18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Cascading and replicating the OGC Sensor Observation Service

Havlik, D.[1], Schimak, G.[1] and Bleier, T.[1]

[1] Austrian Research Centers GmbH. - ARC, Donau City Strasse 1; A1220 Vienna, Austria
Email: denis.havlik@arcs.ac.at

Abstract: The Sensor Service Architecture (SensorSA) developed by the Sensors Anywhere (SANY) Integrated Project demonstrates the feasibility of building full-fledged Environmental Information Systems without proprietary protocols. In a quest for maximal interoperability, the SensorSA embraces the sensor-specific service interfaces, data encodings and XML schemata of the OGC Sensor Web Enablement (OGC SWE) framework of open standards. In particular, the real-time and archived observations from sensors and other sensor-like data sources are published through OGC Sensor Observation Service (SOS).

Unfortunately, the OGC SWE does not foresee any standard mechanisms for data replication, caching and offsetting the load to a higher level service (cascading). In this article, we discuss the need for caching, replication and service cascading in Environmental Information Systems; demonstrate how to support these tasks within the frame of the current (v1.0) SOS specification, and propose simple extensions that could greatly improve the efficiency and robustness of SOS cascading and replication.

In order to simplify the problem at hand, we presume that most SOS operations responses can be adequately handled through existing HTTP caching mechanisms and concentrate on a simplified problem of pre-fetching and caching the response to "Get Observation" operation.

Keywords: *Sensor Web Enablement (SWE), Sensor Observation Service (SOS), Environmental Information Systems(EIS)*

INTRODUCTION

In the past decade Environmental Information Systems (EIS) have undergone a technical (r)evolution through a paradigm shift towards Service Oriented Architecture (SOA) (Erl, 2008) and inclusion of sophisticated processing capabilities (Usländer, 2008b). SOA based solutions are inherently well-suited for building of large decentralized networks and networks-of-networks, such as those envisaged in Global Earth Observation System of Systems (GEOSS), the Global Monitoring for Environment and Security (GMES); "Infrastructure for Spatial Information in the European Community" (INSPIRE) (EC, 2007); and "Shared Environmental Information System" (SEIS) (EC, 2008). However, none of the existing solutions for sensor-specific SOA architectures and data models have been widely accepted so far.

Sensor Service Architecture (SensorSA) developed within Sensors Anywhere (SANY) (Havlik, 2006) Integrated Project demonstrates the feasibility of building full-fledged EIS without proprietary protocols. SensorSA builds on the foundations of the "Reference Model for the ORCHESTRA Architecture" (Usländer, 2007); embraces the sensor-specific service interfaces and XML encodings (SensorML, TML, O&M) of the OGC Sensor Web Enablement (SWE) (Botts 2007, Simonis, 2008); defines restrictions, extensions and interpretations of SWE data- and meta-information models; and promotes event-based interaction models. More information on SensorSA can be found in (Usländer, 2009). Full SensorSA v1 specifications (Usländer, 2008) and related documents (e.g. requirements) are available on the SANY web site (http://sany-ip.eu).

Most of the data flow in OGC SWE and SensorSA networks is generated by SOS services. In many cases, the reliability and performance of a network can therefore be greatly improved through pre-fetching and replication of the data served through SOS servers (Figure 1). In SensorSA, this idea has been explored through "Cascading SOS" (SOS-X) (Havlik, 2008; Havlik 2009).

The Sensor Observation Service (Na & Priest 2007) specification defines four SOS Profiles: "Core", "Transactional", "Enhanced", and "Entire". The Core Profile defines following mandatory operations:

- GetCapabilities, gives an overview of the supported operations and indicates which data is offered by the service.
- DescribeSensor, returns a SensorML or TML description of one specific sensor, sensor system or data producing procedure.
- GetObservation, provides access to actual observations, encoded in observations and Measurements (O&M) XML dialect. This operation allows filtering of data by spatio-temporal constraints, phenomenon and value constraints.

The Transactional Profile (SOS-T) adds support for registering new procedures ("RegisterSensor") and uploading new observations ("InsertObservation"). The Enhanced Profile defines additional operations for accessing the observations (GetResult and GetObservationByID), describing the result model (DescribeResultModel) and obtaining additional meta-information on spatio-temporal extent of the data set (GetFeatureOfInterest, GetFeatureOfInterestTime, DescribeFeatureType). If a SOS implementation supports all SOS operations, it implements the Entire Profile of the SOS specification.

The reference implementation of the SOS specification developed by 52-North (http://52north.org/), which is also used as a front-end for the SOS-X, currently implements the Core Profile, Transactional Profile, GetResult and GetFeatureOfInterest operations.

In this paper, we ignore the SOS operations which are likely to return relatively small amounts of data (e.g. "GetCapabilities", "DescribeSensor") and concentrate on actual payload of observations returned by SOS "GetObservations" operation. In addition, we also ignore some peculiarities of the SOS specification that complicate the task of designing SOS client applications, such as the great variety of allowed data models for observations and the lack of server side paging support.

Another simplification involves the Observation and Measuremetns (O&M) (Cox, 2007) data model: all discussion in this paper is based on the default O&M model used by 52 North SOS service, which is also the only data model currently supported by Cascading SOS. The data returned by "GetObservation" operation is presumed to be encoded in Q&M XML dialect and to consist of a (relatively small) "header" part with description of the data set, and (potentially very large) "payload" part. Payload is basically a table of data, with two compulsory columns (measurement reference time and the geo-localization information expressed through Feature of Interest), and one or more columns representing the actual observations (values, status information, etc). Caching of more complex data models is feasible, but will not be implemented by SANY.

1. CASCADING SOS AT WORK

SensorSA is designed with scalability in mind. It can be used to implement a local EIS, but its real targets are national and multi-national networks of networks. Some of the real-world issues encountered in large decentralized networks include:

- The network connection between the information provider and information users is a-priory unreliable and in some cases prohibitively slow.
- The number of potential information users is enormous. Most institutional information providers can assure reliable service in "normal circumstances", but publicly available services can be overloaded in the case of an emergency. Services provided by independent data providers (especially those run by amateurs) could be easily overwhelmed by relatively low number of users.
- Interactive use of services is limited by a combination of (low) bandwidth and (large) size of the transferred data.

Figure 1 illustrates these issues with three example users (organizations), using the data from three SOS

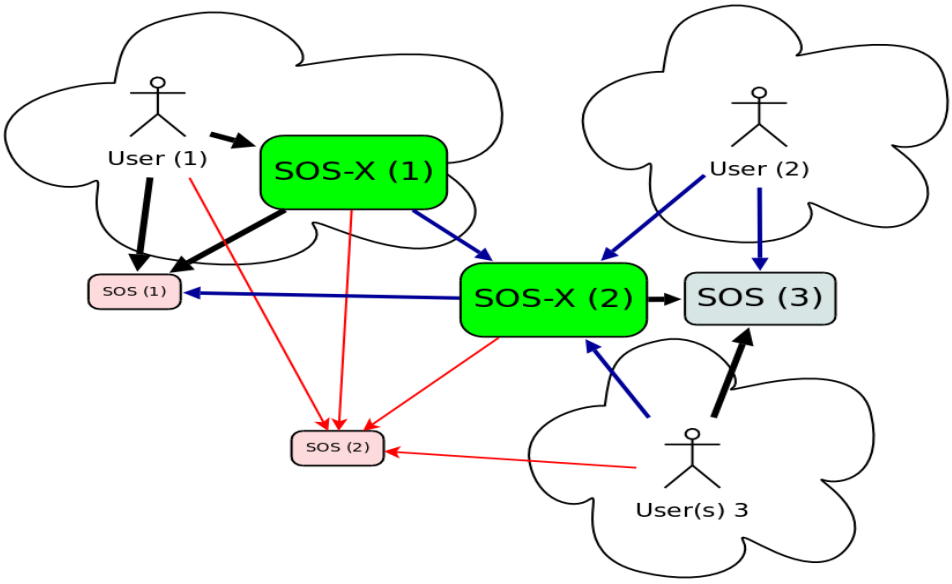

Figure 1. Load balancing and traffic optimization with Cascading SOS. Size and color of the SOS servers represent the server reliability. Size and colors of the arrows represent the reliability of the network connections.

servers and two cascading SOS servers.

- SOS 1 and SOS2 services are "local" services hosted on sub-standard servers. In addition, the SOS 2 server is also connected through unreliable network connection.
- SOS 3 and both SOS-X servers are considered powerful enough for daily work and smaller emergencies.
- SOS-X 2 server primarily acts as an alternative source of SOS 3 data. The free capacity of the SOS-X 2 service is used to replicate the data from other available sources (SOS 2, SOS 3).
- SOS-X 1 server gets most of its data through SOS-X 2 server, with the possible exception of the SOS 1 data, which may be fetched directly.

The best case scenario (fast and reliable network connection to reliable SOS service) is illustrated by User 1 working on data provided by SOS 3 service. The worst case scenario (slow and unreliable connections to an unreliable SOS service) would be an emergency situation where all three users attempt to directly fetch the data from the SOS 2 service.

Smart SOS clients could automatically use the SOS-X 2 service as an alternative source of SOS 3 data, or even use both servers in parallel to achieve higher transfer rates. However, this kind of operation is not supported by SOS specification and has not been implemented in any of the existing SOS client applications

we are aware of. An alternative way to improve the SOS 3 performance in emergency situations has been described in (Havlik 2008): the original SOS 3 server is moved to the background, and replaced by a group of SOS-X servers. Each of the SOS-X servers is configured to act as an exact replica of the original system, and a load balancer assures the load is evenly distributed over all servers. Thanks to SOS-X caching mechanism, most of the requests can be handled by one of the SOS-X servers, without the need to consult the original SOS service.

User 2 and User 3 (in Figure 1) do not have a cascading SOS in-house. Both users have good connections to the SOS 3 server, and to the SOS-X 2 service. Let us assume that the SOS 3 serves the data from a medium sized Air Quality Monitoring System with 100 Monitoring stations. Each station measures five parameters once in 30 minutes, for a total of 1000 observations every hour. These observations are kept in the archive for decades. An archive with a single year of observations contains almost 9 Million observations. For simplicity, let us assume that we need 100 bytes to encode a single observation and associated meta-information. Transferring almost 900MB over Internet connection is feasible, but slow. Transferring one year of observations from the SOS 3 service to either user 2 or user 3 could take over two hours with a 1Mbit connection. For a comparison, transferring a single day of data could be done in 20 seconds, and a single set of ½ hour measurements (or a yearly set of observations from a single sensor) could be transferred in less then a second.

Equipped with an intelligent SOS client[1], User 2 and User 3 should be able to download the data set from the SOS 3 and explore it off-line. In addition, they can interactively explore small subsets of data with an SOS client (e.g. latest set of observations from all stations, or one year of observations from a single station.

SOS 1 and SOS 2 may contain data from small monitoring networks with only ten monitoring stations. These servers are weak in terms of processing power, and connected to the Internet through a rather slow and unreliable connection. Assuming the 100 kbit/s connection, one year worth of data from these services to User 3 also takes two hours, but the service is more likely to become overloaded than SOS 3. Serving the same data set from SOS-X 2 service (assuming 1 MBit connection) can be done in approximately 12 minutes.

Finally, User-1 has installed an SOS-X server on his local network (LAN). On a Gigabit Ethernet LAN, User 1 can expect up to 1000 times faster transfer rates for SOS 3 data (served through in-house SOS-X server) than Users 2 and User 3. In the praxis, the XML encoding and decoding may significantly slow down the transfer rate, but the User 1 should nevertheless be able to explore the SOS 3 data set interactively.

2. CACHING AND REPLICATION OF ENVIRONMENTAL DATA

In section 1 of this paper, we presumed that caching and replication works perfectly for all data sources, all the time. This is not necessarily the case. Assuring data integrity in a general purpose distributed database is a non-trivial issue, and far beyond the scope of the SensorSA and SANY. Fortunately, the nature of Environmental Monitoring greatly simplifies the problem at hand. Caching and replication becomes almost trivial under following assumptions:

1. Most of the data is archived and does not change over time.
2. The propagation of data changes is always from original data sources to replicas.
3. The service interface provides a simple mechanism for retrieving only data which changed since the last replica/cache synchronization.

The first assumption is valid in most cases. In real world, changing of observations is often very improbable. For example, the air quality system mentioned above produces a small amount of data every 30 minutes, and stores this data in large archives over years. The raw data undergoes a “quality assurance” (QA) process, which may result in invalidating or even correcting part of the data. The QA process is finalized within a couple of hours, and the final (quality controlled) data is unlikely to be changed in the future. In many cases, only the final data will be freely available on the Internet.

Nevertheless, the SOS specification allows changing of the data associated with the observations. Moreover, the observations do not necessarily need to be a result of a physical measurement process. In fact, SOS specification allows embedding of arbitrary data generators (e.g. models, fusion engines, or even random

1 Client application should be capable of requesting the data in smaller chunks.

number generators) within the SOS. Unfortunately, the SOS does not provide the information on the volatility of observations, at least not in a form that could be easily used by the cascading SOS[2].

The second assumption is valid by SOS-X design: caching of the write operations is dangerous, and may lead to deadlocks and data inconsistencies. In addition, the SOS-X can generate new data, e.g. by averaging the input obtained from one or more sources. This should not be edited. As a consequence, SOS-X only advertises its data sources, and does not implement the Transactional SOS Profile..

Unfortunately, the last assumption is invalid for currently available (SOS v. 1.0) services. As mentioned in the introduction, the "GetObservation" call usually returns a document encoded in "Observations and Measurements" (O&M) XML dialect. 52 North SOS servers and cascading SOS encode the "payload" as a table of data, where each row in this table corresponds to a single observation. The first two columns in this table localize the observation in time and two dimensional space (Feature of Interest = FoI). Further columns contain the actual measurements and various associated meta-information. For example, a simple temperature observation may be expressed as: (1) time, (2) Feature of Interest (FoI), measured temperature in °C, and the status/validation information. The observation may, but often does not contain additional (meta-) information which could be used to retrieve the data changed since last synchronization. In particular:

- the observation often does not contain an explicit unique key. Consequently, one can not always distinguish between "new" observation which needs to be added to the database, and the "altered" observation which replaces an older version of the same data set.
- Second, the Observations usually do not contain a time stamp of the "last data change".
- Finally, the SOS Specification does not explicitly forbid deleting of observations. Currently, there is no way to request a list of deleted data sets from the SOS server.

Finally, an SOS service may drop the observations from the database e.g. in order to free storage space. For example, the SensorSA Data Acquisition System (Havlik, 2009) is an experimental data logger with a SOS interface. This kind of device is resource-depleted and usually keeps the data for a relatively short period of time (e.g. one day to one month). Currently, there is no standardized way for advertising the time span for which the data remains valid, nor for advertising the time span for which the data will be accessible

3. REPLICATION-FRIENDLY SOS

In the previous section, we indicated the key issues preventing reliable caching and replication of observations in SWE- and SensorSA- compliant service networks. In this section, we propose some ways of overcoming these shortcomings. The strategies for solving these issues split up into three categories:

- First, a service provider may decide to avoid some "bad practices" without actually advertising this fact at the SOS level. For example, the service provider may decide to avoid deleting of data on his SOS server, keep the archived data available indefinitely, or to assure the uniqueness of an observation for a certain combination of parameters (e.g. timestamp + FoI + observed property). This kind of solutions is easy to implement and therefore makes sense for an R&D project. Unfortunately, this is also a very fragile solution and should not be used in an operational environment.
- Second, a service provider may decide to add missing meta-information to SOS output without changing the SOS specification. The SOS data model is very flexible, and this strategy can indeed solve most of the issues mentioned in section 2. Table 1 summarizes the types of additional meta-information which could be provided by the SOS, and suggests the strategies for advertising and using this meta-information.
- Finally, the SOS service interface may be changed to provide better support for cascading and replication of what? In particular, a service request which returns all changes (new, modified and deleted observations) would greatly improve the feasibility of caching and replication.

4. EVENT DRIVEN REPLICATION

SensorSA foresees two main types of interaction (Muehl, 2006): request/reply and an event driven interaction model. In the request/reply interaction model, the information consumer (client) "pulls" the information from the information provider (server) and in the event driven interaction model, the information provider (producer of notifications) notifies the information consumer of a new "observation" (event).

2 In some cases, this information could be extracted from the "processing" part of the process description.

In an event-driven interaction model, the information consumer declares its interest in observations of a certain type, chooses between digest mode and immediate change propagation, and waits until the information provider sends a notification. The idea of event driven information dispatching is currently implemented and tested in Sensor Service Architecture Data Acquisition System (SensorSA DAS) prototype (Havlik, 2009).

Table **1**. Meta-information for caching and replication

Name	Importance	Position	Discussion
Unique Key	required	Procedure, Observation	Unique reference for an observation. Without an unique key, it is impossible to match the "old" version of an observation with the updated one. Unique key could be an additional column in GetObservation payload, or a combination of existing columns (e.g. time+FoI). The unique key is procedure and/or service specific, and should be announced at the procedure level (DescribeSensor/SensorML).
Modification Time	required	Observation	The time at which the information has been published at the SOS service. This timestamps can be used by client applications (e.g. SOS-X) for filtering the result set, and requesting only the data which changed since the last synchronization. If not advertised, the modify time could be presumed equal to the time at which the measurement was made (sampling time), or to the time at which the measurement analysis has been performed (result time). In many cases this is not true, e.g. because the observations may be collected in-house and published once in a week, or because the data was first published as "raw" data, and the quality assurance (status) related parameters changed at a later time.
Time to Live (TTL)	optional	Observation	The time until which the observation may be considered valid. An observation with expired TTL should be discarded from the cache and re-fetched from the original server. This information may be specific to observation. For example, the raw air-quality observations may be published with an TTL of 24 hours, and the final quality assured observations with the TTL of one month or more. If not advertised, the TTL is assumed to be "indefinitely".
Frequency of updates	optional	Procedure	Expected frequency at which new data will be made available at the server. This information can be used by client applications to optimize the frequency of service calls. The frequency of updates may be different from the frequency of measurements, e.g. in the case the sensor data are delivered in bursts rather than continuously. This information is specific to the measurement procedure. If not advertised, the frequency of updates is considered to be the same as the frequency of measurements.
Archive Time	optional	Procedure	Intended duration for which the observations remain available on the server. This information can be used by clients to decide whether a local copy of the data should be stored in a local database Archive time may be both, shorter or longer than TTL, and a single SOS may maintain different archive times for different types of observations, e.g. one week for environmental observations, and one year for the management-related observations. This information should therefore be advertised at the level of procedures (DescribeSensor/SensorML). If not advertised, the archive time is assumed to be "indefinitely"

5. DISCUSSION AND CONCLUSIONS

Sensor Web Enablement (SWE) Architecture developed by the Open Geospatial Consortium has the potential to become the first widely accepted Service Oriented Architecture for Environmental Information Systems (EISs). However, SWE is still in development and currently exhibits a number of issues which negatively affect the reliability and scalability of the SWE-based EISs. Relatively modest changes in Sensor Observation Service specification, such as those proposed above could greatly improve the reliability and usability of the SWE in the context of large distributed networks. Even greater improvements could be reached through a consequent use of the event-driven paradigm.

The solutions proposed in sections 3 and 4 are still under discussion within SANY consortium, and presented here in order to get a wider feedback before official presentation at the OGC Sensor Web Enablement working group. Final "best practice" recommendation of the SANY Consortium, may differ from the one proposed in this paper.

ACKNOWLEDGMENTS

SANY IP is an European Integrated Projects co-funded by the Information Society and Media DG of the European Commission within the RTD activities of the Thematic Priority Information Society Technologies. The authors wish to thank the SANY Consortium for their valuable input.

REFERENCES

Botts, M.. Percivall, G., Reed, C. and Davidson, J. (eds) (2007), OGC® Sensor Web Enablement: Overview And High Level Architecture, *OGC White Paper* OGC 07-165, 28. Dec. 2007, http://portal.opengeospatial.org/files/?artifact_id=25562

Cox, S. (Ed.) (2007), Observation & Measurements - Part 1: Observation Schema., OpenGIS® Specification 07-022r1.

EC (2007), *Directive 2007/2/EC of the European Parliament and of the Council* establishing an Infrastructure for Spatial Information in the European Community (INSPIRE).

EC (2008), European Commission, *Communication from the Commission to the Council, the European Parliament, The European Economic and Social Committee and the Committee of the Regions* - Towards a Shared Environmental Information System (SEIS), COM (2008) 46 final.

Erl, T., (2008), SOA: Principles of Service Design. ISBN 0-13-234482-3. Prentice Hall.

Havlik D., Bleier, T. and Schimak G. (2008), "Sharing sensor data with SensorSA and cascading Sensor Observation Service"; in Kooistra, L., and Ligtenberg, A., 2008 (Eds.); Proceedings of International Workshop Sensing a Changing World 2008. *CGI report* CGI-08-005, Wageningen University and Research Centre, Wageningen, The Netherlands. ISSN 1568-1874

Havlik, D., Bleier, T., Bartha, M., Ponweiser, T. and Schimak, G. (2009), Sensors and sensor-like data sources in Sensor Service Architecture, submitted to Elsevier Journal *Environmental Modelling & Software*

Havlik, D., Schimak, G., Denzer, R. and Stevenot, B. (2006), "Introduction to SANY (Sensors Anywhere) Integrated Project", in: Klaus Tochtermann, Arno Scharl (eds). *Proceedings of the 20-th international conference "Informatics for environmental protection"*, Graz, Austria, September 6-8 2006, ISBN: 3-8322-5321-1

Muehl, G., Fiege, L., Pietzuch, P.R. (2006). Distributed Event-Based Systems. Springer Verlag, Berlin. ISBN-10: 3540326510. Juli 2006.

Na, A.; Priest, M (Eds.) (2007), OpenGIS Sensor Observation Service (v.1.0.0), *OGC Implementation Standard* No. 06-009r6; https://portal.opengeospatial.org/files/?artifact_id=26667

Simonis, I. (Ed.) (2008), OGC® Sensor Web Enablement Architecture Version: 0.1.0, *OGC Best Practices Document* 06-021r2.

Usländer, T. (2008b), The Growing Importance of Open Service Platforms for the Design of Environmental Information Systems, in: *Proceedings of the International Congress on Environmental Modelling and Software* (iEMSs 2008), ISBN: 978-84-7653-074-0, Vol. 3, pp. 1628-1635.

Usländer, T. (Ed.) (2008), Specification of the Sensor Service Architecture v1, *SANY – Sensors Anywhere FP6 Integrated Project deliverable* D2.3.2, http://sany-ip.eu/publications/2420, 2008

Usländer, T., Havlik, D., Jacquesc, P., Schimak, G., Schlobinski, S., Simonis, I. and Watson, K. (2009), "Building Environmental Applications based upon an Open Sensor Service Architecture ", submitted to Elsevier Journal *Environmental Modelling & Software*

Usländer, Th. (Ed.) (2007), Reference Model for the ORCHESTRA Architecture v.2.1, *OGC "Best Practices" paper* No. 07-097, http://www.opengeospatial.org/standards/bp