Indoor Positioning Using the OpenHPS Framework

Maxim Van de Wynckel, Beat Signer

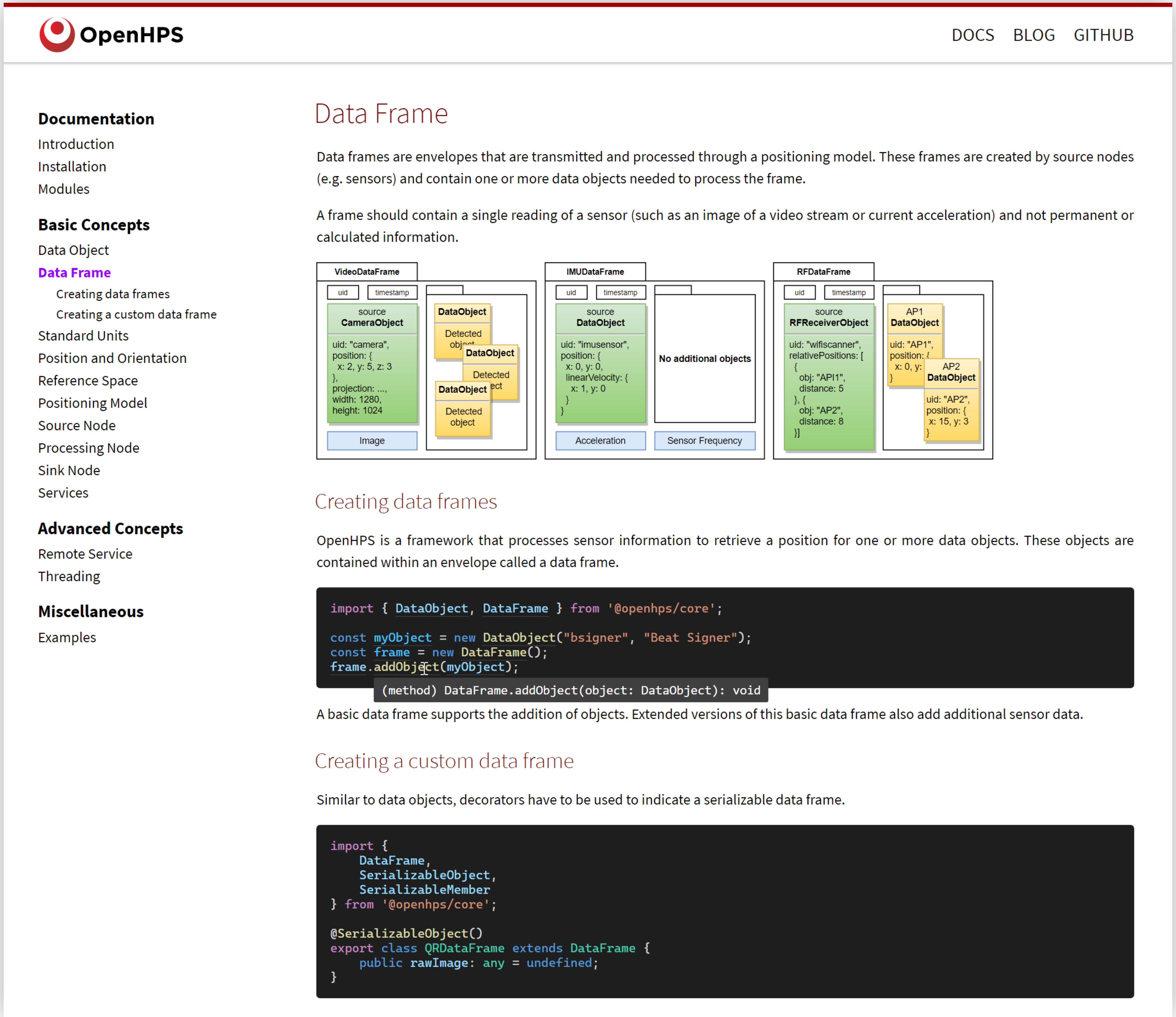
Web & Information Systems Engineering Lab
Vrije Universiteit Brussel

VUB VRIJE UNIVERSITEIT BRUSSEL WISE WEB & INFORMATION SYSTEMS ENGINEERING OpenHPS

What is OpenHPS?

OpenHPS

An Open Source Hybrid Positioning System



OpenHPS

DOCS BLOG GITHUB

Documentation

- Introduction
- Installation
- Modules

Basic Concepts

- Data Object
- Data Frame
  - Creating data frames
  - Creating a custom data frame
- Standard Units
- Position and Orientation
- Reference Space
- Positioning Model
- Source Node
- Processing Node
- Sink Node
- Services

Advanced Concepts

- Remote Service
- Threading

Miscellaneous

- Examples

Data Frame

Data frames are envelopes that are transmitted and processed through a positioning model. These frames are created by source nodes (e.g. sensors) and contain one or more data objects needed to process the frame.

A frame should contain a single reading of a sensor (such as an image of a video stream or current acceleration) and not permanent or calculated information.

Creating data frames

OpenHPS is a framework that processes sensor information to retrieve a position for one or more data objects. These objects are contained within an envelope called a data frame.

```
import { DataObject, DataFrame } from '@openhps/core';

const myObject = new DataObject("bsigner", "Beat Signer");
const frame = new DataFrame();
frame.addObject(myObject);
```

```
(method) DataFrame.addObject(object: DataObject): void
```

A basic data frame supports the addition of objects. Extended versions of this basic data frame also add additional sensor data.

Creating a custom data frame

Similar to data objects, decorators have to be used to indicate a serializable data frame.

```
import {
    DataFrame,
    SerializableObject,
    SerializableMember
} from '@openhps/core';

@SerializableObject()
export class QRDataFrame extends DataFrame {
    public rawImage: any = undefined;
}
```

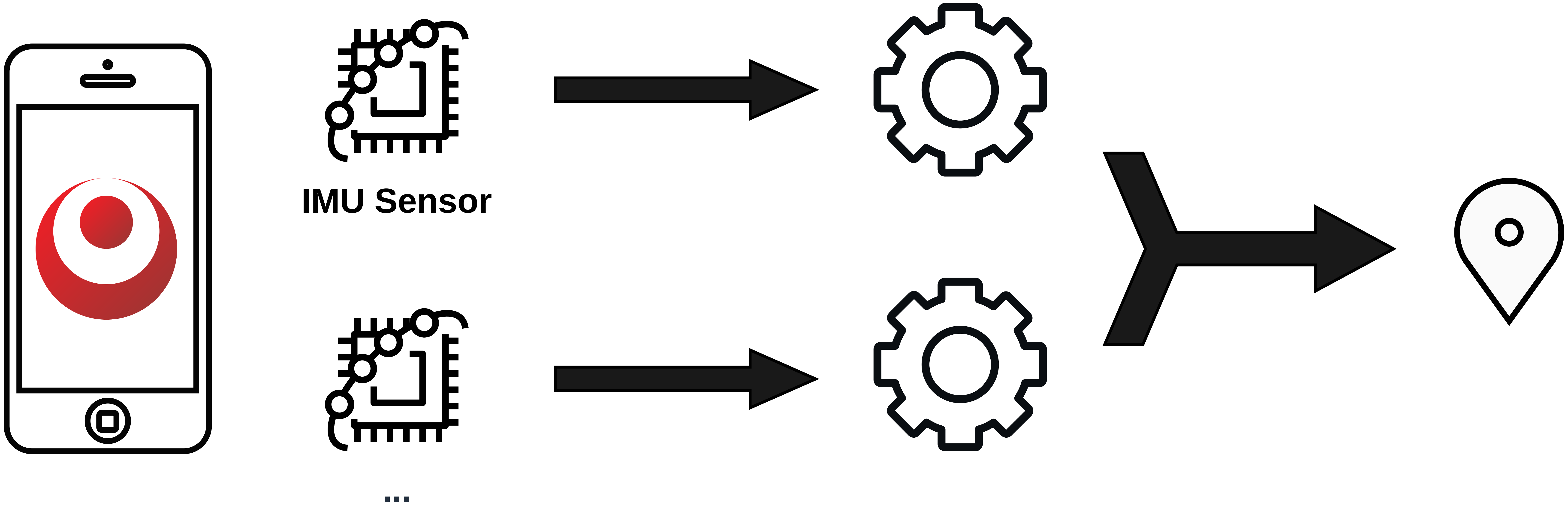
What is OpenHPS?

OpenHPS

An Open Source Hybrid Positioning System

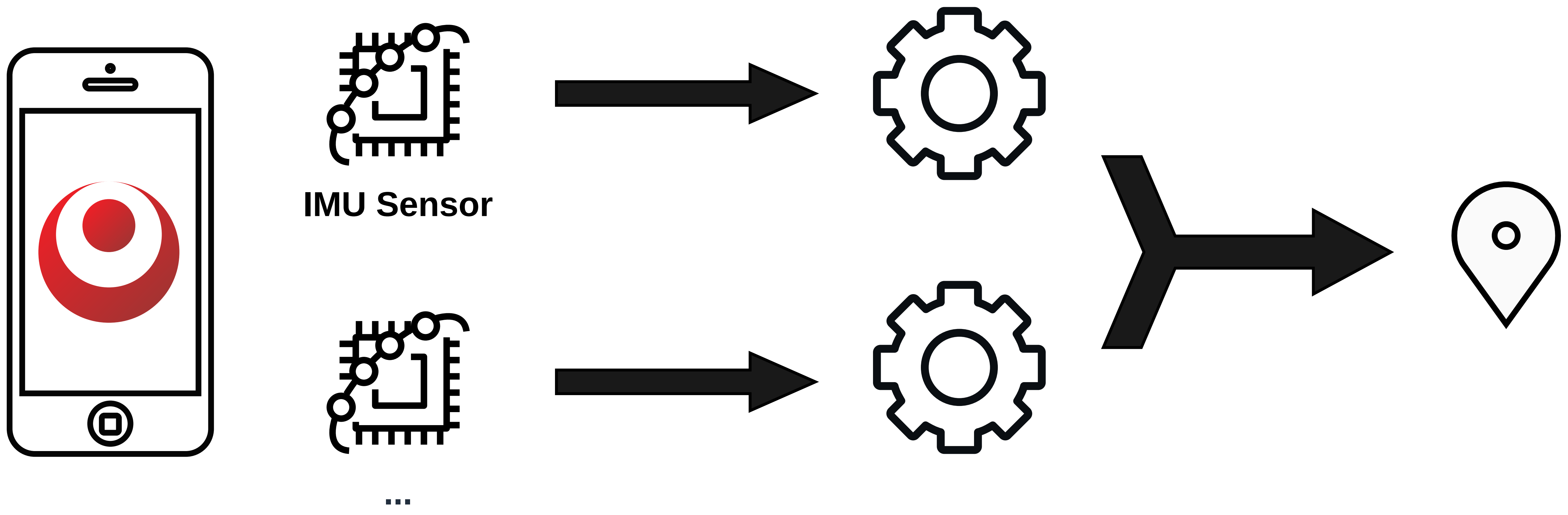
- Any technology
- Any algorithm
- Various use cases
- Flexible processing and output
  - Accuracy over battery consumption, reliability, ...
- Aimed towards
  - Developers
  - Researchers

Process Network Design



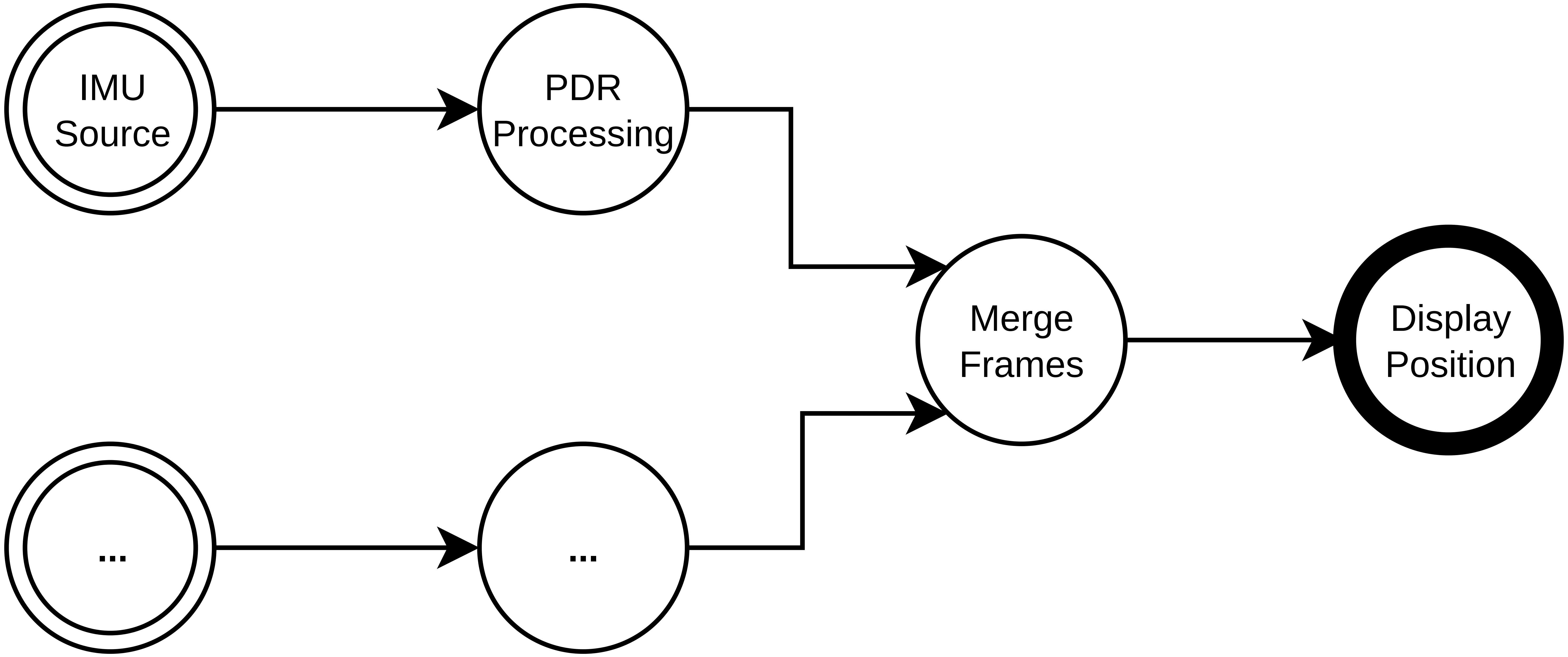
Process Network Design ...

OpenHPS



IMU Sensor

...



IMU Source

PDR Processing

Merge Frames

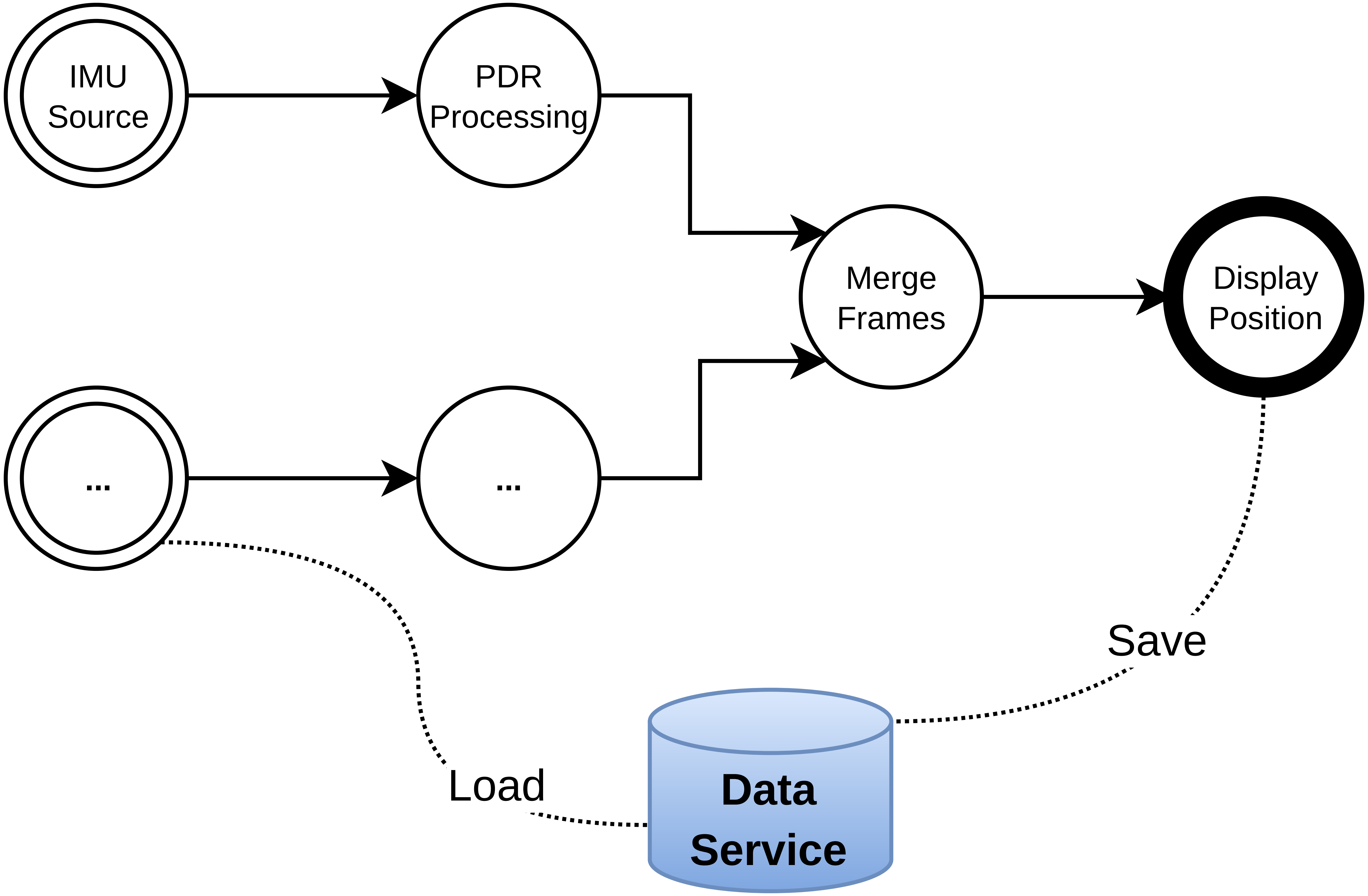
Display Position

...

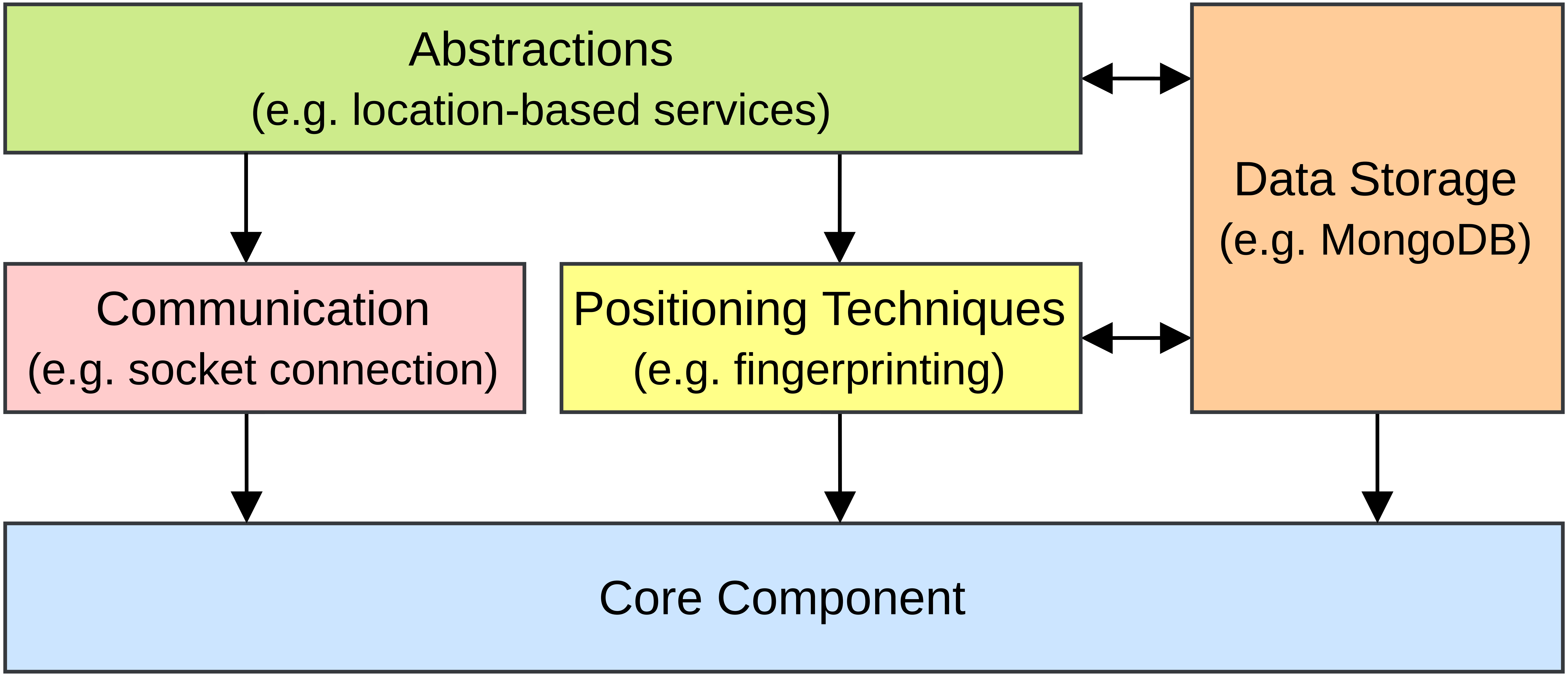
...

Process Network Design ...

OpenHPS



Modularity



Abstractions
(e.g. location-based services)

Communication
(e.g. socket connection)

Positioning Techniques
(e.g. fingerprinting)

Data Storage
(e.g. MongoDB)

Core Component

Modularity ...

Communication

Socket, MQTT, REST API, ...

Data Storage

MongoDB, LocalStorage, RDF, ...

Positioning Algorithms

IMU, fingerprinting, OpenVSLAM, ...

Abstractions

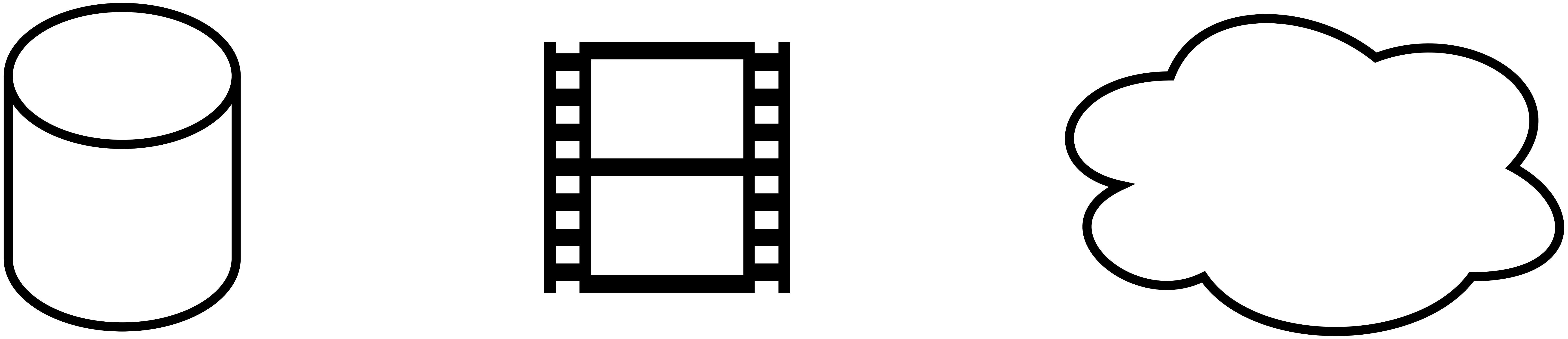
Geospatial, location-based services, geojson, ...

Other

React-Native, NativeScript, Sphero, ...

Data Processing

OpenHPS



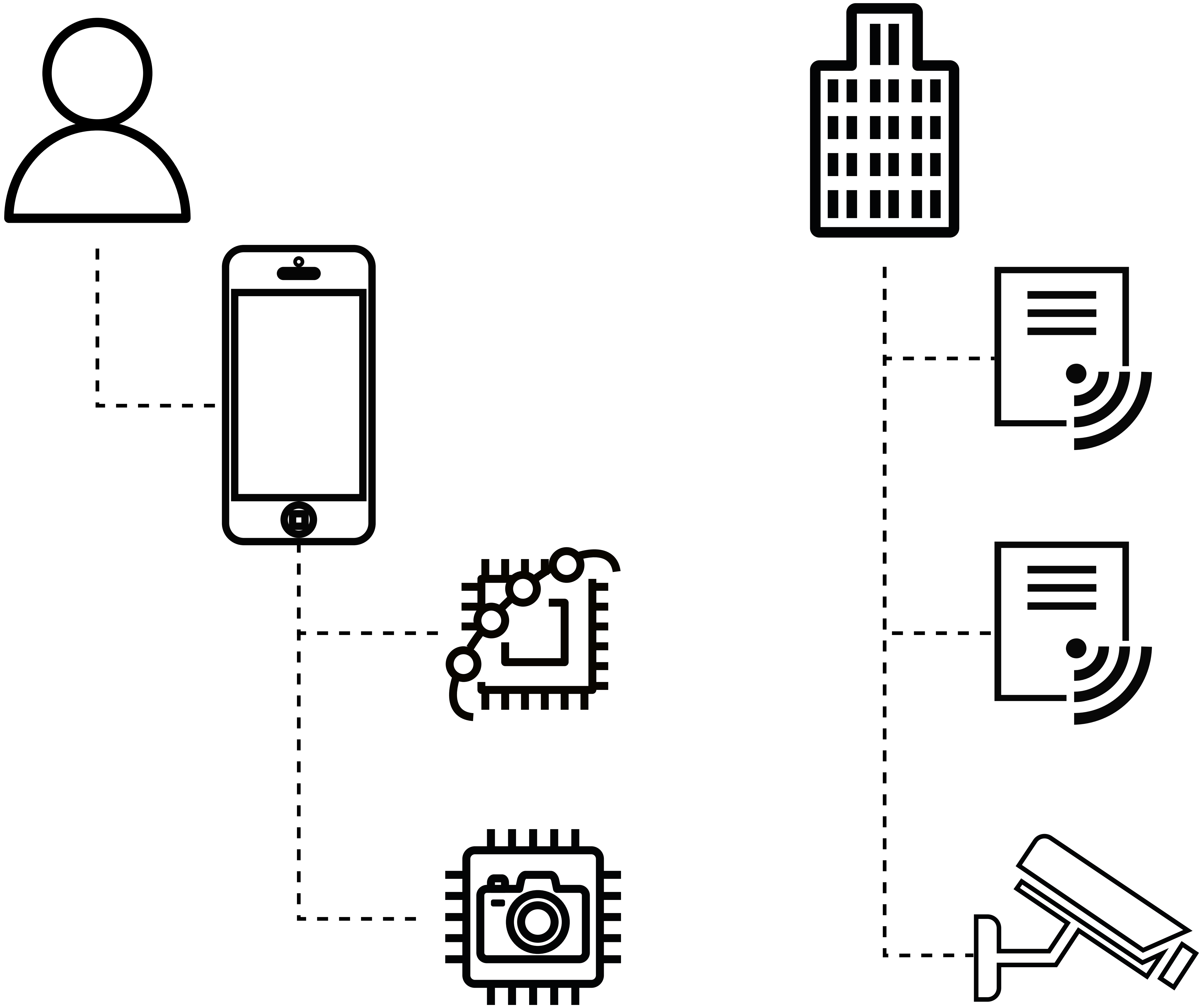
Knowledge

Raw Data

Processed Data

DataObject

OpenHPS



Absolute and Relative Positions

Absolute

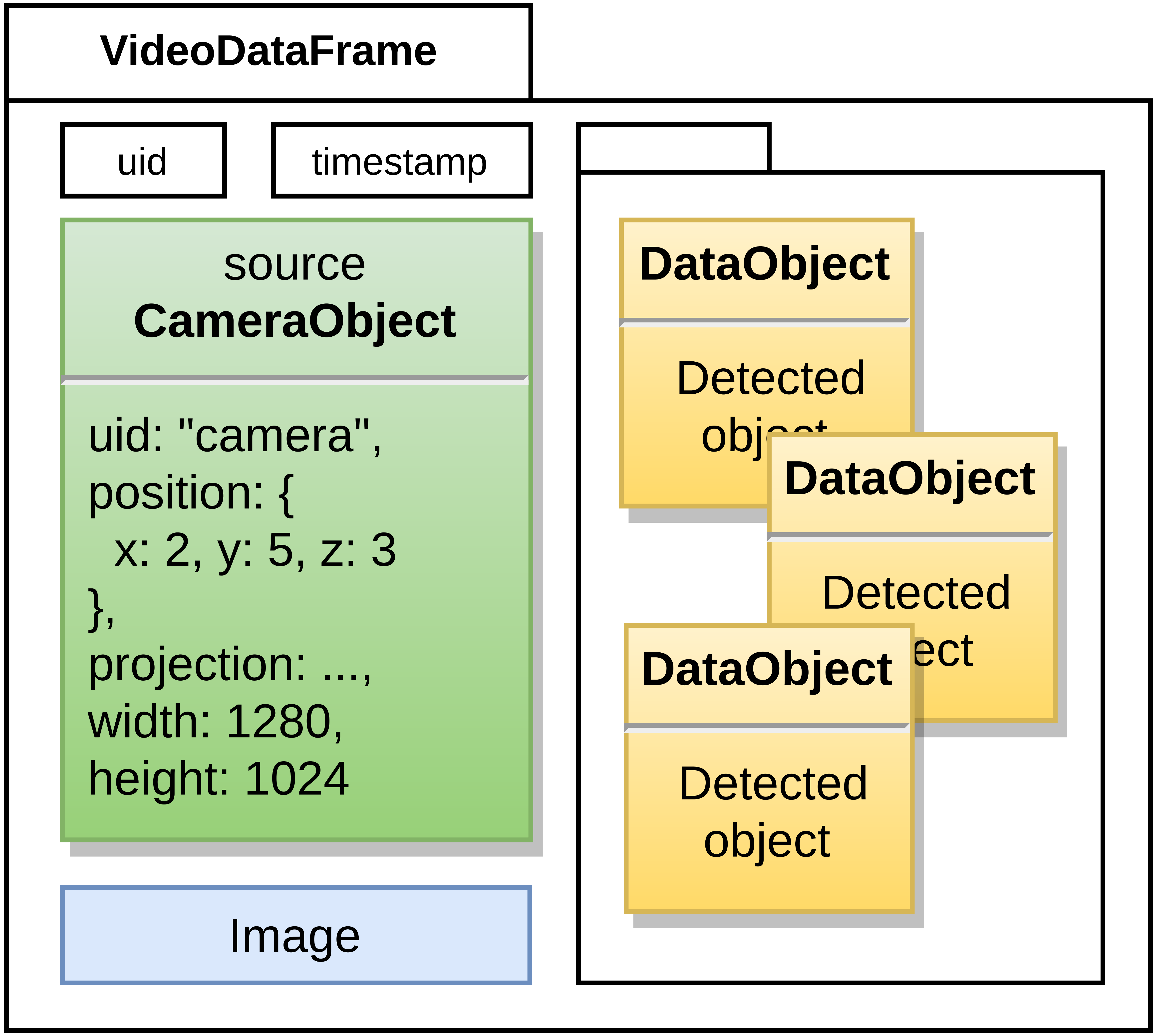
- 2D, 3D, Geographical, ...

Relative

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

DataFrame

OpenHPS



VideoDataFrame

uid

timestamp

source
CameraObject

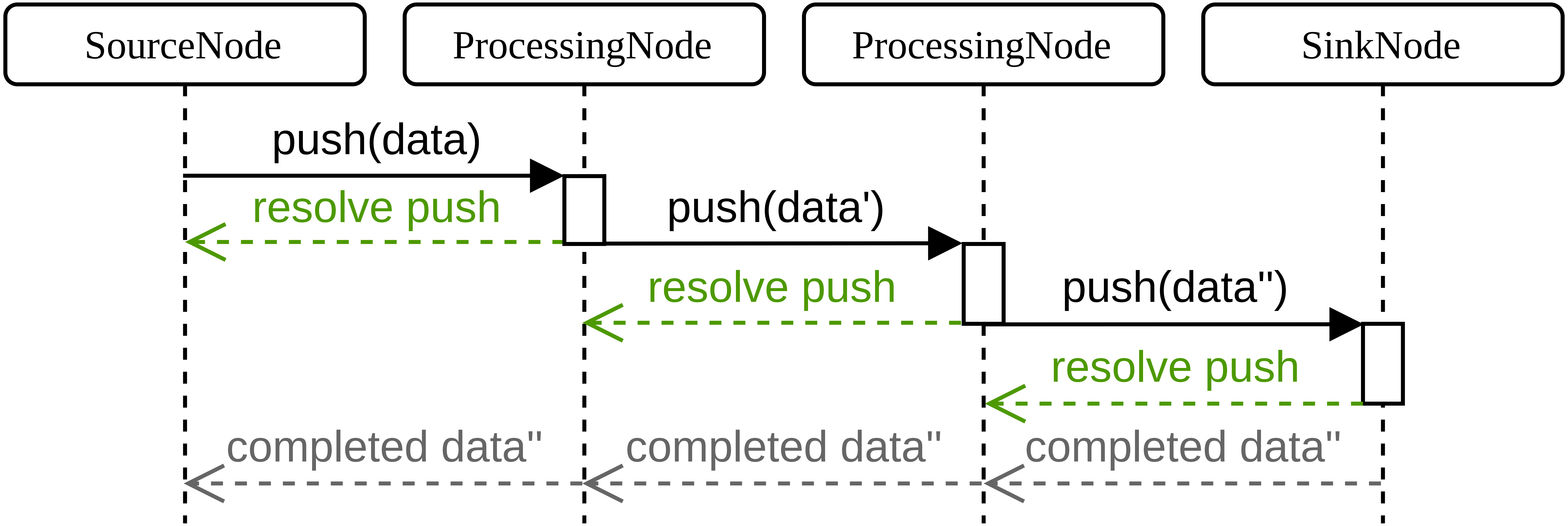
```
uid: "camera",
position: {
  x: 2, y: 5, z: 3
},
projection: ...,
width: 1280,
height: 1024
```

Image

DataFrame ...

OpenHPS

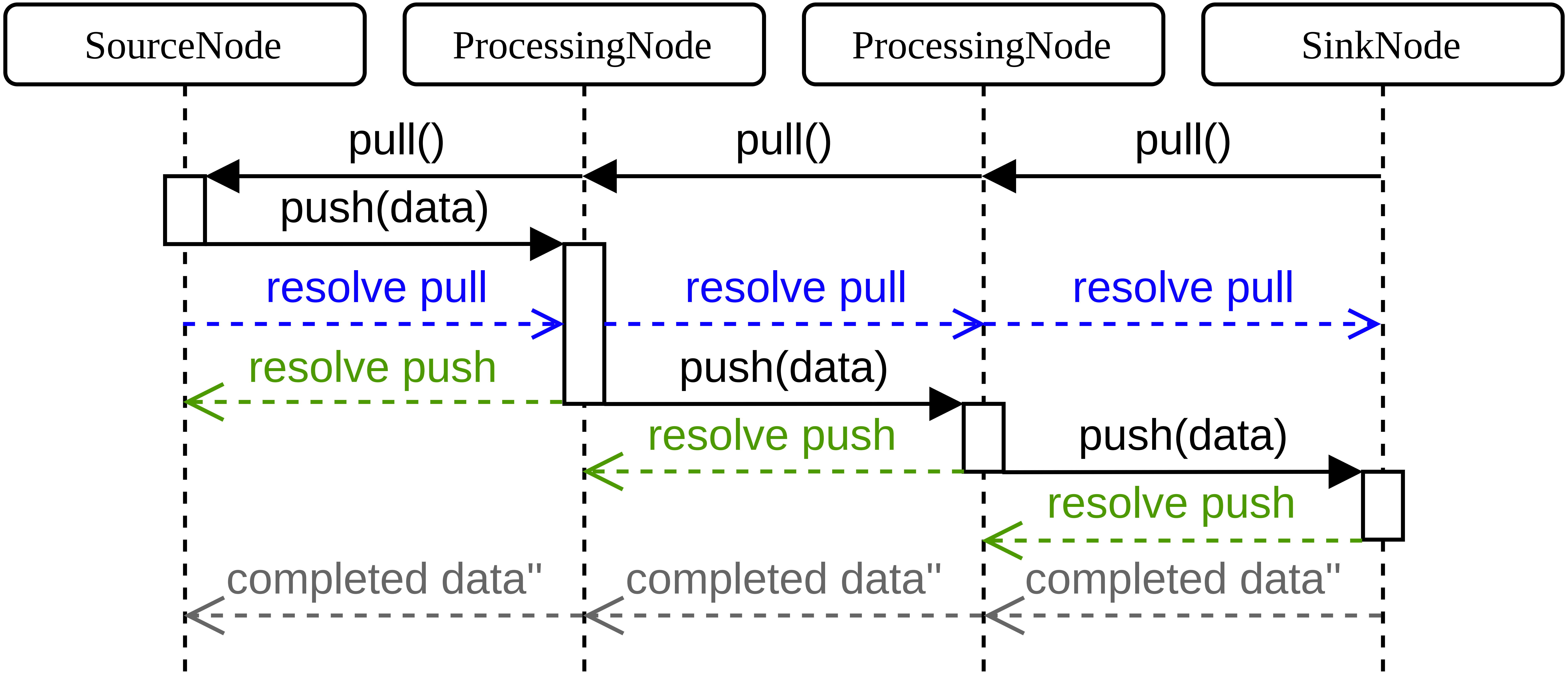
Pushing Data



DataFrame ...

OpenHPS

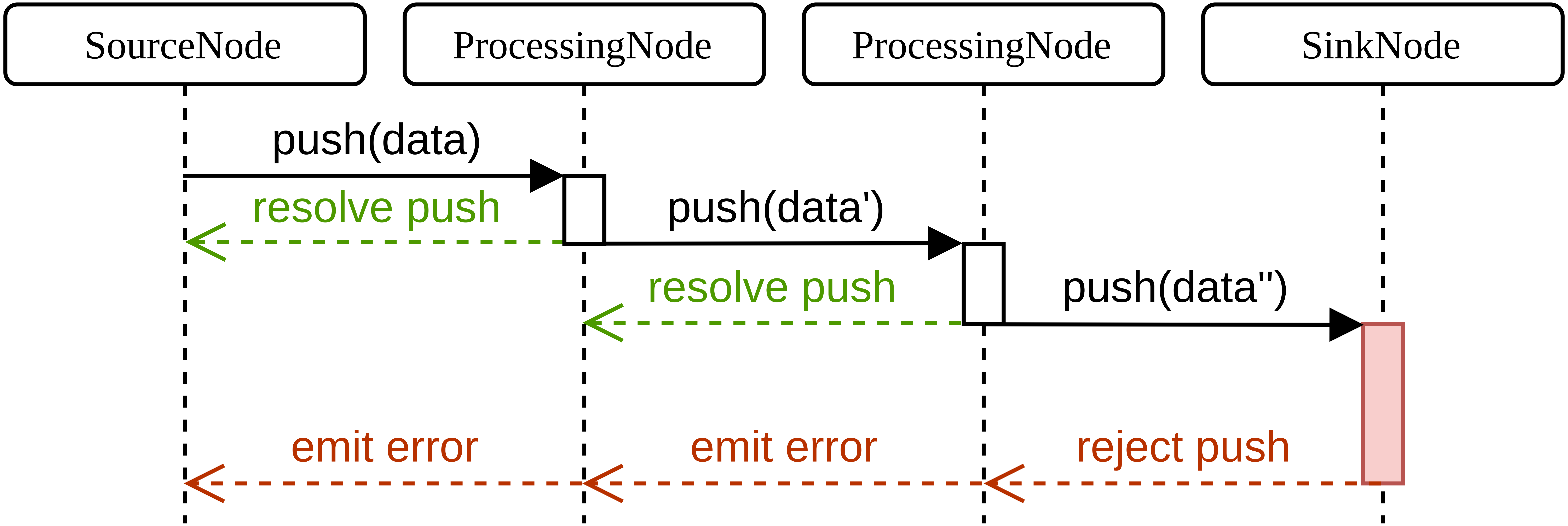
Pulling Data



DataFrame ...

OpenHPS

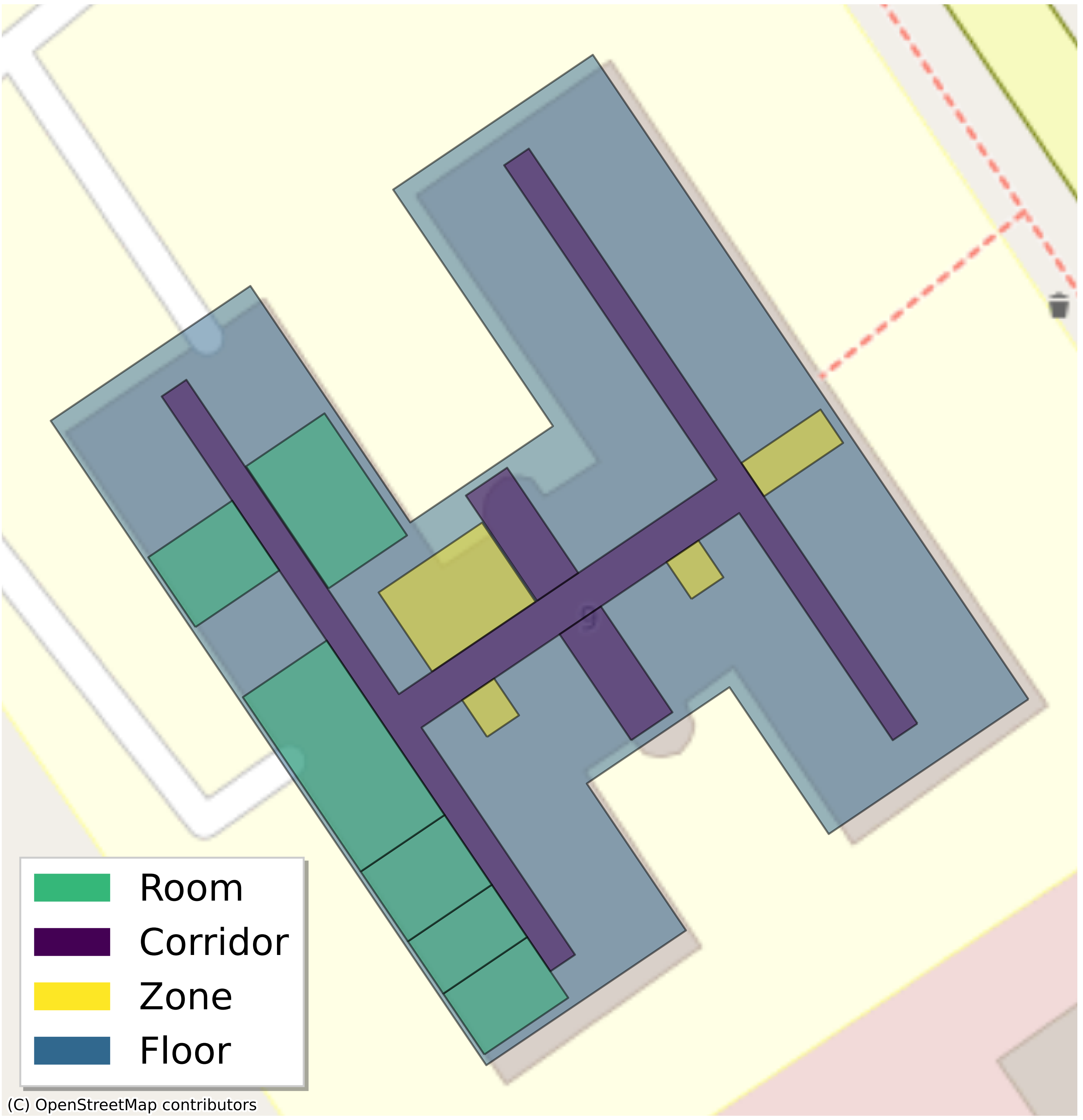
Pushing Error



SymbolicSpace

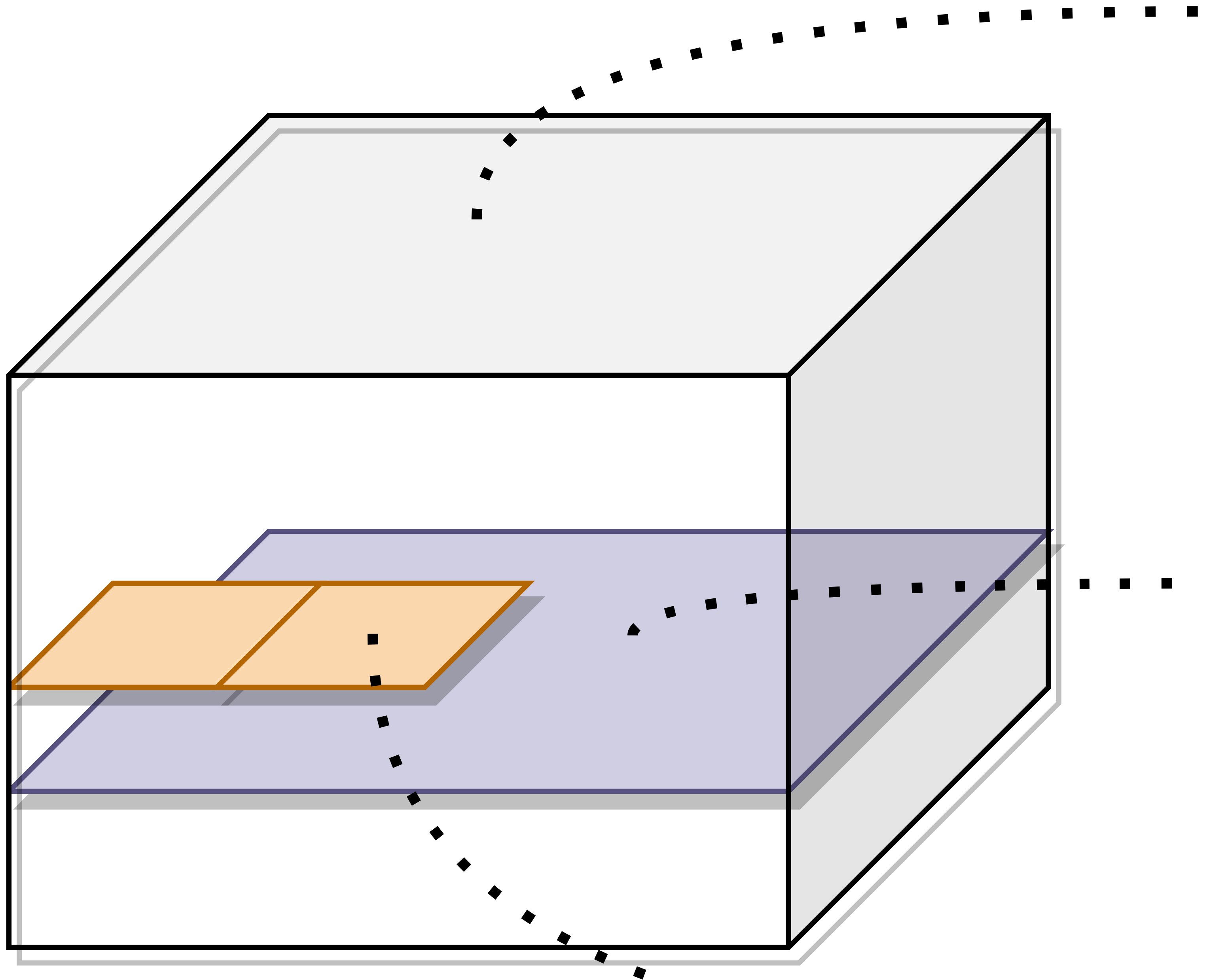
An object that semantically defines a space

- Spatial hierarchy
- Graph connectivity with other spaces
- Geocoding
- GeoJSON compatibility
- Can be used as a location
- Can be extended ...



SymbolicSpace ...

OpenHPS

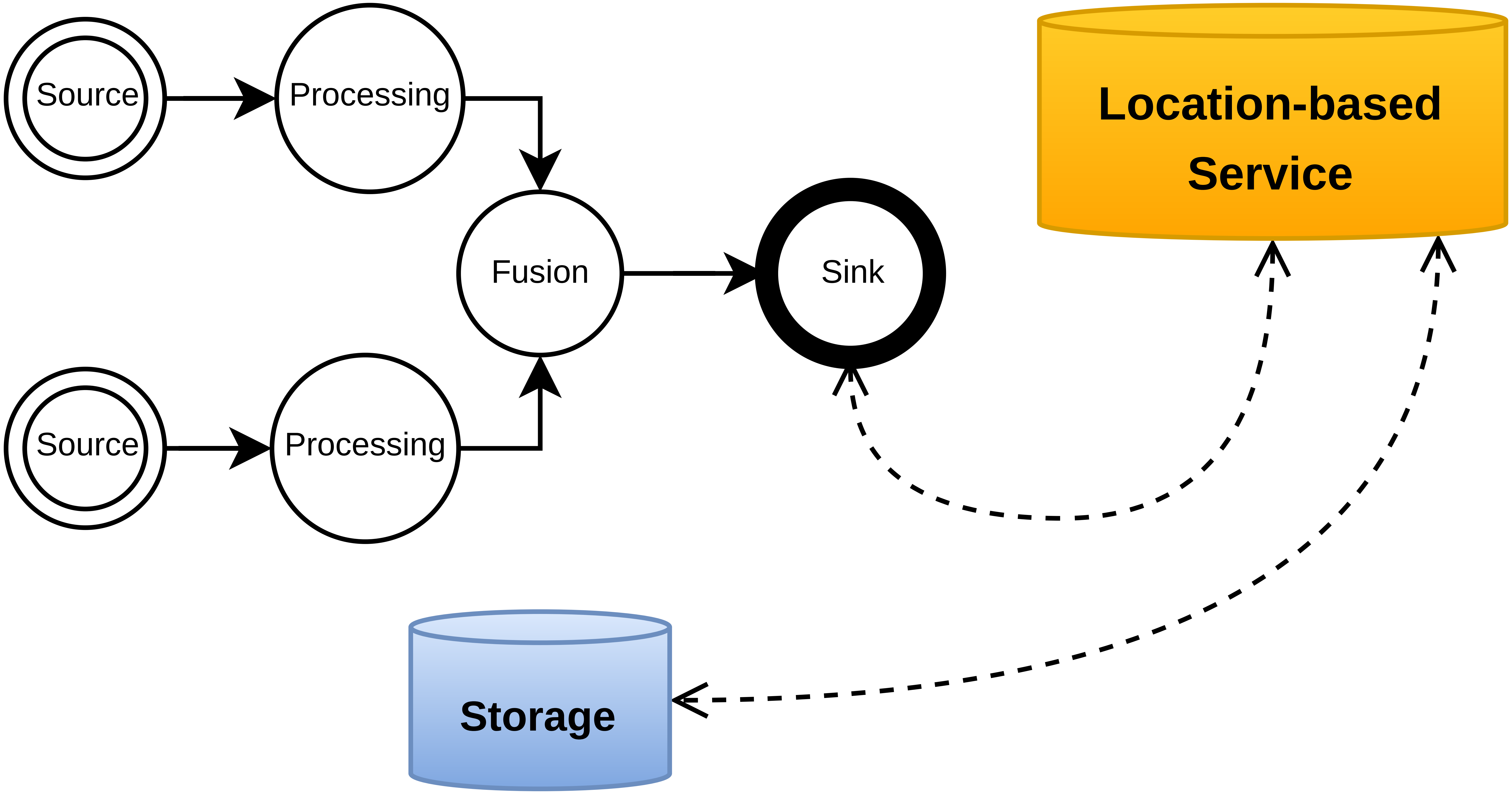


```
const building = new Building("PL9")
    .setBounds({
        topLeft: new GeographicalPosition(
                50.8203,
                4.3922),
        width: 46.275,
        height: 37.27,
        rotation: -34.04
    });

const floor = new Floor("PL9.3")
    .setBuilding(building)
    .setFloorNumber(3);

const office = new Room("PL9.3.58")
    .setFloor(floor)
    .setBounds([
        new Absolute2DPosition(4.75, 31.25),
        new Absolute2DPosition(8.35, 37.02),
    ]);
```

Location-based Service

OpenHPS

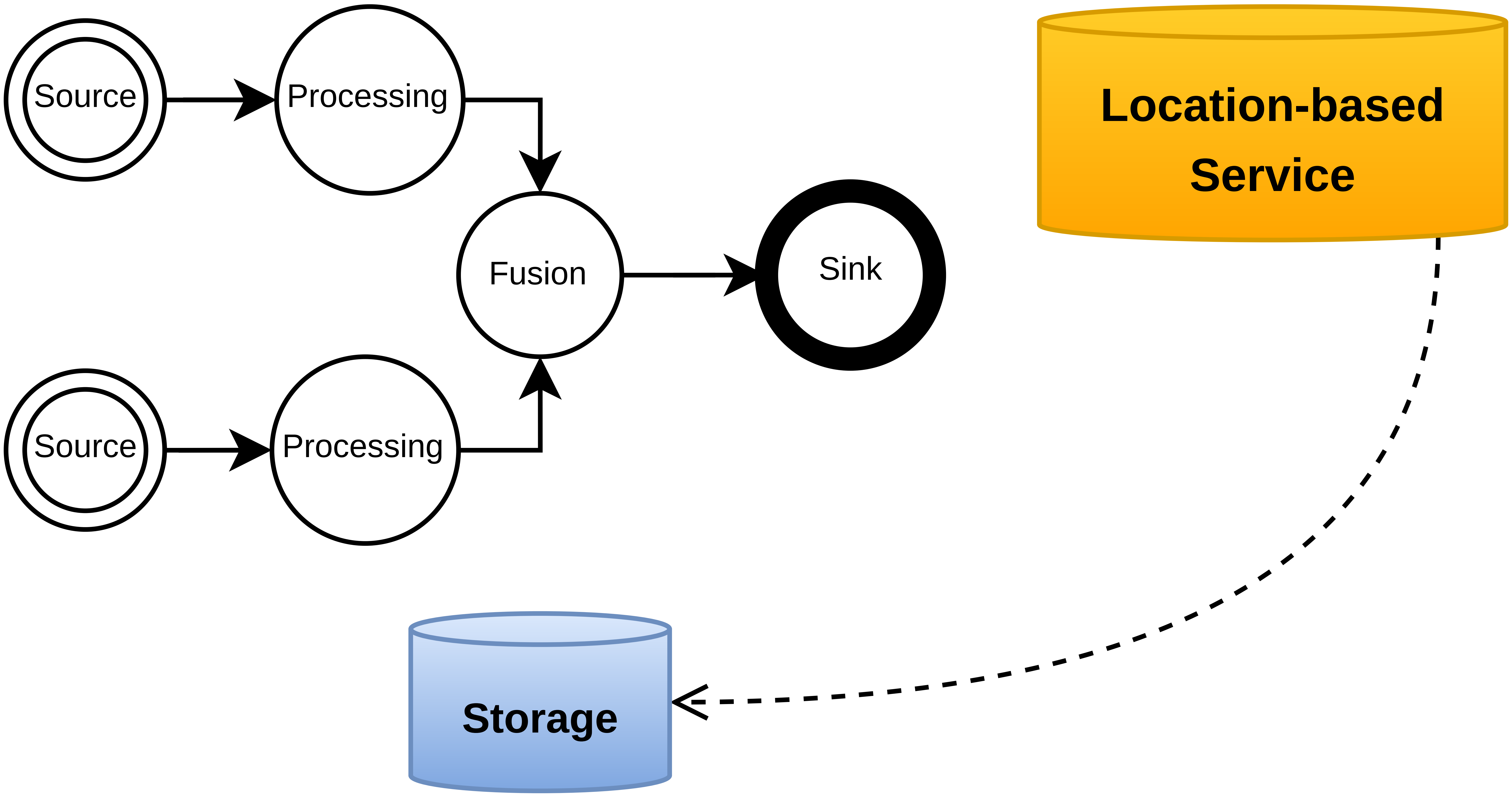
```
getCurrentPosition("me", ...)
```



Location-based Service ...

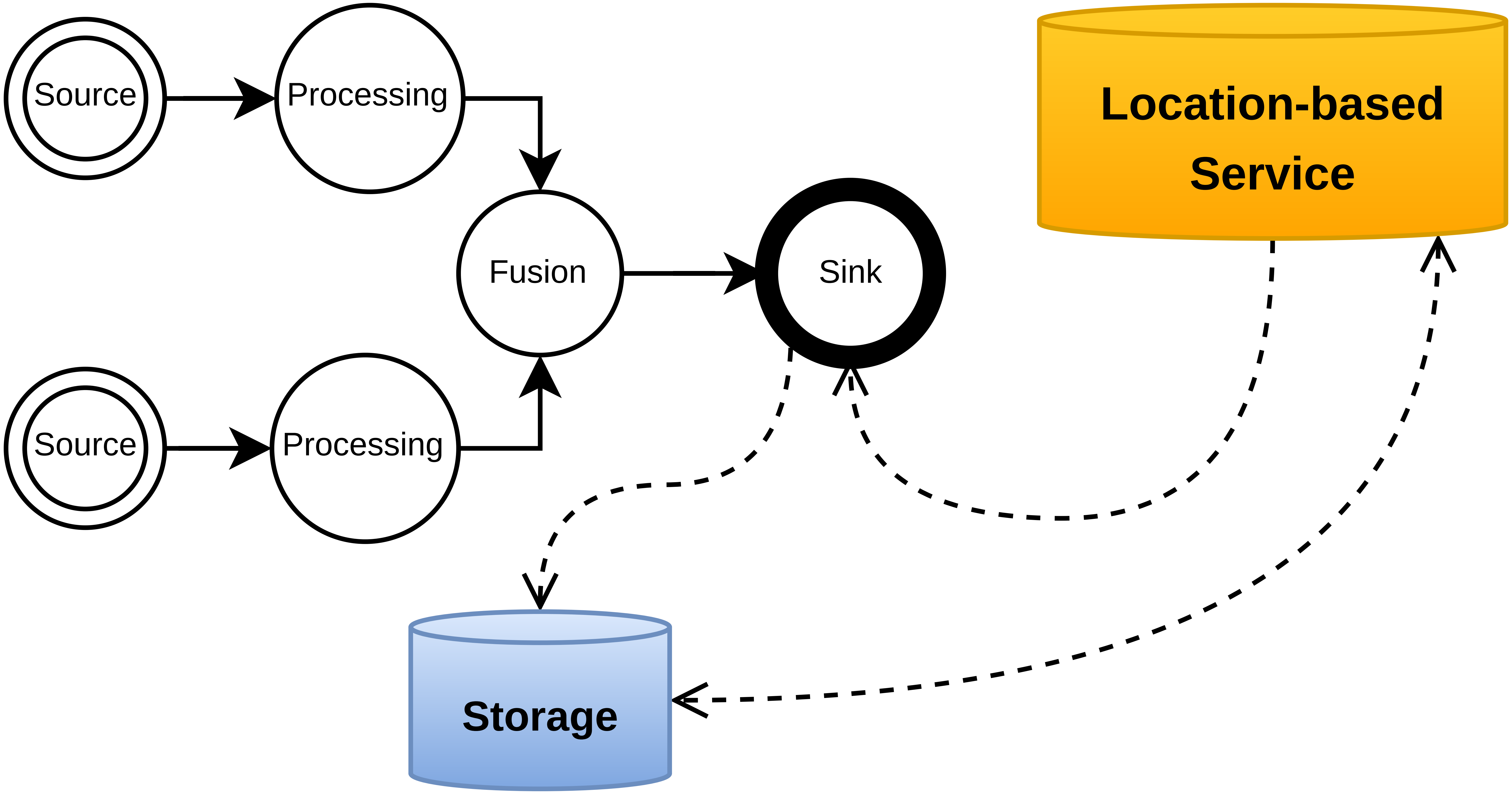
OpenHPS

```
setCurrentPosition("me", ...)
```



Location-based Service ...

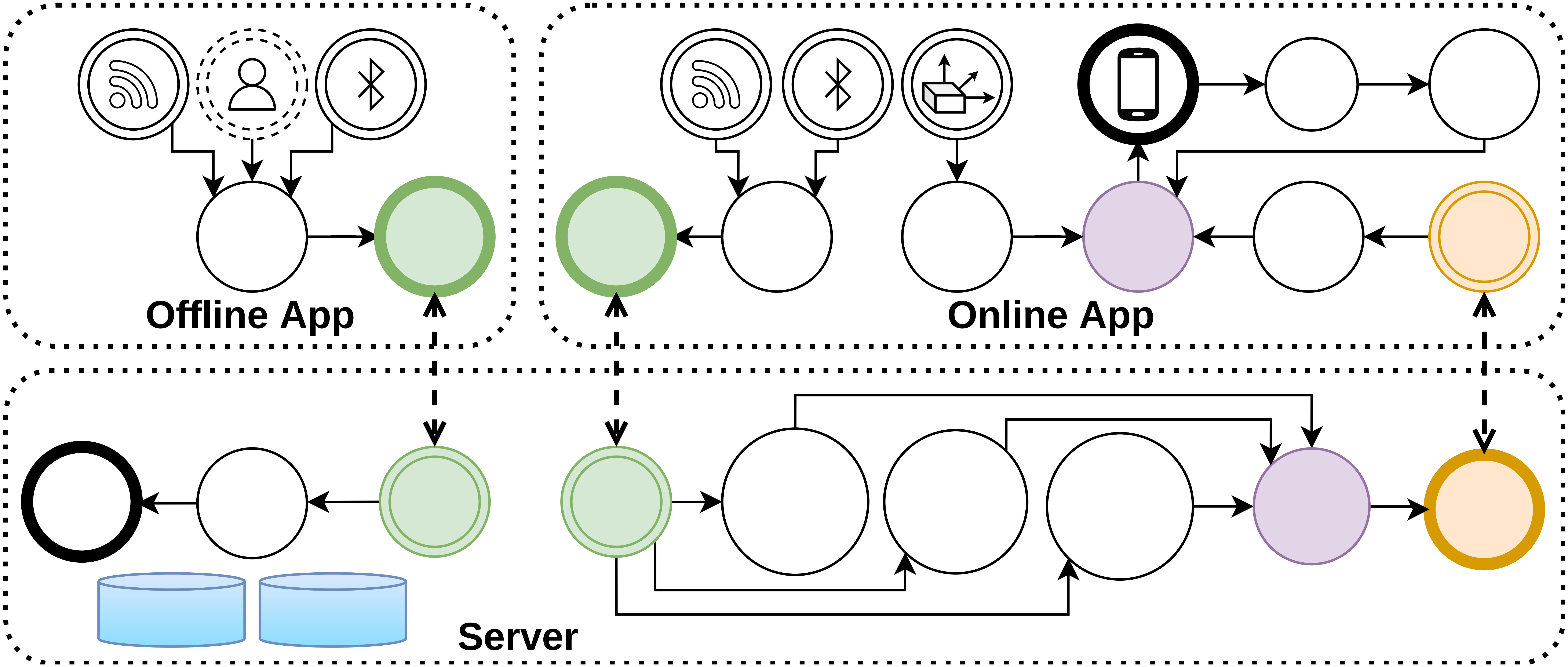
`watchPosition("me", ...)`



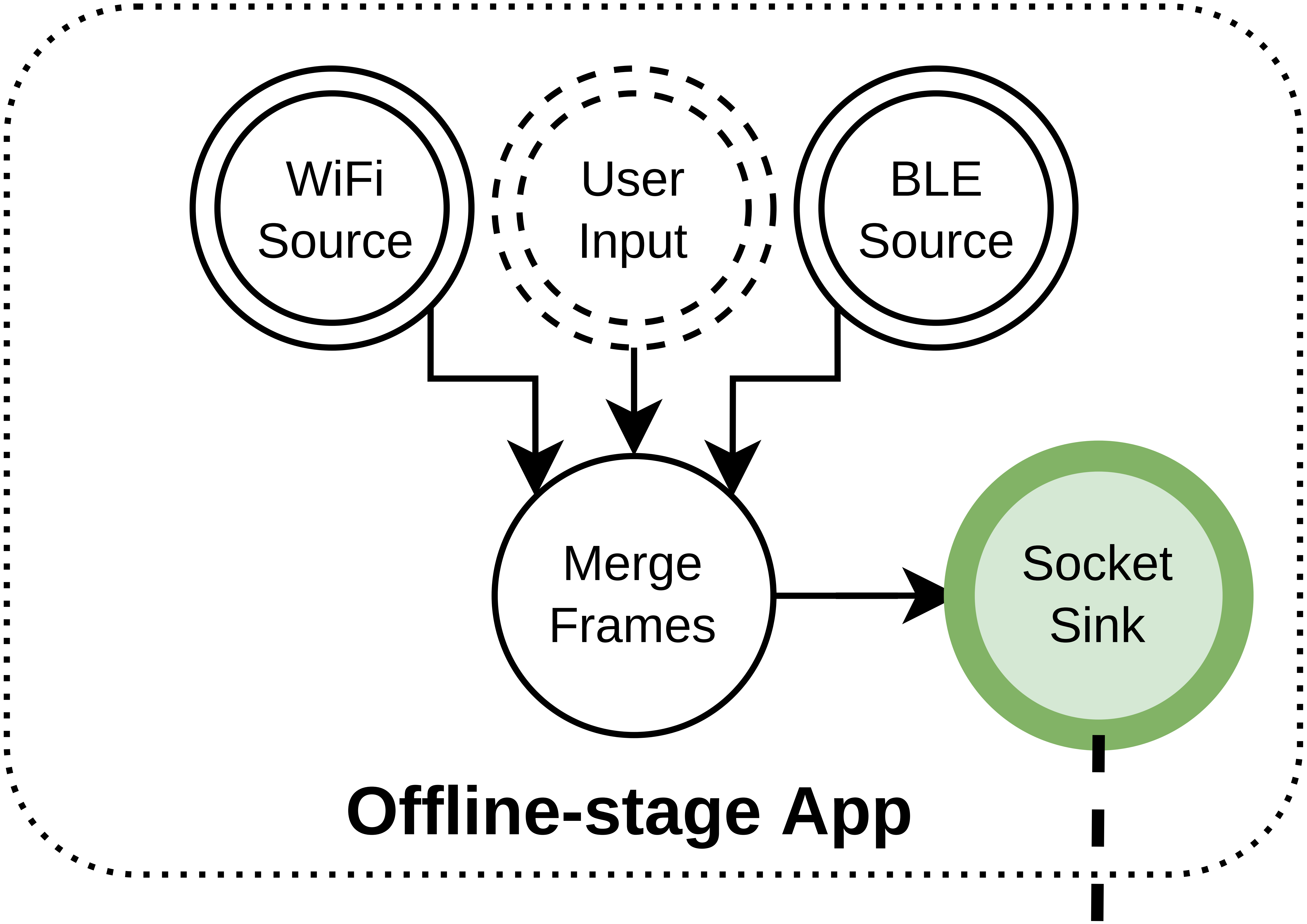
Demonstration

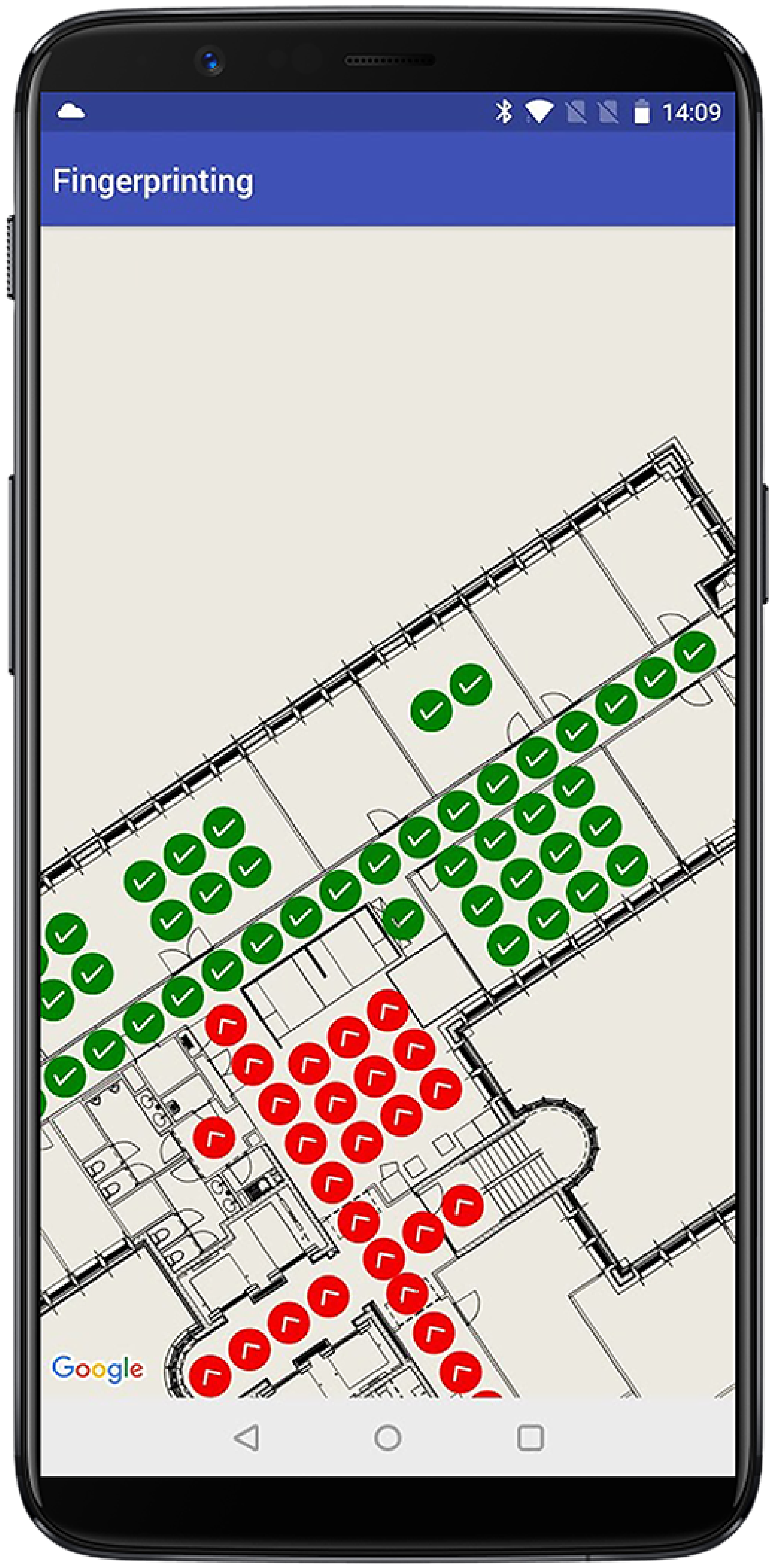
OpenHPS

- Indoor positioning **use case**
- Use **existing techniques**
- Validation of **flexibility** and modularity

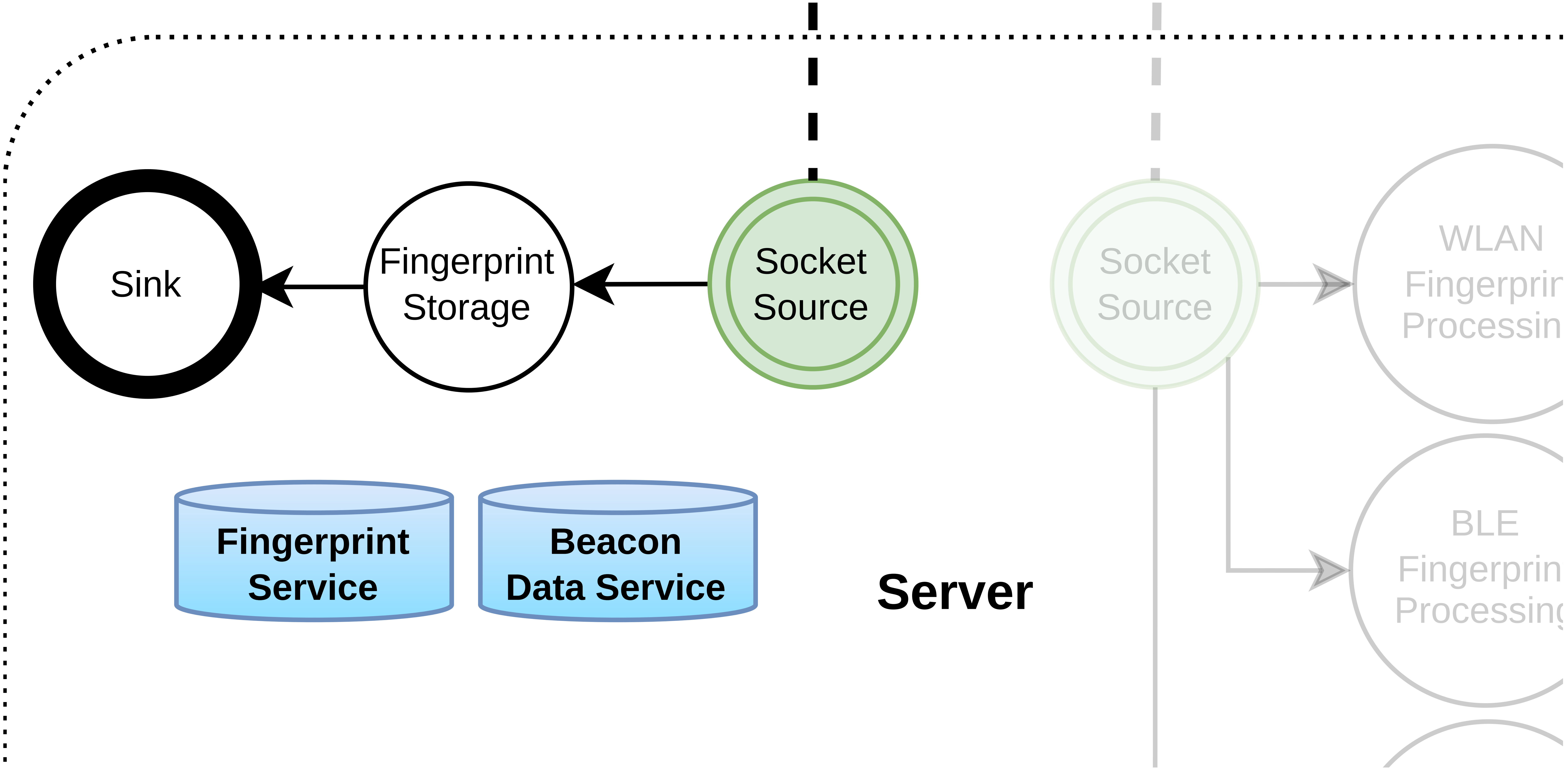


Positioning Model



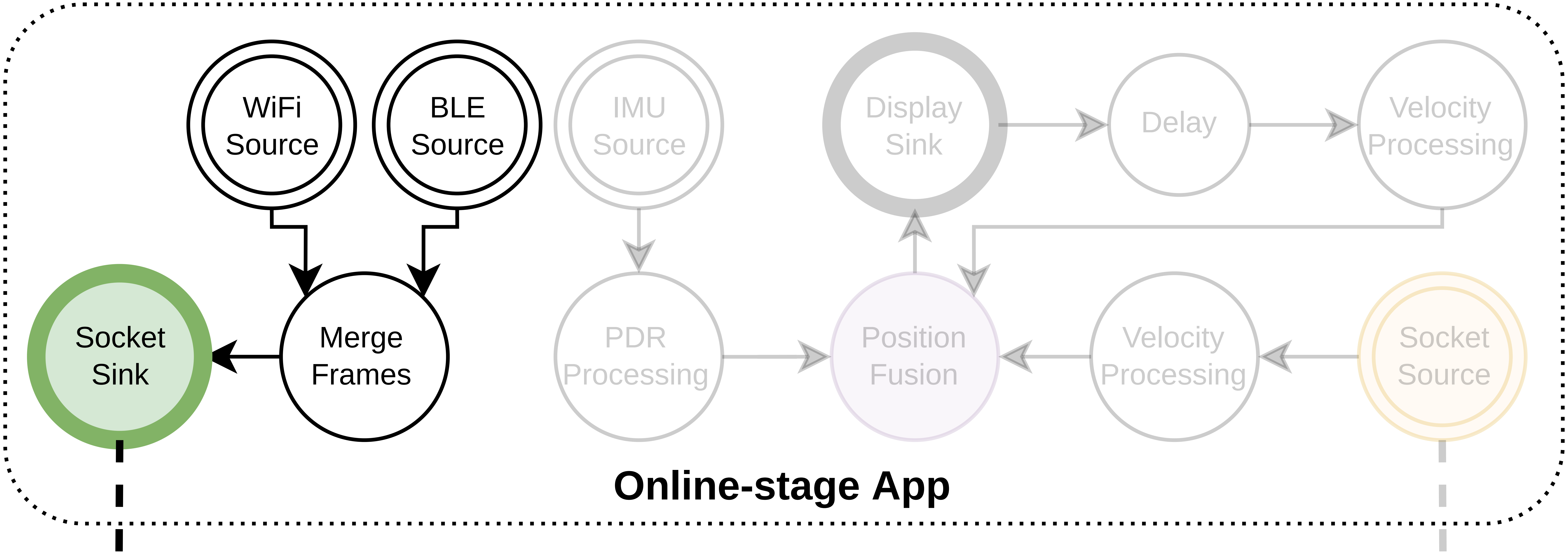


Positioning Model ...



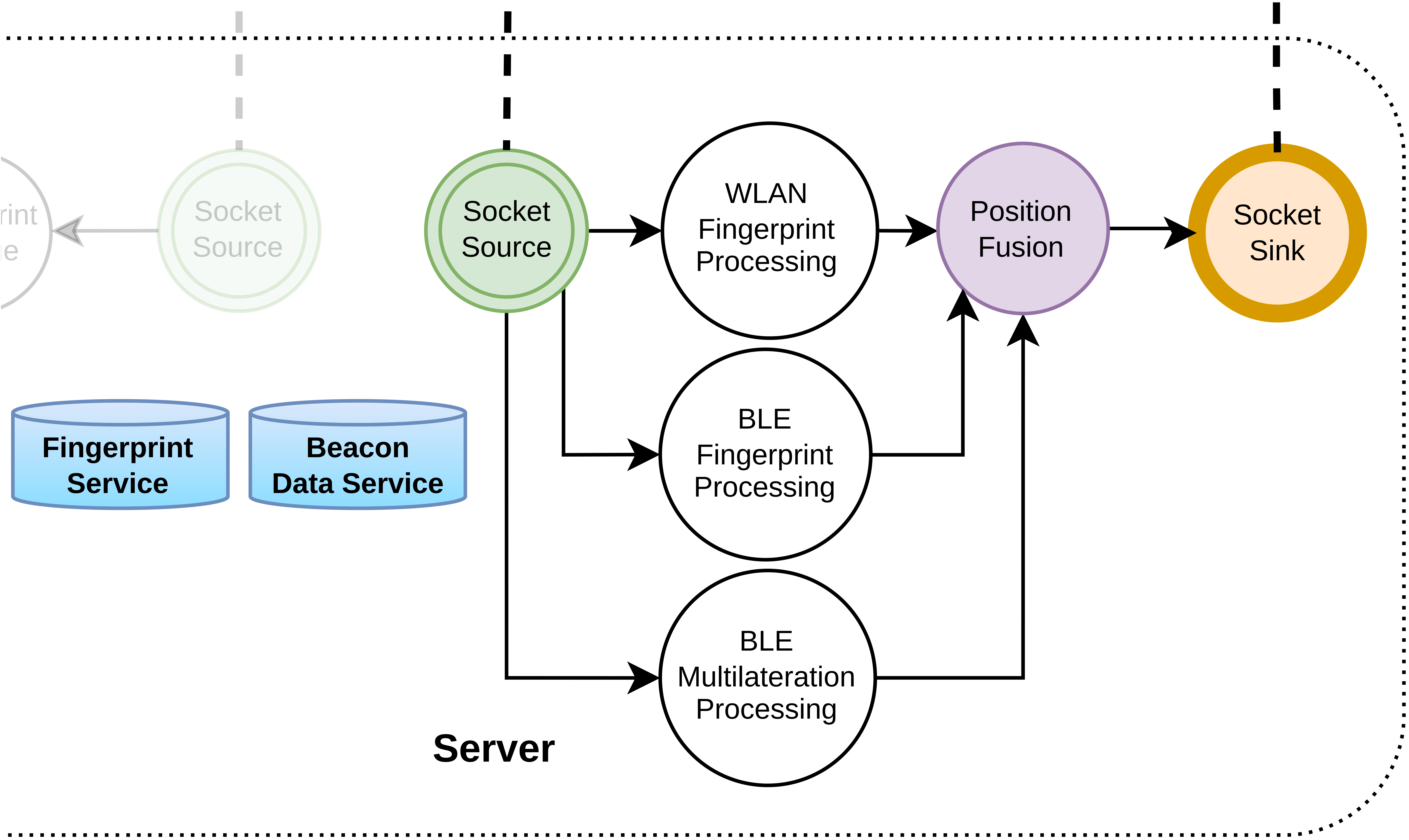
Positioning Model ...

OpenHPS

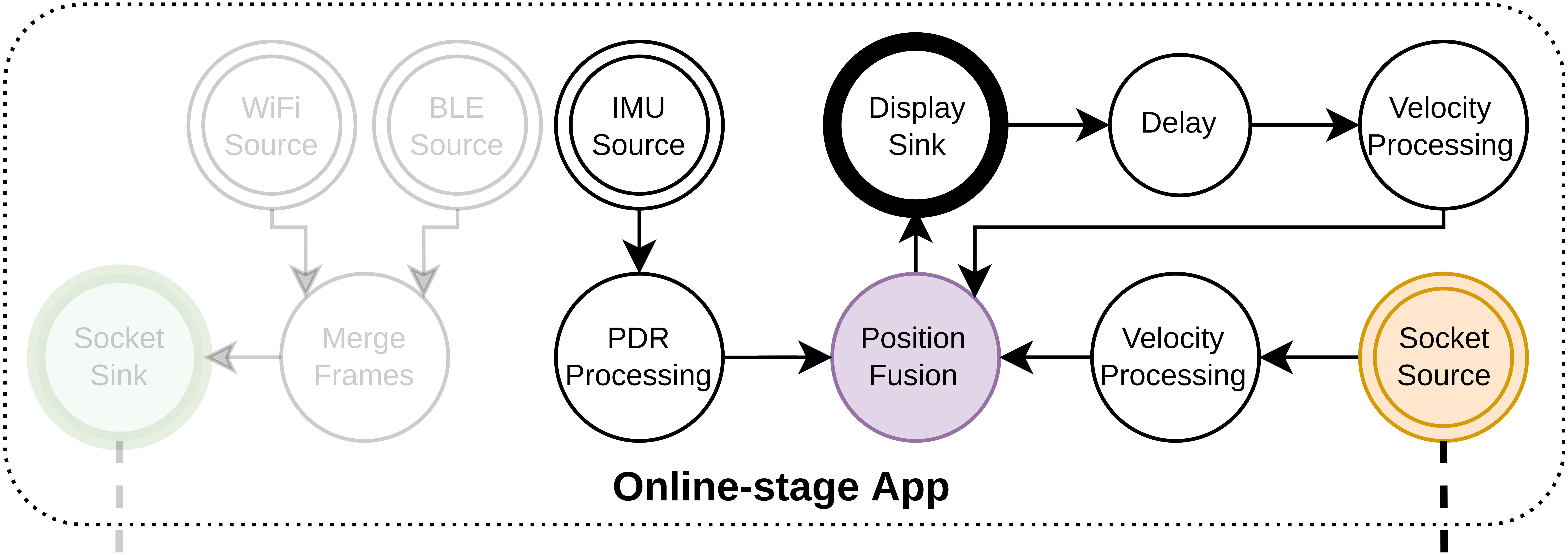


Positioning Model ...

OpenHPS



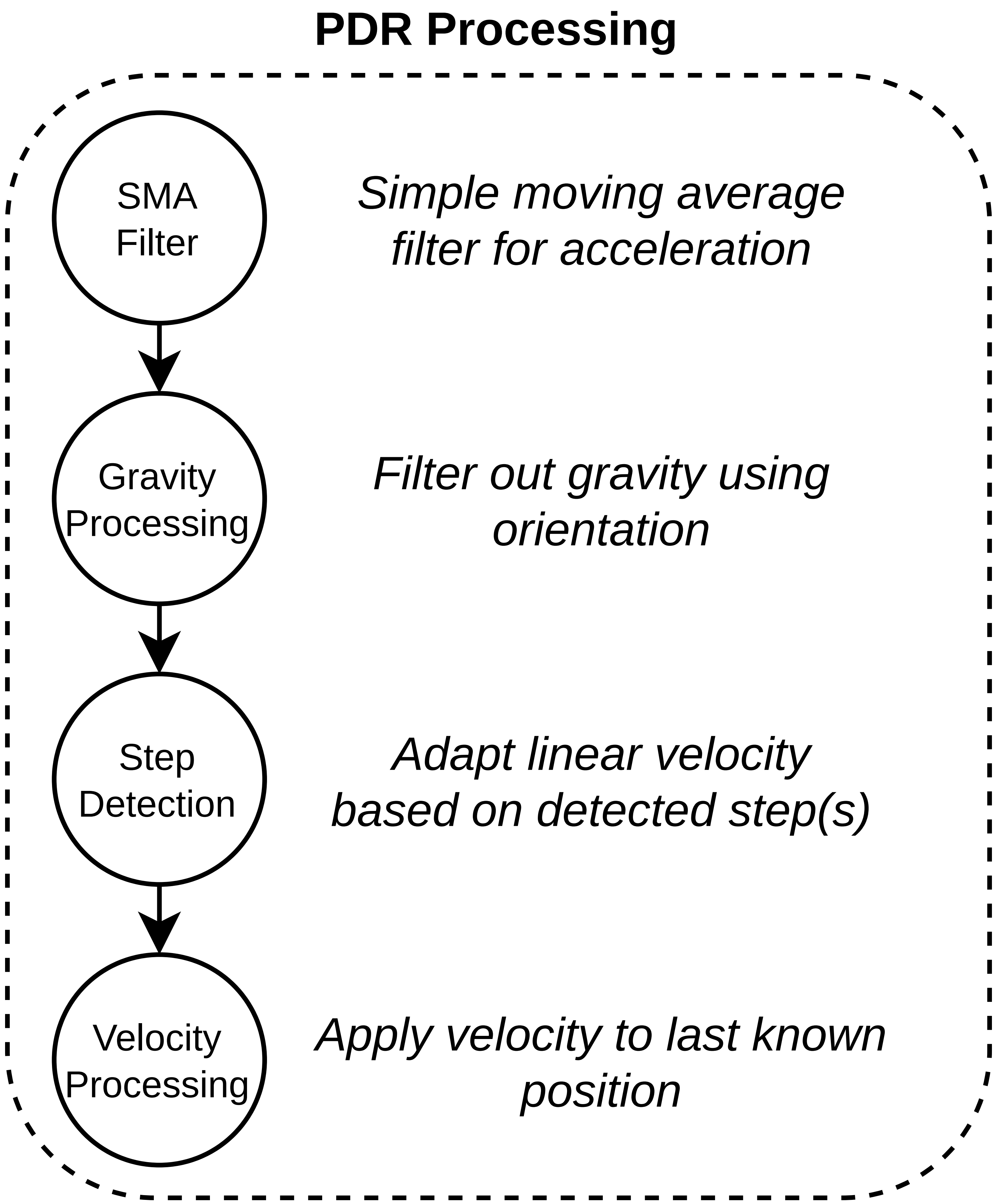
Positioning Model ...



Positioning Model

OpenHPS

Online App


PDR Processing

Simple moving average filter for acceleration

Filter out gravity using orientation

Adapt linear velocity based on detected step(s)

Apply velocity to last known position

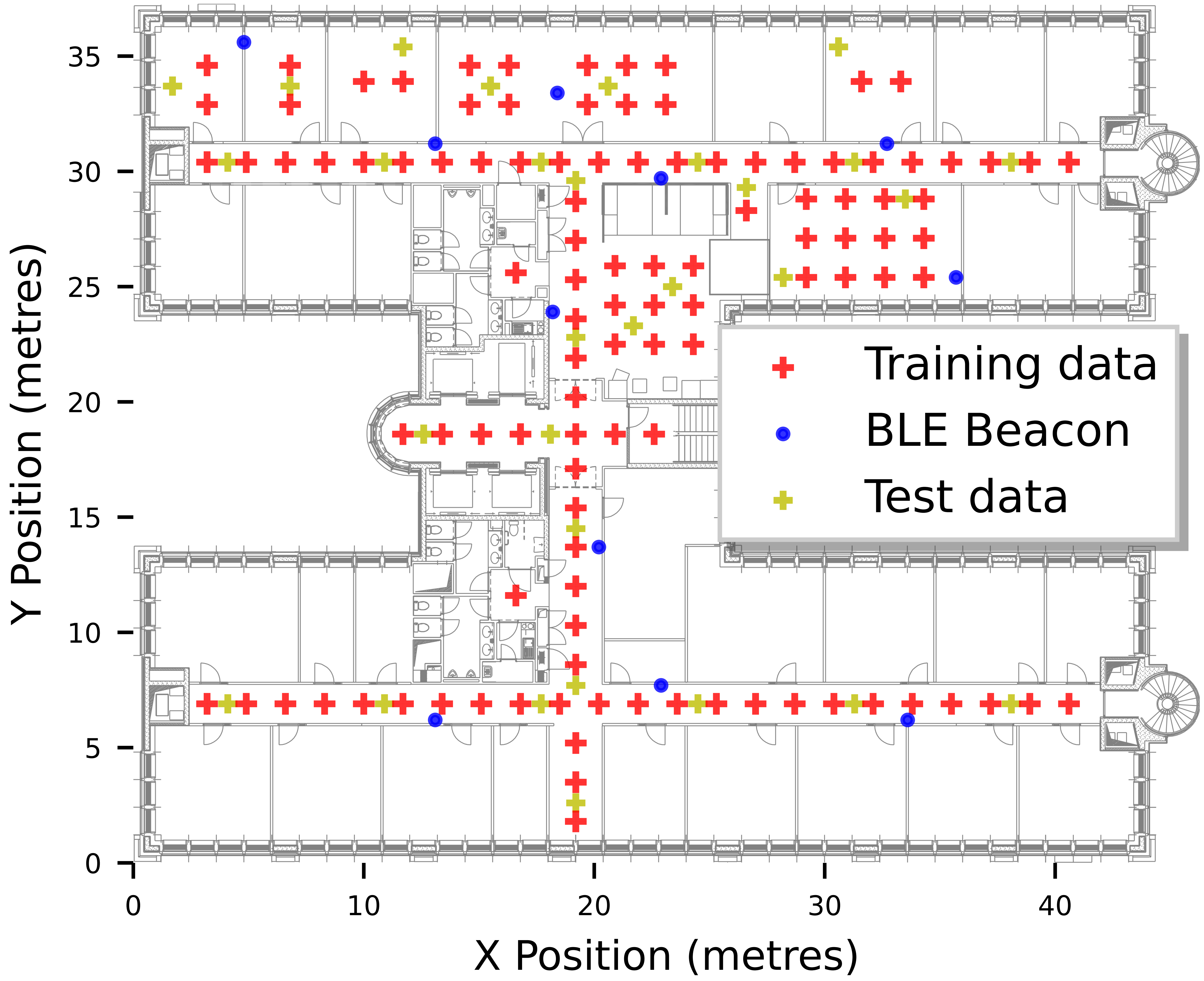
Positioning Model

Online App

```
ModelBuilder.create()
    .addShape(GraphBuilder.create()
        .from(new IMUSourceNode({
            source: new DataObject(phoneUID),
            interval: 20,
            sensors: [
                SensorType.ACCELEROMETER,
                SensorType.ORIENTATION
            ]
        }))
        .via(new SMAFilterNode(
            frame => [frame, "acceleration"],
            { taps: 10 }
        ))
        .via(new GravityProcessingNode({
            method: GravityProcessingMethod.ABSOLUTE_ORIENTATION
        }))
```

Dataset

OpenHPS



Dataset ...

OpenHPS

Total BLE Beacons: 11

Total detected WLAN access points: 220

Total stable WLAN access points: 199

	Training	**Test**
Datapoints	110	30
Total fingerprints	440	120
Duration (per orientation)	20s	20s
Avg. WLAN Scans (per fingerprint)	6	6
Avg. BLE Advertisements (per fingerprint)	16	15

Validation Results

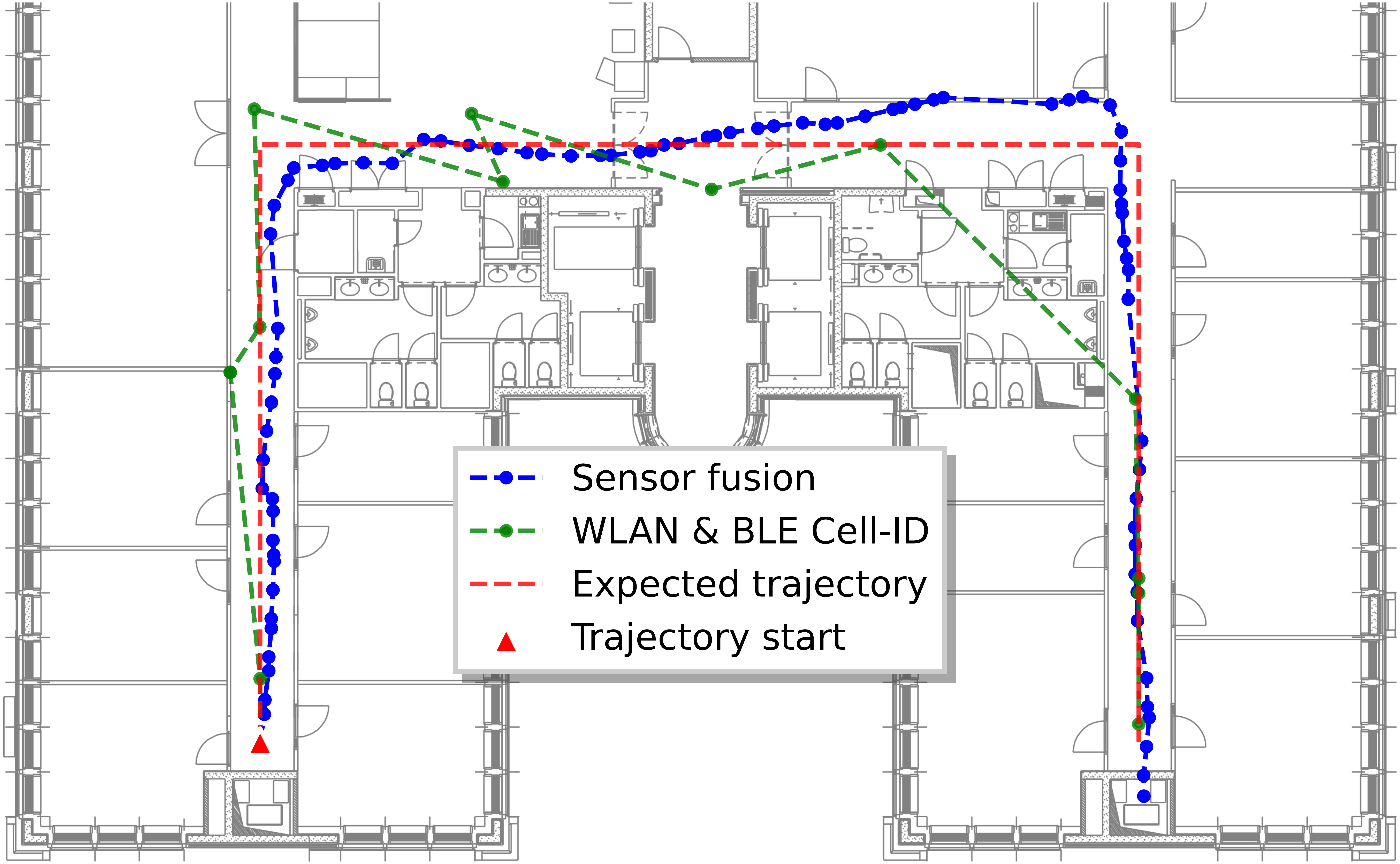
OpenHPS

Static Positioning

	WLAN **fingerprinting**	**BLE** **fingerprinting**	**BLE** **multilateration**	**Fusion**
failed points	0	6	12	0
average error	1.23 m	3.23 m	4.92 m	1.37 m
minimum error	0.01 m	0.17 m	0.74 m	0.01 m
maximum error	4.77 m	15.39 m	19.26 m	9.75 m
hit rate	95.82 %	80.83 %	52.50 %	**96.67 %**

Validation Results ...

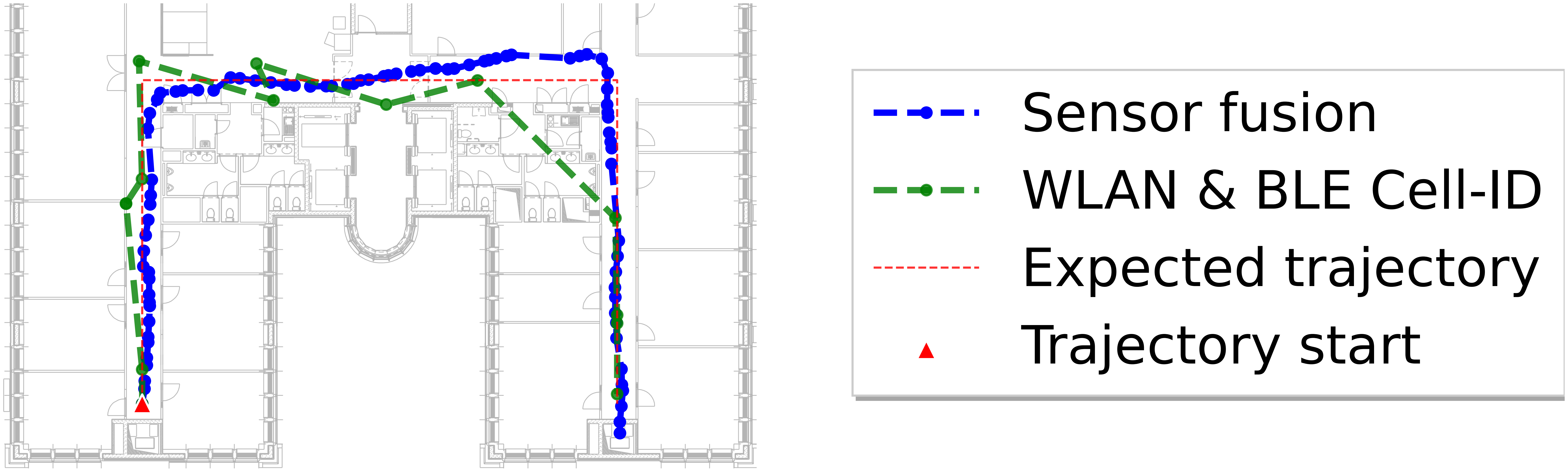
Trajectories



Validation Results ...

Trajectories

	WLAN + BLE	WLAN + BLE + IMU
average error	3.28 m	1.26 m
maximum error	9.60 m	3.10 m
average update frequency	3.04 s	0.52 s



Contributions and Conclusions

OpenHPS

- OpenHPS: **open source** framework for hybrid positioning
  - Aimed towards **developers** and **researchers**
- **Abstractions** such as location-based services and spaces
- Validation of an indoor positioning use case
- Configurable and interchangeable **nodes** and **services**
- **Public dataset** with multiple orientations

Visit https://openhps.org for additional resources, documentation, source code and more!