


Indoor Positioning Using the OpenHPS Framework

Maxim Van de Wynckel, Beat Signer

*Web & Information Systems Engineering Lab
Vrije Universiteit Brussel*

An Open Source Hybrid Positioning System





[DOCS](#)
[BLOG](#)
[GITHUB](#)

Documentation

- [Introduction](#)
- [Installation](#)
- [Modules](#)

Basic Concepts

- [Data Object](#)
- [Data Frame](#)

- Creating data frames
- Creating a custom data frame

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.

VideoDataFrame

uid	timestamp
source	
CameraObject	
uid: "camera", position: { x: 2, y: 5, z: 3 }, projection:, width: 1280, height: 1024	
Image	

DataObject
Detected object
DataObject
Detected object
DataObject
Detected object

IMUDataFrame

uid	timestamp
source	
DataObject	
uid: "imusensor", position: { x: 0, y: 0, linearVelocity: { x: 1, y: 0 } }	
Acceleration	
Sensor Frequency	

No additional objects

RFDDataFrame

uid	timestamp
source	
RFRceiverObject	
uid: "wifiscanner", relativePositions: [{ obj: "AP1", distance: 5 }, { obj: "AP2", distance: 8 }]	

AP1 DataObject
uid: "AP1",
position: {
 x: 0, y:
}
 AP2 DataObject
uid: "AP2",
position: {
 x: 15, y: 3
}

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;
}
```

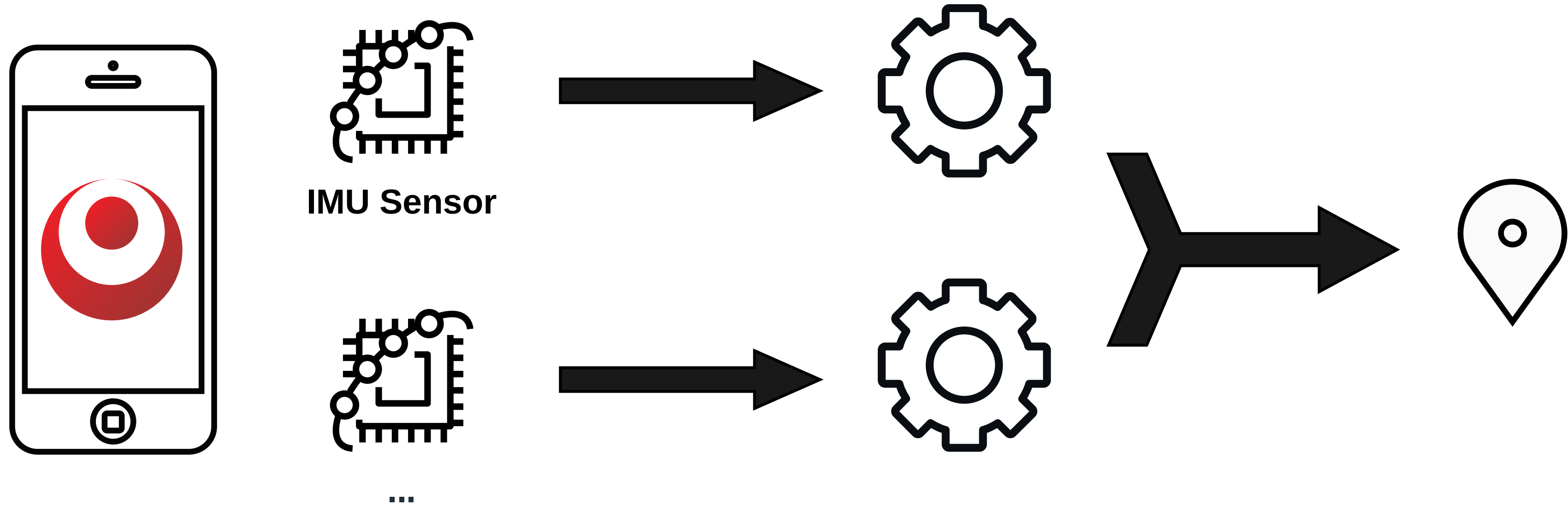
What is OpenHPS?



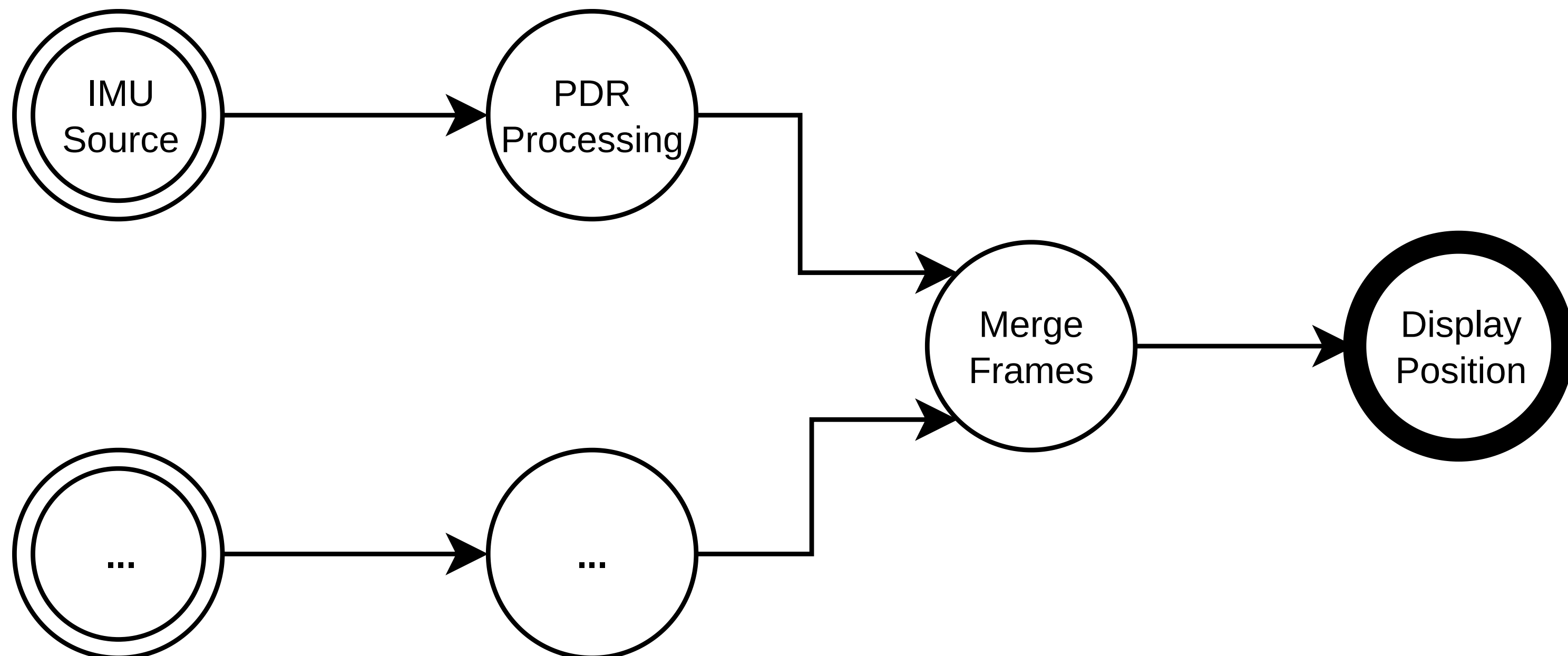
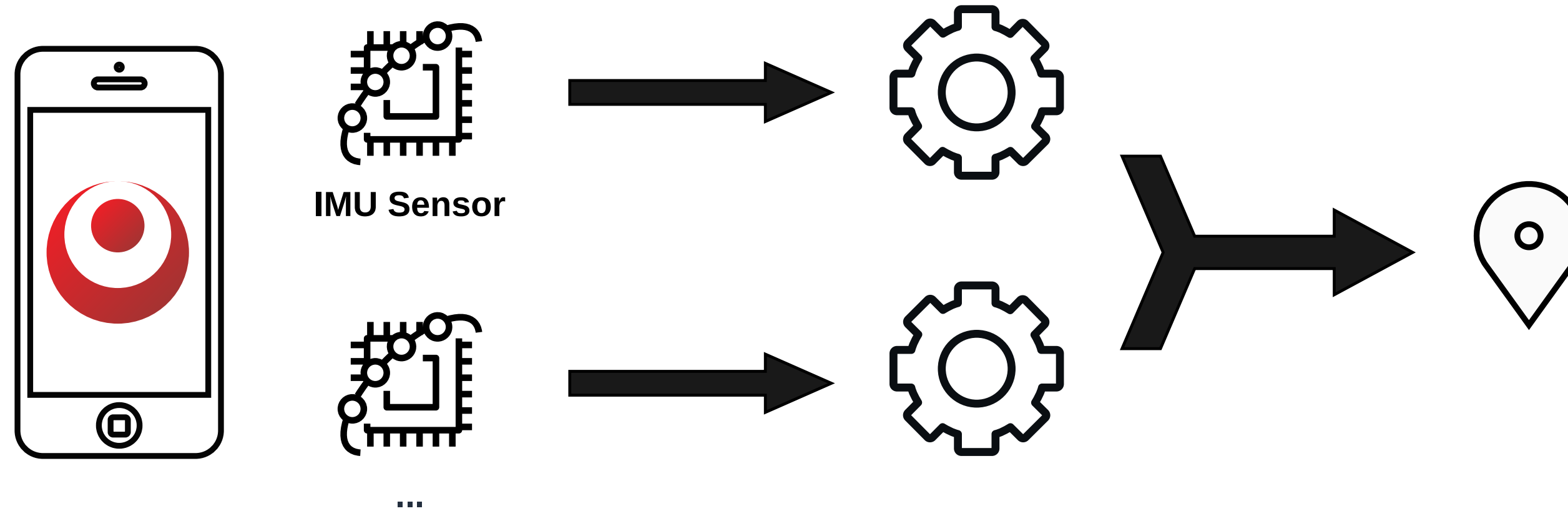
An Open Source Hybrid Positioning System

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

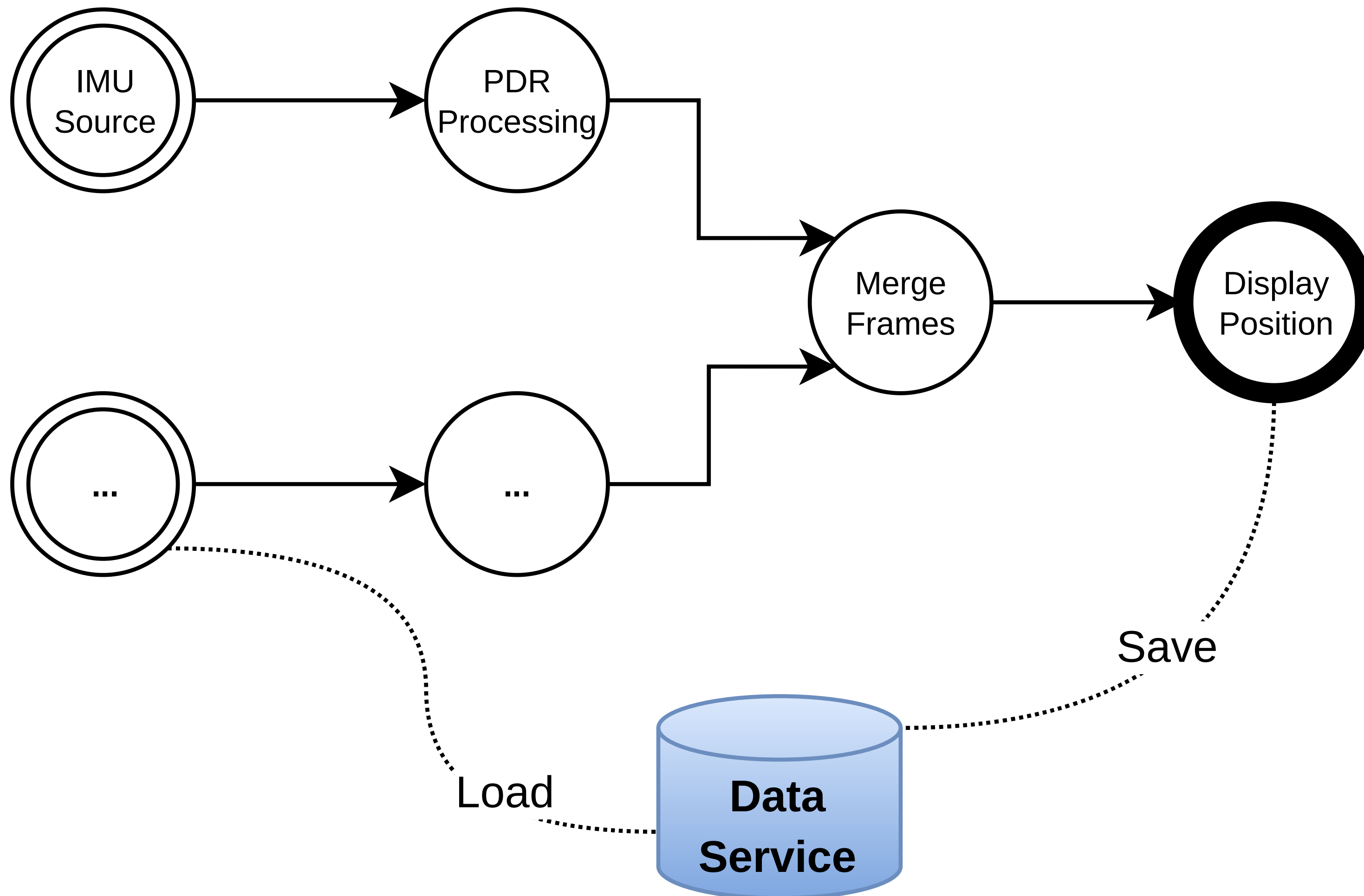
Process Network Design



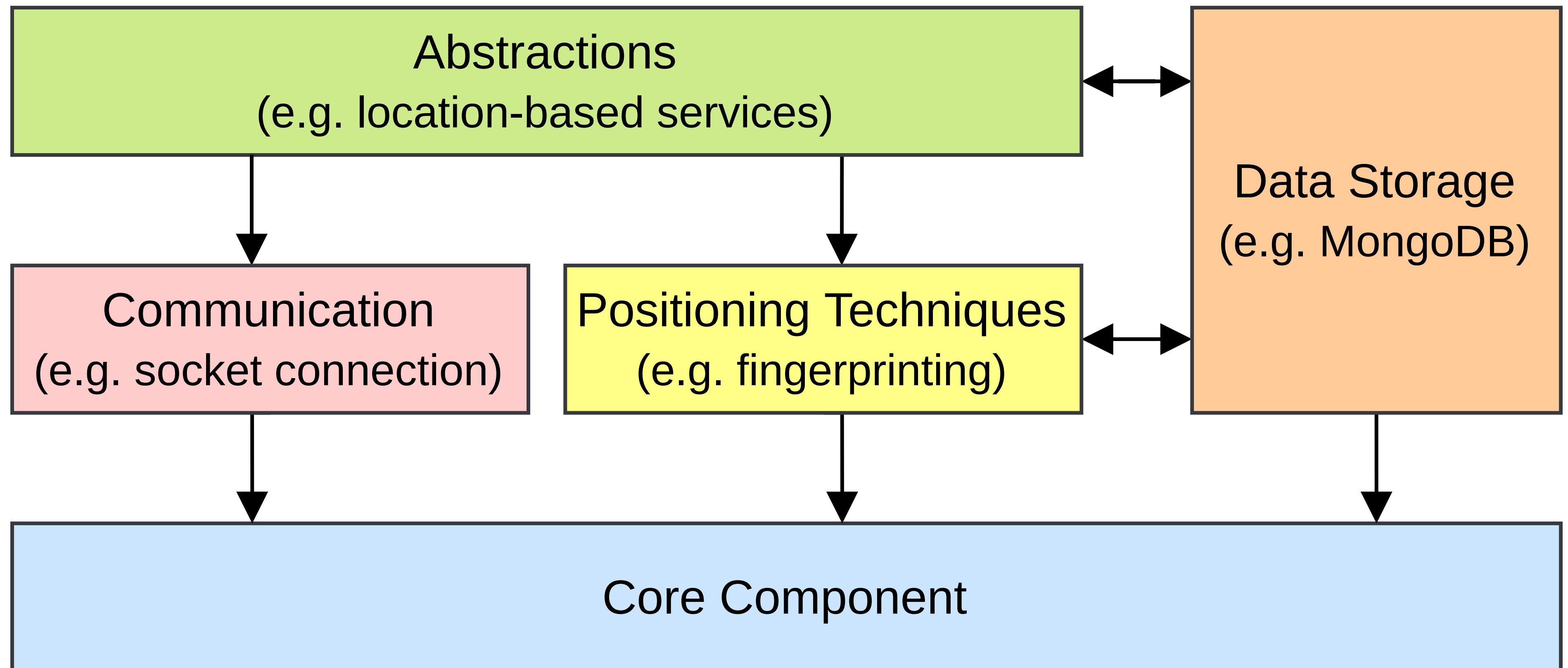
Process Network Design ...



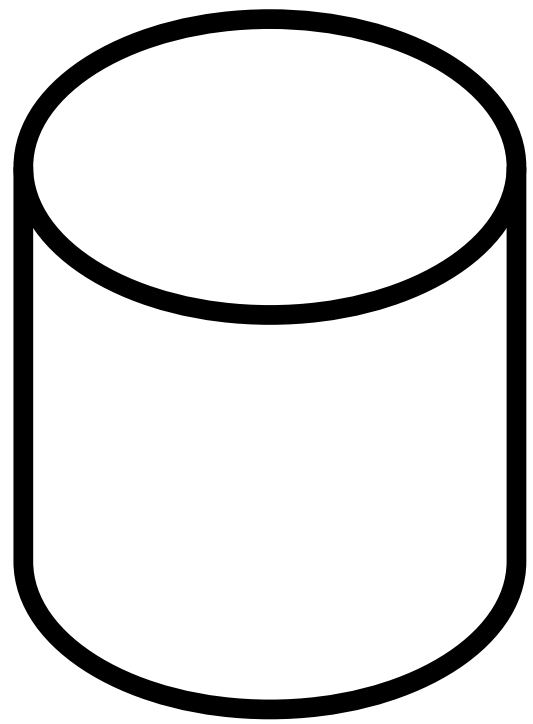
Process Network Design ...



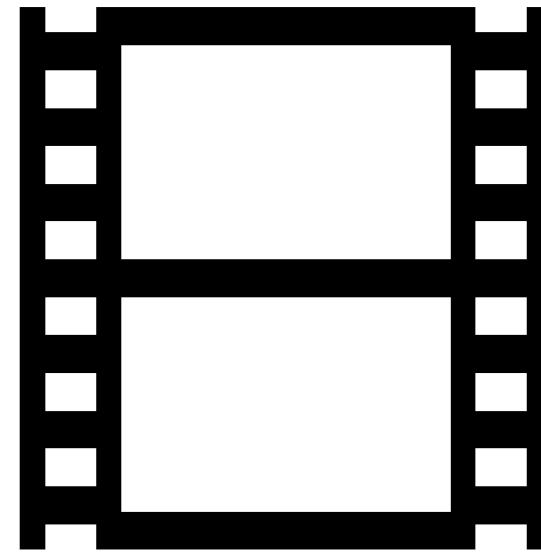
Modularity



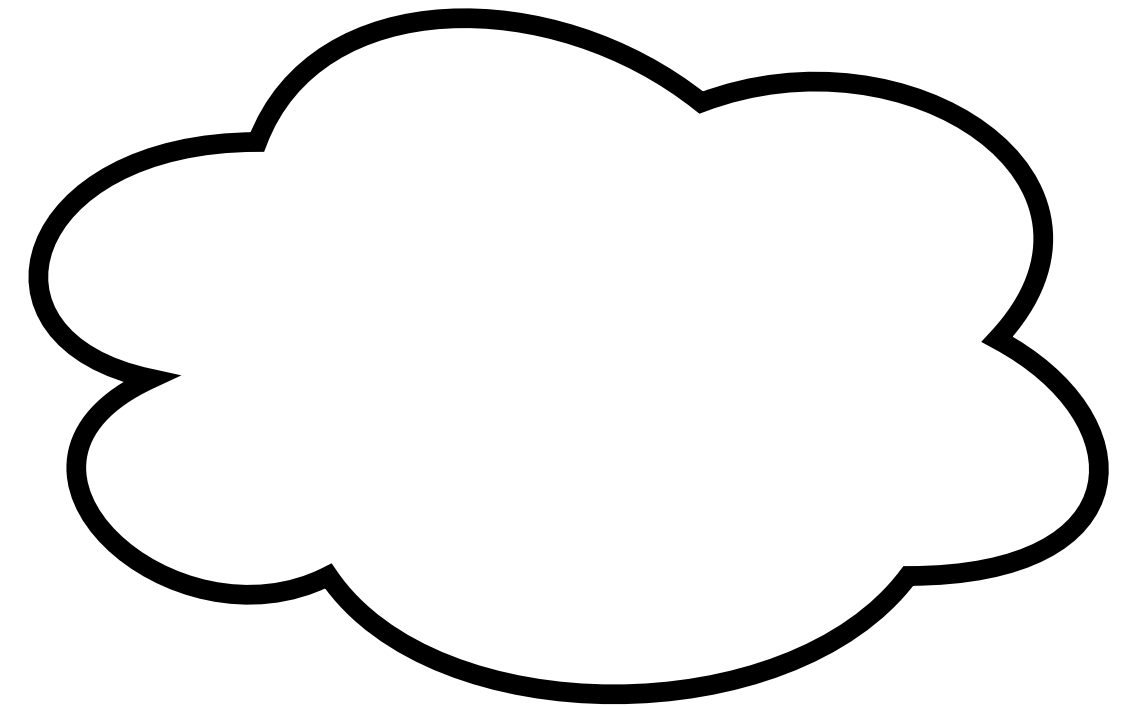
Data Processing



Knowledge

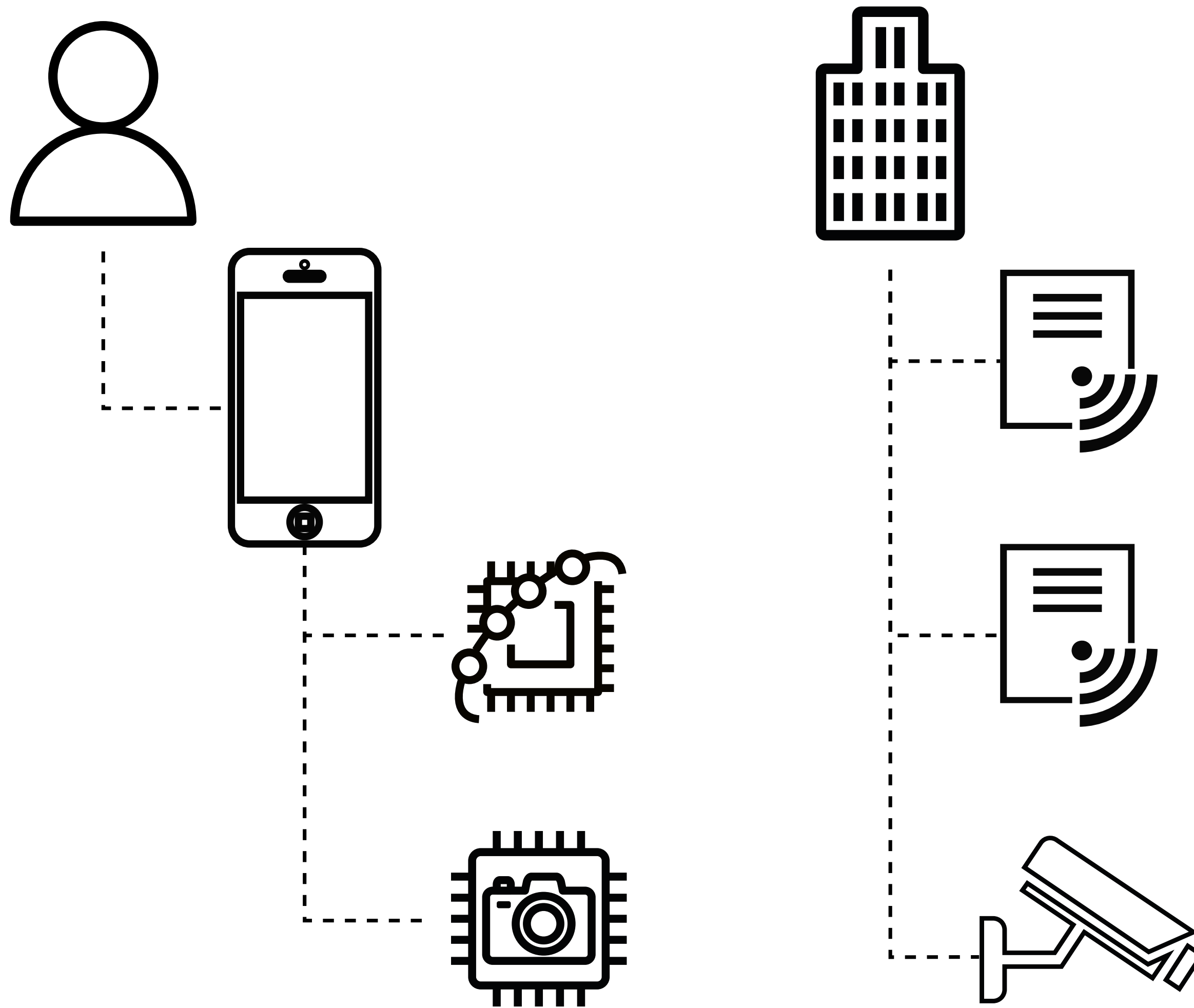


Raw Data



Processed Data

DataObject



Absolute and Relative Positions

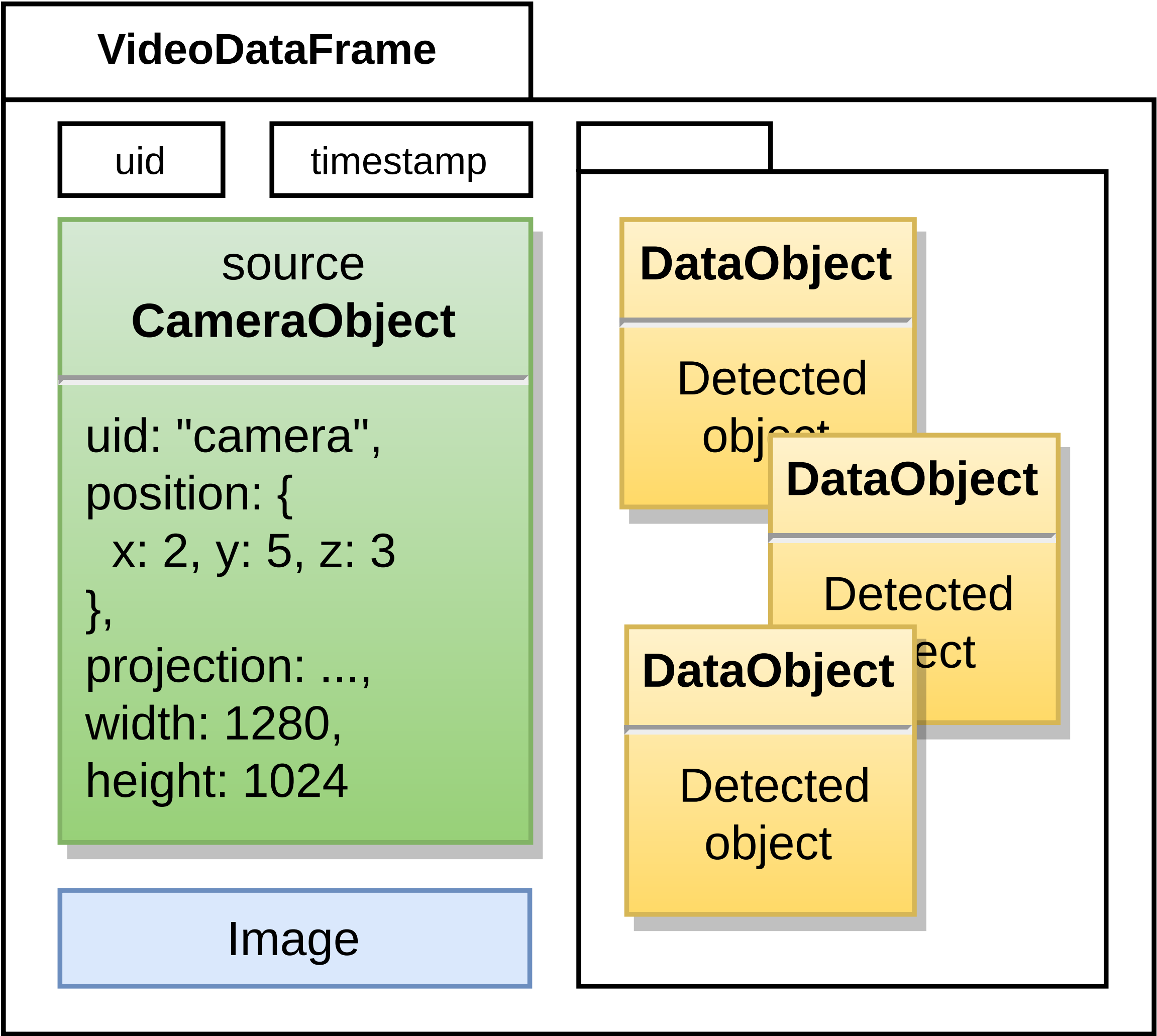
Absolute

- ▶ 2D, 3D, Geographical, ...

Relative

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

DataFrame



SymbolicSpace

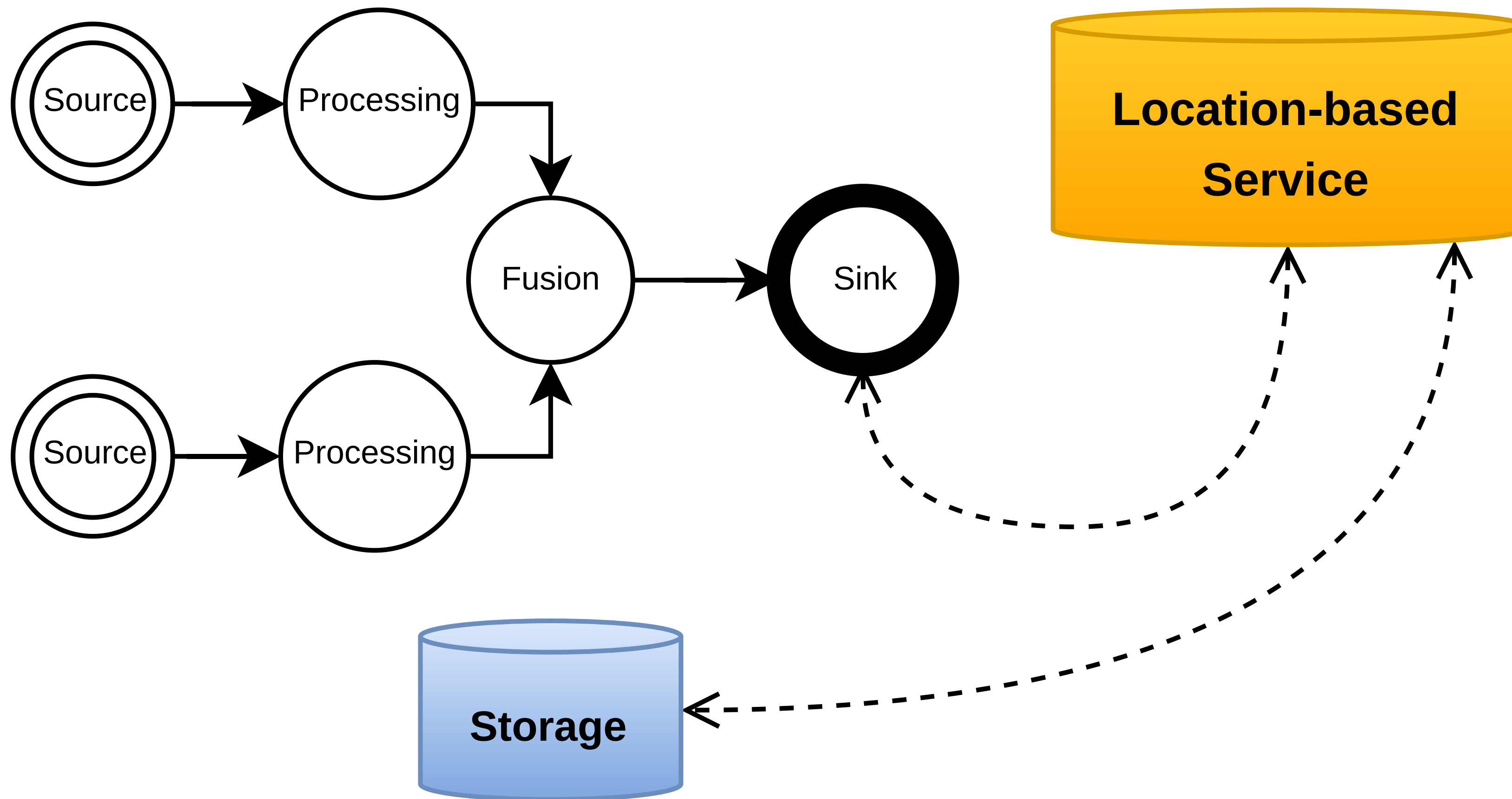
An object that semantically defines a space

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



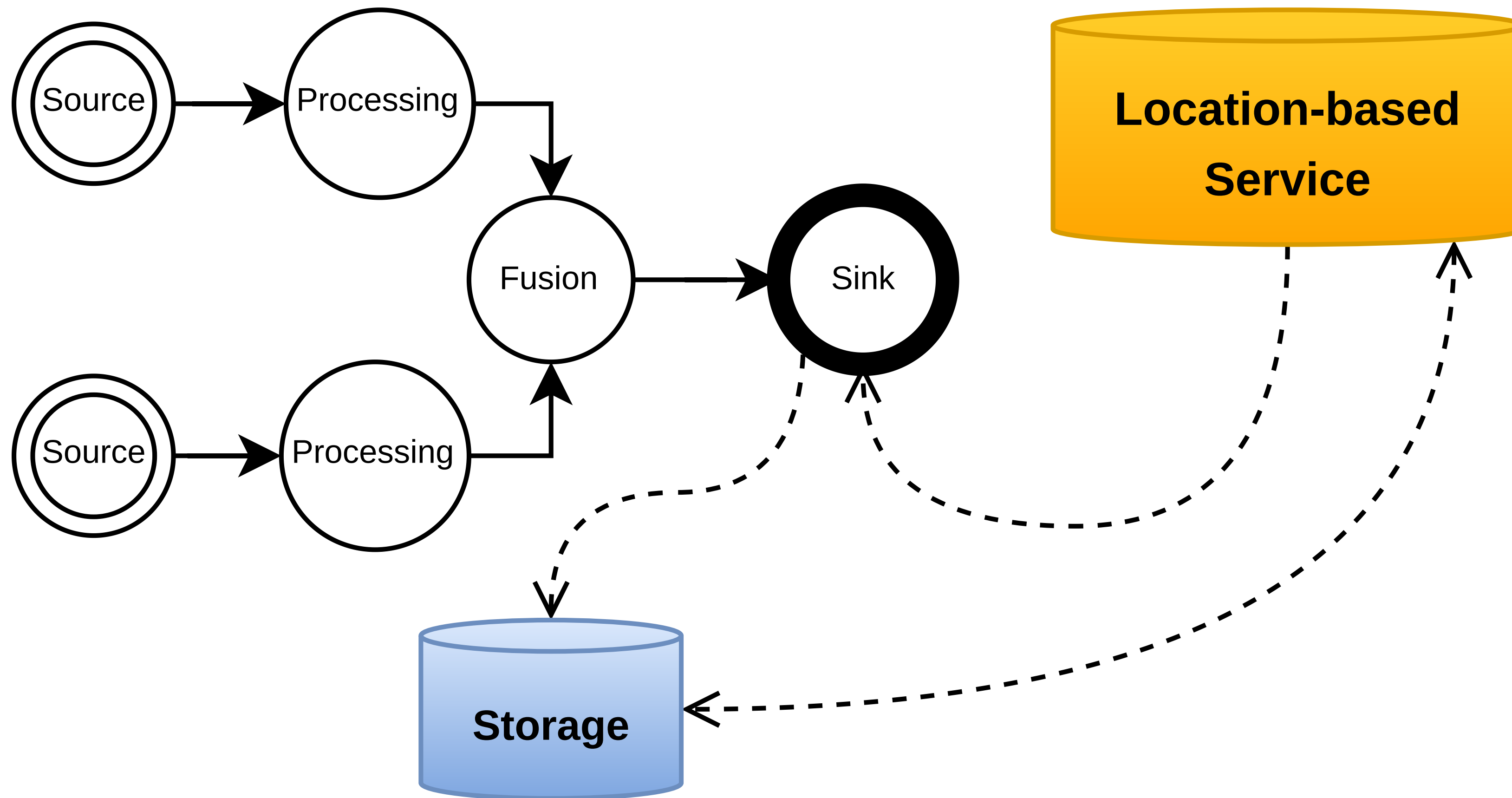
Location-based Service

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



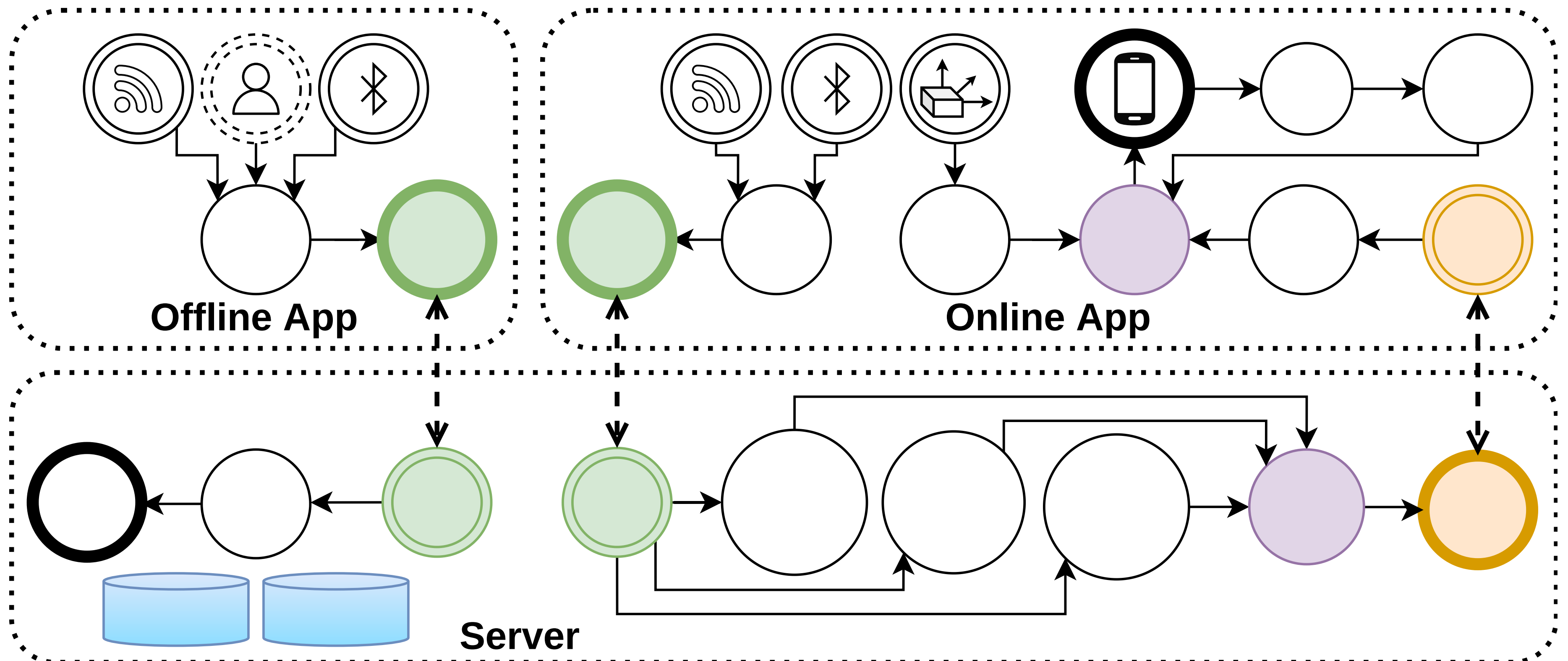
Location-based Service ...

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

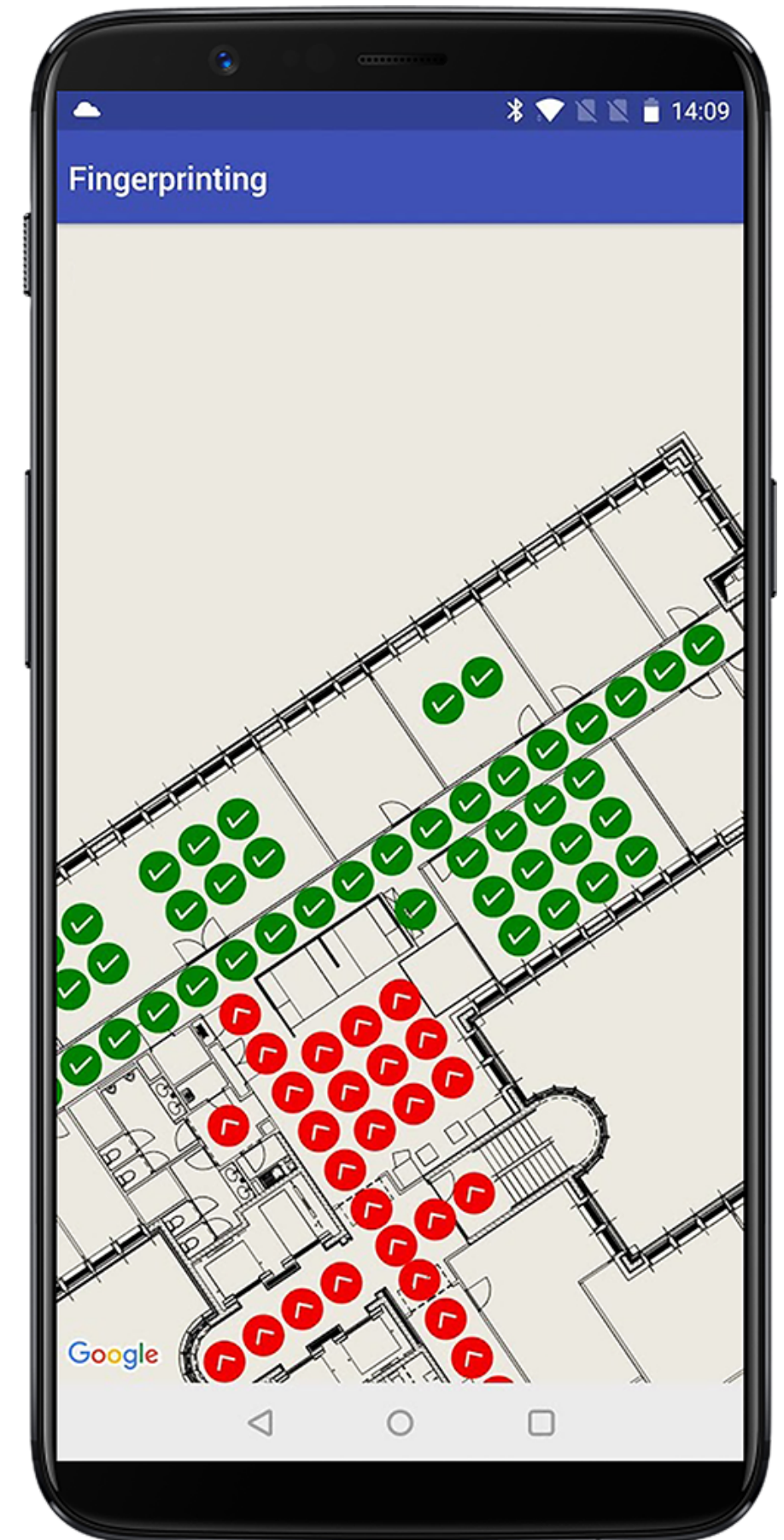
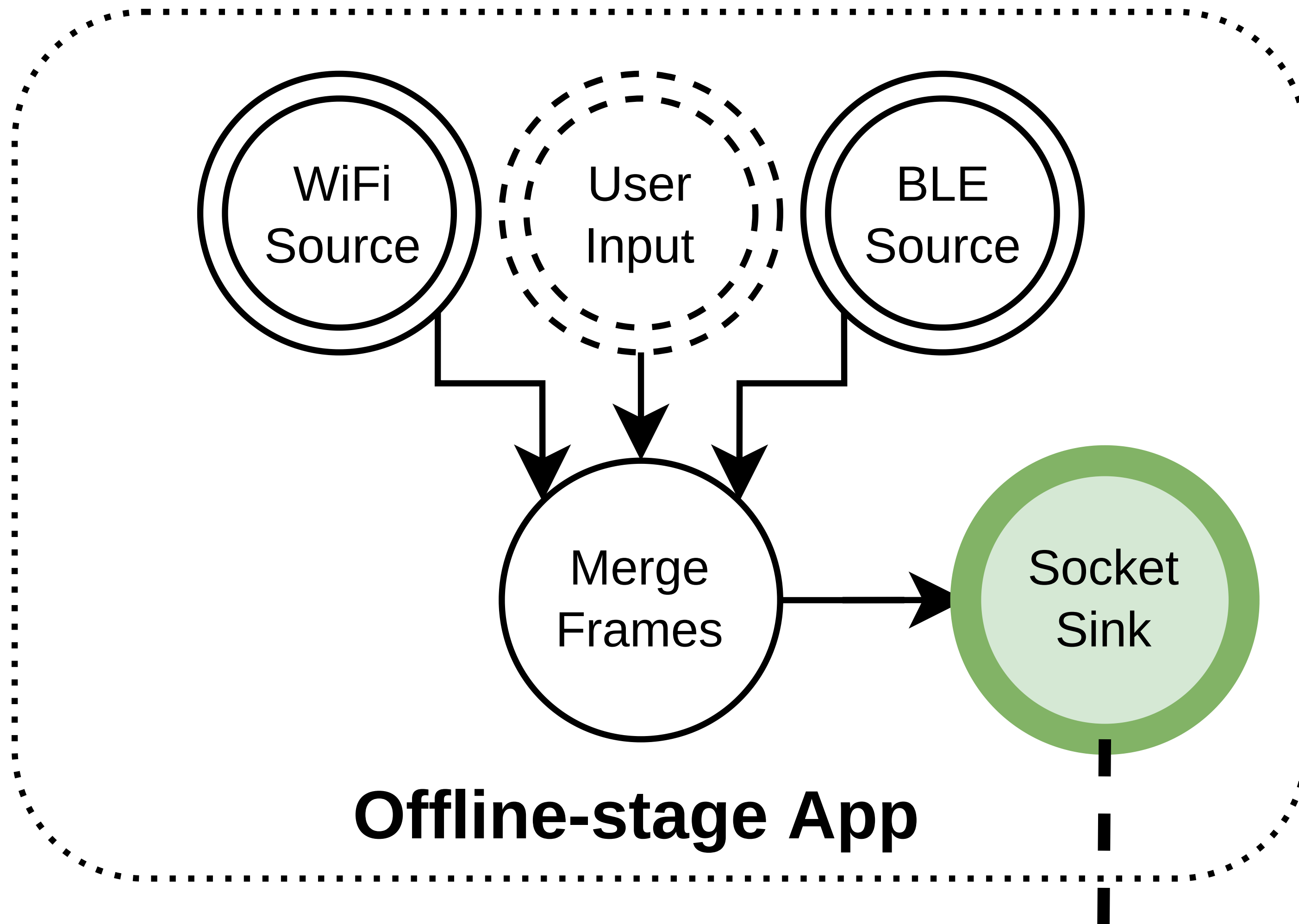


Demonstration

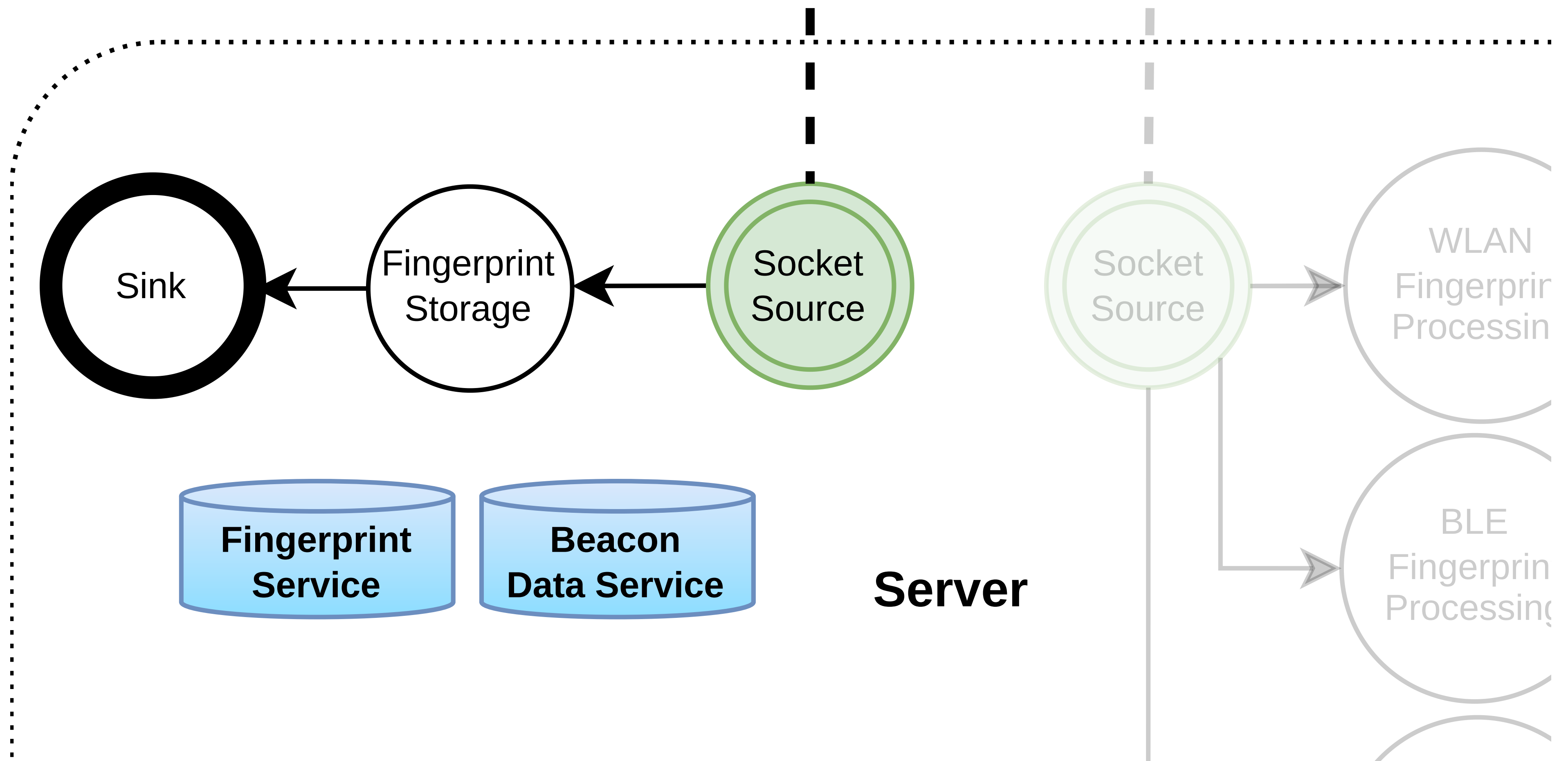
- ▶ Indoor positioning **use case**
- ▶ Use **existing techniques**
- ▶ Validation of **flexibility** and modularity



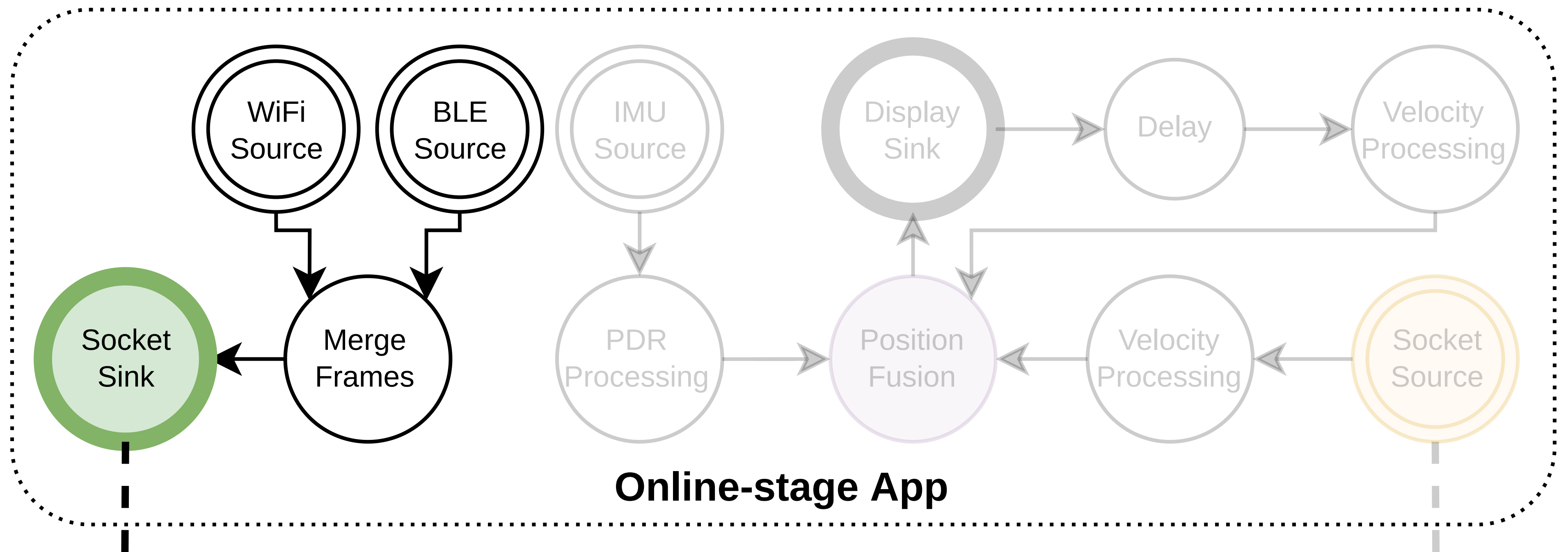
Positioning Model



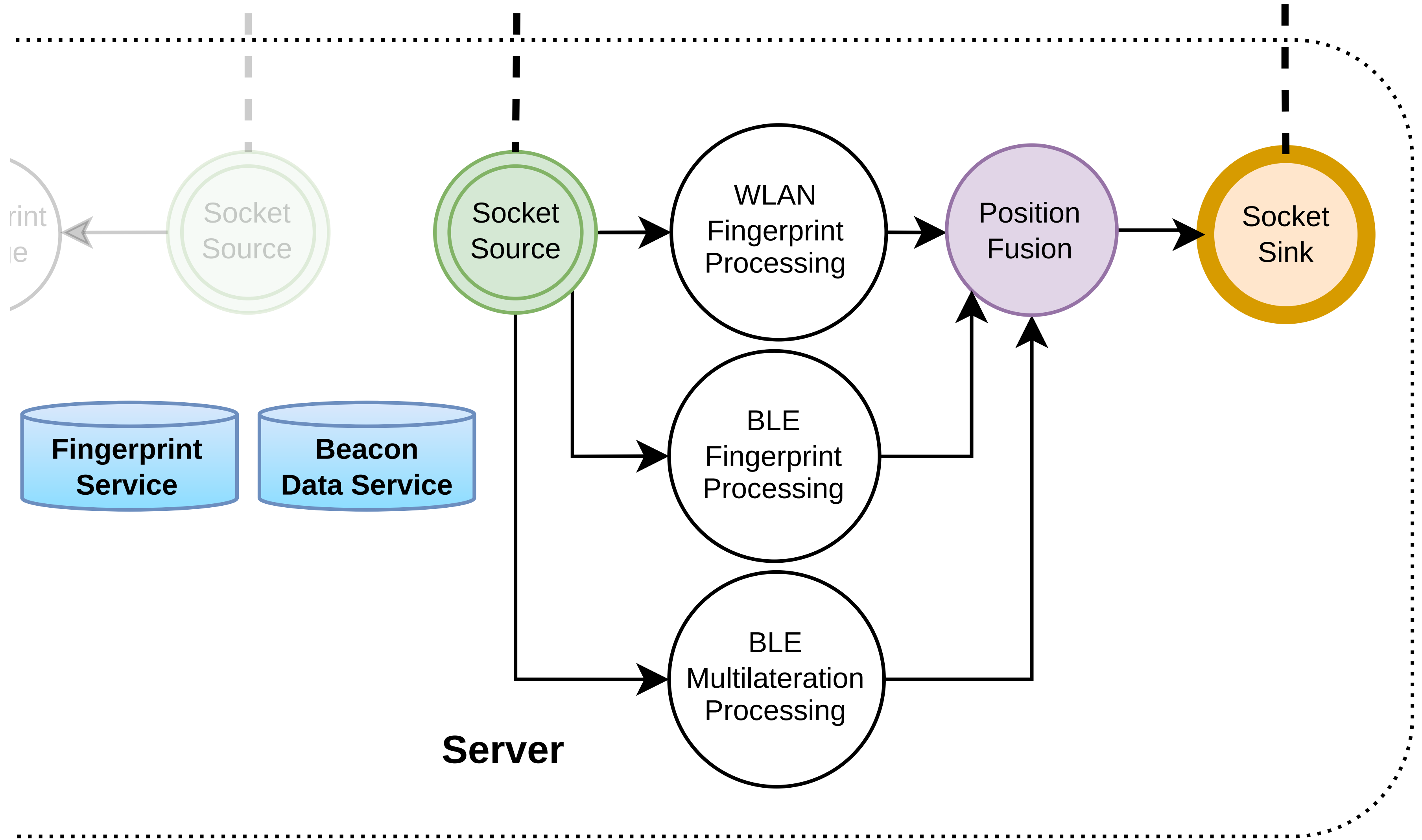
Positioning Model ...



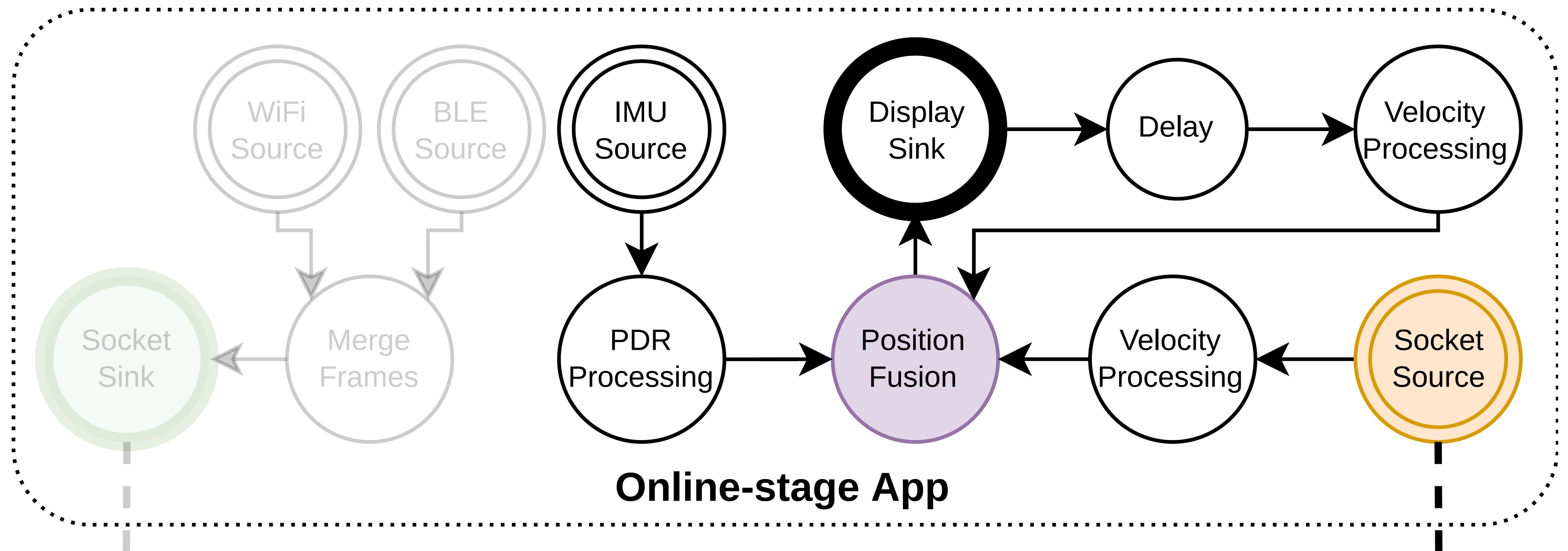
Positioning Model ...



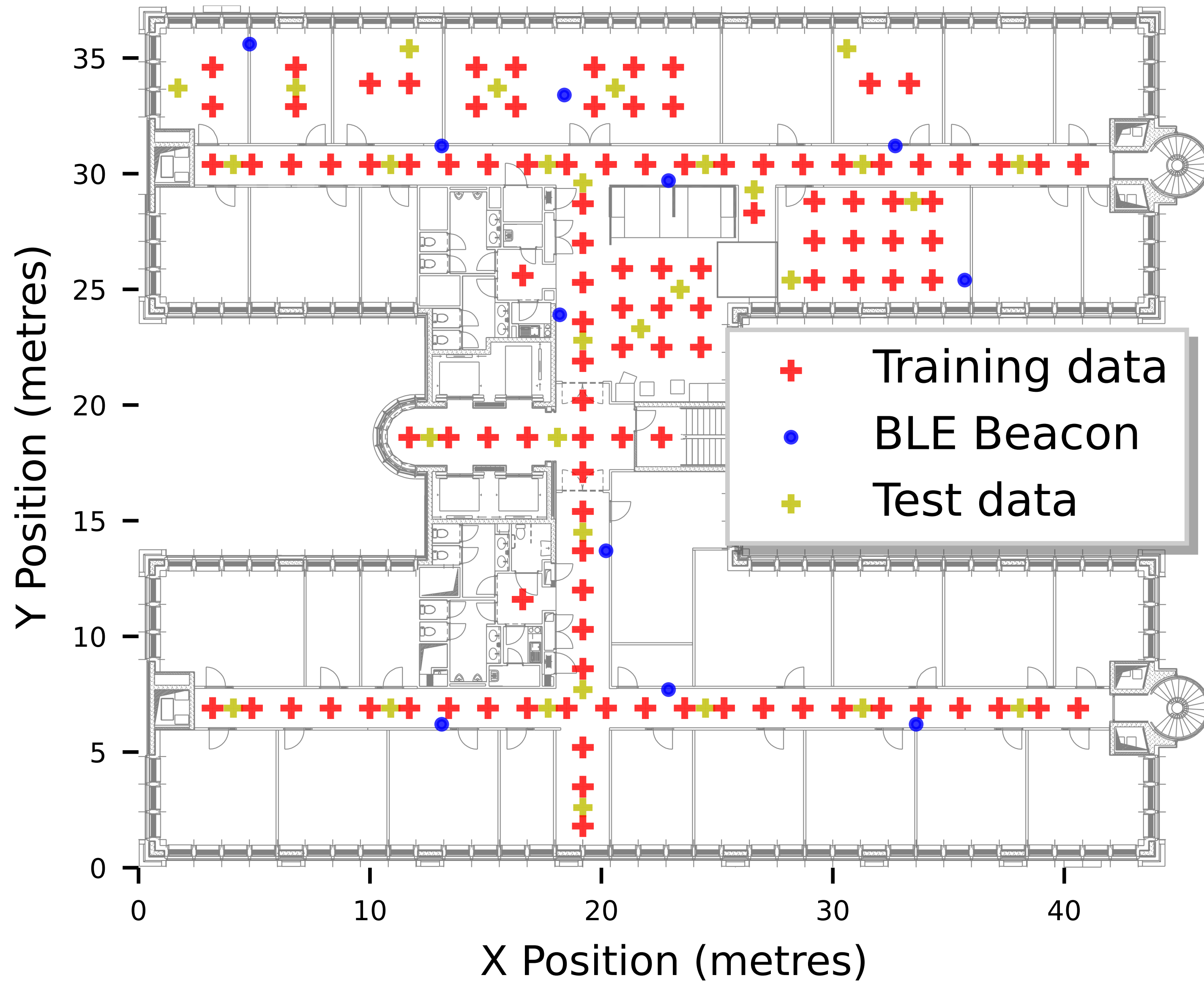
Positioning Model ...



Positioning Model ...



Dataset



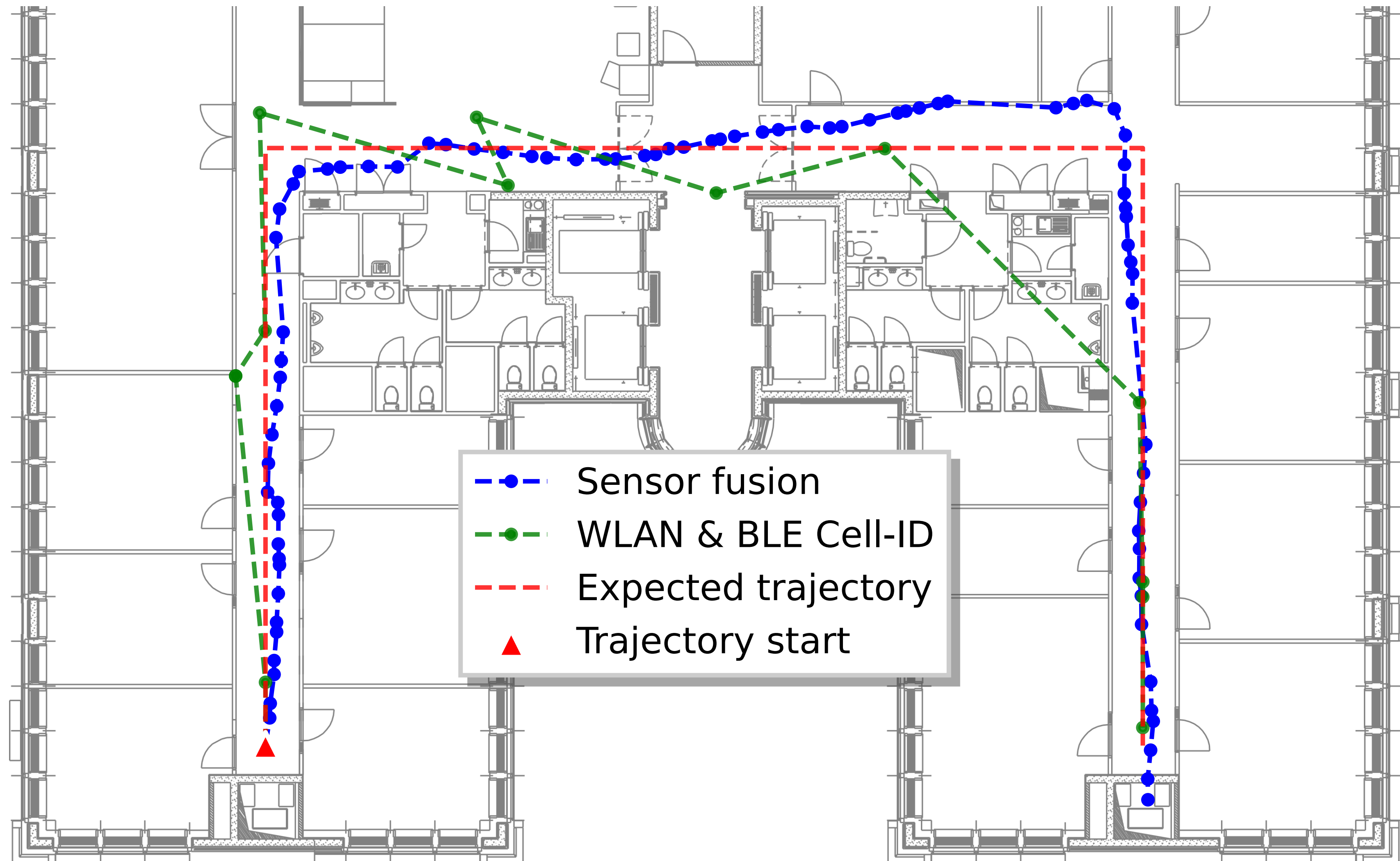
Validation Results

Static Positioning

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

Validation Results ...

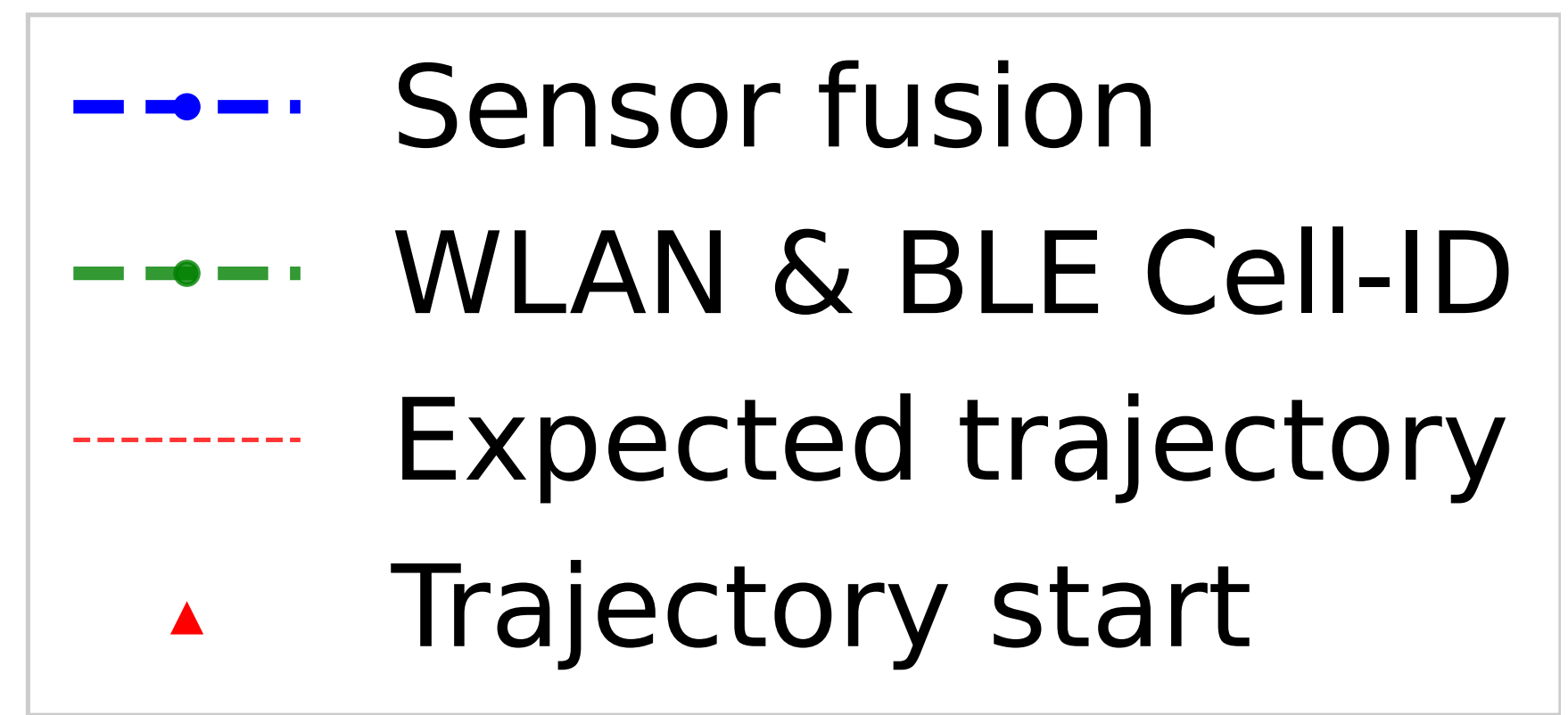
Trajectories



Validation Results ...

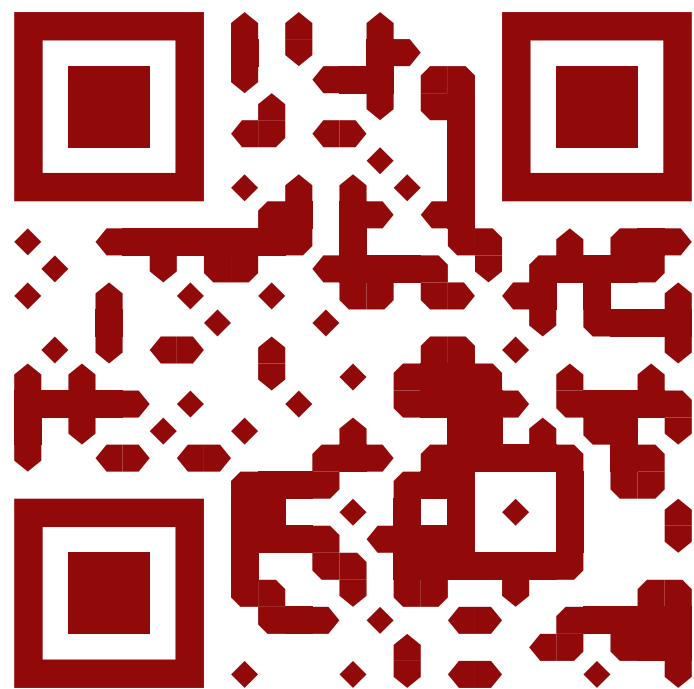
Trajectories

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



Contributions and Conclusions

- ▶ 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!