The Sensor Bus – Integrating Geosensors and the Sensor Web

Arne BRÖRING[1], Simon JIRKA[1], Theodor FOERSTER[2]

[1]52°North, Muenster, Germany
broering@52north.org, jirka@52north.org

[2] Institute for Geoinformatics – University of Muenster, Germany
theodor.foerster@uni-muenster.de

Abstract

In the past, a multitude of projects have demonstrated the applicability of OGC's Sensor Web Enablement (SWE) standards. SWE services have been used to encapsulate heterogeneous geosensors for web-based discovery, access, tasking, and alerting. Thereby, the integration of geosensors had to be established by manually adapting each SWE service implementation. This approach is cumbersome and leads to an extensive integration effort in large scale sensor network systems. To overcome these obstacles this work presents the Sensor Bus, an open source project facilitating the integration of new geosensors into the Sensor Web.

Motivation

Geosensors have become an important technology for monitoring environmental phenomena such as wind, rain, floods or earthquakes. Scenarios in which geosensor networks can be utilized include early warning systems, hazard management or precision agriculture [1]. To benefit from the enhanced data acquisition by geosensor networks and to maximize the information effectiveness in environmental decisions a coupling of available sensor data with other spatio-temporal resources (e.g. maps) is required. Integrating geosensor networks on the application level is ensured by the standards framework of the Sensor Web Enablement (SWE) [2] initiative of the OGC.

But there are obstacles in the current way of integrating sensors and sensor data into the Sensor Web. Generally, the Sensor Web as established by the SWE initiative which focuses on interacting with the upper application level. The interaction with the lower sensor network level is not yet sufficiently described. Hence, there is a gap between these two layers.

In various projects, the SWE framework[1] has been applied to enable the interoperable usage of geosensor networks. The industrial fire scenario, which has been part of the EC-funded project OSIRIS[2], exemplarily shows the existence of the gap between the two layers. The use case [3] includes three types of sensors, smoke detectors, cameras and thermometers, to detect fires in industrial facilities. On the Sensor Web level three services are involved. The Sensor Observation Service (SOS) is used to access data gathered by the different sensors. The tasking of sensors to modify internal parameters can be achieved via the Sensor Planning Service (SPS). To allow users a subscription for certain alerts and events (e.g. detection of smoke) the Sensor Alert Service (SAS) is applied.

The communication between sensors and their sensor network gateway is realized by a proprietary protocol based on ZigBee. The Sensor Web on the other hand is based on internet protocols. The SWE services encapsulate the sensor network and hide the lower level protocols. The connection

---

[1] http://www.ogcnetwork.net/swe
[2] http://www.osiris-fp6.eu/

between the two layers is established by manually adapting the services to the specific sensor types. Those proprietary bridges have to be built for each pair of web service implementation and sensor type. This approach is contrary to our aim of reaching interoperability and sustainable software development. Further on, it leads to an extensive amount of adaption effort which is the key cost factor in large-scale sensor network developments [4].

Coherent concepts are missing for an intermediary layer to connect the two distinct layers by guaranteeing a sufficient performance and simultaneously ensuring a high level of adaptivity for diverse sensor types. The open source project presented in this work tackles these obstacles by establishing such an intermediary layer, the *Sensor Bus*. It bridges the gap between sensor network layer and Sensor Web layer by realizing a set of interaction patterns between the two distinct layers [5].

The Sensor Bus Approach

Fig. 1 depicts the components of the Sensor Bus. A client located on the application layer invokes a SWE service for a specific functionality such as the retrieval of sensor observations or the submission of a sensor task. The Sensor Bus maintains associations to these services as well as associations to sensor gateways which provide access to connected sensors. The sensor gateway establishes the communication between its associated sensors and the upper layer. From a hardware perspective, sensor gateway and sensor may merge in certain scenarios to a single component (e.g., a weather station which comprises multiple sensors and is equipped with an advanced computing unit acting as the gateway to the associated sensors).

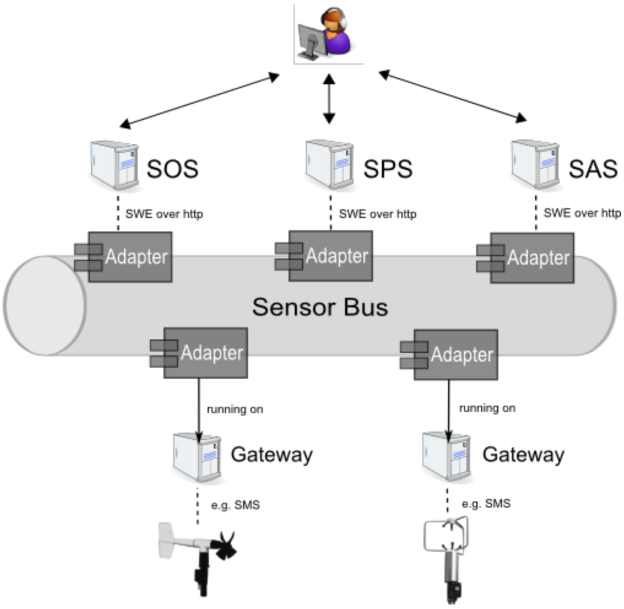

Figure 1: Overview of Sensor Bus concept

The Sensor Bus follows the Message Bus pattern [6] and incorporates a common communication infrastructure, a shared set of adapter interfaces, and a well-defined message protocol.

The communication is established through a publish/subscribe mechanism [7]. Services as well as sensors can publish messages to the Sensor Bus. Also, these components can subscribe to the Sensor Bus to receive messages using push-based communication. The Sensor Bus forwards the posted messages to all subscribed components. The different components (i.e., sensors and SWE services) can subscribe and publish through interfaces defined by the Sensor Bus. For these interfaces, pluggable adapters can be developed by sensor vendors or service providers. The adapters convert the service or sensor specific communication protocol to the internal protocol of the Sensor Bus. Other than physical buses, used for example in computer hardware, the Sensor Bus is a logical bus and reflects a bus topology to external components (sensors and services).

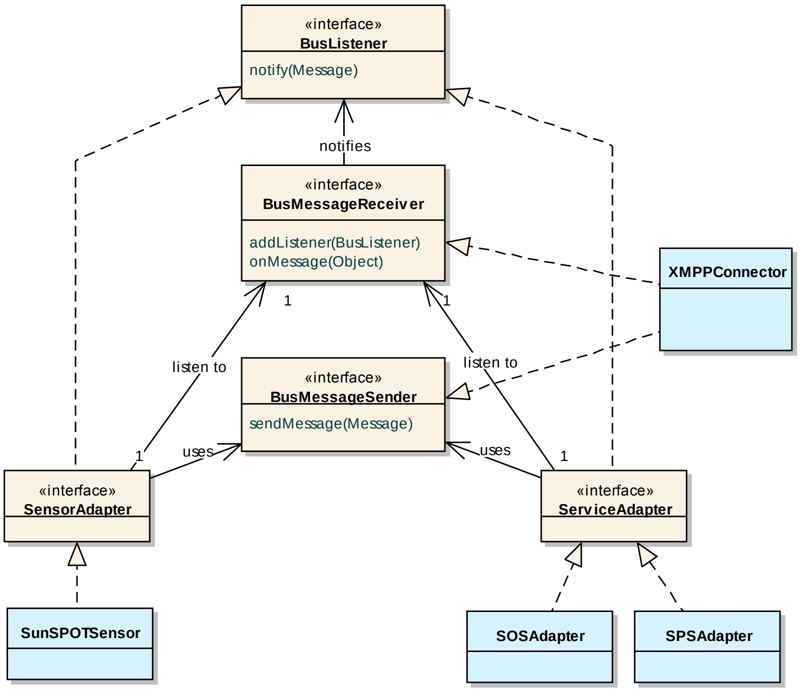

Figure 1: Sensor Bus architecture

Fig. 2 shows the architecture of the Sensor Bus as a UML class diagram. The *SensorAdapter* (e.g., an adapter for the SunSPOT[3] sensor platform) and the *ServiceAdapter* (e.g., adapters for SOS and SPS) are used to connect sensors and services to the Sensor Bus. Both interfaces are *BusListeners* so that they can be notified by the *BusMessageReceiver* for retrieving messages sent over the bus. *SensorAdapter* and *ServiceAdapter* transmit their messages to the bus through the *BusMessageSender*. *BusMessageReceiver* and *BusMessageSender* offer an interface that abstracts from the underlying communication infrastructure of the bus. The *XMPPConnector* in Fig. 2 is an example implementation of the two interfaces to realize the Sensor Bus based on an XMPP[4] server. Since the two interfaces abstract from the communication infrastructure, it is easy to exchange the implementing class and realize the Sensor Bus based on other messaging technologies (e.g., IRC or JMS). The implementation of *BusMessageReceiver* calls in case of an incoming message *onMessage()* to notify the listeners. The concrete sensor and service adapters, acting as listeners, analyze the incoming message and react on it according to their specifications.

Conclusions and Future Work

In this paper, we introduce the concept of an intermediary layer, the Sensor Bus, to close the gap between the sensor network layer and the Sensor Web layer. Due to the simplicity of the bus protocol, an integration of new sensors into the Sensor Web becomes very easy.

---

[3] http://www.sunspotworld.com

[4] http://www.xmpp.org

This decreases the entry threshold for sensor vendors to bring their devices into the interoperable Sensor Web.

Also, the approach of an underlying message bus allows an automatic notification of SWE services about changes in the sensor network. So far, new data or changed metadata has to be communicated separately to the various SWE services which may quickly lead to data inconsistency within the Sensor Web.

The presented architecture of the Sensor Bus will be brought into the standardization process of the SWE initiative at the OGC. Also, 52° North will use the Sensor Bus in the upcoming OGC Web Services testbed (OWS-7).

The Sensor Bus is published as an open source project[5] within the 52° North Sensor Web community[6]. The implementation shows the applicability of the approach, but also exposes working packages for the future. The technological mechanisms for true *sensor plug & play* will be developed based on the described concepts. Current work in progress is a SensorML[7] profile defining a generic driver interface to automatically create the communication logic for adapting sensors to the Sensor Bus. Soon, the approach will be applied to the existing water gage network of the regional German watershed monitoring agency Wupperverband to demonstrate its applicability in real-world scenarios.

References

[1] D. Shepherd and S. Kumar, *Distributed Sensor Networks*. Chapman & Hall, 2005, ch. Microsensor Applications.

[2] M. Botts, G. Percivall, C. Reed, and J. Davidson, "OGC (R) Sensor Web Enablement: Overview and High Level Architecture," *Lecture Notes In Computer Science*, vol. 4540, pp. 175–190, 2008.

[3] S. Jirka, A. Bröring, and C. Stasch, "Discovery Mechanisms for the Sensor Web," *Sensors*, vol. 9, 2009.

[4] K. Aberer, M. Hauswirth, and A. Salehi, "Middleware support for the Internet of Things," *5. GI/ITG KuVS Fachgespraech - Drahtlose Sensornetze*, pp. 15 – 19, 2006.

[5] A. Broering, T. Foerster, and S. Jirka, "Interaction Patterns for Bridging the Gap between Sensor Networks and the Sensor Web," in *WoT 2010: First International Workshop on the Web of Things*, Mannheim, Germany, March 29. - April 2. 2010; forthcoming.

[6] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing, 2003.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Resusable Object-Oriented Software*. Addison-Wesley Professional, 1995.

---

[5] http://www.52north.org/sensorBus

[6] http://www.52north.org/swe

[7] http://www.ogcnetwork.net/sensorml