



From Raw Sensor Data to Semantic Web Triples

Information Flow in Semantic Sensor Networks

Nikolaos Konstantinou, Ph.D.
Autonomic and Grid Computing group
Athens Information Technology



Lecture Outline

- Introduction
- Sensor data
- Semantic web
- Context-awareness
- The GSN middleware
- Exposing sensor data as triples
- Semantic Integration using SPARQL



Introduction

- The Situation
 - Decrease in the value of sensors encourage the shift from Desktop to Ubiquitous Computing
- The Problem
 - Extracting meaning from a network of deployed sensors
- Why is this a problem?
 - Raw sensor data is useless unless properly annotated
 - Limited resources in terms of processing, storage capabilities and bandwidth
- What is the suggested solution?
 - Establish ways to automatically process and manage the data



Concepts involved (1)

- Context-awareness
 - Context-aware systems are able to sense and measure their environment and include these measurements in their behavior
- Data fusion
 - Combine information and data residing at disparate sources, in order to achieve improved accuracies and more specific inferences than could be achieved by the use of a single data source alone
 - Data fusion spans various levels, from signal and object refinement (low level) to situation and threat assessment (high level)



Concepts involved (2)

- Information Integration
 - Unify information originating from various sources in order to allow its processing as a whole
 - Obstacles include
 - Heterogeneity in source schemas and data
 - Various Technical Spaces
 - Semantic and syntactic differences
- *Semantic* Information Integration
 - The resulting integration scheme carries its semantics
- Information Merging
 - Unification of the information at the implementation level



Concepts involved (3)

- Information Aggregation
 - Report the mean value of a set of measurements
 - E.g. the average temperature of a set of temperature sensors
- Information Annotation
 - Inclusion of metadata next or in the actual data
 - E.g. ID3 tags in mp3
- *Semantic* Information Annotation
 - Unambiguously define information
 - Third parties can understand the information



Lecture Outline

- Introduction
- Sensor data
- Semantic web
- Context-awareness
- The GSN middleware
- Exposing sensor data as triples
- Semantic integration using SPARQL



Sensors and Sensor Data (1)

- Sensors are devices that measure physical properties
 - Temperature, motion, light, humidity sensors
 - Also cameras, microphones, GPS-enabled smartphones
- Sensors provide data that can be
 - Streamed data
 - Audio/Video content
 - Event-based
 - Temperature measurement
 - RFID tag read
 - Light curtain interrupt

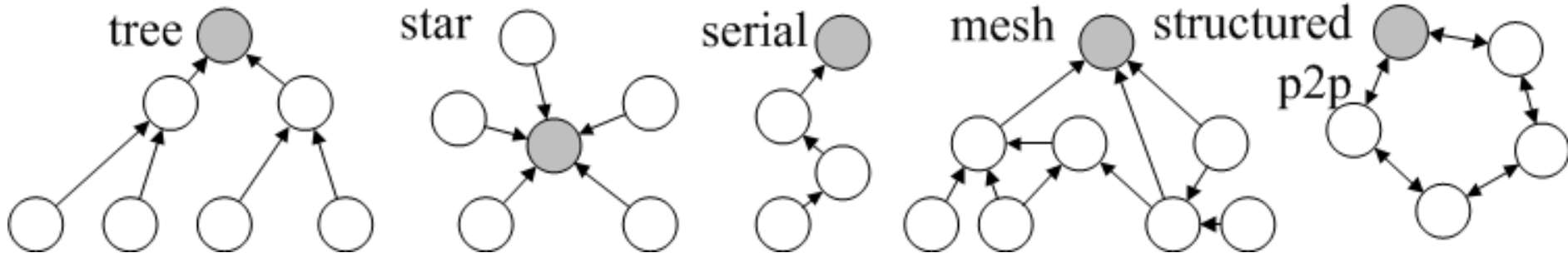


Sensors and Sensor Data (2)

- Why is sensor data any different than other forms of data, e.g. multimedia?
 - Synchronization issues
 - Apply acceptance thresholds
 - Erroneous measurements
 - Apply aggregation
 - Limited resources in the nodes
 - Caution when designing *where* the actual processing takes place
 - Keep a sliding window
 - Heavy process in the Gateway Nodes
 - Streaming may lead to packet losses
 - Reconstruct, or take decisions based on what you have

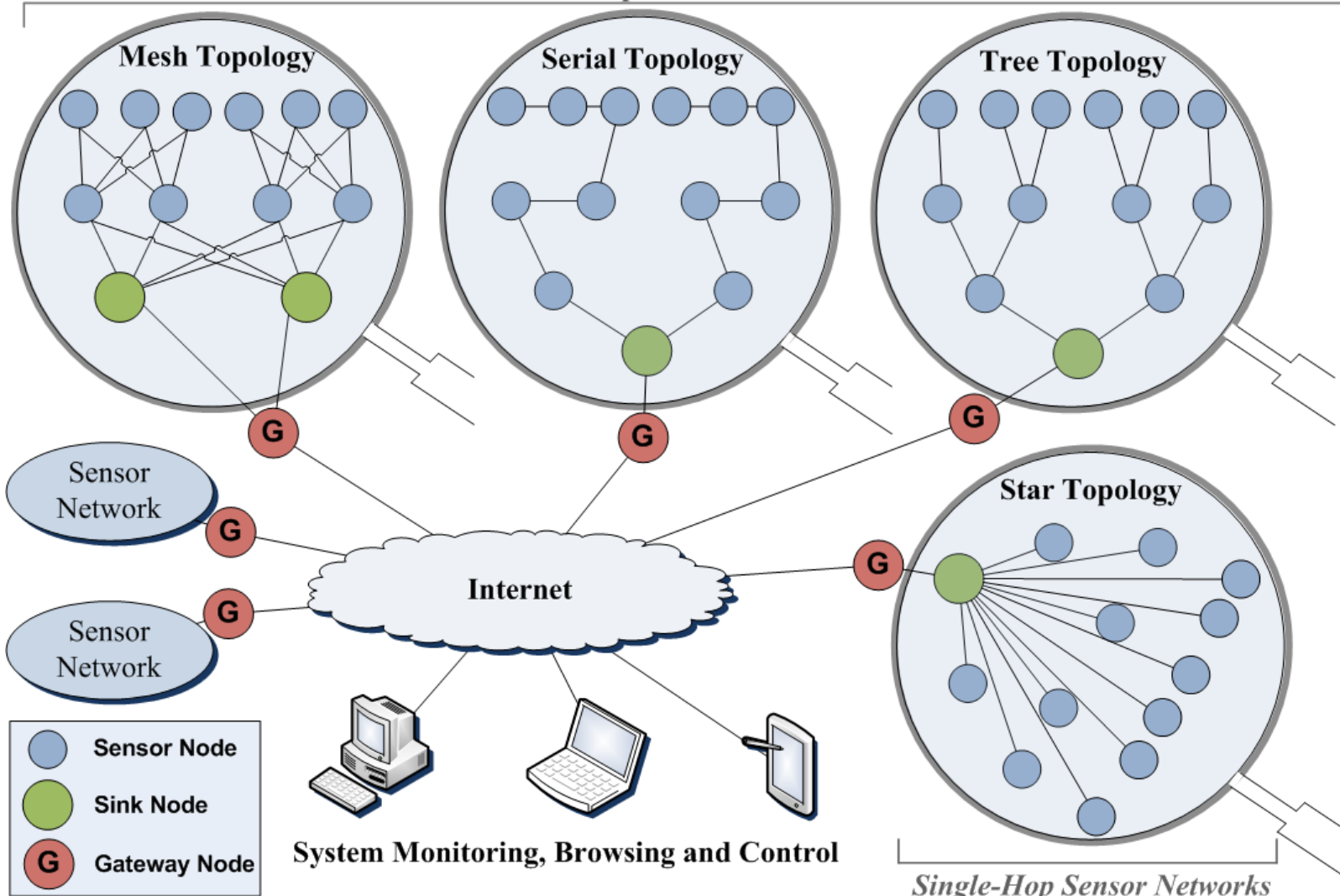
Sensor Network Topologies (1)

- Sink nodes collect information
 - Higher processing capabilities



Sensor Network Topologies (2)

Multi-Hop Sensor Networks



Single-Hop Sensor Networks



Lecture Outline

- Introduction
- Sensor Data
- **Semantic Web**
- Context-Awareness
- The GSN Middleware
- Exposing Sensor Data as triples
- Semantic Integration using SPARQL



Why Semantic Web (1)

- Knowledge in the form of a graph
 - (subject, property, object)
- Information is assigned an unambiguously defined meaning, its semantics
 - Queries can be posed by any third parties
 - Ontology, a well defined vocabulary
- Numerous ontologies already available and interconnected on the Web
 - Can and should be used when integration is a goal

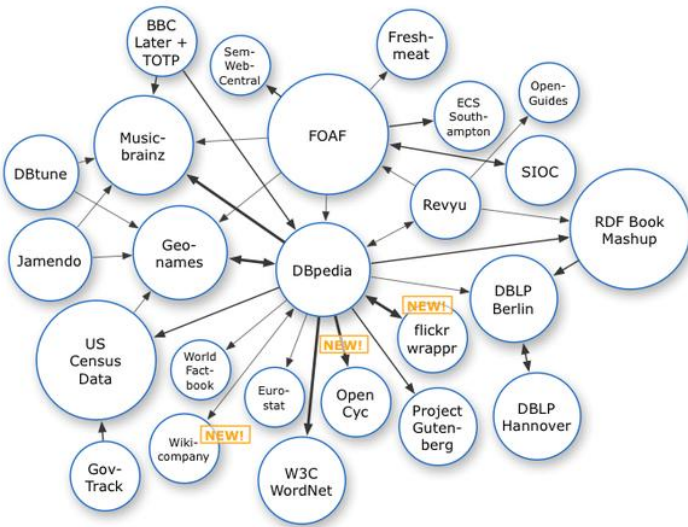




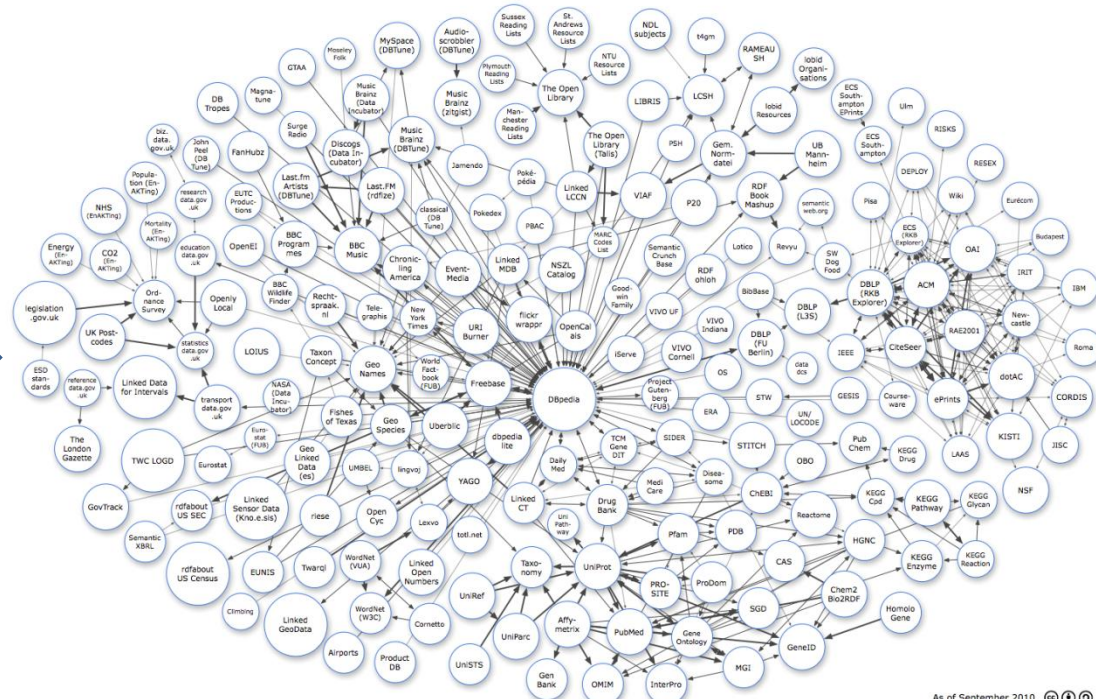
Why Semantic Web (2)

- Enables semantic annotation and integration
- Enables reasoning
 - Extract implicit information from the explicitly asserted
 - Assure concept consistency and satisfiability
- Open source tools available
 - Protégé, Jena, Virtuoso, ...
- Allows information to be exposed as Linked Open Data

The Linked Open Data Cloud



2007



2010

Source: linkeddata.org

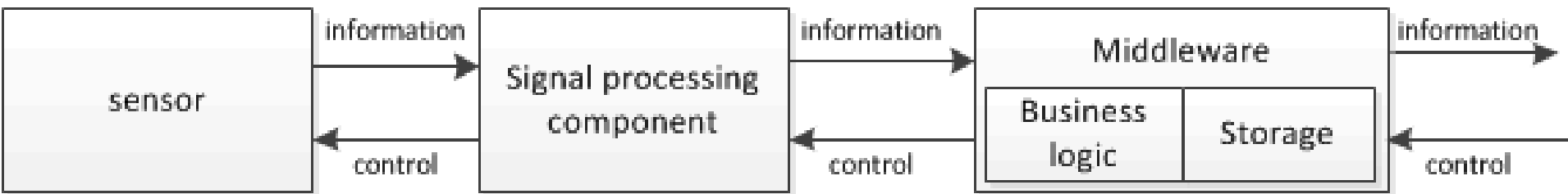
As of September 2010



Lecture Outline

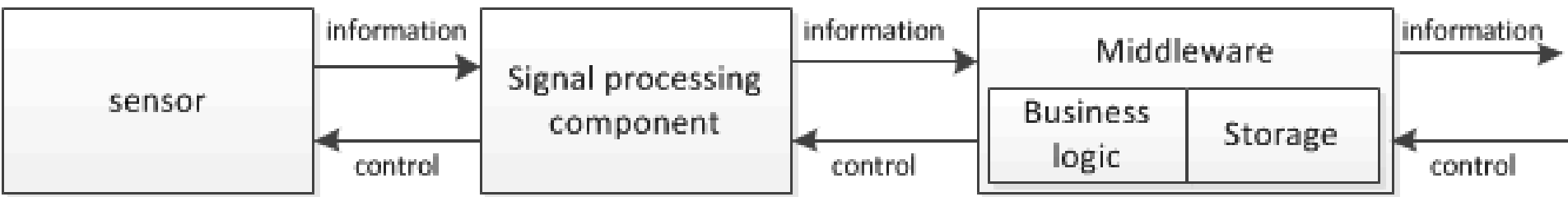
- Introduction
- Sensor data
- Semantic web
- **Context-awareness**
- The GSN middleware
- Exposing sensor data as triples
- Semantic integration using SPARQL

Context-awareness: The big picture (1)



- The sensors capture information from the environment
- The signal processing components produce structured information
 - Face detector, Body Tracker, Vehicle tracker, Smoke detection etc.

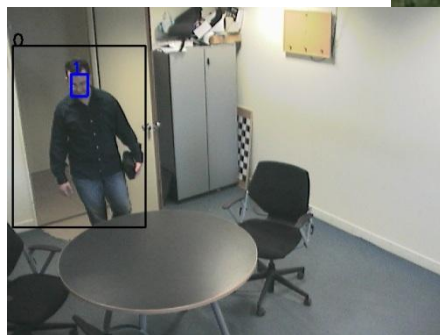
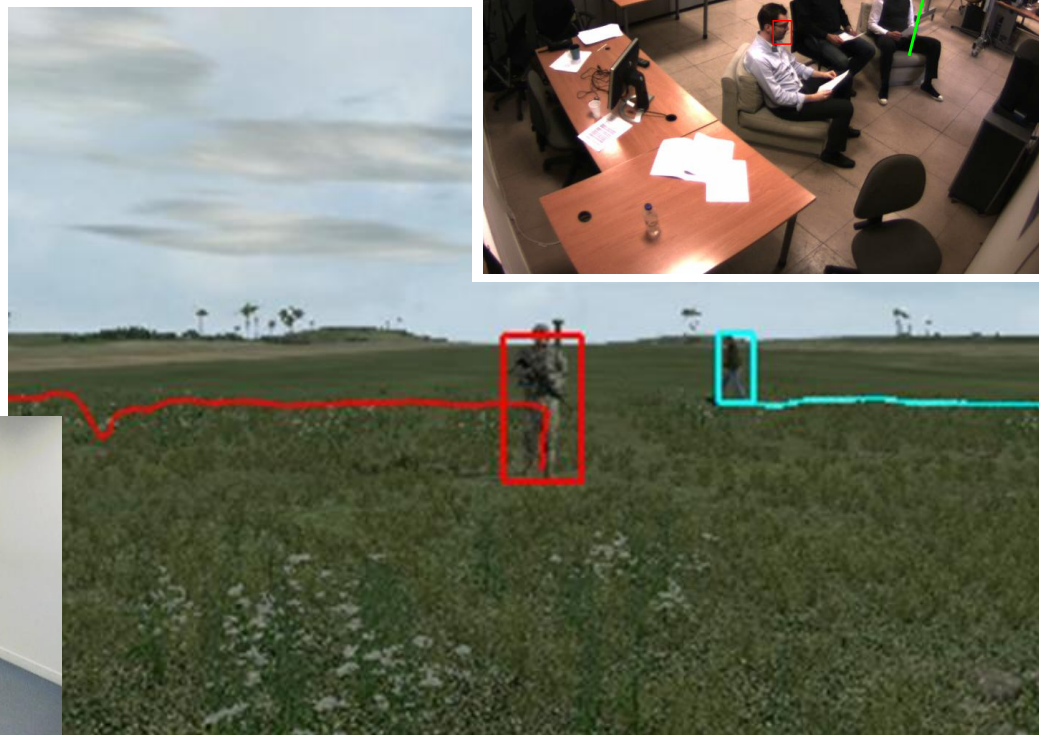
Context-awareness: The big picture (2)



- The middleware
 - Business logic: Program, configure, monitor and control system behavior
 - Storage layer
 - Support database: limited historicity, sliding window
 - Archive database: enables further processing
- Note the Information/Control duality

Signal Processing Components (1)

- Much work carried out in the AGC lab
 - Image Processing
 - Face Detection/Recognition/Tracking
 - Body Tracking
 - Audio Processing
 - A/V Localization
 - Voice activity detection





Signal Processing Components (2)

- Challenges
 - Processing is resource-hungry
 - Heterogeneous technologies must be combined
 - Production Algorithms in C++, prototypes in Matlab
 - Multidisciplinary skills required
 - Well-defined Interfaces need to be developed using RMI/Sockets/Web services/JNA
 - Video processing differs from streaming processing
 - Processing an avi file differs (greatly!) from processing an rtp stream



Lecture Outline

- Introduction
- Sensor data
- Semantic web
- Context-awareness
- **The GSN middleware**
- Exposing sensor data as triples
- Semantic integration using SPARQL



Global Sensor Networks (1)

- Open-source, java-based *middleware* solution
- Available online at sf.net/projects/gsn/
- Adaptability
 - Everything is a virtual sensor
 - Virtual sensors rely on wrappers
 - Every data producer can be integrated into the GSN with a virtual sensor and wrapper

Global Sensor Networks (2)

- Simplicity
 - Configurable without compiling source code
 - Web application for sensor management
- Scalability
 - Allows communication between nodes
 - Allows data aggregation and fusion using an SQL-like declarative language



The screenshot shows a web browser window titled "GSN Server :: GSN" with the address bar set to "localhost:22001". The page features a yellow header with a sun icon and the text "GSN Server :: GSN". Below the header, a welcome message states: "Welcome to Global Sensor Networks. The first ten sensors are displayed by default, but you can easily close them with the *close all* button. By clicking on a virtual sensors on the left sidebar, it will bring it to the top of the list." Below this message, there is a control for "Auto-refresh every : 1min" with a "refresh" button and a "close all" button. The main content area displays two sensor entries:

Real-Time	Addressing	Structure	Description	Download
temperature	28/03/2011 14:45:42 +0300			
id	3612			
temperature	23			
unit	C			

Real-Time	Addressing	Structure	Description	Download
facedetector	28/03/2011 14:45:42 +0300			
face_id	699			
time_stamp	1,22338363988222E15			
confidence	7			
name	vmil			

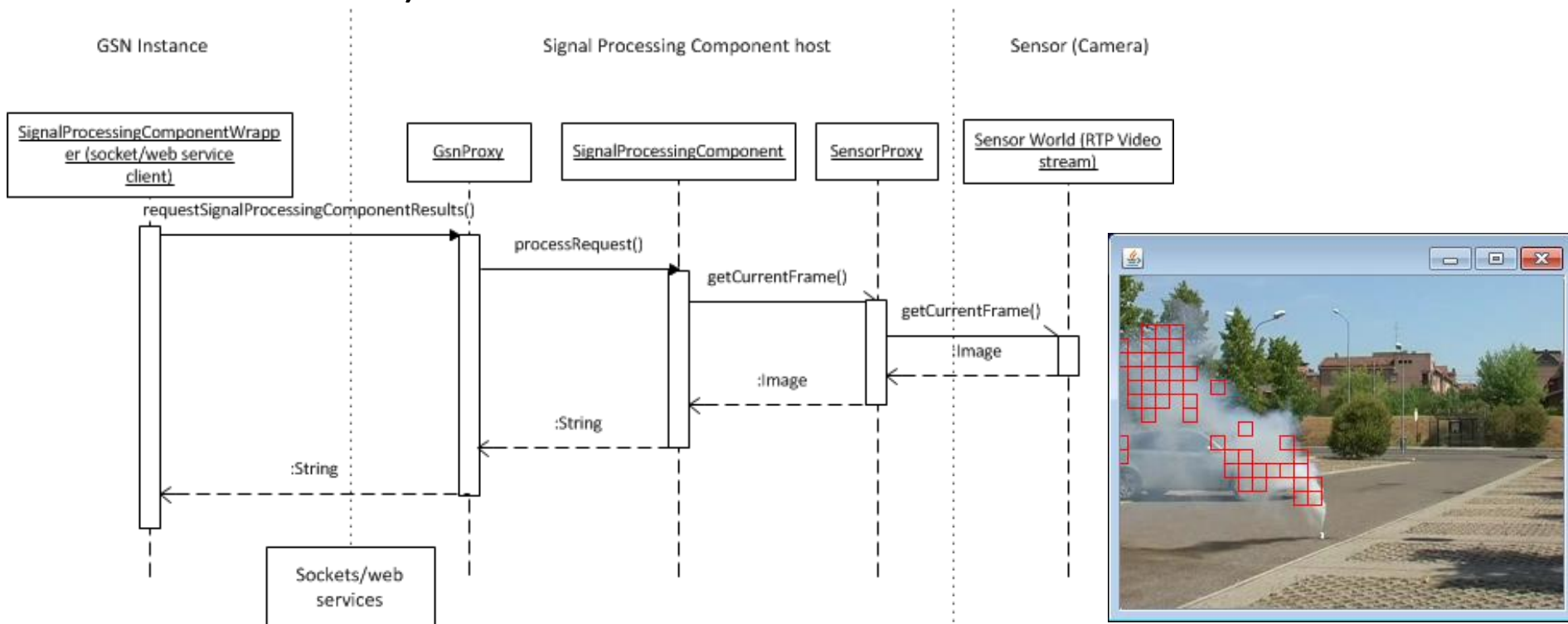
On the right side, there is a sidebar with the following information:

- Description:** Not Provided
- Author:** GSN Development Team. (my email address)
- Virtual sensors:**
 - Others
 - facedetector
 - temperature

At the bottom of the page, it says "Powered by GSN, Distributed Information Systems Lab, EPFL 2006" and includes "XHTML VALID" and "CSS VALID" buttons.

Global Sensor Networks (3)

- Example
 - Integrating a Signal Processing component (e.g. a Smoke detector) into GSN





Example of data aggregation using GSN

```
<output-structure>
```

```
...
```

```
<field name="TEMPERATURE" type="int" />
```

```
...
```

```
</output-structure>
```

```
...
```

```
<storage history-size="24h" />
```

```
...
```

```
<streams>
```

```
<stream name="input1">
```

```
<source alias="source1" sampling-rate="1" storage-size="1">
```

```
<address wrapper="temperature">
```

```
<predicate key="sampling-rate">10000</predicate>
```

```
</address>
```

```
<query>select TEMPERATURE from wrapper</query>
```

```
</source>
```

```
<query>select avg(TEMPERATURE) from source1</query>
```

```
</stream>
```

```
</streams>
```

measured
properties

sliding
window
size

data source

aggregated
output



Example of data fusion using GSN

```
<storage history-size="1h" />
<streams>
  <stream name="teststream" rate="1000">
    <source name="source1" alias="source1" storage-size="100" slide="0" sampling-rate="1">
      <address wrapper="remote-rest">
        <predicate key="HOST">localhost</predicate><predicate key="PORT">22001</predicate>
        <predicate key="QUERY">select FACE_COUNT from doorwatcher</predicate>
      </address>
      <query>select FACE_COUNT AS S1 from wrapper</query>
    </source>
    <source name="source2" alias="source2" storage-size="100" slide="0" sampling-rate="1">
      <address wrapper="remote-rest">
        <predicate key="HOST">localhost</predicate><predicate key="PORT">22002</predicate>
        <predicate key="QUERY">select TAG from touchatag</predicate>
      </address>
      <query>select TAG AS S2 from wrapper</query>
    </source>
    <query>
      select source1.S1 as S1OUT, source2.S2 as S2OUT
      from source1, source2
      where source1.S1 > 0 AND source2.S2="04dddc9232580"
    </query>
  </stream>
</streams>
```

source 1

source 2

fused output



The Storage Layer (1)

- Schema according to the Virtual Sensors
 - Virtual Sensor definition example:

```
<virtual-sensor name="temperature" priority="10">
```

```
...
```

```
<output-structure>
```

```
<field name="ID" type="int" />
```

```
<field name="SENSORTIME" type="time" />
```

```
<field name="TEMPERATURE" type="double" />
```

```
<field name="UNIT" type="varchar(255)" />
```

```
</output-structure>
```



The Storage Layer (2)

- Schema according to the Virtual Sensors
 - Schema auto-generated SQL create statement:

```
CREATE TABLE `gsn1`.`temperature` (  
  `PK` bigint(20) NOT NULL AUTO_INCREMENT,  
  `timed` bigint(20) NOT NULL,  
  `ID` int(11) DEFAULT NULL,  
  `SENSORTIME` time DEFAULT NULL,  
  `TEMPERATURE` double DEFAULT NULL,  
  `UNIT` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`PK`),  
  UNIQUE KEY `temperature_INDEX` (`timed`)  
) ENGINE=MyISAM AUTO_INCREMENT=122 DEFAULT CHARSET=latin1
```

} added by GSN

} depending on the virtual sensor description



The Storage Layer (3)

- Historical data
 - According to the Virtual Sensor definition
 - Tuple-based: `<storage history-size="1" />`
 - Time-based: `<storage history-size="1m" />`
- Out-of-the-box support for
 - Mysql
 - SQL Server
 - Oracle
 - H2
 - Can be extended to support other RDBMS's

gsn1.temperature: 10 rows total (approximately)

PK	timed	ID	TEMPERATURE	UNIT
14983	1301312947745	3612	23	C
15023	1301312959138	3612	23	C
15018	1301312957746	3612	23	C
15013	1301312956379	3612	23	C
15008	1301312954970	3612	23	C
15003	1301312953579	3612	23	C
14998	1301312952170	3612	23	C
14993	1301312950700	3612	23	C
14988	1301312949223	3612	23	C
15028	1301312960725	3612	23	C



The Storage Layer (4)

- Storage Layer can be
 - Centralized (in the Central Control node)
 - Data is pushed to the Central Control node
 - Distributed (in the Gateway nodes)
- Interfaces
 - Legacy (ODBC/JDBC)
 - Web Services (SOAP or RESTful)
 - SPARQL Endpoints
 - Allow Semantic Information Integration



Lecture Outline

- Introduction
- Sensor data
- Semantic web
- Context-awareness
- The GSN middleware
- Exposing sensor data as triples
- Semantic integration using SPARQL



Exposing sensor data as triples (1)

- OpenLink Virtuoso universal server can be used as
 - A web application server
 - A relational database repository
 - A triplestore
 - A web service server
- Open-source version available at <http://virtuoso.openlinksw.com/>
- Cluster Configuration
 - Parallel and Horizontal scaling



Exposing sensor data as triples (2)

- Virtuoso RDF Views
 - Export relational data as triples
- SPARQL 1.1 support, plus
 - Full Text Queries
 - Geo Spatial Queries
 - Business Analytics and Intelligence
 - SQL Stored Procedure and Built-In Function exploitation from SPARQL
 - Create, Update, and Delete (SPARUL)
- Backward-chaining OWL reasoner

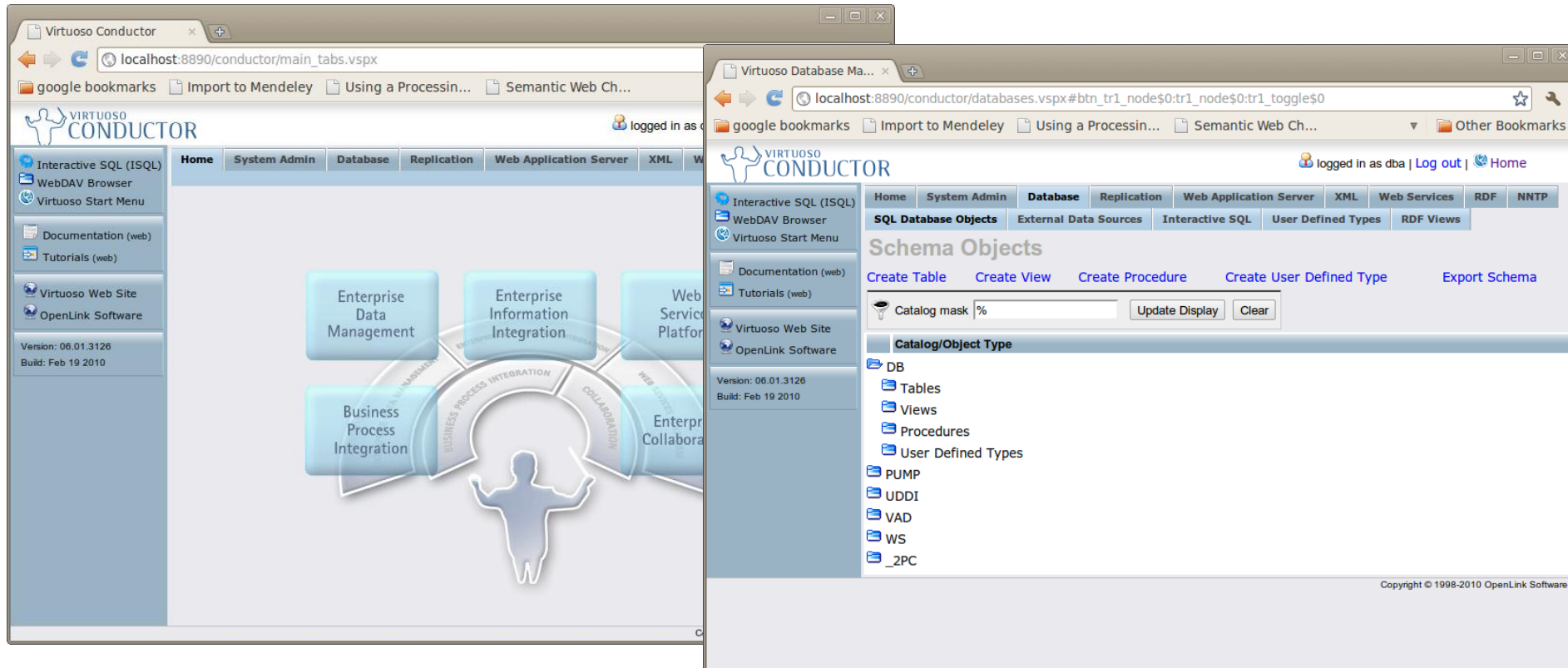


Exposing sensor data as triples (3)

- Using Virtuoso, RDF Data can also be accessible via
 - ODBC/JDBC
 - ADO.NET (Entity Frameworks compatible)
 - OLE DB
 - XMLA (XML for Analysis) data providers / drivers
- Using the “Sponger” RDF-izer, RDF data can be extracted from non-RDF sources (e.g. with XSLT)

Publishing RDF using Virtuoso (1)

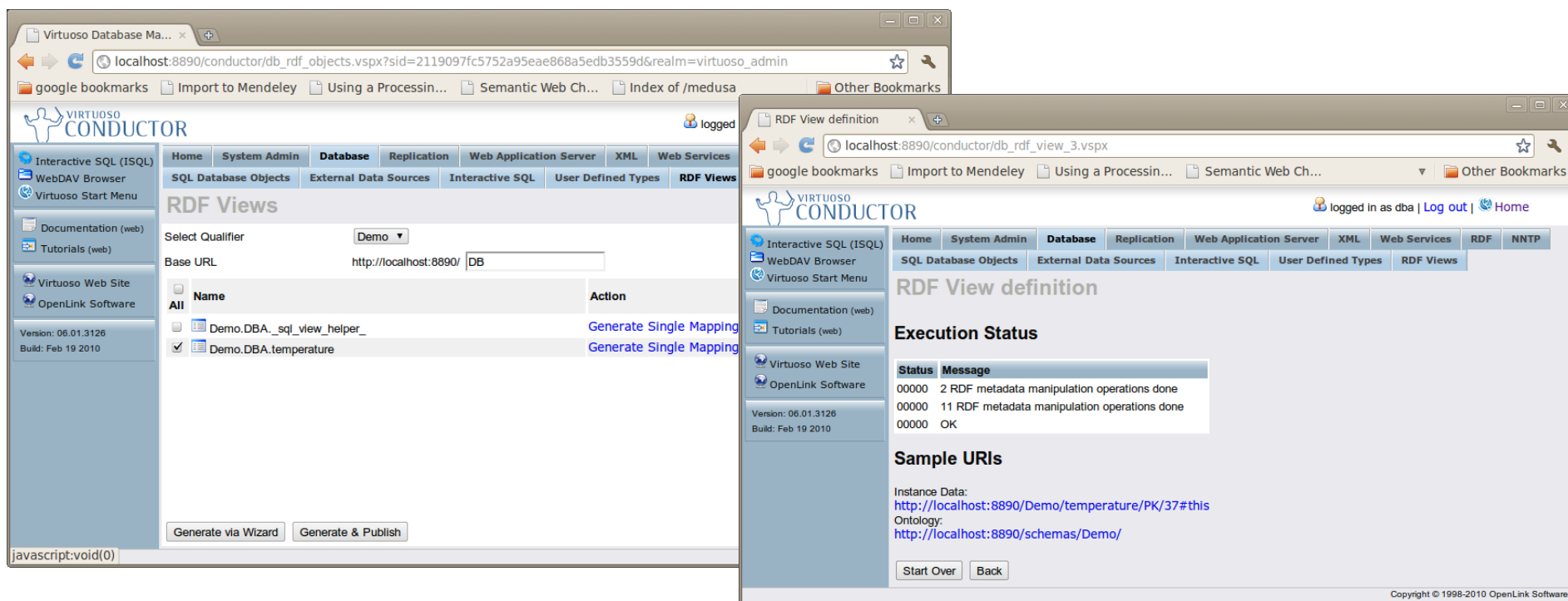
- Conductor: a GUI for server administration
- Virtuoso can be used as a DBMS



The image displays two overlapping screenshots of the Virtuoso Conductor web interface. The background screenshot shows the main dashboard with a navigation menu on the left and a central diagram illustrating various integration capabilities: Enterprise Data Management, Enterprise Information Integration, Web Service Platform, Business Process Integration, and Enterprise Collaboration. The foreground screenshot shows the 'Schema Objects' page, which is used for managing database objects. It includes a 'Catalog mask' input field and a list of object types such as DB, Tables, Views, Procedures, User Defined Types, PUMP, UDDI, VAD, WS, and _2PC. The interface is logged in as 'dba' and shows the version 06.01.3126, build Feb 19 2010.

Publishing RDF using Virtuoso (2)

- Can be combined with GSN to process sensor data streams and export them as RDF
- Create RDF Views over the relational data



The image shows two overlapping screenshots of the Virtuoso Database Manager interface. The background screenshot displays the 'RDF Views' management page, where a table lists existing views. The foreground screenshot shows the 'RDF View definition' page, which displays the execution status of a view definition.

RDF Views Table:

Name	Action
Demo.DBA._sql_view_helper_	Generate Single Mapping
<input checked="" type="checkbox"/> Demo.DBA.temperature	Generate Single Mapping

RDF View definition Execution Status:

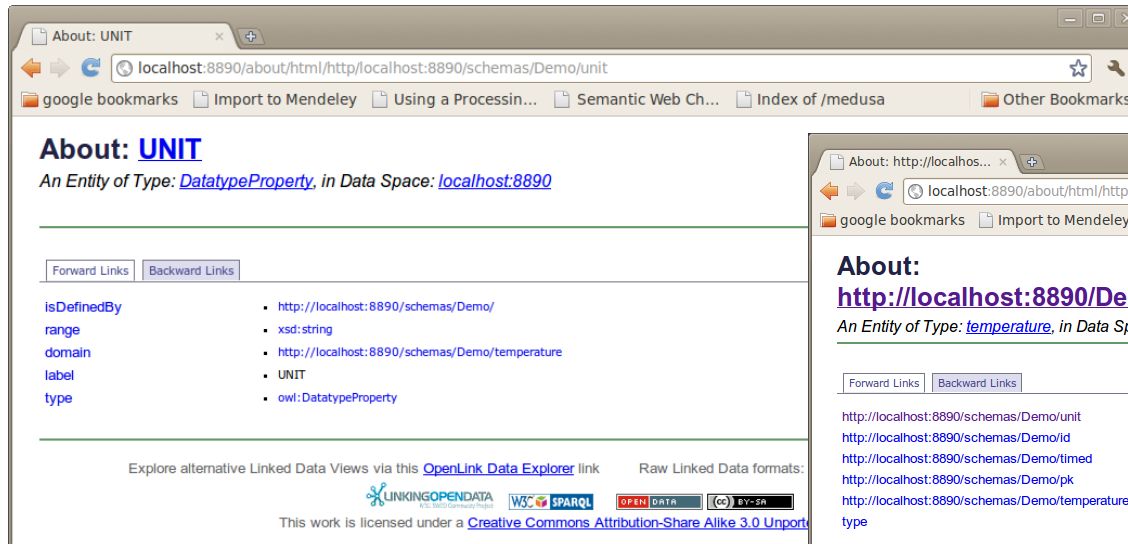
Status	Message
00000	2 RDF metadata manipulation operations done
00000	11 RDF metadata manipulation operations done
00000	OK

Sample URIs:

Instance Data:
<http://localhost:8890/Demo/temperature/PK/37#this>
Ontology:
<http://localhost:8890/schemas/Demo/>

Publishing RDF using Virtuoso (3)

- Browseable repository
- A URI for every resource
- Example: Measurement URI



About: UNIT

An Entity of Type: [DatatypeProperty](#), in Data Space: [localhost:8890](#)

Forward Links Backward Links

isDefinedBy

- <http://localhost:8890/schemas/Demo/>
- xsd:string

range

- <http://localhost:8890/schemas/Demo/temperature>

domain

- UNIT

label

- UNIT

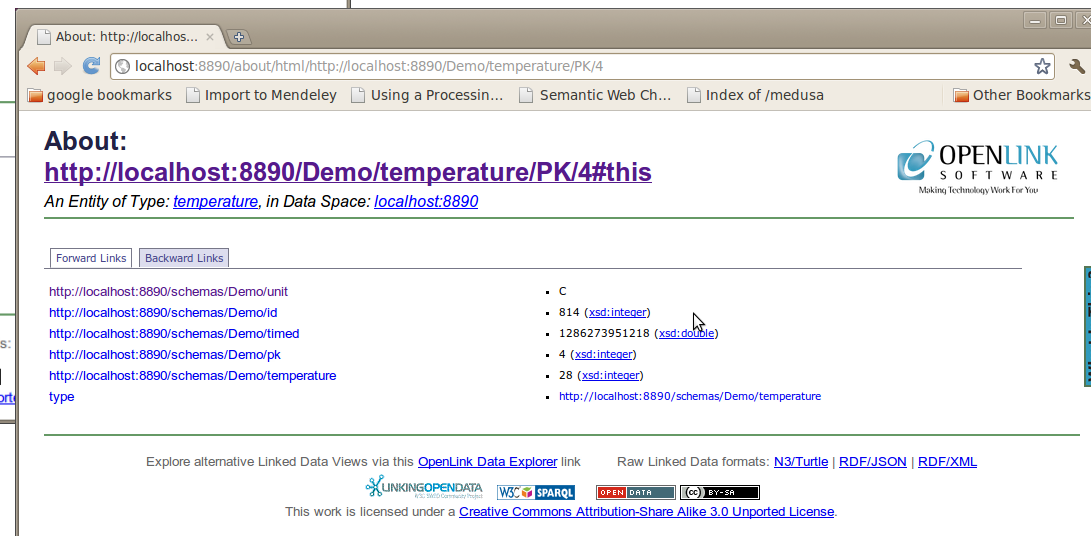
type

- owl:DatatypeProperty

Explore alternative Linked Data Views via this [OpenLink Data Explorer](#) link Raw Linked Data formats:

LINKINGOPENDATA W3C SPARQL OPEN DATA CC BY-SA

This work is licensed under a [Creative Commons Attribution-Share Alike 3.0 Unported License](#).



About: <http://localhost:8890/Demo/temperature/PK/4#this>

An Entity of Type: [temperature](#), in Data Space: [localhost:8890](#)

Forward Links Backward Links

OPENLINK SOFTWARE
Making Technology Work For You

http://localhost:8890/schemas/Demo/unit

- C
- 814 (xsd:integer)
- 1286273951218 (xsd:double)
- 4 (xsd:integer)
- 28 (xsd:integer)
- <http://localhost:8890/schemas/Demo/temperature>

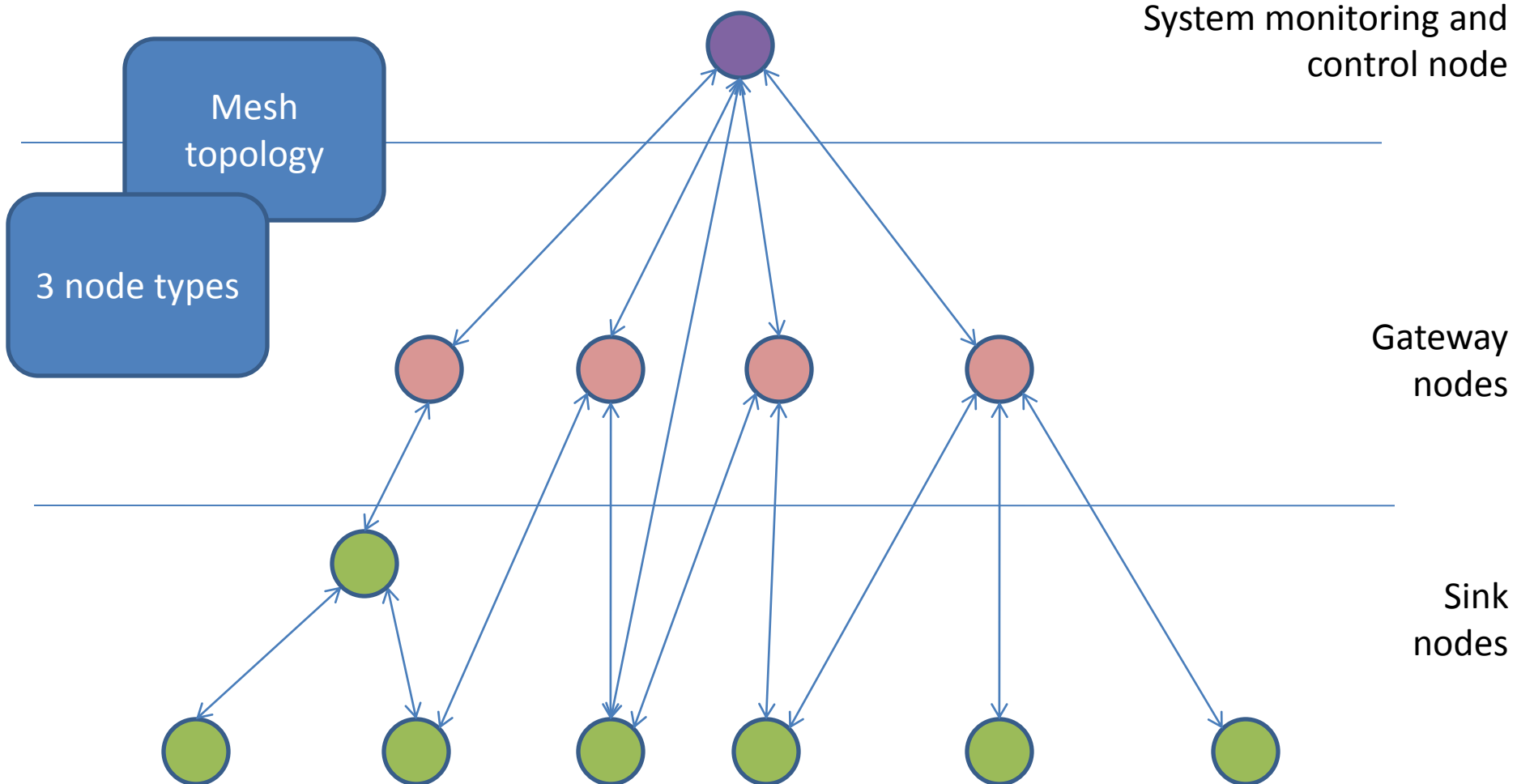
type

Explore alternative Linked Data Views via this [OpenLink Data Explorer](#) link Raw Linked Data formats: [N3/Turtle](#) | [RDF/JSON](#) | [RDF/XML](#)

LINKINGOPENDATA W3C SPARQL OPEN DATA CC BY-SA

This work is licensed under a [Creative Commons Attribution-Share Alike 3.0 Unported License](#).

Semantic Sensor Network Example (1)





Semantic Sensor Network Example (2)

- Sink node
 - Operation relies on a relational database
 - Limited historical data
 - Keep a “sliding window”
 - Based on time or tuples
 - Do not have semantic capabilities
 - One Database per Sink node



Semantic Sensor Network Example (3)

- Gateway Node
 - Operation relies on a semantically-enabled knowledge base
 - Supported by inference procedures
 - Maintains historical/archived information
 - One Knowledge Base per Gateway node
 - Appropriate for Higher Level Fusion (e.g. threats, events)

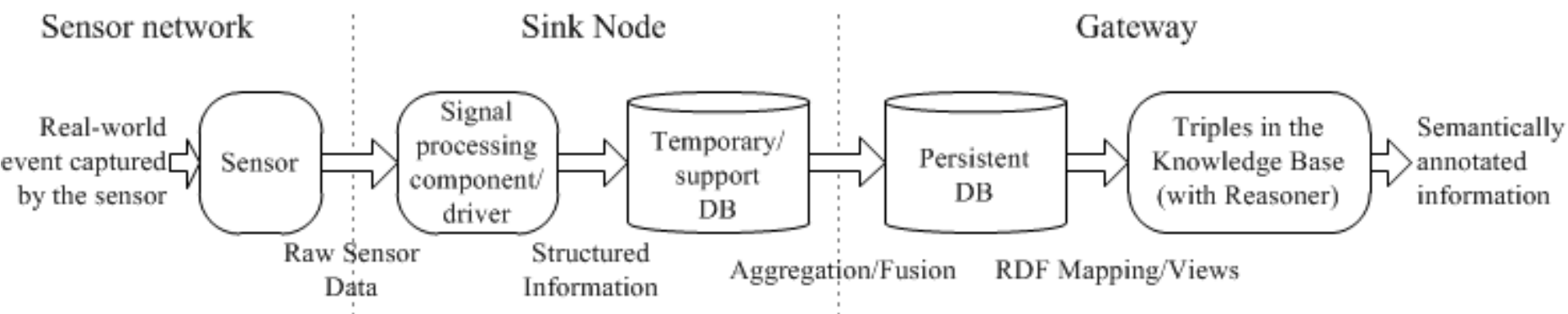


Semantic Sensor Network Example (4)

- Central Control node
 - Monitors and controls the network
 - Provides system-wide services
 - E.g. directory services, secure authentication
 - Can store its view over the network for intelligence extraction

Information Flow

- An Information flow example in a decentralized Semantic Sensor Network





Lecture Outline

- Introduction
- Sensor data
- Semantic web
- Context-awareness
- The GSN middleware
- Exposing sensor data as triples
- Semantic Integration using SPARQL



Semantic integration using SPARQL (1)

- SPARQL: An SQL-like language for querying RDF graphs
- `SELECT-FROM-WHERE` syntax
- `WHERE` conditions are triple patterns
- `SELECT ?x ?y ?z`

`WHERE`

`{ ?x ?y ?z }`

returns all the triples in the graph



Semantic integration using SPARQL (2)

- XML over HTTP (RESTful approach)
 - http://demo.openlinksw.com/sparql?default-graph-uri=urn:lsid:ubio.org:namebank:11815&should-sponge=soft&query=SELECT+*+WHERE+{?s+?p+?o}&format=text/html
- No create/update/delete capabilities



Semantic integration using SPARQL (3)

- SPARQL queries can be named and stored
 - A query named *sparql-demo* listens to:
<http://localhost:8890/DAV/sparql-demo>
- Can return results over HTTP (XML by default)
- MIME type of the RDF data
 - 'rdf+xml' (default) | 'n3' | 'turtle' | 'ttl'



Semantic integration using SPARQL (4)

- SPARQL results example in RDF/XML

```
<ROOT>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rs="http://www.w3.org/2005/sparql-results#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <rs:results rdf:nodeID="rset">
    <rs:result rdf:nodeID="sol193">
      <rs:binding rdf:nodeID="sol193-0" rs:name="x"><rs:value rdf:resource="http://localhost:8890/Demo/temperature/PK/4#this"/></rs:binding>
      <rs:binding rdf:nodeID="sol193-1" rs:name="y"><rs:value rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"/></rs:binding>
      <rs:binding rdf:nodeID="sol193-2" rs:name="z"><rs:value rdf:resource="http://localhost:8890/schemas/Demo/temperature"/></rs:binding>
    </rs:result>
  ...
</rs:results>
</rdf:RDF>
</ROOT>
```



Thank you!

Questions?