A Middleware for Fast and Flexible Sensor Network Deployment*

Karl Aberer, Manfred Hauswirth, Ali Salehi
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
[karl.aberer, manfred.hauswirth, ali.salehi]@epfl.ch

ABSTRACT

A key problem in current sensor network technology is the heterogeneity of the available software and hardware platforms which makes deployment and application development a tedious and time consuming task. To minimize the unnecessary and repetitive implementation of identical functionalities for different platforms, we present our Global Sensor Networks (GSN) middleware which supports the flexible integration and discovery of sensor networks and sensor data, enables fast deployment and addition of new platforms, provides distributed querying, filtering, and combination of sensor data, and supports the dynamic adaption of the system configuration during operation. 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. In this demonstration, we specifically focus on the deployment aspects and allow users to dynamically reconfigure the running system, to add new sensor networks on the fly, and to monitor the effects of the changes via a graphical interface. The GSN implementation is available from http://globalsn.sourceforge.net/.

1. INTRODUCTION

The growing diversity of available software and hardware platforms in the sensor network domain creates problems when these sensor networks are being deployed and applications being developed on top of them. At the moment developers and administrators have to come up with specialized software and deployment strategies for each of the platforms. This is a time consuming and tedious task which in other, admittedly more "stable" domains has been addressed by abstracting from the physical view of a system to a logical model and implementing this logical model in terms of a middleware. The general purpose of middleware systems is to provide powerful abstractions codifying the essential requirements and concepts of a domain and providing flexible means for extending it to the concrete physical environment. This speeds up deployment and additionally pushes standardized APIs which simplifies application development and applications become "portable" over all physical systems included in the middleware.

In the sensor network context, the current research mainly focuses on routing, aggregation, and energy efficient data management algorithms inside one sensor network. The deployment, application development, and standardization aspects are usually not addressed. However, as the price of wireless sensors diminishes rapidly we can expect to see large numbers of heterogeneous sensor networks being deployed in different locations around the globe and being managed by different organizations. A major challenge in such a "Sensor Internet" environment is minimizing the deployment efforts which is a key cost factor in large systems. The requirements of the software infrastructure for processing, storing, querying and publishing data produced from a sensor network, however, are similar and the main difference between various software platforms is due to different abstractions for the available sensors. Additionally, having a generic middleware for sensor networks not only cuts costs, but also opens the possibility of sharing and integrating data among heterogeneous 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 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).

For a detailed description and analysis of all aspects of GSN we refer the reader to [1].

2. VIRTUAL SENSORS

As noted above, a small set of powerful, easily combinable abstractions are key to successful middleware design. 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
- a declarative SQL-based specification of the data stream processing performed in a virtual sensor
- functional properties related to persistency, error handling, life-cycle management, and physical deployment

*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.

VLDB '06, September 12-15, 2006, Seoul, Korea.

To support rapid deployment, these properties of virtual sensors are provided in a declarative deployment descriptor. Figure 1 shows a fragment of such a virtual sensor definition which defines a sensor returning an averaged temperature. The <life-cycle> element enables the control of network deployment aspects such as the number of threads available for processing, the <storage> element controls how stream data is persistently stored (among other attributes this controls the temporal processing), and the element <output-structure> defines the structure of the produced output stream. To specify the processing of the input streams we use SQL queries which refer to the input streams by the reserved keyword WRAPPER. The attribute wrapper="remote" indicates that the data stream is obtained from the Internet through GSN (thus logical addressing is possible).

```
 ...
<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" disconnect-buffer="10">
     <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 1: Virtual sensor definition (fragment)

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)
4. a windowing mechanism which allows the user to define count- or time-based windows on data streams.

In this way it is always possible to trace the temporal history of data stream elements throughout the processing history. Multiple time attributes can be associated with data streams and can be manipulated through SQL queries. In this way sensor networks can be used as observation tools for the physical world, in which network and processing delays are inherent properties of the observation process which cannot be made transparent by abstraction.

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. By default the new data stream element is timestamped using the local clock of the virtual sensor provided that the stream element had no timestamp.
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.

Additionally, GSN provides a number of possibilities to control the temporal processing of data streams, for example, bounding the rate of a data stream in order to avoid overloads of the system which might cause undesirable delays, sampling of data streams in order to reduce the data rate, bounding the lifetime of a data stream in order to reserve resources only when they are needed, etc.

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. At the moment GSN supports SQL queries with the full range of operations allowed by the standard syntax, i.e., joins, subqueries, ordering, grouping, unions, intersections, etc. The advantage of using SQL is that it is well-known and SQL query optimization and planning techniques can be directly applied.

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. This paradigm enables on-demand use and combination of sensor networks. Virtual sensor descriptions are identified by 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, for example, geographical location and sensor type. GSN nodes communicate among each other in a peer-to-peer fashion. Figure 2 depicts the internal architecture of a GSN node.

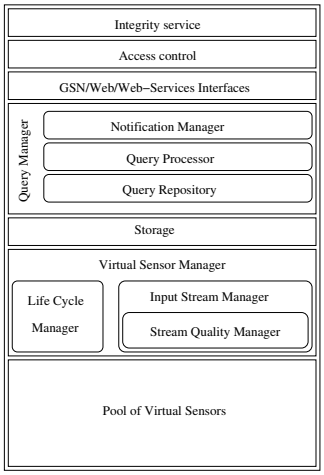

Figure 2: GSN container architecture

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 lifecycle 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