PerLa
PERvasive LAnguage

PerLa: LANGUAGE and MIDDLEWARE

F.A. Schreiber, R. Camplani, M. Fortunato, M. Marelli, G. Rota

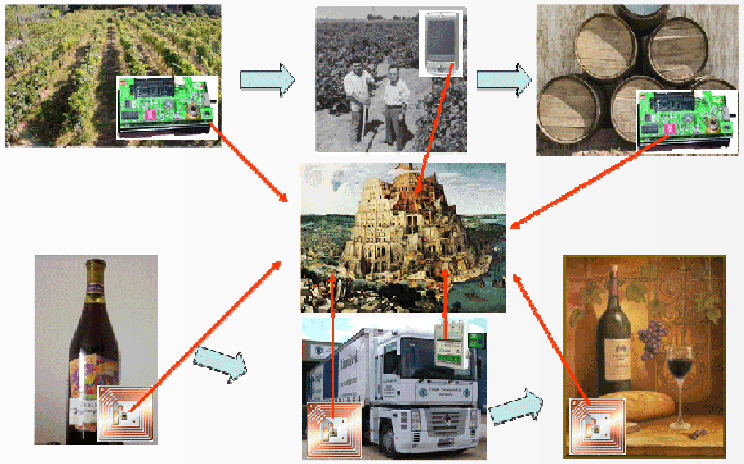
Politecnico di Milano,
Dipartimento di Elettronica e Informazione,
Milano, Italy

OUTLINE

- Introduction
  - Pervasive Systems
  - Open Issues
- State of the art
- Proposed solution: **PerLa**
  - **PerLa** internals
    - Frontend
    - Middleware
    - Low-Levels
- Real Testbed: Lecco's deployment
- Future works

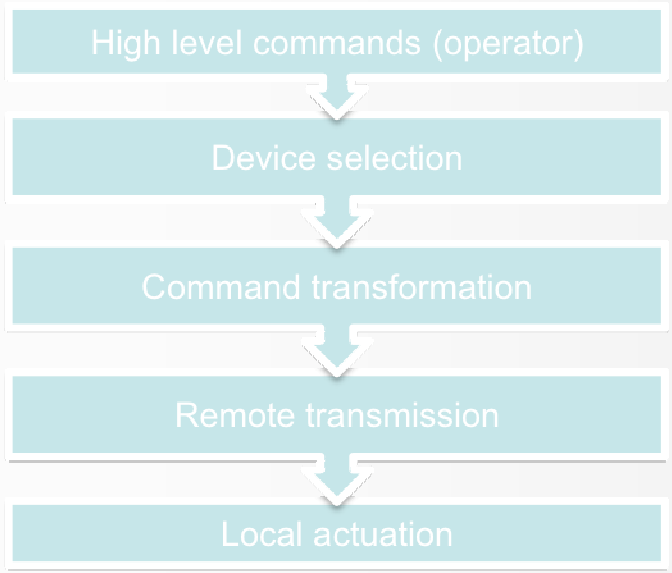
INTRODUCTION: PERVASIVE SYSTEMS

- A pervasive system is composed of heterogeneous devices:
  - *RFID* tags
  - Sensor motes
  - *PDA*
  - Actuators
- Pervasive systems scenarios:

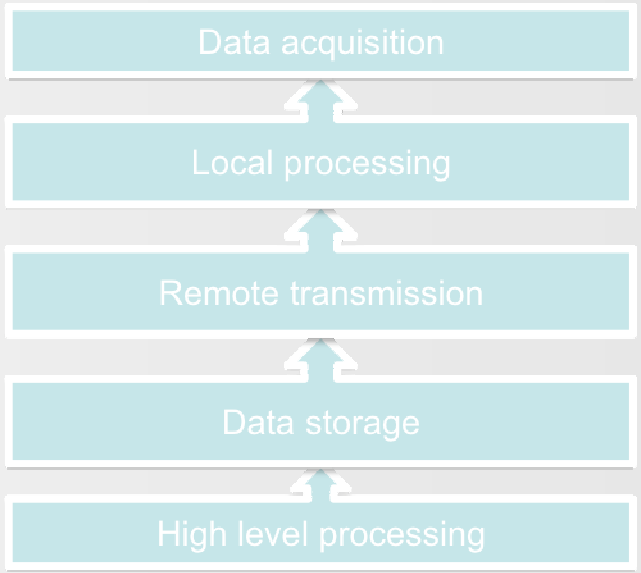


TYPICAL APPLICATIONS IN PERVASIVE SYSTEMS

- **DATA** life cycle



- **COMMANDS** life cycle



What about a real deployment?

REAL WORLD APPLICATION OF PERVASIVE SYSTEMS

- First examples [1][2][3][4] are "**EMBEDDED**" systems
  - Only support for **SPECIFIC HARDWARE**
  - **AD - HOC** transmission
    - Data dependent!
  - **DEDICATED** server application
    - "SQL-in-the-code" paradigm

A more "engineered" approach?

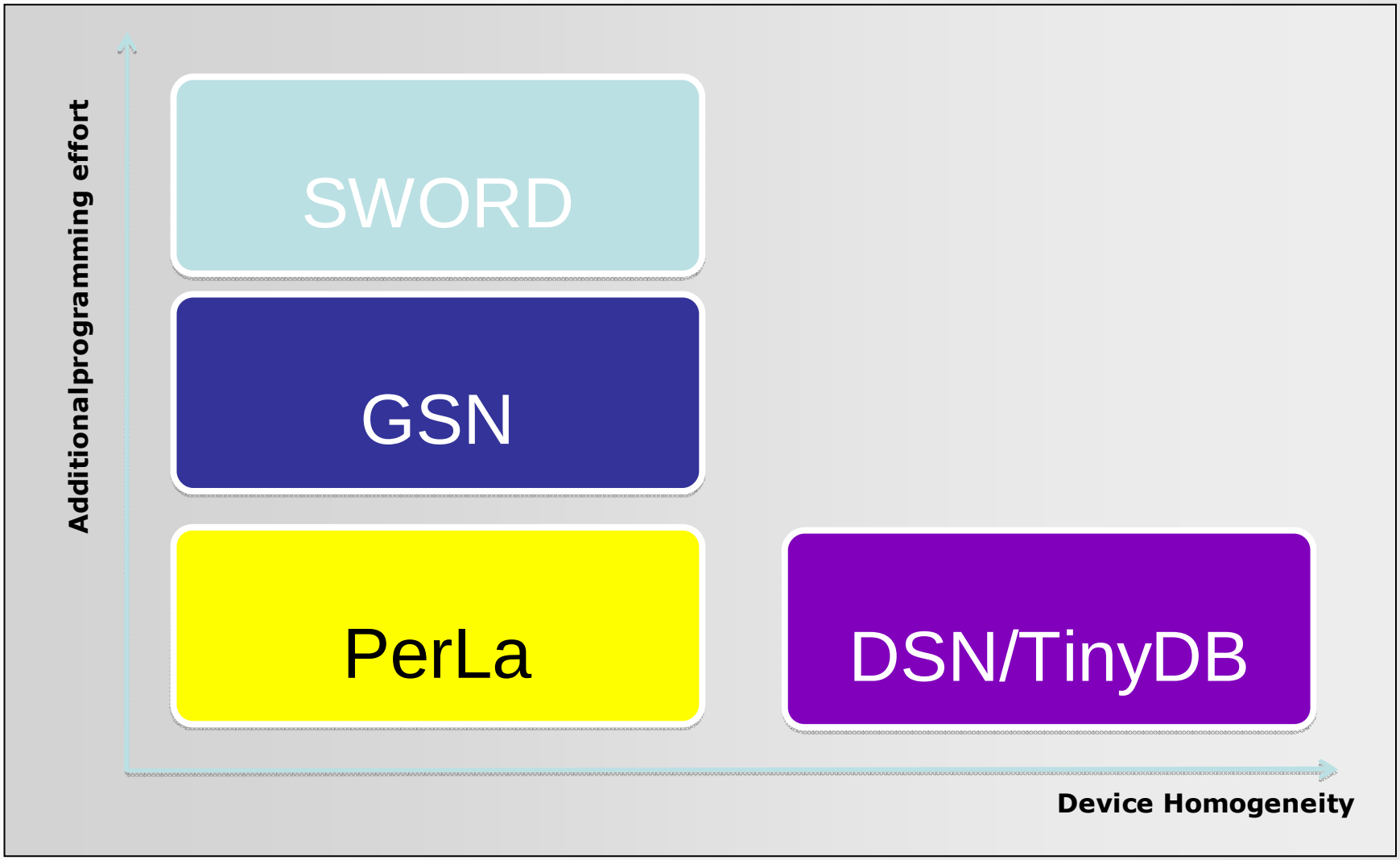
STATE OF THE ART (1)

- There are some projects aiming at identifying **APPROACHES** to manage pervasive systems
  - The key idea:
    - An **HIGH LEVEL LANGUAGE** to define the envisaged pervasive system (data, alarms, etc..)
  - Most important projects
    - *TinyDB [5]*
    - *DSN [6]*
    - *GSN [7]*
    - *SIEMENS SWORD [8]*

STATE OF THE ART (2)

	TinyDB	GSN	DSN	SWORD
Data gathering	✔	✗	✔	✗
Configurability	—	✗	✗	✗
Data aggregation	✔	—	✔	✗
High level integration	✔	✔	✔	✔
Re-Usability	—	✔	—	✔
Low Level software support	✔	✗	✔	✗
Heterogeneity supp.	✗	✔	✗	✔

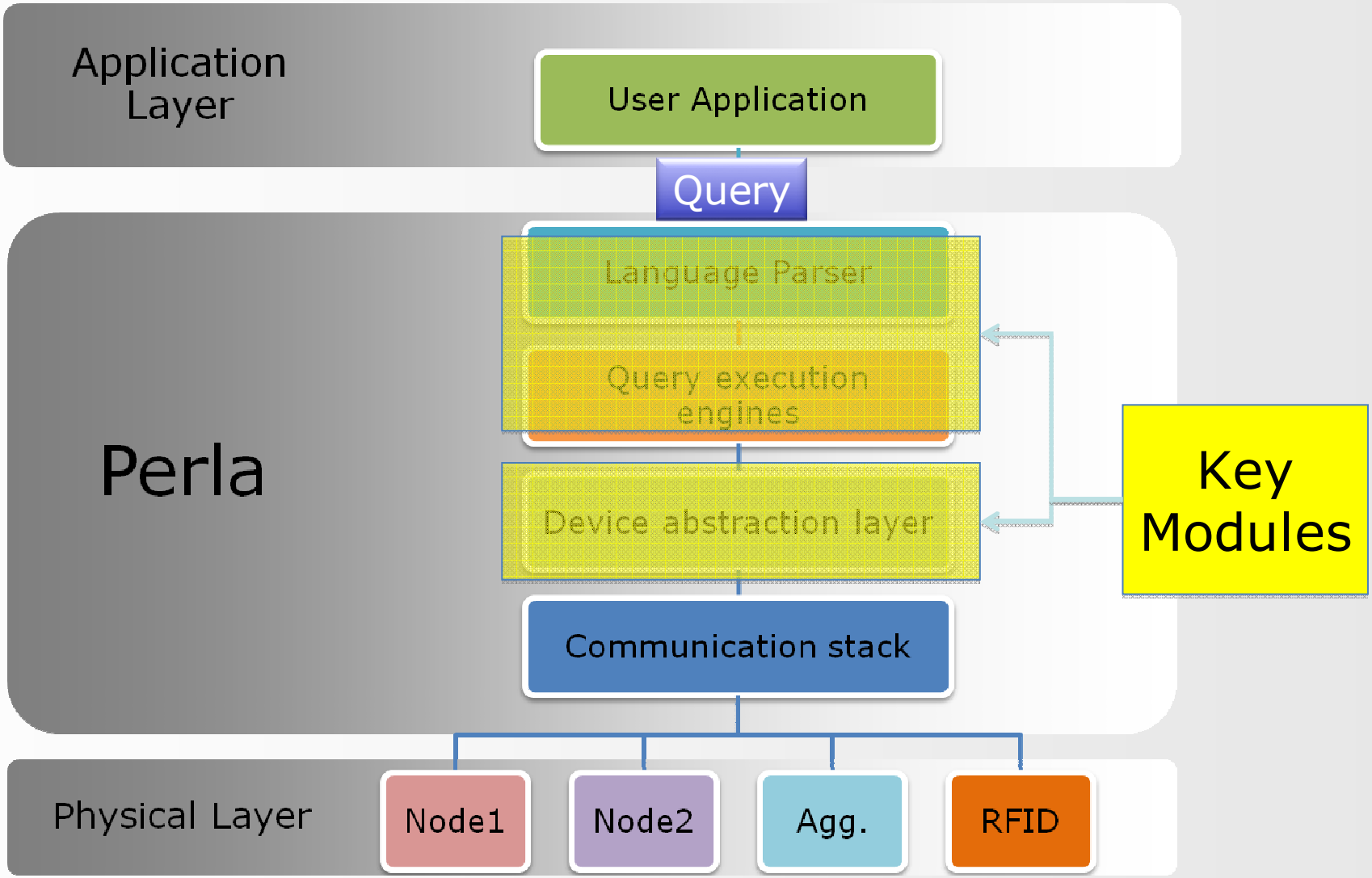
STATE OF THE ART (3)



PERLA OVERVIEW (1)

- Improvement to the state of the art:
  - Use of the database abstraction
    - defines a user friendly language to handle pervasive systems
      - similar as possible to SQL
      - **DSN** is based on Snlog, not widely known.
  - Heterogeneity
    - deploy-time
    - run-time
    - **TinyDB** and **DSN** only support a single homogenous network
  - Middleware
    - makes the support for new devices easy
    - reduces the amount of the needed low level code
    - **GSN**, **SWORD** do not provide low level interfaces for devices
      - TCP/IP+XML-based protocol
      - No support for low level devices firmware

PERLA OVERVIEW (2)



PERLA: KEY FEATURES

High level interface: the language

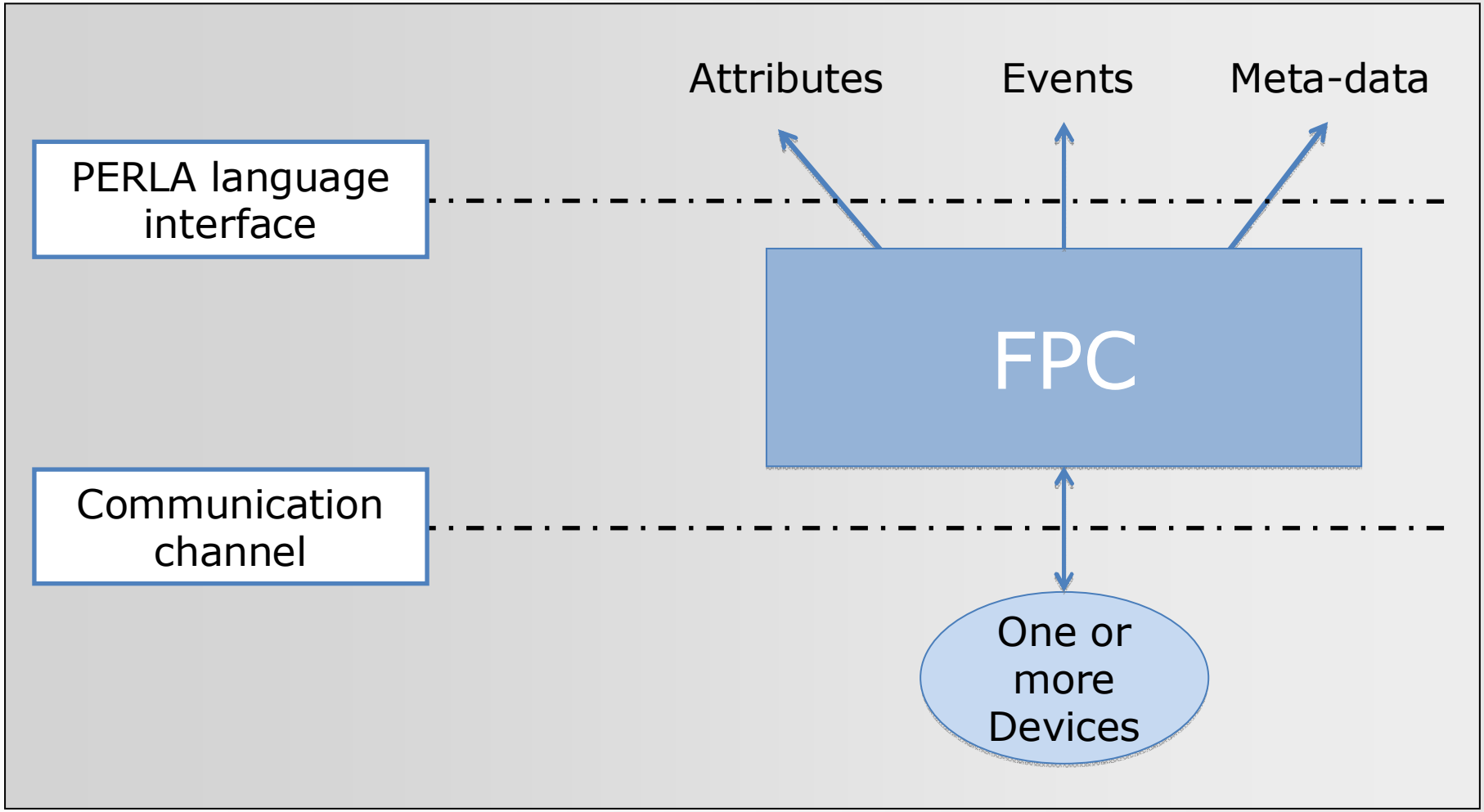
SQL-like syntax

- Three levels of queries
  - High level query (HLQ)
    - Equivalent to SQL for streaming DB
  - Actuation query (AQ)
    - Executes commands, set parameters on devices
  - Low level query (LLQ)
    - Defines the behaviour of a single or of a group of devices

Low level interface: the hardware abstraction layer

- Devices as a Functionality Proxy Component (FPC)
- An FPC provides:
  - Attribute reading (id, temperature, pressure, power level, last sensed RFID reader, …)
  - Event notification (last sensed RFID reader changed, …)
  - Meta-description (name, data type, …)

LANGUAGE - FPC INTERFACE



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

Data structures

- Two types of data structures
  - STREAM TABLES
    - Unbounded lists of records
    - Queries can perform
      - insert (insertion of a new record)
      - read (extract a data window [ts, size])
  - SNAPSHOT TABLES
    - Set of records produced by a query in a given period t
    - Content refreshed every period t

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

LLQ: PHYSICAL DEVICE MANAGEMENT

- Both sampling and data operations management can be executed:
  - **PERIODICALLY**
  - **EVENT BASED**

- ***Example***: *RFID* abstraction
  - **RFID TAG AS A SENSOR**
    - sampled data → id of the last reader which sensed the tag
  - **READER AS A SENSOR**
    - sampled data → id of the last tag sensed by the reader
  - EVENT BASED SAMPLING
    - when the corresponding FPC senses the reader firing

LLQ: NON FUNCTIONAL CHARACTERISTICS

- Non functional fields exposed by *FPC* are expressed in an abstract way and **TRANSLATED** in concrete values handled by physical devices

- ***Example***: the power level in a device
  - voltage value
  - predicted from the number of performed operations
  - set to 100% for a.c. powered devices

LLQ: AN EXAMPLE

Sample the temperature every 30 seconds and, every 10 minutes, report the number of samples that exceeded a given threshold

```
INSERT INTO STREAM Table (sensorID, temperature)
LOW:
  EVERY 10 m
  SELECT ID, COUNT(temp, 10 m)
```

DATA MANAGEMENT SECTION

Event based activation

Time based activation

```
  SAMPLING
    EVERY 30 s
    WHERE temp > 100
```

SAMPLING SECTION

Event based sampling

Timebasedsampling

```
  EXECUTE IF
powerLevel> 0.2 AND EXISTS (temp)
```

EXECUTION CONDITIONS SECTION

PILOT JOIN OPERATION

- The **PILOT JOIN** operation activates the execution of a low level query on FPC sconditioned by values sampled on **OTHER NODES**
- Two types of pilot join are supported:
  - **EVENT BASED** pilot join
  - **CONDITION BASED** pilot join
- Example:
  - Monitor the temperature of all the pallets in trucks whose current position is in a given parking area
    - Temperature sensors on pallets
    - Position sensors on trucks

PILOT JOIN BaseStationList ON
currentBaseStation = baseStationList.baseStationID

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 OutOfTemperatureRangePallets (palletID ID) AS
LOW:
    EVERY 10 m
    SELECT ID
    SAMPLING
        EVERY 10 m
        WHERE temp > [threshold]
    PILOT JOIN TrucksPositions
        ON baseStationID = TrucksPositions.linkedBaseStationID
```

HIGH LEVEL QUERIES

- Perform complex SQL queries on windows extracted from one or more input streams
  - **TIME DRIVEN**
  - **EVENT DRIVEN**
- Data manipulation provided by
  - **STREAM TABLES**
    - Unbounded lists of records
    - Queries can perform
      - insert (generates an insertion event)
      - read (extracts a data window[ts, size])
  - **SNAPSHOT TABLES**
    - Set of records produced by a query in a given period
- Every record is time-stamped

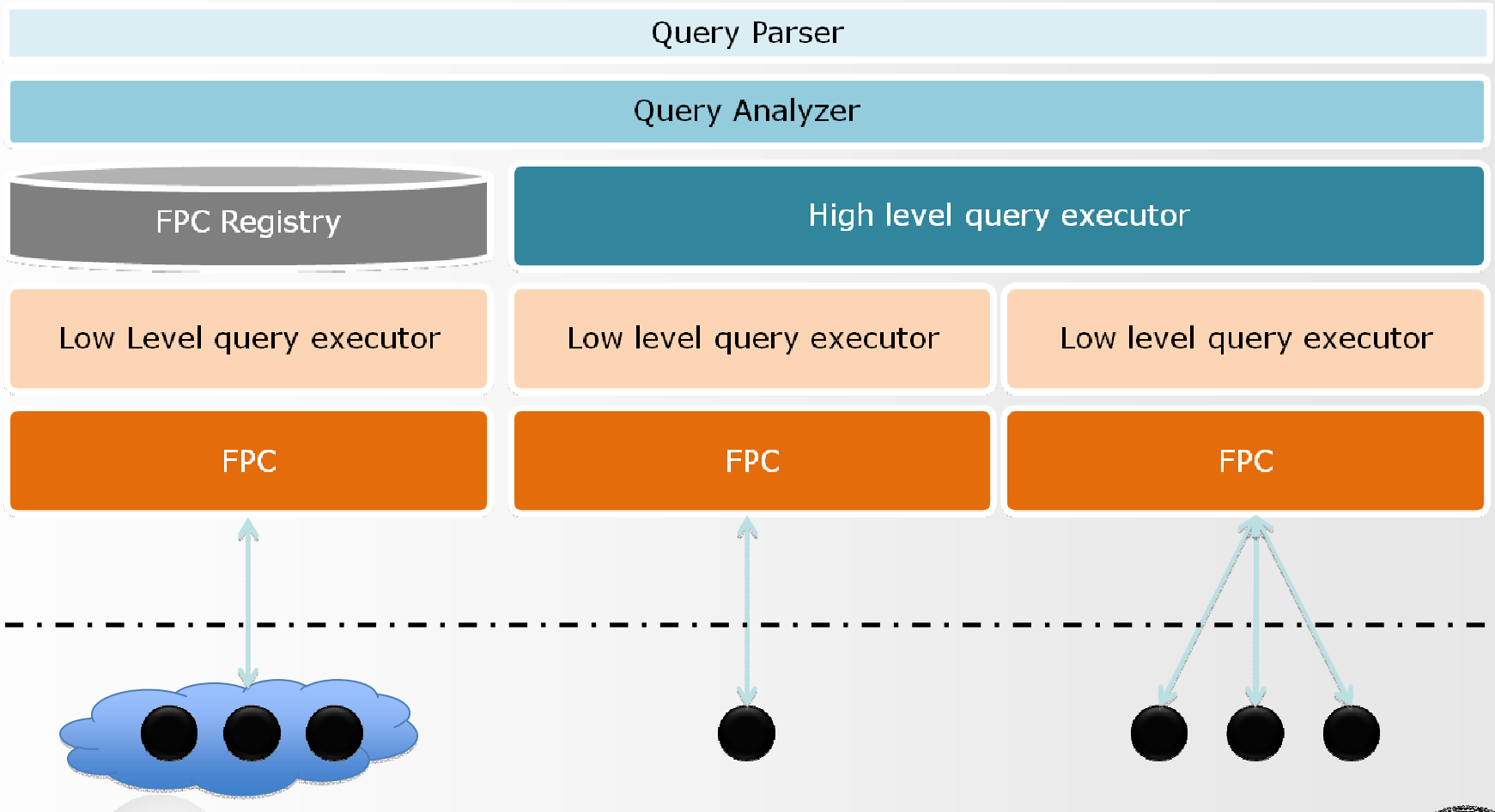
QUERY EXAMPLE (2)

```
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 MIDDLEWARE

- A middleware is needed to provide an implementation of the logical object



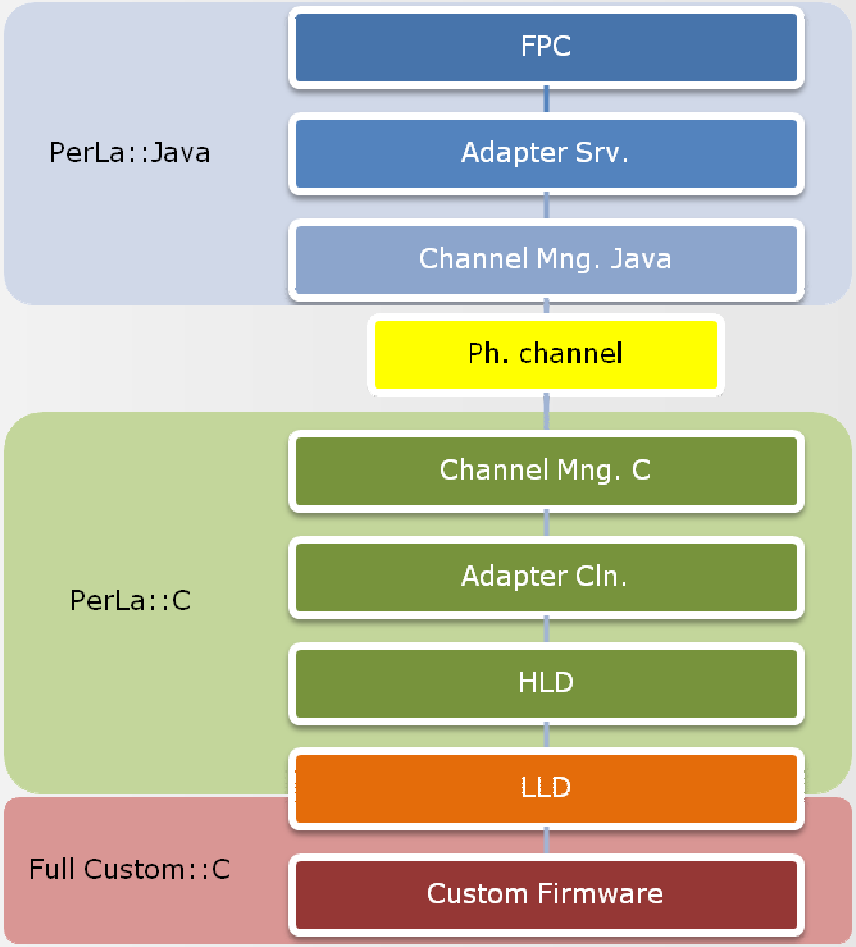
PerLa
PERvasive LAnguage

MIDDLEWARE GOALS

- Providing an **ABSTRACTION** for each device
- Supporting the **EXECUTION OF PERLA QUERIES**
- **PLUG & PLAY** support: allows devices to automatically start query execution when they are powered on
- making the **DEFINITION** and the **ADDITION** of new devices (and new technologies) easy, reducing the amount of the needed low level code

FUNCTIONALITY PROXY COMPONENT (FPC)

- The FPC is defined as a Java object representing a physical device.
- The FPC must be instantiated on a system capable of:
  - Running a Java Virtual Machine (JVM)
  - Connecting to a TCP-IP network
- The middleware manages the **COMMUNICATION PROTOCOL** between FPC and physical device



LOW LEVEL SUPPORT: HLD AND LLD

- PerLa provides a portable framework, called **HLD** (**High Level Driver**), which completely abstracts the hardware of the single device
- **HLD** is a set of common components that take care of the communications with the FPC
  - Channel virtualization
  - Data encapsulation
- The **LLD (Low Level Driver)** is the software needed by the **HLD** to access the hardware features of the sensor
  - It has to be written by the user
  - PerLa provides bindings and interfaces

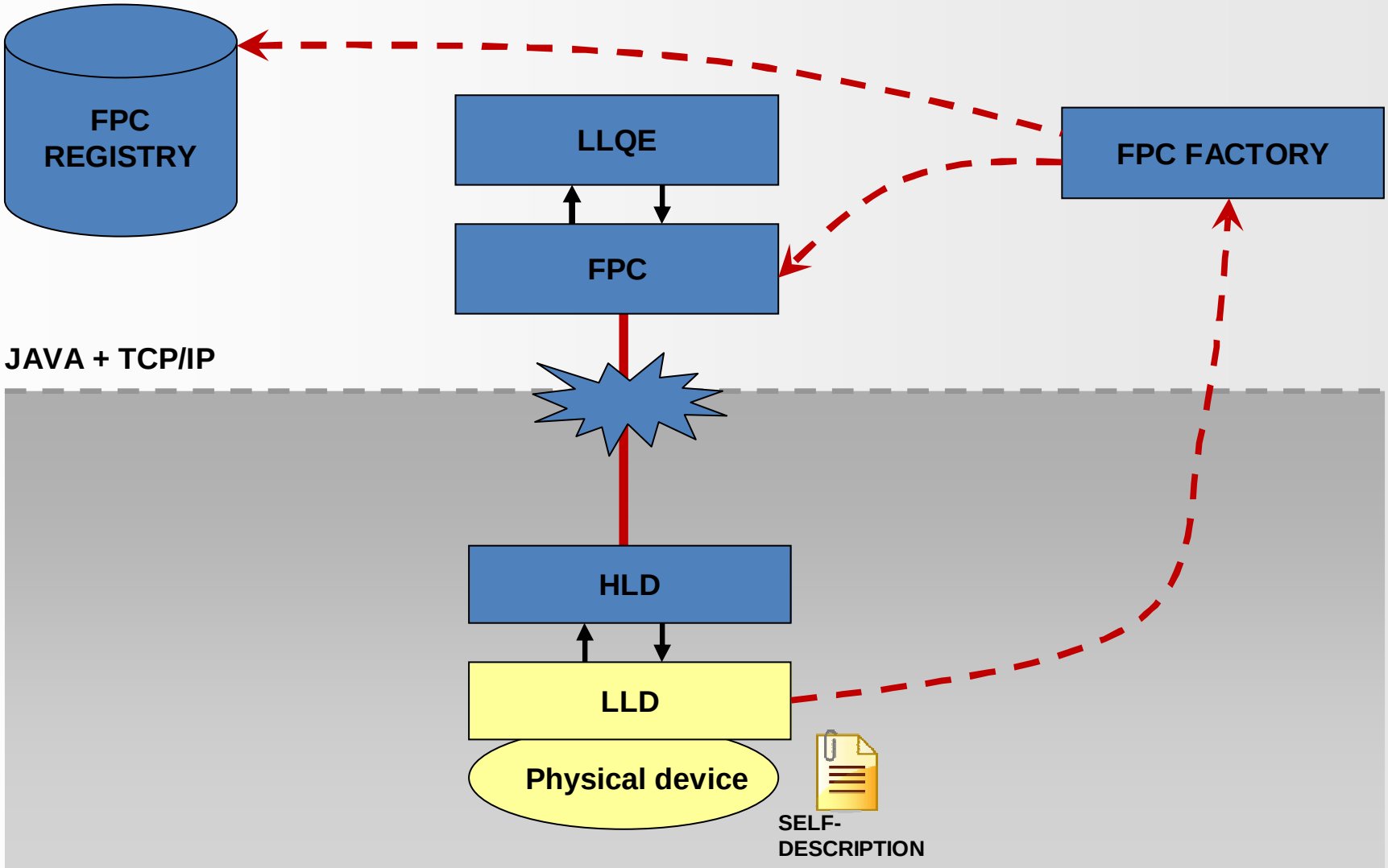
PLUG & PLAY SUPPORT (1)

- **PLUG & PLAY** at device start-up requires:
  - **DYNAMIC GENERATION** of the FPC
  - On the fly binding mechanism to handle connection between FPC and physical
  - Insertion of new FPC*s* into the **REGISTRY**

How to build an FPC to handle a new device ?

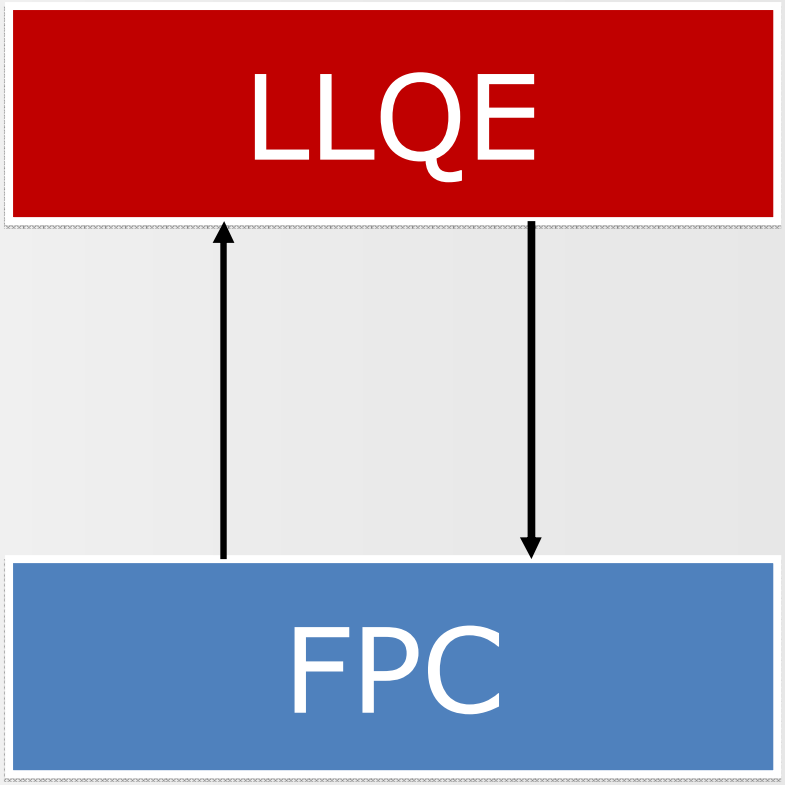
- By means of a xml-based **DEVICE-DESCRIPTION**
  - Sent by the device itself
  - Defines available data streams and events raised
  - Specify the message protocol used by the device
    - Commands format
    - Data format

PLUG & PLAY SUPPORT (2)

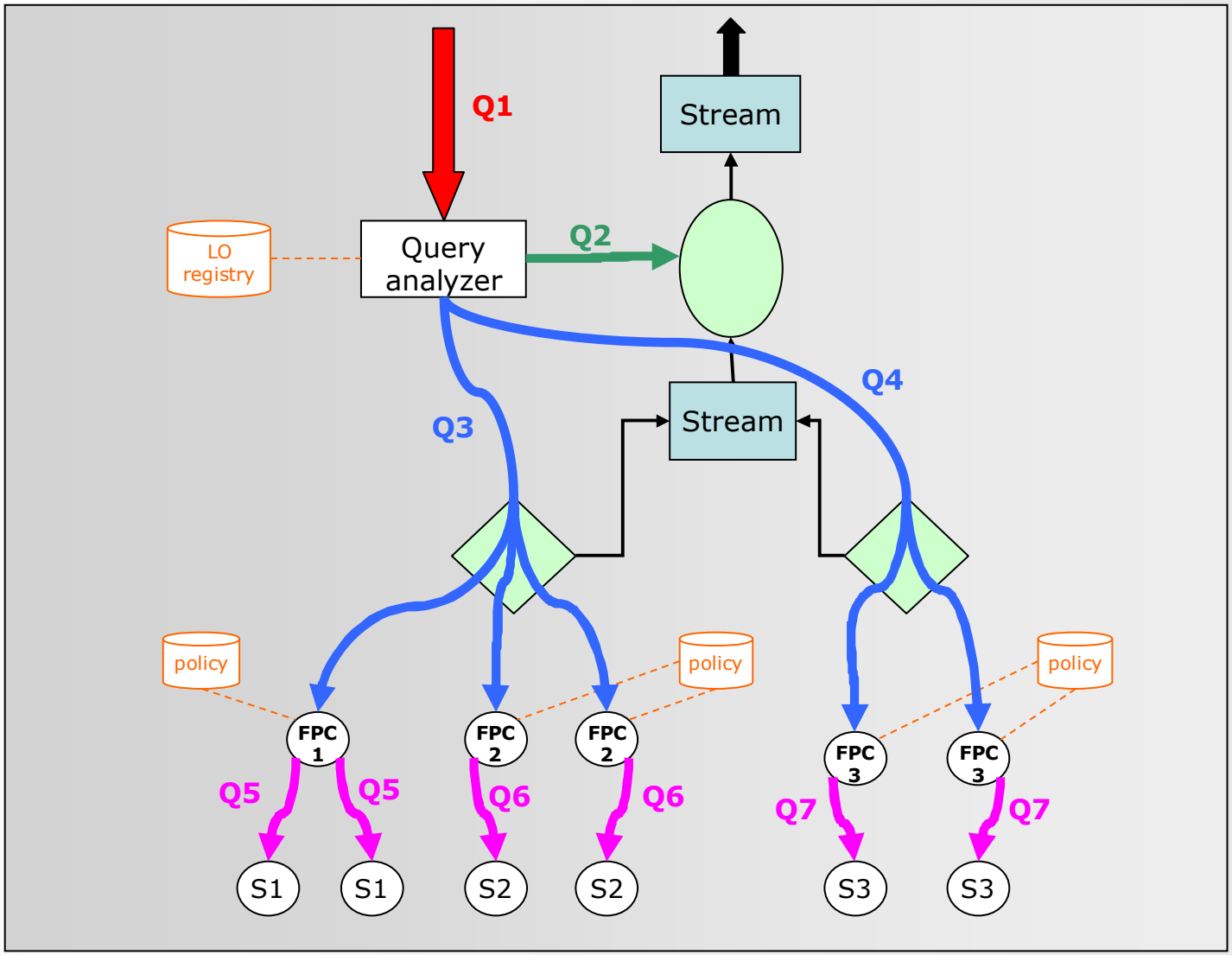


LOW LEVEL QUERY EXECUTOR

- The **LLQE** (**Low Level Queries Executor**) is a Java component placed on top of FPC.
  - Retrieve needed data from the underlying FPC and to compute **QUERY RESULTS**.
- An **LLQE** supports the simultaneous execution of all the low level queries running on the node.



QUERY DEPLOYMENT



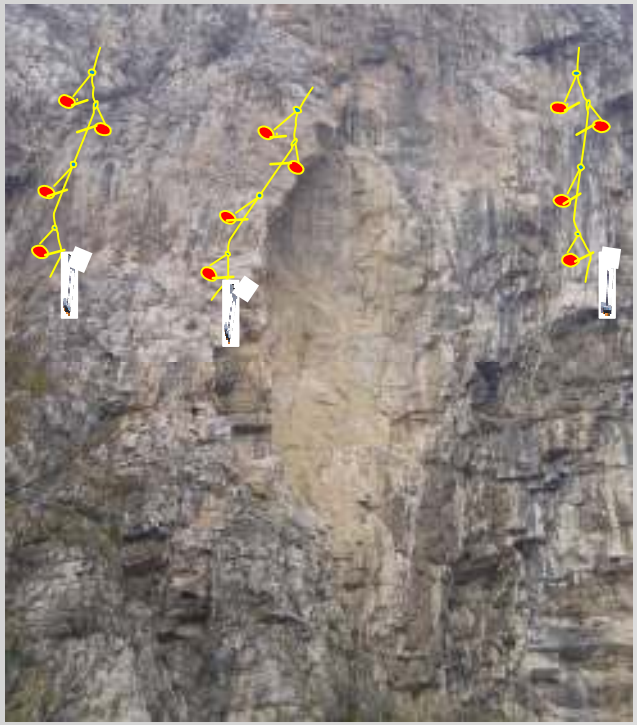
PerLa
PERvasive LAnguage

REAL TESTBED: LECCO'S DEPLOYMENT

Rockfall in Monte San Martino, Lecco

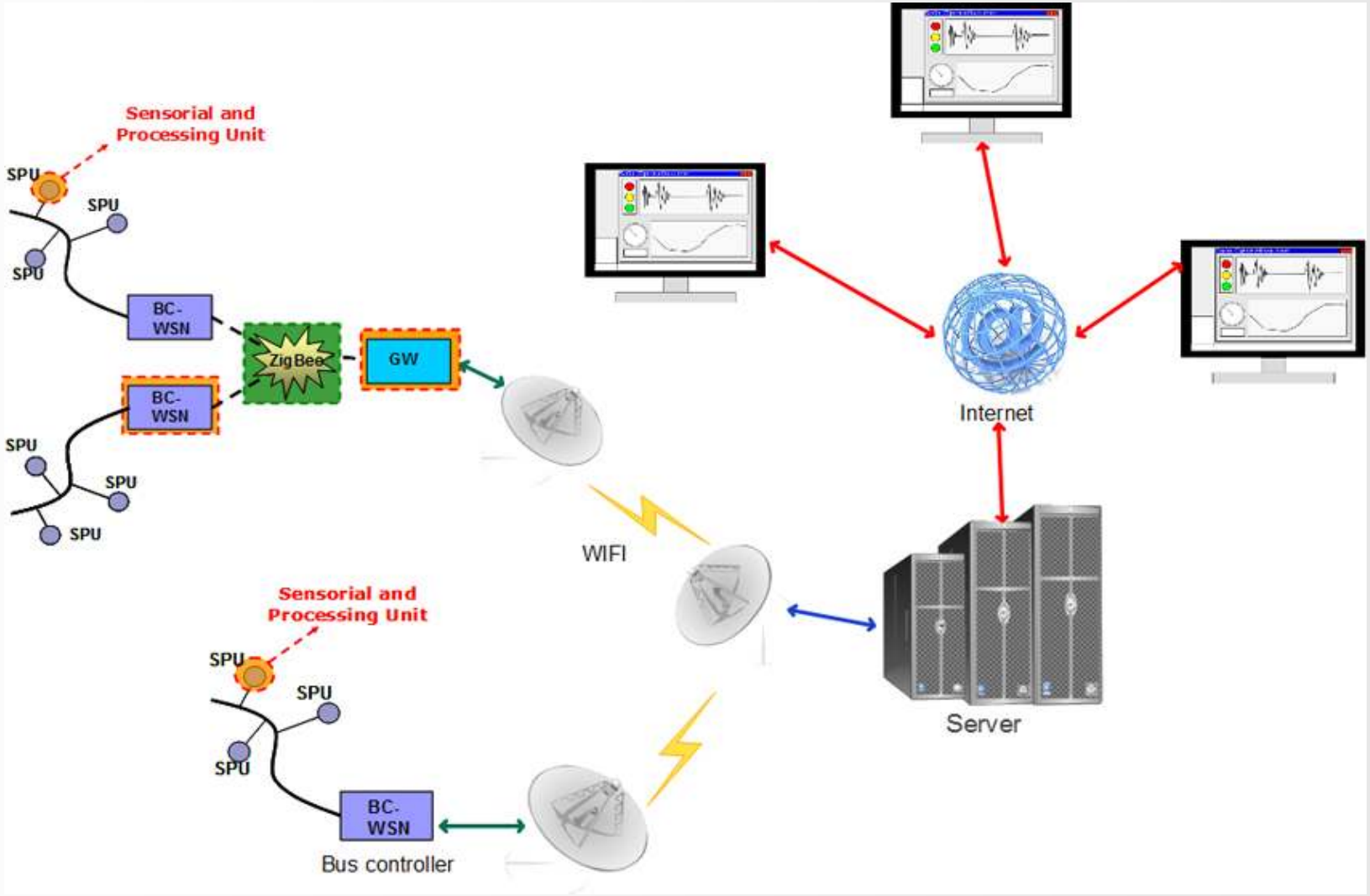
A POSSIBLE DEPLOYMENT OF THE REAL-TIME MONITORING SYSTEM

Particular of the crown where sensors will be deployed:
already collapsed site size (LxHxD) 10x40x10m

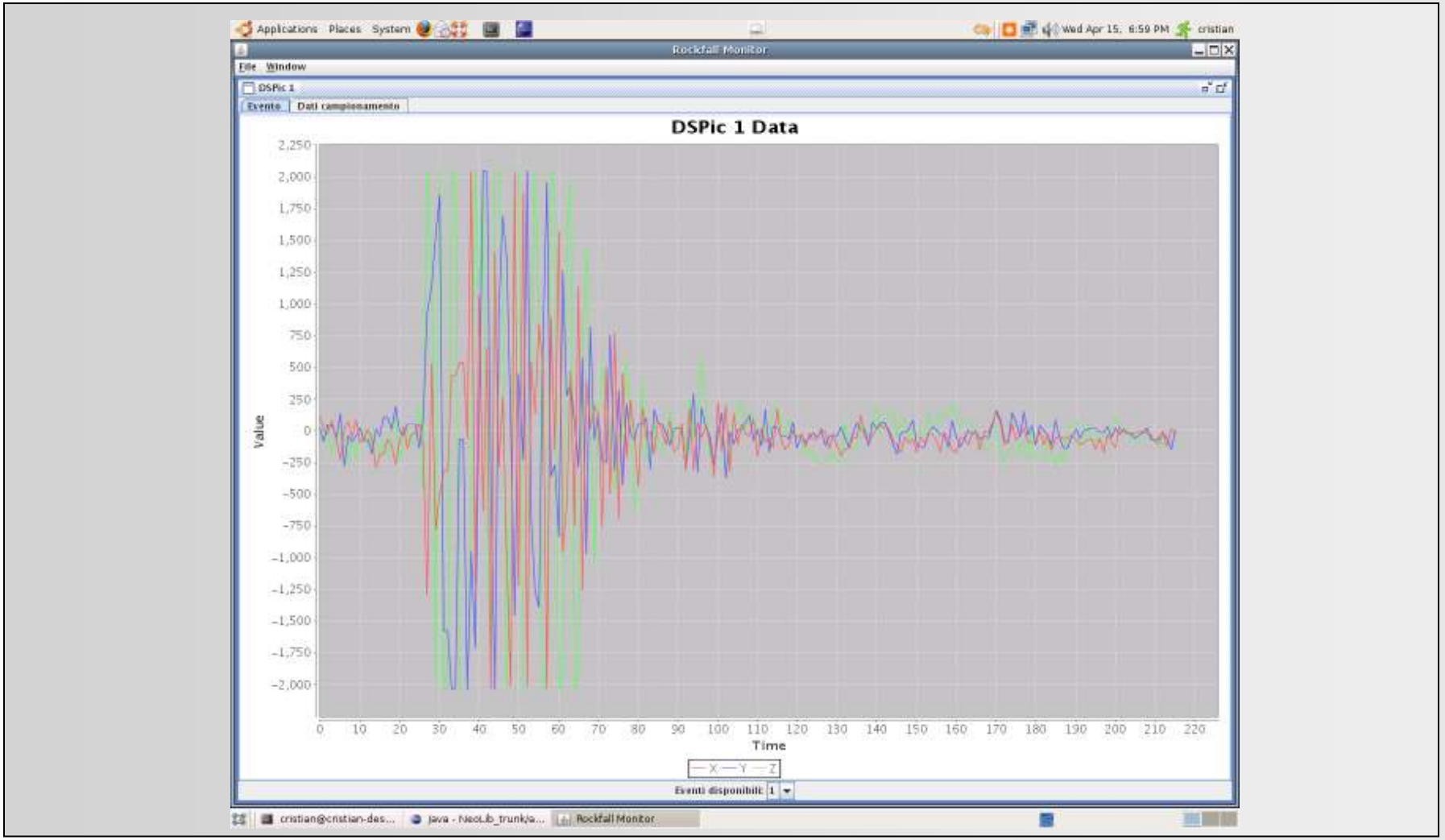


Campus Point with the control room @ 2.5Km

SYSTEM ARCHITECTURE: TODAY



SOFTWARE FRONT-END



PerLa
PERvasive LAnguage

PerLa: LANGUAGE and MIDDLEWARE
Schreiber et al. 14/05/2009

POLITECNICO MILANO

TESTBED QUERY

```
SET
    acquisitionType = 0x1,
    command = 0x13,
    rs1 = 0x3,
    rs2 = 0x4,
    taps = NEW(CONSTANTVECTORINTEGER, “3, 5, 10, 23, 1, 43”)
ON 3, 4, 5
```

FUTURE WORKS

- **Communication protocols**
  - Integration with content-dependent routing protocols
- **Selflet**
  - Similarities and differences with PerLa
  - Possible integration scenarios
- **Context Awareness**
  - Exploitation of the existing commands (Pilot Join, Execute If)
- **NanoQueries**

NANOQUERIES

- Every node is capable of simple, atomic operations
  - Operations are common among different nodes
  - Simple operations can be composed to perform more complex computations
  - LLQs can be executed directly from the nodes

- Complex mathematical functions can be easily defined with NanoQueries through operation composition

THANK YOU

Questions?

BIBLIOGRAPHY

- [1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk and J. Anderson, "Wireless sensor networks for habitat monitoring", in Proceedings of ACM international workshop on Wireless sensor networks and applications, pp. 88–97, (2002).
- [2] C. Hartung, R. Han, C. Seielstad and S. Holbrook, "FireWxNet: a multitiered portable wireless system for monitoring weather conditions in wildland fire environments", in Proceedings of International conference on Mobile systems, applications and services, pp. 28–41 (2006).
- [3] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradoffs and early experiences with zebranet", in Arthicetrual Support for Programming Languages and Operating Systems (ASPLOS 2002), October 2002.
- [4] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network", Wireless Sensor Networks, Proceeedings, pp. 108–120 (2005).
- [5] D. Chu, L. Popa, A. Tavakoli, J. Hellerstein, P. Levis, S. Shenker, and I. Stoica, "The design and implementation of a declarative sensor network systems," T.R. UCB/EECS-2006-132, pp. 1–14, 2006.
- [6] D. Chu, L. Popa, A. Tavakoli, J. Hellerstein, P. Levis, S. Shenker, and I. Stoica, "The design and implementation of a declarative sensor network systems," T.R. UCB/EECS-2006-132, pp. 1–14, 2006.
- [7] K. Aberer, M. Hauswirth, and A. Salehi, "A middleware for fast and flexible sensor network deployment," Proceedings of the 32nd international conference on Very large data bases, pp. 1199–1202, 2006.
- [8] Siemens Sword: internal communication