Middleware support for the "Internet of Things"*

Karl Aberer, Manfred Hauswirth, Ali Salehi

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

Abstract. The "Internet of Things" intends to enhance items of our everyday life with information and/or processing and interconnect them, so that computers can sense, integrate, present, and react on all kinds of aspects of the physical world. As this implies that enormous numbers of data sources need to be connected and related to each other, flexible and dynamic middleware support essentially providing zero-programming deployment is a key requirement to cope with the sheer scale of this task and the heterogeneity of available technologies. In this paper we briefly overview our Global Sensor Networks (GSN) middleware which supports these tasks and offers a flexible, zero-programming deployment and integration infrastructure. GSN's central concept is the virtual sensor abstraction which enables the user to declaratively specify XML-based deployment descriptors in combination with the possibility to integrate sensor network data through plain SQL queries over local and remote sensor data sources. The GSN implementation is available from http://globalsn.sourceforge.net/.

1 Introduction

In the sensor network domain most of the ongoing work focuses on energy efficient routing, aggregation, and data management algorithms inside a single sensor network. The deployment, application development, and standardization aspects received little attention so far. However, as the price of sensors diminishes rapidly we can soon expect to see very large numbers of heterogeneous sensor devices and sensor networks being deployed to support the vision of the "Internet of Things" [1]. Major challenges in this "Sensor Internet" environment are (1) minimizing the development and deployment efforts which are key cost factor in large-scale systems and (2) the data-oriented integration of very large numbers of data sources. Despite the heterogeneity of the available platforms, their requirements for processing, storing, querying and publishing data, however, are similar and the main differences among various software platforms are the different abstractions for the sensors and sensor networks.

Our Global Sensor Networks (GSN) middleware addresses these problems and provides a uniform platform for fast and flexible integration and deployment of heterogeneous sensor networks. The design of GSN follows four main design goals: Simplicity (a minimal set of powerful abstractions which can be easily configured and adopted), adaptivity (adding new types of sensor networks and dynamic (re-) configuration of

* 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 was (partly) carried out in the framework of the EPFL Center for Global Computing.

data sources has to be supported during run-time), scalability (peer-to-peer architecture), and light-weight implementation (small memory foot-print, low hardware and bandwidth requirements, web-based management tools).

In the following, we will give a concise overview of GSN and how it provides an infrastructure to support the "Internet of Things". Section 2 discusses the virtual sensor abstraction, Section 3 highlights GSN's data stream processing functionalities, and Section 4 describes its architecture. For a detailed description of these issues we refer the reader to [2]. Section 5 then discusses GSN's dynamic plug-and-play functionalities which rely on simple semantic descriptions based on the IEEE 1451 standard [3] to identify sensors and dynamically integrate them into GSN. This is a key enabling technology in GSN supporting fully dynamic and zero-programming integration of data sources in GSN.

2 Virtual sensors

The key abstraction in GSN is the *virtual sensor*. Virtual sensors abstract from implementation details of access to sensor data and they are the services provided and managed by GSN. A virtual sensor corresponds either to a data stream received directly from sensors or to a data stream derived from other virtual sensors. 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:

- metadata used for identification and discovery
- the structure of the data streams which the virtual sensor consumes and produces
- an SQL-based specification of the stream processing performed in a virtual sensor
- 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 1 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>
 ...
```

Fig. 1. 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. A detailed description of virtual sensors is provided in [2].

3 Data stream processing

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. Informally, the processing steps then are as follows:

1. The new data stream element is timestamped using the local clock.
2. Based on the timestamps for each input stream the stream elements are selected according to the definition of the time window and the resulting sets of relations are unnested into flat relations.
3. The input stream queries are evaluated and stored into temporary relations.
4. The output query for producing the output stream element is executed based on the temporary relations.
5. The result is permanently stored if required and all consumers of the virtual sensor are notified of the new stream element.

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.

4 GSN architecture

GSN follows a container-based architecture and each container can host and manage one or more virtual sensors concurrently. 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, i.e., this provides a simple model of identification and discovery of virtual sensors through metadata. GSN nodes communicate among each other in a peer-to-peer fashion. Figure 2 depicts the internal architecture of a GSN node.

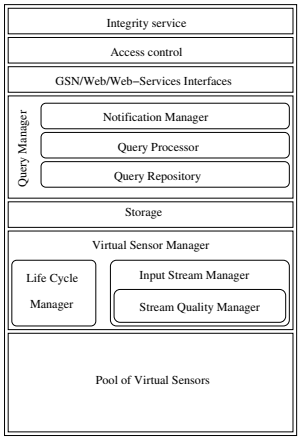


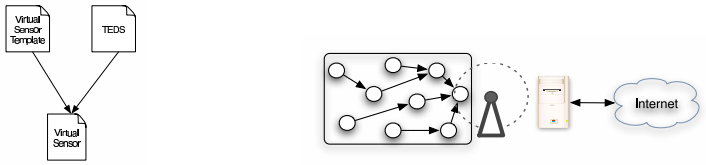
Fig. 2. GSN container architecture

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. The top three layers deal with access mechanisms, access control, and integrity and security.

5 Zero-programming deployment

As described before, GSN reduces deployment to writing an XML file describing the sensor-dependent properties. While having a single XML file (30-50 lines) describing a virtual sensor is much simpler than low-level device-dependent programming, human intervention is still required. To overcome this step, GSN uses the IEEE 1451 standard [3] for automatic detection, configuration and calibration. An IEEE 1451-compliant sensor provides a Transducer Electronic Data Sheet (TEDS) which is stored inside the sensor. The TEDS provides a simple semantic description of the sensor, i.e., it describes the sensor's properties and measurement characteristics such as type of measurement, scaling, and calibration information. A large number of sensors is already compliant to IEEE 1451 and this number is growing steadily.

To support truly zero-programming, GSN uses the TEDS self-description feature for the dynamic generation of virtual sensor descriptions by using a virtual sensor description template and deriving the sensor-dependent fields of the template from the data extracted from the TEDS as shown in Figure 3(a).



(a) Automatic generation (b) Setup

Fig. 3. Zero-programming deployment

The node setup to support plug-and-play deployment is shown in Figure 3(b). At least one base station capable of interacting with sensing devices (e.g., mica2, mica2dot, BTnode) is needed. When a sensor node enters a detection area (depicted by an antenna in the figure), GSN detects it, requests its TEDS, and dynamically instantiates a new virtual sensor based on the TEDS. This means that by using TEDS GSN can detect, identify, and automatically deploy sensors without human intervention. The generated virtual sensor description is immediately included into the detecting GSN node's repository and all processing dependent on the new sensor is executed. This is done on-the-fly while GSN is running. At the moment TEDS provides only that information on a sensor which enables interaction with it. Thus for some parts of the generated virtual sensor description, e.g., security requirements, storage and resource management, etc., we use default values.

The deployment of a sensor (network) in GSN implies making the produced data accessible through the Internet via web services and a web interfaces. Thus also all remote processing dependencies at any other node in the network of GSN nodes are triggered. This enables truly global integration of mobile sensors. GSN periodically asks for TEDS from all sensors to ensure that they are alive. If no response is provided by a sensor GSN removes the previously created virtual sensor, frees the associated resources and notifies dependent query and nodes.

6 Conclusions

GSN provides a flexible middleware for deployment of sensor networks meeting the challenges that arise in real-world environments. Its plug-and-play deployment functionality makes it specifically interesting as an infrastructure for the "Internet of Things," enabling completely new types of applications. For example, users or things could be equipped with RFID tags not only holding static data but also specifying data-processing tasks (queries), which GSN can recognize and include into its stream processing. This provides data processing dynamicity besides physical mobility without additional efforts for deployment. The GSN implementation is available from http://globalsn.sourceforge.net/ and a detailed description of GSN in provided in [2].

References

1. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet of Things. Scientific American (2004)
2. Aberer, K., Hauswirth, M., Salehi, A.: The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks. Technical Report LSIR-REPORT-2006-006, Ecole Polytechnique Fédérale de Lausanne (2006)
3. NIST: IEEE1451 (2006) http://ieee1451.nist.gov/.