Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Enabling distributed, compute-intensive FaaS on the edge with COMPSs

Francesc Lordan

14/04/2019

Edge Computing Workshop@ASPLOS'19

Outline

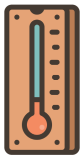
- Motivation
- Programming Model: COMPSs
- Decentralizing the COMPSs runtime
- Use Case: Cagliari Airport
- Conclusion

Motivation

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

IoT/Edge Platforms

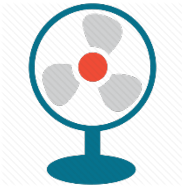
3 types of elements composing the system:

Events/Data Generators (sensors)

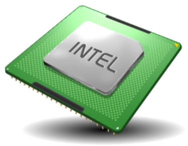
elements monitorizing certain condition and informing about it

- Continuosly generating information → Streams of data
- Eventually notifying a change → Events




Event-reactors/ Data Consumers (displays)

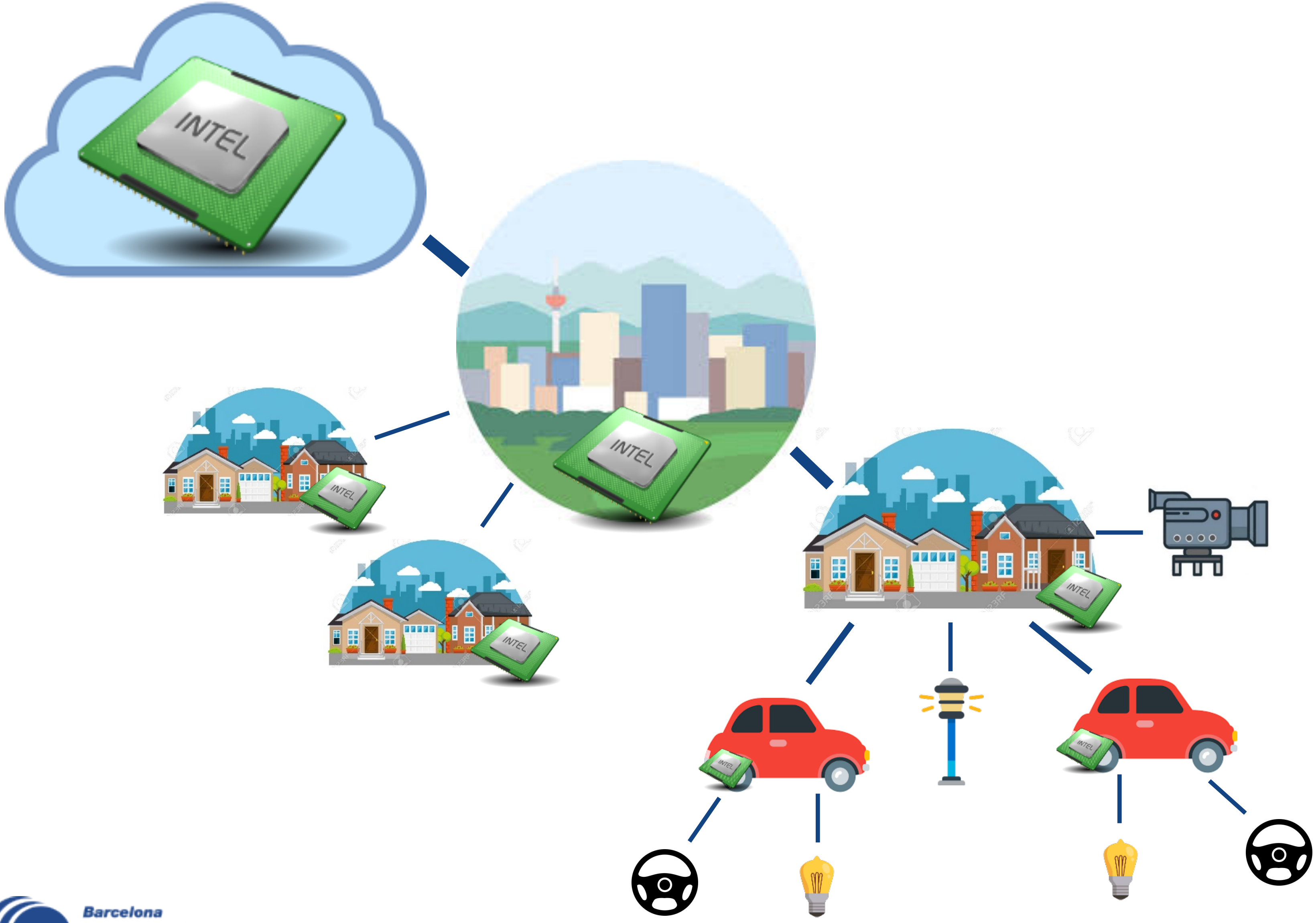
elements on which the platform realizes actions (changing its state)



Computing elements

elements within the infrastructure with ability to process and transform information

Cloud-Edge Continuum



Computation on IoT/Edge platforms

3 purposes for computation:

- Sense-process-actuate:

  A sensor detects something and triggers a computation to give a proper response to such event.

- Stream processing:

  A sensor continuously provides data that needs to be processed.

- Batch Jobs:

  Analyse big amounts of data collected by the sensors

Programming Model:
COMPSs

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

BSC vision on programming models

Simple Parallel Programming Model

- What do I need to compute?
- What data do I need to use?
- Provide Hints

Let the difficult parts to the runtime

Act on behalf of the user

Enable Monitoring and Analysis

Generate data to evaluate how application performs

Applications

Programming Model:
High-level, clean, abstract

Power to the runtime

Middleware API

Program logic independent of computing platform

General purpose
Task based
Single address space

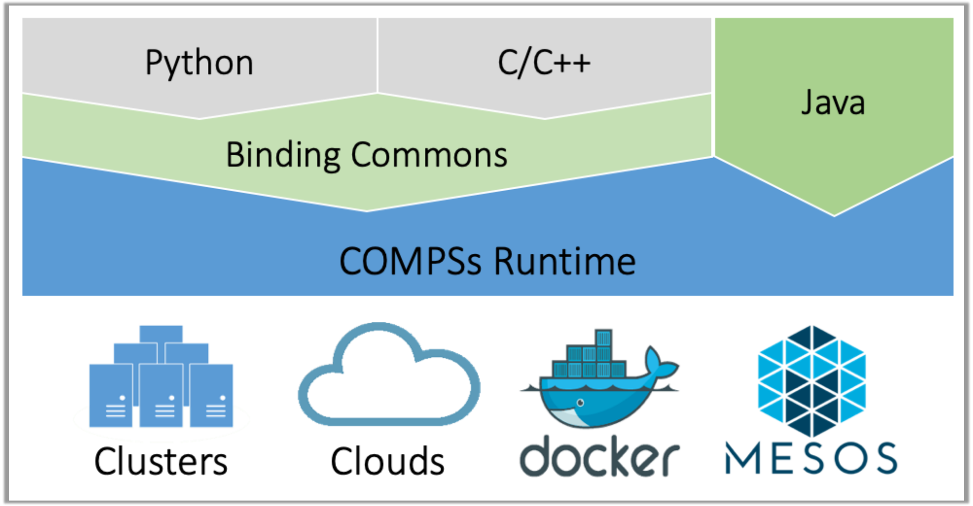
Intelligent runtime, parallelization, distribution, interoperability

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMP Superscalar (COMPSs)

- General purpose programming language + annotations/hints
- Sequential programming with no API calls
- Agnostic of the computing infrastructure
- Task-based: task is the unit of work

- Builds a task graph at runtime that express potential concurrency
  - Implicit workflow
- Exploitation of parallelism
  - … and of distant parallelism



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs Example - Java

Matmul.java

```java
public class Matmul {
    public static void main (String[] args) {
        int[][] A;
        int[][] B;
        ...
        int[][] C = multiply(A, B);
        ...
    }
}
```

Simple.java

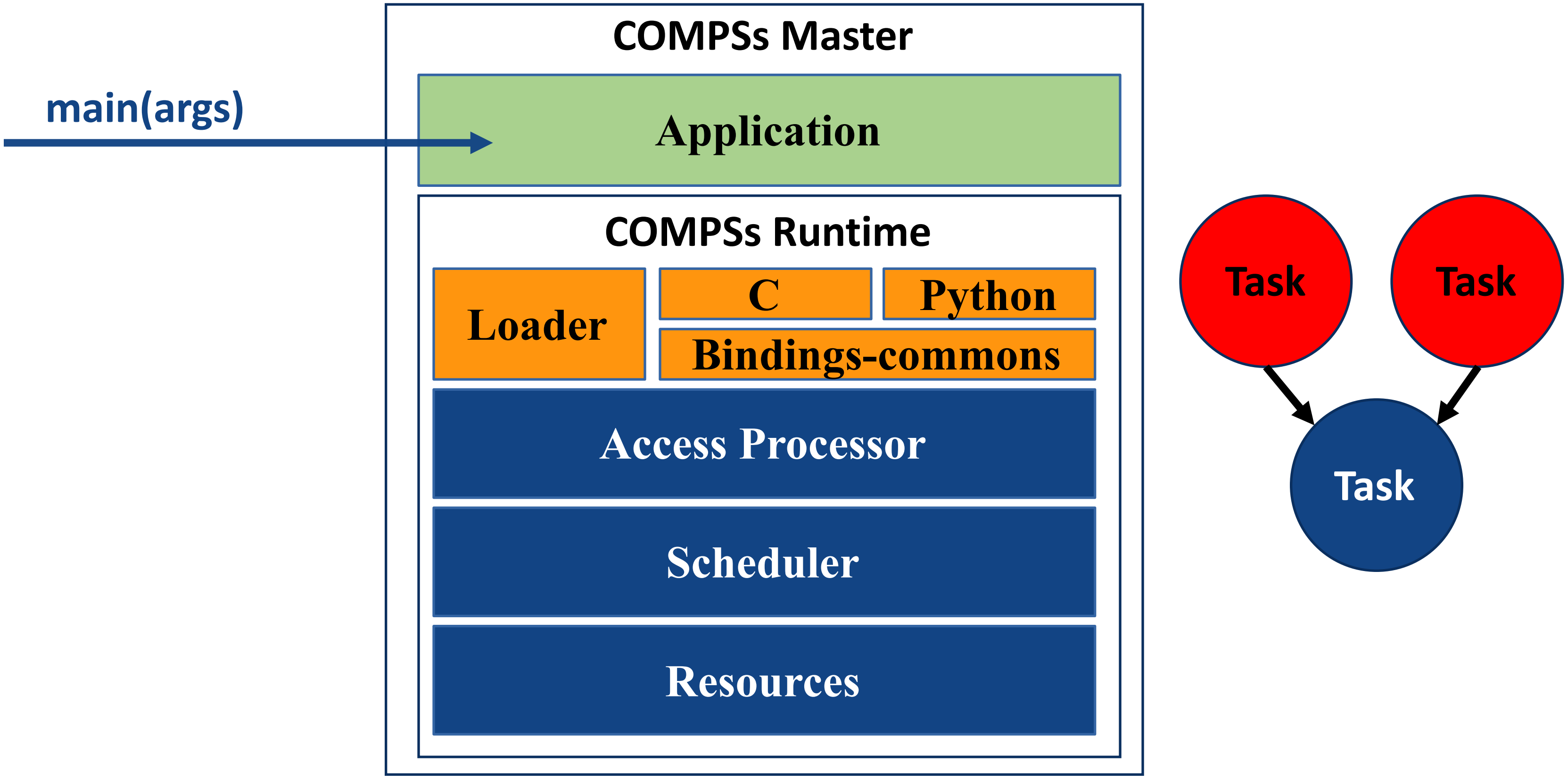
```java
public class Simple {
    public static int[][] multiply (
        int[][] A, int[][] B) {
        // Matrix multiplication code
        // C = AB
        ...
        return C;
    }
}
```

CEI.java

```java
public interface CEI {
    @Method(declaringClass = "Simple")
    public int[][] multiply (
        @Parameter(direction = IN) int[][] A,
        @Parameter(direction = IN) int[][] B
    );
}
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs runtime



Decentralizing the COMPSs runtime

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Environment differences

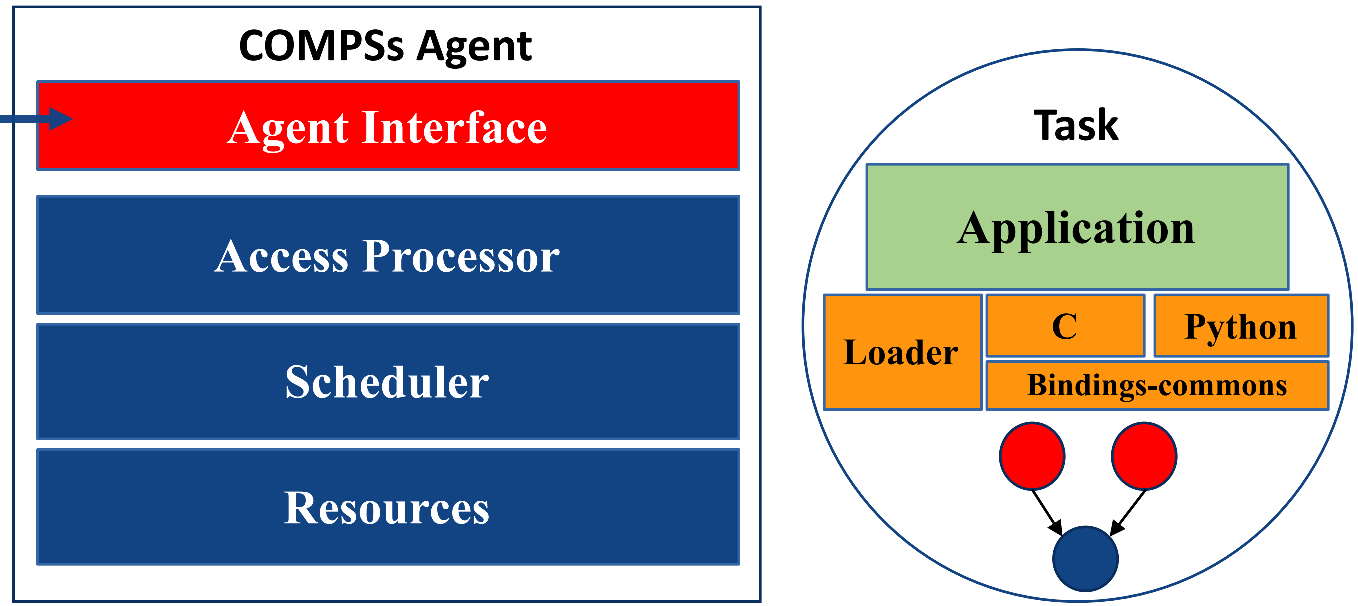
COMPSs' usual flow:

- 1 application
- Infrastructure is stable
- 1 node (MASTER):
  - Spawns tasks
  - Orchestrates the execution over the available resources (WORKERs)

COMPSs for Edge Computing

- N applications at the same time
- Infrastructure is dynamic
- Any agent may generate new tasks
- Execution orchestration is a shared responsibility

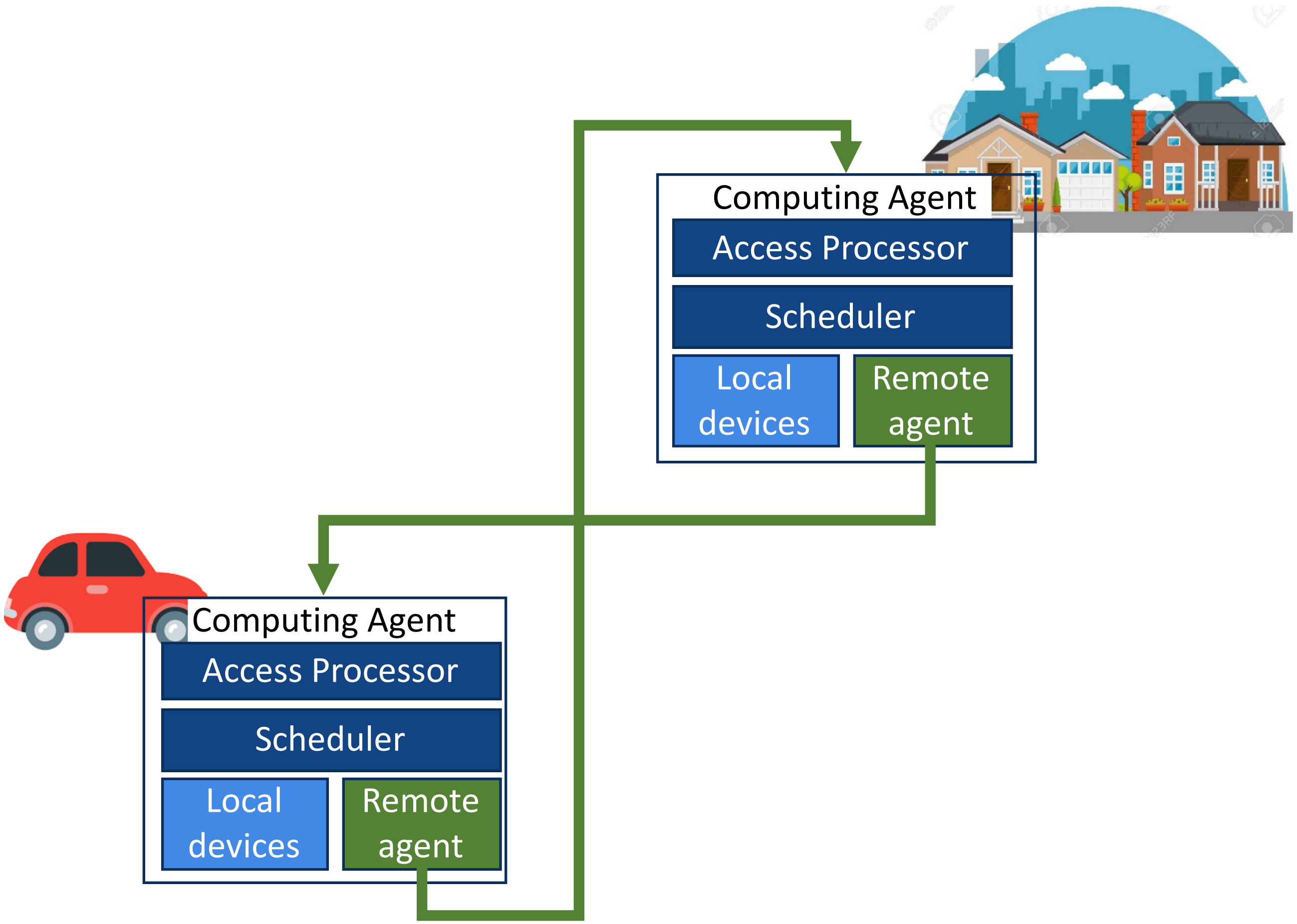
COMPSs agent



Start App Request

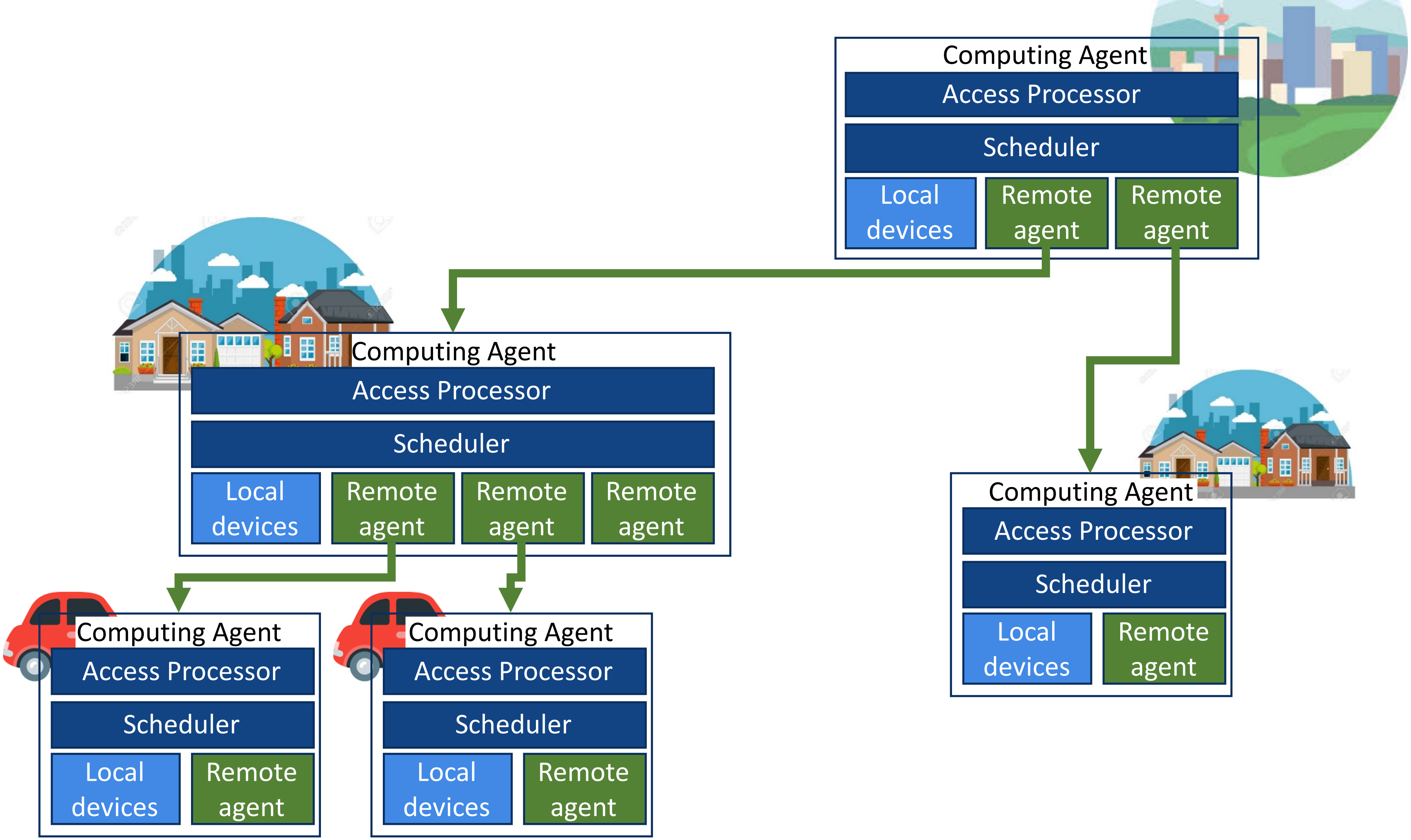
- Class
- Method
- Params
- CEI

Agents interaction



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Computing agents hierarchy



Use Case:
Cagliari (IT) airport

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Smart Fog-Hub Service

In 2017 more than 4 billion passengers concentrated in airports

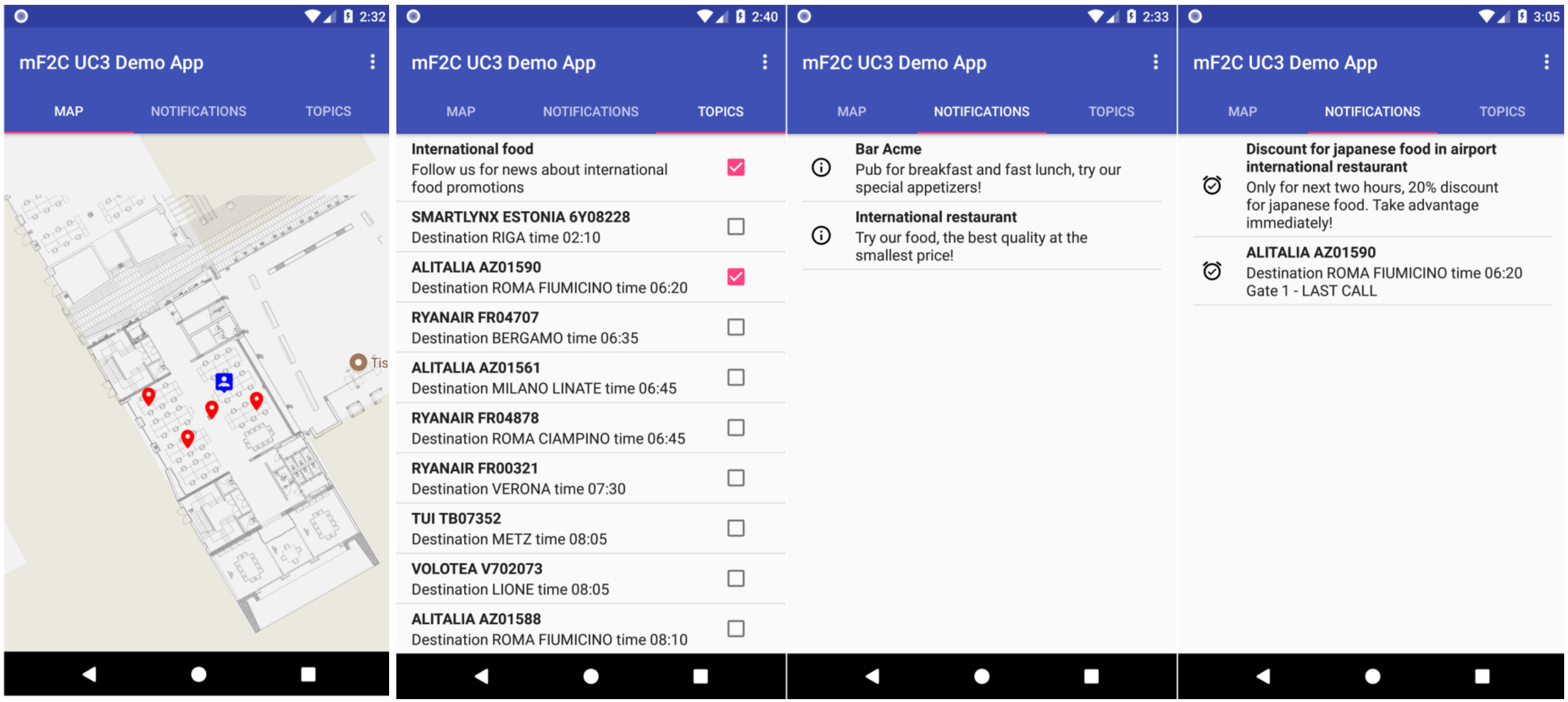
- Features
  - track the presence of people and objects in the field
  - proximity marketing services
  - suggestions on best use of airport services
  - Recommender system based on consumers' behavior

- Objectives
  - Reduce latency and response times
  - Capability to distribute computing in case of overloading
  - Pleasant experience for travelers while in the airport field

Customer Experience

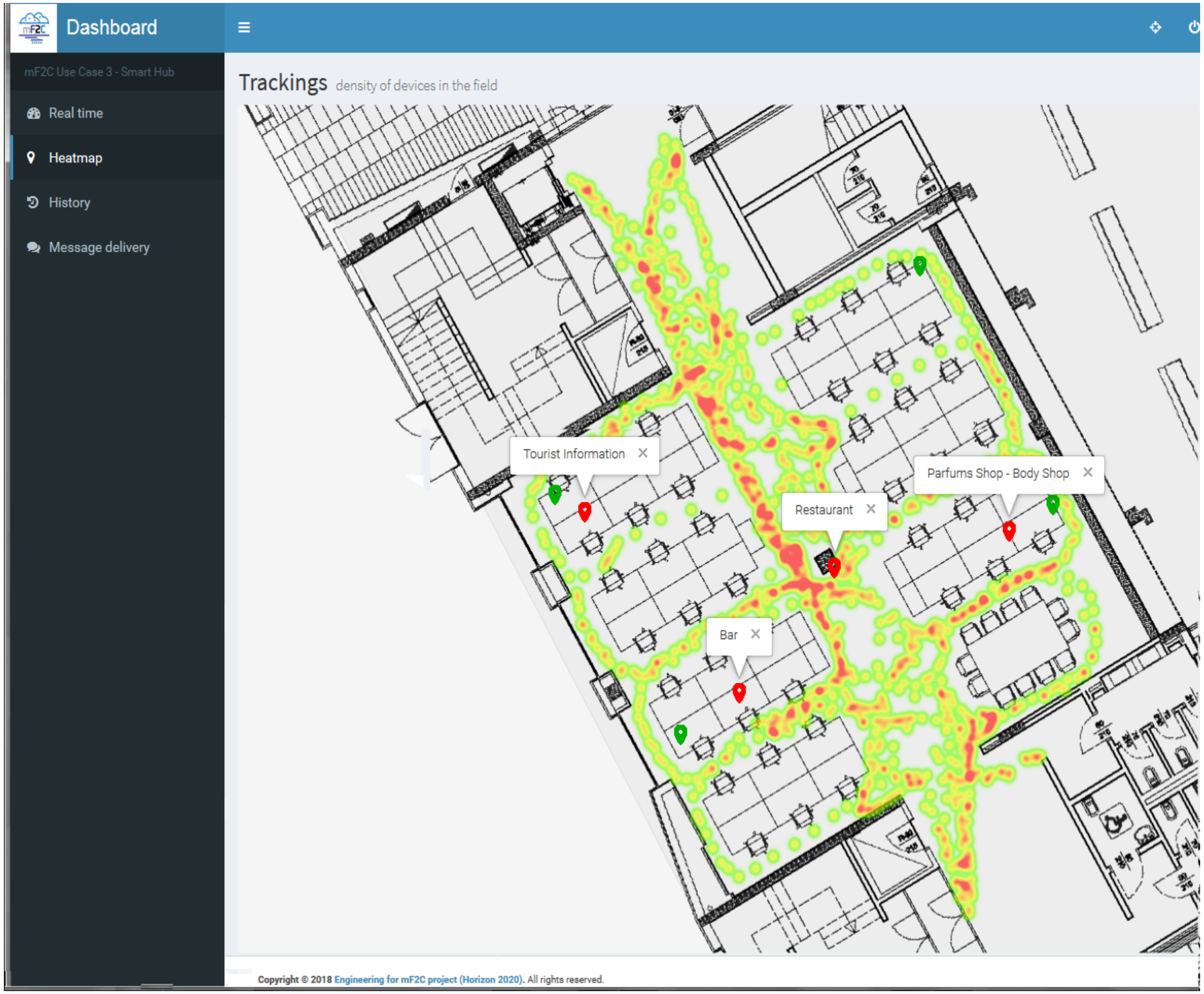
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Android App



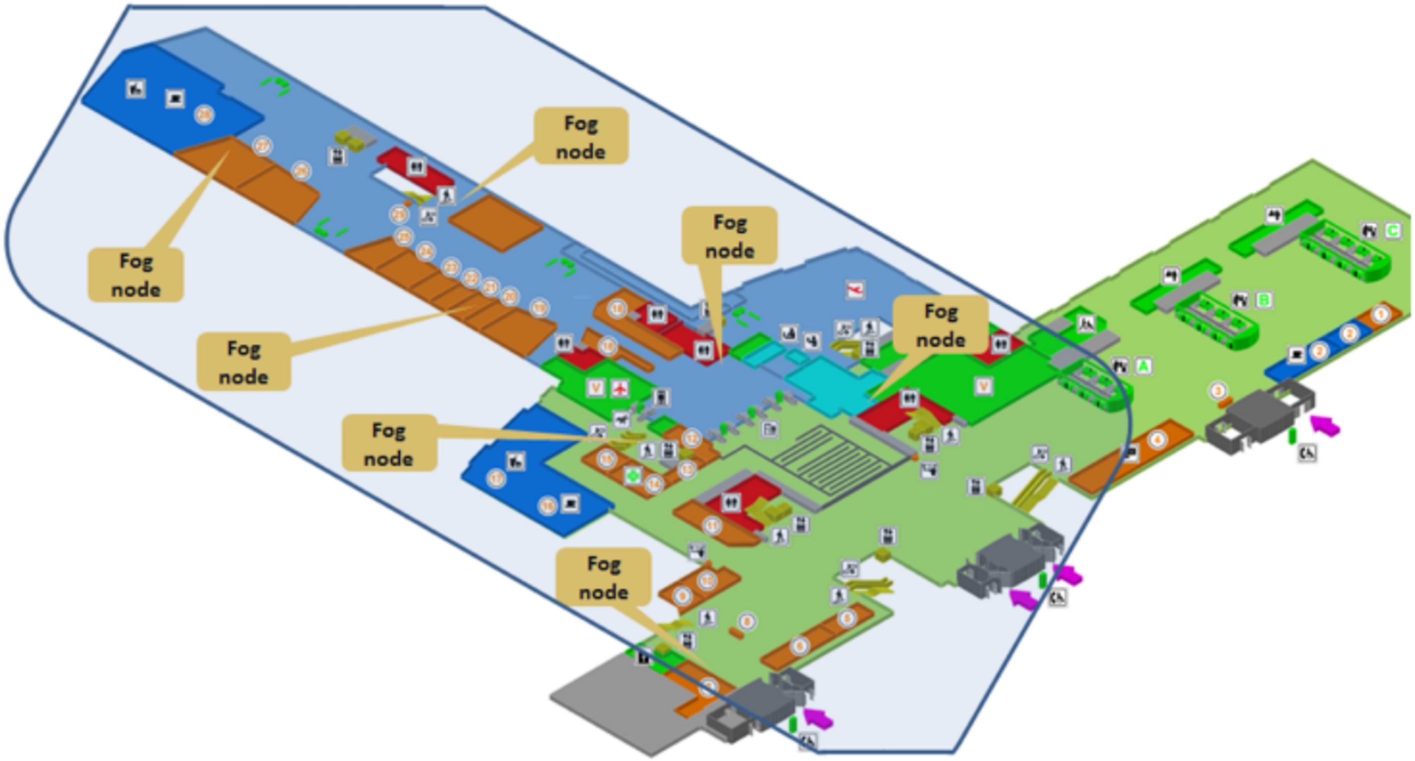
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Dashboard



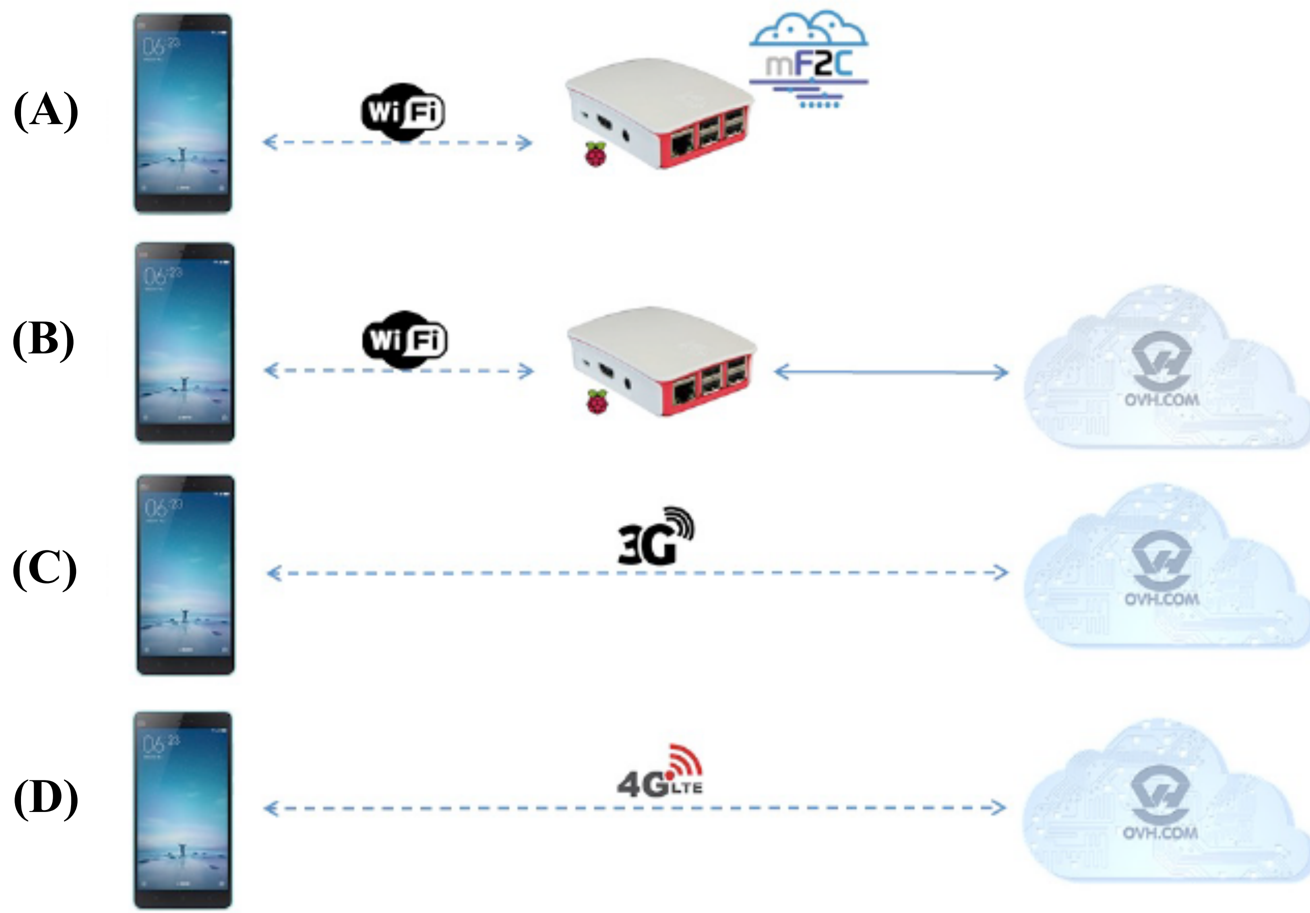
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Use Case deployment

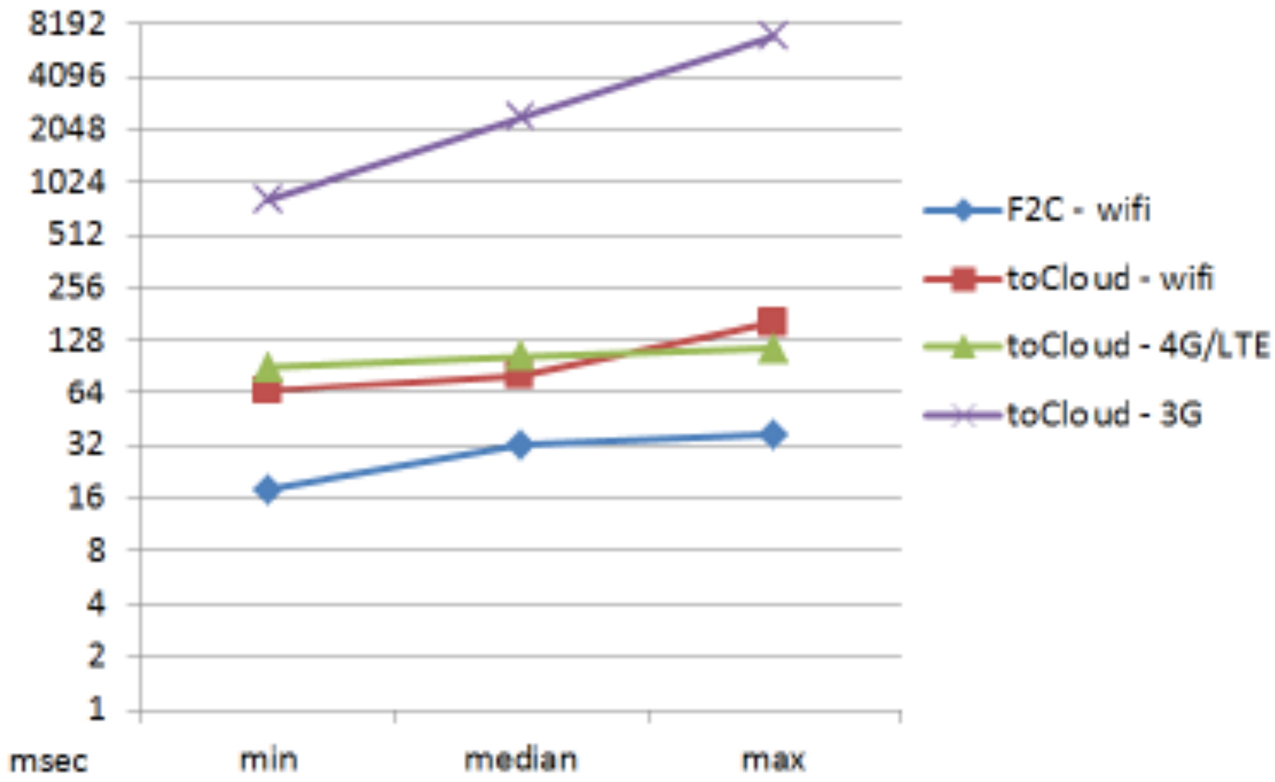


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Benchmarking scenarios



Test Results



Conclusions

Summary

- IoT/Edge infrastructures are composed of **autonomous** nodes with network and computing capabilities

- COMPSs
  - Developers code being unaware of the parallelism and infrastructure-related concerns
  - Detects tasks and the parallelism inherent in the application
  - Orchestrates the execution of these tasks on the available infrastructure
  - Supports computation on the three scenarios (batch, stream and sense-process-actuate)

- COMPSs agents
  - Allows devices to remain autonomous and compute in an isolated manner
  - By interacting with other agents, and agent has access to the available computing power

- The airport use case shows the viability and the benefits of the presented solution

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

THANK YOU!

francesc.lordan@bsc.es

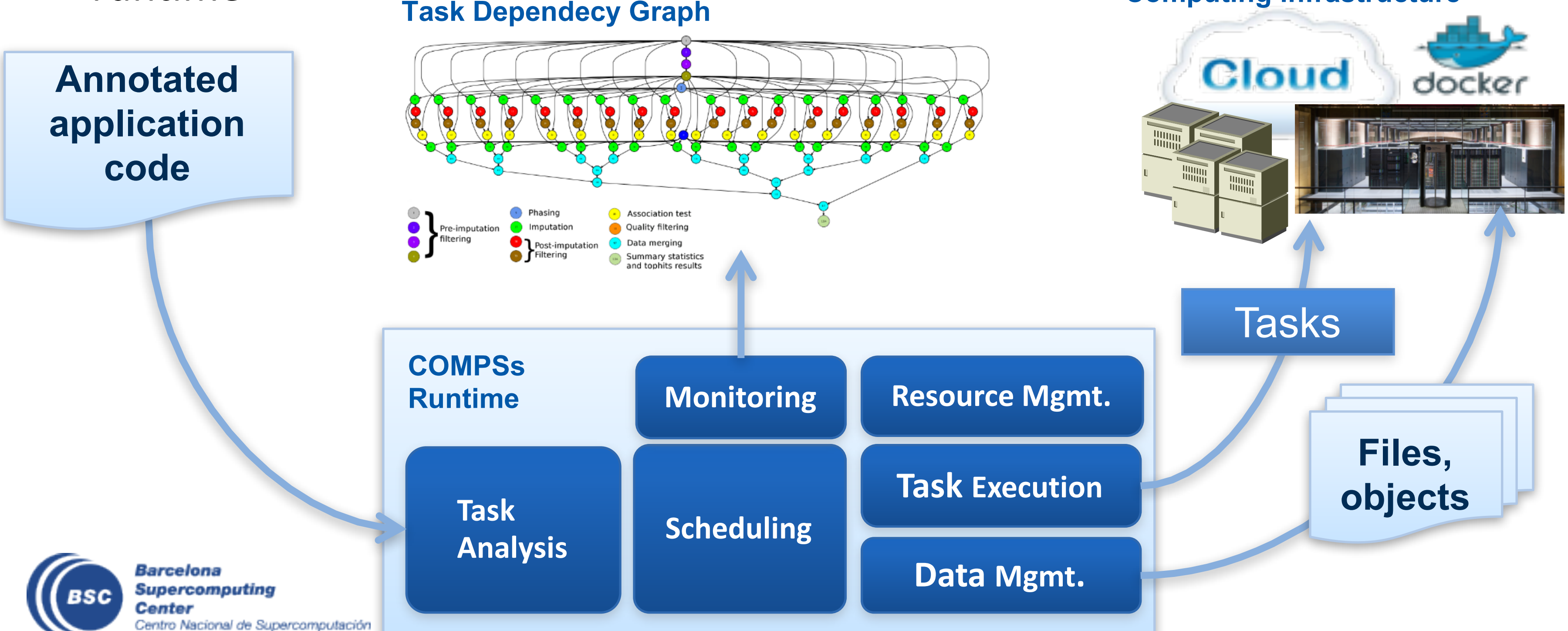
www.bsc.es

COMPSs

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs Runtime

- COMPSs applications executed in distributed mode following the master-worker paradigm
- Sequential execution starts in master node
- Tasks are offloaded to worker nodes
- All data scheduling decisions and data transfers are performed by the runtime

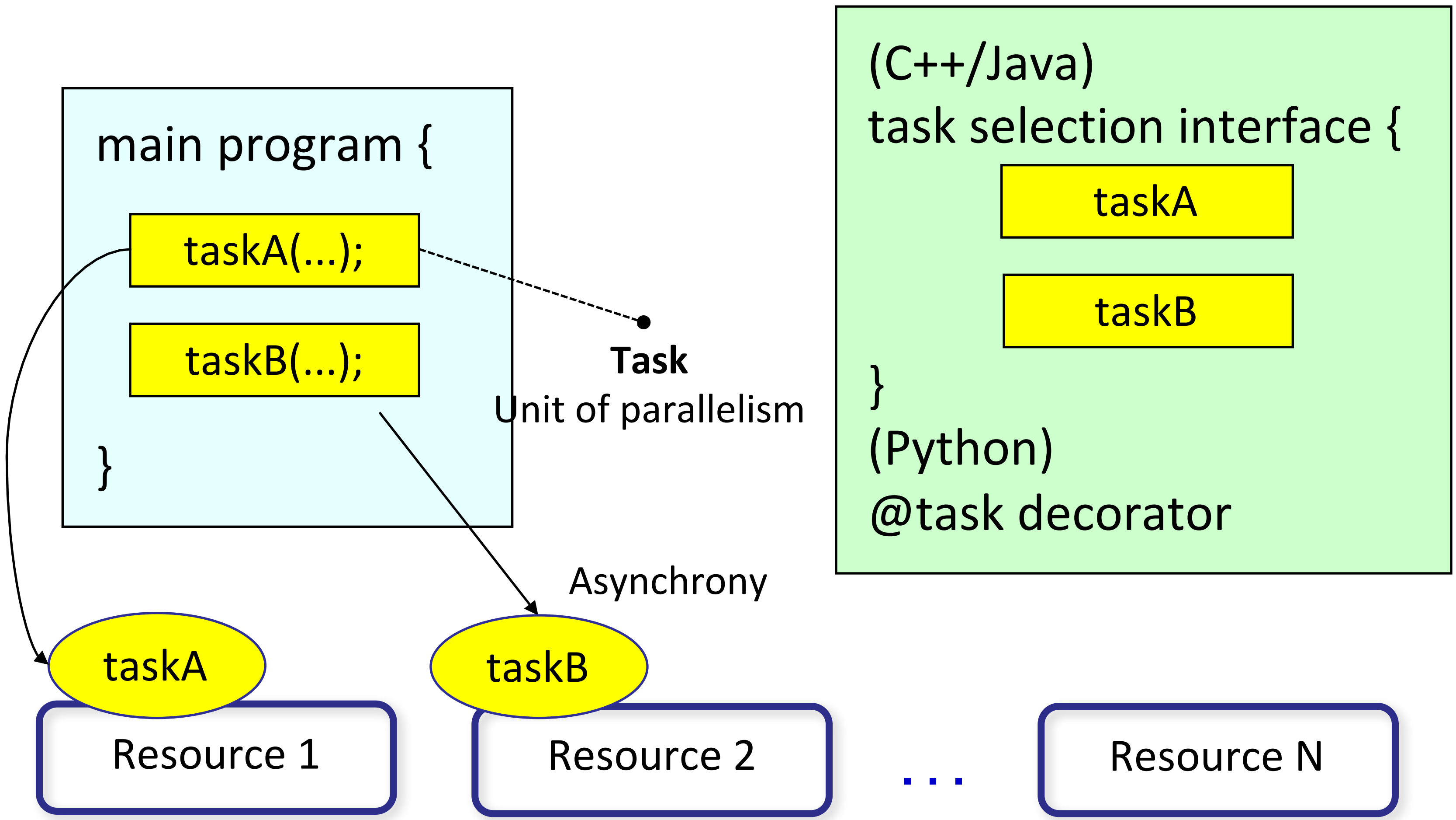


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Programming Steps

1. Identify tasks

2. Select tasks



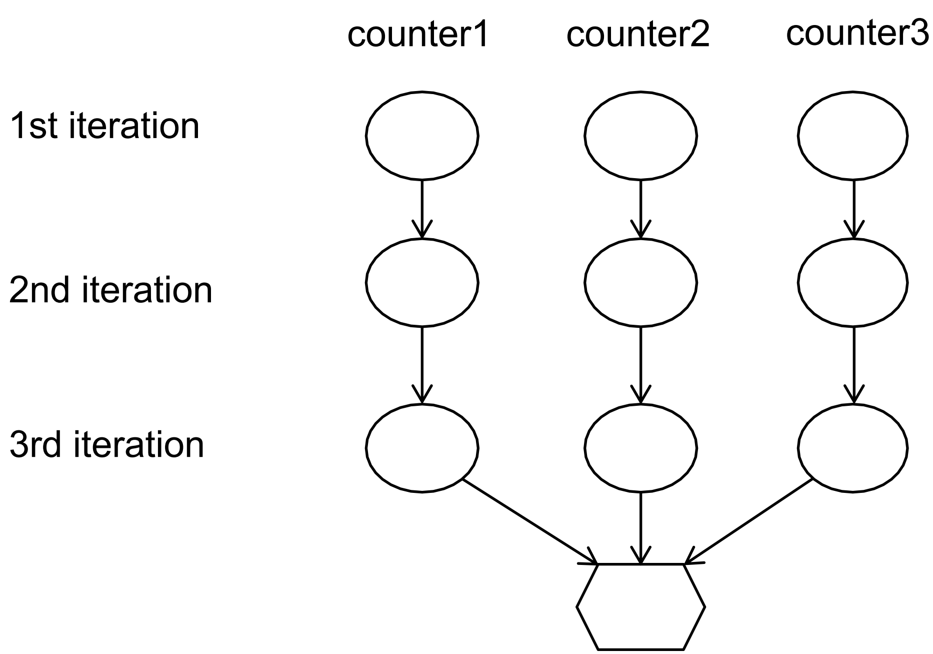
Programming Model Example

```
for (i = 0; i < 3; i++) {
        increment(counter1);
        increment(counter2);
        increment(counter3);
}
printCounters(counter1, counter2, counter3);
```

```
public interface SimpleItf {

    @Method(declaringClass = “SimpleImpl")
    void increment(
         @Parameter(type = FILE, direction = INOUT)
         String counterFile
    );

}
```

Implementation

Parameter metadata



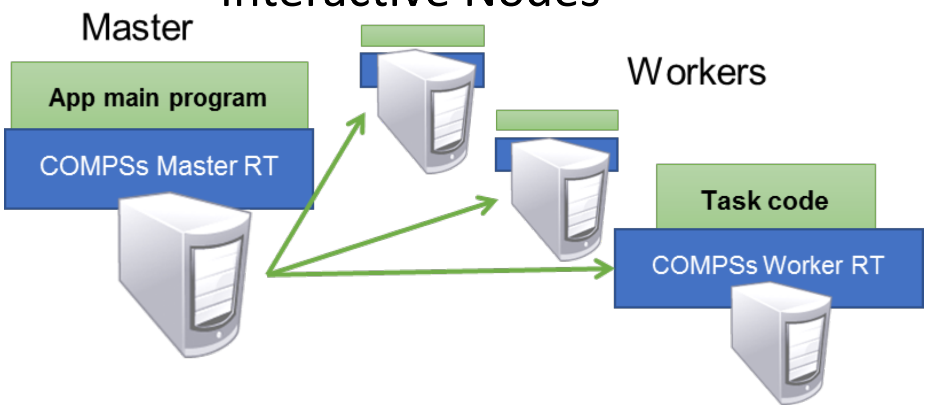
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Advanced Features

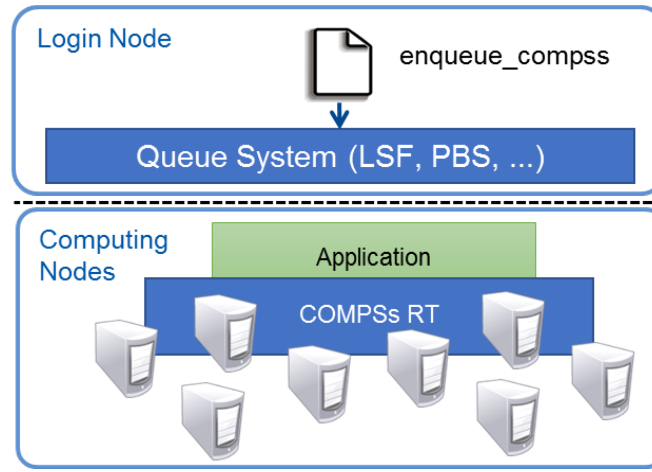
- Constraints to support heterogeneous tasks
  - @Constraints(…)
- Versioning
  - @Implements(…)
- Combination of binary execution
  - @Binary(…)
- Integration with Programming Models
  - @MPI(..) @Decaf(…)
  - COMPSs + OmpSs
- Nested
  - @COMPSs(…)

Execution Environments

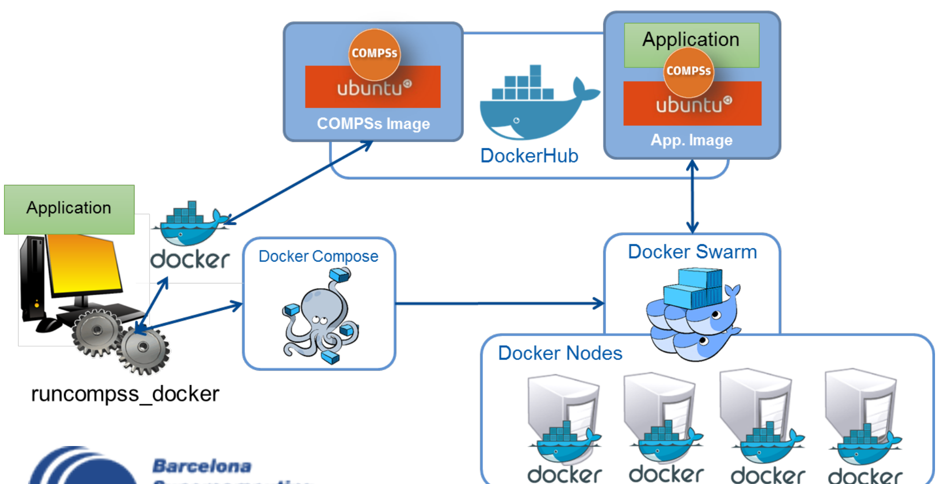
- Interactive Nodes



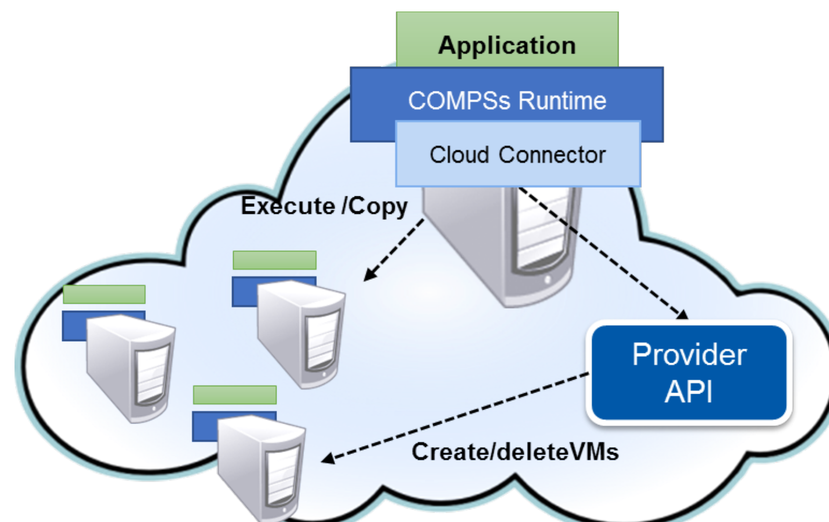
- Clusters



- Containers



- Clouds

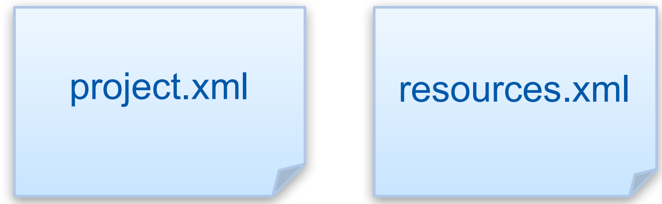


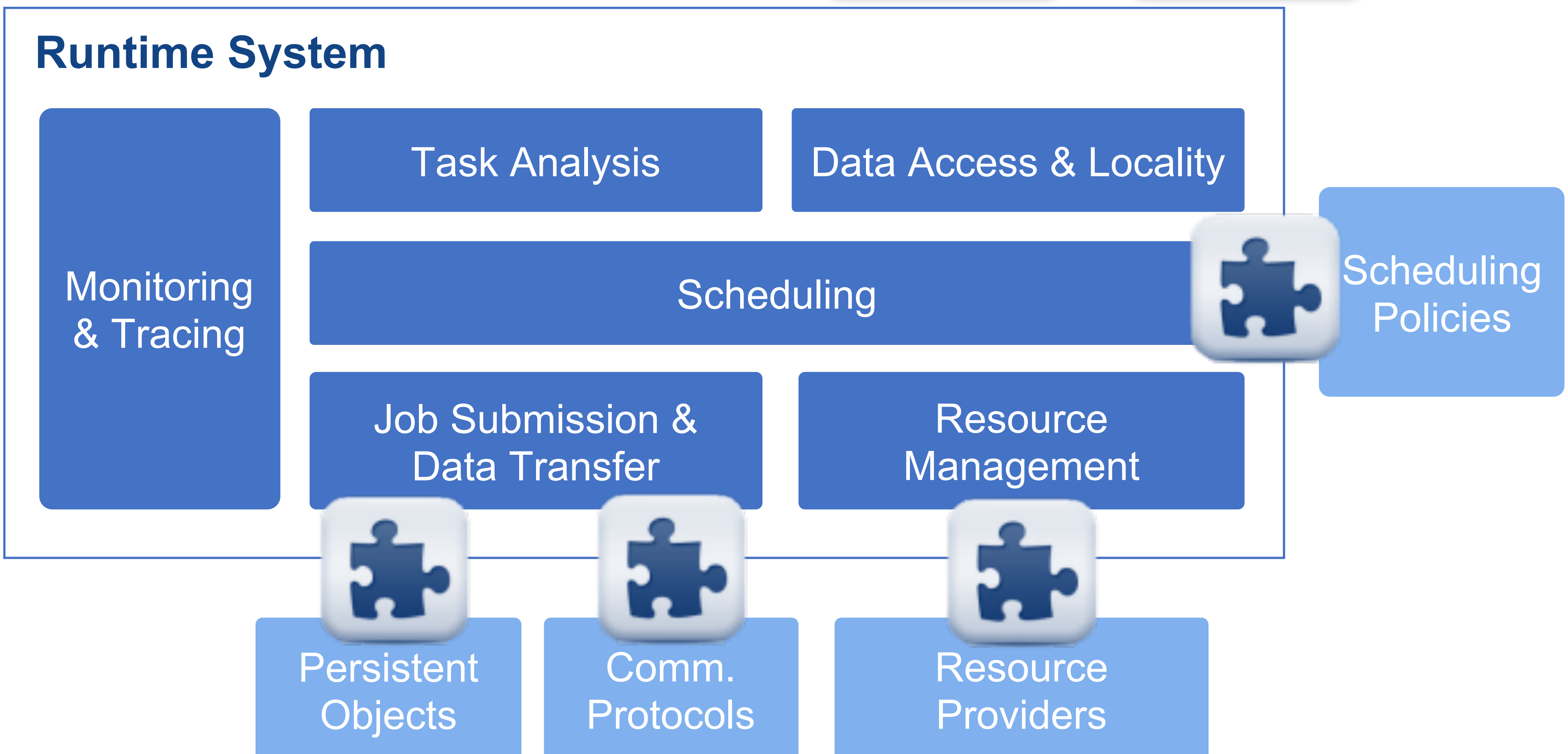
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Runtime Extensions

Execution commands:

- runcompss (interactive & cloud)
- enqueue_compss (clusters)
- Runcompss_docker (socker clusters)





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

mF2C

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Background – Open Fog

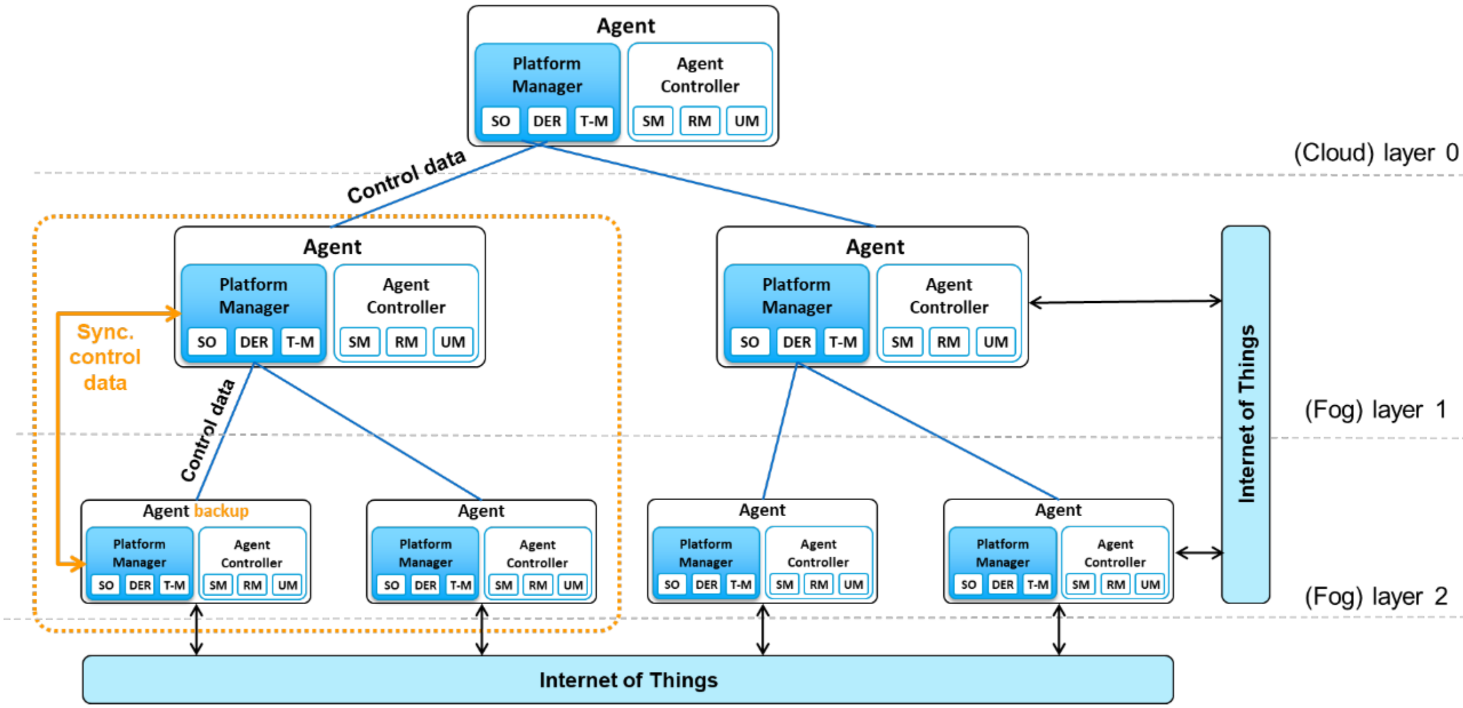
The architecture is an extension of the traditional cloud computing model

- Processes are moved from the cloud to the edge of the network, in Fog nodes
- Deployments can reside on multiple layers of a network topology,
- Deployments retain all the benefits of cloud computing, such as containerization, virtualization, orchestration, resource-efficient management

Fog Nodes peculiarities

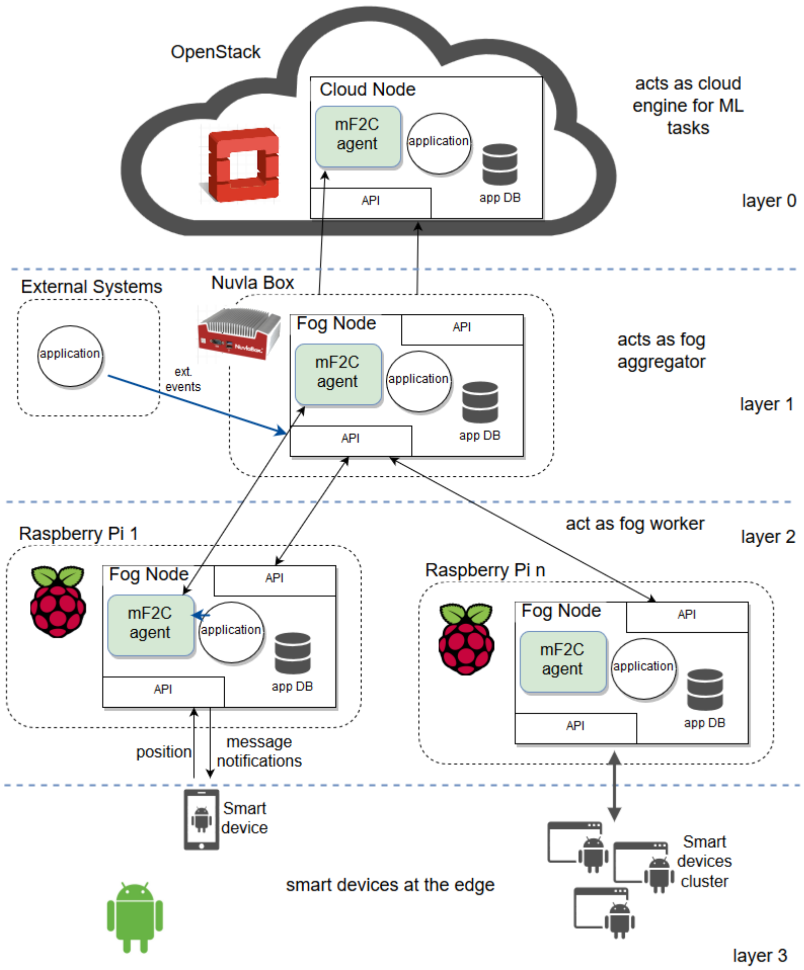
- Autonomous processing
- Local storage and IP communications
- Hosted in open (even hostile) fields
- Capable of acting in mobility

mF2C architecture



Use case architecture

- **Cloud** – based on a OpenStack instance, wired connected with the fog layers, providing scalable computing power for Machine Learning algorithms
- **Edge Fog** – with a fog aggregator based on Nuvlabox with 8GB, providing real-time computing and storage resources and 6 rPI with 1GB hub providing session management and fast response to the edge devices
- **Edge IoT** – Android smartphones connected to the edge nodes through wifi, and using an Android app to interact with the system



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Testbed

- Proximity processing
  - Client – call a request for a list of nearby POIs using geographic position
  - Server – calculates the POIs in proximity and returns a JSON array

- Client

  Smartphone XIAOMI with Android 5.0.2, running app calling a Rest HTTP API

- Server
  - A VM runs a dockerized image with proximity calculation

  1. rPI3 with 1GB RAM
  2. VM running on a public cloud (4-core processor and 4GB RAM)

Interaction with IoT devices

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Interaction with sensors

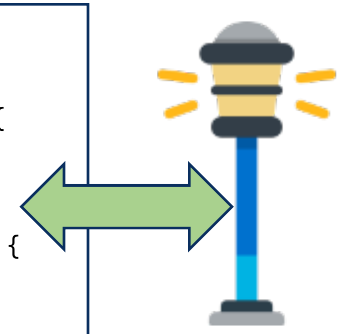
- Every sensor is attached to one device with computing capabilities
  - Events:
    the device trigger a function execution on any COMPSs agent
  - Streams:
    the device publishes the stream of data directly

Interaction with displays/devices

- Through a Controller class
  - Handles all the communication with the device
  - Offers a simple API to interact with the device
- Each instance of this controller class interacts with one device
- Device can be controlled by any agent if the object is transferred

```
public void task(Streetlamp sl) {

   …
   sl.on();
   …

}
```

```
class Streetlamp {

   public void on() {
      …
   }

   public void off() {
      …
   }
}
```



Interaction with displays/devices

- Using the storage framework
  - Controller class is a Persistent Object
  - One Controller instance is made persistent for each device using a universal ID
  - Any agent able can interact with any registered resource by using an ID

Device Manager

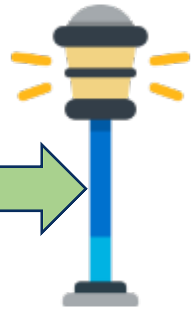
```
Streetlamp sl = new Streetlamp();
sl.makePersistent("light21758");
```

COMPSs task

```
public void task() {

   …
   Streetlamp sl;
   sl = Storage.getByAlias("light21758");
   sl.on();
   …

}
```

Storage Framework

```
class Streetlamp
  extends StorageObject {

   public void on() {
     …
   }

   public void off() {
     …
   }
}
```



Barcelona Supercomputing Center
Centro Nacional de Supercomputación