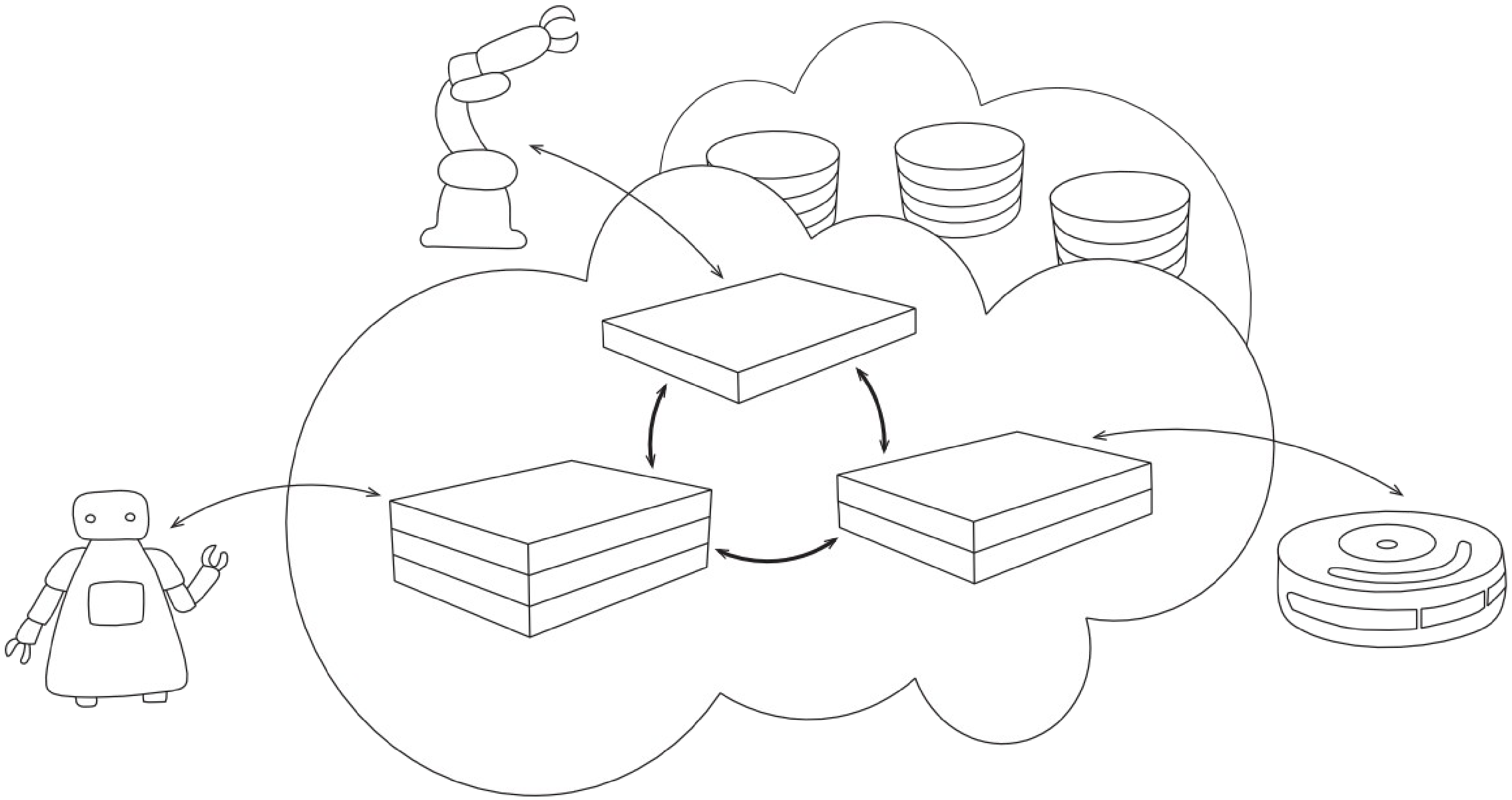
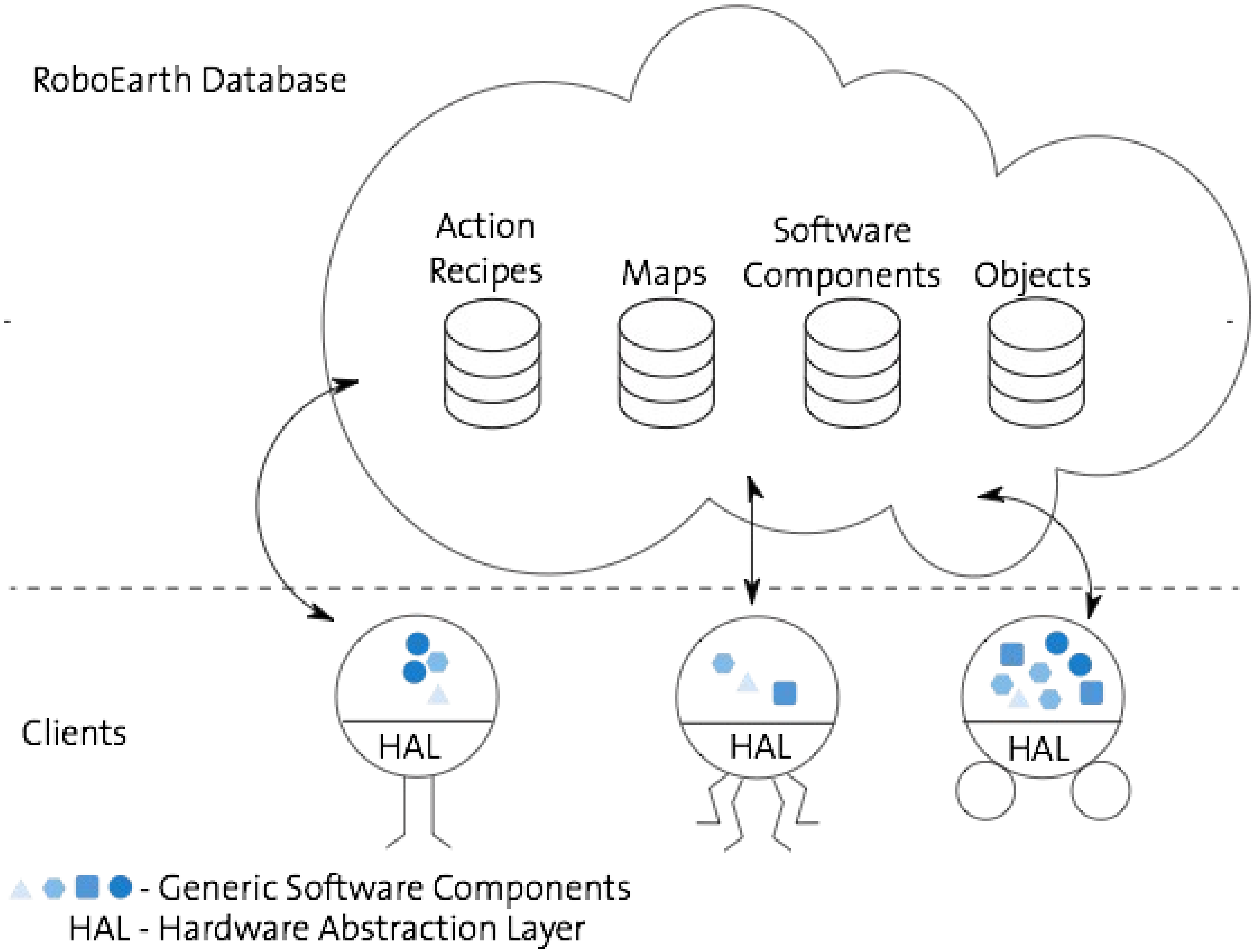


# Rapyuta: A Cloud Robotics Platform

Gajamohan Mohanarajah, PhD candidate, ETH Zurich

RoboEarth

# Motivation - I



RoboEarth Database

Action Recipes

Maps

Software Components

Objects

Clients

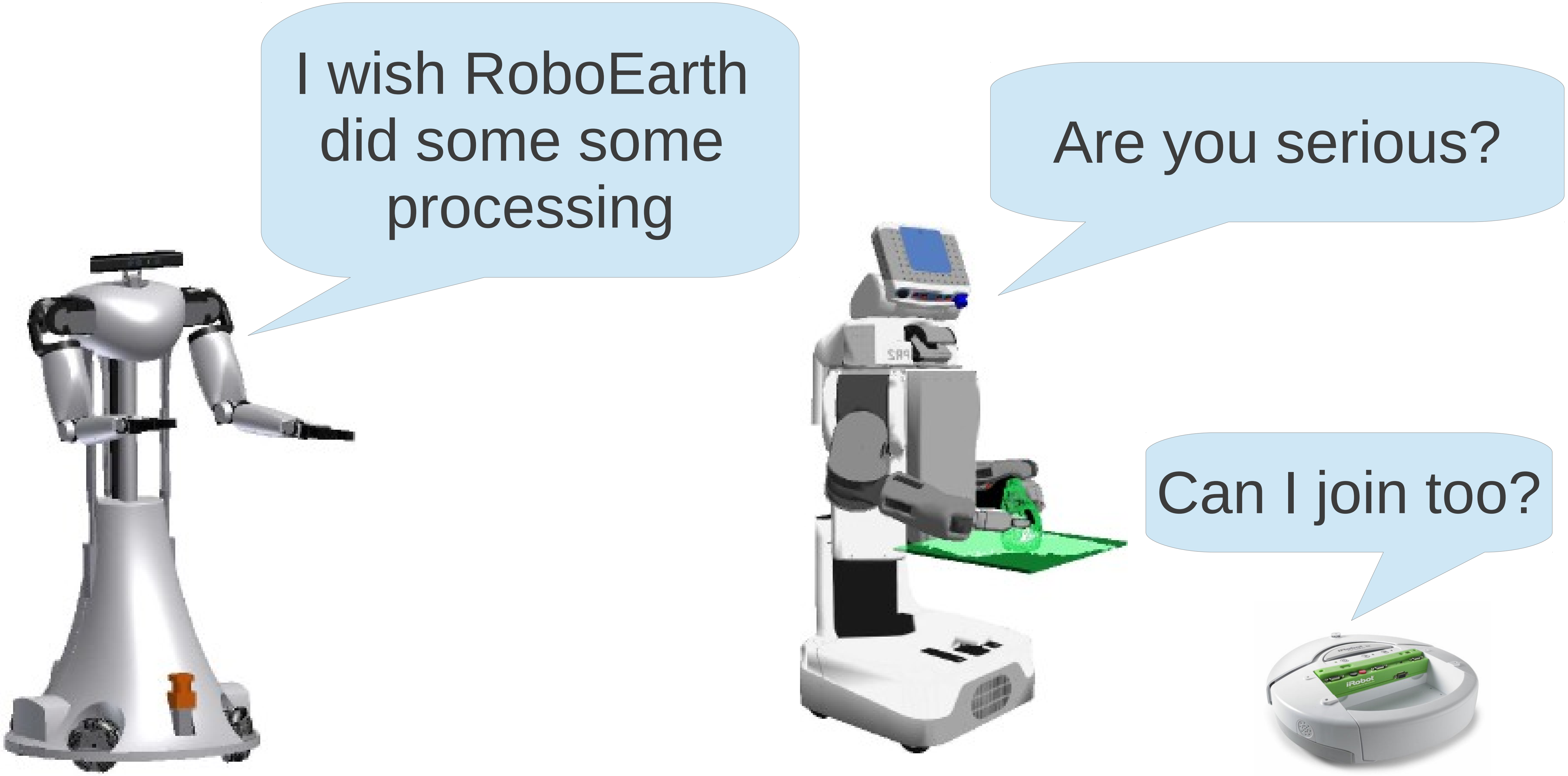
HAL

HAL

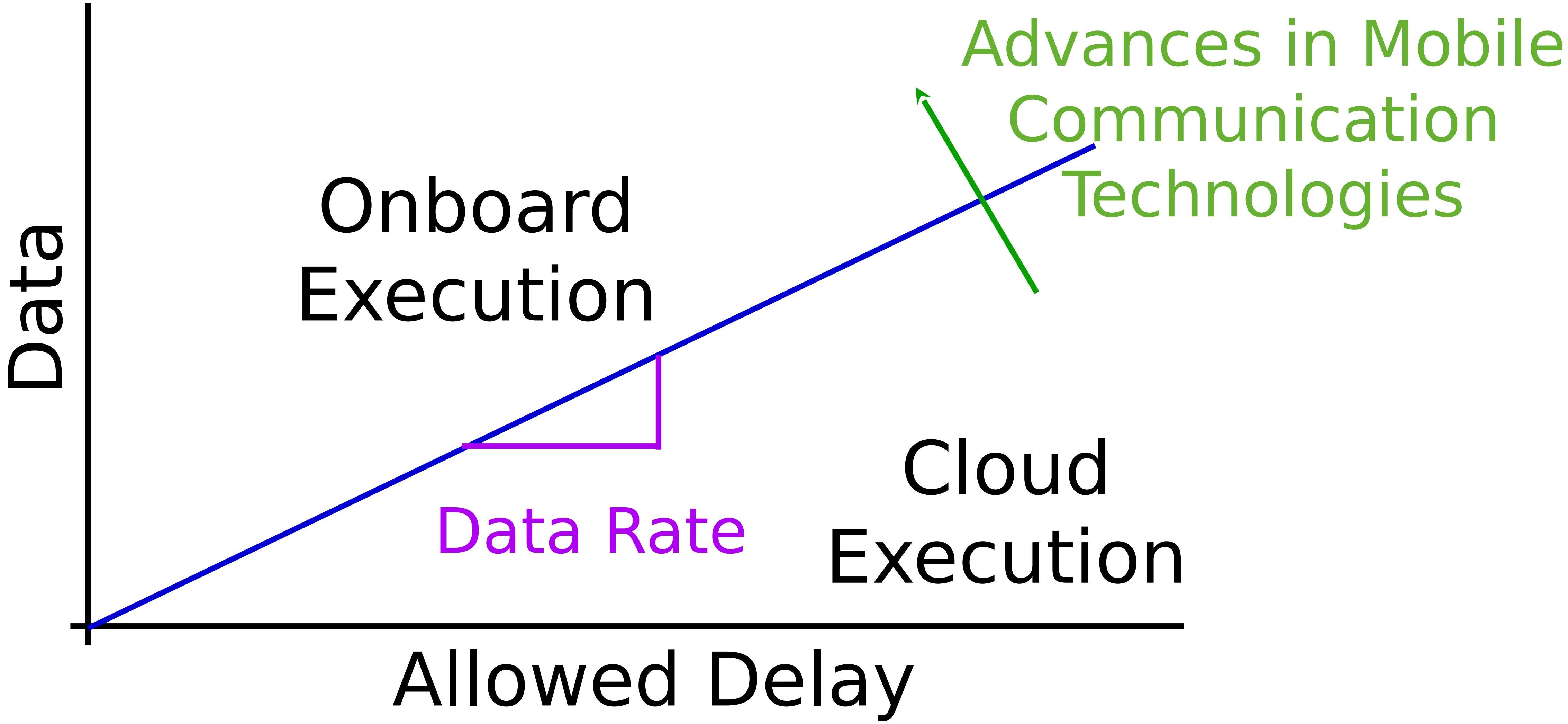
HAL

▲ ● ■ ● - Generic Software Components
HAL - Hardware Abstraction Layer

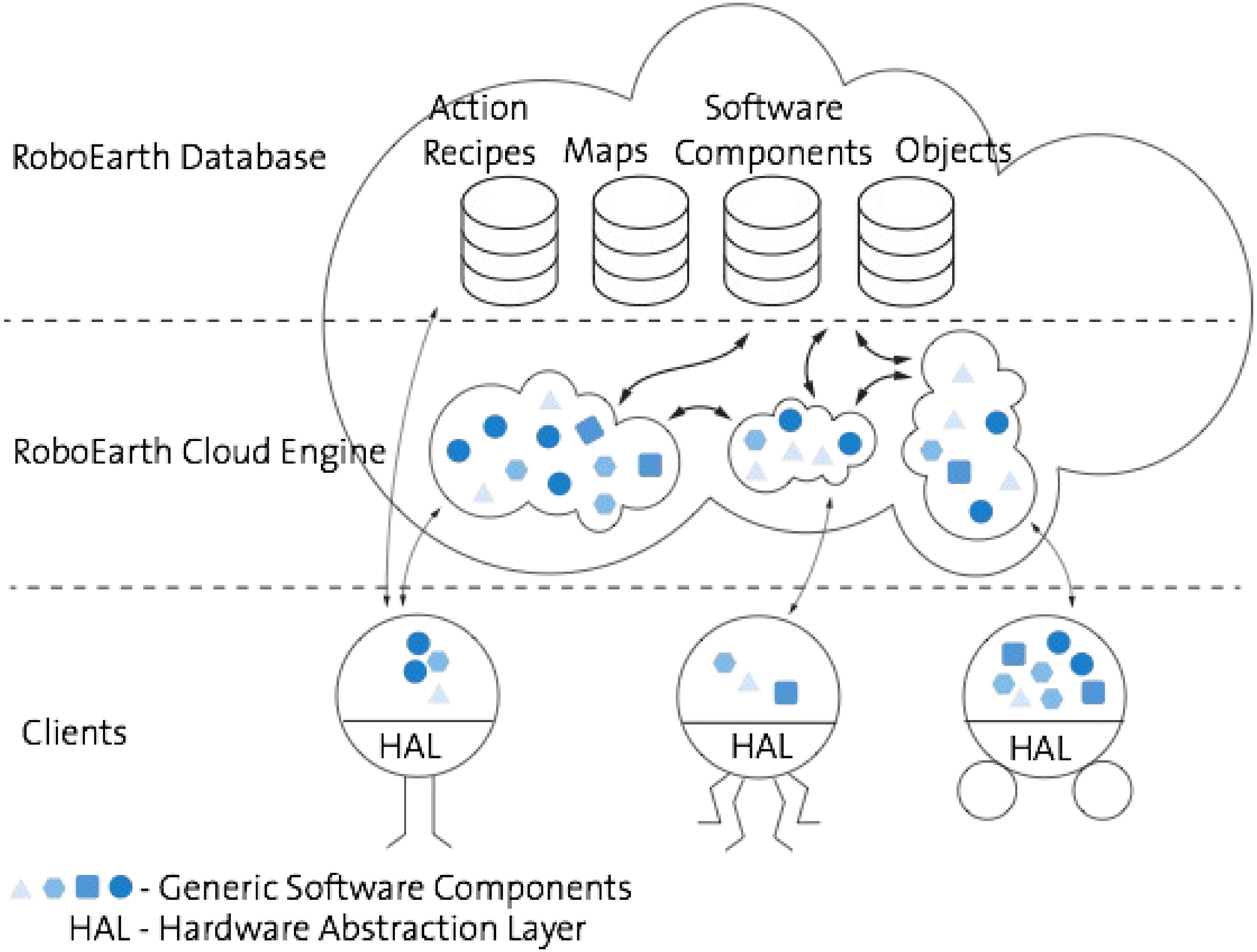
# More challenges along the way...



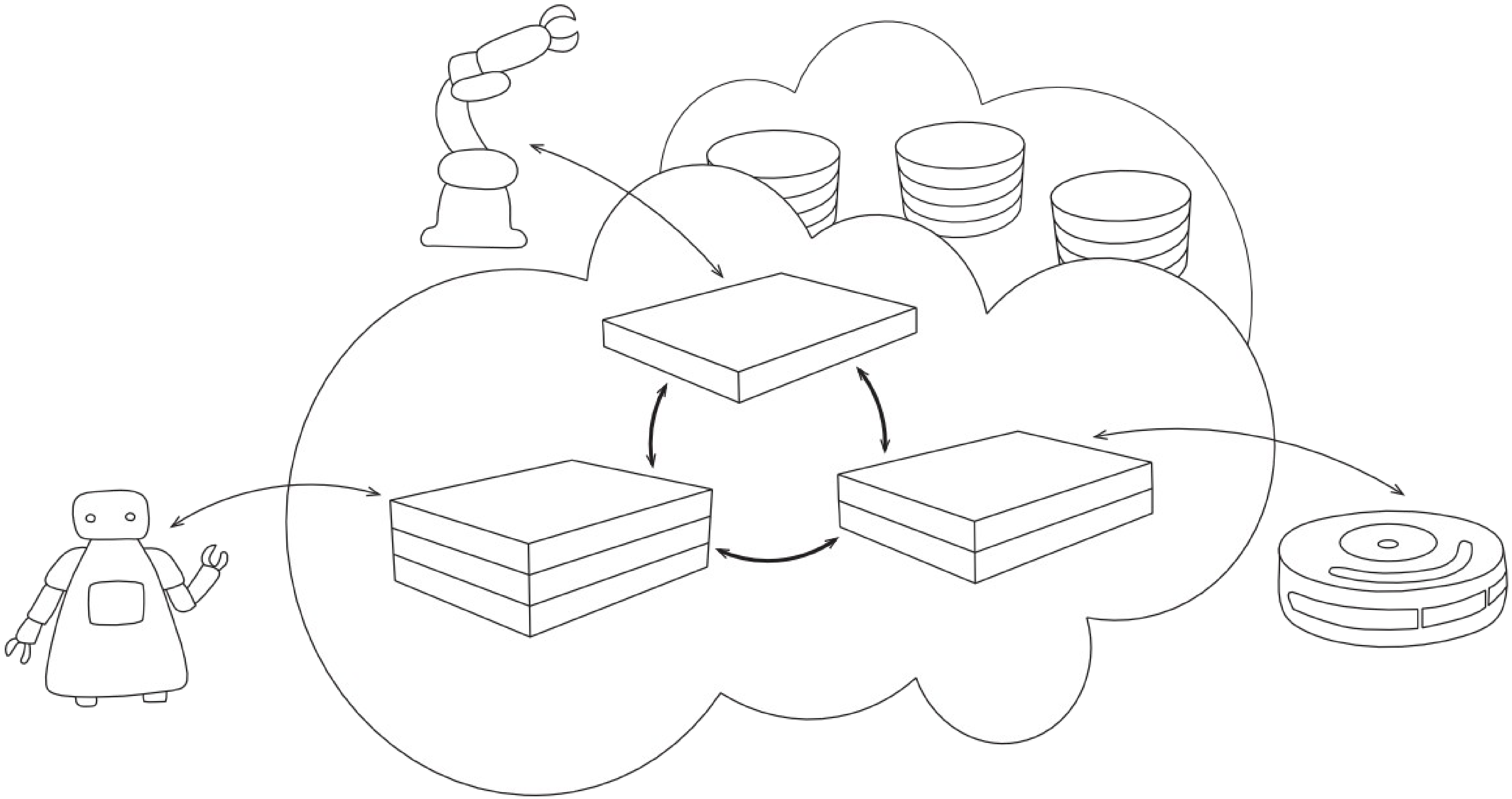
# Motivation – II



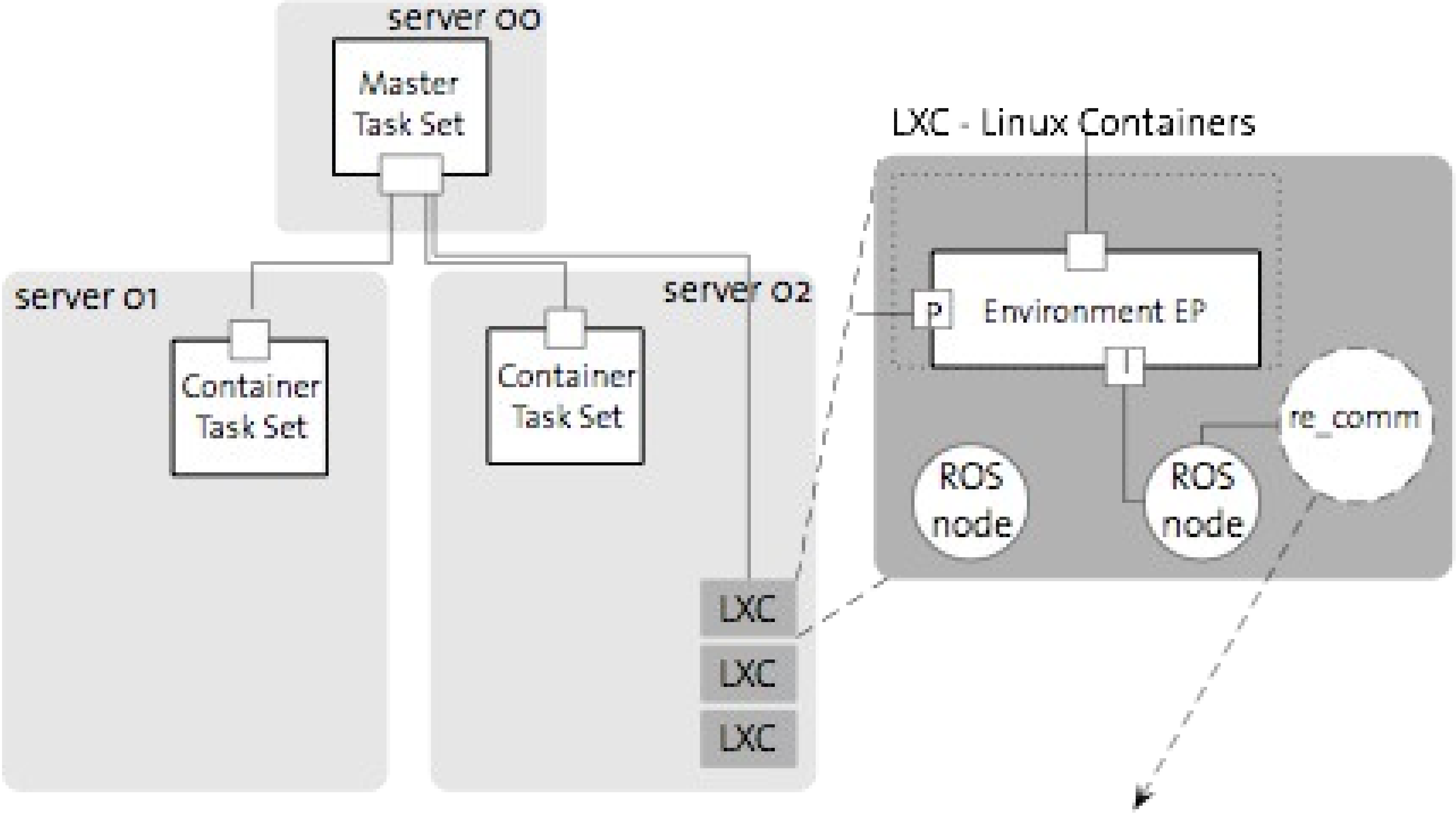
# The Complete Picture!



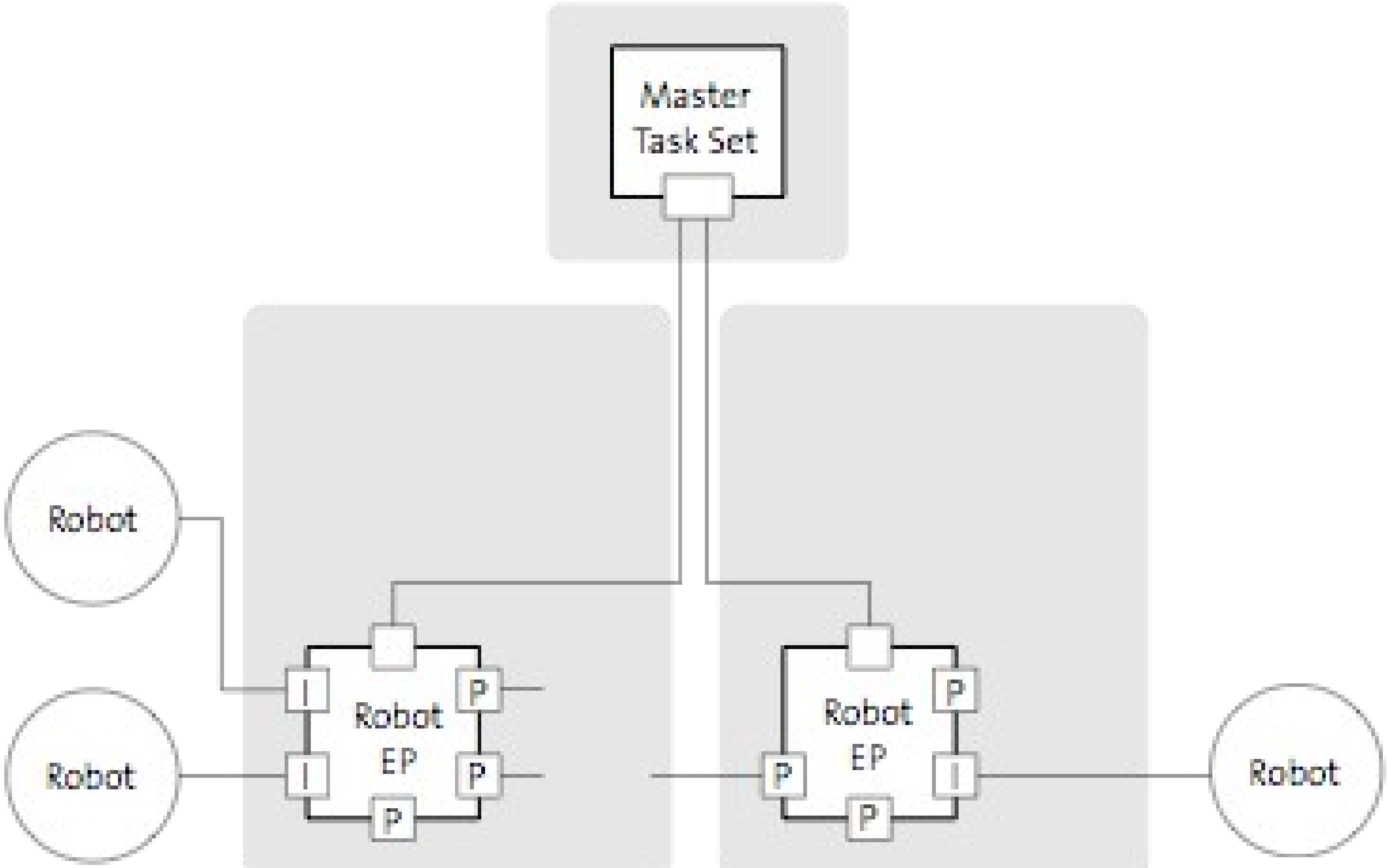
# Let's look into details



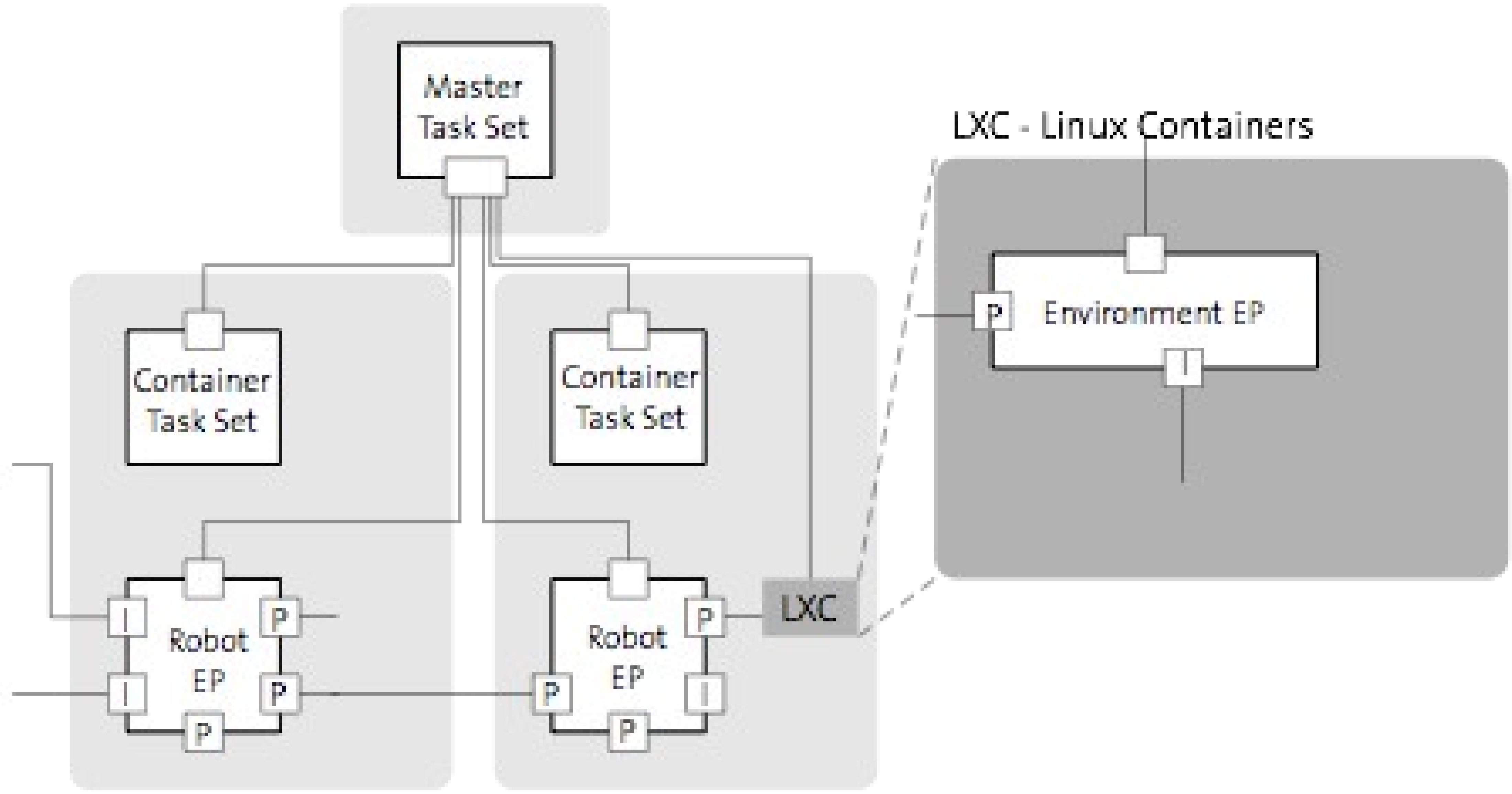
# Details


server 00
Master Task Set
server 01
Container Task Set
server 02
Container Task Set
LXC
LXC
LXC
LXC - Linux Containers
P
Environment EP
I
re_comm
ROS node
ROS node

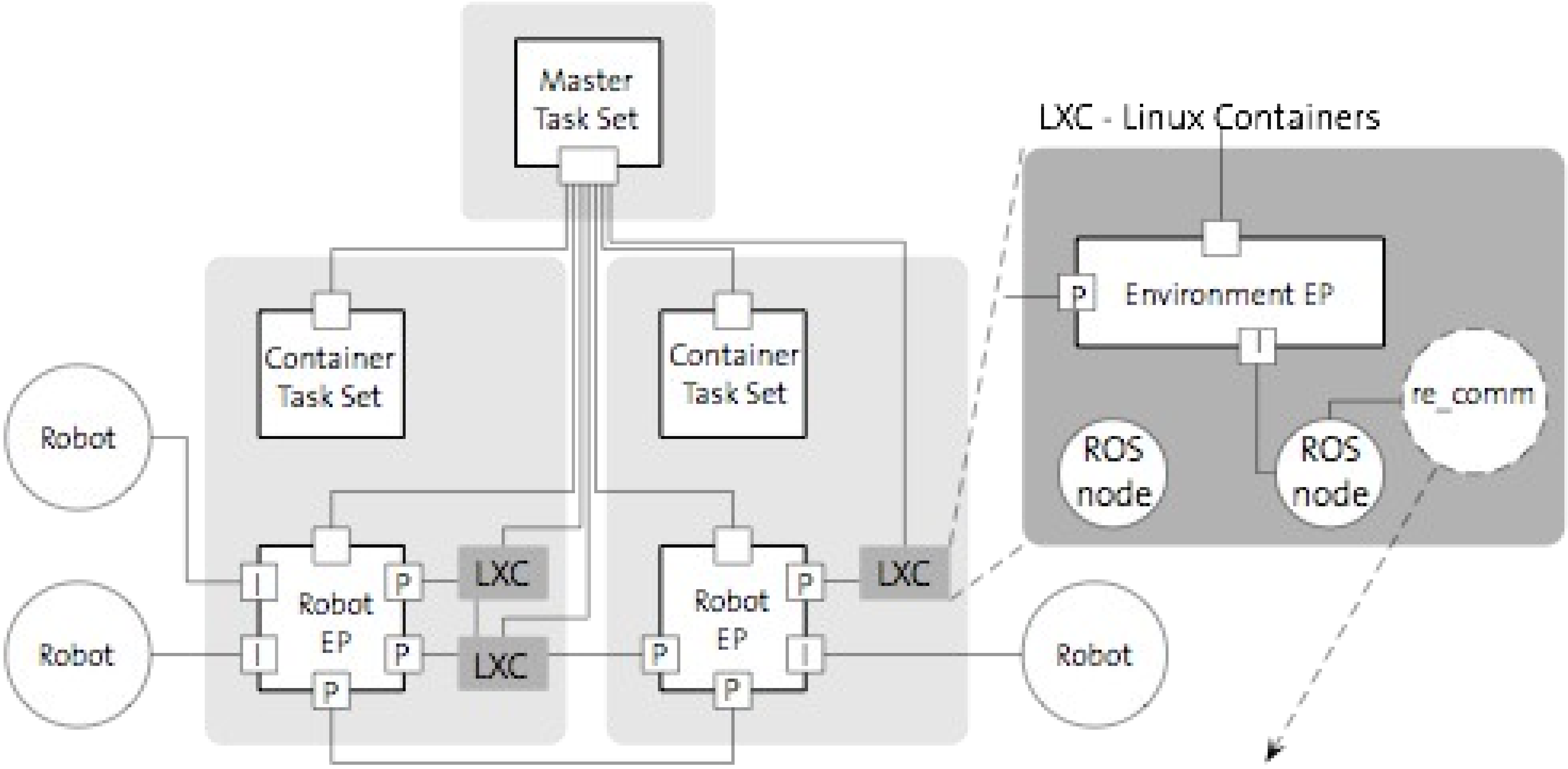
# Details


Master
Task Set
Robot
Robot
Robot
EP
I
I
P
P
P
Robot
EP
P
P
I
P
Robot

# Details

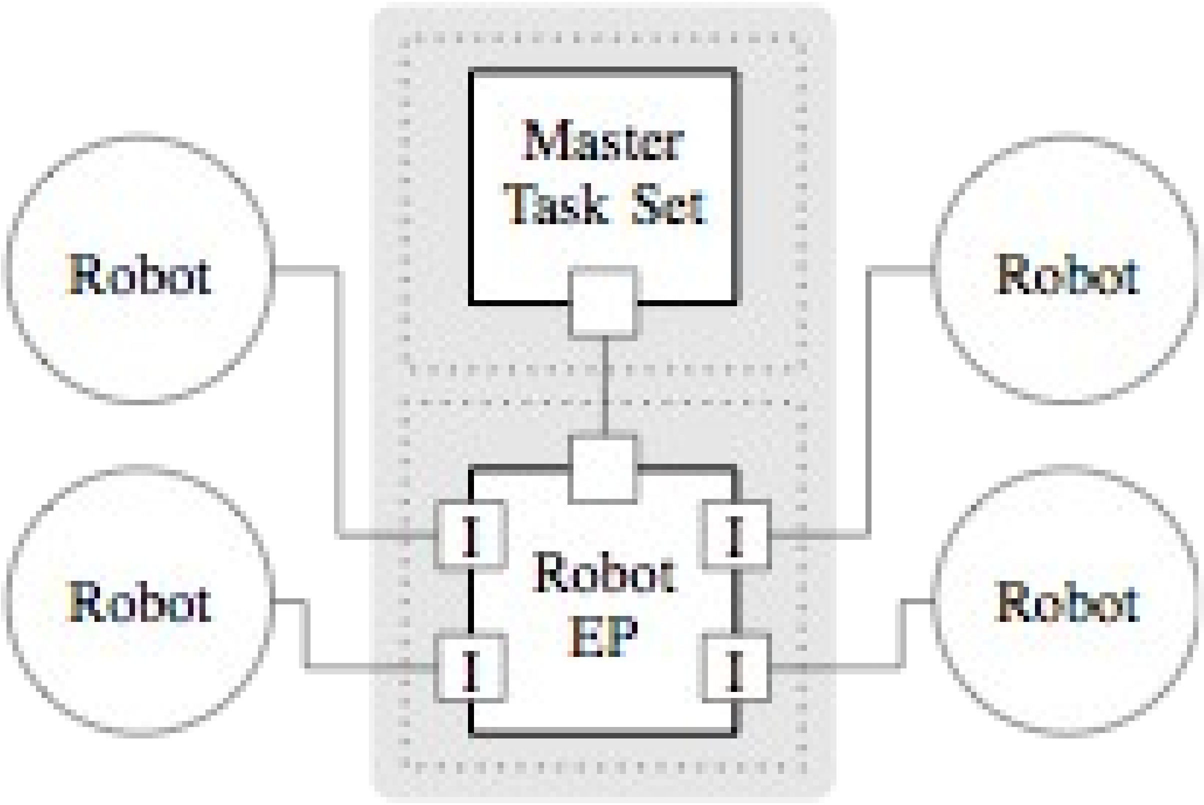

Master Task Set
LXC - Linux Containers
Environment EP
P
I
Container Task Set
Container Task Set
Robot EP
I
I
P
P
P
Robot EP
P
P
I
P
LXC

# Details


Master Task Set
LXC - Linux Containers
Container Task Set
Container Task Set
Environment EP
P
I
re_comm
ROS node
ROS node
Robot
Robot
Robot
Robot EP
Robot EP
I
I
P
P
P
P
P
I
P
LXC
LXC
LXC

# Other use cases

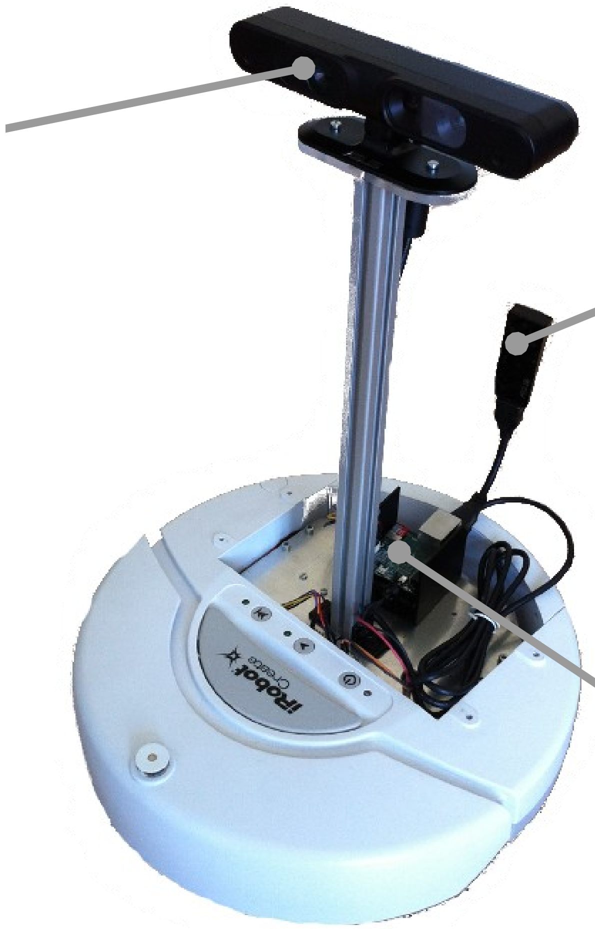
- Robot Networking



- rosbridge like functionality
- multi master functionality

# Turtlebot: Internet Edition

PrimeSense (200$)



ASUS USB-N53 (35$)

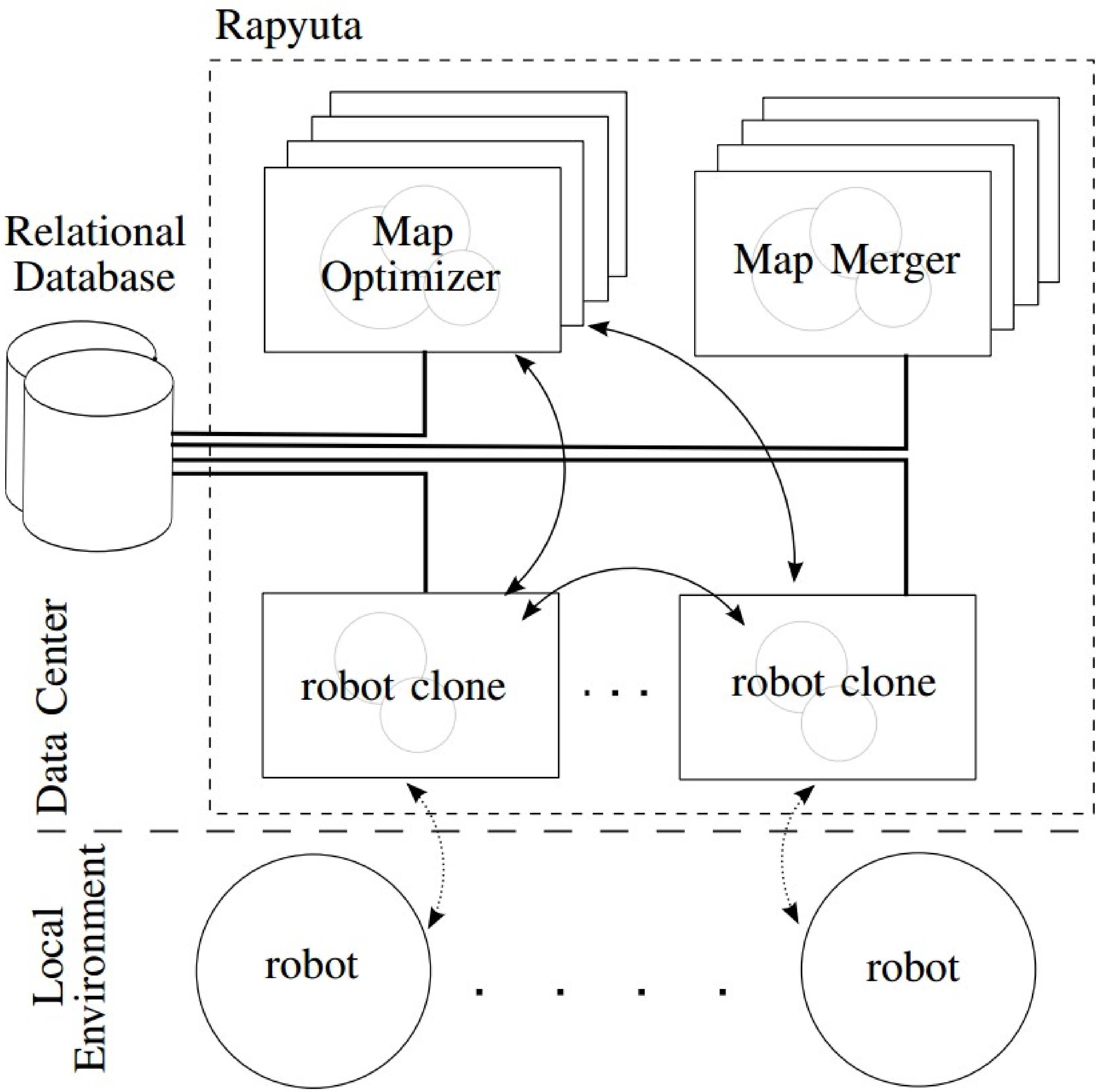
ODROID U2 (~90$)

Total price < 550$

Task: Tokyo Big Sight #610, Amazon Ireland server, map it in real time!

RoboEarth

# Architecture Overview

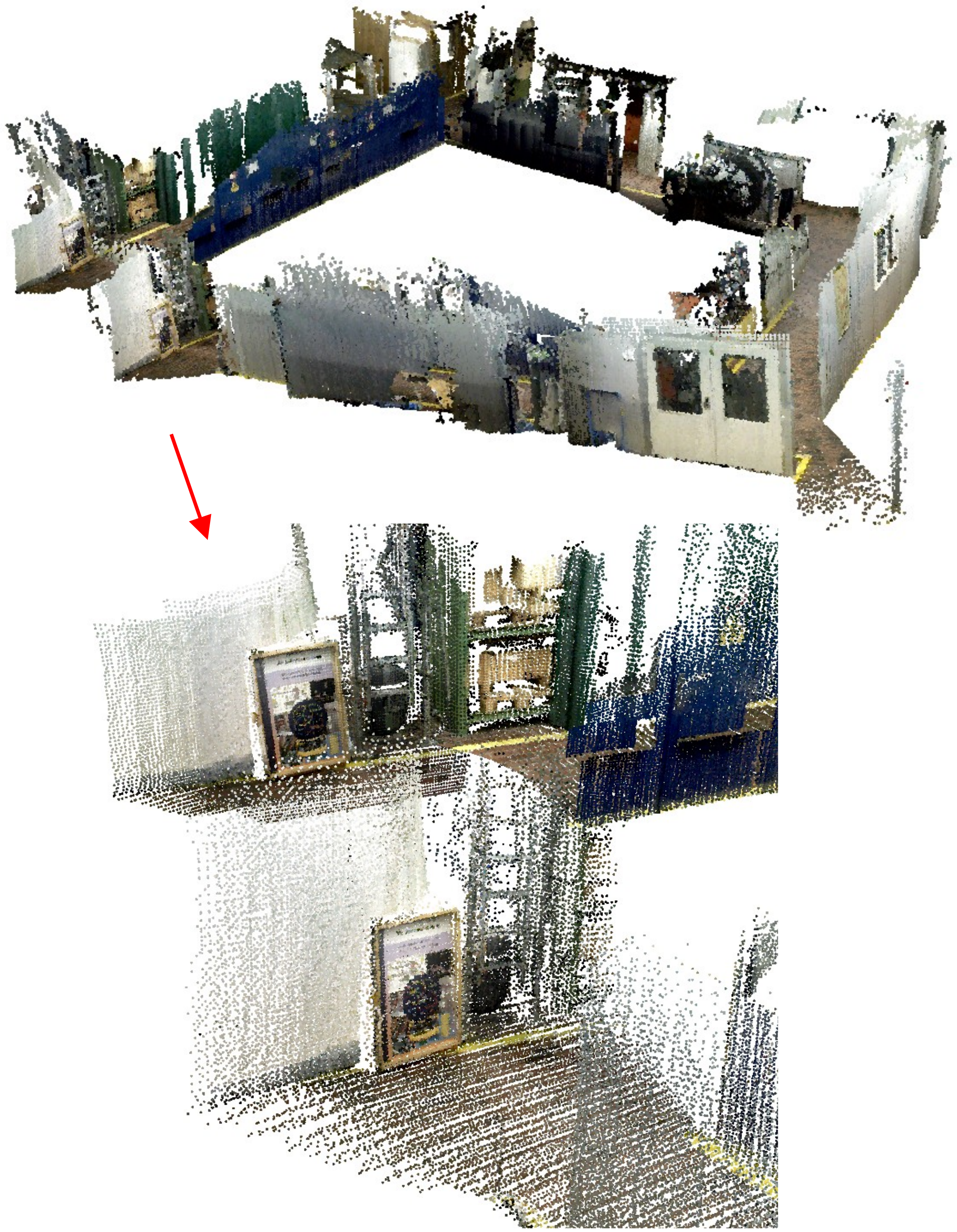


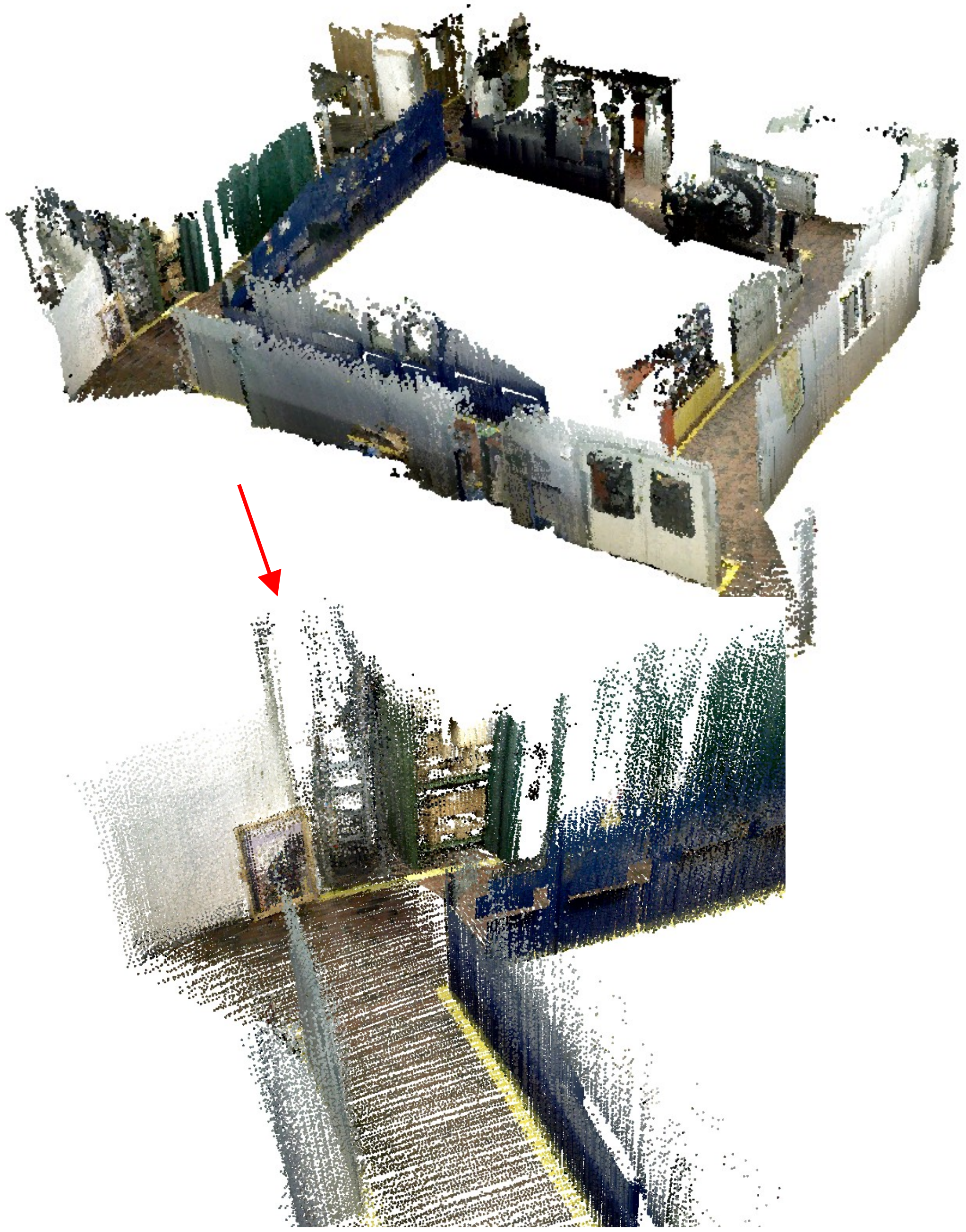
RoboEarth

# Visual Odometry

- Process of determining camera pose from image sequence
- Usually feature-based incremental frame to frame matching
- Our approach - keyframe based dense visual odometry
- Dense visual odometry - uses all pixels to estimate the pose (based on Steinbruecker et al. 2011 & Kerl et al. 2013)
- Keyframe - reference frame that is used to estimate relative pose of other frames

# Loop Closure and Global Optimization



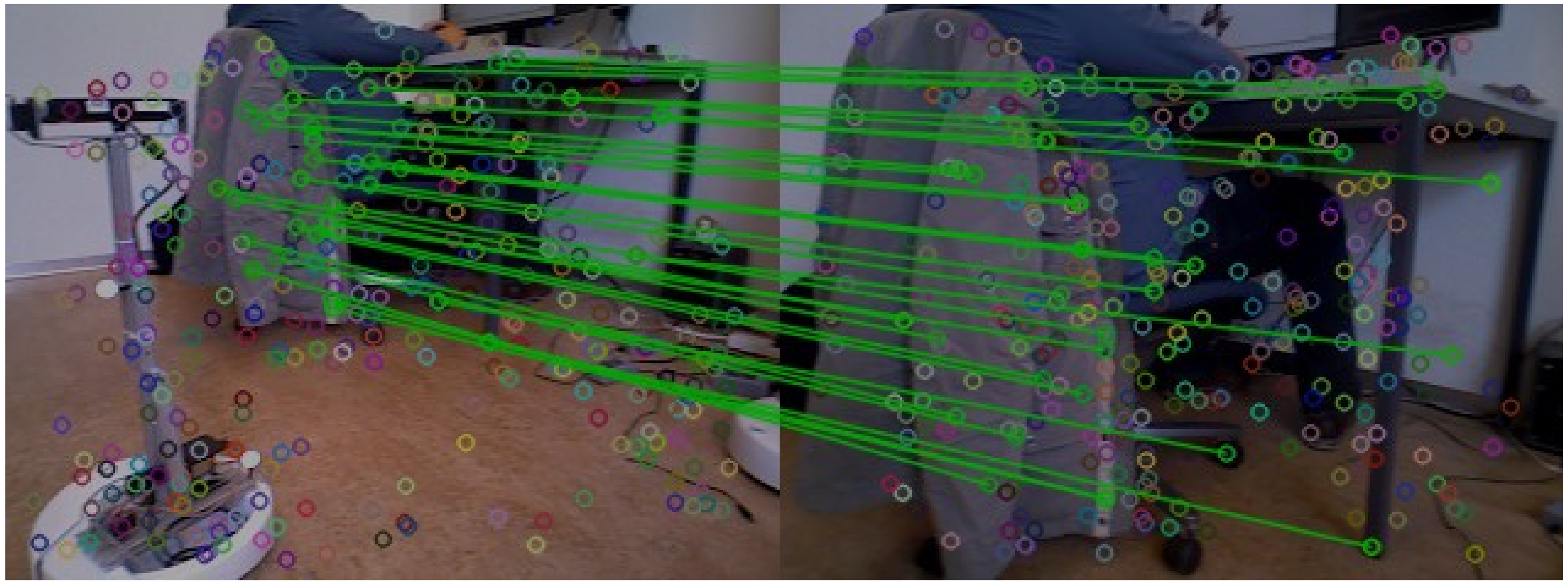


# Global Optimization Algorithm

- Step 1 : Build error graph of the problem
  - Find constraints (edges) between all pairs of keyframes (parallelize)
- Step 2 : Optimize the global error function
  - Open-source packages - g2o, ceres

# Map Merging

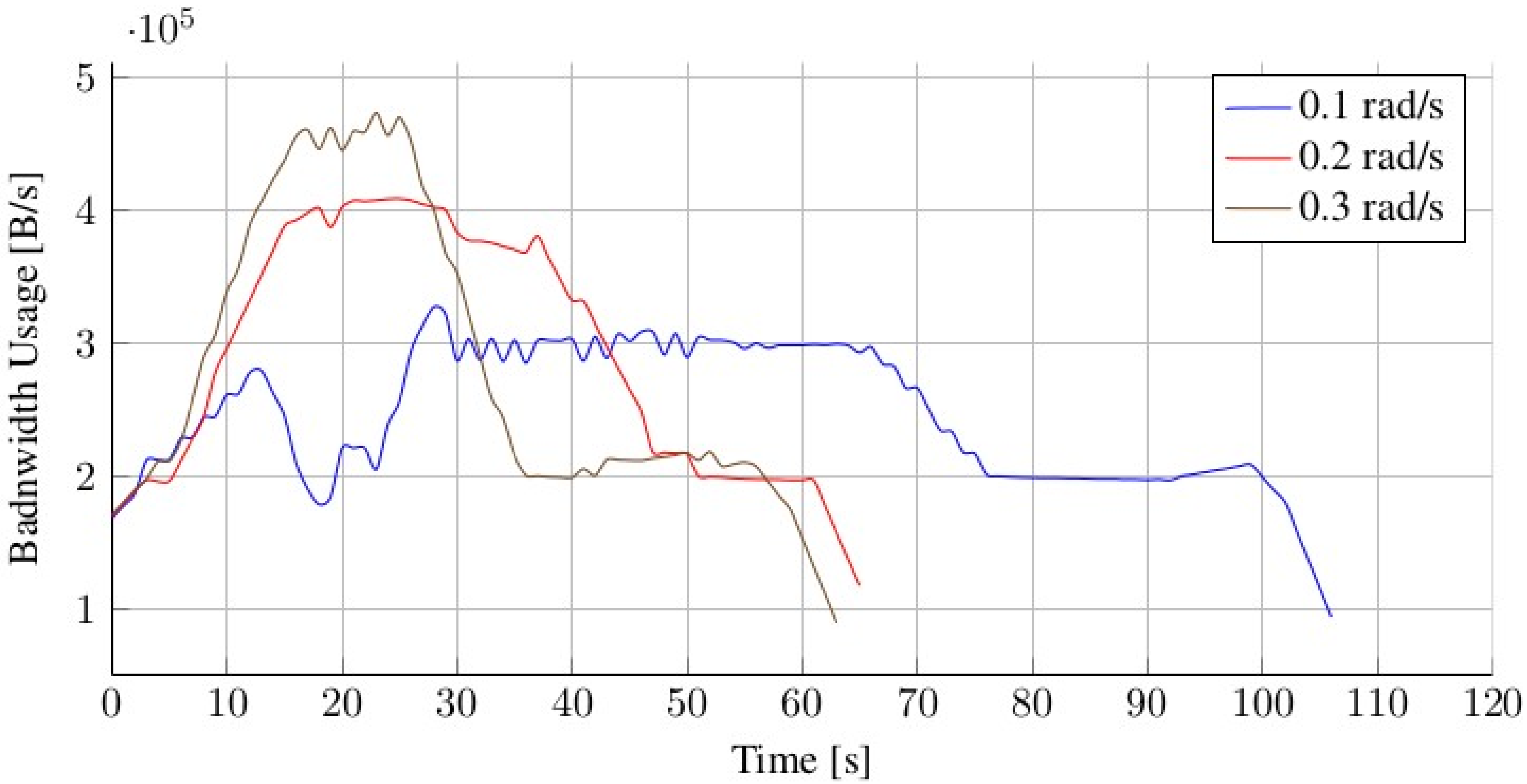
- Select a random frame from one map
- Extract features
- Try to match with frames from other maps



RoboEarth

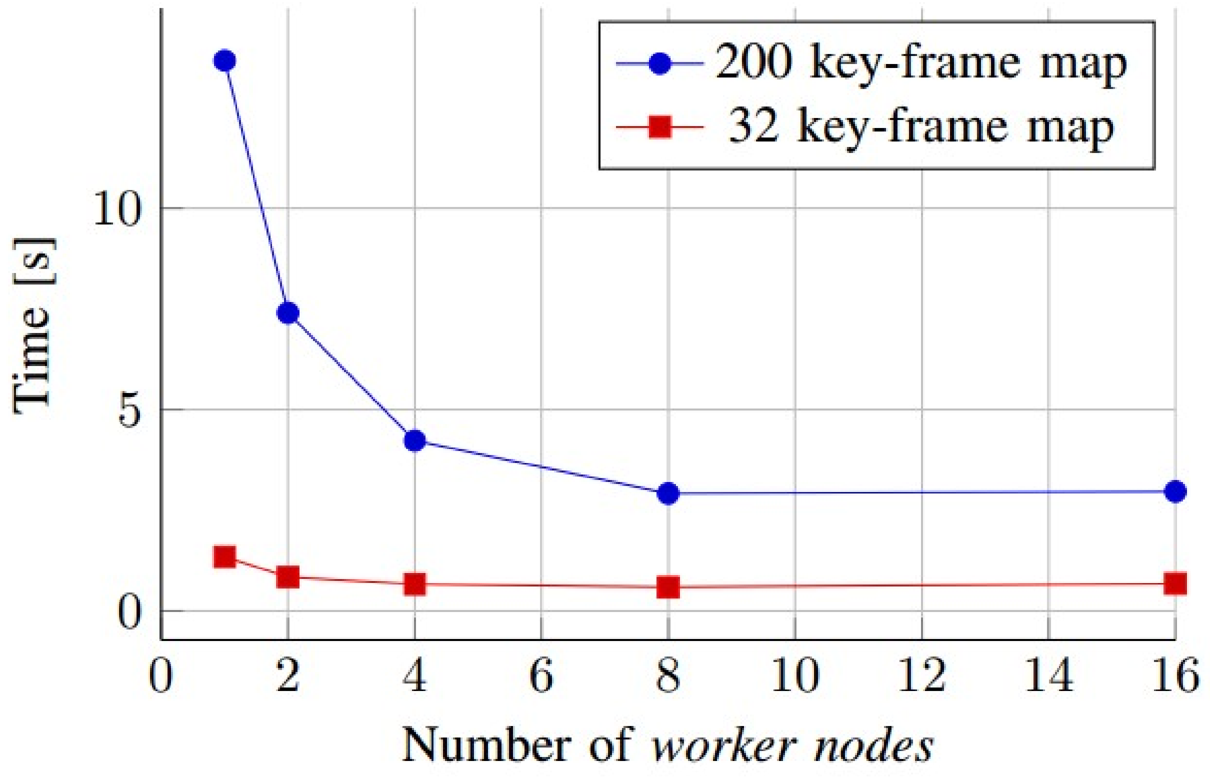
# Evaluation of Bandwidth

- 3.5 MB/s for every frame (h.264 + FFV1)
- 300-450 KB/s for keyframes only
- Bandwidth is proportional to the speed of the robot



# Evaluation

- Parallel optimization speedup more visible on bigger datasets
- Computation time stops decreasing after 8 machines due to communication costs



# Demo

# Conclusion

- Rapyuta Overview / Architecture
- Cloud based multi-robot 3D mapping system

# Thank you for your attention

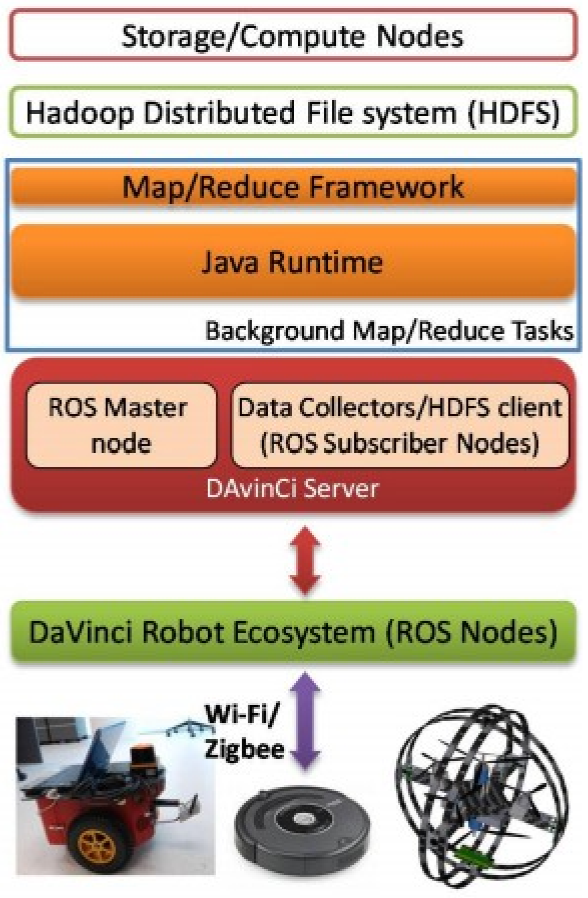
- Questions?

# Why not use a general PaaS for robotics?

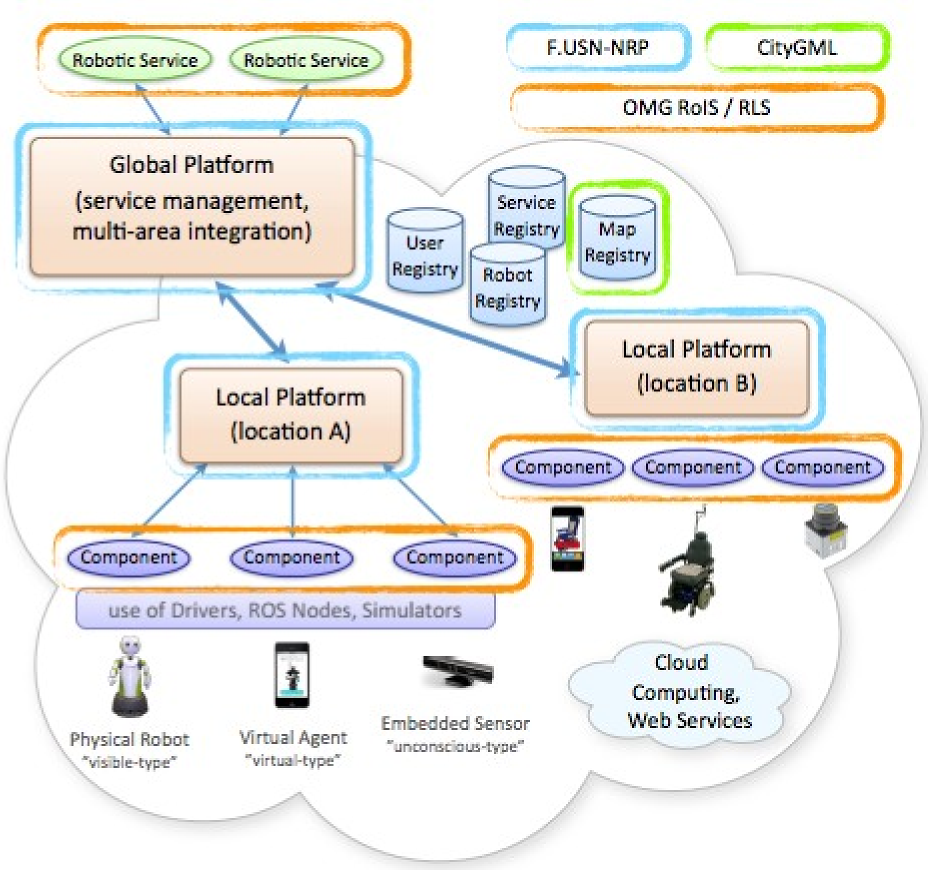
- Limitations:
- General: Most supports single process apps
- Google App Engine: Programing APIs, execution time limit
- Heroku: No WebSockets, No inter communication between containers
- OpenShift/Cloud Foundry (New!): General limitations still apply

# Other robotic PaaS like frameworks

- Davinci: Single ROS system in the cloud, Hadoop back end for mapping


Storage/Compute Nodes
Hadoop Distributed File system (HDFS)
Map/Reduce Framework
Java Runtime
Background Map/Reduce Tasks
ROS Master node
Data Collectors/HDFS client (ROS Subscriber Nodes)
DAvinCi Server
DaVinci Robot Ecosystem (ROS Nodes)
Wi-Fi/ Zigbee

- UNR-PF (ongoing): Cloud networked robotics platform


Robotic Service
Robotic Service
F.USN-NRP
CityGML
OMG RoIS / RLS
Global Platform (service management, multi-area integration)
User Registry
Service Registry
Robot Registry
Map Registry
Local Platform (location B)
Local Platform (location A)
Component
Component
Component
Component
Component
Component
use of Drivers, ROS Nodes, Simulators
Physical Robot "visible-type"
Virtual Agent "virtual-type"
Embedded Sensor "unconscious-type"
Cloud Computing, Web Services

# Challenges

- Limited computation power on each robot
- Limited bandwidth and unreliable communication
- Much available computation power on the cloud, but not suitable for real-time processing