PerLa
PERvasive LAnguage

SHORT INTRODUCTION TO PerLa MIDDLEWARE

Guido Rota

Politecnico di Milano,
Dipartimento di Elettronica e Informazione,
Milano, Italy

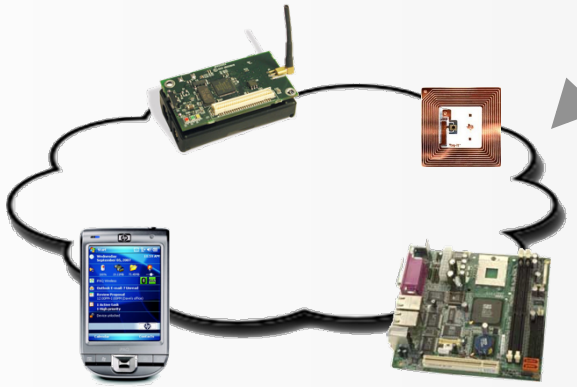
PerLa
PERvasive LAnguage

LOW LEVEL QUERIES (LLQ)

HIGH LEVEL QUERIES (HLQ)

ACTUATION QUERIES (AQ)

CONTEXT MANAGEMENT



FULLY DECLARATIVE SQL-LIKE HIGH LEVEL LANGUAGE

to query

PERVASIVE SYSTEMS

hiding the complexity of handling

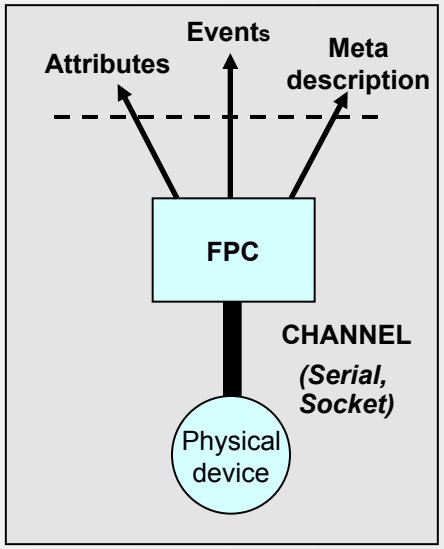
DIFFERENT TECHNOLOGIES

PerLa
PERvasive LAnguage

POLITECNICO MILANO

FPC ABSTRACTION

- The **LANGUAGE SEMANTICS** is defined on the concept of **Functionality Proxy Component (FPC)**
- Each device is abstracted as an FPC:
  - **ATTRIBUTES**
    (*id, temperature, pressure, power level, last sensed RFID reader, …*)
  - **EVENTS** (*last sensed RFID reader changed, …*)
  - **META-DESCRIPTION** (*name, data type, … for each attribute*)



THE LANGUGE: OVERVIEW

- LANGUAGE FEATURES
  - Data representation (FPC abstraction)
  - Physical device management
  - **FUNCTIONAL** characteristics
    - Raw data manipulation
    - Provide query results
    - Set sampling parameters
  - **NON-FUNCTIONAL** characteristics
    - Constraints on the functionality
    - QOS (mainly power management)
    - Determine the participation of a node to a query

LOW LEVEL QUERIES

- Define the behaviour of a single or of a group of devices abstracted by an FPC
  - Precise definition of **SAMPLING** operations
    - read attributes from a device
    - insert values into a temporary buffer (local buffer)
  - Perform simple **SQL OPERATIONS** (filtering, grouping, …)
    - on data in the local buffer
  - Insert records in the final data structure

QUERY EXAMPLE (1)

```
CREATE SNAPSHOT TrucksPositions (linkedBaseStationID ID) WITH
DURATION 1 h AS LOW:
    SELECT linkedBaseStationID
    SAMPLING
        EVERY 1 h
        WHERE is_in_CriticalZone(locationX, locationY)
    EXECUTE IF deviceType = "GPS"
```

```
CREATE OUTPUT STREAM Table (rfid STRING, counter INTEGER) AS LOW:
    EVERY 10 min
    SELECT lastReaderId, COUNT(*, 10 min)
    SAMPLING
        ON EVENT lastReaderChanged
    EXECUTE IF ID=[tag]
    TERMINATE AFTER 1 SELECTIONS
```

HIGH LEVEL QUERIES

- Perform complex SQL queries on data windows extracted from one or more input streams
  - **TIME DRIVEN**
  - **EVENT DRIVEN**
- Every record is time-stamped
- Similar to queries used in streaming DataBases

QUERY EXAMPLE (2)

```
CREATE OUTPUT STREAM LowPoweredDevices (sensorID ID) AS LOW:
    EVERY ONE
    SELECT ID
    SAMPLING EVERY 24 h
        WHERE powerLevel < 0.15
    EXECUTE IF deviceType = "WirelessNode"
```

```
CREATE OUTPUT STREAM NumberOfLowPoweredDevices (counter INTEGER) AS
HIGH:
    EVERY 24 h
    SELECT COUNT(*)
    FROM LowPoweredDevices(24 h)
```

PerLa
PERvasive LAnguage

PerLa MIDDLEWARE OVERVIEW

PerLa MIDDLEWARE GOALS (1)

- The main goals of the middleware:

  1. provide an **ABSTRACTION** of all the devices connected to the system

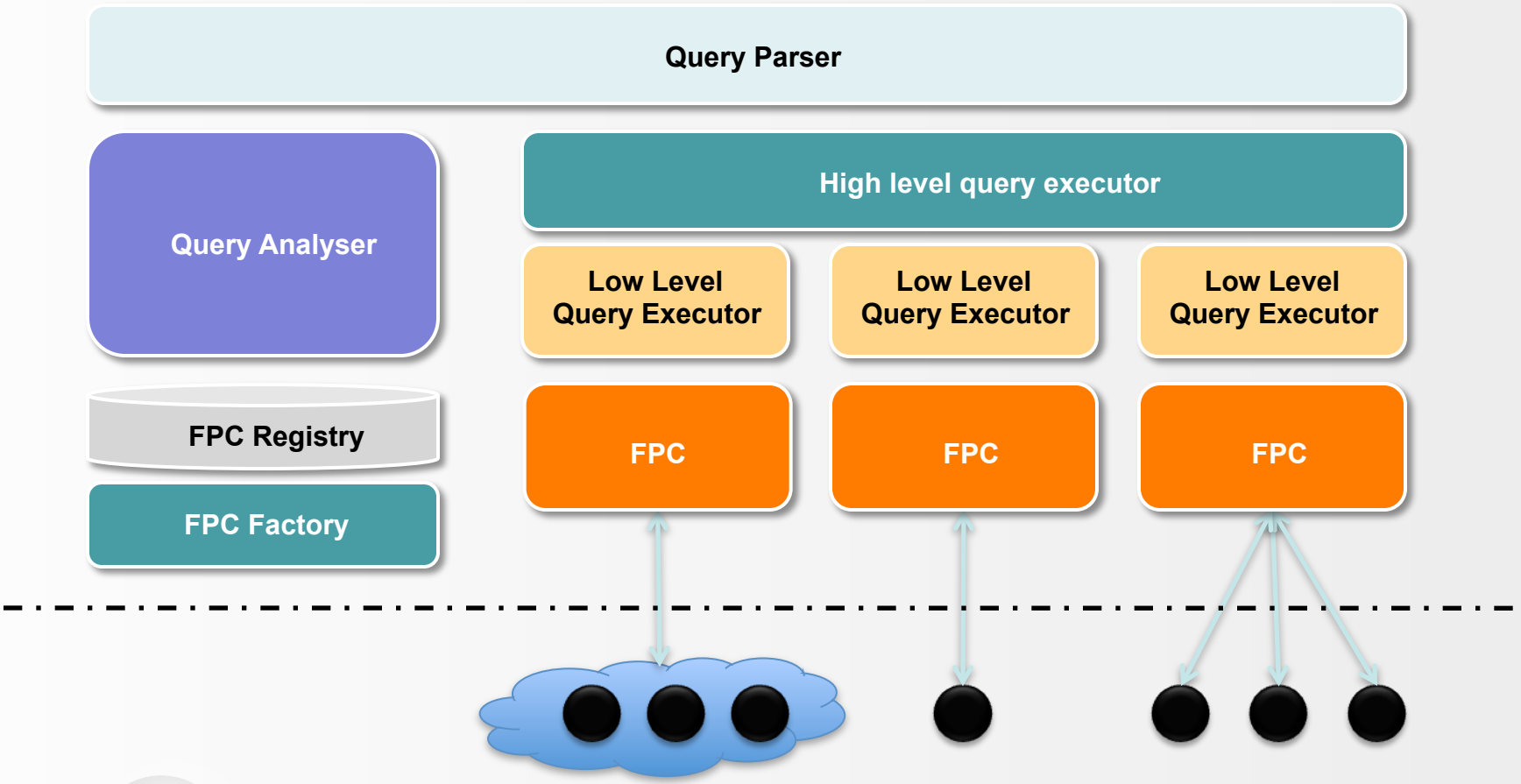
  2. support the **EXECUTION OF PERLA QUERIES**

  3. allow devices to automatically start query execution immediately after power-on (**PLUG & PLAY**)

  4. ease the **DEFINITION** and the **ADDITION** of new devices and technologies – **CODE-FREE** device introduction

PerLa MIDDLEWARE GOALS (2)

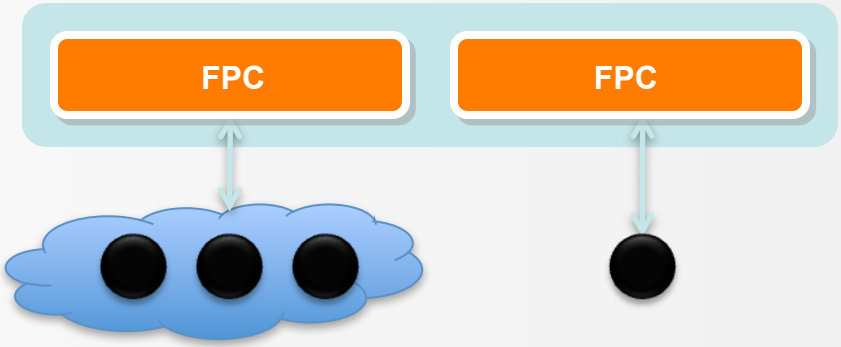
- Comprises all the software components needed to achieve the aforementioned goals



PerLa MIDDLEWARE GOALS (3)

1. provide an **ABSTRACTION** for all the devices connected to the system

- The **Functionality Proxy Component (FPC)** is used to provide this abstraction
- A single FPC works as a proxy for **ONE OR MORE** physical devices

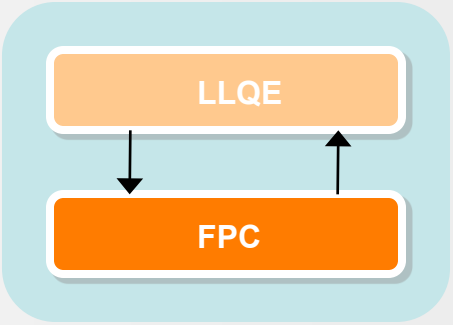


- By means of the FPC, **MANY HETEROGENEOUS DEVICES** can be accessed through the **SAME INTERFACE**

MIDDLEWARE GOALS (4)

2. support the **EXECUTION OF PERLA QUERIES**

- The **LLQE (Low Level Queries Executor)** is a Java component placed on top of the *FPC*.
- Retrieves data from the an *FPC* and to computes **QUERY RESULTS**.

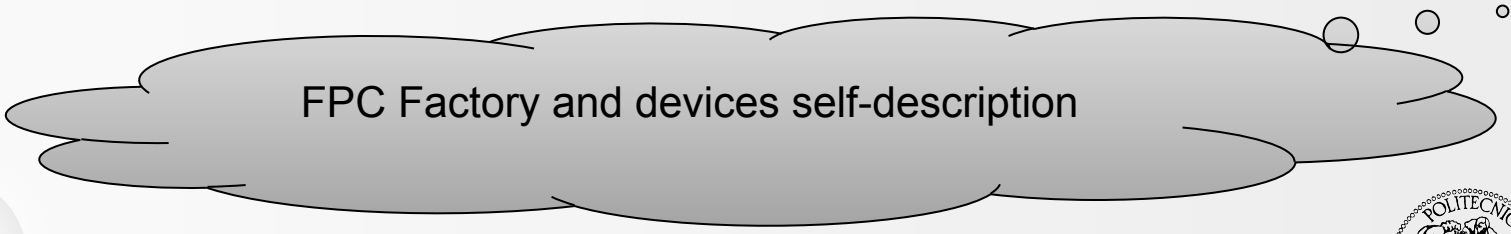


D. Viganò – Low Level Query Executor

MIDDLEWARE GOALS (5)

3. allow devices to automatically start query execution immediately after power-on (**PLUG & PLAY**)

- A **PLUG & PLAY** behavior at device start-up requires that:
  - The device **ESTABLISHES A CONNECTION** with the PerLa middleware
  - An *FPC* object, tailored to specifically work with the device, is **DYNAMICALLY GENERATED** and instantiated
  - The generated *FPC* is **REGISTERED** in the *FPC* Registry, enabling the device to be used at query-time



MIDDLEWARE GOALS (6)

4. ease the **DEFINITION** and the **ADDITION** of new devices and technologies – **CODE-FREE** device introduction

- A device **SELF-DESCRIPTION** file is introduced to allow a complete automatic generation of the Java *FPC*
- A low level **C LIBRARY** is provided in order to reduce the amount of *C* code needed to manage a node based on a new technology
  - **HLD**: middleware-provided *C* code (FPC-device communication, timer multiplexing, sampling scheduling, …)
  - **LLD**: user-written *C* code (low level sampling routines, communication channel initialization, …)

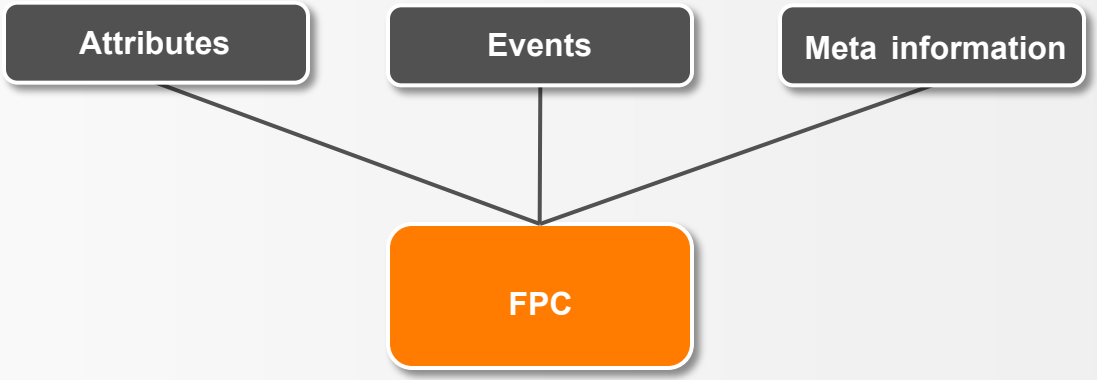
PerLa

PERvasive LAnguage

The FPC and the Device

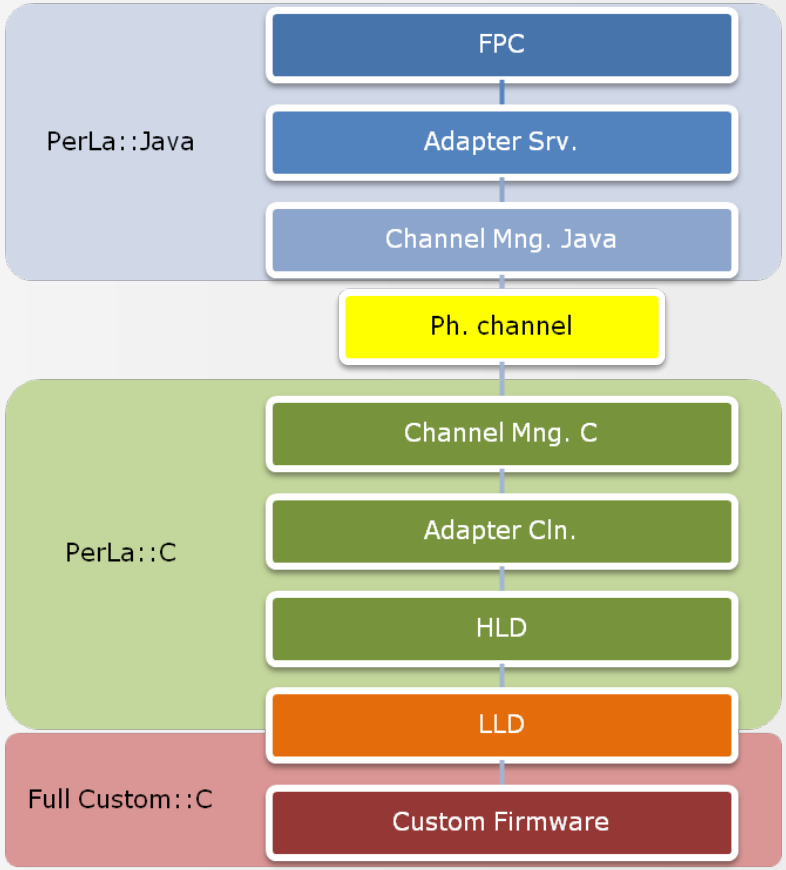
FPC Features (1)

- The **FPC** is a **HIGH-LEVEL ABSTRACTION** of the physical devices which provides:
  - Methods to enumerate the attributes and events of a node
  - An interface to set or retrieve attributes on the device
  - Notifications to the system in response to events sensed by the devices



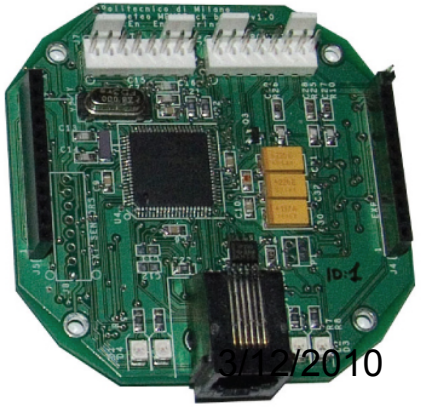
FPC Features (2)

- The **FPC** is a **JAVA OBJECT** that acts as a proxy between the physical device and the rest of the middleware
  - The communication between the Low Level Query Executor and the hardware devices are mediated by the *FPC*
  - Every *FPC* is tailored to fit a single sensor (or a group of them)
  - The system is provided with a *FPC* Factory, which automatically assembles *FPCs* on behalf of the user



BOUNDARIES OF THE MIDDLEWARE: HLD and LLD

- PerLa Middleware **C PORTABLE LIBRARY** (HLD, High Level Driver)
  - Communications with other PerLa Middleware components
  - General device management components (Timers, signal handling, Input/Output management…)
- The user is just required to write the missing software needed to access his device's hardware (LLD, Low Level Driver)
  - Standard library of device drivers (CAN bus, Digital-IO, …)



DEVICE DESCRIPTOR (1)

- The device descriptor is an XML file that describes **CAPABILITIES** and **FEATURES** of a device

- Main sections of the descriptor:
  - Attributes and events of the device
  - Device features (available memory, uptime, available commands)
  - Network features (contact method, uptime, address format)
  - Message format expected by the *LLD*

- Most of the boilerplate code needed to handle a device is generated at run-time using the information contained in the Device Descriptor

DEVICE DESCRIPTOR (2)

- A (very) simple Device Descriptor

C-like message structure

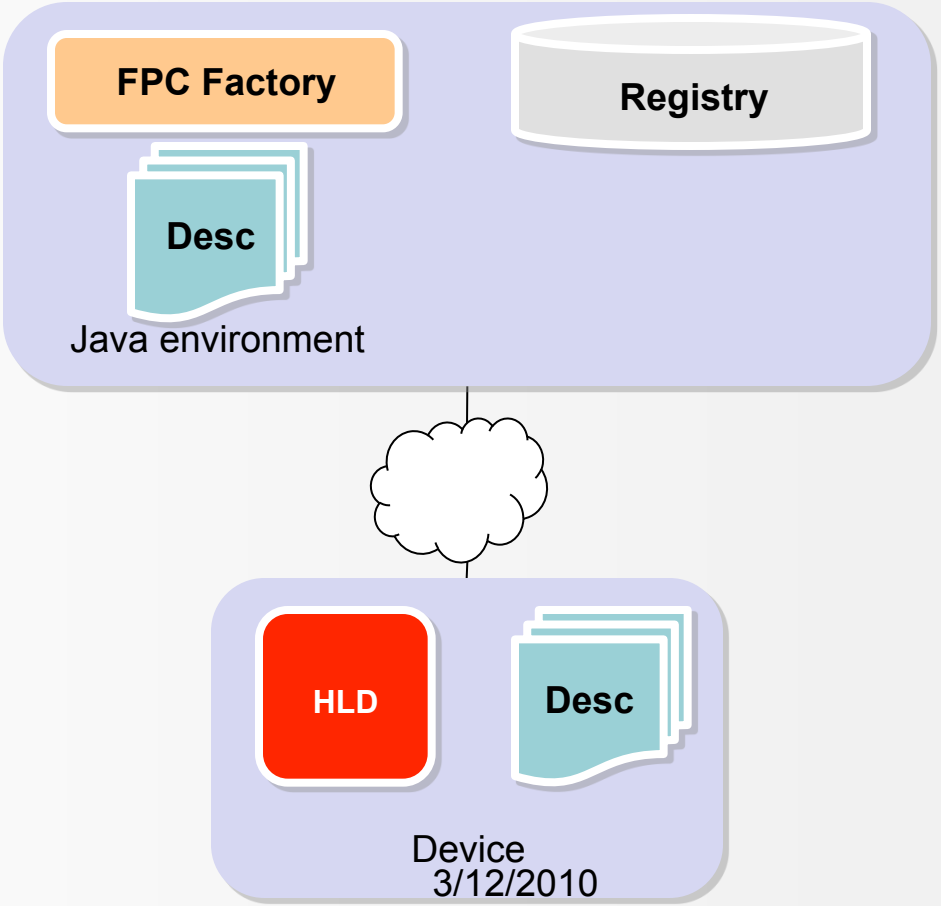
```
typedef struct{
   uint64_t timestamp;
   int16_t panelVoltage;
   int16_t panelCurrent;
   int16_t batteryVoltage;
   uint8_t flags;
} DataMessage;
```

XML Descriptor

```
<parameterStructure name="DataMessage”>
  <endianess>BigEndian</endianess>
    <parameterElement name="timestamp">
      <length>8</length>
      <type nameType="long">
        <sign>unsigned</sign>
      </type>
      <attributeType>probing</attributeType>
      <permission>r</permission>
      <continuousValue />
    </parameterElement>
    <parameterElement name="panelVoltage">
      <length>2</length>
      <type nameType="long">
        <sign>signed</sign>
      </type>
      <attributeType>probing</attributeType>
      <permission>r</permission>
      <continuousValue />
    </parameterElement>
    ...
</parameterStructure>
```

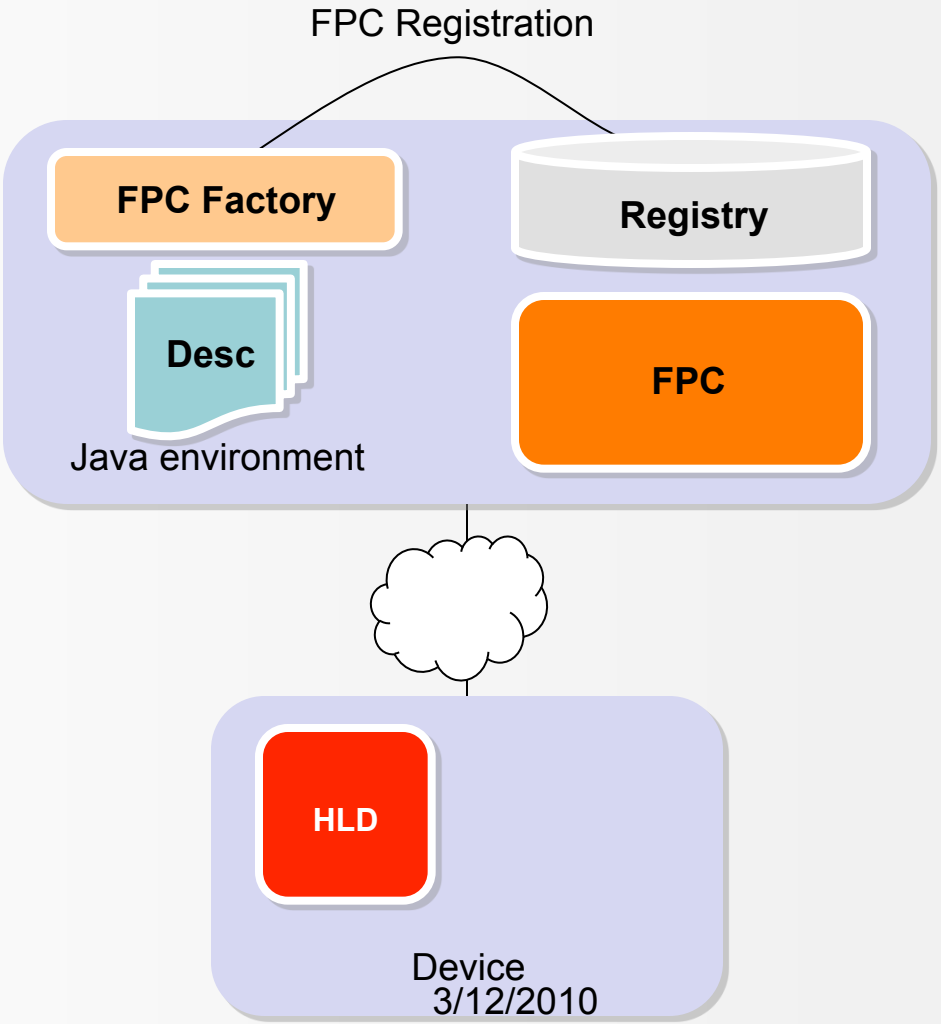
FACTORY AND FPCs ASSEMBLING (1)

- Device Binding process: **PLUG & PLAY**
- Once started up the *HLD* sends the **DEVICE DESCRIPTOR** towards the nearest Java device suitable to host a *FPC*



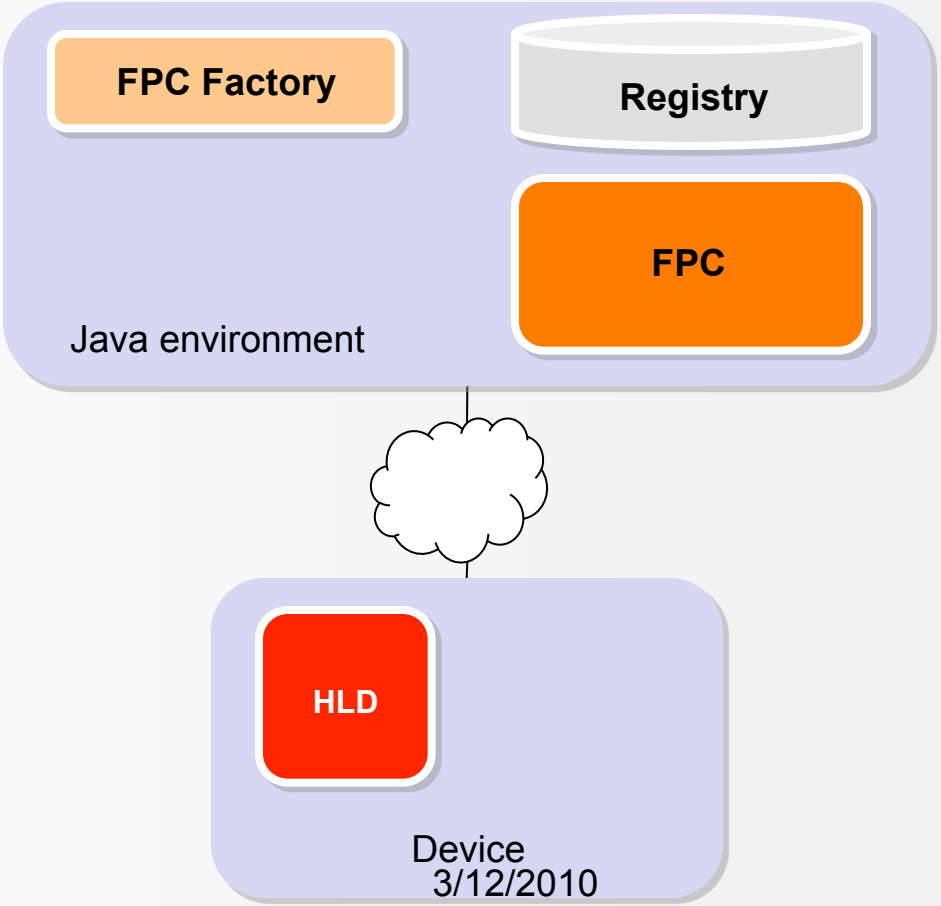
FACTORY AND FPCs ASSEMBLING (2)

- The FPC Factory assembles the *FPC* and registers it in the Registry



FACTORY AND FPCs ASSEMBLING (3)

- The newly created *FPC* acknowledges the device of its creation
- The device is ready to be used



PerLa
PERvasive LAnguage

POLITECNICO MILANO

FACTORY AND FPCs ASSEMBLING (4)

- The *FPC* is **CREATED AD-HOC** for every physical device
  - The Factory, after parsing the Descriptor, combines different modules and configures them
  - New modules can be added if not present in the default library
  - Data structures are dinamically created on the fly

- The nodes don’t have to comply with our rules, we comply with theirs!

FACTORY AND FPCs ASSEMBLING (5)

- Data structure creation

XML Descriptor

```
<parameterStructure name="DataMessage”>
  <endianess>BigEndian</endianess>
    <parameterElement name="timestamp">
      <length>8</length>
      <type nameType="long">
        <sign>unsigned</sign>
      </type>
      <attributeType>probing</attributeType>
      <permission>r</permission>
      <continuousValue />
    </parameterElement>
    <parameterElement name="panelVoltage">
      <length>2</length>
      <type nameType="long">
        <sign>signed</sign>
      </type>
      <attributeType>probing</attributeType>
      <permission>r</permission>
      <continuousValue />
    </parameterElement>
    ...
</parameterStructure>
```

Java Class

```
public class DataMessage{
   ...
   @SimpleField(size = 8, sign = Sign.UNSIGNED)
   private long timestamp;

   @SimpleField(size = 2, sign = Sign.SIGNED)
   private int panelVoltage;

   @SimpleField(size = 2, sign = Sign.SIGNED)
   private int panelCurrent;

   @SimpleField(size = 2, sign = Sign.SIGNED)
   private int batteryVoltage;

   @SimpleField(size = 1, sign = Sign.UNSIGNED)
   private int flags;

   ...
}
```

OPEN POINTS

- There is currenlty an implementation of the FPC Factory, but needs to be expanded

PerLa

PERvasive LAnguage

Low Level Query Executor

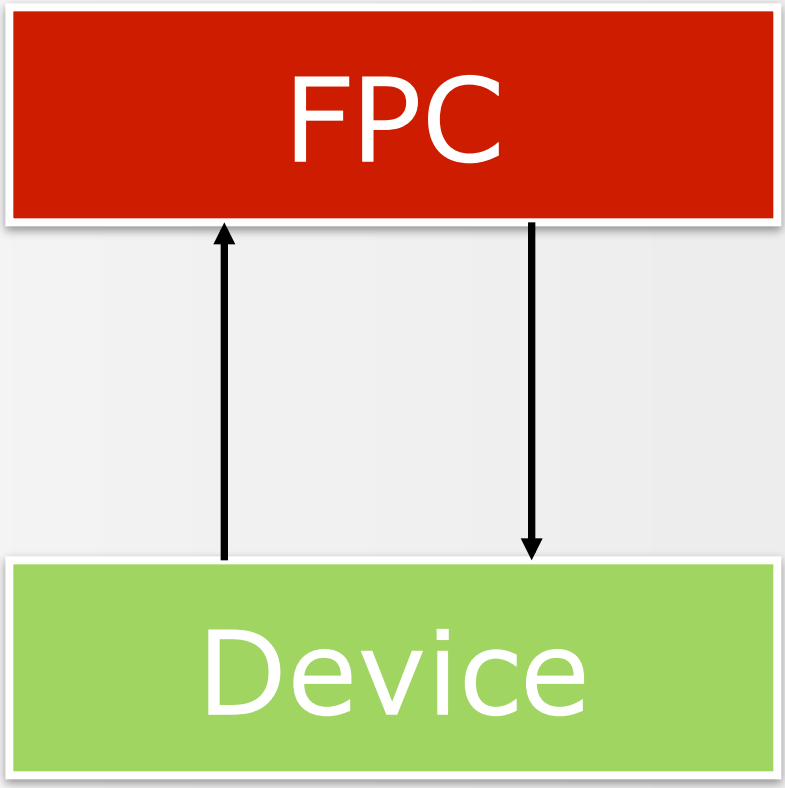
LOW LEVEL QUERY EXECUTION

-summary-

- What and why
  - **GOALS, FUNCTIONALITIES, POSITION and STRUCTURE**
- How
  - **...THE WORK GETS DONE**
- Open points
  - **FUTURE DEVELOPMENTS**

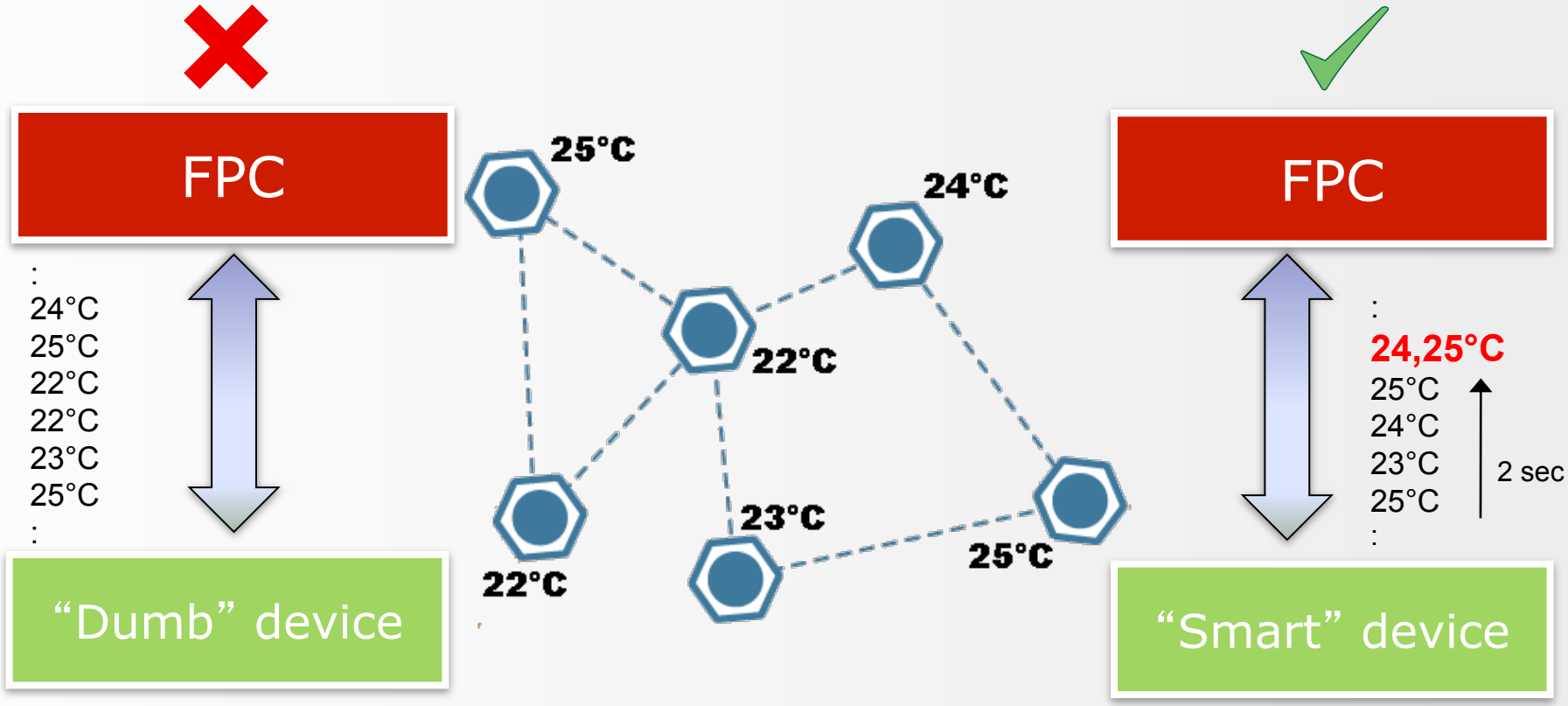
WHAT: Goals and functionalities

- **FPC** is the key component charged of dealing with the multitude of devices...
- ...but its tasks don't include data processing. Let's see it in an example!



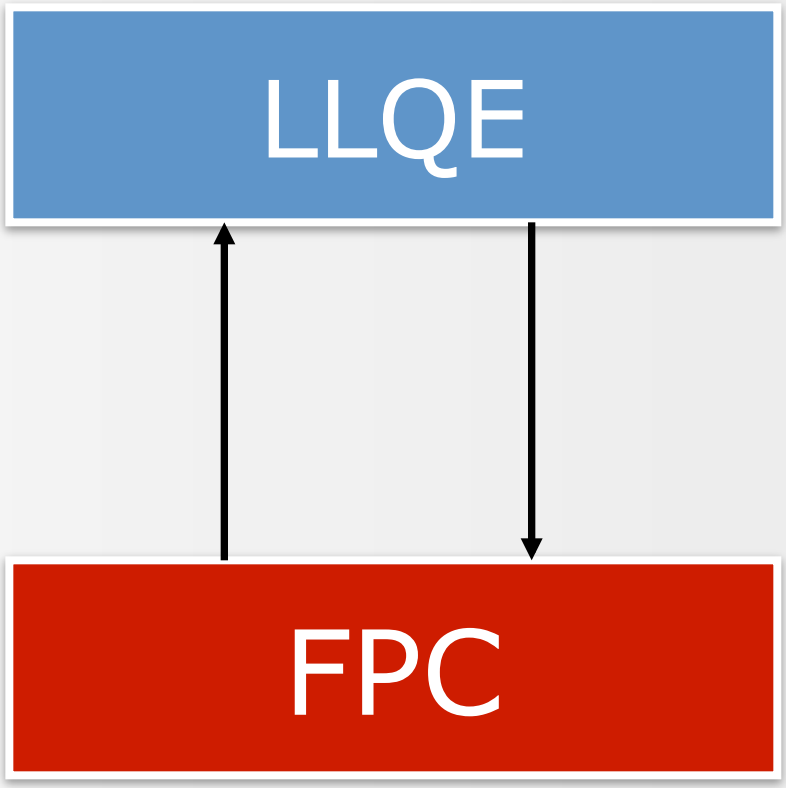
WHAT: Goals and fucntionalities

```
SELECT  temp, AVG(temp,2s);
WHERE  temp > 22
```



WHAT: Goals and functionalities

- A large number of nowadays sensors don't have full computational power
- A **LOW LEVEL EXECUTOR** must be introduced to supply for the computability lacks of such devices
- Finds its location over the FPC, to which is strictly linked

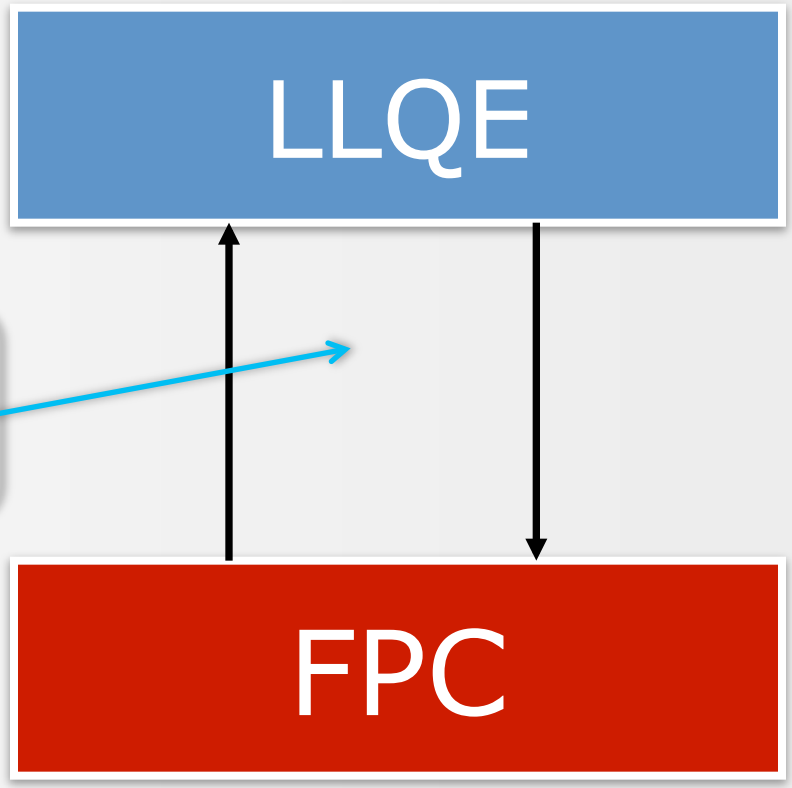


WHAT: Goals and functionalities

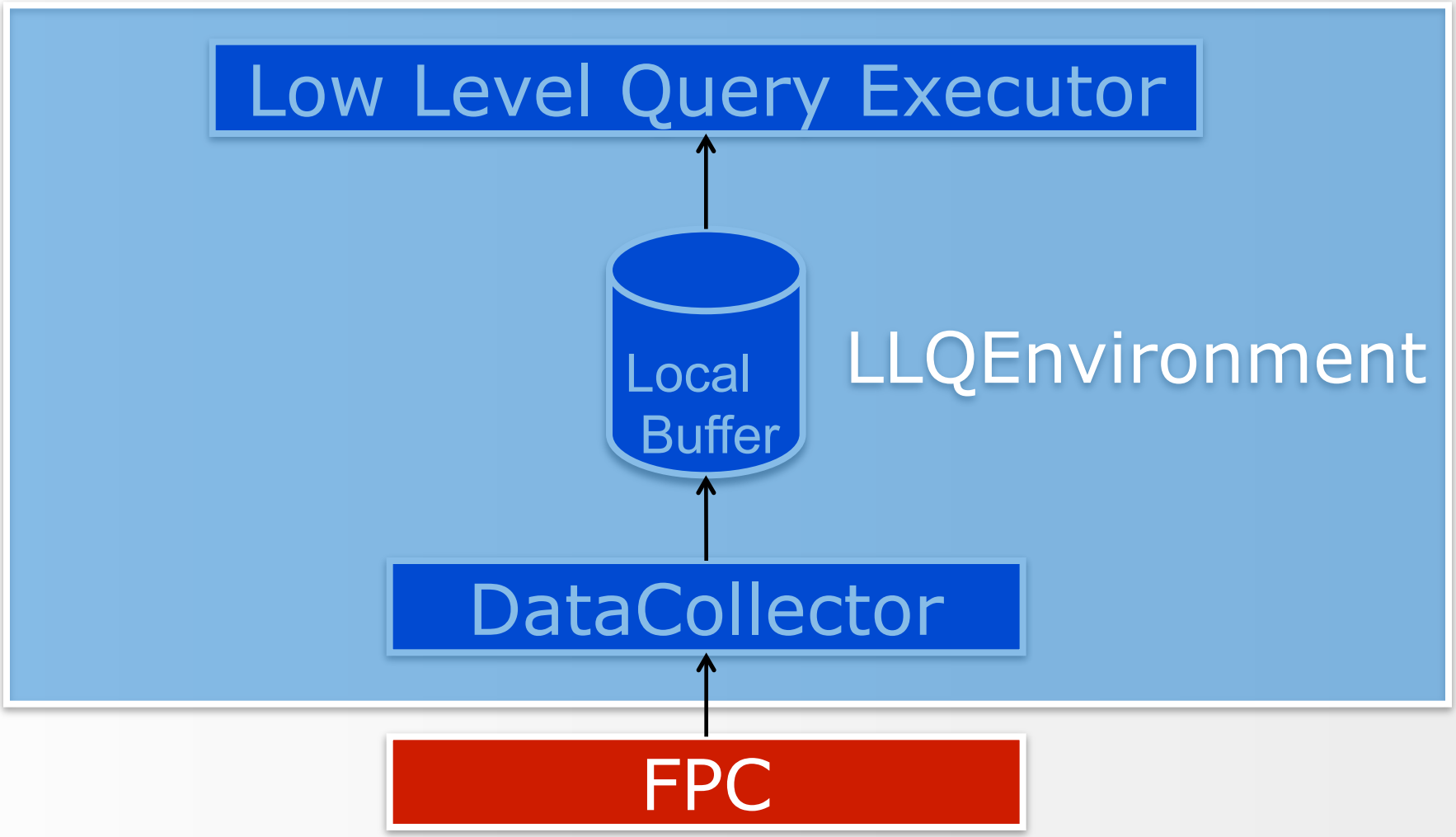
LOW LEVEL EXECUTOR tasks are wider than those suggested by its name:

1. **PHYSICALLY RETRIEVE** data from the network
2. **FILTER DATA** according a condition
3. **PROCESS DATA**

The internal structure reflects this order

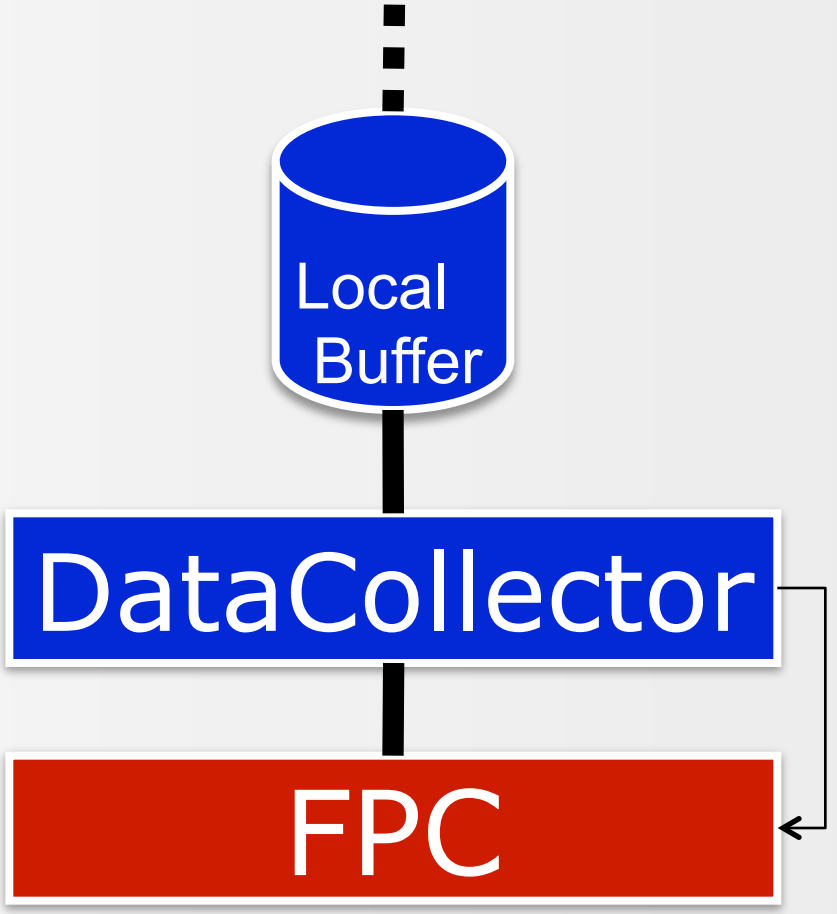


WHAT: LLQE Structure and position



HOW: Pipes vs. method calls

- All the previous structures are created at runtime by the **QUERY ANALYZER** component
- DataCollector, FPC and Local Buffer are uncoupled thanks to **PIPES**



TESTBED QUERY: ArtDeco Demo

```
CREATE STREAM DataFromWineyard (SensorID ID, avgTemp FLOAT, avgHum
FLOAT, varTemp FLOAT, varHum FLOAT) AS
HIGH:
    EVERY 1 h
    SELECT AVG(temperature, 10 m), AVG(humidity, 10 m), VARIANCE
(temperature, 10 m), VARIANCE(temperature, 10 m)
    FROM RawDataFromWineyard
    GROUP BY SensorID
```

```
CREATE STREAM RawDataFromWineyard (SensorID ID, temperature FLOAT,
humidity FLOAT) AS
LOW:
    EVERY 10 m
    SELECT humidity, temperature
    SAMPLING EVERY 60 s
```

Conclusion and open points

- The interface beetween FPC and DataCollector entity has been defined and consolidated
- LLQ Executor (data processing part) is currently under implementation: we're currently dealing with aggregates calculus.
- An interface towards the HLQs "world" must be designed and explored, as well as the HLQs "world" itself.
- A complete errors managing policy must be designed and implemented

PerLa

PERvasive LAnguage

Context Management

CONTEXT

"a set of variables and parameters that may be of interest for an agent and that influence its actions"

Or, better yet:

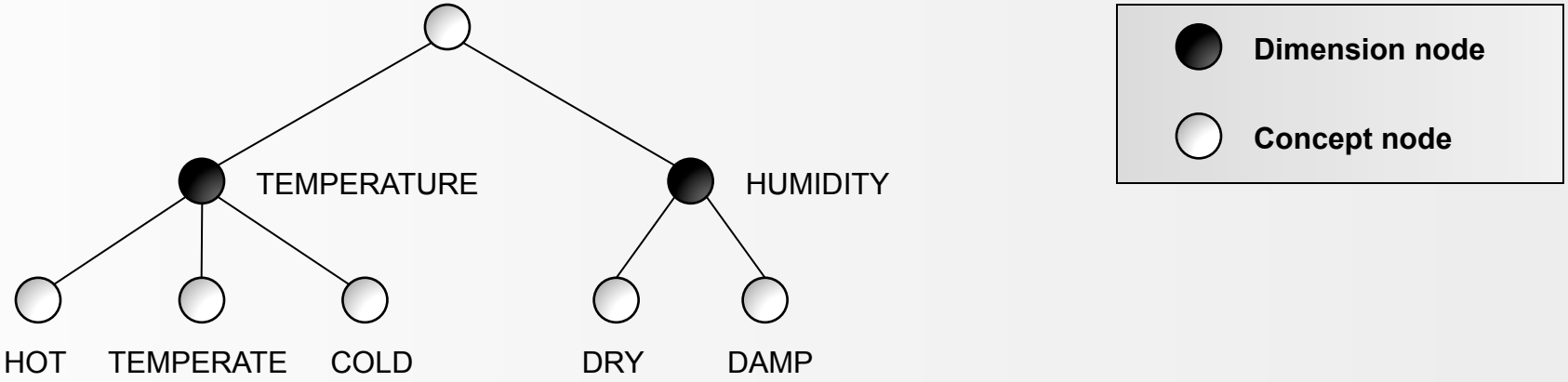
"an active process dealing with the way humans weave their experience within their whole environment, to give it meaning"

- What is the CONTEXT useful for?
  - **Data tailoring:** customizing and adapting data to the specific user's context and needs
  - **Power management:** managing the activation and deactivation of sensing devices
  - **UI customization:** adapting the user interface to the particular situation of use
- Context management in PerLa
  - **BEFORE:** context managed with the PILOT JOIN operator
  - **NOW:** context is managed using ad-hoc tools, based on the CDT (Context Dimension Tree) context model

CDT: CONTEXT DIMENSION TREE (1)

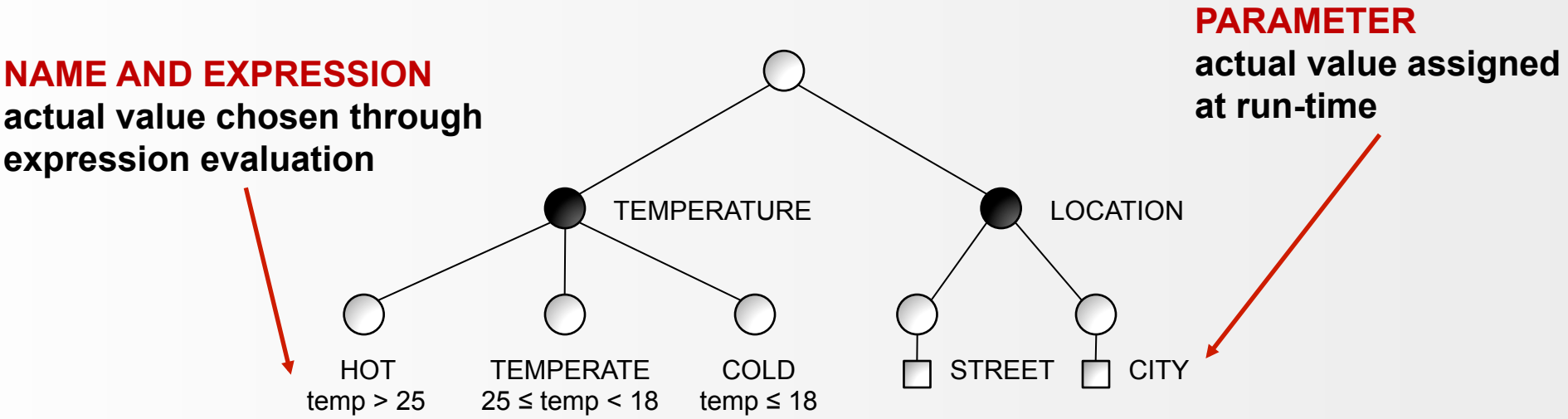
- The CDT is a context modeling tool
  - Hierarchical representation of the context model
  - Allows users to systematically model the context environment
  - Supports the discovery of the user's current context

- CDT basic concepts:
  - **DIMENSION NODE:** a specific aspect of the modeled environment
  - **CONCEPT NODE:** model the different values of a given dimension
  - **SIBLING NODES ARE MUTUALLY EXCLUSIVE**



CDT: CONTEXT DIMENSION TREE (2)

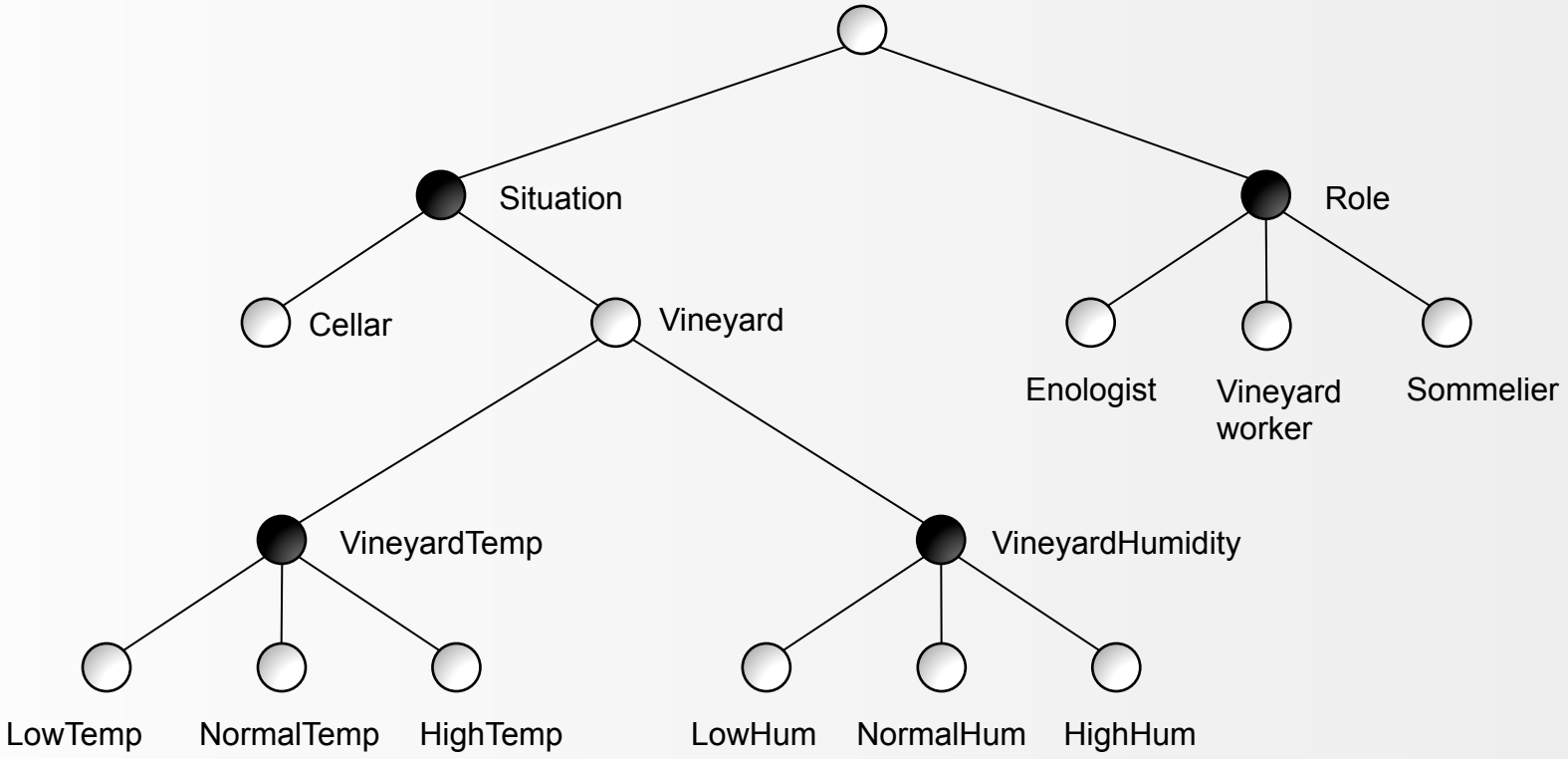
- Concept nodes (i.e., dimension values) are defined through:
  - Value name and expression (e.g., hot = temperature > 25)
  - Parameter (e.g., humidity)



- A context is defined as the conjunction of different <dimension, value> pairs (e.g., TEMPERATURE = HOT $\wedge$ HUMIDITY = 70%)
  - Not all feasible context are meaningful or possible!

CDT EXAMPLE

- An instance of CDT (excerpted from the ART DECO case study)



TREE CREATION (1)

DATA GATHERING QUERY
PerLa query employed to gather the data required to evaluate the dimension values

CREATE DIMENSION <DimensionName>
(**CHILD OF** <FatherName> | **EVALUATED ON** <LowPerLaQuery>)
“**{** ”
(
VALUE <ValueName>
(**WHEN** <BooleanExpression>)
“**;**”
)+
“ **}** ”

CONCEPT CREATION STATEMENT
definition of the dimension values

TREE CREATION (2)

```
CREATE DIMENSION Situation
EVALUATED ON CREATE SNAPSHOT situationData (nodeId ID, humidity,
                        temperature, locationX, locationY)
                        WITH DURATION 1h AS LOW:
                    EVERY ONE
                    SELECT ID, humidity, temperature, locationX, locationY
                    SAMPLING
                        EVERY 1h
{
        VALUE Vineyard
        WHEN is_in_Vineyard(locationX, locationY);

        VALUE Cellar
        WHEN is_in_Cellar(locationX, locationY) ;
}
```

TREE CREATION (3)

```
CREATE DIMENSION VineyardHumidity
CHILD OF Vineyard
{
        VALUE LowHum
        WHEN humidity <30% ;

        VALUE NormalHum
        WHEN  humidity <70% AND humidity >30% ;

        VALUE CriticalHum
        WHEN  humidity >70%;
}
```

CONTEXT DEFINITION (1)

CONTEXT ACTIVATION CONDITION
conjunction of <dimension, value> pairs

CREATE CONTEXT <*ContextName*>
ACTIVE IF <*ActivationCondition*>
ON ENABLE <Contextual*Query*>
[**ON DISABLE** <*ActualizationQuery*>]
REFRESH <*Condition*>

CONTEXT REFRESH CONDITION
dictates when the ACTIVE IF has to be re-evaluated

ON ENABLE/ON DISABLE QUERIES
run when the context is activated and deactivated

CONTEXT DEFINITION (2)

```
CREATE CONTEXT FrostAlarm
ACTIVE IF VineyardTemperature =  LowTemp AND
          VineyardHumidity = HighHum
ON ENABLE  CREATE OUTPUT STREAM MonitoringFrost (nodeId ID,
                    temperature FLOAT, humidity FLOAT,
                    locationX FLOAT, locationY FLOAT) AS LOW:
               EVERY ONE
               SELECT ID, temperature, humidity, locationX, locationY
               SAMPLING
                    EVERY 1m
ON DISABLE Drop MonitoringFrost
REFRESH EVERY 1h
```

OPEN POINTS

- The current solution does not manage parametric dimensions and values
- The context management engine has been designed but not yet implemented

Bibliography

[1] A Middleware Architecture for Data Management and Integration in Pervasive Heterogeneous Systems
F.A. Schreiber, R. Camplani, M. Fortunato, M. Marelli
Politecnico di Milano, Dipartimento di Elettronica e Informazione, Milano, Italy

[2] Scoping in Wireless Sensor Networks
Jan Steffan Ludger Fiege Mariano Cilia Alejandro Buchmann
Department of Computer Science, Darmstadt University of Technology

[3] A Context-Aware Approach to Conserving Energy in Wireless Sensor Networks
Chong, Krishnaswamy, Loke
School of Computer Science and Software Engineering
Monash University,Melbourne, Australia

[4] Proactive Context-Aware Sensor Networks
Sungjin Ahn and Daeyoung Kim
Real-time and Embedded Systems Laboratory,
Information and Communications University (ICU)

[5] Energy Efficient Spatial Query Processing in Wireless Sensor Networks
Kyungseo Park, Byoungyong Lee, Ramez Elmasri
Computer Science and Engineering, University of Texas at Arlington

[6] Context-Aware Sensors
Eiman Elnahrawy and Badri Nath
Department of Computer Science, Rutgers University

Bibliography

[7] A Self-Adaptive Context Processing Framework for Wireless Sensor Networks
Amirhosein Taherkordi, Romain Rouvoy, Quan Le-Trung, and Frank Eliassen
University of Oslo, Department of Informatics

[8] A spatial extension of TinyDB for wireless sensor networks
Paolino Di Felice, Massimo Ianni, Luigi Pomante
DIEI DEWS - Universita dell'Aquila, Italy

[9] Efficient and Practical Query Scoping in Sensor Networks
Henri Dubois-Ferri`ere
School of Computer and Communication Sciences
EPFL Lausanne, Switzerland - Department of Computer Science UCLA, Los Angeles, CA

[10] Adaptive middleware for contextaware applications in smarthomes
Markus C. Huebscher DSE Group, Department of Computing Imperial College London
Julie A. McCann DSE Group, Department of Computing Imperial College London

[11] TinyDB: An Acquisitional Query Processing System for Sensor Networks
SAMUEL R. MADDEN Massachusetts Institute of Technology
MICHAEL J. FRANKLIN and JOSEPH M. HELLERSTEIN UC Berkeley
WEI HONG Intel Research, Berkeley

[12] Context Discovery in Sensor Networks
Chia-Hsing HOU, Hung-Chang Hsiao, Chung-Ta King, Chun-Nan Lu
Computer Science, National Tsing-Wua University, Hsinchu, Taiwan,

Bibliography

[13] Progettazione dei dati con l'uso del contesto
C. Bolchini, G. Orsi, E. Quintarelli, F. A. Schreiber, L. Tanca
Dipartimento di Elettronica e Informazione – Politecnico di Milano

[14] And what can context do for data?
C. Bolchini, C. A. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, L. Tanca
Dipartimento di Elettronica e Informazione – Politecnico di Milano

[15] Context-based data tailoring for Mobile Users
Letizia Tanca
Dipartimento di Elettronica e Informazione – Politecnico di Milano

PerLa
PERvasive LAnguage

FUTURE WORKS

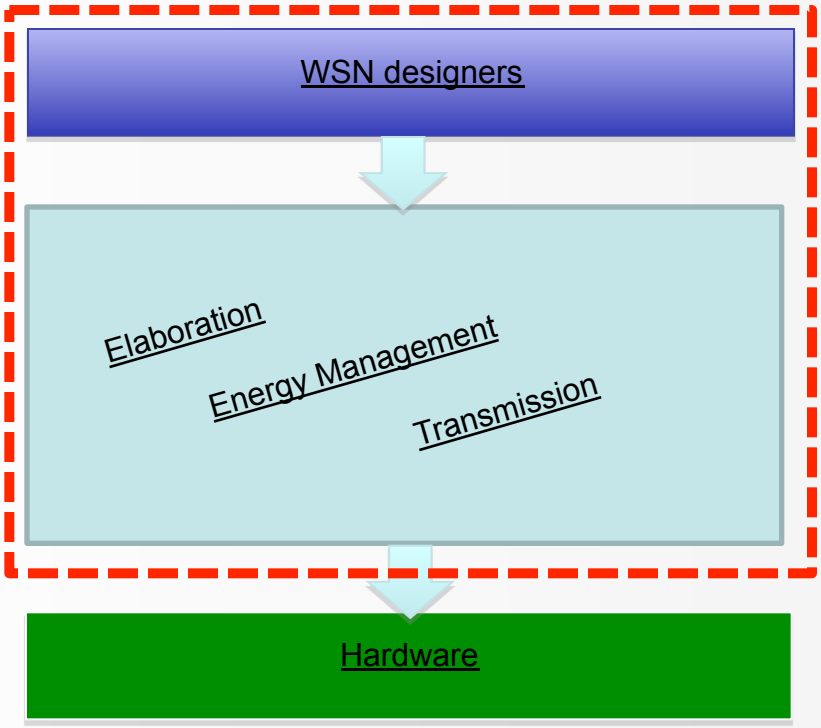
R. Camplani
Politecnico di Milano,
Dipartimento di Elettronica e Informazione,
Milano, Italy

FUTURE WORKS

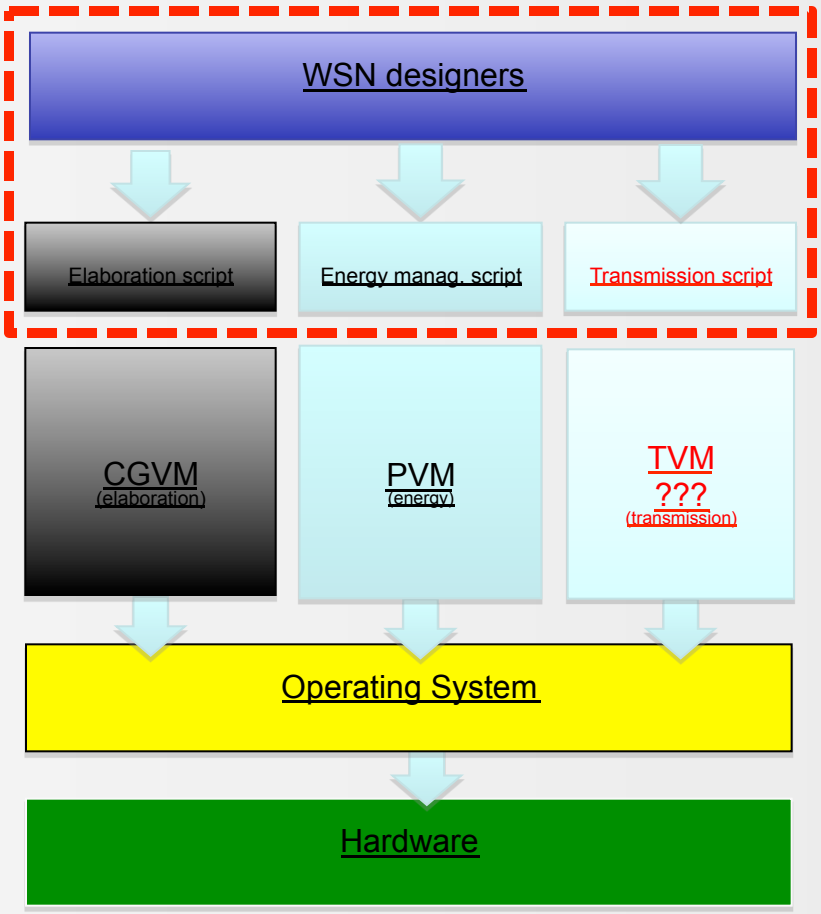
- Think about intelligent **Context Discovery**
- Add support for context-based routing
  - Up to now, routing is used only as data transport
    - “net neutrality” vision
- Push the “intelligence” towards the nodes
  - Node behavior determined at run-time
    - Data elaboration
    - Routing strategies
    - Power management

New: low level approach

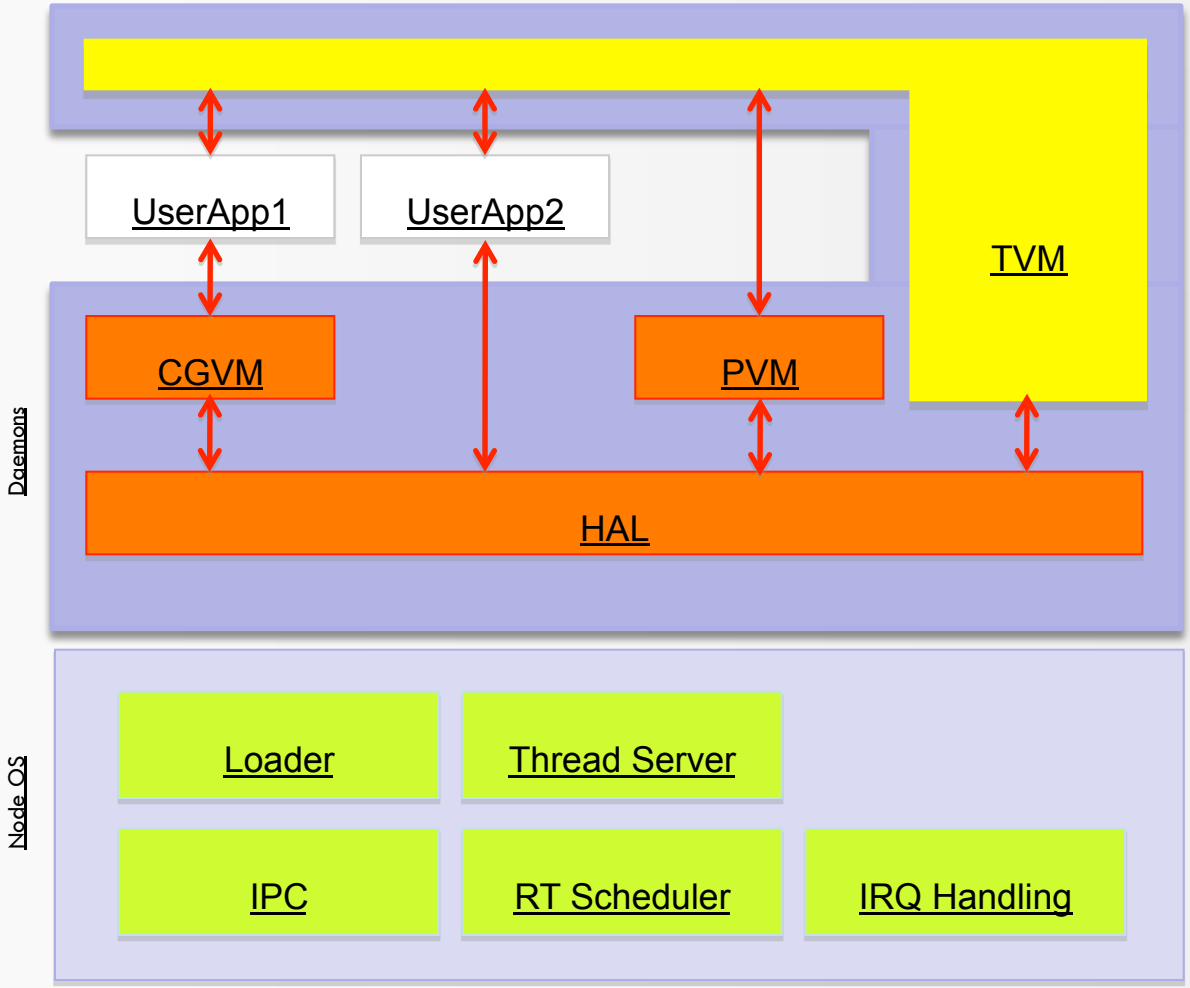
Traditional approach



Proposed approach



Node SW architecture



Middleware evolution