Sensor Bus: An Intermediary Layer for Linking Geosensors and the Sensor Web

Arne Broering
International Institute for Geo-Information Science and Earth Observation (ITC)
University of Twente
Enschede, Netherlands
broering@52north.org

Theodor Foerster
Institute for Geoinformatics (IfGI)
University of Münster
Münster, Germany
theodor.foerster@uni-muenster.de

Simon Jirka
$52^{\circ}North$
Initiative for Geospatial Open Source Software
Münster, Germany
jirka@52north.org

Carsten Priess
Institute for Geoinformatics
University of Münster
Münster, Germany
carsten.priess@uni-muenster.de

ABSTRACT

In recent years, the standards of OGC's Sensor Web Enablement (SWE) initiative have been applied in a multitude of projects to encapsulate heterogeneous geosensors for web-based discovery, tasking and access. Currently, SWE services and the different types of geosensors are integrated manually due to a conceptual gap between these two layers. Pair-wise adapters are created to connect an implementation of a particular SWE service with a particular type of geosensor. This approach is contrary to the aim of reaching interoperability and leads to an extensive integration effort in large scale systems with various types of geosensors and various SWE service implementations.

To overcome this gap between geosensor networks and the Sensor Web, this work presents an intermediary layer for integrating these two distinct layers seamlessly. This intermediary layer is called the *Sensor Bus* as it is based on the message bus architecture pattern. It reduces the effort of connecting a sensor with the SWE services, since only the adaption to the Sensor Bus has to be created. The communication infrastructure which acts as the basis for the Sensor Bus is exchangeable. In this work, the Sensor Bus is based on Twitter. The involved SWE services as well as connected geosensors are represented as user profiles of the Twitter platform.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Communications Applications; D.2 [**Software Engineering**]: Software Architectures

Keywords

Sensor Web, geosensor networks, SWE, Twitter

1. INTRODUCTION

Nowadays, sensors become smaller, cheaper, more reliable, more power efficient and more intelligent. Geosensors, as kinds of sensors delivering an observation with georeferenced location [33], are increasingly used in various applications ranging from environmental monitoring, precision agriculture to early warning systems [29]. The sensors utilized in these applications may be stationary or mobile, either on land, water or in the air and could gather data in an in-situ or remote manner. Due to this variety, a coherent infrastructure has become necessary to integrate heterogeneous sensors in a platform independent and uniform way. The Sensor Web is such an infrastructure for sharing, finding and accessing sensors and their data across different applications [25]. The Sensor Web is to sensors what the World Wide Web (WWW) is to general information sources - an infrastructure allowing users to easily share their sensor resources. It hides the underlying layers with the network communication details, and heterogeneous sensor hardware, from the applications built on top of it.

The Sensor Web Enablement (SWE) [5] initiative of the Open Geospatial Consortium (OGC)[1] standardizes Web Service interfaces and data encodings for the Sensor Web. In recent years, these SWE standards have demonstrated their practicability and suitability in various projects (e.g., [10, 27, 19, 34]) and applications (e.g., [15, 1, 8, 14]). However, due to the missing interoperability between the two layers (geosensor network and Sensor Web), it is currently not possible to dynamically install geosensors on-the-fly with a minimum of human configuration effort, to enable a plug & play of geosensors.

[1] http://www.opengeospatial.org

Dynamically installing geosensors on the Sensor Web in a plug & play manner requires advanced concepts. Generally, the SWE standards focus on interacting with the upper application level. They are designed from an application-oriented perspective. As a result, the interaction between the Sensor Web and the underlying geosensor network layer has not been sufficiently described yet. The Sensor Web is based on the WWW and its related protocols. On the other hand, sensor network technologies are based on lower-level protocols such as Bluetooth[2], ZigBee[3], the IEEE 1451 standards family [21] or proprietary protocols. From an application perspective, the SWE services encapsulate the sensor network and hide these lower-level protocols. Currently, the Sensor Web and geosensor network layer are integrated by manually building proprietary bridges for each pair of Web Service implementation and sensor type. This approach is cumbersome and leads to extensive adaption effort. Since the price of sensor devices is decreasing rapidly, the manual integration becomes the key cost factor in developing large-scale sensor network systems [3]. Concepts are missing which facilitate the connection of the two layers.

After giving an overview of the SWE standards (Section 2) and related work (Section 3), the concept of an intermediary layer between Sensor Web and geosensor network layer, the *Sensor Bus*, is described (Section 4). In a next step (Section 5) the Sensor Bus is implemented by the example of Twitter followed by an analysis of this implementation (Section 6). The paper ends with a conclusion and discussion of future work.

2. SENSOR WEB ENABLEMENT

The OGC is an industry consortium comprising over 380 members defining interoperable services for the Geospatial Web. The SWE initiative [5] as part of OGC's specification program develops standards to integrate sensors into the Geospatial Web. In particular, SWE incorporates data models for describing sensors (SensorML [4]) as well as gathered sensor data (Observations & Measurements [11]). The main Web Service interfaces are the Sensor Observation Service (SOS), the Sensor Alert Service (SAS), and the Sensor Planning Service (SPS). The SWE standards framework can be used to set up a Sensor Web.

The SOS [24] is designed for accessing real time as well as historic sensor data, and sensor metadata. Its operations allow users to request sensor data based on spatial, temporal and thematic filter criteria. Additionally, the SOS offers operations for inserting observations and sensors. Requested sensor data are encoded conforming to the Observations & Measurements standard and the sensor metadata are returned as SensorML documents. A complementary approach for accessing sensor data is offered by the SAS [31]. While the SOS follows the pull-based communication paradigm, the SAS is capable of pushing sensor data to subscribers. Subscribers of the SAS define particular alert conditions to specify the situations they are interested in and want to receive data accordingly (e.g., when sensor x measures temperature over 30° C). To control and task sensors the SPS [32] can be used. A common application of SPS is to define simple sensor parameters such as the sampling rate but also to define more complex tasks such as mission planning of satellite systems. Further on, the SPS offers a broad range of operations for managing submitted sensor tasks (e.g., deleting, updating and canceling tasks).

Also, service interfaces for the resource discovery within the Sensor Web have been developed and added to the SWE framework. The Sensor Instance Registry (SIR) and the Sensor Observable Registry (SOR) [20] enable searching for sensors, sensor datasets, or SWE services.

3. RELATED WORK

GeoSWIFT [22] is a three-layered architecture for distributed geospatial information infrastructures to realize the Sensor Web. It advocates the usage of OGC standards to expose sensors and sensor data. Additionally to a standardized Web Service interface for sensor data access, a server component is introduced which integrates and fuses heterogeneous sensing sources. However, the design of the integrator component is not described.

Sgroi et al. [28] developed a set of service interfaces usable as an application programming interface for sensor networks. Similar to the SWE framework of services, the approach follows a top down view on sensor networks independent of a particular implementation or hardware platform. Besides an essential query/command service, auxiliary services such as locationing, timing, and a concept repository are defined. By offering this functionality the approach focuses on the needs of wireless sensor networks. The SWE framework instead, focuses on serving general functionalities needed by geosensor network applications and is not specialized on wireless sensor networks.

The SOCRADES architecture [12] is designed to couple the Internet of Things infrastructure with a Service Oriented Architecture. The architecture comprises multiple services providing middleware functionality such as device managing, eventing or service discovery. The integration of sensors into the infrastructure is done by implementing sensor gateways which hide the communication protocol and expose the sensor functionality as device level Web Services. In contrast to the SWE framework, the operations of individual services are not standardized.

Emerging Sensor Web portals provide central web platforms where people can upload and share sensor data. These portals offer different kinds of visualizations on registered sensors and collected data. Examples for such systems are SensorMap and the underlying SenseWeb infrastructure [26], SensorBase [9] as well as Sensorpedia[4]. Besides mechanisms for integration and registration of sensors and the upload of sensor data, also the discovery of sensors is supported. However, the centralized approach of these platforms differs from the decentralized approach of SWE. Also, the approaches do not comprise interfaces for tasking sensors which is in scope of this work.

The Sensor Web Agent Platform (SWAP) [23] combines the paradigms of Web Services and Multi Agent Systems. By

[2] http://www.bluetooth.org
[3] http://www.zigbee.org
[4] http://www.sensorpedia.org

building on OGC's SWE framework, the proposed architecture shall improve the integration of arbitrary sensors into workflows on the application level. This is done by introducing a three tier architecture comprising sensor, knowledge and application layer. Different kinds of agents residing on the three layers provide certain functionality and facilitate the development of new applications. While this work facilitates the integration of sensors with applications, the integration of sensors with the Sensor Web is out of scope.

Also an agent based system is IrisNet [17]. It uses organizing agents to store sensor data in a hierarchical, distributed database and sensing agents which collect the sensor data. The authors envision a worldwide Sensor Web by focusing on data collection and query answering. The architecture lacks particular mechanisms for an easy integration of new sensors.

Similar to the goal of this work, the Global Sensor Network middleware [2] focuses on a flexible integration of sensor networks to enable fast deployment and addition of new sensors. Its central concept is the virtual sensor abstraction with XML-based deployment descriptors in combination with data access through plain SQL queries. GSN provides distributed querying, filtering, and combination of sensor data as well as the dynamic adaption of a system during runtime. However, service interfaces for tasking and controlling sensors are not provided to the application layer.

Similar to the GSN approach is Hourglass [30], which provides an architecture for connecting sensors to applications. It offers discovery and dataprocessing services and tries to hide internals of sensors from the user. It focuses on maintaining the quality of service of data streams.

None of the existing solutions address the seamless integration of sensors with standardized Web Service interfaces as defined by OGC. Aim of this work is to leverage SWE technology and its benefits by facilitating the integration of new sensors into the Sensor Web. Another unique characteristic of the approach presented here is the possibility to adapt the concepts to different communication infrastructures acting as base technology (e.g., XMPP or JMS). So in future, it might be considered to use existing systems as the basis of the proposed Sensor Bus and reuse their functionality.

4. THE SENSOR BUS ARCHITECTURE

In the following, the concept and architecture of the Sensor Bus, an intermediary layer integrating geosensor networks and the Sensor Web, are described. The architecture can be adapted to different communication infrastructures - an implementation based on Twitter is presented in Section 5.

Fig. 1 depicts an overview of the components involved in the Sensor Bus architecture. 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 supply 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). Also, it is possible that the gateway gives access to a whole sensor network by acting as a sink node.

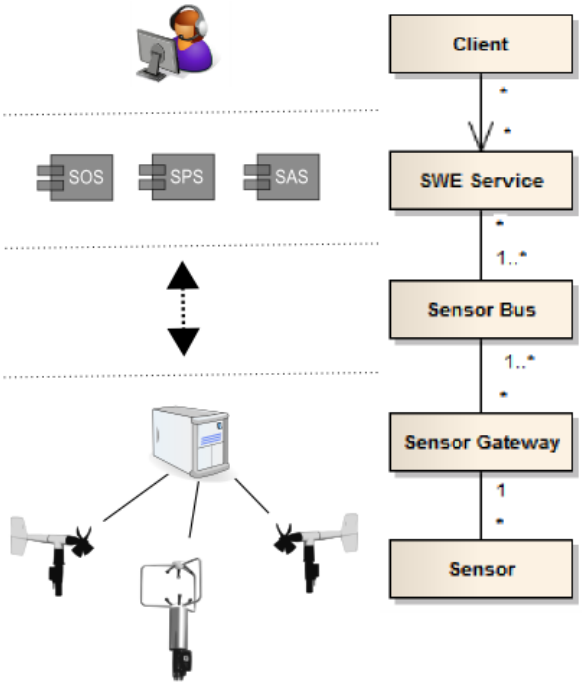

Figure 1: Sensor infrastructure stack

We propose that the intermediary layer is externally designed as a logical bus - the Sensor Bus (see Fig. 2). Aligned with the Message Bus pattern [18], the Sensor Bus incorporates (1) a common communication infrastructure, a shared set of (2) adapter interfaces, and a well-defined (3) message protocol.

The common communication infrastructure (1) is established through a publish/subscribe mechanism [16] based on the underlying messaging technology (e.g., Twitter or XMPP). Services as well as sensors can publish messages to the bus and are also able to subscribe to the bus for receiving messages in a push-based communication style. The underlying messaging technology takes care of forwarding the posted messages to the specific subscribed components. The different components (i.e., sensors and SWE services) can subscribe and publish through interfaces (2). 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 bus protocol (3).

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). While there is no single instance representing the Sensor Bus, the adapters of those components, together with the underlying messaging technology, form the Sensor Bus.

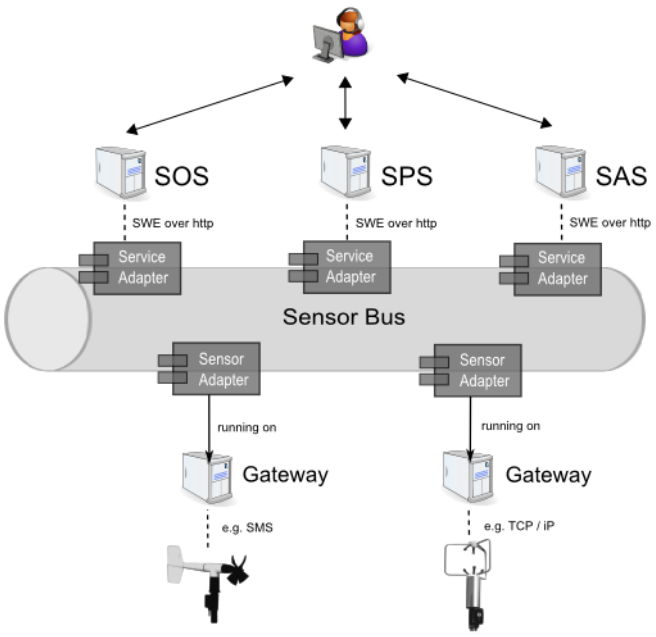

Figure 2: Structure of the Sensor Bus

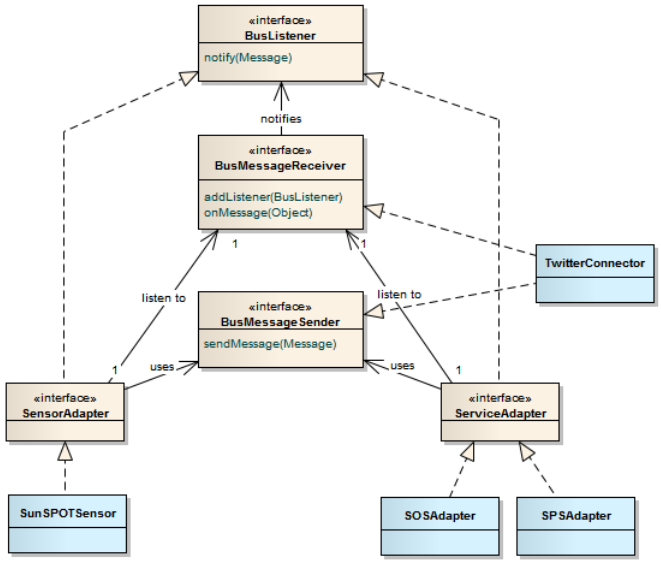

Figure 3: Sensor Bus core components as UML class diagram

The interfaces which are used to realize the Sensor Bus are depicted in Fig. 3. The `SensorAdapter` (e.g., an adapter for the SunSPOT[5] 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` hide from the underlying communication infrastructure of the bus. The `TwitterConnector` in Fig. 3 is an example implementation of the two interfaces to realize the Sensor Bus based on Twitter[6]. 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., instant messaging systems). 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.

The interactions between the geosensor network layer, the Sensor Web and the intermediary layer are realized through particular bus messages. A detailed analysis of interaction patterns which emerge when introducing the intermediary layer is conducted in [6]. The necessary bus messages are listed in Table 1. The message format is compact to preserve bandwidth and system resources. Single message fields are divided by a separator sign.

The subscription of a service at the Sensor Bus is conducted by a sequence of messages. First, the *RegServ* message is called to publish the URL of the service. Subsequently, *SubServ* messages are sent over the bus to subscribe the service for certain sensors. In future, more advanced subscription parameters are possible. For example, a service could subscribe for sensors of a particular geographic region or for sensors observing particular phenomena or features.

The subscription of a service for a particular sensor requires the existence of unique identifiers for the sensor. Unified Resource Identifiers (URIs) are therefore used here. The discovery of sensors and the look-up of corresponding URIs can be done through the Sensor Instance Registry (SIR) [20]. As a SWE service, the SIR is also registered at the Sensor Bus and is notified when new sensors appear. It is automatically supplied with their metadata and fills its search indexes so that clients can discover them.

The subscription of a service at the Sensor Bus by the means of the *RegServ* and *SubServ* message is only necessary and meaningful if the Sensor Bus or the underlying messaging technology is capable of managing the associations between service and sensor. The realization of the Sensor Bus using Twitter 5 is light-weight and does not support such functionality. Here, the service adapter 'knows' in which sensors it is interested and handles incoming messages accordingly.

To subscribe a sensor at the Sensor Bus and publish its existence the *RegSen* message is sent. It comprises the sensor identifier as well as the URL of the sensor description, a SensorML document. The message is forwarded to the services which are subscribed for the sensor.

For publishing new data the sensor adapter transmits the *PubData* message via the Sensor Bus containing the time when the data was observed as well as the data itself. Service adapters receive this message and transform the data into the service specific protocol. For example, the message `PubData*sunspotA1*2010-03-02T15:52:43*-2` is transform-

[5] http://www.sunspotworld.com
[6] http://www.twitter.com

Table 1: Sensor Bus messages.

Interaction	Bus Message Protocol
Service Registration	1. RegServ*<service URL> 2. SubServ*<service URL>*<sensor A id> 3. SubServ*<service URL>*<sensor B id> ...
Sensor Registration	SenReg*<sensor id>*<sensor description URL>
Data Publication	PubData*<sensor id>*<time tag>*<data>
Sensor Tasking	1. PubTask*<sensor id>*<task id> 2. TaskParam*<task id>*<param 1>*<value 1> 3. TaskParam*<task id>*<param 2>*<value 2> ... X. DoTask*<task id>

ed by a Sensor Observation Service (SOS) adapter to an InsertObservation request as shown in Listing 1. Information which is not contained in the message itself but necessary within the InsertObservation request (e.g., the unit of measure and the URL of the observed feature) is taken from the sensor description.

The data are then collected and stored by the SOS and henceforth available to clients via the standardized SOS interface. It can be accessed and retrieved in a pull-based manner. To provide the data in a push-based way, a Sensor Alert Service can be registered at the Sensor Bus. The SAS receives the incoming data, filters it by certain predefined criteria and directly forwards it to interested clients.

```
<InsertObservation service='SOS' version='1.0.0'>
  ...
  <om:Observation>
    ...
    <gml:timePosition>
      2010-03-02T15:52:43
    </gml:timePosition>
    ...
    <om:procedure
      xlink:href="http://server/sensors#sunspotA1"/>
    <om:observedProperty
      xlink:href="urn:ogc:phenomenon:temperature"/>
    <om:featureOfInterest
      xlink:href="http://server/features#myHouse"/>
    <om:result
      xsi:type="gml:MeasureType"
      uom="urn:ogc:uom:deg">
      -2
    </om:result>
  </om:Observation>
</InsertObservation>
```

Listing 1: Example of an SOS InsertObservation request.

The tasking of a sensor is triggered by a client request to a Sensor Planning Service (SPS). A client sends a Submit operation request to the SPS to submit a sensor task. An example for such a request is shown in Listing 2. The camera 'myCam123' is tasked to change its focal length to '22.0' millimeters and to change its looking direction to 'North'. The allowed values for the task parameterization can be requested by the client through an afore invoked DescribeTasking operation request. The service adapter of the SPS transforms the Submit request to a sequence of sensor tasking messages. It starts with a *PubTask* message to publish an identifier for the task and the identifier of the sensor (e.g., 'myCam123') which shall execute the task. In subsequent *TaskParam* messages the parameters of the task are transmitted. For example the second parameter of the Submit request of Listing 2 is transformed to `TaskParam*myCam123_task1*LookingDirection*North`. Finally, the task is invoked by sending the *DoTask* message.

```
<Submit service='SPS' version='2.0.0'>
  <sensorIdentifier>myCam123</sensorIdentifier>
  <taskingParameters>
    <ParameterData>
      <encoding>
        <TextEncoding tokenSeparator=","/>
      </encoding>
      <values>22.0,North</values>
    </ParameterData>
  </taskingParameters>
  ...
</Submit>
```

Listing 2: Example of an SPS Submit request.

The sensor adapter of 'myCam123' registered at the Sensor Bus receives the tasking message sequence and translates it to the concrete sensor protocol (e.g., through specific sensor drivers). Subsequently, it is forwarded to the sensor gateway and eventually to the sensor.

5. IMPLEMENTATION OF THE SENSOR BUS USING TWITTER

In general, the outlined architecture of the Sensor Bus, as the intermediary layer, is adaptable to different underlying messaging technologies. We implemented the Sensor Bus in four different ways. These implementations are using Internet Relay Chat (IRC), Extensible Messaging and Presence Protocol (XMPP), Java Message Service (JMS), or Twitter as the underlying messaging technology to establish the publish/subscribe mechanism of the Sensor Bus. In this article we present the implementation of the Sensor Bus using Twitter. An extensive evaluation and comparison of all applied technologies is current work in progress.

To plug a sensor into the Sensor Bus a sensor administrator implements an adapter (Section 4) for the particular type of her/his sensor. This adapter has only to be implemented for each type of sensor once. In this work, we created an adapter for the SunSPOT sensors.

In case of Twitter the services and sensors are available as Twitter profiles. For the realization of the Sensor Bus, the sensor administrator must create a Twitter profile for the sensor since an automatic creation is not possible. The *description URL* of the sensor's Twitter profile points to its metadata description stored at a web-accessible location so that it can be accessed by services at anytime. The sensor description is a SensorML encoded document containing for example detailed information about the sensor's geographic location. The sensor adapter is usually deployed on the computing unit of the sensor gateway. Here, a SunSPOT, which may act as a network sink to multiple SunSPOTs, is connected via USB to a computer with network connection. This computer represents the sensor gateway and hosts the sensor adapter. This adapter is accompanied by a configuration file containing information about the endpoint of the bus communication infrastructure (e.g., address, port, or in this case the Twitter account identifier). Finally, when the sensor adapter is started, it posts a *RegSen* message to the mirco blog of the sensor's Twitter profile. Subsequently, the sensor forwards its measured data to the sensor gateway which uses the sensor adapter to send the data to the Sensor Bus. The specific implementation of the `BusMessageSender` (Section 4) posts these data contained in a *DataPub* message as a tweet to the micro blog of the sensor.

To attach a service to the Sensor Bus the provider has to implement an adapter[7] for the service. In this work, we created an exemplary adapter for the Sensor Observation Service. Additionally, the service provider has to create a Twitter account for the service.

The runtime instance of the service adapter and the actual Web Service are usually but not necessarily running on the same machine. After starting the service adapter, it registers the service at the bus. The service adapter is accompanied by a configuration file defining the sensors (i.e., the Twitter account IDs of those sensors) for which it shall be subscribed. To establish the publish/subscribe mechanism in case of Twitter, the Twitter account of the service is registered as a follower at the sensor's account and vice versa. Since the Twitter architecture is pull-based, the service adapters and sensor adapters have to regularly check the micro blogs (available as feeds) of the accounts which they are following. So, if a sensor adapter posts a new *DataPub* message to its micro blog the service adapters, which are following, get aware of that by checking the senor's feed, transform the message to the service specific protocol and forward it.

6. ANALYSIS OF TWITTER IMPLEMENTATION

Building the Sensor Bus on Twitter enables reusing functionality offered by the messaging platform. For example, security mechanisms can be easily incorporated in the implemented approach, since authentication functionality is provided by Twitter. Also, scalability and reliability of the Sensor Bus is managed by Twitter.

However, there are some disadvantages in using Twitter as the communication infrastructure of the Sensor Bus. In general, the pull-based design of Twitter does not allow a true push-based Sensor Bus. Instead, the message retrieval has to be realized by regularly submitted API queries. Another disadvantage is the limited update rate of Twitter's search index which means that for example a data publication message posted by a sensor adapter is not instantly accessible by a service adapter.

Further on, there are functional limitations related to a Twitter profile. Besides the restriction of the length of a single message (i.e., a so-called 'tweet') to 140 characters, a Twitter account (a) cannot submit more than 150 requests per hour and (b) cannot send more than 1.000 tweets a day. Restriction (a) results in a limited update rate for the services listening to the Sensor Bus. By using the method *statuses/friends_timeline* of the Twitter API a service adapter can maximally query 150 times an hour the recently posted tweets of all sensors it is following[8].

A more significant disadvantage is restriction (b). In the current design of the Sensor Bus implementation, a sensor posts one data value per tweet. Due to (b), this results in a maximum sampling rate of around 40 measurements per hour. In many sensor network applications, this would be unacceptable.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we outline the need for mechanisms to close the gap between geosensor networks and the Sensor Web. We propose to achieve this integration by introducing an intermediary layer realized as a message bus - the Sensor Bus. The Sensor Bus incorporates a publish/subscribe mechanism and defines certain messages to realize the interaction with the Sensor Web layer and geosensor network layer.

An implementation of the Sensor Bus is published as open source to the $52°North$ Sensor Web community[9]. We have implemented the adaptable Sensor Bus concept in different ways based on instant messaging technologies or message oriented middleware. An in-depth evaluation of the different approaches including extended scalability and performance tests is planned future work.

In this work, we present an implementation of the Sensor Bus based on Twitter. Building the Sensor Bus on existing messaging infrastructures is beneficial to reuse functionality such as authentication, as well as scalability and reliability management. However, the limited functionality of Twitter (e.g., the maximum of 140 characters per tweet or the rate limiting for API queries) does not leverage the full functionality of SWE services. Use cases of a certain complexity cannot be handled by such a solution (e.g., satellite mission planning).

We aim at contributing the concept of the Sensor Bus to OGC's SWE initiative. The Sensor Bus enables an automatic notification of SWE services about changes in sen-

[7]The mid-term aim is to establish a library of adapters for certain types of sensors as well as services. This will allow service providers and sensor vendors to reuse existing adapter implementations to plug their components into the bus.

[8]To solve this issue Twitter offers the possibility for applying to *whitelist* certain accounts or IP addresses. A whitelisted entity can submit 20.000 requests per hour.

[9]`http://www.52north.org/swe`

sor metadata as well as newly gathered sensor data. So far, new data or changed metadata has to be communicated separately to the different SWE services which may lead to inconsistencies within the Sensor Web quickly. Further developments will base the Sensor Bus on OGC's Sensor Event Service [13] which realizes a publish/subscribe architecture based on Web Service Notification. This will result in an intelligent Sensor Bus allowing complex event processing on sensor data streams.

A next step will be the development of a mechanism for plug & play of geosensors based on the Sensor Bus architecture. Currently, we develop a SensorML application schema defining a generic sensor interface description to automatically generate the communication logic for adapting sensors to the Sensor Bus. Based on these developments, semantic challenges in the context of sensor plug & play [7] will be identified and tackled. Finally, the approach has to be applied in real-world scenarios to demonstrate its benefits in sensor asset management. As the project develops, we anticipate that both our design and our research agenda will evolve as new issues and opportunities arise.

Acknowledgment

This work is financially supported by the project "Flexible and Efficient Integration of Sensors and Sensor Web Services" funded by the European Regional Development Fund (ERDF) for NRW (contract number N 114/2008) of the European Union, as well as the $52°North$ Sensor Web community which envisions a real time integration of heterogeneous sensors into a coherent information infrastructure.

8. REFERENCES

[1] A. Aasa, O. Järv, and R. Ahas. Developing a model to determine the impacts of climate change on the geographical distribution of tourists. In K. Lammert and L. Arend, editors, *Sensing a Changing World 2008*, pages 55 – 58. Wageningen University, 2008.

[2] K. Aberer, M. Hauswirth, and A. Salehi. A middleware for fast and flexible sensor network deployment. In *Proceedings of the 32nd international conference on Very large data bases*, 2006.

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

[4] M. Botts. OGC Implementation Specification 07-000: OpenGIS Sensor Model Language (SensorML). Technical report, Open Geospatial Consortium, 2007.

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

[6] 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.

[7] A. Broering, K. Janowicz, C. Stasch, and W. Kuhn. Semantic Challenges for Sensor Plug and Play. In J. Carswell, S. Fotheringham, and G. McArdle, editors, *Web & Wireless Geographical Information Systems (W2GIS 2009), 7 & 8 December 2009, Maynooth, Ireland*, number 5886 in LNCS, pages 72–86. Springer, 2009.

[8] A. Broering, E. H. Jürrens, S. Jirka, and C. Stasch. Development of Sensor Web Applications with Open Source Software. In *First Open Source GIS UK Conference (OSGIS 2009), 22 June 2009, Nottingham, UK*, 2009.

[9] K. Chang, N. Yau, M. Hansen, and D. Estrin. sensorbase.org - A Centralized Repository to SLOG Sensor Network Data. In *International Conference on Distributed Computing in Sensor Networks (DCOSS) / EAWMS Workshop*, San Francisco, USA, June 2006.

[10] L.-K. Chung, B. Baranski, Y.-M. Fang, Y.-H. Chang, T.-Y. Chou, and B. J. Lee. A SOA based debris flow monitoring system - Architecture and proof-of-concept implementation. In *The 17th International Conference on Geoinformatics 2009*, Fairfax, USA, 2009.

[11] S. Cox. OGC Implementation Specification 07-022r1: Observations and Measurements - Part 1 - Observation schema. Technical report, Open Geospatial Consortium, 2007.

[12] L. de Souza, P. Spiess, D. Guinard, M. Kohler, S. Karnouskos, and D. Savio. Socrades: A web service based shop floor integration infrastructure. *Lecture Notes in Computer Science*, 4952:50, 2008.

[13] J. Echterhoff and T. Everding. OGC Discussion Paper 08-133: OpenGIS Sensor Event Service Interface Specification. Technical report, Open Geospatial Consortium, 2008.

[14] T. Foerster, A. Broering, S. Jirka, and J. Müller. Sensor Web and Geoprocessing Services for Pervasive Advertising. In J. Müller, P. Holleis, A. Schmidt, and M. May, editors, *Proceedings of the 2nd workshop on pervasive advertising - in conjuction with Informatik 2009*, pages 88 – 99, Lübeck, October 2009.

[15] S. Fruijtier, E. Dias, and H. Scholten. Geo Mindstorms: Investigating a sensor information framework for disaster management processes. In L. Kooistra and A. Ligtenberg, editors, *Sensing a Changing World 2008*, pages 50 – 54. Wageningen University, 2008.

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

[17] P. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. Irisnet: An architecture for a worldwide sensor web. *IEEE Pervasive Computing*, pages 22–33, 2003.

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

[19] S. Jirka, A. Broering, and C. Stasch. Applying OGC Sensor Web Enablement to Risk Monitoring and Disaster Management. In *GSDI 11 World Conference, Rotterdam, Netherlands*, June 2009.

[20] S. Jirka, A. Broering, and C. Stasch. Discovery Mechanisms for the Sensor Web. *Sensors*, 9, 2009.

[21] K. Lee. IEEE 1451: A Standard in Support of Smart Transducer Networking. *Instrumentation and*

Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE Volume 2, 2:525 – 528, 2000.

[22] S. Liang, V. Toa, and A. Croitoru. Sensor Web and GeoSWIFT - An Open Geospatial Sensing Service. In *International Society for Photogrammetry and Remote Sensing XXth Congress–Geo-Imagery Bridging Continents*, pages 12–23, 2004.

[23] D. Moodley and I. Simonis. A New Architecture for the Sensor Web: The SWAP Framework. In *5th International Semantic Web Conference ISWC 2006*, Athens, Georgia, USA, 2006.

[24] A. Na and M. Priest. OGC Implementation Specification 06-009r6: OpenGIS Sensor Observation Service (SOS). Technical report, Open Geospatial Consortium, 2007.

[25] S. Nittel. A Survey of Geosensor Networks: Advances in Dynamic Environmental Monitoring. *Sensors*, 9:5664 – 5678, 2009.

[26] A. Santanche, S. Nath, J. Liu, B. Priyantha, and F. Zhao. Senseweb: Browsing the Physical World in Real Time. In *International Conference on Information Processing in Sensor Networks (IPSN)*, Nashville, USA, 2006.

[27] G. Schimak and D. Havlik. Sensors Anywhere - Sensor Web Enablement in Risk Management Applications. *ERCIM News*, The Sensor Web - Bringing Information to Life(76):40–41, 2009.

[28] M. Sgroi, A. Wolisz, A. Sangiovanni-Vincentelli, and J. Rabaey. A Service-Based Universal Application Interface for Ad Hoc Wireless Sensor and Actuator Networks. In W. Weber, J. Rabaey, and E. Aarts, editors, *Ambient intelligence*. Springer Verlag, 2005.

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

[30] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, and M. Welsh. Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. Technical report, Harvard University, EECS, 2004.

[31] I. Simonis. OGC Best Practices 06-028r3: OGC Sensor Alert Service Candidate Implementation Specification. Technical report, Open Geospatial Consortium, 2006.

[32] I. Simonis. OGC Implementation Specification 07-014r3: OpenGIS Sensor Planning Service. Technical report, Open Geospatial Consortium, 2007.

[33] C. Stasch, K. Janowicz, A. Broering, I. Reis, and W. Kuhn. A Stimulus-Centric Algebraic Approach to Sensors and Observations. In *3rd International Conference on Geosensor Networks*, Lecture Notes in Computer Science. Springer, 2009.

[34] C. Stasch, A. C. Walkowski, and S. Jirka. A Geosensor Network Architecture for Disaster Management based on Open Standards. In M. Ehlers, K. Behncke, F. W. Gerstengabe, F. Hillen, L. Koppers, L. Stroink, and J. Wächter, editors, *Digital Earth Summit on Geoinformatics 2008: Tools for Climate Change Research.*, pages 54–59, 2008.