

Global Sensor Networks (GSN)

Middleware for flexible deployment of heterogeneous sensor networks

Antonio Aguilar

Digital Enterprise Research Institute
National University of Ireland, Galway

antonio.aguilar@deri.org

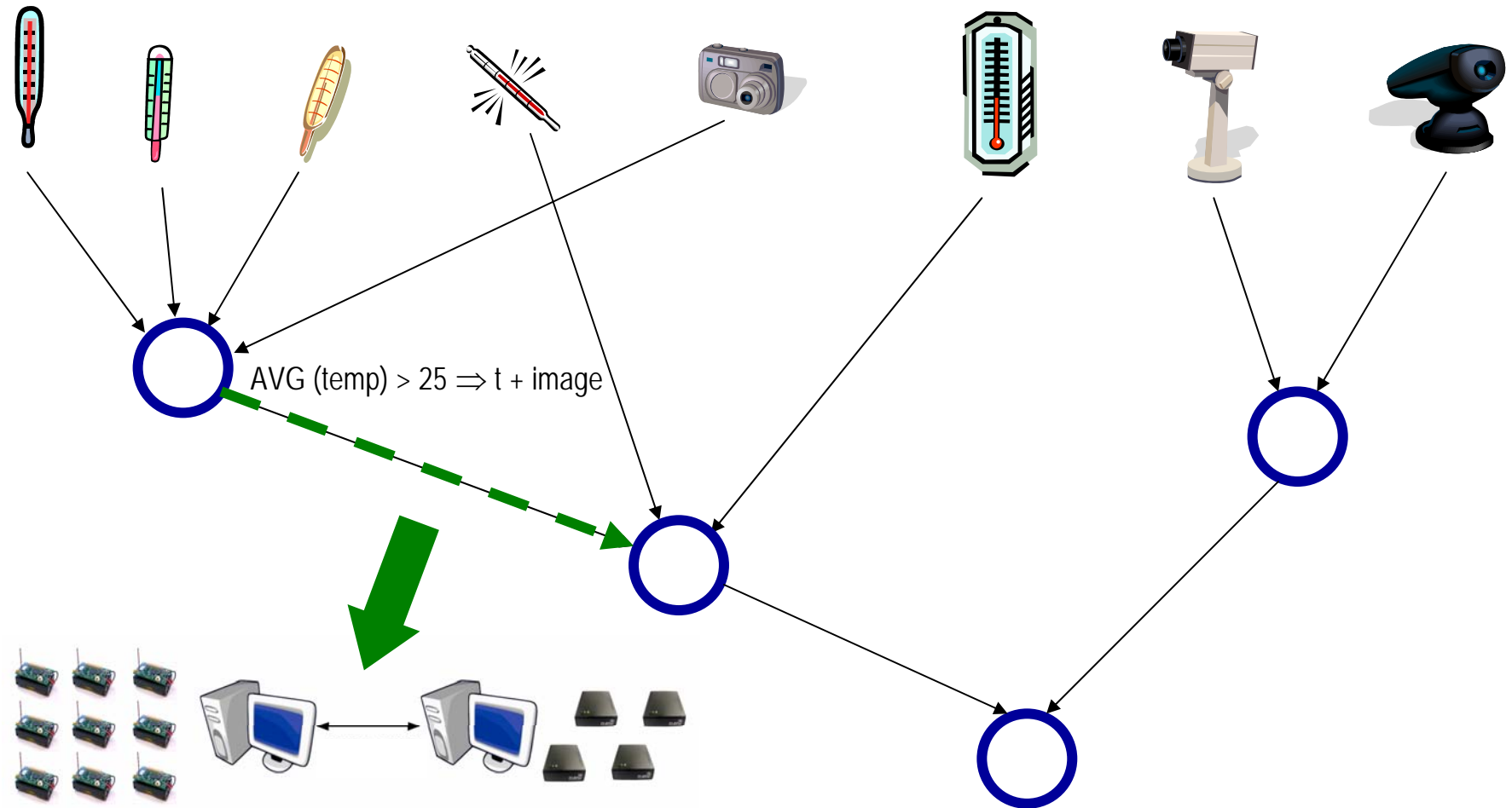
www.deri.ie

© Copyright 2006 Digital Enterprise Research
Institute. All rights reserved.



National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

The "Sensor Internet" at work





Lack of standardization (main obstacle)

- High development costs (heterogeneity, novel technologies)
- Limited portability
- No standard APIs and services for fast and flexible development

Dropping prices for sensors technology (rate)

- ⇒ Increasing number of sensor networks and applications
- ⇒ Large-scale integration of sensor network data

Solutions: Middleware + DSMS → GSN



- **Middleware**
 - Software abstraction that connects software components or applications
 - Provides services (networked services, e.g. Web services)
 - Provides interoperability (common interfaces and APIs)
 - Types: Message, Object, SQL, RPC oriented platforms
- **Data Stream Management System**
 - Data centric systems (efficient processing)
 - Provides data parallelism (data flow)
 - Query processing on data (run continuous queries on data)
 - Event-driven systems
 - Systems: Aurora, Borealis, STREAM, TelegraphCQ, GSN, etc.

Background – Global Sensor Networks (GSN)



- Project started by Ali Salehi in 2005 at the LSIR Laboratory, EPFL (Karl Aberer and Manfred Hauswirth)
- The initial goal was to provide a reusable software platform for processing data from sensor networks.
- The platform was extended to do generic stream data processing
- Registered as Open Source project in January 2006.
- Codebase: ~81kLOC, Visited ~85,000, Downloads: 1600
- Developer base: ~33 users (from mailing list)
- First GSN Workshop in Enschede, Netherlands, 26-27th May, 2008.
- Project Website: <http://gsn.sourceforge.net/>

GSN in a Nutshell

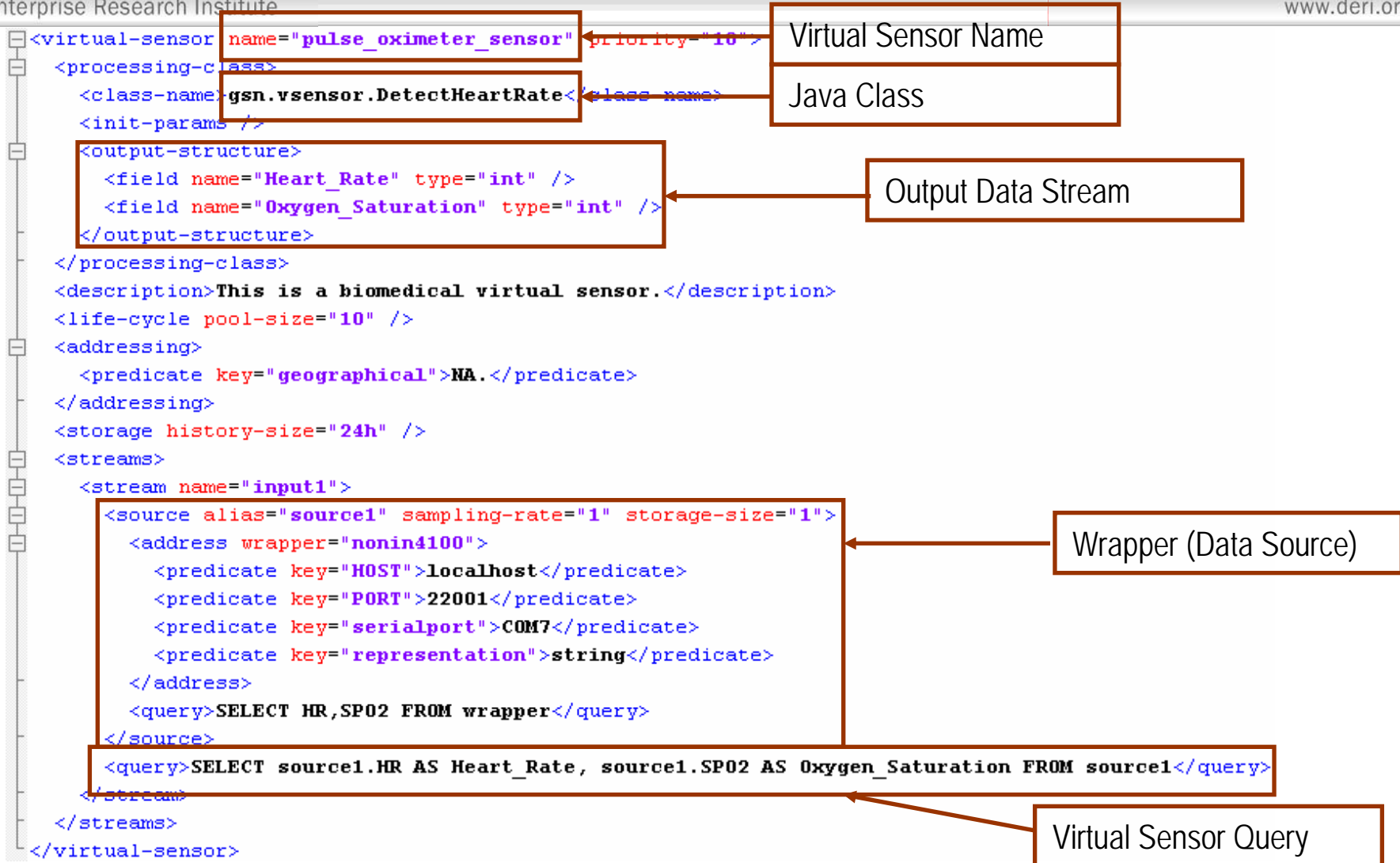


- A middleware platform to integrate data from heterogeneous sensor networks
- Declarative specification of sensor networks and data streams (XML)
- Uses a set of software abstractions to deal with data
- Low effort to add new types of sensor hardware
- SQL-based query processing
- Dynamic (re-)configurable system
- Light-weight implementation (Java)
- Open Source Software



- **Wrappers (abstract from hardware)**
 - Device driver for your sensor hardware
 - Translate sensor data format to GSN data model.
- **Virtual Sensors (abstract from sensor)**
 - A virtual sensor can be **any kind of data producer**
 - A real sensor, a wireless camera, a desktop computer, etc.
 - Virtual sensors are specified in **XML** and have a **Java class**
- **Abstract from implementation details**
 - physical sensors
 - a combination of other virtual sensors
 - 1 virtual sensor = n input data streams + processing + 1 output data stream

GSN Virtual Sensor – Biomedical Sensor (example)

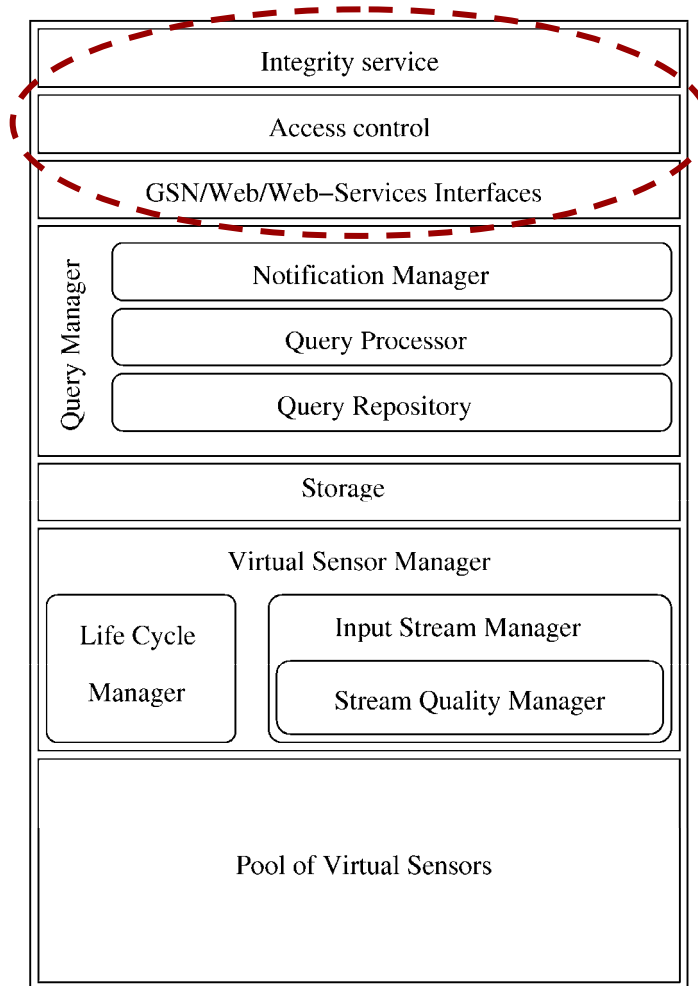


Accessing physical sensors: Wrappers



- **HTTP generic wrapper**
 - devices accessible via HTTP GET or POST requests, e.g., the AXIS206W wireless camera
- **Serial forwarder wrapper**
 - enables interaction with TinyOS compatible motes (standard access in TinyOS)
- **USB camera wrapper**
 - Local USB connection
 - supports cameras with OV518 and OV511 chips
- **TI-RFID wrapper**
 - access to Texas Instruments Series 6000 S6700 multi-protocol RFID readers
- **Bluetooth wrapper**
 - access to devices using the MAC and RFCOMM bluetooth
- **Generic UDP wrapper**
 - any device using the UDP protocol
- **Generic serial wrapper**
 - supports sensing devices which send data through the serial port

System Architecture – GSN node

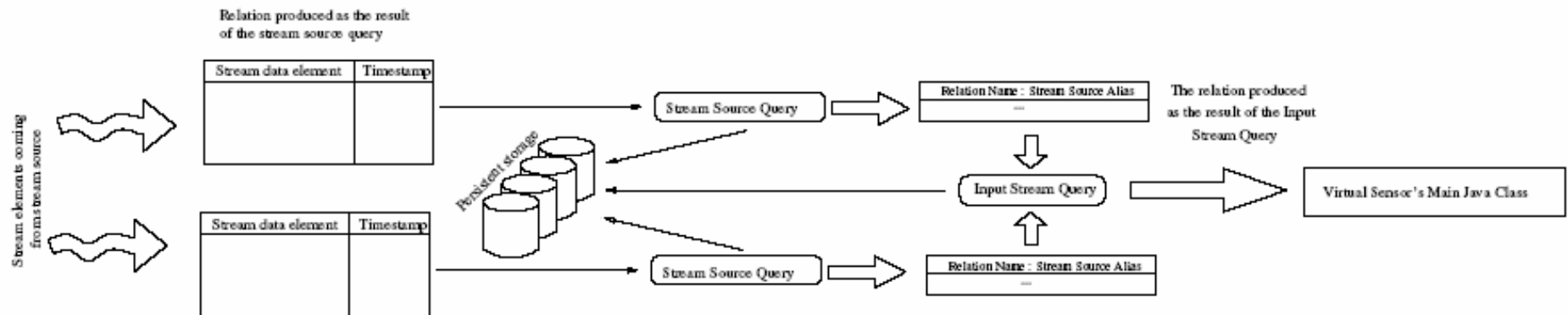


- **Each GSN node hosts a number of virtual sensors**
- **Virtual sensor manager**
 - provides access to the virtual sensors
 - manages the delivery of sensor data
- **Life-cycle manager**
 - provides and manages the resources provided to a virtual sensor
 - manages the interactions with a virtual sensor
 - ensures stream quality
 - manages the life-cycle of sensors
- **Storage layer**
 - persistent storage for data streams
- **Query manager**
 - manages active queries
 - query processing
 - delivery of events and query results to registered, local or remote consumers
- Top layers: **access, access control, and integrity**

GSN Internals (data and time model)



- In GSN a **data stream** is a set of **timestamped tuples**
- The basic unit of data in GSN is a **StreamElement**
- Temporal data processing
 - Implicit timestamp attribute (TIMEID)
 - Implicit timestamp of tuples upon arrival at the GSN wrapper



Event Handling in GSN



- Events are processed based on queries
- Using virtual sensors
- Virtual sensor can be aggregated to form complex event processing (queries)
- This can be used to implement notifications, e.g. email

Event Processing using a Virtual Sensor (example)



```
1 <virtual-sensor name="room-monitor" priority="11">
2   <addressing>
3     <predicate key="geographical">BC143</predicate>
4     <predicate key="usage">room monitoring</predicate>
5   </addressing>
6   <life-cycle pool-size="10" />
7   <output-structure>
8     <field name="image" type="binary:jpeg" />
9     <field name="temp" type="int" />
10  </output-structure>
11  <storage permanent="true" history-size="10h" />
12  <input-streams>
13    <input-stream name="cam">
14      <stream-source alias="cam" storage-size="1"
15        disconnect-buffer-size="10">
16        <address wrapper="remote">
17          <predicate key="geographical">BC143</predicate>
18          <predicate key="type">Camera</predicate>
19        </address>
20        <query>select * from WRAPPER</query>
21      </stream-source>
22      <stream-source alias="temperature1" storage-size="1m"
23        disconnect-buffer-size="10">
24        <address wrapper="remote">
25          <predicate key="type">temperature</predicate>
26          <predicate key="geographical">BC143-N</predicate>
27        </address>
28        <query>select AVG(temp1) as T1 from WRAPPER</query>
29      </stream-source>
30      <stream-source alias="temperature2" storage-size="1m"
31        disconnect-buffer-size="10">
32        <address wrapper="remote">
33          <predicate key="type">temperature</predicate>
34          <predicate key="geographical">BC143-S</predicate>
35        </address>
36        <query>select AVG(temp2) as T2 from WRAPPER</query>
37      </stream-source>
38      <query>
39        select cam.picture as image, temperature.T1 as temp
40        from cam, temperature1
41        where temperature1.T1 > 30 AND
42              temperature1.T1 = temperature2.T2
43      </query>
44    </input-stream>
45  </input-streams>
46 </virtual-sensor>
```

This virtual sensor will only produce data when the query condition is met

```
<query>
  select cam.picture as image, temperature.T1 as temp
  from cam, temperature1
  where temperature1.T1 > 30 AND
        temperature1.T1 = temperature2.T2
</query>
```



- In this tutorial you will learn how to:
 1. Use an example virtual sensor
 2. Write a Wrapper
 3. Write a Virtual Sensor (XML)
 4. Write a Virtual Sensor Java Processing Class
 5. Test your wrapper and virtual sensor
 6. Write a simple application that uses GSN

Writing a Wrapper (example)



```
package gsn.wrappers.general;

import java.io.Serializable;
import gsn.beans.AddressBean;
import gsn.beans.DataField;
import gsn.wrappers.AbstractWrapper;
import org.apache.log4j.Logger;

public class MyWrapper extends AbstractWrapper
{
    private final transient Logger logger = Logger.getLogger(MyWrapper.class);
    private static DataField[] dataField = new DataField[] { new DataField("value", "int", "description") };
    private int value = 0;

    public boolean initialize()
    {
        AddressBean wrapper_params = getActiveAddressBean();

        return true;
    }

    public void run()
    {
        while (isActive())
        {
            // do something
        }
    }

    public DataField[] getOutputFormat()
    {
        return dataField;
    }

    public void finalize()
    {
        threadCounter--;
    }
}
```

Writing a Virtual Sensor Processing Class (example)



```
package gsn.vsensor;

import gsn.beans.StreamElement;
import gsn.beans.VSensorConfig;
import java.util.TreeMap;
import org.apache.log4j.Logger;

public class MyVirtualSensor extends AbstractVirtualSensor
{
    private final static Logger logger = Logger.getLogger(MyVirtualSensor.class);

    public void dataAvailable(String inputStreamName, StreamElement data_stream)
    {
        // process the data stream

        // post the stream data produced
        dataProduced(data_stream);
    }

    public boolean initialize()
    {
        VSensorConfig vsensor = getVirtualSensorConfiguration();
        TreeMap<String, String> params = vsensor.getMainClassInitialParams();

        return true;
    }

    public void finalize()
    {
    }
}
```

Writing an Application using GSN



- GSN provides:
 - Web Service Interface
 - XML-RPC
 - One-Shot query over HTTP (using REQUEST 114)

<http://localhost:22001/gsn?REQUEST=114&name=sensorname>

Further Reading



- About GSN
 - The Book of GSN (see /docs folder in GSN repository)
http://gsn.svn.sourceforge.net/viewvc/*checkout*/gsn/trunk/doc/bookofgsn/thebookofgsn.pdf
- References
 - [The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks](http://www.manfredhauswirth.org/research/papers/LSIR-REPORT-2006-006.pdf)
<http://www.manfredhauswirth.org/research/papers/LSIR-REPORT-2006-006.pdf>
 - [Infrastructure for data processing in large-scale interconnected sensor network](http://www.manfredhauswirth.org/research/papers/GSN-MDM2007.pdf)
<http://www.manfredhauswirth.org/research/papers/GSN-MDM2007.pdf>
- Download GSN:
<http://gsn.sourceforge.net/>