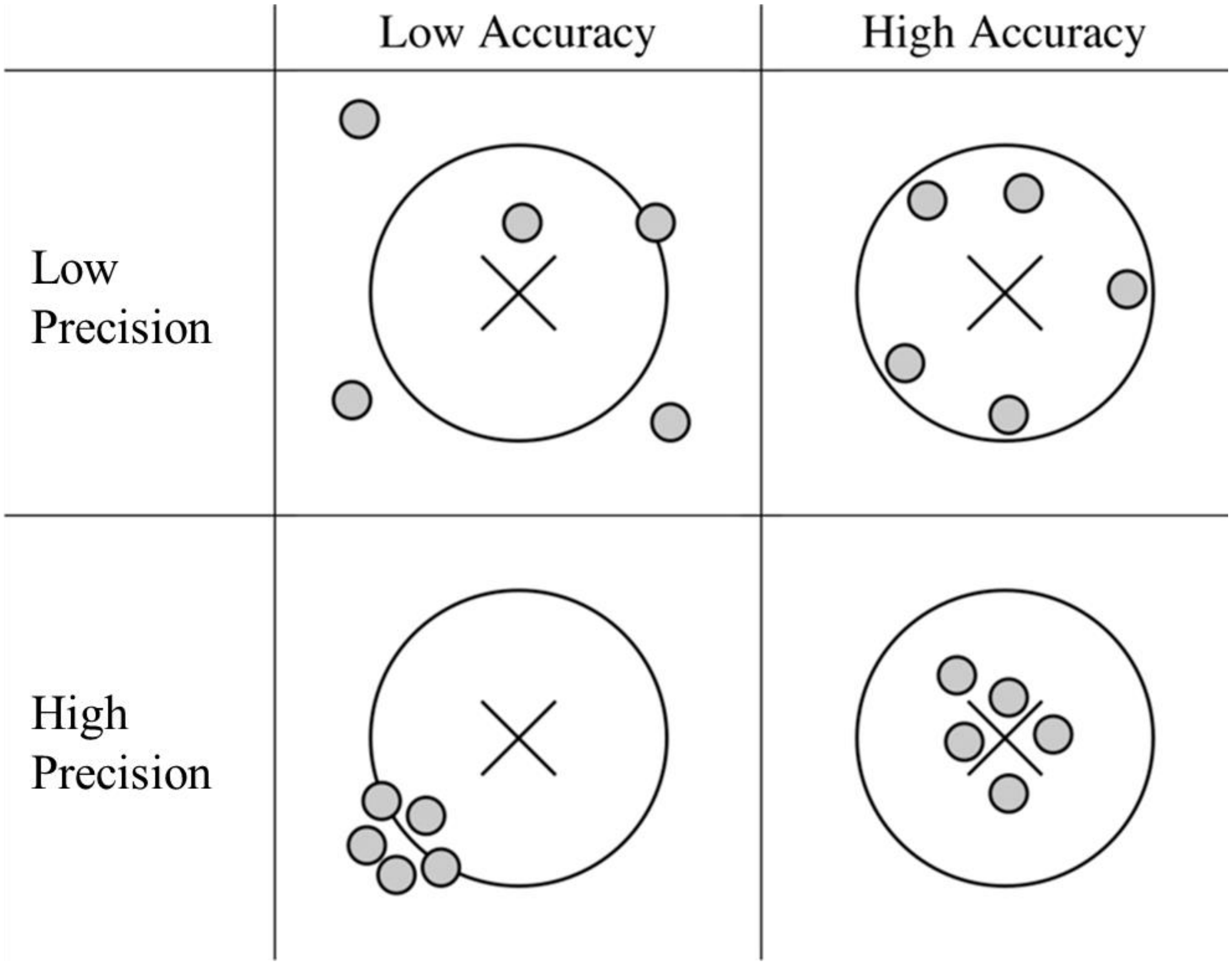
-z
+x
+y
-y
-x
+z
gps
broadcom bc4751
gyroscope ±30
st micro agd8 2103 rad/s
magnetometer ±2mT
akm semicond.
akm 8975
accelerometer ±2G
st micro lis331dlh
sensors.Apple iPad 2 3G
mobile
media
group

Sensor concepts

- Dynamic range: The range of values a sensor can measure
- Saturation: Where the stimulus is greater than the maximum value in the dynamic range or lower than the minimum
- Noise: the amount of random variance in the sensor
- Resolution: The smallest detectable difference between physical values.
  - In Android it is the smallest difference between numbers that can be reported by the sensor.
- Sampling rate: The reciprocal of the time between measurements.
  - In Android Sensor.getMinDelay() reports the minimum possible delay and therefore the maximum sampling frequency

Precision vs. Accuracy

- Precision vs. Accuracy



Accelerometer

- Measures proper acceleration (relative to freefall)
- Units: m/s^2 or g
- Smartphones: High precisions / low dynamic range
  - iPhone 4: Range 2g, Precision: 0.018g

Example	G Force
Standing on earth	1g
Bugatti Veyron (0-100km/h)	1.55g
Space Shuttle	3g
Formula 1	5-6g
Death or serious injury	50g
Shock resistance (Omega watches)	5000g

Magnetometer

- Measures the strength of earth's magnetic field
- Units: Tesla [T]
- iPhone 4: Range 2mT

Example	G Force
Earth's magnetic field (equator)	31µT
Fridge Magnet	5mT
Strong Neodymium Magnet	1.25T
MRI system	1.5 – 3T

Sensors in Android

- Sensors are part of the Android SDK
  - Most smartphones have accelerometer, gyroscope, and proximity.
  - Many have a magnetic field sensor (compass)
- There are two classes of sensor – raw & synthetic.
  - Raw sensors are the actual sensors and correspond to actual hardware.
  - Synthetic sensors are not linked to specific hardware and may be a combination of physical sensors.
- GPS isn't classified as a sensor within Android

Hardware Sensors

Raw

- Sensor.TYPE_LIGHT
- Sensor.TYPE_PROXIMITY
- Sensor.TYPE_PRESSURE
- Sensor.TYPE_TEMPERATURE (deprecated – this measures CPU temp)
- Sensor.TYPE_ACCELEROMETER
- Sensor.TYPE_GYROSCOPE
- Sensor.TYPE_MAGNETIC_FIELD
- Sensor.TYPE_RELATIVE_HUMIDITY
- Sensor.TYPE_AMBIENT_TEMPERATURE
- Sensor.TYPE_HEART_RATE

Synthetic

- Sensor.TYPE_ROTATION_VECTOR
- Sensor.TYPE_LINEAR_ACCELERATION
- Sensor.TYPE_GRAVITY
- Sensor.TYPE_ORIENTATION (deprecated)

Kraken.me – Mobile

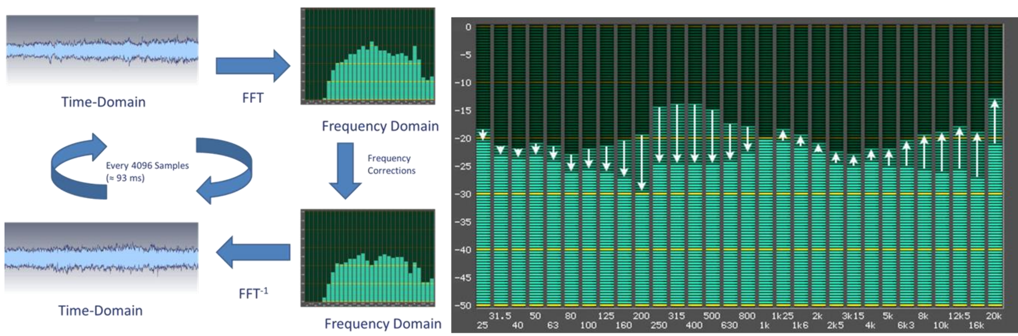
- Hardware Sensors
  - Location
  - Accelerometer
  - Activity
  - Environmental Light
  - Active & Available Connections
- Software Sensors
  - Foreground Applications
  - Calendar Events
  - Mails
  - Call Log
  - Browsing History
  - Ring Mode

Software Sensors

- ContentObserver in Android
  - Some sensors available in iOS
- Advanced access
  - Accessibility Framework
    - Full access to user interface
  - iClouds Backup in iOS

Sensor errors

- Sensors are prone to error
- Various techniques exist to help minimize error, but the main three are re-zeroing, filters and sensor fusion
- Re-zeroing – this involves calibrating the device in a known state



Mulit-hop on-the-fly calibration

- Calibrate a small number of phones
- Use a network of static sensors
  - How do we provide that?
- Multi-hop
  - Calibrated phones act as “beacons”
  - If a phone has reached a sufficient level of calibration accuracy it becomes a beacon
    - How to measure “calibration accuracy”?
- On-the-fly
  - Noise can be very different in close proximity
  - Beam-like handshake?
  - Gamification?

Filters

- Filters are a standard way of reducing the noise in any signal and can help use the signal over time to correct the noise.
- Common filters are
  - Low-Pass - filters out high frequency and smooths data
  - High-Pass - just gives the high frequency changes and can compensate for drift
  - Bandpass – filter off high and low frequency changes and just give information in the frequency range of interest
- More complex filters (such as Kalman filters) can be used, but are more computationally expensive

- Filters on the collected data, e.g., sanity checks or more sophisticated operations

Sensor Fusion

- Sometimes combining sensors can give us better results than just using a single sensor
- Many synthetic sensors do this, but we can do it manually
- For instance, suppose we wish to calculate the current rotation of the device.
  - We could use the gyroscope which is accurate, but gets unphysical quickly (unphysical means it doesn't correspond to the device status)
  - We could use the accelerometer which does correspond to the physical reality, but is very noisy
  - Best approach is to use the accelerometer to keep the gyroscope from drifting

Sensor API – the SensorManager

- Access to the API is through the SensorManager system service
- Use the SensorManager to obtain a list of sensors using the getSensorList() method
- Get the default sensor for a given type using the getDefaultSensor() method
- We cannot presume the existence of any sensor so we should check for it before we use it

```
public class SensorActivity extends Activity, implements SensorEventListener {
    private final SensorManager mSensorManager;
    private final Sensor mAccelerometer;

    public SensorActivity() {
        mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }

    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    public void onSensorChanged(SensorEvent event) {
    }
}
```

Example

```
public static boolean isAccelerometerSupported(Context
context)
   {
       SensorManager sm =
                (SensorManager) context
              .getSystemService(Context.SENSOR_SERVICE);

       List<Sensor> sensors =
sm.getSensorList(Sensor.TYPE_ACCELEROMETER);
       return sensors.size() > 0;
   }
```

SensorEventListener

- The SensorEventListener is an interface that all classes that wish to implement sensors should implement.
- It allows an app to respond to changes in the sensors.
- Classes which implement the interface must implement the methods:

```
public void onAccuracyChanged(Sensor sensor,
int accuracy)

public void onSensorChanged(SensorEvent
event)
```

Sensor Accuracy

- Sensor accuracy is a measure of how reliable of trustworthy the sensors value is. Allowable values are -
  - SensorManager.SENSOR_STATUS_ACCURACY_HIGH
  - SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM
  - SensorManager.SENSOR_STATUS_ACCURACY_LOW
  - SensorManager.SENSOR_STATUS_UNRELIABLE
- Note – an unreliable sensor does not mean it is faulty. It can mean that it simply needs to be calibrated
- Note – binary sensors (e.g., Proximity) are always unreliable

SensorEventListener

- In order to use a SensorEventListener you must register it with the SensorManager, specifying a delay which gives a hint to the system as to how often the sensor should be checked

```
sensorManager.registerListener(this, sensor,
                SensorManager.SENSOR_DELAY_NORMAL);
```

- Where allowable senor delay values are a number of milliseconds or
  - SENSOR_DELAY_FASTEST
  - SENSOR_DELAY_GAME
  - SENSOR_DELAY_UI (Suitable for usual user interface functions, like rotating the screen orientation.)
  - SENSOR_DELAY_NORMAL (The default value.)

SensorEventListener

- SensorEventListeners must be unregistered when your app is paused and re-registered when your app is resumed.
- Not doing so a) wastes battery & b) can cause your app to crash
- Android will unregister your SensorEventListener if the app is killed

```
sensorManager.unregisterListener(this);
```

Example

```
public class SensorActivity extends Activity implements
SensorEventListener {
  private SensorManager mSensorManager;
  private Sensor mLight;

  @Override
  public final void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mSensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
    mLight =
mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
  }
```

Example Continued

```
 @Override
  public final void onAccuracyChanged(Sensor sensor,
int accuracy) {
    Toast.makeText(getApplicationContext(), "Accuracy
has changed!", Toast.LENGTH_SHORT).show();
  }

  @Override
  public final void onSensorChanged(SensorEvent event)
{
    // The light sensor returns a single value.
    // Many sensors return 3 values, one for each axis.
    float lux = event.values[0];
    Toast.makeText(getApplicationContext(), "Let there
be light!", Toast.LENGTH_SHORT).show();
  }
```

Example Continued

```
 @Override
  protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mLight,
SensorManager.SENSOR_DELAY_NORMAL);
  }

  @Override
  protected void onPause() {
    super.onPause();
    mSensorManager.unregisterListener(this);
  }
}
```

Extensibility

- Wearables
- Additional sensors
  - Glucose
  - Environmental pollution
  - Etc.

Characteristics

- Large-scale, long lived, mostly static
- Small-scale, short lived, mostly mobile
- Mobility not an issue
- Mobility by design
- Application-specific
- Application-agnostic
- Very energy-constrained
- Energy no issue
- Multi-hop wireless
- Humans out of the loop

Idea

- **Users** charge their phone
- Steady **improvements in battery**
- **Efficient** hardware

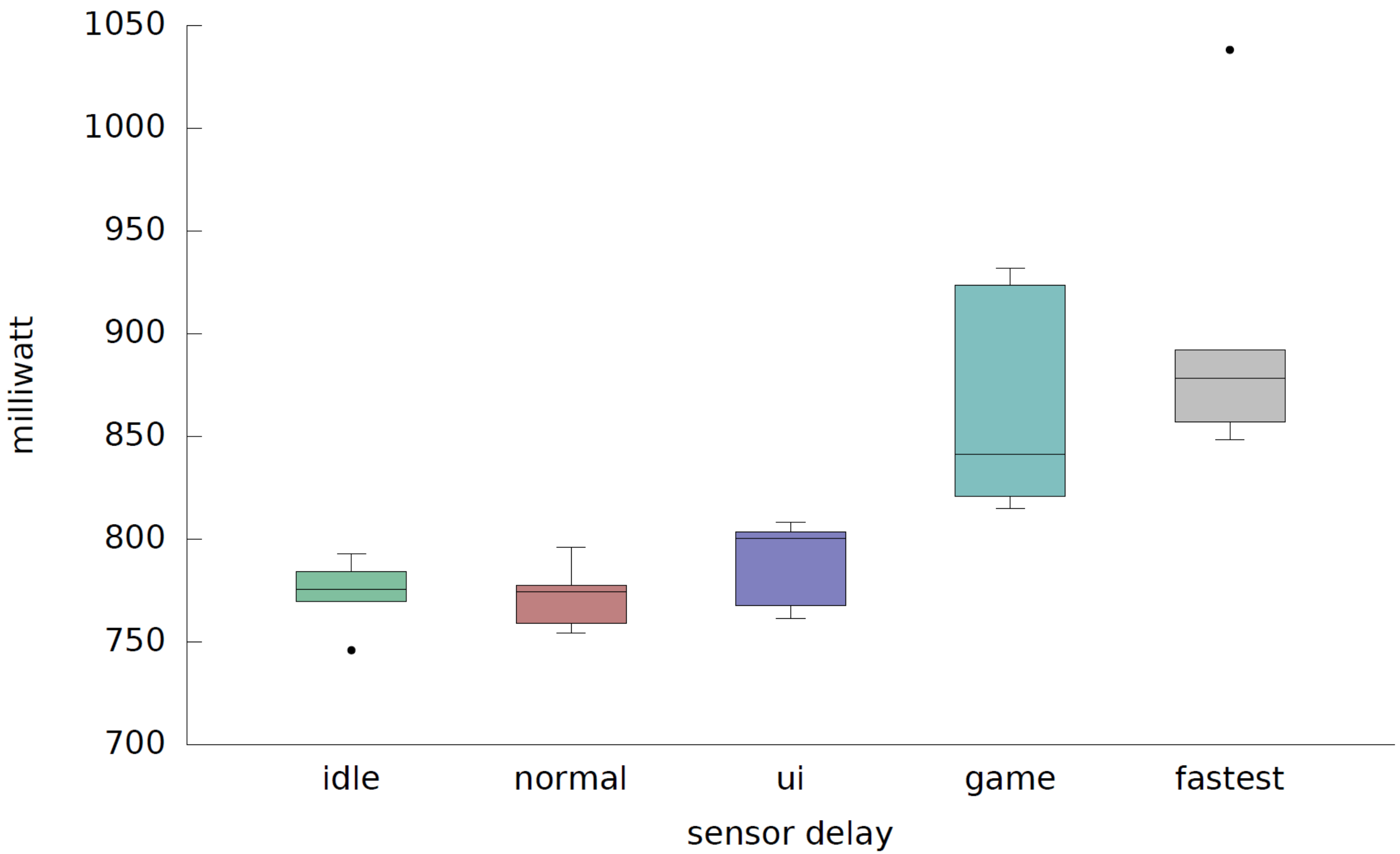
- Low maintenance
- Infinite battery-life

Reality

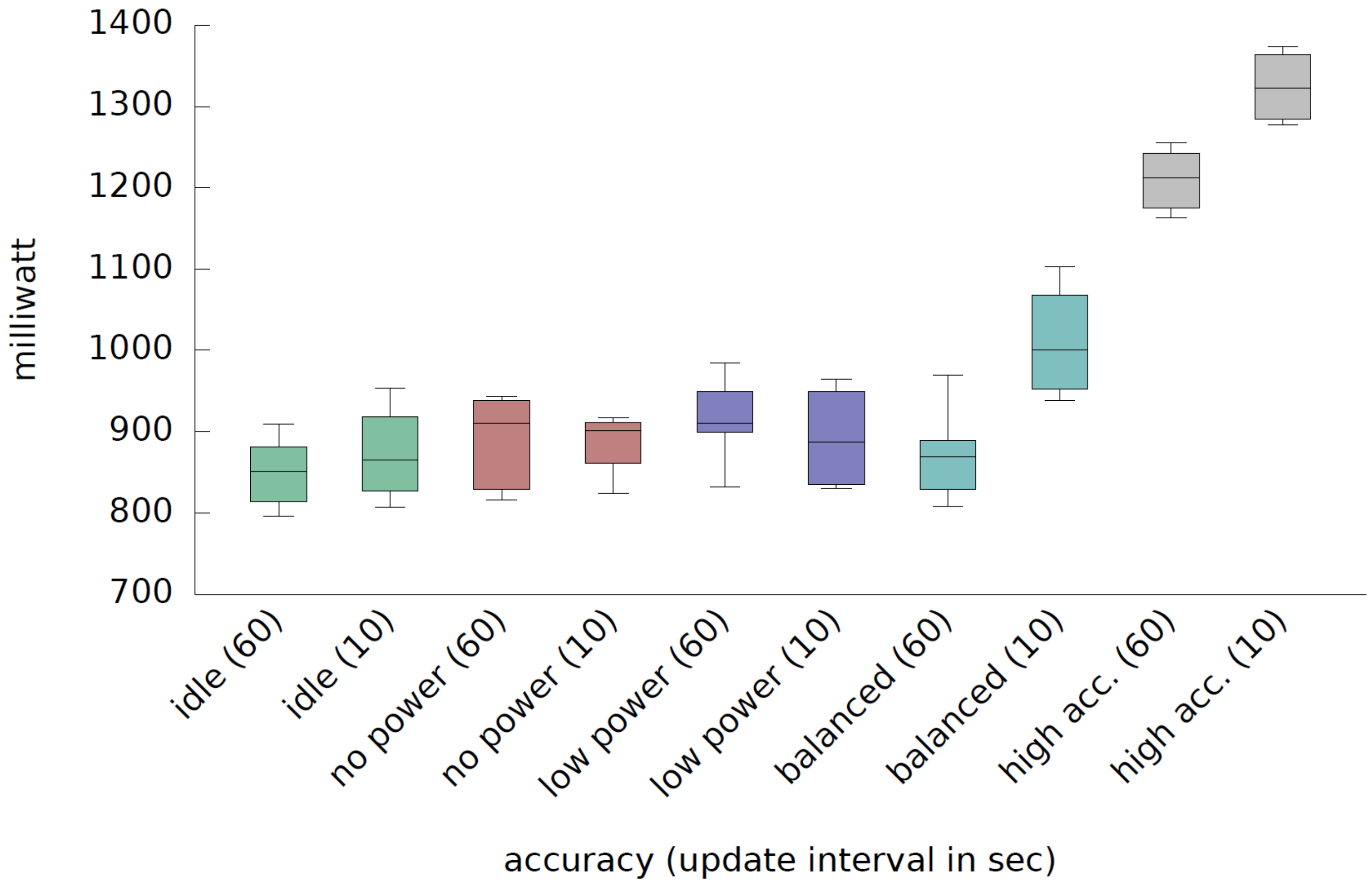
Energy Consumption

- Latency
  - How fast do we need to detect context change?
- Accuracy
  - How accurate does the inferred context need to be?
- LAB abstraction
  - Senergy API

Accelerometer

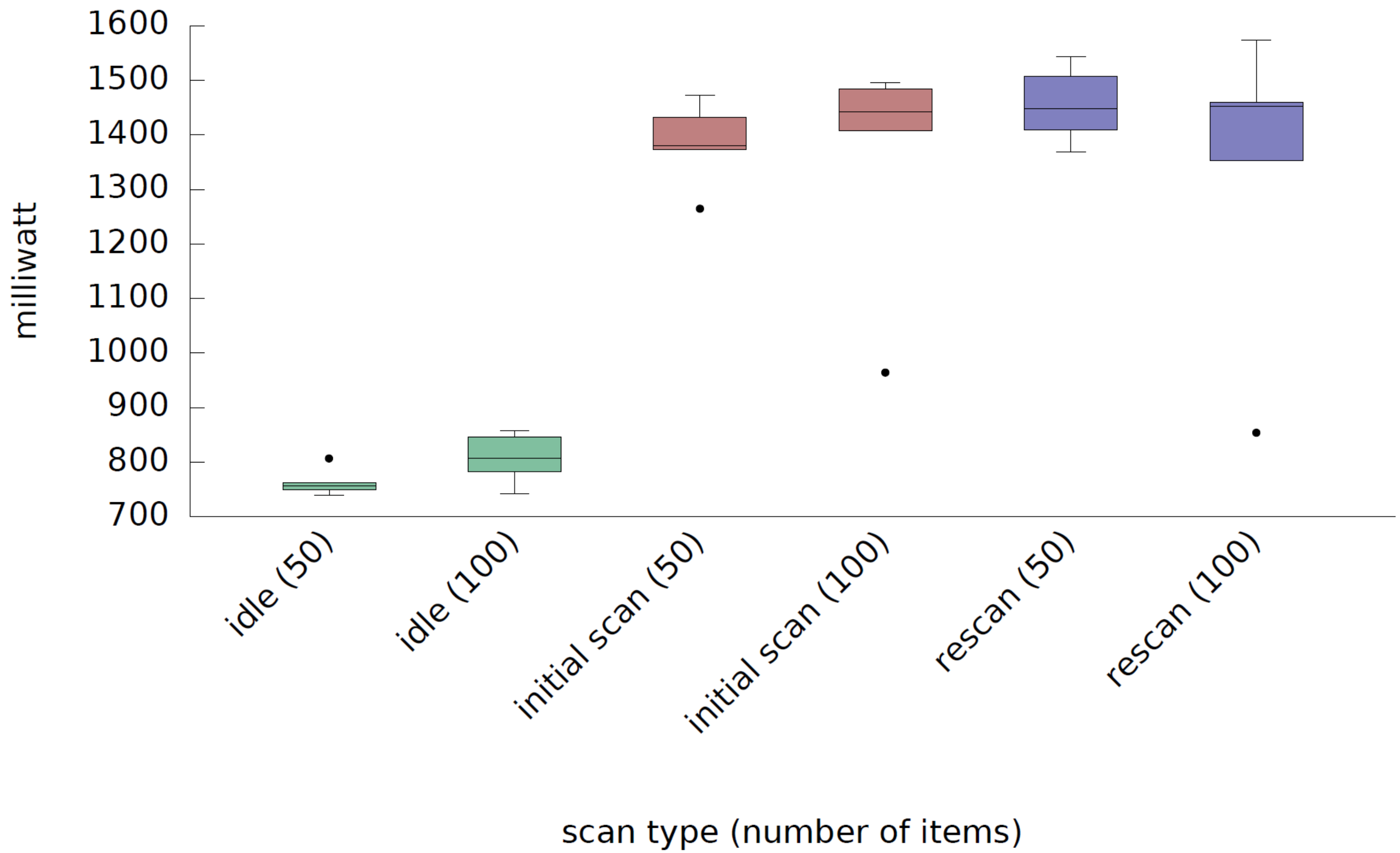


Location



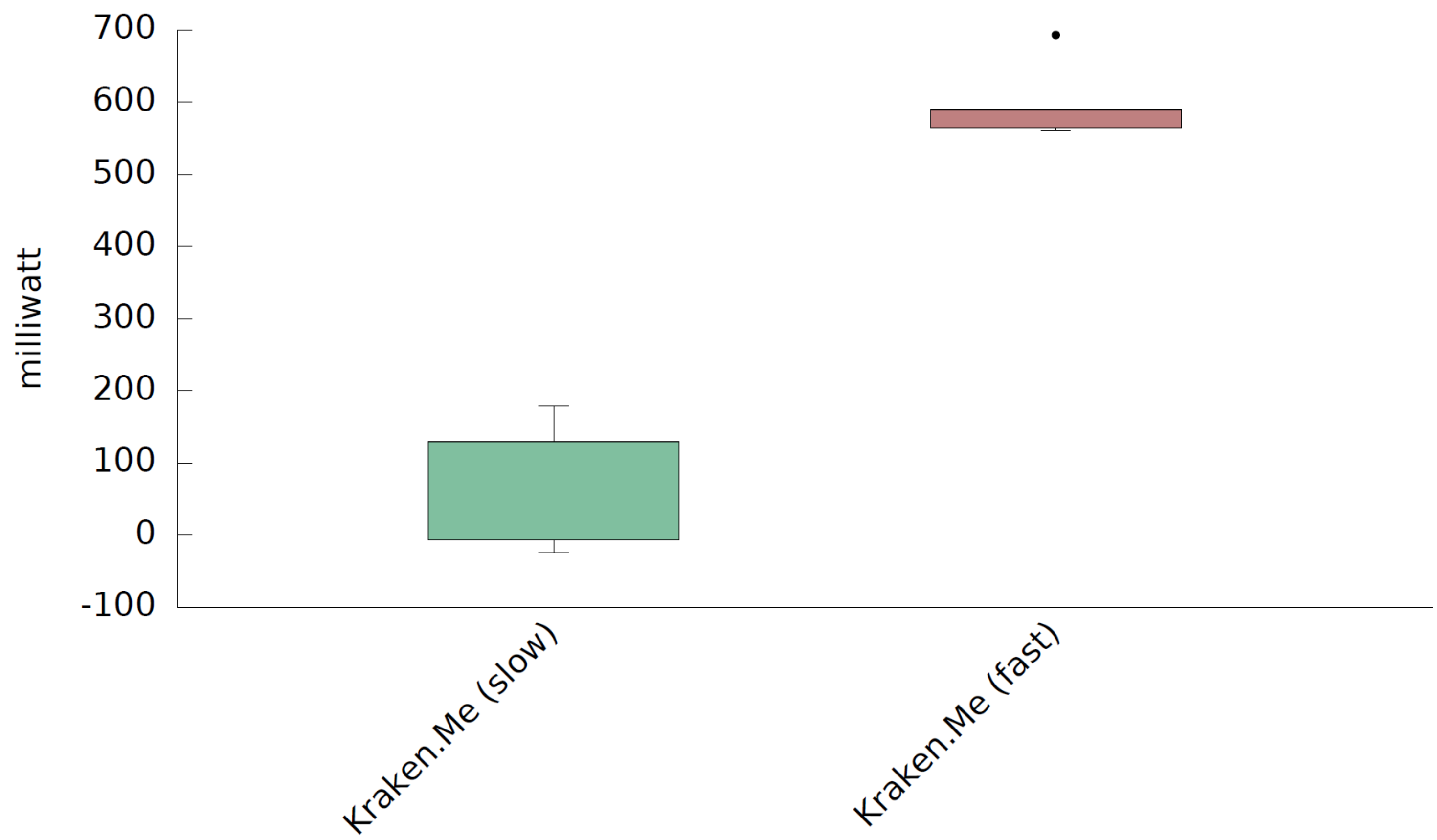
1400
1300
1200
1100
1000
900
800
700
milliwatt
idle (60)
idle (10)
no power (60)
no power (10)
low power (60)
low power (10)
balanced (60)
balanced (10)
high acc. (60)
high acc. (10)
accuracy (update interval in sec)

Calendar



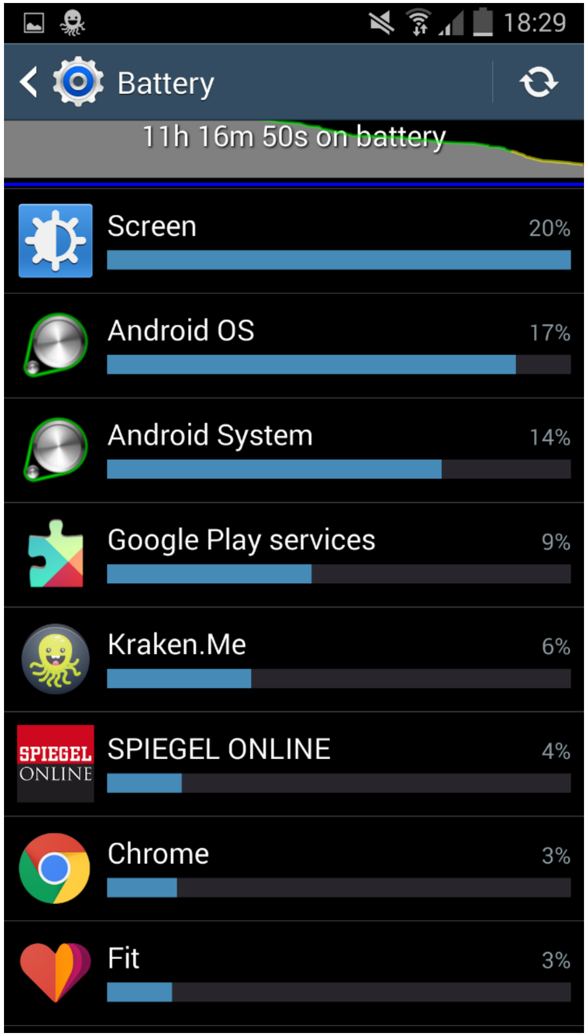
1600
1500
1400
1300
1200
1100
1000
900
800
700
milliwatt
idle (50)
idle (100)
initial scan (50)
initial scan (100)
rescan (50)
rescan (100)
scan type (number of items)

Kraken.me



700
600
500
400
300
200
100
0
-100
milliwatt
Kraken.Me (slow)
Kraken.Me (fast)

Kraken.me


18:29
Battery
11h 16m 50s on battery
Screen
20%
Android OS
17%
Android System
14%
Google Play services
9%
Kraken.Me
6%
SPIEGEL ONLINE
4%
Chrome
3%
Fit
3%

Characteristics

- Large-scale, long lived, mostly static / Small-scale, short-lived, mostly-mobile
- Mobility-not-an-issue / Mobility-by-design
- Application-specific / Application-agnostic
- Very energy-constrained / Energy-no-issue
- Multi-hop wireless / Single-hop wireless
- Humans out of the loop

Wireless Communication

- Bluetooth, NFC

- Cellular Network
- Wifi

- Mobility
- QoS
- Bandwidth / Latency
- Data Caps

Characteristics

- Small-scale, long lived, mostly-static / Large-scale, short lived, mostly-mobile
- Mobility-not-an-issue / Mobility-by-design
- Application-specific / Application-agnostic
- Very energy-constrained / Energy is no issue
- Multi-hop wireless / Single-hop wireless
- Humans out of the loop / Humans in the loop

Human in the Loop

- Limited attention span
- Limited time
- Limited battery

- Why should I participate?
  - Easier for personal sensing!
- Manage the excitement
  - How do I even get attention?

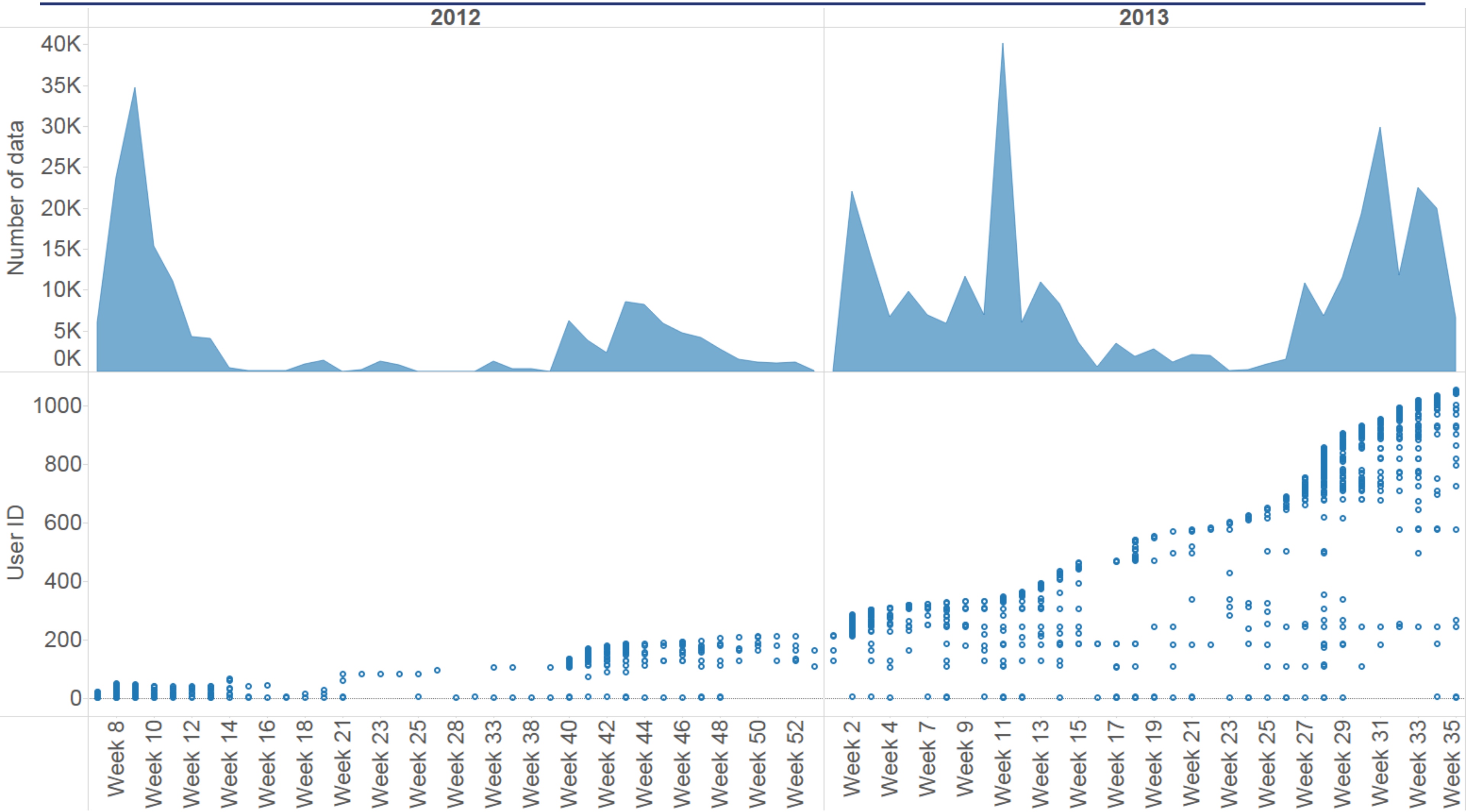
Human in the Loop

- Malicious users
  - Additional to other sensor failures
  - Trust their data
  - They will try to cheat your system
- Feedback
  - User Monitoring
  - Deal with negative feedback
  - Follow the heavy users

Approaches to Mobile Sensing

- Continuous background sensing
  - Privacy?
- Push-based sensing
  - User starts a new measurement / measurement series
- Pull-based sensing
  - Platform asks the user to sense

User numbers



Engagement

- Tangible use case
  - Search, Maps, etc.
  - Mail, Messaging, etc.
- Content
  - Youtube, Facebook, etc.
- Positive Feedback Loops
  - Games: Accessible, but not easy
  - Feeling of Progress
- Analytics: What are the heavy users doing / using?
- Optimize User Interface
- User-created content
- Gamification

Non-Competetive Incentives

Statistics

- Reflect on progress
- Analyze own behavior

Achievements

- Unlock the next stage
- Each stage gets harder
- Quality goals

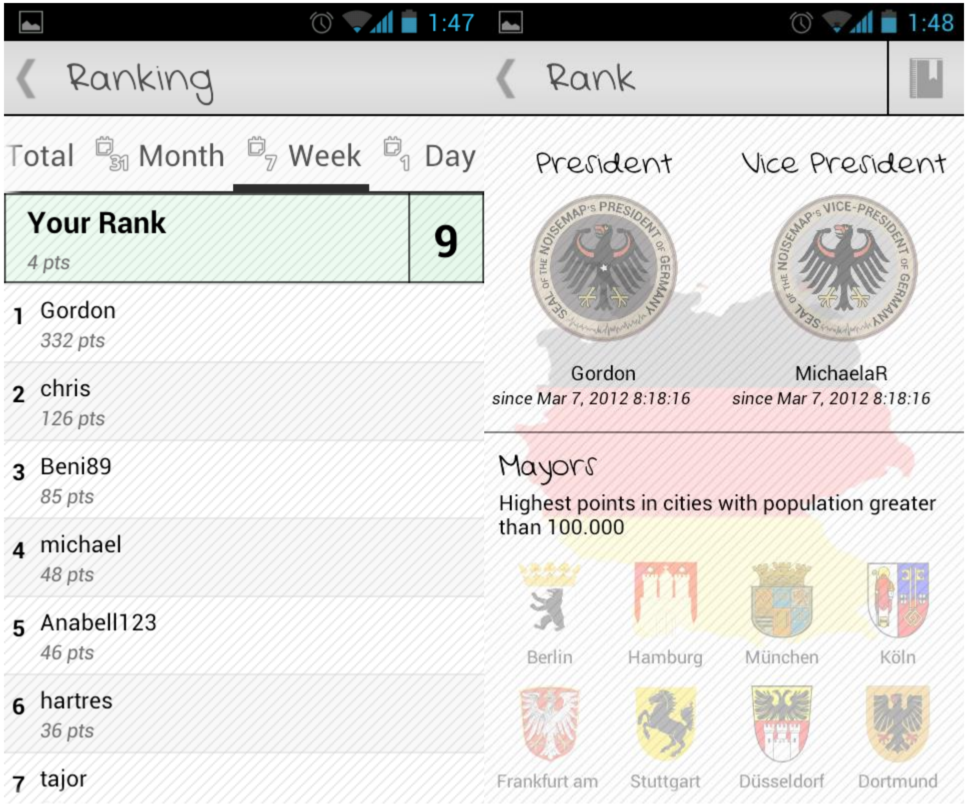
Competitive Incentives

Ranking

- Compete against others

Ranks

- Public achievements
- Limited to geographic entities (countries, cities)
- Winner takes it all



User study

AV1 (v1)

No incentives

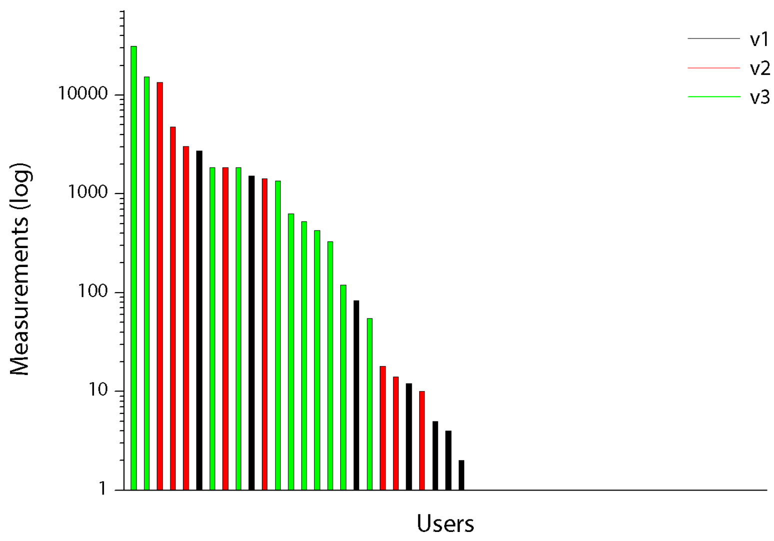
AV2 (v2)

Internal incentives

AV3 (v3)

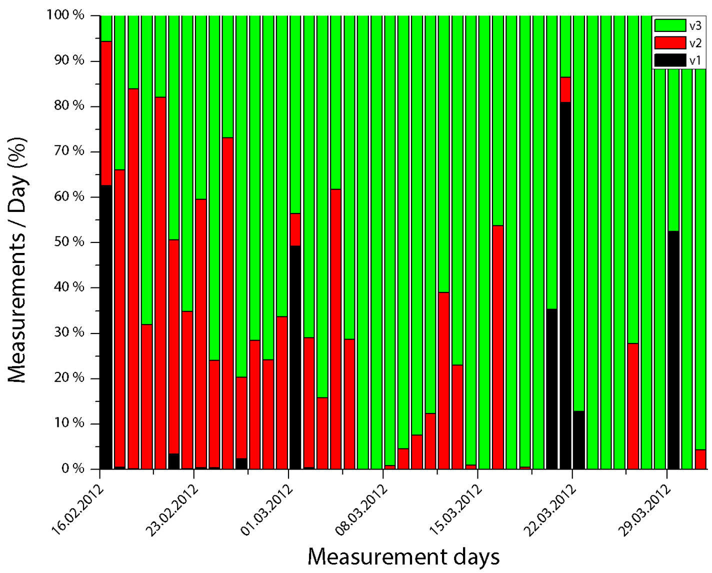
Internal + external incentives

Gamification



- Activity fades over time
  - Less so with incentive schemes
  - Make sure you have a steady flow of new users

- More incentive schemes
  - 401 (v1), 1614 (v2), 3357 (v3) measurements
  - 0.55, 6.55, 11.8 hours
  - 12.9, 34.6, 153.9 kms



Gamification

- Best if part of the „story“

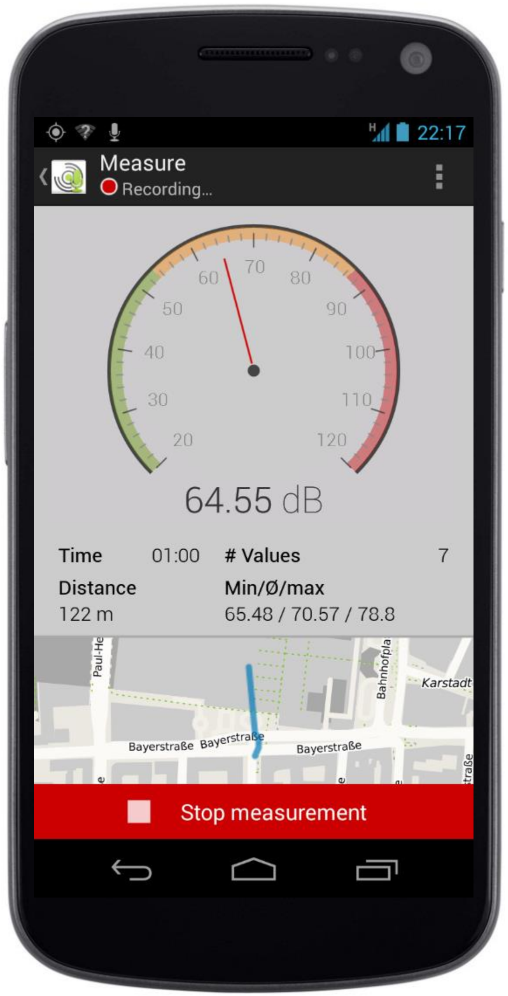
- Try to guide your users
  - Places nobody visits
  - Increase quality of data

- Get them to stick around
  - Until enough data is ready
  - Until they fulfill your micro-goals

Characteristics

- Large-scale, long lived, mostly mobile
- Mobility-by-design
- Application-agnostic
- Energy no issue
- Single-hop wireless
- Humans in the loop
- Privacy & Security

Open questions



- How to scale quicker
  - Scientific app store?
  - Nail it, then scale it vs. Get it out the door fast
    - Develop multiple apps, stick with the successful
  - Analyze user behavior
    - User-specific user interfaces!
  - Media attention
- Bonus questions:
    - How much data is needed?
- Human Actuation
  - Free will!
- Hybrid networks

Mobile phone vs. Car

Mobile phones

- App Store
- Extensibility
- Communication Interfaces

- Small sensors
- Small battery
- User operated

Car

- No unified delivery method
- Hard to extend with additional sensors
- Communication just now

- Sensor are fixed and can be more expensive
- Batteries recharge
- More powerful sensors will be added for autonomous

Summary: Applications

- Personal Sensing
  - Focus on individual benefit
  - E.g., health apps
- Peer Sensing
  - Small group of users
  - E.g., urban games
- Utility Sensing
  - Provides utility for community
  - E.g., Noise mapping

Summary “Sensing”

- From tagging to people-centric sensing
  - **Mapping** the virtual and real-world
  - Measure and act on the **real-world**
  - Measure and act on **humans** and their **environment**

- Challenges
  - Energy
  - Communication
  - Data processing
  - Context-awareness
  - Interaction with humans
  - Privacy