

# Next Generation User Interfaces

## Indoor Positioning and Location Awareness

---

Maxim Van de Wynckel



# Indoor Positioning System



*"An indoor positioning system (IPS) is a network of devices used to locate people or objects where GPS and other satellite technologies lack precision or fail entirely, such as inside multistory buildings, airports, alleys, parking garages, and underground locations."*

- Wikipedia (2022)

- (image) Bonkers.io (2018)



# Location Awareness



*"Location awareness refers to devices that can passively or actively determine their location."*

- Wikipedia (2022)

# Contextual Location Awareness



Using the location of a person or asset as contextual information for **implicit human-computer interaction**.

*NOTE: You will see implicit human-computer interaction in a later lecture*

# Use Cases

## Indoor Navigation



- Gatwick Airport Limited (2017)

# Use Cases ...

## Asset Tracking

- ▶ Track the location of expensive items
- ▶ Material resource planning
- ▶ Personnel tracking



- LetsGoDigital (2021)

- Favendo (2022)

# Use Cases ...

## Implicit Interaction

- ▶ Home automation
- ▶ Resource assigning
- ▶ Machine learning recommendations

# Use Cases ...

## Autonomous Robots



- Jim Martin (2019)

# Positioning Technologies



- ▶ RF-based (Wi-Fi, Bluetooth, UWB)
- ▶ Sound-based (Ultrasound beacons, ambient noise)
- ▶ Camera-based (VSLAM, MTMC, VLC)
- ▶ Inertial Measurement Unit (IMU-based) (acceleration, velocity, geomagnetic)
- ▶ *Many more ...*

# Positioning Algorithms



- ▶ Fingerprinting (k-NN, affinity propagation, clustering, ...)
- ▶ Mathematical (triangulation, multilateration, ...)
- ▶ Dead reckoning
- ▶ Sensor fusion (complementary filters, probabilistic, ...)
- ▶ Cartographing
- ▶ *Many more ...*

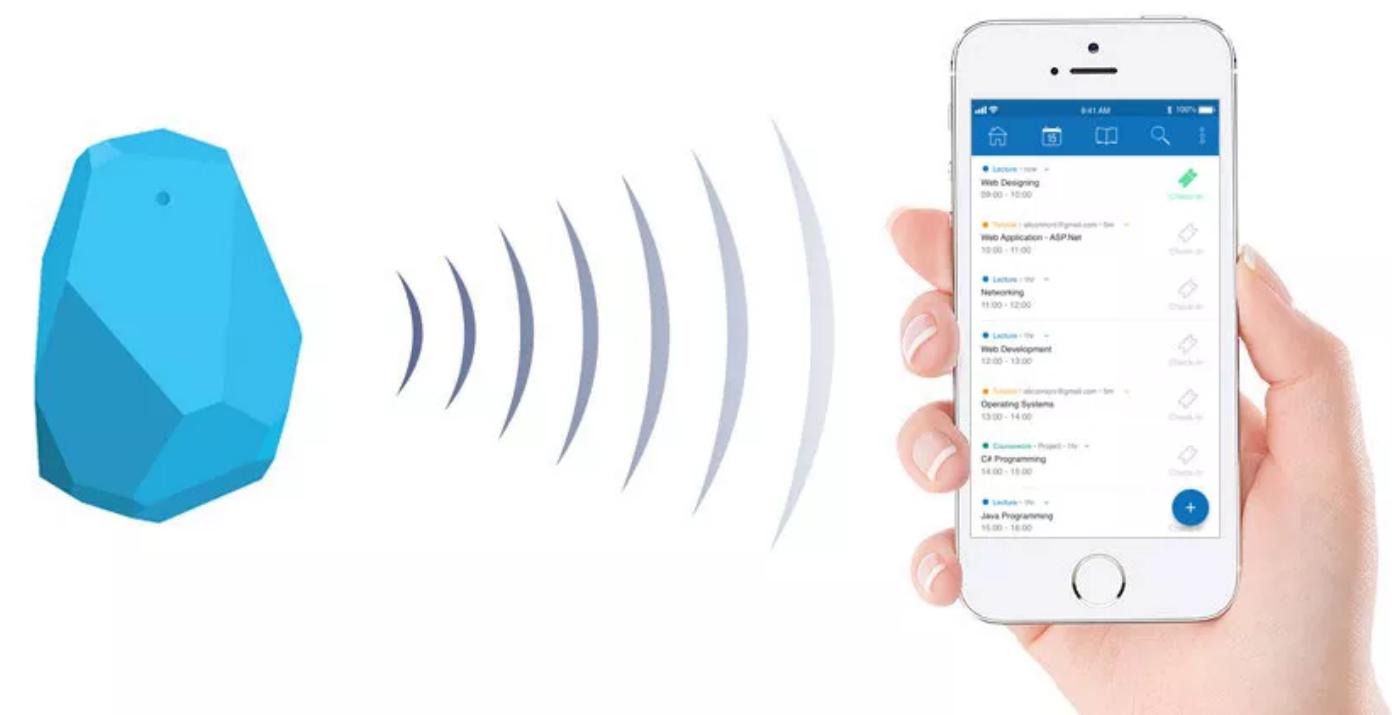
# Bluetooth beacons



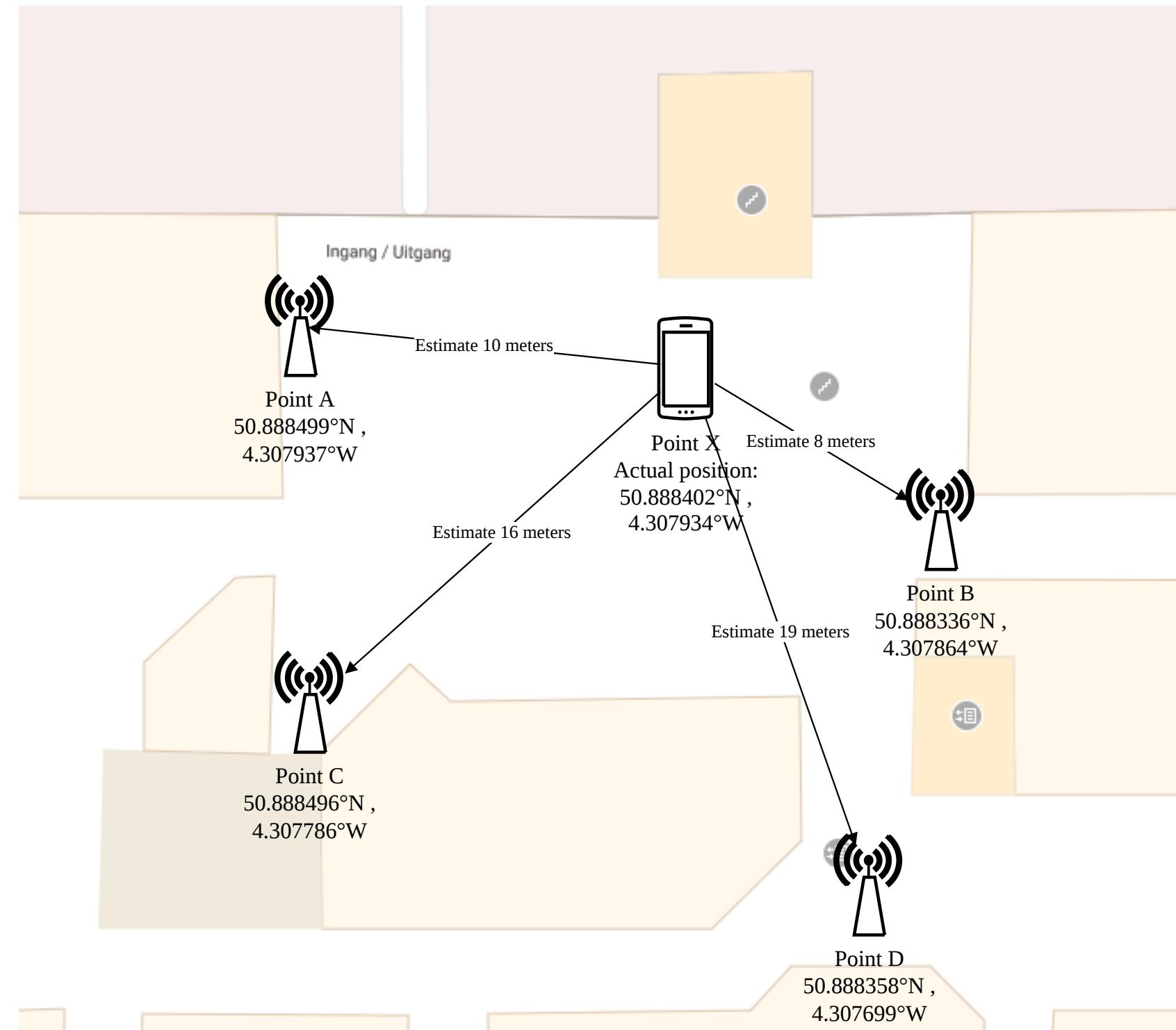
A small (battery powered) device that sends out a BLE advertisement in a fixed interval. The received signal strength is used to estimate the linear distance.

## Algorithms:

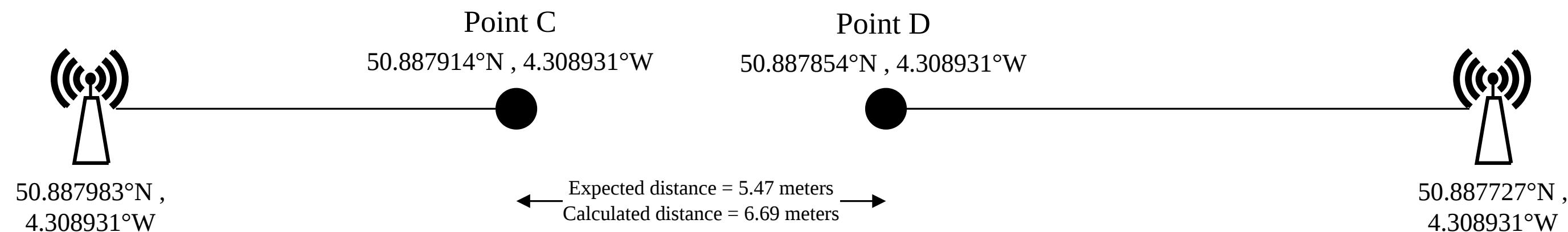
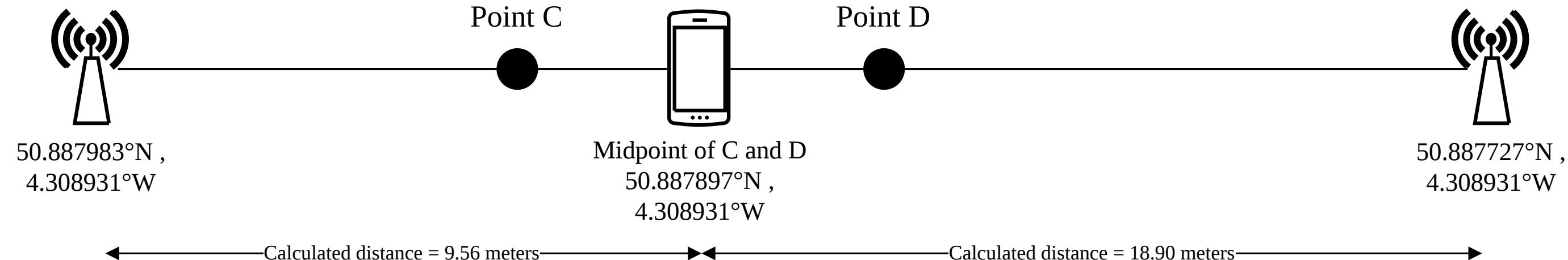
- ▶ Fingerprinting
- ▶ Multilateration, Cell-Identification



# Multilateration (>3 transmitters)

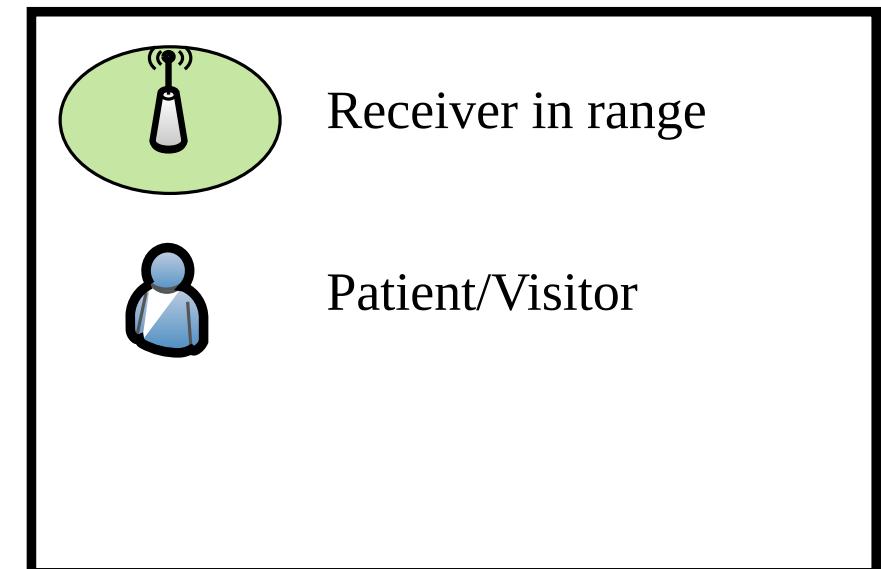
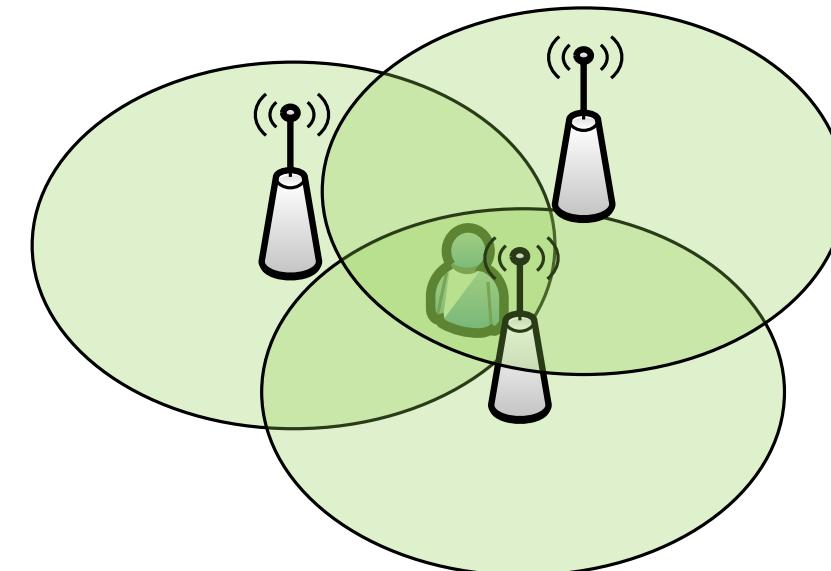
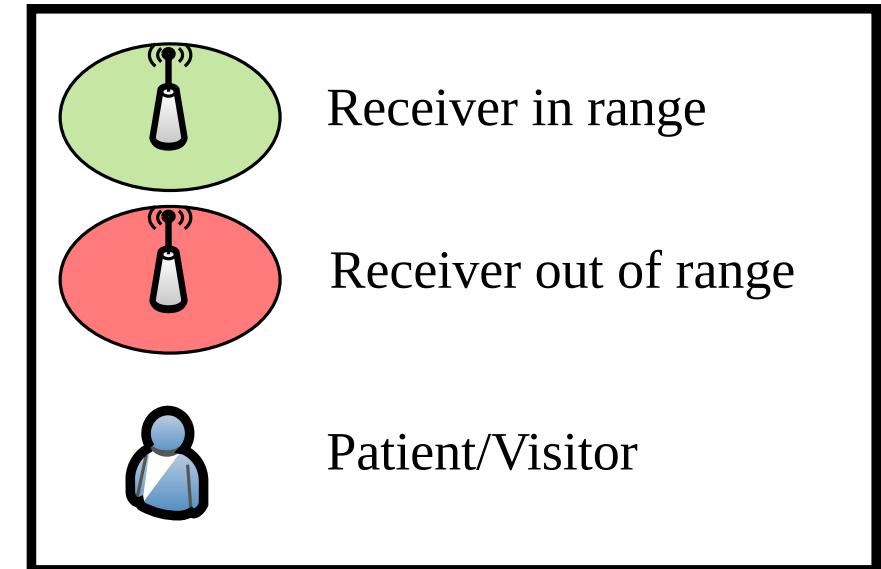
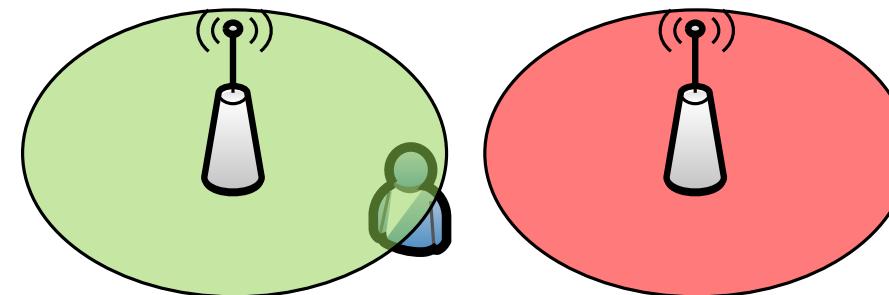


# Multilateration (2 transmitters)



- Van de Wynckel, Maxim (2019)

# Cell-Identification (1 transmitter)



- Van de Wynckel, Maxim (2019)

# Ultra wideband



A small (battery powered) device that sends out an RF signal over a large band (6 - 8 GHz), allowing the use of Time of Flight (ToF) for determining the distance to centimeter precision. More power consumption than BLE.

## Algorithms:

- ▶ Multilateration

# Wi-Fi access points

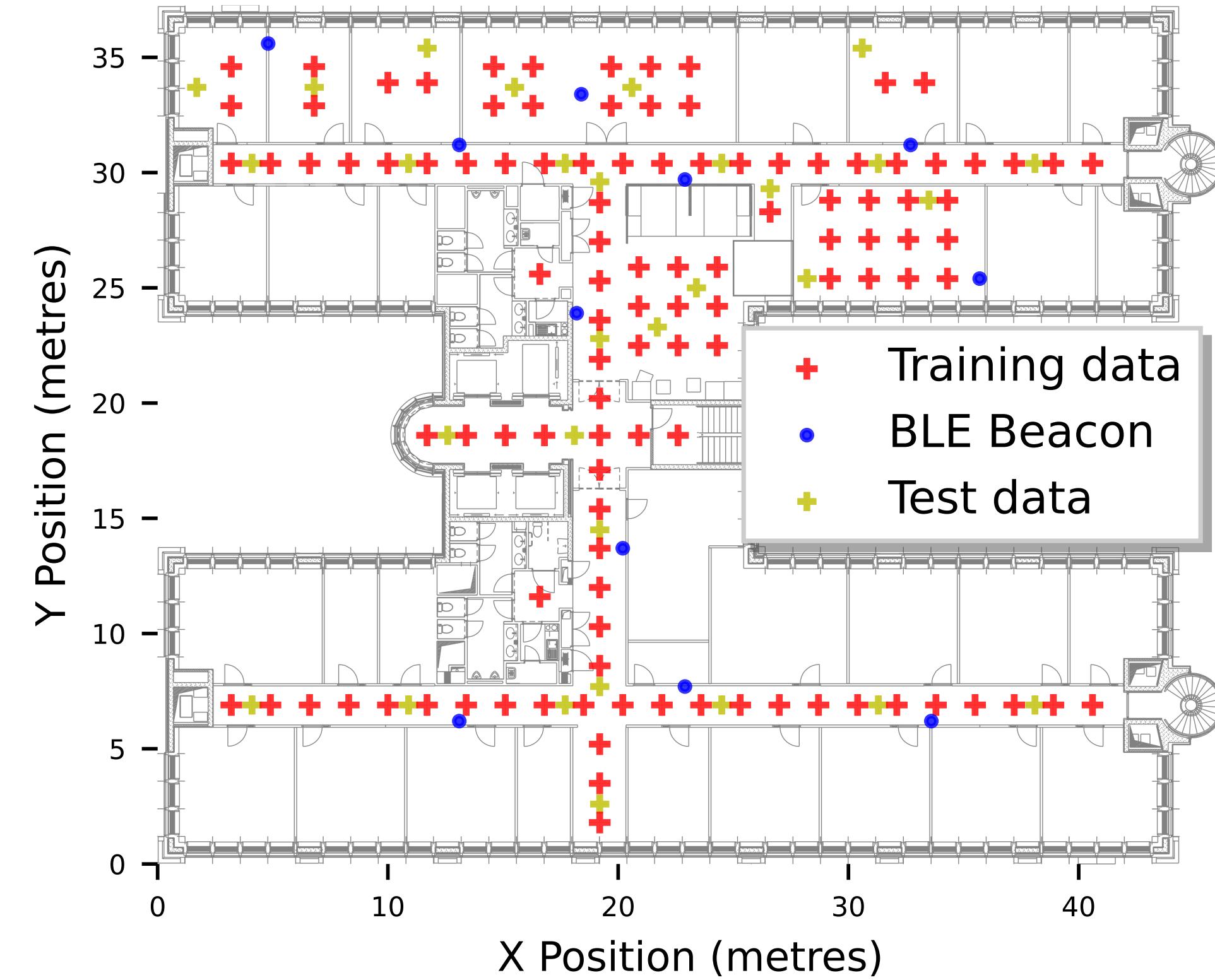


Existing Wi-Fi infrastructure to predict the position based on the signal strength or Time of Flight (ToF) on supported access points.

## Algorithms:

- ▶ Fingerprinting
- ▶ Multilateration

# Fingerprinting



# Fingerprinting ...

- Van de Wynckel, Maxim & Signer, Beat (2021)

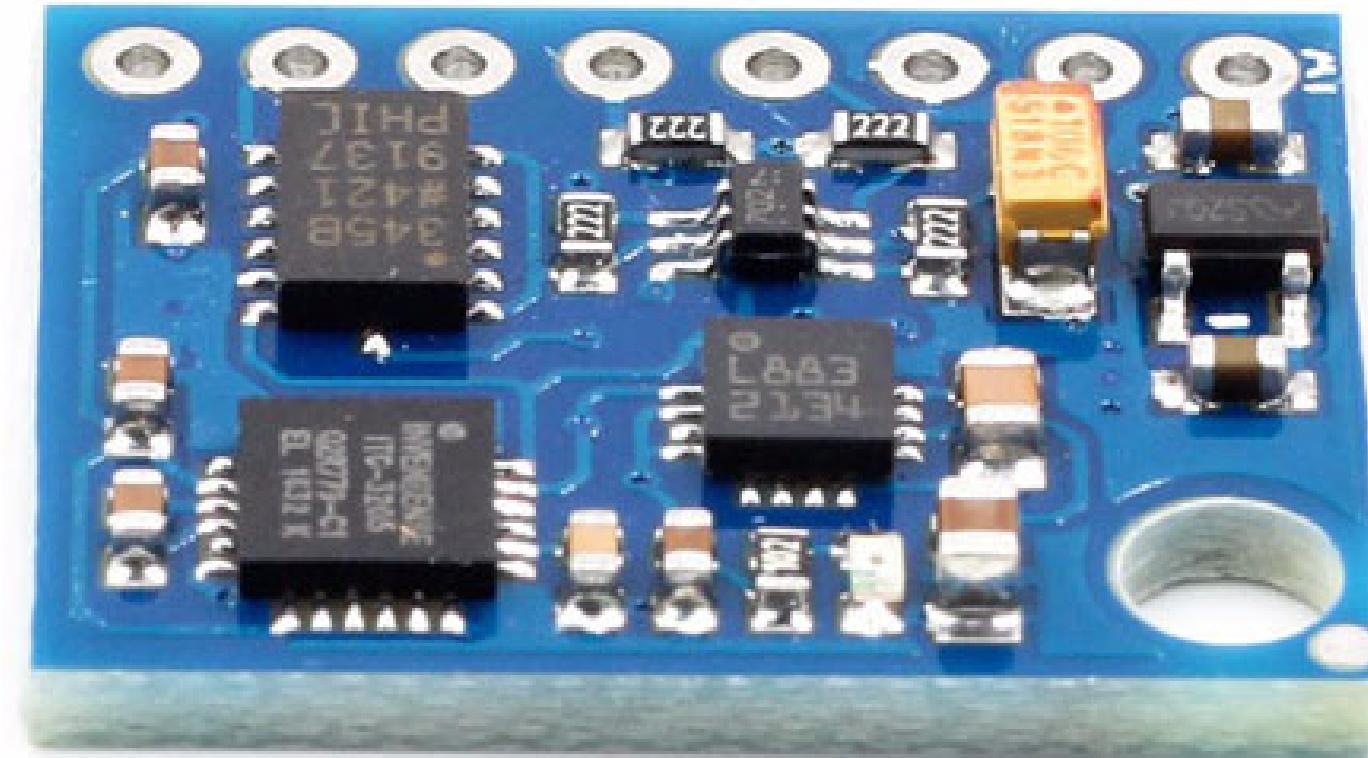
# Inertial Measurement Unit (IMU sensor)



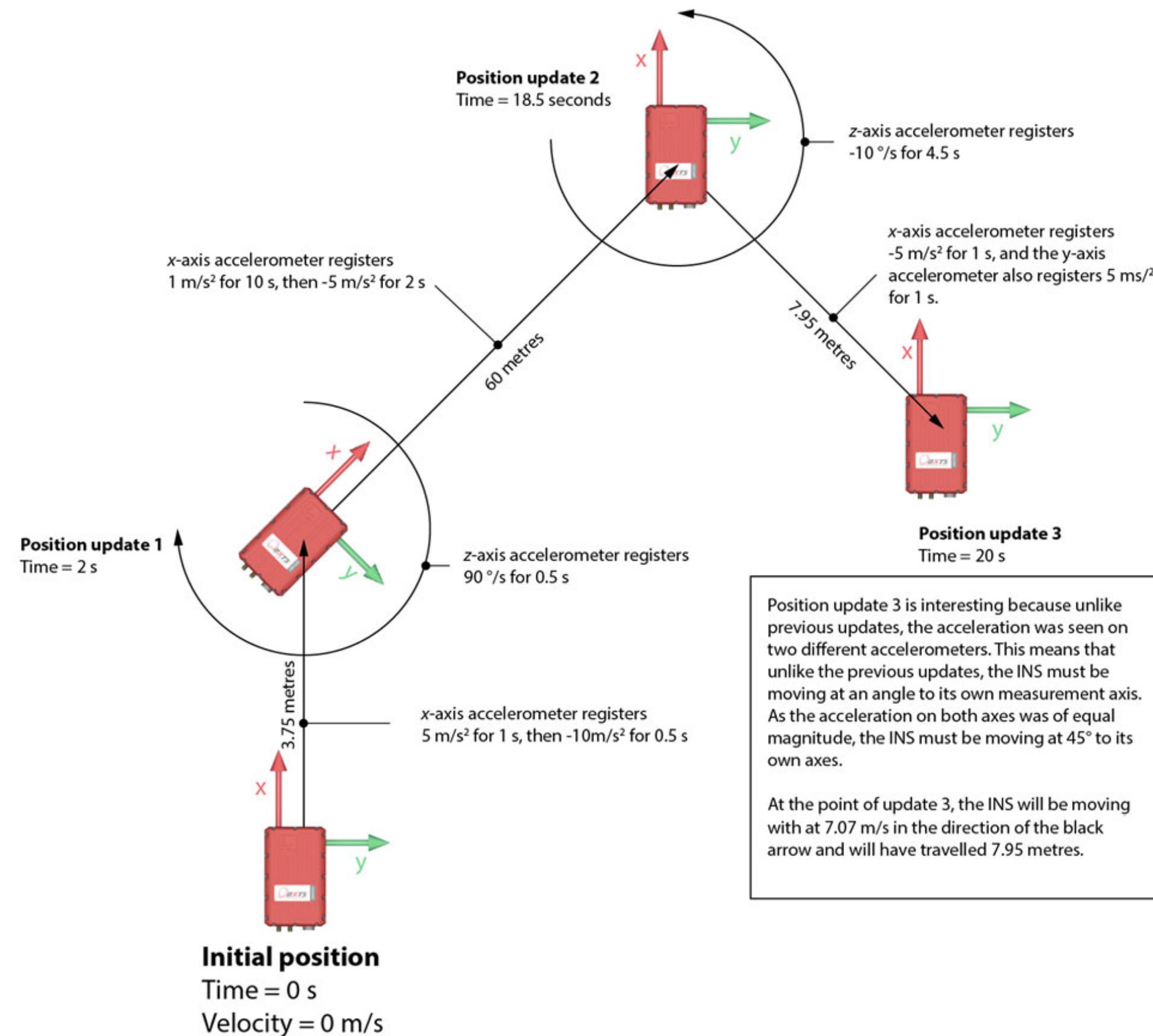
Accelerometer, gyroscope and magnetometer to determine the orientation, linear and angular velocity. Optionally can contain a barometer.

## Algorithms:

- ▶ Dead reckoning (raw, pedometer)
- ▶ Geomagnetic fingerprinting



# Dead Reckoning



# What is OpenHPS?

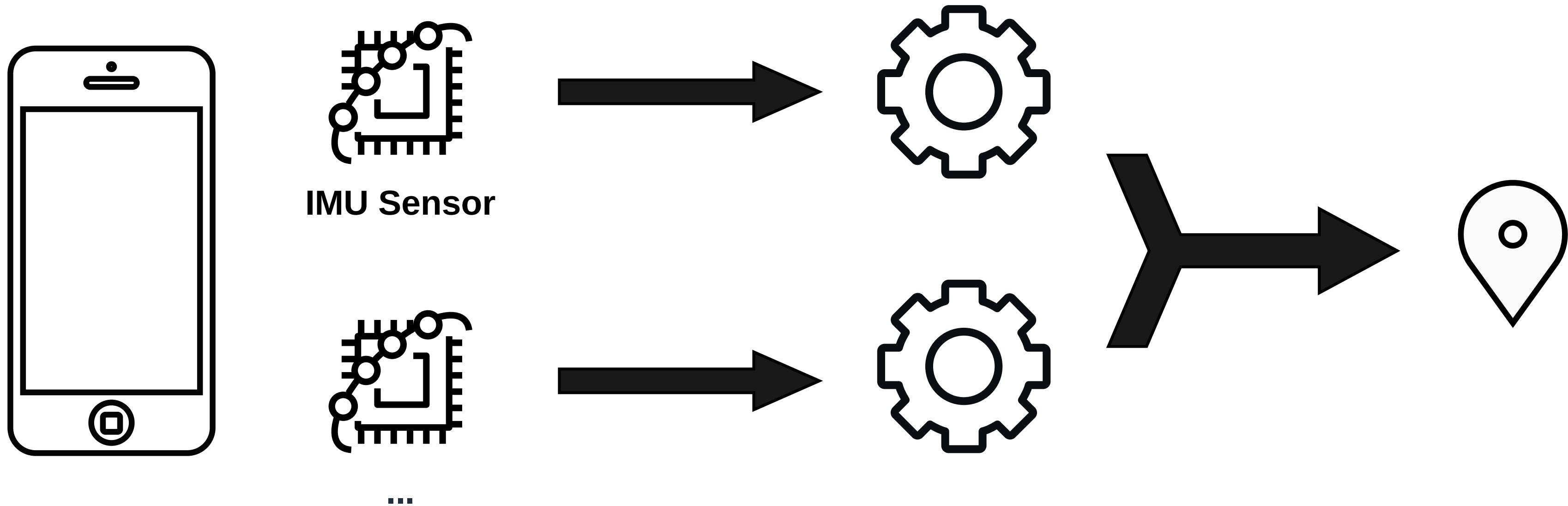


An Open Source Hybrid Positioning System

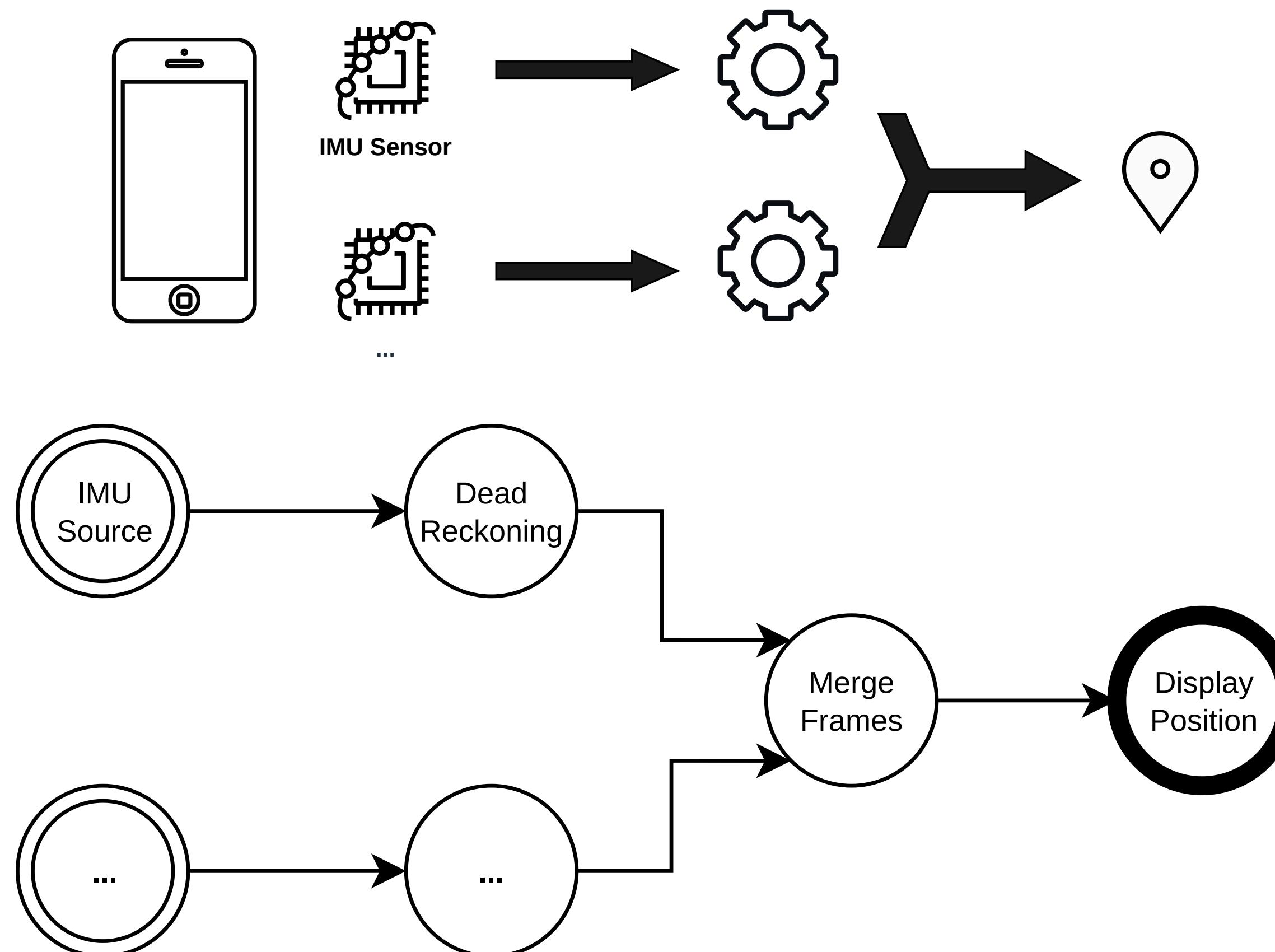
## An Open Source Hybrid Positioning System

- ▶ Any technology
- ▶ Any algorithm
- ▶ Various use cases
- ▶ Flexible processing and output
  - Prefer accuracy over battery consumption, reliability, ...
- ▶ Aimed towards developers and researchers

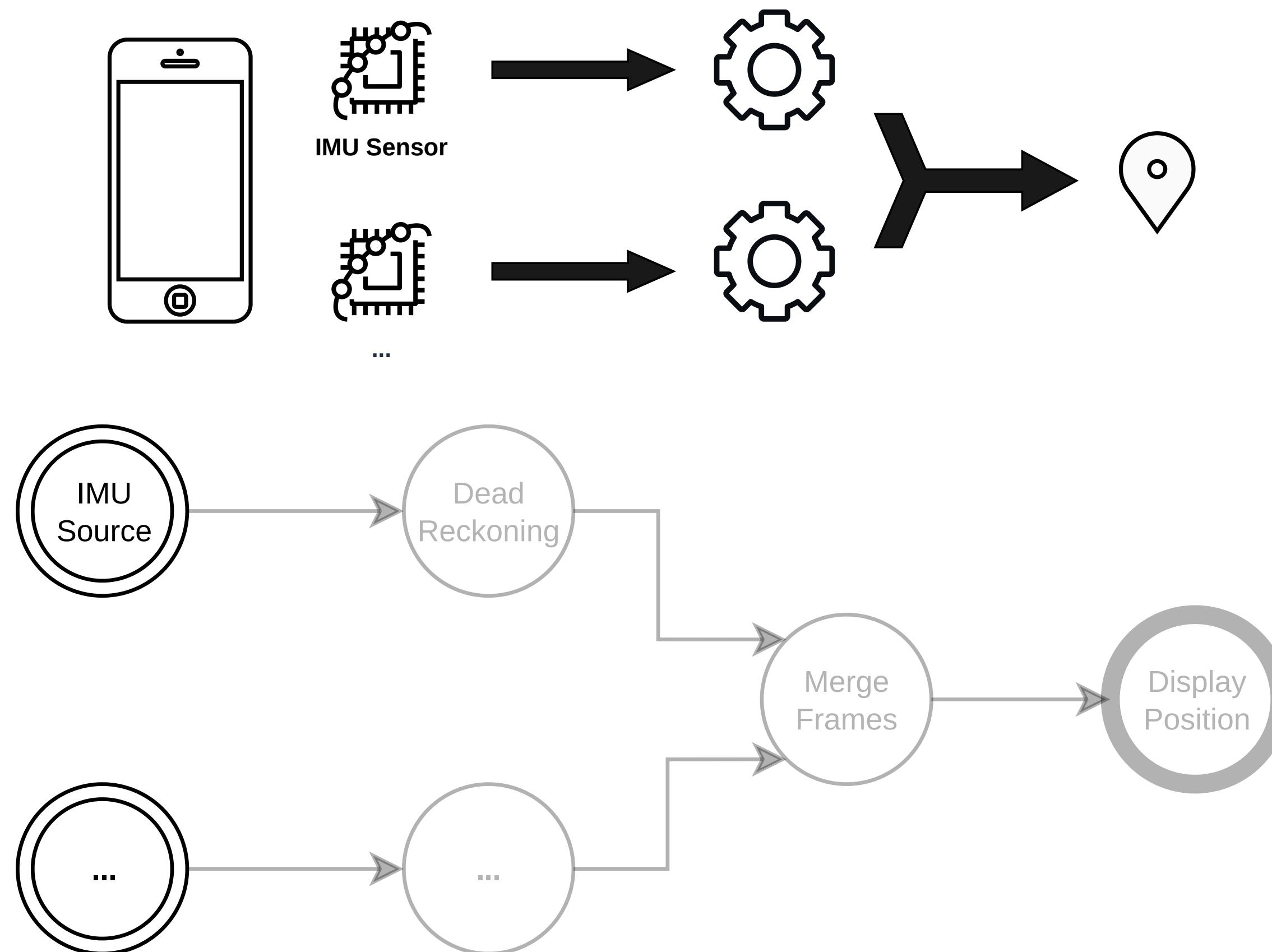
# Process Network Design



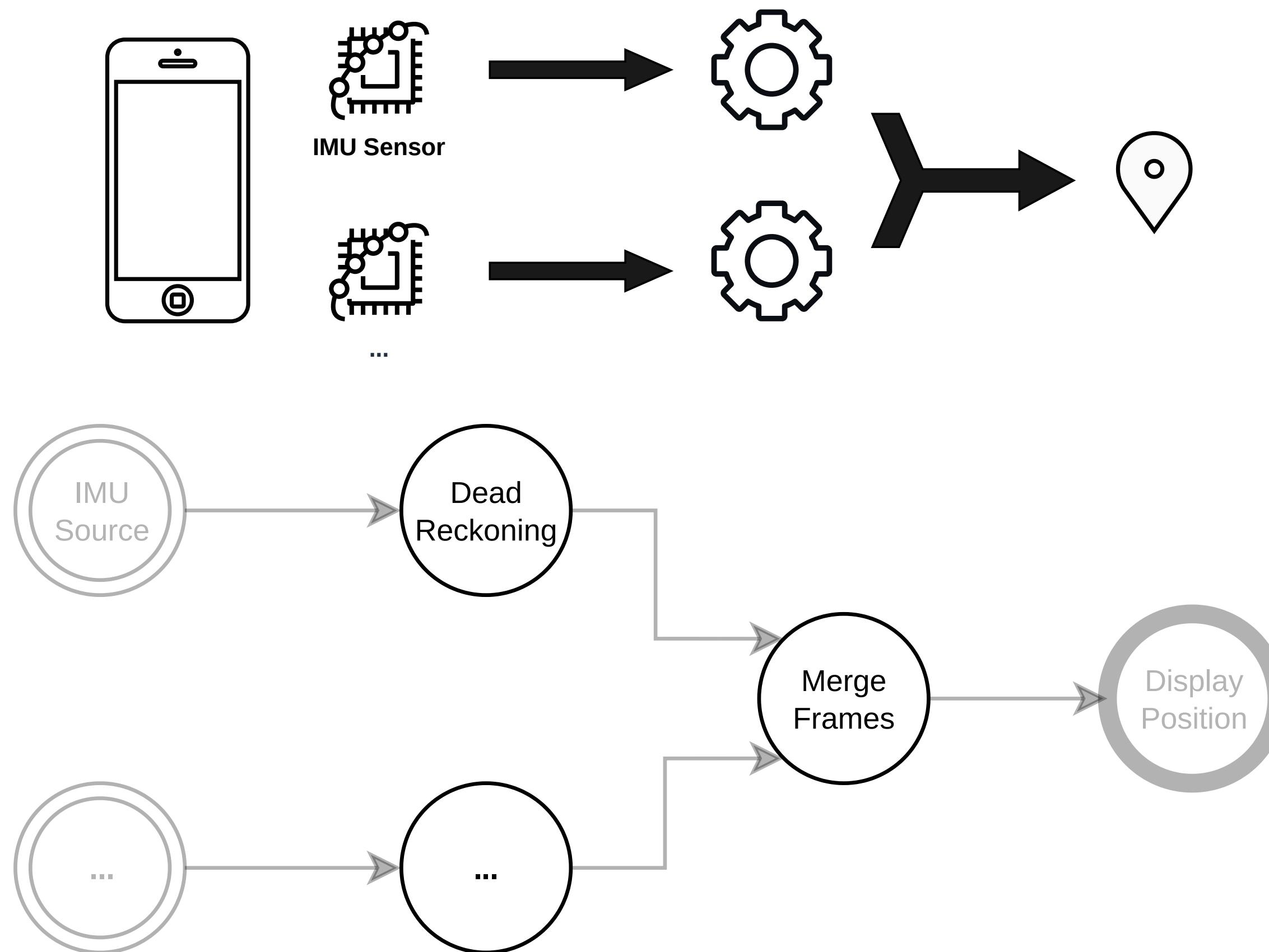
# Process Network Design ...



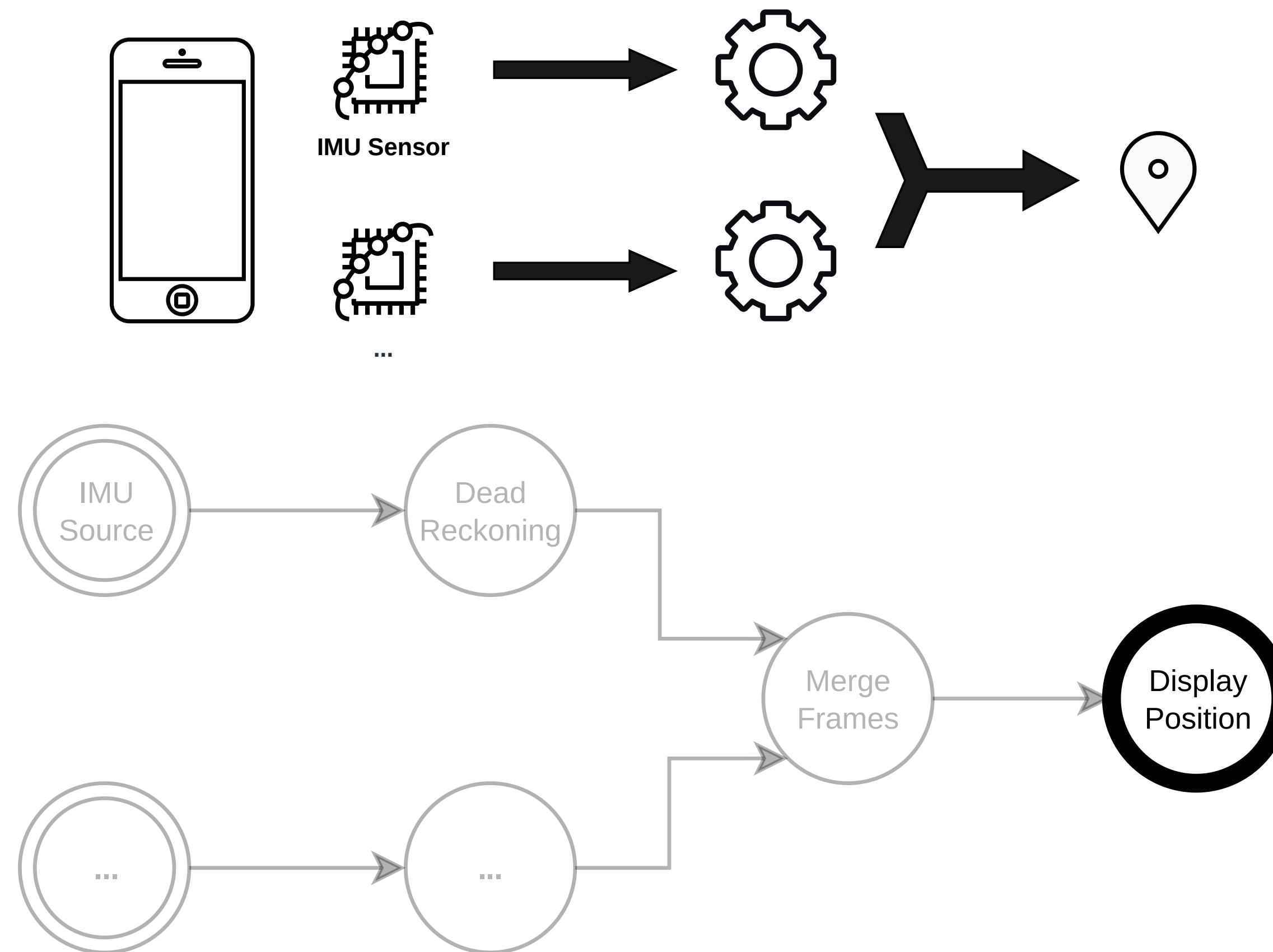
# Process Network Design ...



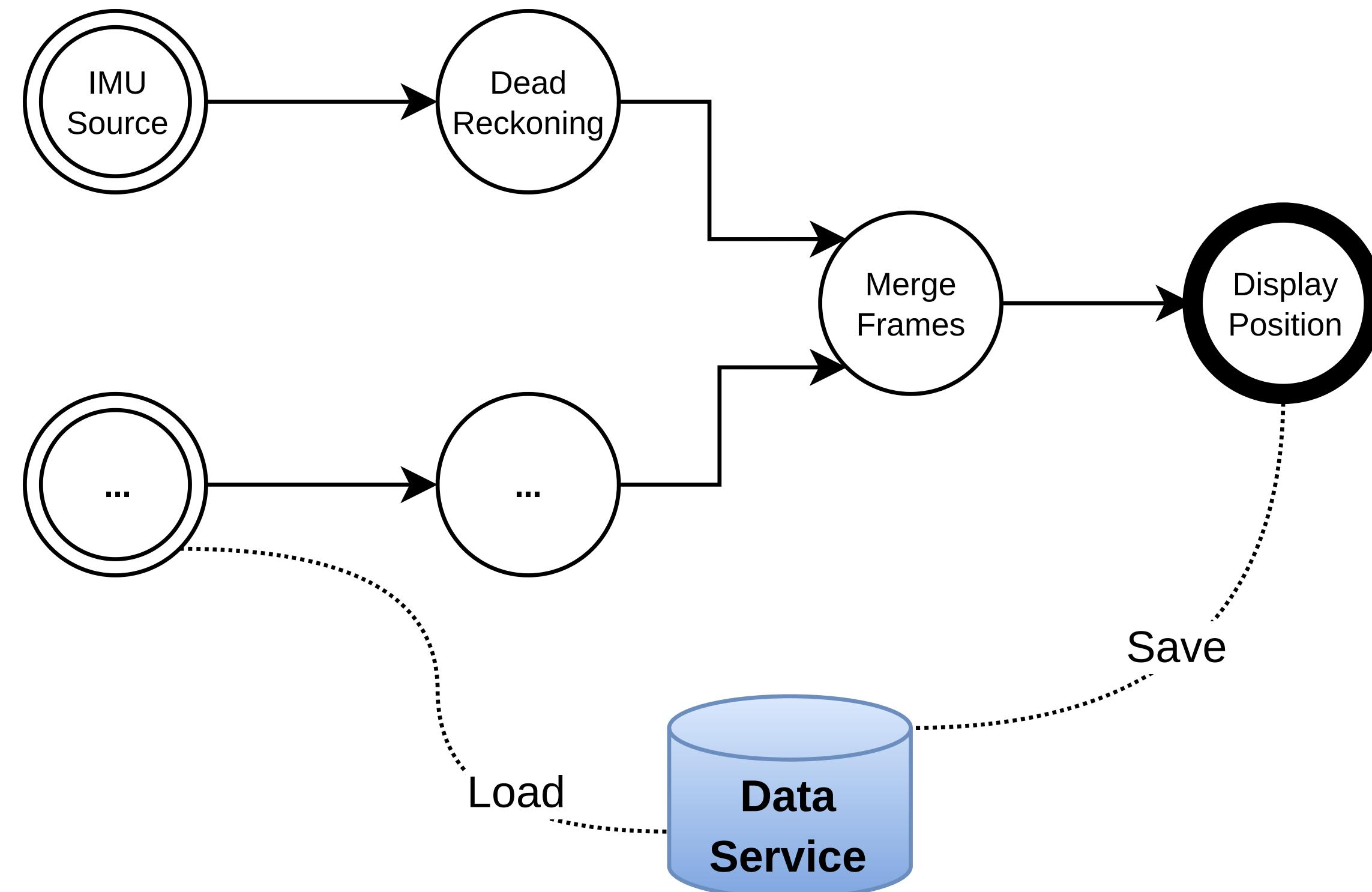
# Process Network Design ...



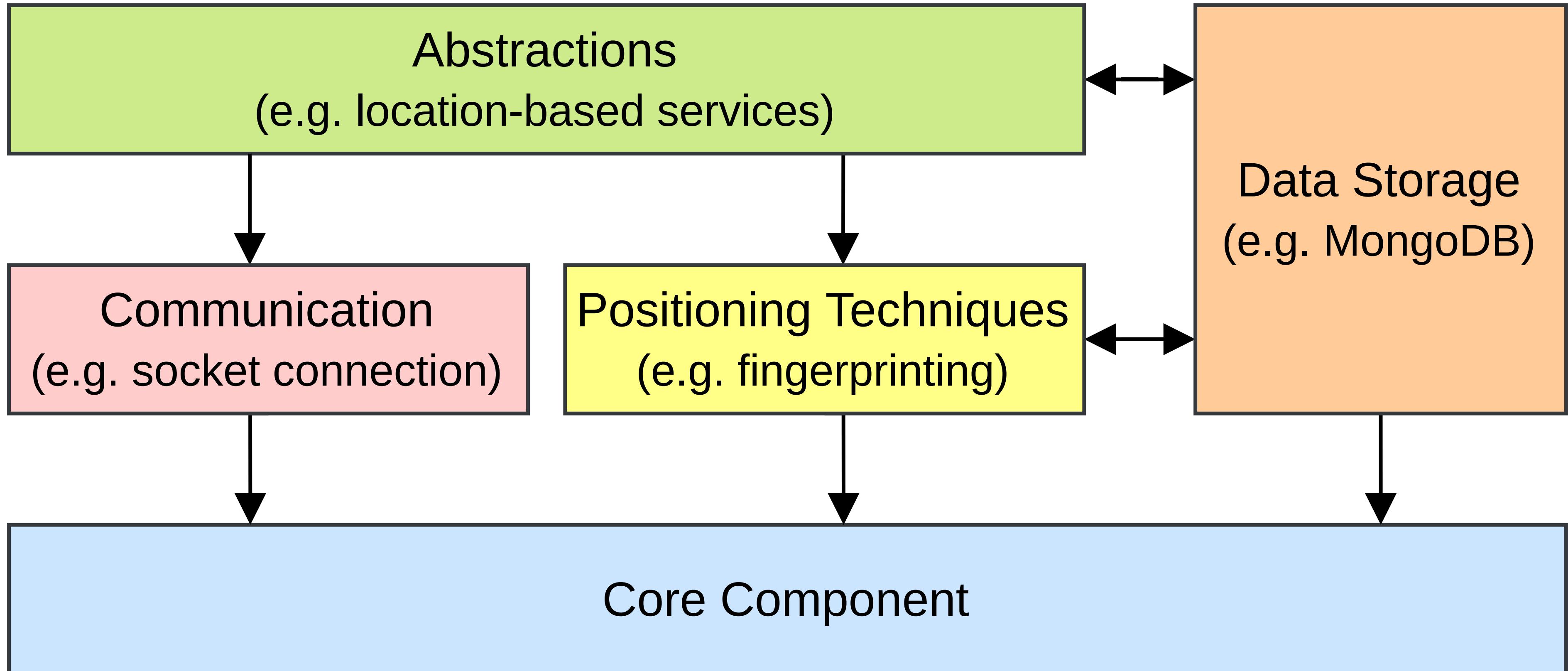
# Process Network Design ...



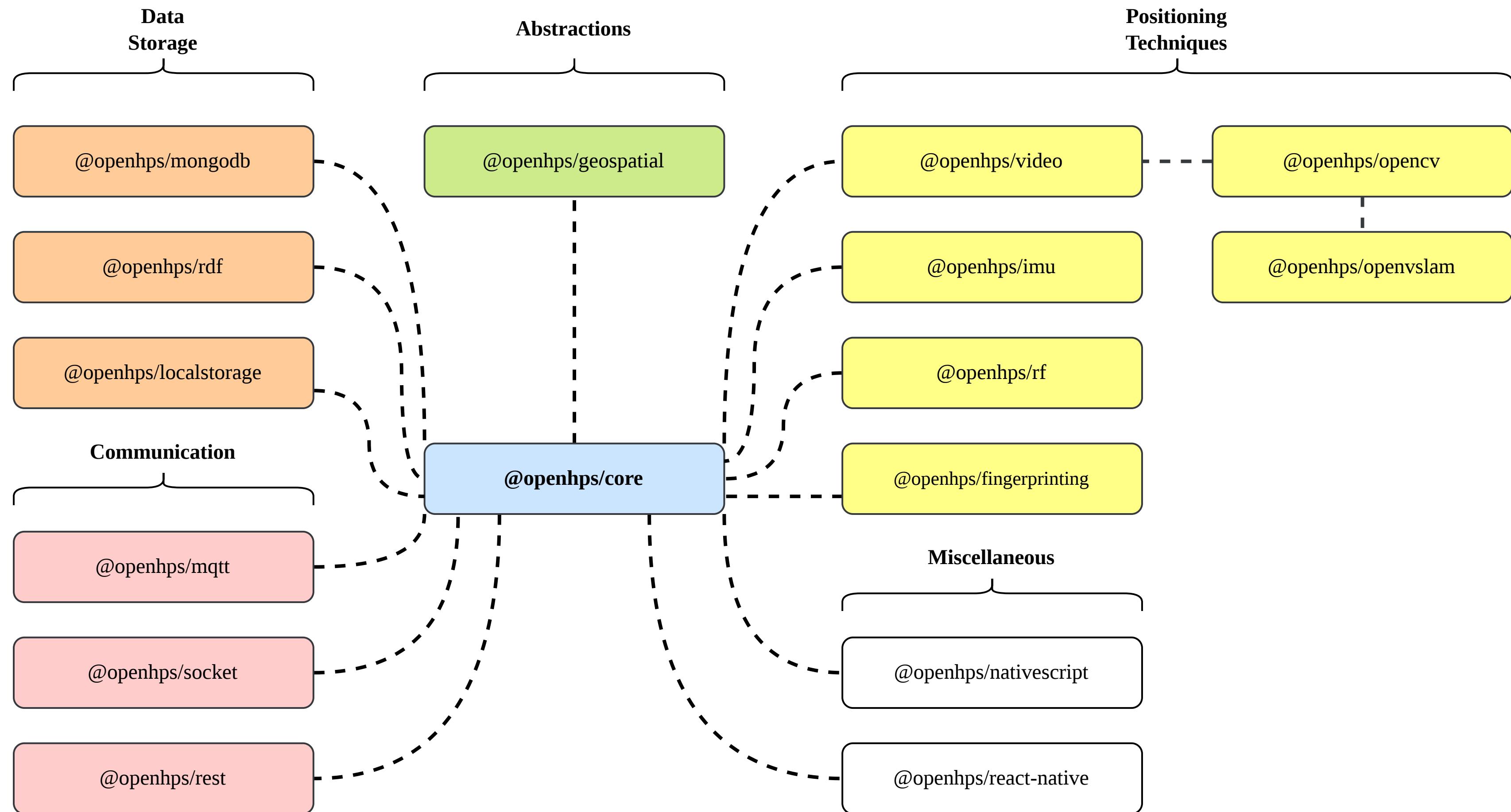
# Process Network Design ...



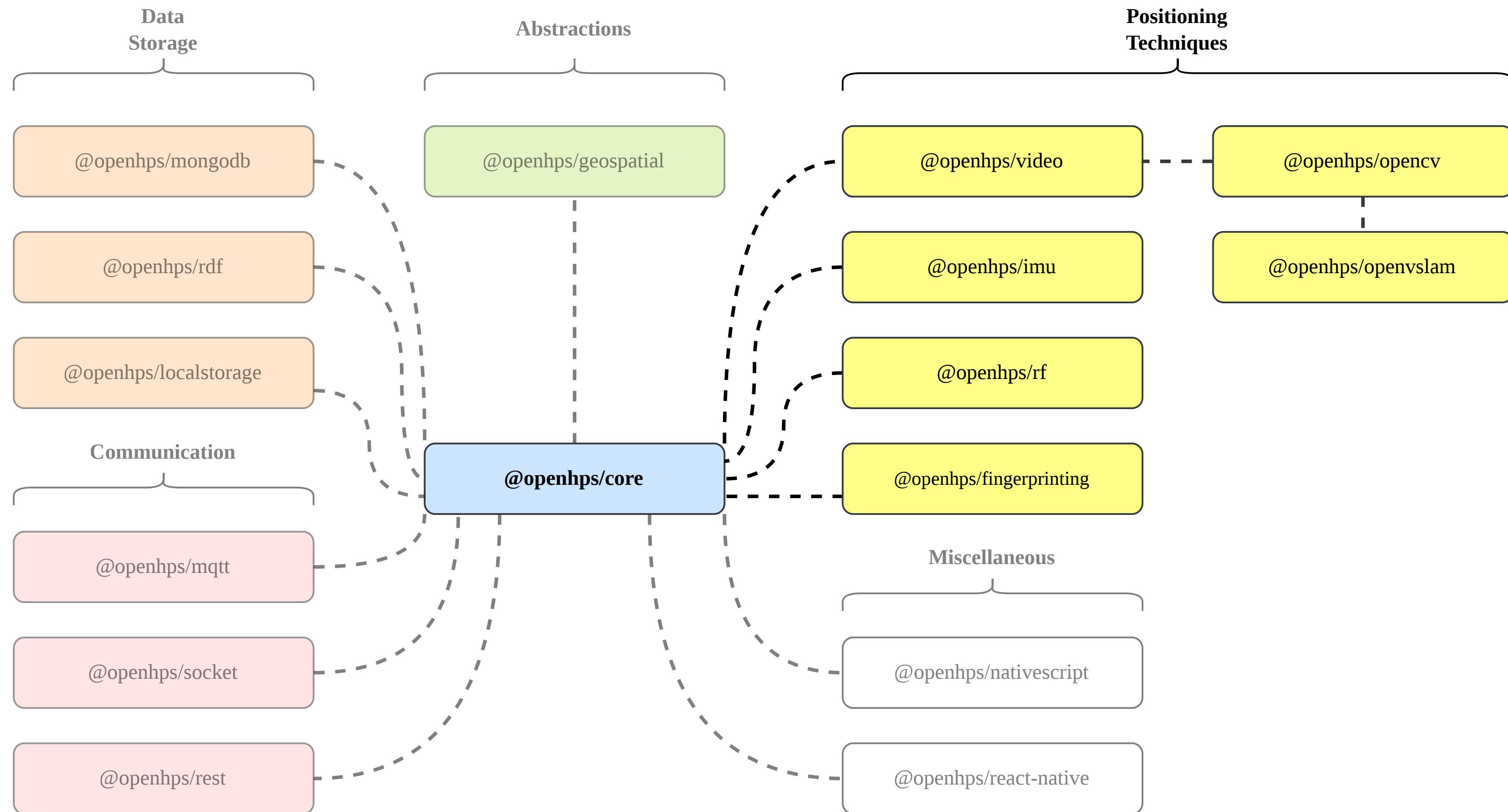
# Modularity



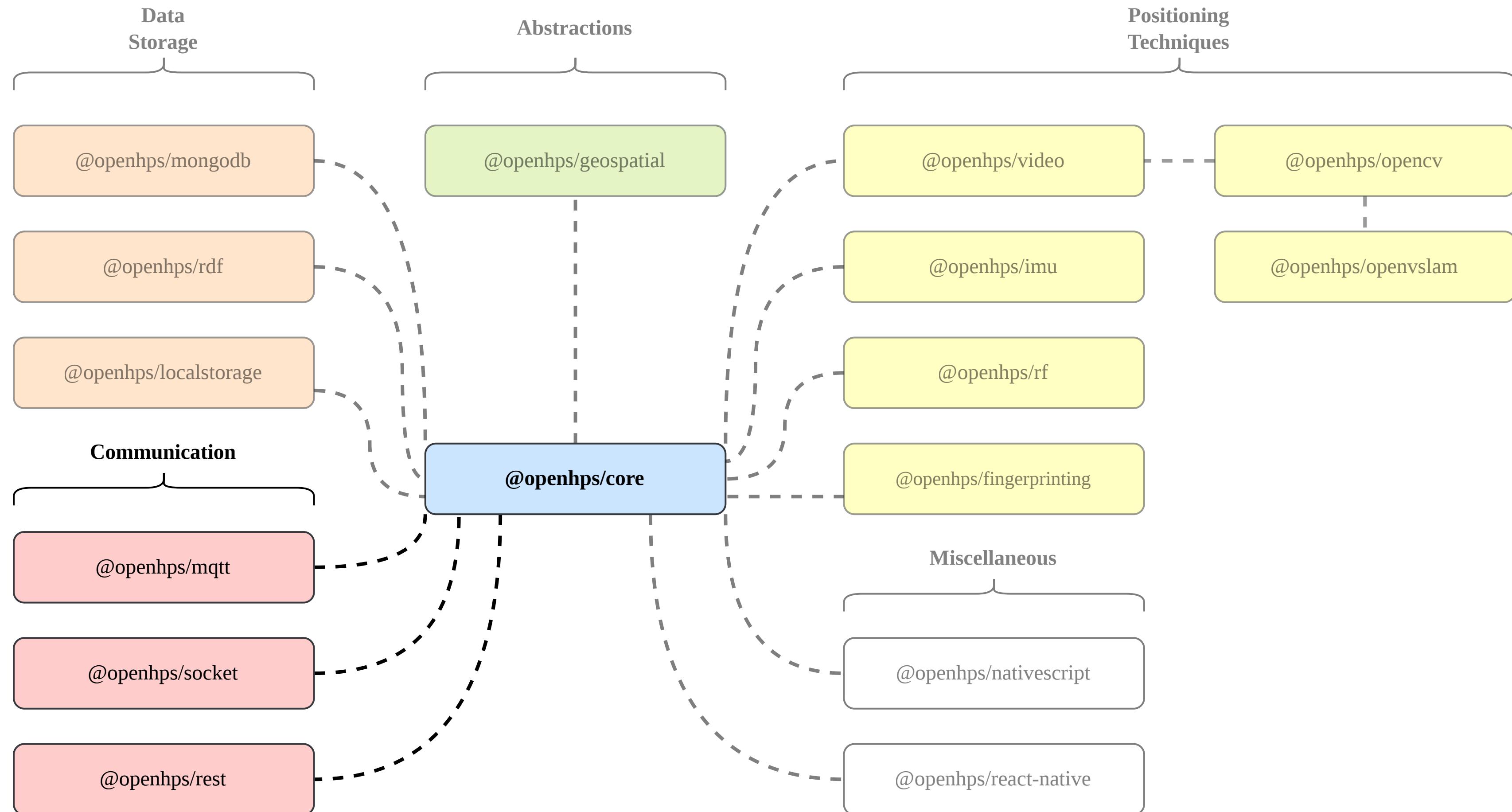
# Modularity ...



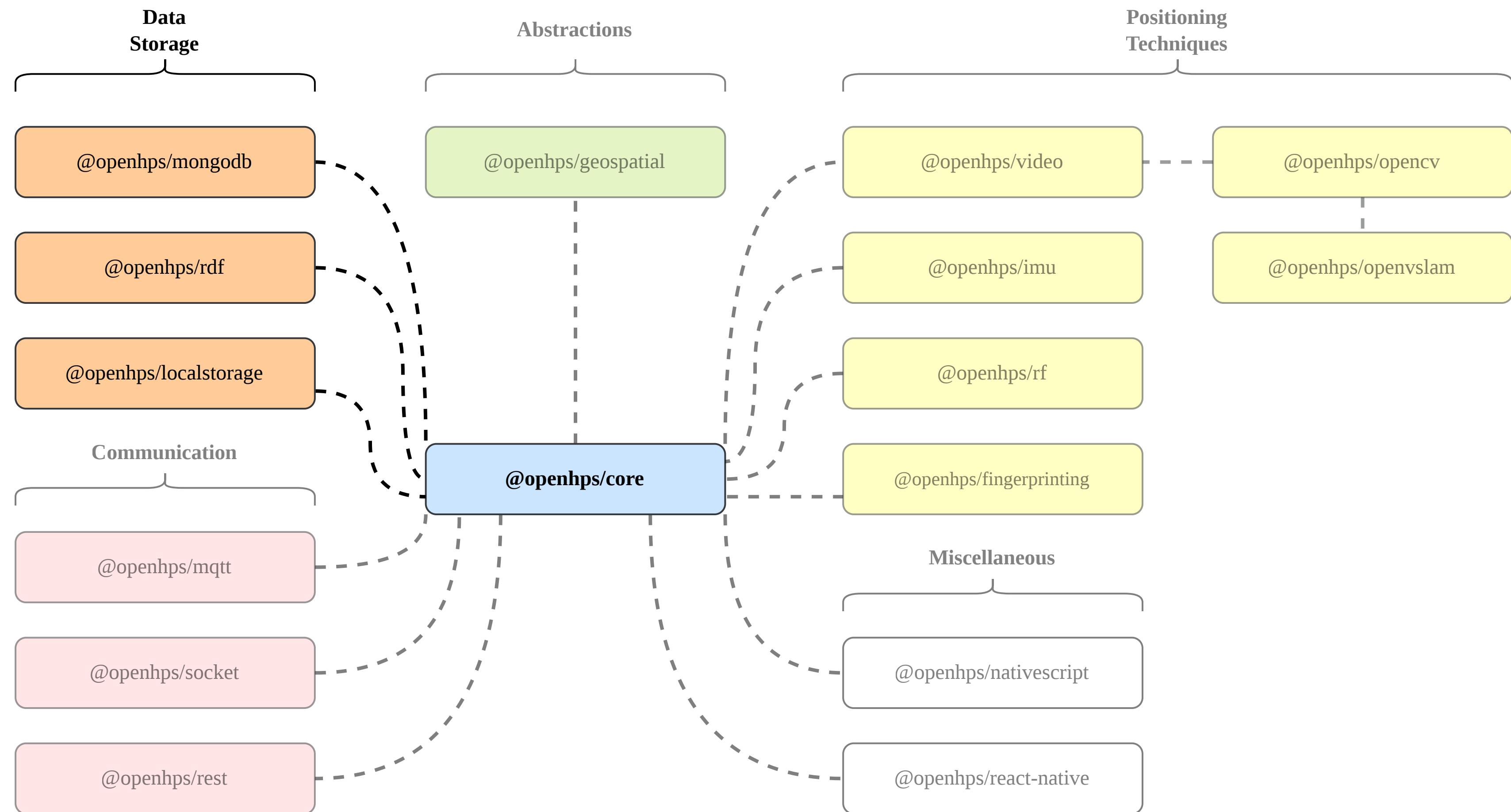
# Modularity ...



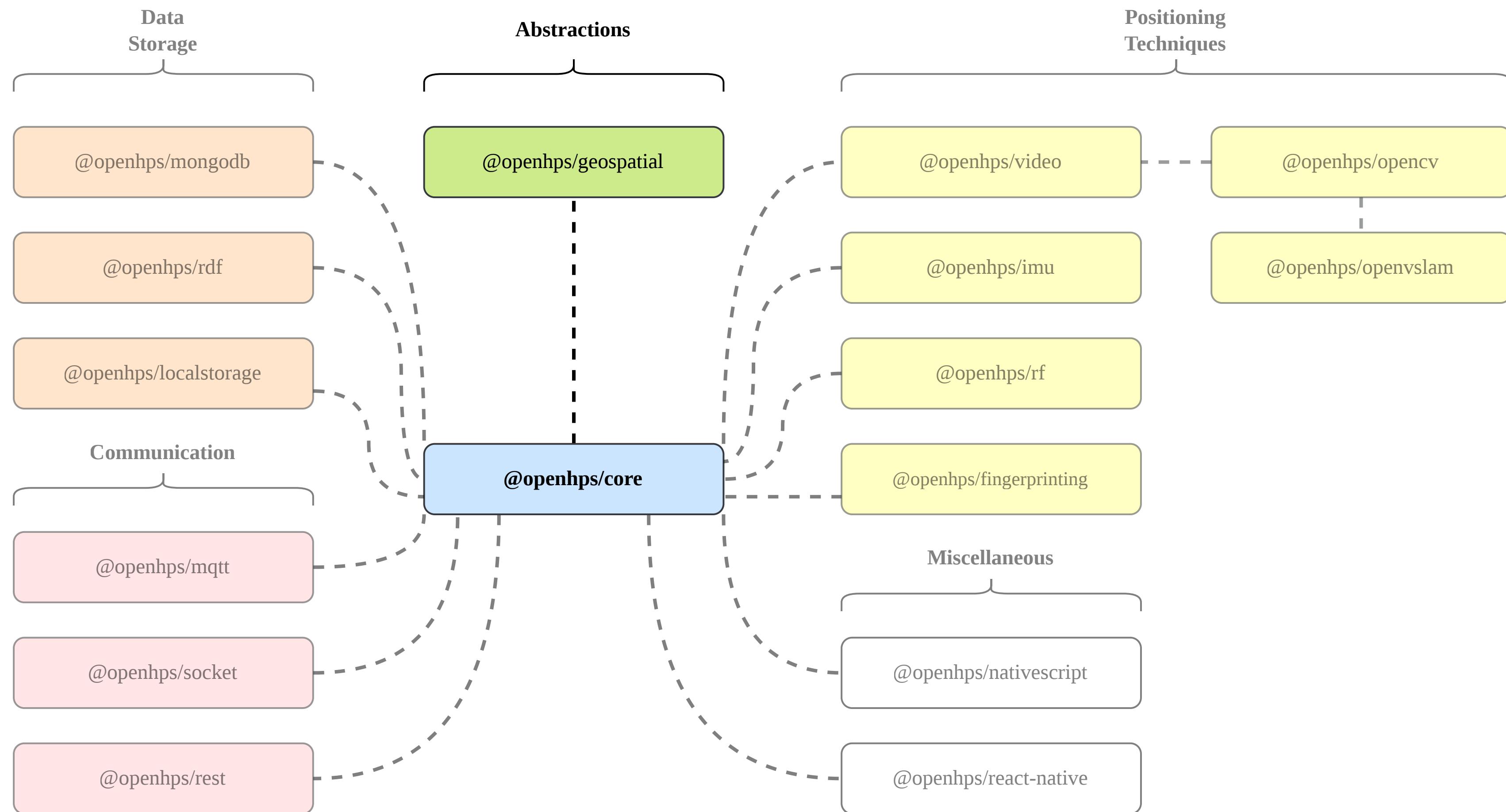
# Modularity ...



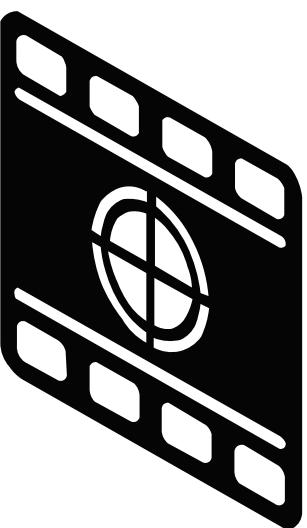
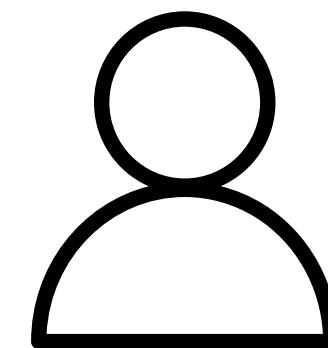
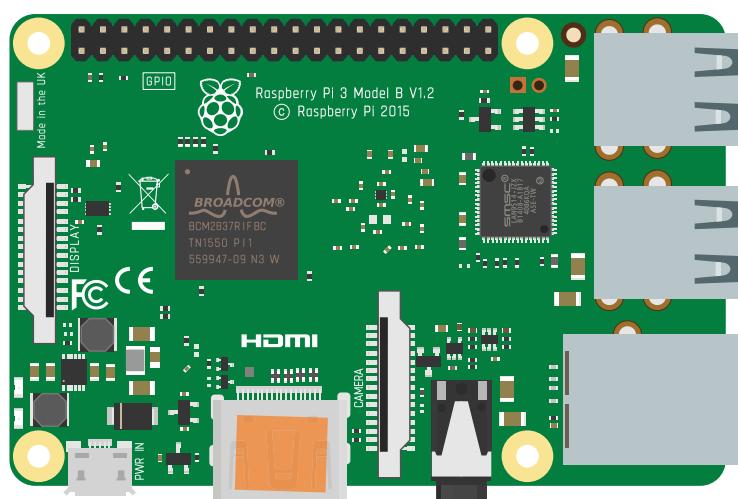
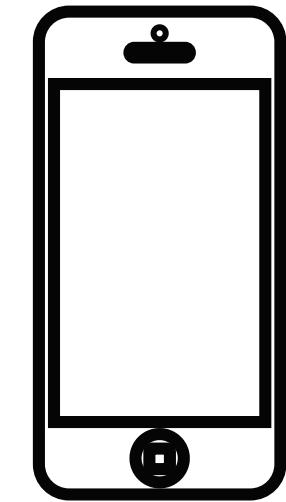
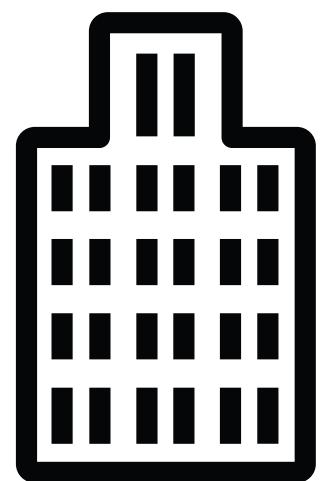
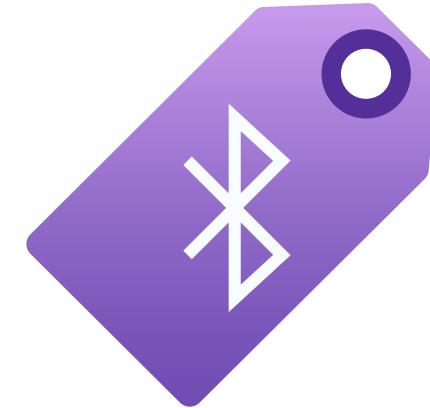
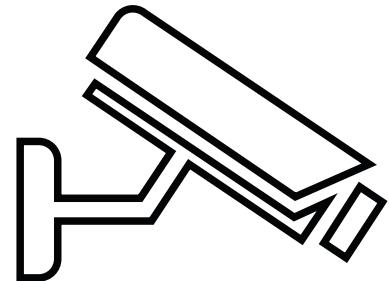
# Modularity ...



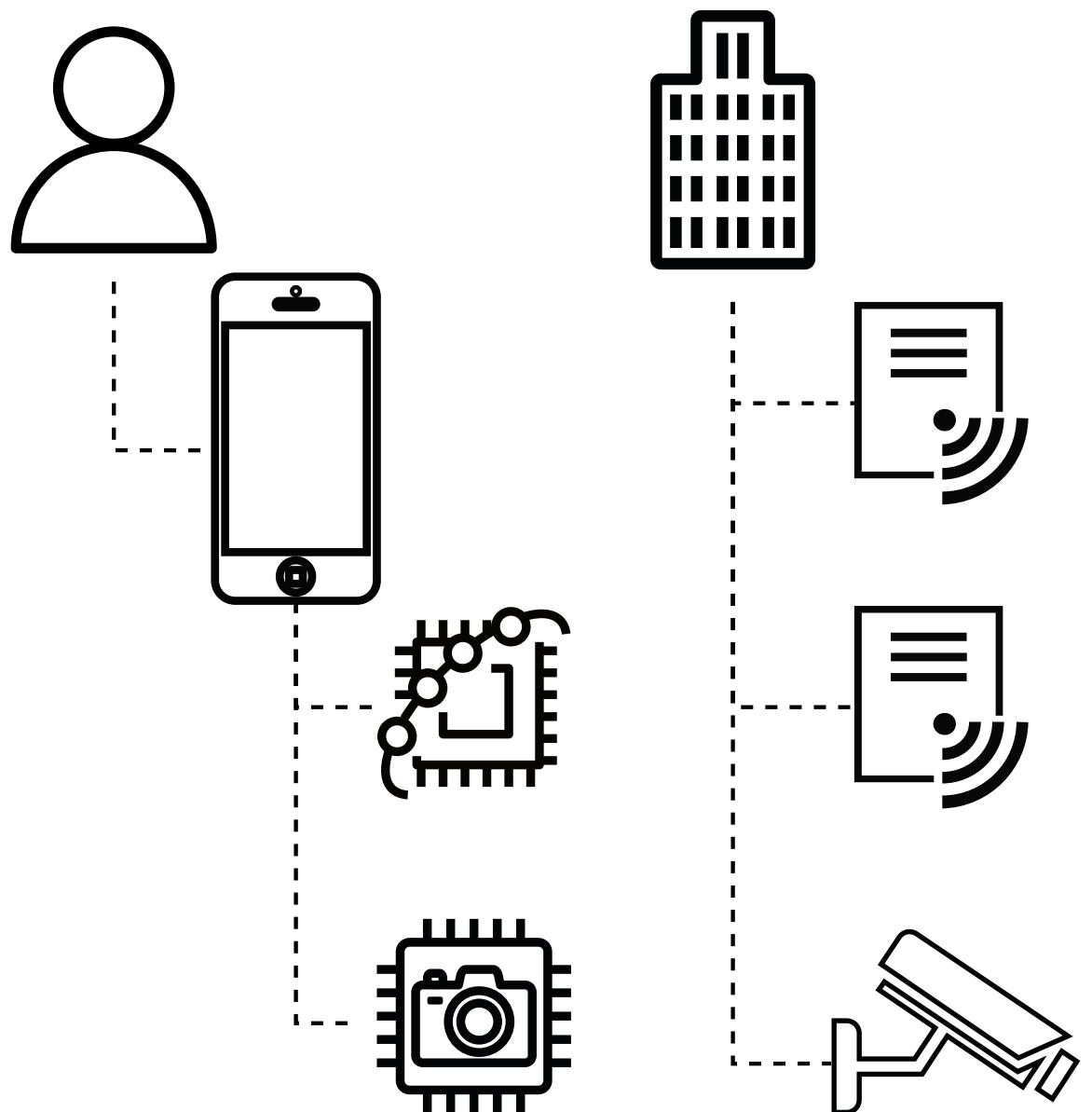
# Modularity ...



# Data Processing



# DataObject



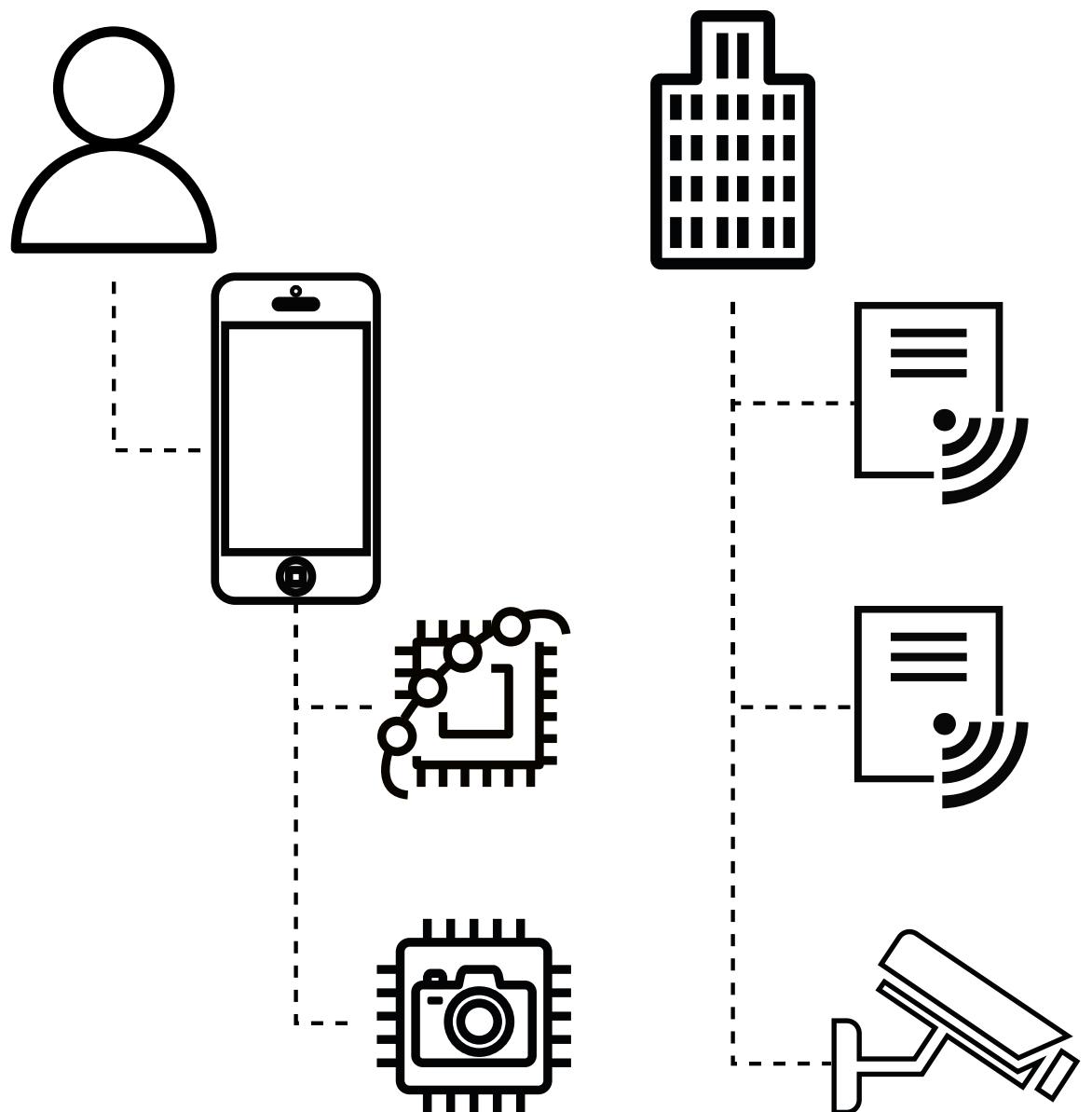
```
// Data object for the person we are tracking
const me = new DataObject("mvdewync@vub.be");
me.displayName = "Maxim Van de Wynckel";

// Phone belonging to the person
const phone = new DataObject()
phone.displayName = "Maxim's Phone";
phone.setParent(me);

// Watch belonging to the person
const watch = new DataObject();
watch.displayName = "Maxim's Android Watch";
watch.setParent(me);

// IP camera identified by MAC
const camera = new CameraObject("80:bb:7c:37:0e:02");
camera.width = 1980;
camera.height = 1024;
camera.fps = 30;
camera.setPosition(/* ... */);
```

# DataObject



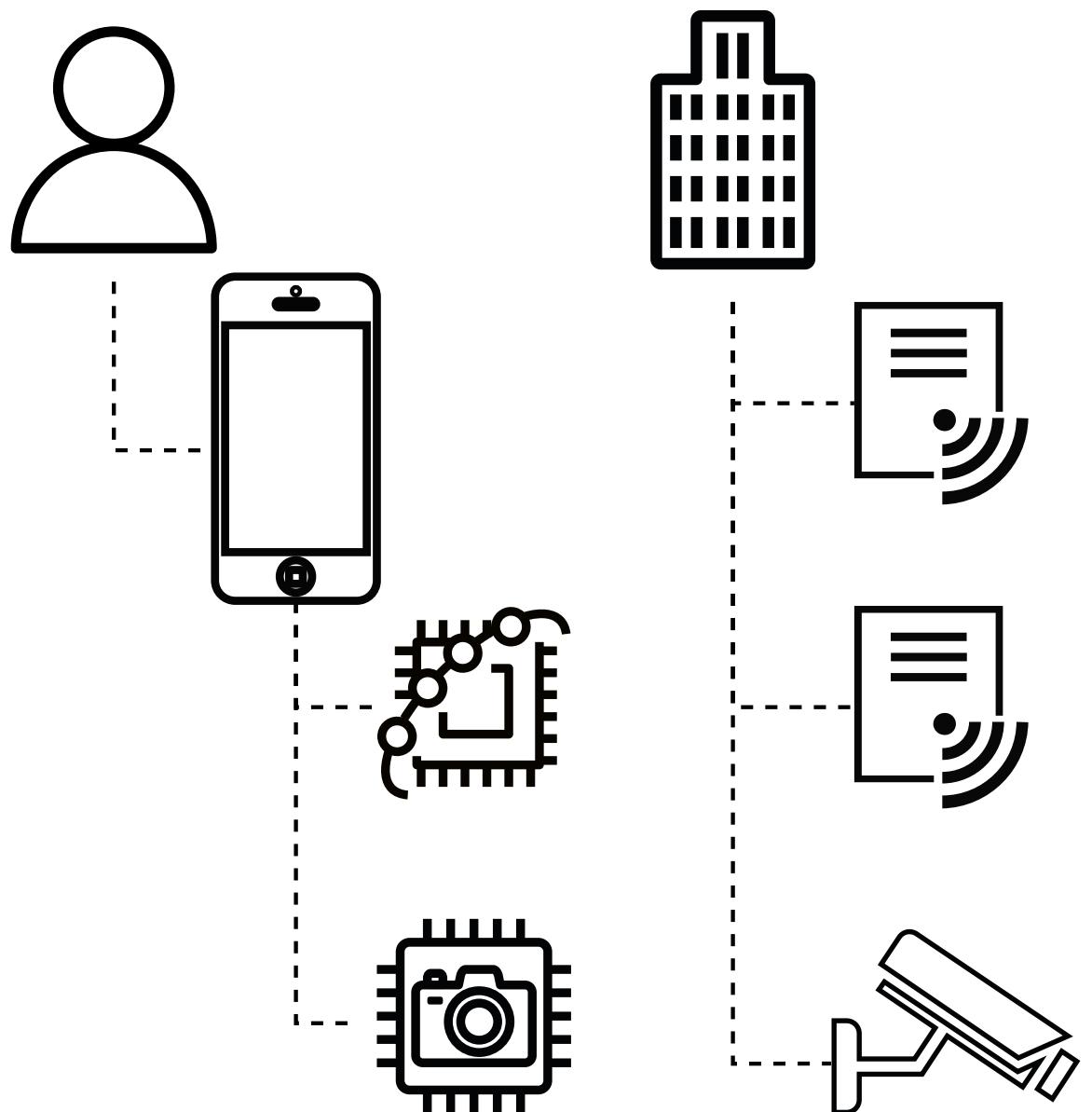
```
// Data object for the person we are tracking
const me = new DataObject("mvdewync@vub.be");
me.displayName = "Maxim Van de Wynckel";

// Phone belonging to the person
const phone = new DataObject();
phone.displayName = "Maxim's Phone";
phone.setParent(me);

// Watch belonging to the person
const watch = new DataObject();
watch.displayName = "Maxim's Android watch";
watch.setParent(me);

// IP camera identified by MAC
const camera = new CameraObject("80:bb:7c:37:0e:02");
camera.width = 1980;
camera.height = 1024;
camera.fps = 30;
camera.setPosition(/* ... */);
```

# DataObject



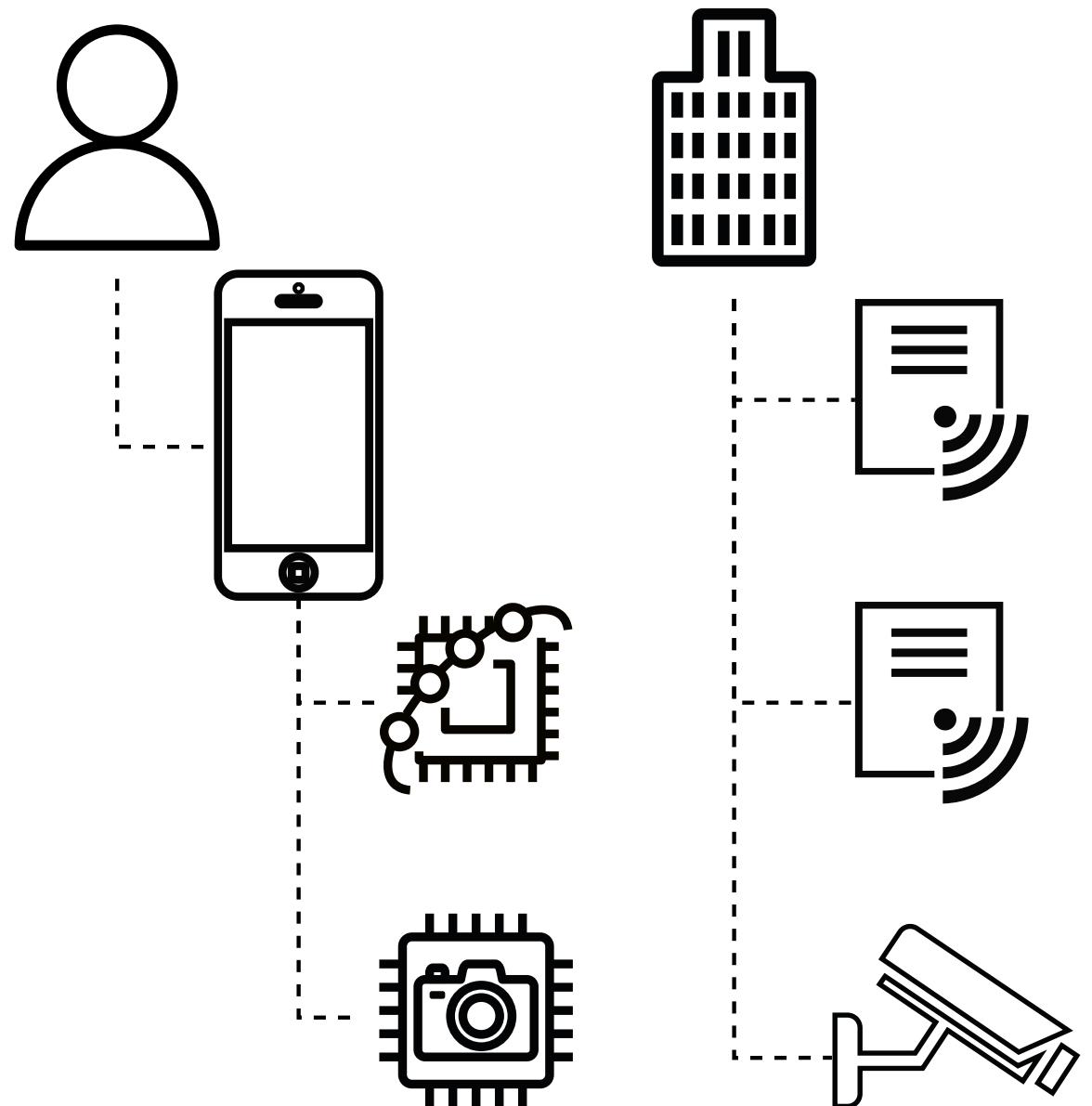
```
// Data object for the person we are tracking
const me = new DataObject("mvdewync@vub.be");
me.displayName = "Maxim Van de Wynckel";

// Phone belonging to the person
const phone = new DataObject();
phone.displayName = "Maxim's Phone";
phone.setParent(me);

// Watch belonging to the person
const watch = new DataObject();
watch.displayName = "Maxim's Android watch";
watch.setParent(me);

// IP camera identified by MAC
const camera = new CameraObject("80:bb:7c:37:0e:02");
camera.width = 1980;
camera.height = 1024;
camera.fps = 30;
camera.setPosition(/* ... */);
```

# DataObject



```
// Data object for the person we are tracking
const me = new DataObject("mvdewync@vub.be");
me.displayName = "Maxim Van de Wynckel";

// Phone belonging to the person
const phone = new DataObject();
phone.displayName = "Maxim's Phone";
phone.setParent(me);

// Watch belonging to the person
const watch = new DataObject();
watch.displayName = "Maxim's Android watch";
watch.setParent(me);

// IP camera identified by MAC
const camera = new CameraObject("80:bb:7c:37:0e:02");
camera.width = 1980;
camera.height = 1024;
camera.fps = 30;
camera.setPosition(/* ... */);
```

# Absolute and Relative Positions



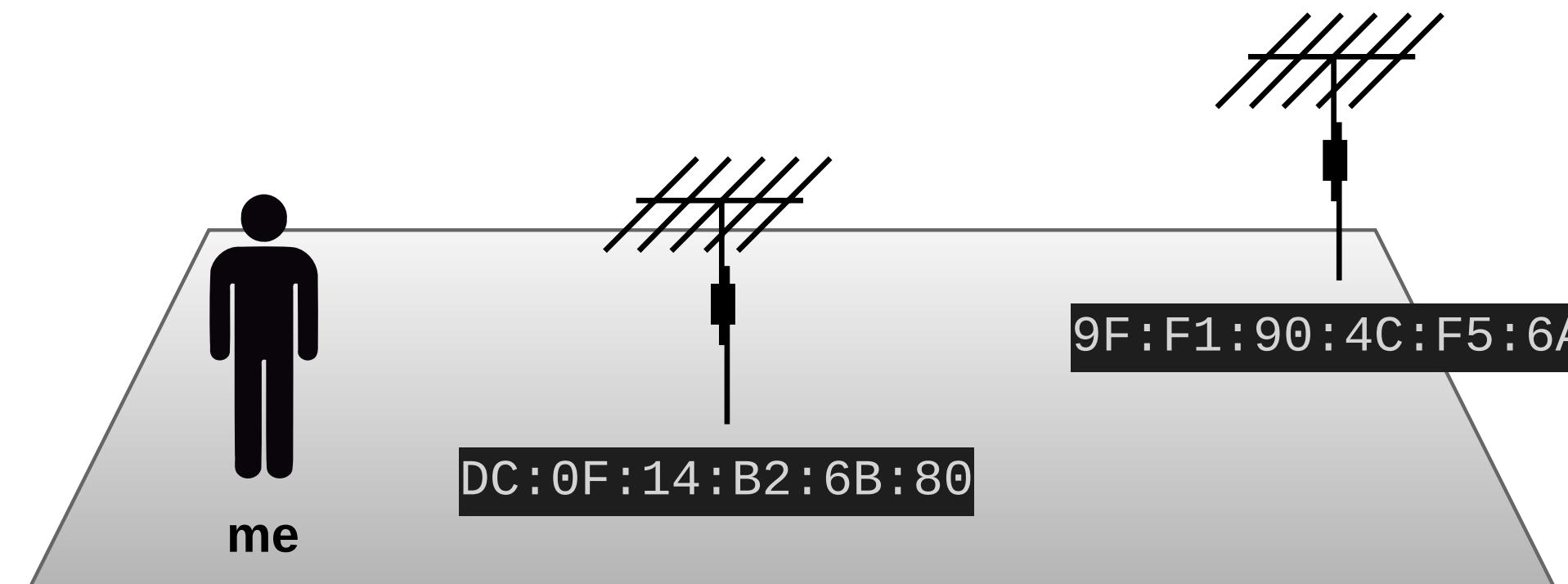
## Absolute

- ▶ 2D, 3D, geographical, ...
- ▶ Within a reference space

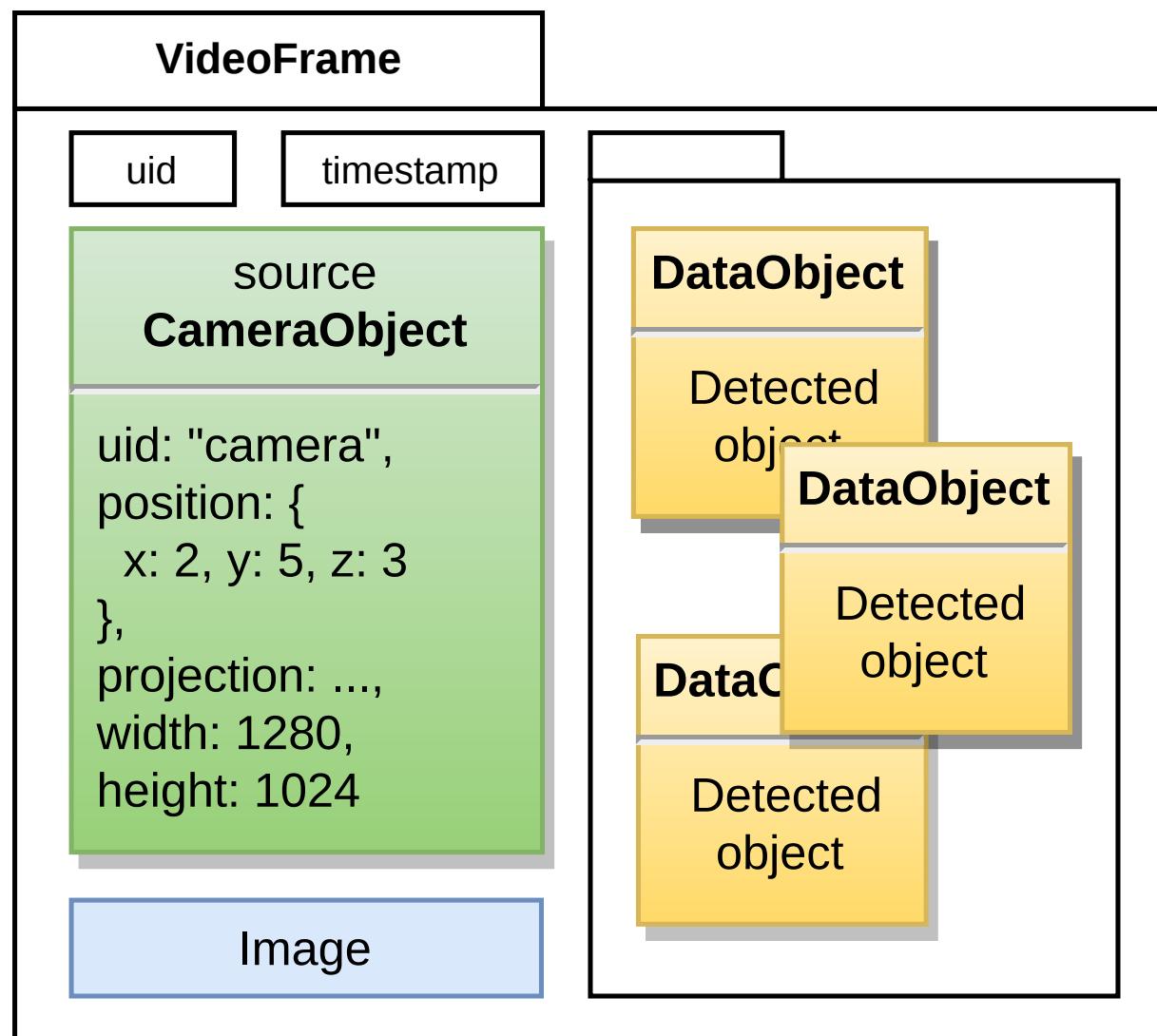
## Relative

- ▶ Distance, angle, velocity, ...
- ▶ Relative to another *object*

```
// Absolute geographical position  
me.setPosition(new GeographicalPosition(  
    50.8204, 4.3921  
));  
  
// Relative position(s) to another object  
me.addRelativePosition(new RelativeDistance(  
    "9F:F1:90:4C:F5:6A", 5.2, LengthUnit.METER  
));  
me.addRelativePosition(new RelativeDistance(  
    "DC:0F:14:B2:6B:80", 1.4, LengthUnit.METER  
));
```



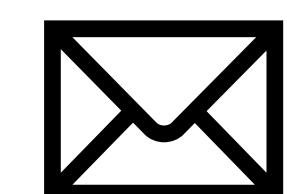
# DataFrame



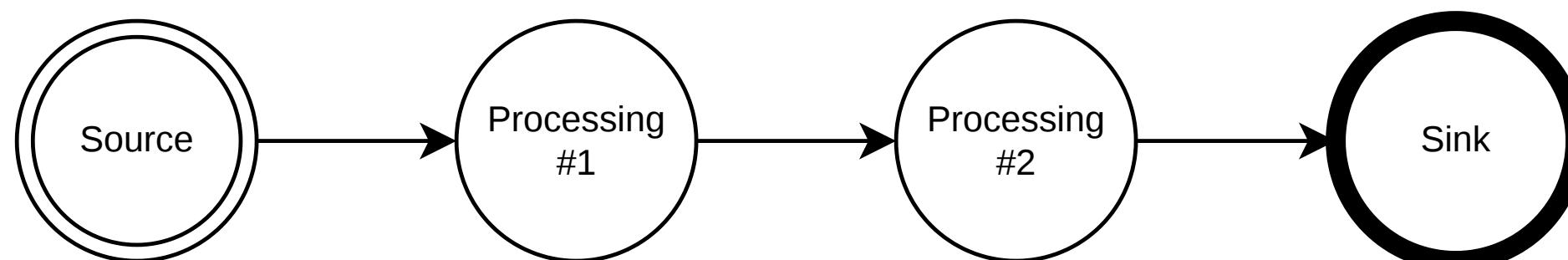
```
// Sensor that captured the frame
const camera = new CameraObject();

// Create a new frame
const frame = new VideoFrame();
frame.source = camera;
frame.image = myImage;

// Add detected objects to frame
frame.addObject(/* ... */);
frame.addObject(/* ... */);
frame.addObject(/* ... */);
```

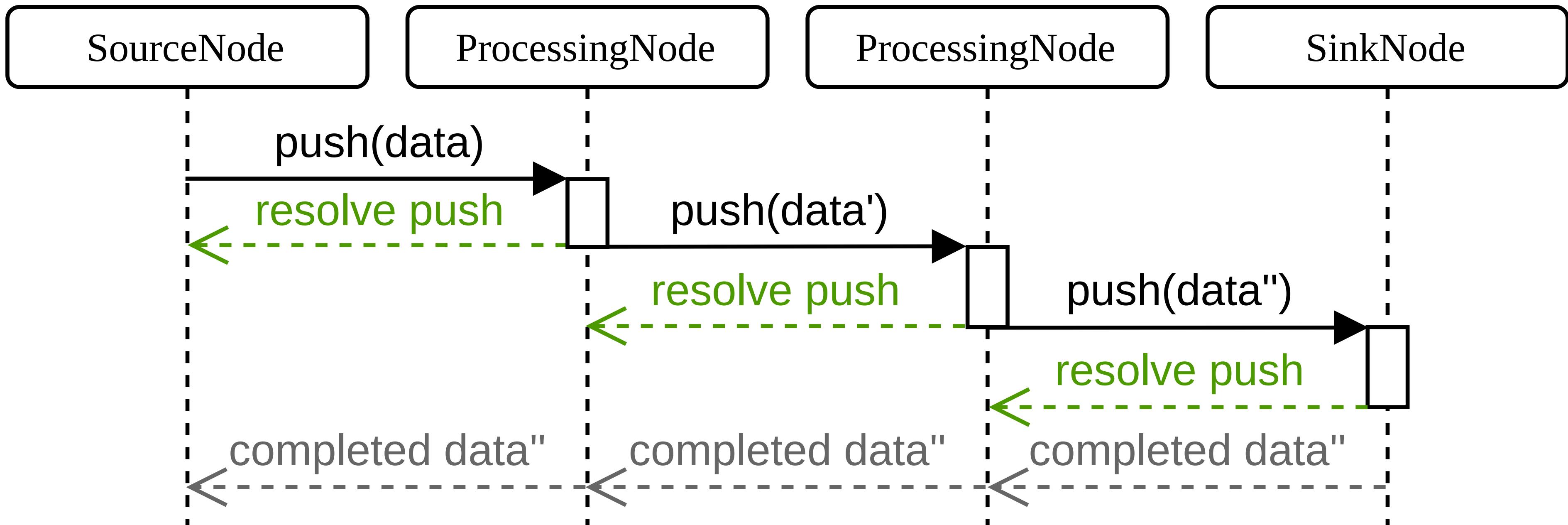


**DataFrame**



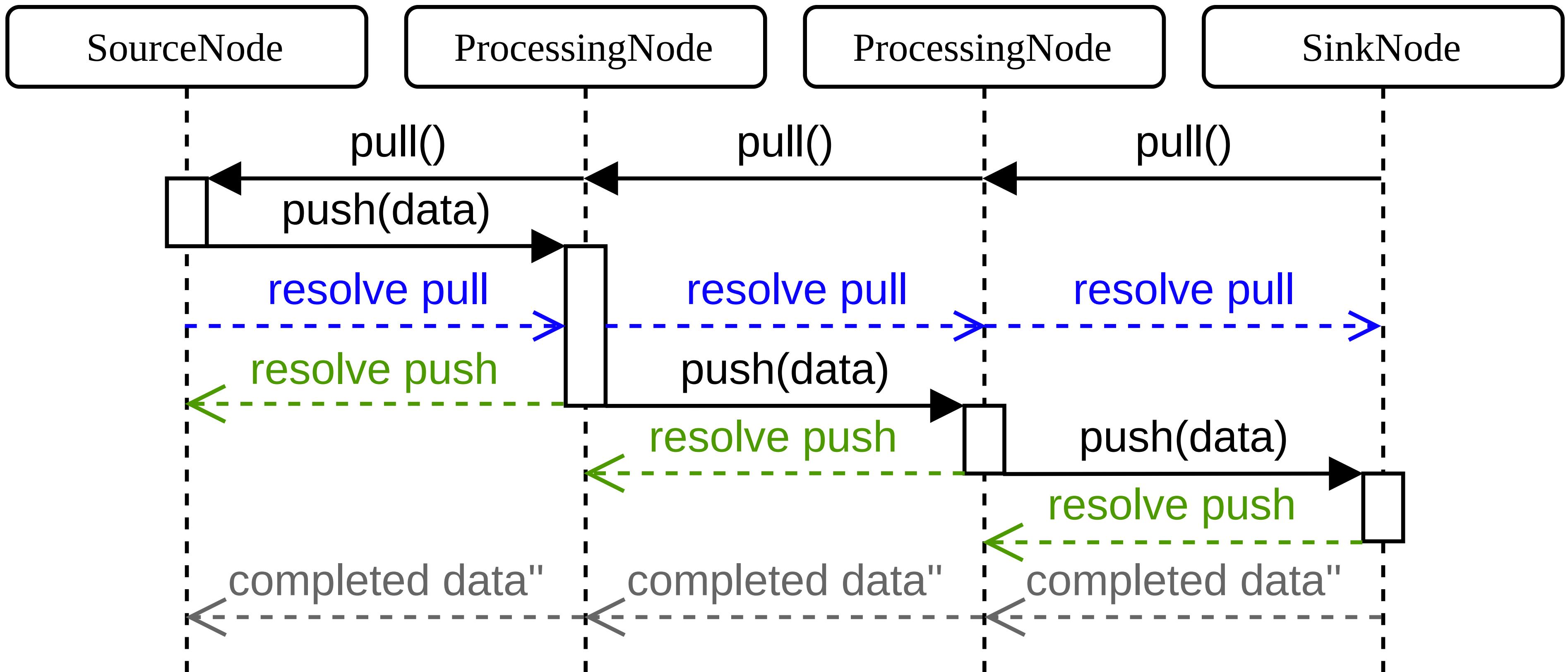
# DataFrame ...

## Pushing Data



# DataFrame ...

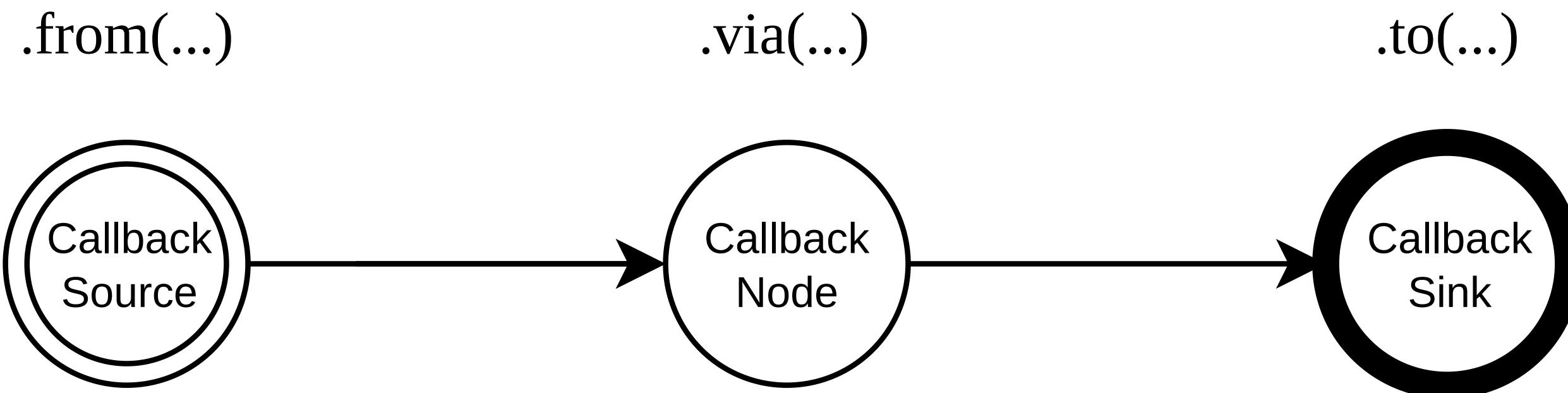
## Pulling Data



# Positioning Model



```
ModelBuilder.create()
  .from(new CallbackSourceNode(() => {
    const myObject = new DataObject("mvdewync");
    const frame = new DataFrame();
    frame addObject(myObject);
    return frame;
  )))
  .via(new CallbackNode((frame: DataFrame) => { /* ... */ }))
  .to(new CallbackSinkNode((frame: DataFrame) => { /* ... */ }))
  .build().then((model: Model) => { /* ... */});
```



# Exercises Dataset



## OpenHPS Fingerprinting Dataset Overview

The OpenHPS fingerprinting dataset of March 2021 contains **110 training** and **30 test** locations in **4 different orientations**. We collected information on Wi-Fi access point signal strengths, **11 BLE beacons** and IMU data at a requested refresh rate of 50Hz. More information about the contents of the dataset can be found [here](#).

BLE Beacons Train Data Test Data

dataset

Our dataset is recorded on Sunday the 7th of March 2021 in the building: "Pleinlaan 9" on the third floor.

### Wireless Access Points

Our dataset captures signal strength from **220** wireless access points. Based on the SSID before anonymisation, we identified **199** access points to be stable (university network, department network, ...). Other (unstable) access points were identified as hotspots, printers and unknown company networks inside the same building.

ID	SSID_ID	FREQUENCY	STABLE
WAP_001	1	2,452	1
WAP_002	1	5,180	1
WAP_003	2	2,432	0
WAP_004	3	2,462	0
WAP_005	4	2,462	1
WAP_006	5	5,220	1
WAP_007	6	5,180	1
WAP_008	6	5,220	1
WAP_009	7	2,412	1
WAP_010	6	5,220	1

# Exercise 1 - Multilateration



## Question 1

*Create a new Bluetooth beacon object with the ID column as its uid and an absolute 3d position for the X, Y and Z coordinates. Return the beacon object in this function.*

## Question 2

*Create a relative position for the rssi and beacon. Use the RelativeRSSI class in combination with a new BLEObject that uses the beacon name as its UID*

# Exercise 1 ...

## Question 3

*Convert the RSSI to a distance using log distance propagation. Assume we have a calibrated received signal strength of -69 at 1 meter distance. Play around with the environment factor to see what works best.*

## Question 4

*Use the RSSI converted to a distance (from question 2) to calculate the position with multilateration.*

# Exercise 2 - Fingerprinting



## Question 1

*Create a fingerprint service that will store the scene analysis. Experiment with the default RSSI value and grouping*

## Question 2

*Create a KNN fingerprinting node to convert sensor data to a position. Experiment with the parameters to obtain a better result.*

# Exercise 3 - Location Awareness



## Question 1

*Load the spaces/features defined in our dataset (spaces.geo.json). The file is already loaded as the variable GEOJSON\_FEATURES.*

## Question 2

*Emit the event of entering and exiting a space. This event should only trigger when you enter a new space. The 'enterspace' event should only trigger when you enter a new space. The 'exitspace' event should only trigger when you leave a space.*