STUDIO ED IMPLEMENTAZIONE DI UN SISTEMA DI INTERFACCIAMENTO “PLUG & PLAY” DI DISPOSITIVI AD UNA RETE DI SENSORI

Valerio PONTE matr. 700017
Relatore: Prof. Fabio A. Schreiber
Correlatore: Ing. Romolo Camplani

INTRODUZIONE

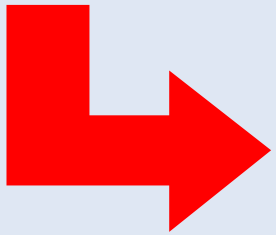
- Crescente diffusione dei sistemi pervasivi
  - WSN
- Diverse problematiche
  - Tecnologia
    - Gestione energetica
    - Comunicazione
  - Informatica
    - Eterogeneità dei dispositivi
    - Gestione dei dati prodotti
    - Telecontrollo e modifica parametri

INTRODUZIONE

- Diverse Problematiche
  - Tecnologia
  - Informatica
    - Eterogeneità dei dispositivi
    - Gestione dei dati prodotti
    - Telecontrollo e modifica parametri

INTRODUZIONE

- Diverse Problematiche
  - Tecnologia
  - Informatica
    - Eterogeneità dei dispositivi
    - Gestione dei dati prodotti
    - Telecontrollo e modifica parametri

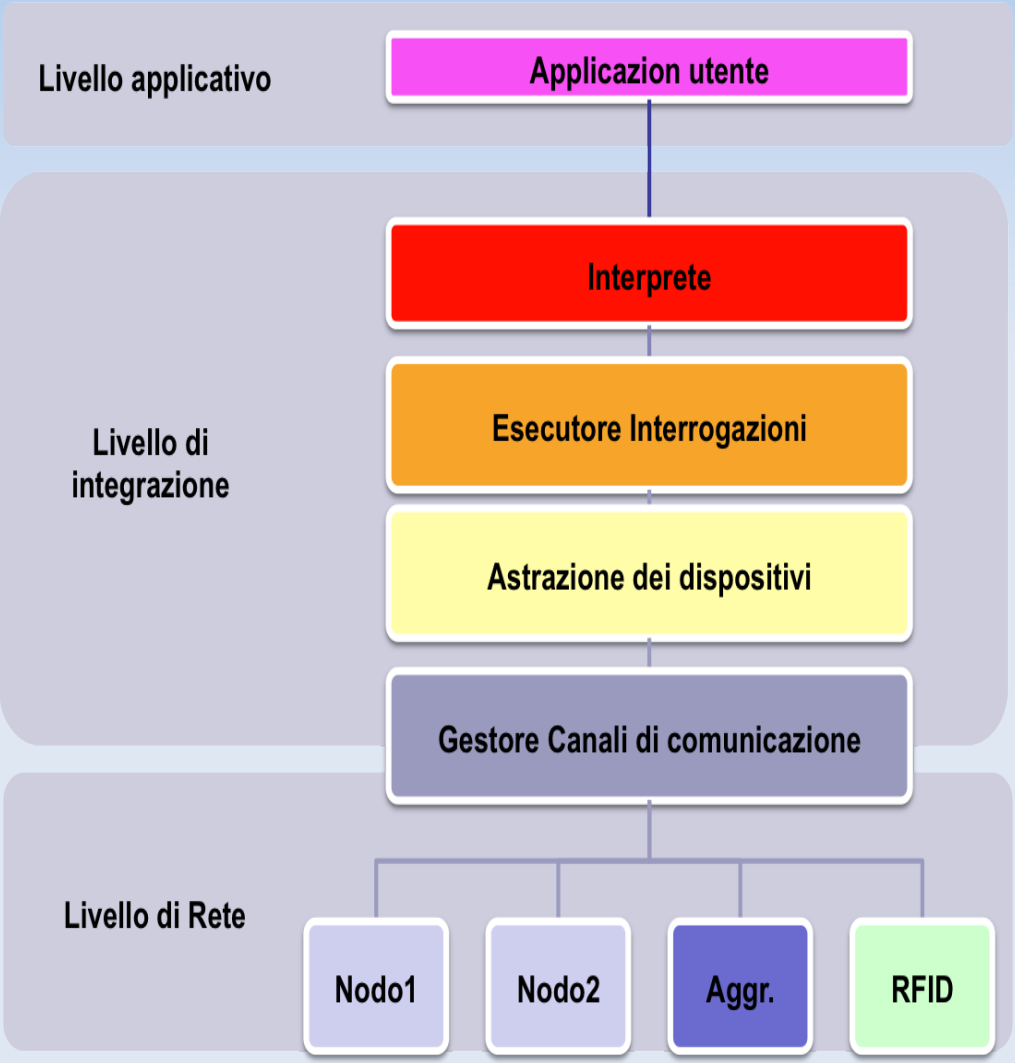


Progetto PerLa

ANALISI CRITICA

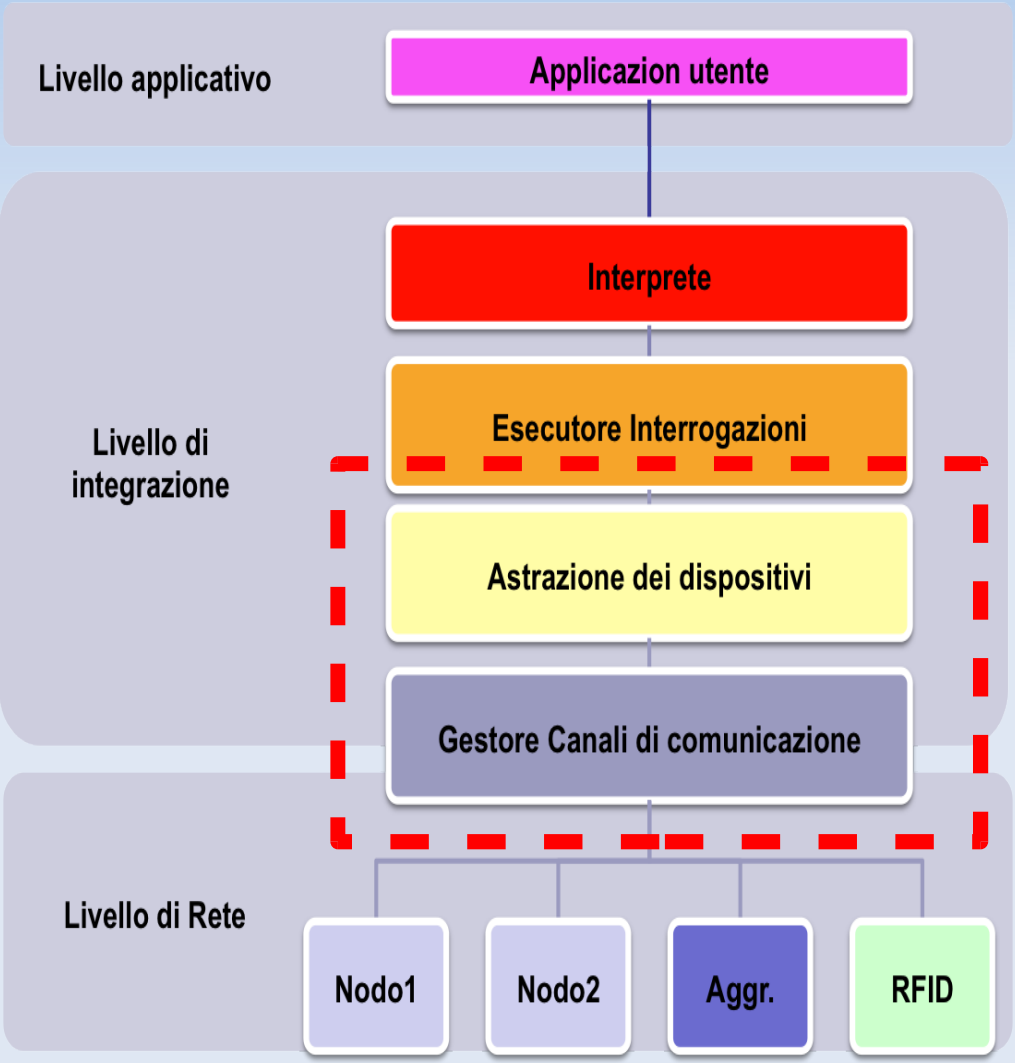
	TinyDB	GSN	DSN	SWORD	PERLA
Supporto Per L'acquisizione dei dati	✔	✗	✔	✗	✔
Supporto per la configurazione della rete	--	✗	✗	✗	✔
Aggregazione dei dati	✔	--	✔	✗	✔
Integrazione ad alto Livello	✔	✔	✔	✔	✔
Riusabilità	--	✔	--	✔	✔
Supporto a basso livello	✔	✗	✔	✗	✔
Supporto per reti eterogenee	✗	✔	✗	✔	✔

ARCHITETTURA PERLA



- PerLa integra un sistema pervasivo in un sistema informativo più ampio
- PerLa è costituito da linguaggio di interrogazione e da middleware
- Middleware
  - Provvede un astrazione unica a dispositivi eterogenei
  - Supporta l'esecuzione delle query

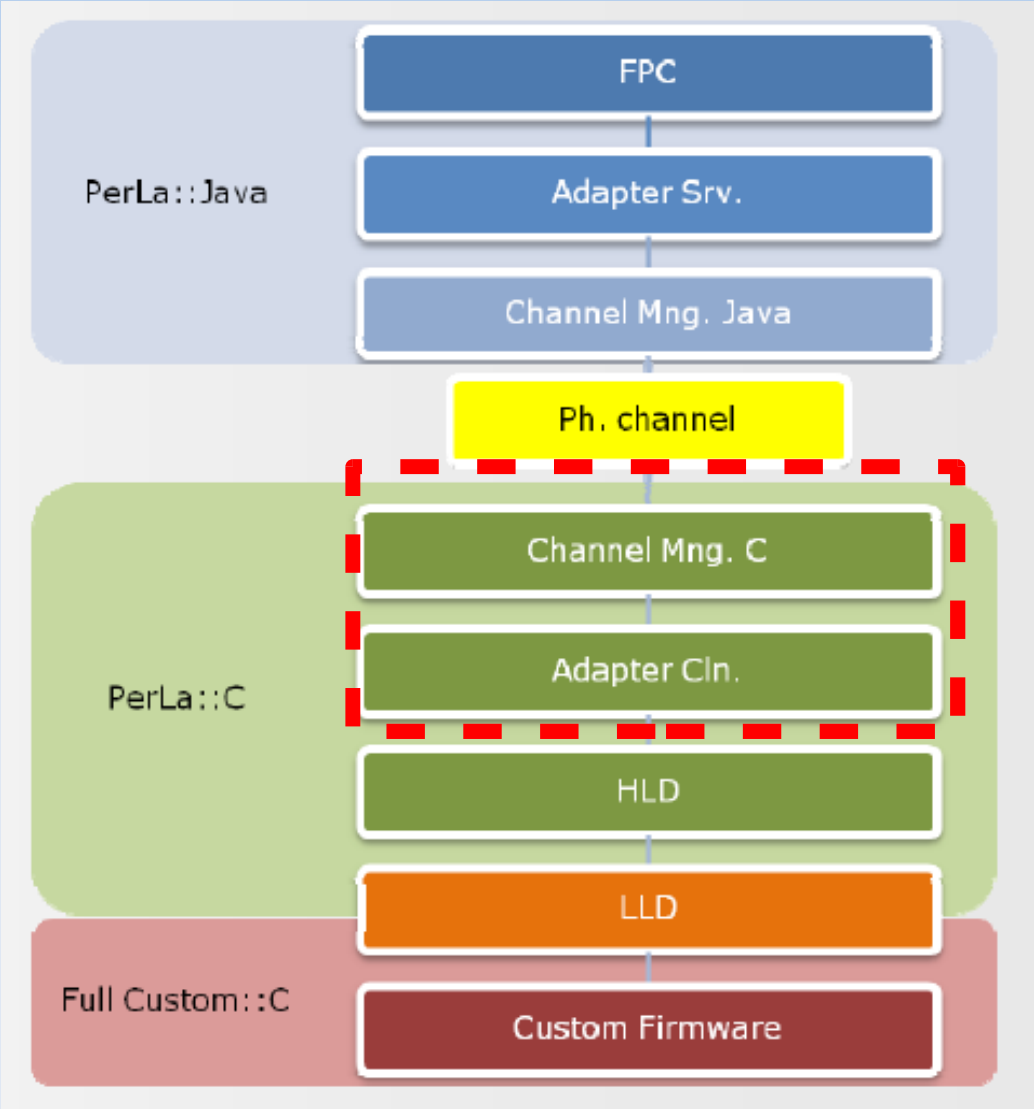
ARCHITETTURA PERLA



- PerLa integra un sistema pervasivo in un sistema informativo più ampio
- PerLa è costituito da lingu
  da n
- Mid
  - F
    u
    eterogenei
  - Supporta l'esecuzione delle query

Le due parti sviluppate:
• Comunicazione tra nodi e sistema
• Registrazione di nuovi nodi al sistema

INTERFACCIA DI LIVELLO FISICO



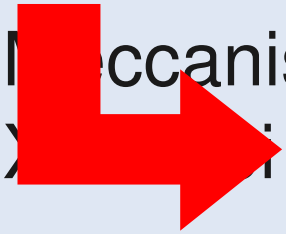
- Gestione di diversi canali/protocolli di comunicazione
- Abilitazione alla comunicazione di diversi dispositivi
- Due componenti
  - Channel Manager – gestisce il canale di comunicazione
  - Adapter Client – multiplexer

FASE DI BINDING

- Problema
  - Come aggiungere una nuova classe di dispositivi al sistema?
    - Nuovi tipi di dati
    - Formato dei dati
    - Parametri supportati
- Soluzione
  - Meccanismo Plug & Play basato su una descrizione XML dei dispositivi

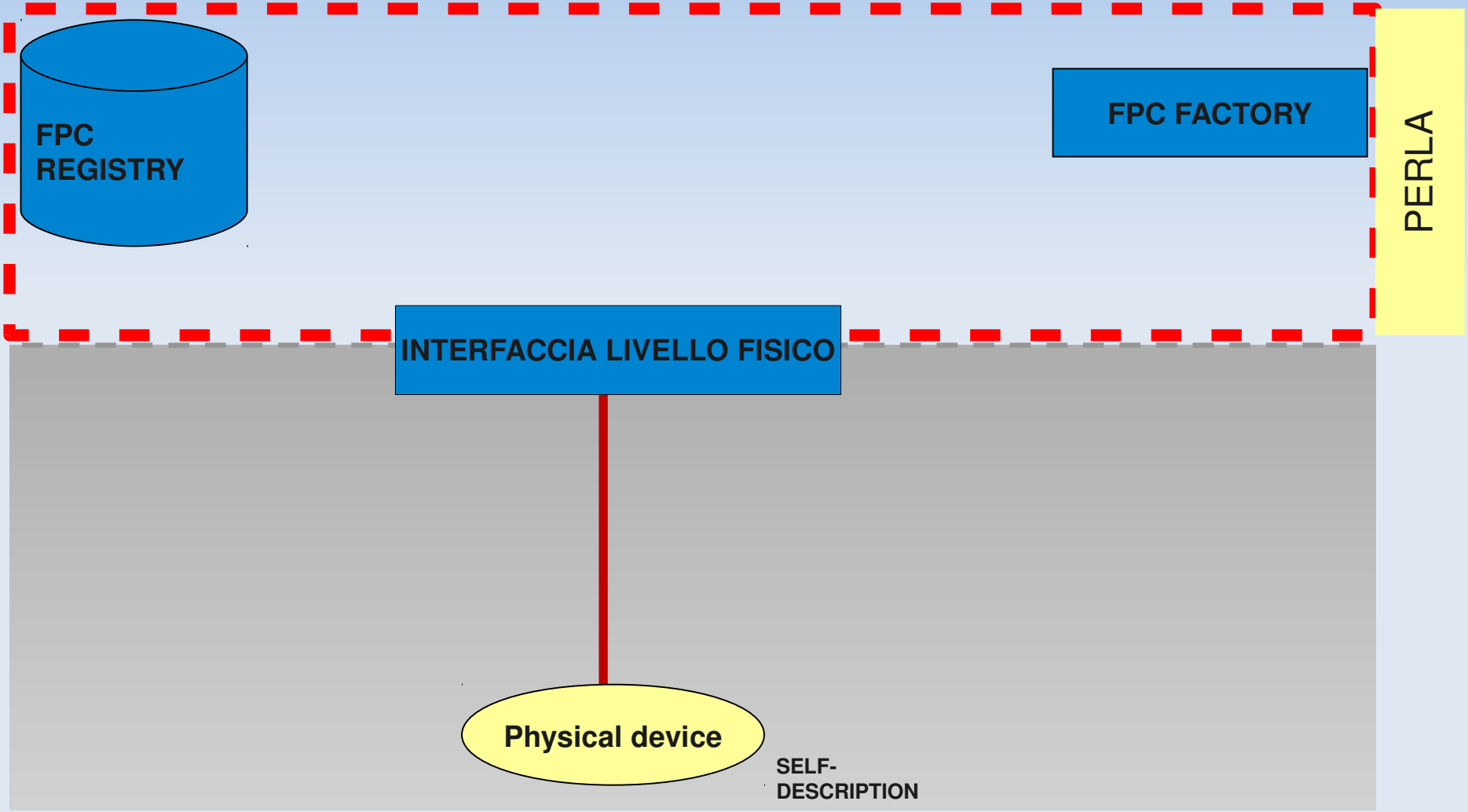
FASE DI BINDING

- Problema
  - Come aggiungere una nuova classe di dispositivi al sistema?
    - Nuovi tipi di dati
    - Formato dei dati
    - Parametri supportati
- Soluzione
  - M[illegible]eccanis[illegible]izione
    X[illegible]i C[illegible]

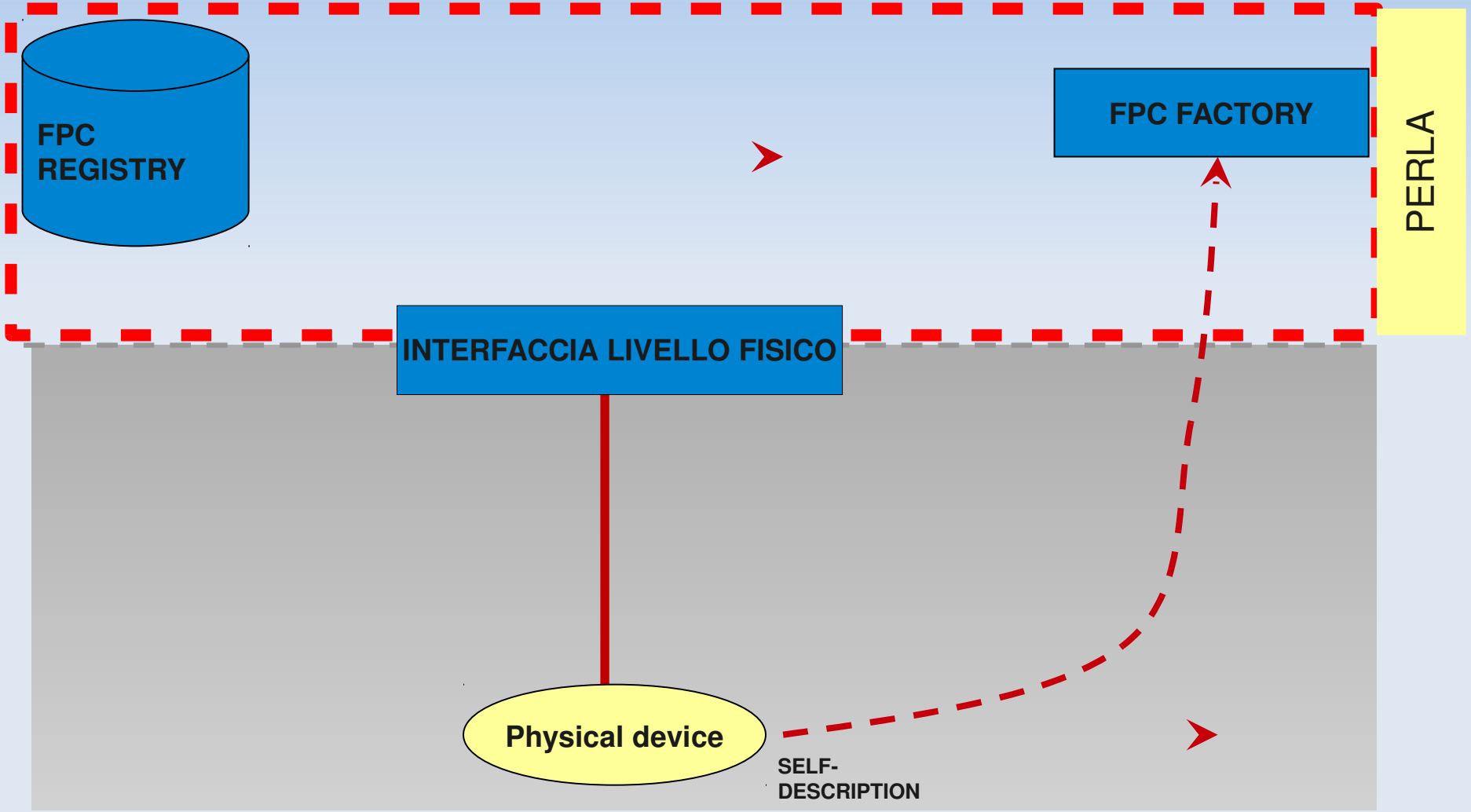


- ✓ Non richiede scrittura di nuovo codice
- ✓ Meccanismo automatico e trasparente

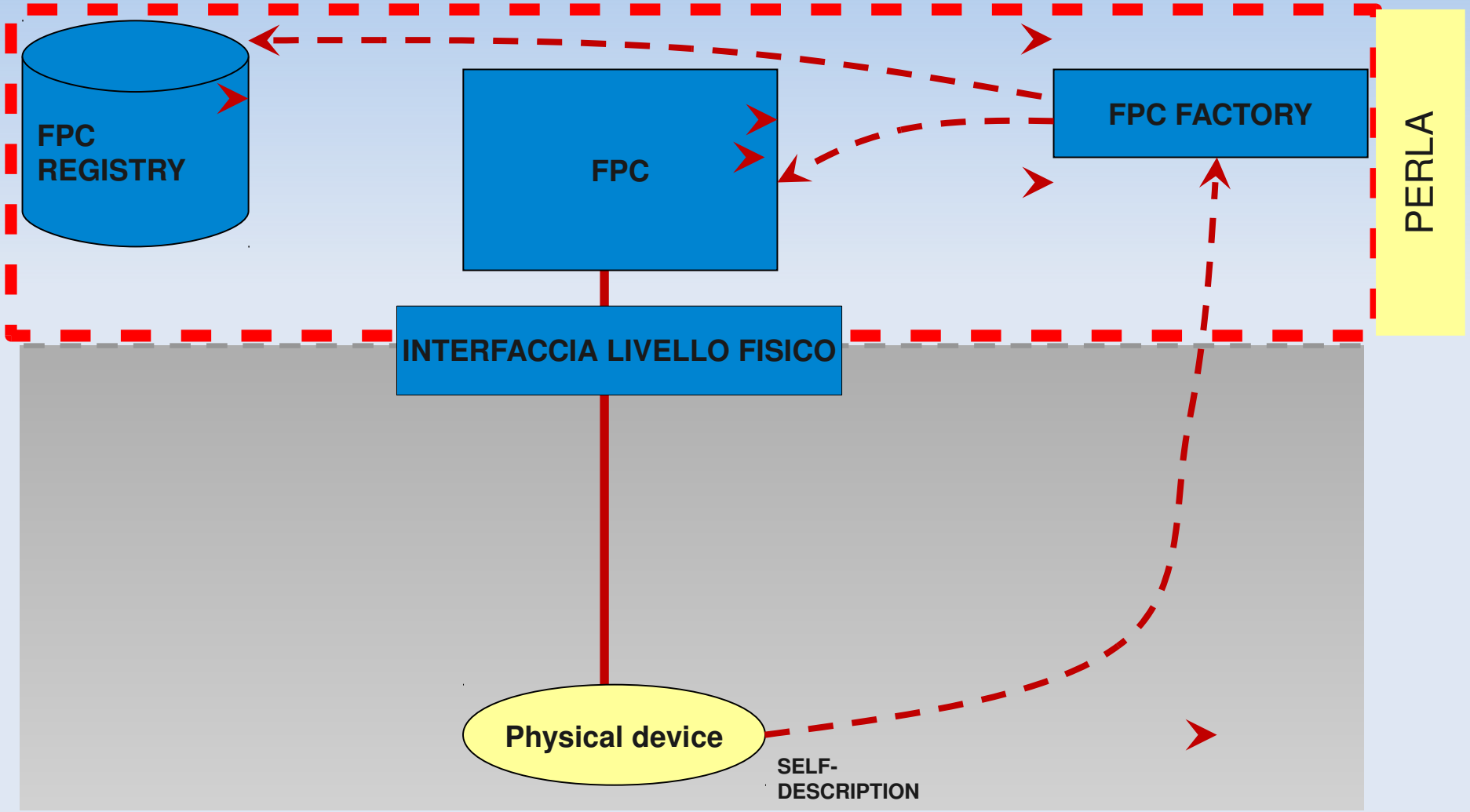
MECCANISMO DI PLUG & PLAY



MECCANISMO DI PLUG & PLAY



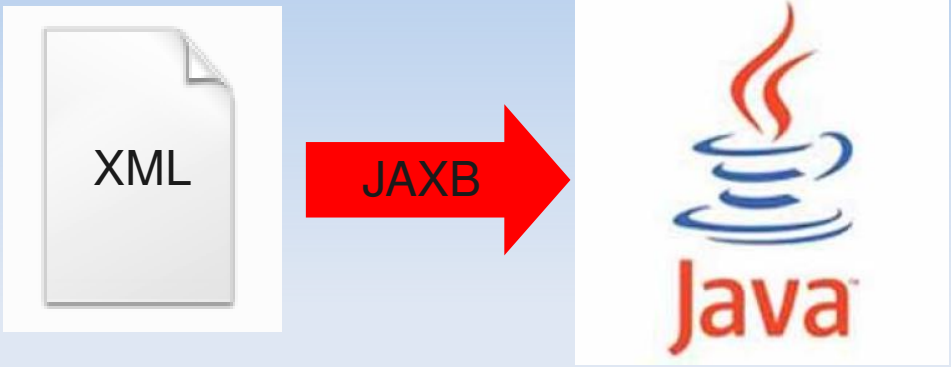
MECCANISMO DI PLUG & PLAY



PLUG & PLAY – FPC FACTORY

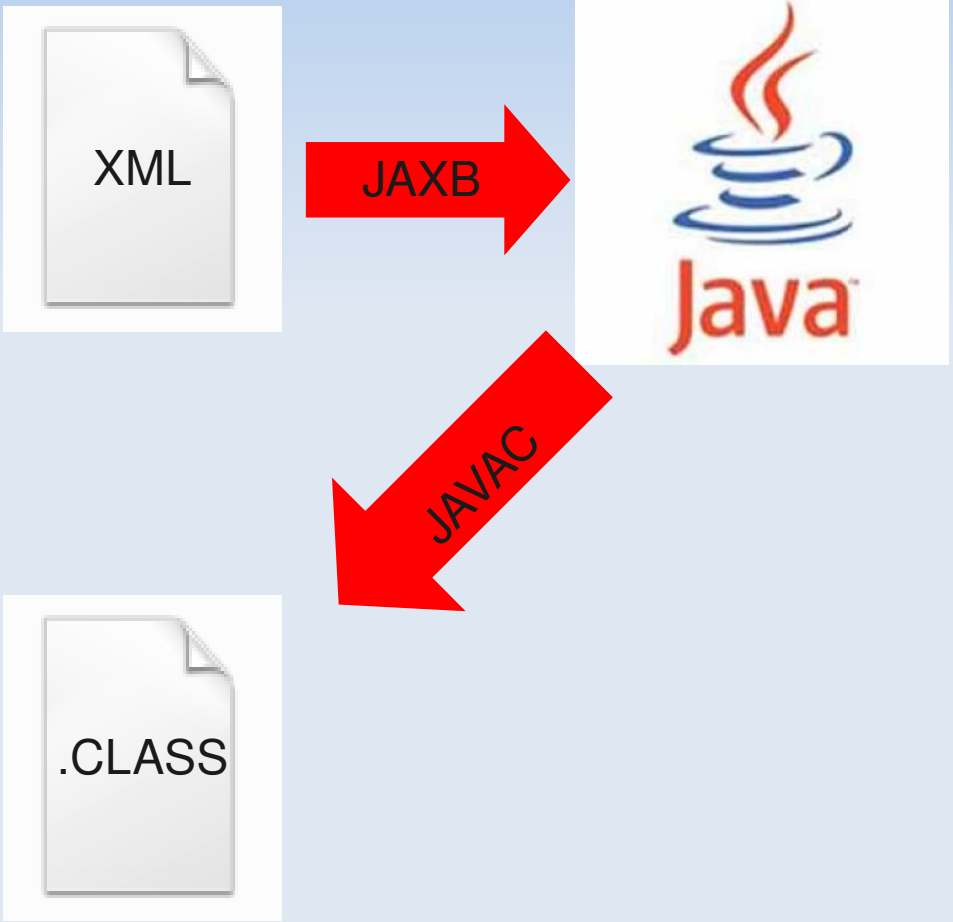
- Descrittore XML **validato** da uno Schema XML precedentemente definito

PLUG & PLAY – FPC FACTORY



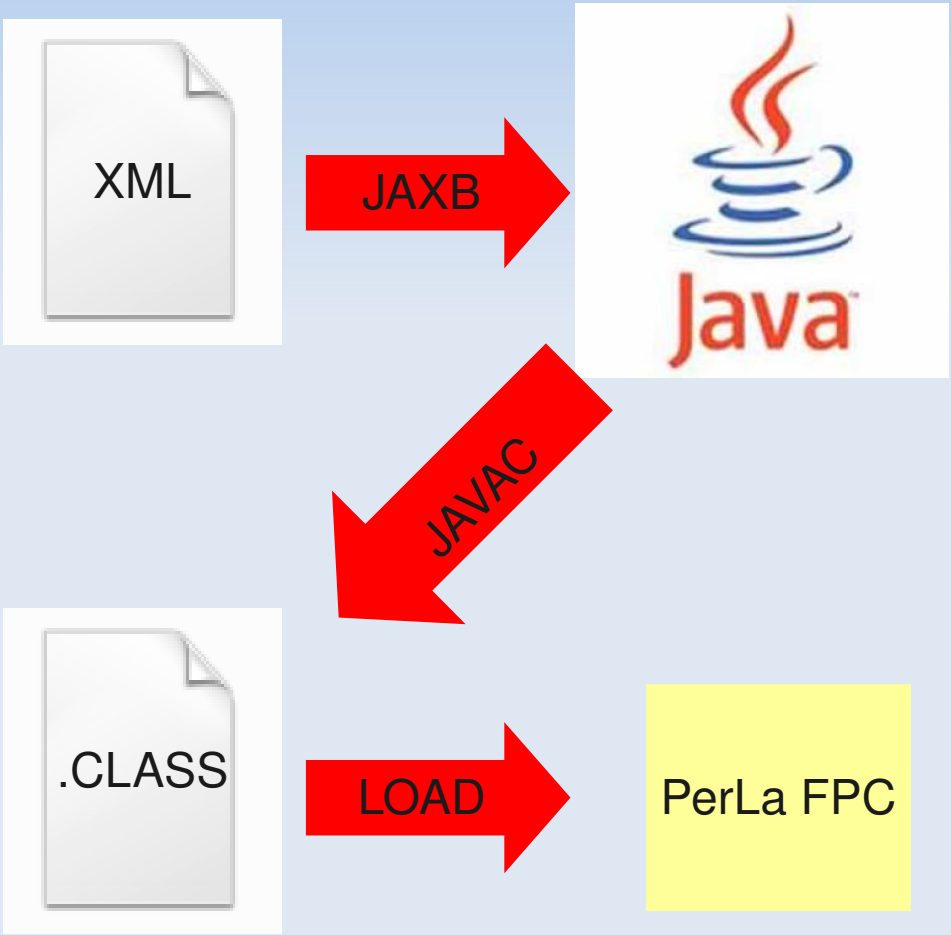
- Descrittore XML **validato** da uno Schema XML precedentemente definito
- Run time mapping XML – Java
  - JAXB

PLUG & PLAY – FPC FACTORY



- Descrittore XML **validato** da uno Schema XML precedentemente definito
- Run time mapping XML – Java
  - JAXB
- Compilazione codice generato

PLUG & PLAY – FPC FACTORY



- Descrittore XML **validato** da uno Schema XML precedentemente definito
- Run time mapping XML – Java
  - JAXB
- Compilazione codice generato
- Wrapping

PLUG & PLAY – MAPPING

```
<perlaDeviceElement name="esempio">
 <perlaSingleDevice nodeId="1">
  <parameterStructure name="e">
   <parameterElement name="param">
    <length>2</length>
    <type nameType="int">
     <sign>signed</sign>
    </type>
   </parameterElement>
   <type>EsempioXML</type>
   <size>2</size>
   <endianess>BigEndian</endianess>
  </parameterStructure>
 </perlaSingleDevice>
</perlaDeviceElement>
```

```
package org.dei.perla.sys.device.fpc.esempio;

/* IMPORT */

@StructInfo(endianness =
Endianness.BIG_ENDIAN, totalStructSize = 2)
public class EsempioXML extends AbstractData{

 public EsempioXML() {
  super();
 }

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

 public int getparam() {
  return param;
 }

 public void setparam(int param) {
  this.param = param;
 }
}
```

PLUG & PLAY – MAPPING

```
<perlaDeviceElement name="esempio">
 <perlaSingleDevice nodeId="1">
  <parameterStructure name="e">
   <parameterElement name="param">
    <length>2</length>
    <type nameType="int">
     <sign>signed</sign>
    </type>
   </parameterEl
   <type>Esempio
   <size>2</size
   <endianess>Bi
  </parameterStr
 </perlaSingleDe
</perlaDeviceEle
```

```
package org.dei.perla.sys.device.fpc.esempio;

/* IMPORT */

@StructInfo(endianness =
Endianness.BIG_ENDIAN, totalStructSize = 2)
public class EsempioXML extends AbstractData{
```

```
= Sign.SIGNED)
```

Elemento radice.
Specifica il nome del package
in cui si andranno a trovare
le classi generate

```
 public void setparam(int param) {
  this.param = param;
 }
}
```

PLUG & PLAY – MAPPING

```
<perlaDeviceElement name="esempio">
 <perlaSingleDevice nodeId="1">
  <parameterStructure name="e">
   <parameterElement name="param">
    <length>2</length>
    <type nameType="int">
     <sign>signed</sign>
    </type>
   </parameterElement>
   <type>EsempioXML</type>
   <size>2</size>
   <endianess>BigEndian</endianess>
  </parameterStructure>
 </perlaSingleDevice>
</perlaDeviceEle
```

```
package org.dei.perla.sys.device.fpc.esempio;

/* IMPORT */

@StructInfo(endianness =
Endianness.BIG_ENDIAN, totalStructSize = 2)
public class EsempioXML extends AbstractData{

 public EsempioXML() {
  super();
 }

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

 public int getparam() {
```

```
am) {
```

Una struttura dati.
Corrisponde ad una classe pubblica.
Può contenere altre strutture,
array, parametri.

PLUG & PLAY – MAPPING

```
<perlaDeviceElement name="esempio">
 <perlaSingleDevice nodeId="1">
  <parameterStructure name="e">
   <parameterElement name="param">
    <length>2</length>
    <type nameType="int">
     <sign>signed</sign>
    </type>
   </parameterElement>
   <type>EsempioXML</type>
   <size>2</size>
   <endianess>BigEndian</endianess>
  </parameterStructure>
```

Un parametro.
Viene rappresentato
come una variabile Java.

```
package org.dei.perla.sys.device.fpc.esempio;

/* IMPORT */

@StructInfo(endianness =
Endianness.BIG_ENDIAN, totalStructSize = 2)
public class EsempioXML extends AbstractData{

 public EsempioXML() {
  super();
 }

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

 public int getparam() {
  return param;
 }

 public void setparam(int param) {
  this.param = param;
 }
}
```

CONCLUSIONI

- Il sistema sviluppato gestisce
  - Comunicazione tra nodi e sistema
  - Connessione di nuovi dispositivi
- Punti aperti
  - Rimozione di un dispositivo non gestita
  - Performance dell'FPC in una situazione reale?
  - Creazione e invio dell'XML non automatici