Zero-programming Sensor Network Deployment[*]

Karl Aberer[†], Manfred Hauswirth[‡], Ali Salehi[†]
[†]Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[‡]Digital Enterprise Research Institute (DERI), National University of Ireland, Galway

Abstract

The availability of cheap and smart wireless sensing devices provides unprecedented possibilities to monitor the physical world. On the technical side these devices introduce several original research problems, many of them related to the integration of the rampant technology proposals. Global Sensor Networks (GSN) is a platform which provides a scalable infrastructure for integrating heterogeneous sensor network technologies using a small set of powerful abstractions. GSN supports the integration and discovery of sensor networks and sensor data, provides distributed querying, filtering, and combination of sensor data, and offers dynamic adaption of the system configuration during operation through a declarative XML-based language, and enables zero-programming deployment and management.

1. Towards a Sensor Internet

In 1999, *Business Week* named networked micro-sensor technology as one of the 21 most important technologies of the 21st century. To date, the research in the sensor network community has mainly focused on routing protocols and information collection and aggregation in a single sensor network with multiple different sensors connected through wireless links. As the prices of sensors decrease rapidly, there will soon exist huge numbers of sensor networks in different places, managed by different organizations. To fully exploit the potential of this "Sensor Internet," platforms enabling the fast and easy deployment, integration and management of the sensor sites and the produced data streams will be required. At the moment these issues are mostly addressed in an ad-hoc fashion.

Given the current growth rate, we may soon expect to arrive at a situation comparable to the publication of documents on the Web whose success is mainly based on sharing a few simple logical abstractions (URL, hyperlink, HTML) and basic communication protocols (HTTP, more recently Web Services) that provide universal access and linking among autonomously published data sources. The Global Sensor Networks (GSN) platform takes up these successful ideas and aims at making publication and access to sensor networks and sensor data as simple, powerful, and flexible as accessing Web documents. The design of GSN follows four basic goals:

Simplicity: The system is based on a minimal set of powerful abstractions which can be easily configured and adopted to the user's needs. Sensor networks and data streams are defined in a declarative way using SQL as data manipulation language. XML is used as the syntactic framework for system configuration.

Adaptivity: Adding new types of sensor networks and dynamic (re-)configuration of data sources is supported during run-time without having to interrupt ongoing system operation. To that end we use a container-based implementation facilitating dynamic reconfiguration.

Scalability: By targeting a very large number of data producers and consumers with a variety of application requirements, GSN has to consider scalability issues specifically for distributed query processing and distributed discovery of sensor networks. To meet this requirement, the design of GSN is based on a peer-to-peer architecture.

Light-weight implementation: GSN can be deployed in standard computing environments without excessive hardware or bandwidth requirements. It is portable, requires minimal configuration, and is easy to install and use.

In this paper we provide a brief overview of GSN. We sketch a simple usage scenarios in Section 2, provide an overview of the underlying conceptual model in Section 3, discuss basic abstractions in Section 4, introduce GSN's architecture in Section 5, and present related work in Section 6 before concluding. A detailed description of GSN is provided in [1] and the implementation is available from http://globalsn.sourceforge.net/.

2. A Simple Application Example

To illustrate the potential applications and flexibility of GSN we provide a simple application scenario in a (university) building as shown in Figure 1. We assume the following, fairly typical hardware setup: wireless cameras with built-in HTTP access; wireless sensors (motes) equipped

[*]The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 and by the Líon project supported by Science Foundation Ireland.

with light, sound, temperature, pressure, etc. sensors (we assume that all motes of the same type form a sensor network); RFID tags which are attached to the key rings of people, and to books, mobile phones and laptops in the buildings; and several RFID readers whose coverage ranges are shown in yellow (gray) in Figure 1.

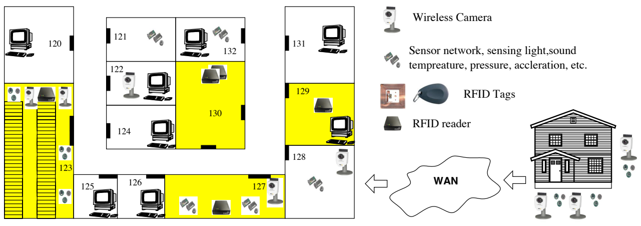

Figure 1. A simple scenario

Further we assume that: each computer runs a GSN instance; the webcams are accessed directly via HTTP for which some GSN instance holds the corresponding virtual sensor definitions; and the other sensors (motes) or a complete sensor network can either be accessed via the local area network or they are physically connected to one of the computers close to them.

In this setup GSN allows the user to accomplish a large variety of tasks. For example, the library manager can register a query to be notified when there are more than 15 books (equipped with RFIDs) in one room in addition to the monthly report on the most popular books of the month (e.g., to buy more of them). Individual users can post one-shot queries to the library (room 123) to get the status of certain books or, in case a book of interest is currently not available, can register a continuous query to be notified when the book is returned to the library. Or, a person in room 128 may be interested to receive a stream of camera images whenever a movement in the house is detected or a sound sensor observes some noise above a certain threshold. If someone loses a mobile phone (with an RFID tag attached to its battery slot), one can check for its last location in the building simply by posting a query on the previous observations provided by all sensor networks deployed in the building or register a continuous query to be notified (e.g., via email) whenever the mobile phone's RFID tag is detected.

3. Global Sensor Network Model

The current view on wireless sensor networks is based on a *physical view*: Single sensors or whole sensor networks are connected via wireless connections to an access point. Ad-hoc routing protocols route data to selected sensors (sinks) that communicate with an access point which is connected to the Internet and which can process and make available the data received from the sensors. Applications wanting to use sensor data need to first identify the access points, possibly through multiple layers, and then follow the specific access protocols defined.

A major problem of this model is the dependency of the access to sensor data on the specific implementation. Since sensors can be considered as data sources GSN adopts the principle of *data independence* as a fundamental idea. This abstraction is well-accepted and has proven extremely successful in the design of database systems. It allows the users of GSN to abstract from the highly heterogeneous physical infrastructures of current sensor network platforms, just as in a database system where the user abstracts from the specific implementation of physical data storage and access.

GSN provides a *logical view* on sensor networks based on the *virtual sensor* abstraction. Virtual sensors abstract from the implementation details to access sensor data and model sensor data as temporal streams of relational data. Virtual sensors can also represent derived views on sensor data streams, resulting from post-processing and combination of sensor data from different sources. This is shown informally by the conceptual data flow in Figure 2 which illustrates how data streams from different sensors and sensor networks can be integrated. Virtual sensors are bound to physical nodes in the network by deploying them to *GSN containers* that host virtual sensors. This approach supports our goal of simplicity, as by one single abstraction users can work with sensor data from heterogeneous sensor data sources and any data derived from raw sensor data.

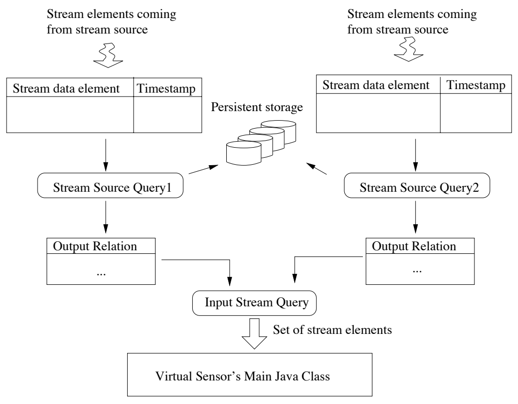

Figure 2. Conceptual data flow in a GSN node

4. Virtual Sensors and Data Stream Processing

A virtual sensor can have any number of input streams and produces one output stream. The specification of a virtual sensor provides all necessary information required for deploying and using it, including (1) metadata used for identification and discovery, (2) the structure of the data streams which the virtual sensor consumes and produces (3) an SQL-based specification of the stream processing performed in a virtual sensor, and (4) functional properties related to persistency, error handling, life-cycle, management, and physical deployment

To support rapid deployment, these properties of virtual sensors are provided in a declarative deployment descriptor. Figure 3 shows a fragment of a virtual sensor definition which defines a sensor returning an averaged temperature from a remote virtual sensor (`wrapper="remote"`) which is accessed via the Internet through another GSN instance (GSN instances cooperate in a peer-to-peer fashion).

```
 ...
<life-cycle pool-size="10" />
<output-structure>
  <field name="TEMPERATURE" type="integer"/>
</output-structure>
<storage  permanent-storage="true" size="10s" />
<input-stream name="dummy" rate="100" >
  <stream-source alias="src1" sampling-rate="1"
                              storage-size="1h">
     <address wrapper="remote">
       <predicate key="type" val="temperature" />
       <predicate key="location" val="bc143" />
     </address>
     <query>select avg(temperature)
                   from WRAPPER</query>
   </stream-source>
  <query>select * from src1</query>
</input-stream>
 ...
```

Figure 3. Virtual sensor definition (fragment)

To specify the processing of the input streams we use SQL queries which refer to the input streams by the reserved keyword `WRAPPER`. The `<input-stream>` element provides all definitions required for identifying and processing an input stream of the virtual sensor. The `<life-cycle>` element defines deployment aspects such as the number of threads available for processing, the `<storage>` element controls how stream data is stored persistently (among other attributes this controls the temporal processing), and `<output-structure>` defines the structure of the produced output stream.

In GSN a data stream is a sequence of timestamped tuples. The order of the data stream is derived from the ordering of the timestamps and the GSN container provides basic support to manage and manipulate the timestamps. These services essentially consist of the following components: (1) a local clock at each GSN container, (2) implicit management of a timestamp attribute, (3) implicit timestamping of tuples upon arrival at the GSN container (reception time), and (4) a windowing mechanism which allows the user to define count- or time-based windows on data streams. Multiple time attributes can be associated with data streams and can be manipulated through SQL queries. The production of a new output stream element of a virtual sensor is always triggered by the arrival of a data stream element from one of its input streams. GSN's query processing approach is related to TelegraphCQ as it separates the time-related constructs from the actual query. Temporal specifications, e.g., the window size, are provided in XML in the virtual sensor specification, while data processing is specified in SQL with the full range of operations allowed by the standard syntax.

Detailed descriptions of virtual sensors and GSN's data stream processing are provided in [1].

5. System Architecture and Implementation

Similar to application servers, GSN provides an environment in which sensor network services can be specified and deployed in a simple and flexible manner by hiding most of the system complexity. GSN follows a container-based architecture and each container can host and manage any number of virtual sensors. The container manages every aspect of the virtual sensors at runtime including remote access, interaction with the sensor network, security, persistence, data filtering, concurrency, and access to and pooling of resources which enables on-demand use and combination. Virtual sensor descriptions hold user-definable key-value pairs which are published in a peer-to-peer directory so that virtual sensors can be discovered and accessed based on any combination of their properties which provides a simple model for identification and discovery of virtual sensors through metadata. GSN nodes communicate among each other in a peer-to-peer fashion. By viewing GSN containers as cooperating peers in a decentralized system, we tried avoid some of the intrinsic scalability problems of many other systems which rely on a centralized or hierarchical architecture. Targeting a "Sensor Internet" as the long-term goal, we also take into account that such a system will consist of "Autonomous Sensor Systems" with a large degree of freedom and only limited possibilities of control, similarly as in the existing Internet. Figure 4 depicts the internal architecture of a GSN node.

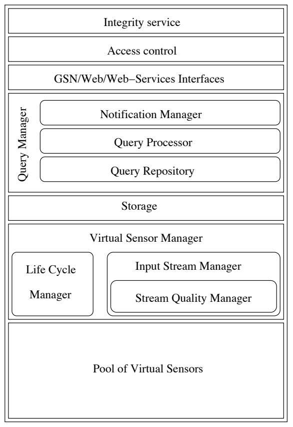

Figure 4. GSN container architecture

The virtual sensor manager (VSM) is responsible for providing access to the virtual sensors, managing the delivery of sensor data, and providing the necessary administrative infrastructure. Its life-cycle manager (LCM) subcomponent provides and manages the resources provided to a virtual sensor and manages the interactions with a virtual sensor (sensor readings, etc.) while the input stream manager (ISM) manages the input streams and ensures stream quality (disconnections, unexpected delays, missing values, etc.). The data from/to the VSM passes through the storage layer which is in charge of providing and managing persistent storage for data streams. Query processing is done by the query manager (QM) which includes the query processor being in charge of SQL parsing, query planning, and execution of queries (using an adaptive query execution plan).

The query repository manages all registered queries (subscriptions) and defines and maintains the set of currently active queries for the query processor. The notification manager deals with the delivery of events and query results to the registered clients. The notification manager has an extensible architecture which allows the user to customize it to any required notification channel (email, SMS, triggering actuators such as an epuck robot, etc.). The top three layers deal with access mechanisms (Web browser interface, web services, etc.), (fine-grained) access control, and integrity and security (electronic signatures, encryption, etc.).

GSN is implemented in Java (approx. 30,000 LOC at the moment). For each type of sensor a wrapper is required. Currently wrappers for all major TinyOS platforms (Mica2, Mica2Dot, etc.) are implemented as well as for wired and wireless (HTTP-based) cameras (e.g., AXIS 206W), several RFID readers (TI, Alien Technology), Bluetooth devices, Shockfish, WiseNodes, epuck robots, etc. For deploying a virtual sensor the user has to specify an XML deployment descriptor, i.e., a virtual sensor which can be added to / removed from a running GSN system at any time without interrupting the system's operation. Additionally, GSN supports plug-and-play detection and deployment of sensors based on the IEEE 1451 standard as GSN uses the standardized Transducer Electronic Data Sheet (TEDS) self-description feature for dynamic generation of virtual sensor descriptions. By using re-usable wrappers abstracting from the physical access details along with declarative virtual sensor descriptions and the IEEE 1451 based plug-and-play feature, GSN provides truly zero-programming deployment and management support (including sensor mobility).

6. Related Work

So far only few architectures to support interconnected sensor networks exist. Probably the closest approach to GSN is the work by Sgroi et.al. [4] who suggest basic abstractions, a standard set of services, and an API to free application developers from the details of the underlying sensor networks. However, the focus is on systematic definition and classification of abstractions and services, while GSN takes a more general view and provides not only APIs but a complete query processing and management infrastructure with a declarative language interface. Hourglass [5] provides an infrastructure for connecting sensor networks to applications and offers topic-based discovery and data-processing services. Similar to GSN it tries to hide internals of sensors from the user but focuses on maintaining quality of service of data streams in the presence of disconnections while GSN is more targeted at flexible configurations, general abstractions and distributed query support. HiFi [2] provides efficient, hierarchical data stream query processing to acquire, filter, and aggregate data from multiple devices in a static environment (with homogeneous sensor networks) while GSN takes a peer-to-peer perspective assuming a dynamic environment with heterogeneous sensor networks and allowing any node to be a data source, data sink, or data aggregator. IrisNet [3] proposes a two-tier architecture consisting of sensing agents (SA) which collect and pre-process sensor data and organizing agents (OA) which store sensor data in a hierarchical, distributed XML database. This database is modeled after the design of the Internet DNS and supports XPath queries. In contrast to that, GSN follows a symmetric peer-to-peer approach and supports publish/subscribe besides active queries. Besides these architectures, a large number of complimentary work on query processing in sensor networks exist, Aurora, STREAM, TelegraphCQ, and Cougar being the most prominent ones.

7. Conclusions

To enable the vision of a "Sensor Internet" we suggest to adopt the concepts of successful, global information technologies such as the Web, whose success is based on sharing a few simple logical abstractions and basic communication protocols that provide universal access and linking among autonomously published data sources. GSN follows this model by making publication of sensor data and access to sensor networks and sensor data as simple, powerful, and flexible as accessing Web documents. GSN hides arbitrary data sources behind its virtual sensor abstraction and provides simple and uniform access to the host of heterogeneous technologies available through powerful declarative specification and query tools. A detailed description of GSN is provided in [1] and the implementation is available from http://globalsn.sourceforge.net/.

References

[1] K. Aberer, M. Hauswirth, and A. Salehi. The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks. Technical Report LSIR-REPORT-2006-006, EPFL, 2006.

[2] M. Franklin, S. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. Wu, O. Cooper, A. Edakkunni, and W. Hong. Design Considerations for High Fan-in Systems: The HiFi Approach. In *CIDR*, 2005.

[3] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. IrisNet: An Architecture for a World-Wide Sensor Web. *IEEE Pervasive Computing*, 2(4), 2003.

[4] M. Sgroi, A. Wolisz, A. Sangiovanni-Vincentelli, and J. M. Rabaey. A service-based universal application interface for ad hoc wireless sensor and actuator networks. In *Ambient Intelligence*. Springer Verlag, 2005.

[5] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, and M. Welsh. Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. Technical Report TR-21-04, Harvard University, Electrical Engineering and Computer Science, 2004.