An Architecture to Provide Quality of Service in OGC SWE Context

Thiago Caproni Tavares, Regina Helenna Carlucci Santana, Marcos José Santana, and Júlio Cezar Estrella

Institute of Mathematics and Computer Science, ICMC/USP, Avenida Trabalhador São Carlense, 400 - São Carlos, Brasil
{thiagocp,rcs,mjs,jcezar}@icmc.usp.br
http://www.icmc.usp.br

Abstract. The aim of this paper is to describe an architecture named SWARCH (Sensor Web Architecture) that provides quality of service in the context of Sensor Web Enablement (SWE) standards. Sensor Web Enablement is a set of standards proposed by OGC (OpenGis Consortium). These standards provide a transparent and interoperable way to access data measured by sensors. Thus, SWARCH adds to these features of the SWE standard ways of service selection that meet several quality requirements such as response time, availability of sensors, measurement reliability, among others. Quality requirements are defined by users and a broker in the architecture. This broker allows appropriate selection of the sensor network that matches to the QoS parameters. To validate our results, a case study showing reductions up to 50% and 25% in access times to SOS and SES services are presented.

Keywords: Web service, Sensor Networks, Performance Evaluation.

1 Introduction

A sensor network is composed of sensors that monitors one or a set of phenomena, and whose results are sent to an application or a final user [1,11]. A challenge in sensor networks utilization is in the feasibility of managing and provision the necessary information use in different applications. On the one side, we have the infrastructure composed by the sensors and the use of these sensors and strategies of the information obtained through them. On the other side, some applications or observers need to receive and process the information. Thus, sensor networks must have an infrastructure for communication, between sensors and between network and its observers. Middlewares that provides tools to manage these communications can be developing to facilitate the use of sensor networks [10].

An approach that has been proposed in the literature considers the sensor network as a Web Service [4]. Besides, middlewares using the concepts of service oriented architecture (SOA) have been widely discussed in the literature [6,5]. The Open Geospatial Consortium (OGC) has been working on the definition of standards and programming frameworks [7]. In this context, SWE (Sensor

S. Basu et al. (Eds.): ICSOC 2013, LNCS 8274, pp. 592–599, 2013.

Web Enablement) was proposed. It consists in a set of standards, protocols and interfaces that provides a framework to create sensor system following the principles of service-oriented architectures. Nevertheless, a gap in studies of sensor networks exposed as a service-oriented architecture was found. Mechanisms of Quality of Service are underexplored. Thus, this work presents architecture, named SWARCH, which allows the quality of service provisioning to the context of SWE standards.

This paper is organized as follows. Section 2 shows the SWE standards and a gap regard to quality of service. Section 3 presents the architecture to guarantee QoS in the SWE context. Section 4 discusses a case study to validate the SWARCH. Finally, Section 5 shows conclusion and future work which can be developed from the present work.

2 Background

The SWE defines sensor as devices discovered and accessed by means of a standardization of protocols and interfaces. It can be defined as an infrastructure that enables the integration of sensing resources. Applications or users can discover, access, modify and register sensing and alert services through a standardized way using sensor Web infrastructure. In [2], it is presented an overview of these patterns that are divided in two models. The information model comprises a set of standards that defines data models. These data models are used for the encoding of sensor observations as well as for its metadata. Two patterns are highlighted in information model: Observation & Measurements ((O&M)) and Sensor Model Language (SensorML). The SensorML determines a XML encoding to the description of sensors. It defines the location, input and output data, and the phenomena that is observed by the sensors. In turn, O&M standard defines a schema for the description of observations carried out by sensors. The interface models are used to provide an access mechanism to measured data from sensors through Web service interfaces. Thus, four main services are defined in SWE. The Sensor Observation Service (SOS) is a service that allows to insert and to retrieve sensing data. The Sensor Event Service (SES) is service that allows the registration of users and/or applications in an alert system. The Sensor Planning Service (SPS) is used to modify settings of the sensors into the sensor network. Finally, the Web Notification Service (WNS) is a service that provides an asynchronous notification mechanism between SWE services and clients or between other SWE services.

Additionally, it is being discussed by the community a standard of sensor discovery, which are referred to as SOR (Sensor Observation Registry) and SIR (Sensor Instance Registry). These registry services still have not become SWE standards. A gap found in the SWE specifications refers to QoS constraints. SWE standards do not treat this question in their specifications. The OGC itself assumes this gap considering it as a challenge to be achieved. “OGC standards provide an important framework for addressing semantics, but more work needs to be done to enable fusion of data from diverse sensor types. Data quality and

quality of service are important issues to be addressed in sensor web standards development activities" [8]. In [4], it is presented a survey of abstraction mechanisms of sensor networks. The authors conclude that QoS treatments are still little explored in this area.

A work considering QoS criteria in the context of the standards SWE is presented in [9]. The authors provide a search service that take into account nonfunctional and functional requirements in service selection. The search engine used by the authors does not perform a direct association with the services specifications provided by SWE such as SIR, SOS, and SES. The authors propose an abstract registry that is used to store the functional and nonfunctional requirements. The approach to provide quality of service presented in this paper differs in various aspects in relation to the proposal presented in [9]. Section 3 describes the architecture for QoS provision in the sensor network context and details each component of SWARCH.

3 SWARCH Description

The reference architecture of SWE standards follows a model where there are a client, a registry, and a server. In this model, the client searches a sensor system in the registry. So, the registry returns the sensors that can meet it, functionally. In turn, the client, after discovering service, performs some interaction with it. The aim of the proposed architecture in this work adds the features already implemented in the standard SWE architecture with the selection of services through QoS criteria. The SWARCH, presented in Figure 1, is composed by four components: Client, Broker, Registry, and Services. In short, clients send requests to the Broker that has the responsibility to find services with a specific QoS provisioning. A set of messages that are mostly defined by the SWE standards is used in SWARCH. The exception occurs only for the message 2 that is a composition of the SIR search message where is added an element of quality of service. Therefore, regarding other messages we can highlight the message 1 that represents a request to insert the sensor system description, and it must be held by the service provider. Additionally, message 3 corresponds to a search message that is submitted to the service registry. The Broker in SWARCH extracts message 3 from the message 2. Messages 4 and 5 are messages that carry measurements and alert notifications of sensors, respectively.

The Broker is divided into four major modules. The first interaction is made between Client and WSModule on Broker (message 2). The Client sends a SOAP message that contains two information. The first one sets the search message that will be used for the query in the SIR. In turn, the second information defines the QoS parameters that will be used in the selection of the service. The WSModule receives the SOAP message from the Client. So, WS-Module extracts the encapsulated information and forwards to SearchModule. After, the SearchModule uses the message to make the search query in the SIR, and returns the response message sent by the registry in an array. This array contains the sensor system descriptions (SensorML) of services found in SIR. In sequence, the SearchModule

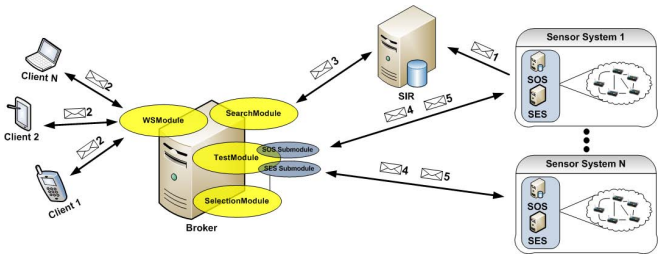

Fig. 1. Swarch Components

updates the array with descriptions of sensors that are cached in the TestModule. The TestModule, upon receiving this query, updates the information from sensors that were not yet inserted into the cache. However, the TestModule returns only the sensors that were present in the cache prior to accepting this new update, and this must be done due to the fact that there is no time to return QoS information of sensors that are not in the cache. The returned array cannot have sensor descriptions, i.e. the size of array is zero. Furthermore, array with size higher than zero means that there is quality of service information to at least one sensor in cache. In this way, the SearchModule prompts the SelectionModule to return the service that best meets the QoS parameters specified by the Client. In contrast, there is a random service selection, since no QoS information is registered in cache. However, the Broker will contain information of quality of service of these services in the next search. Finally, the SearchModule encapsulates service description document in a SOAP response message and sends it to the Client. The architecture presented in this Section is implemented in a prototype to validate the idea in inserting a QoS Broker in SWE context. The prototype has been implemented using the software components provided by 52º North Initiative. Section 4 presents a validation of this prototype.

4 Case Study

The use of a mechanism that supports service discovery based on quality of service criteria is justified for the following scenario. Several companies provide sensing data over the Web using the standards of the SWE. These companies offer a natural disaster sensing service that monitors water level concentration in a specific city. Thus, developers can implement several types of applications that may have different restrictions regarding quality of service. In this case, the architecture proposed in this work selects not only functional aspects of sensing system, but also nonfunctional aspects. The validation of proposal architecture in this work is developed by simulating a scenario where 12 companies (each one offering the same service and data types) provide sensing information of level of water concentration to a particular region. The Section 4.1 presents the evaluation scenario used to validate our approach.

4.1 Evaluation Scenario

The 12 companies were instantiated on 12 virtual machines that are instantiated on 3 real machines in a cluster of computers. The real machines have the following characteristics: Intel(R) Core(TM)2 Quad CPU Q9400 of 2.66GHz, memory of 8 GB RAM DDR 3, and disk size of 500 GB. In turn, the 12 virtual machines have different settings. The virtual machines with low capability are configured with 1 processor and 512 MB of memory. On other hand, medium machines are configured with 1 processor and 1GB of memory. Finally, high machines are configures with 2 processor and 2 GB of memory.

The SOS services configured on virtual machines contain a data base with the levels of concentration of water. The insertion of the observations in the data base mimics the behavior of sensor networks that sends an observation to the service every 5 minutes to SOS during one month. In this validation it is used an observation filter that restricts the observation period into 3 days. It is considered two factors in validation: Broker utilization and amount of threads (clients). The response time is the metric utilized in experiments. Two types of experiments were performed. The first experiment accesses SWE services (SOS and SES) without Broker intermediation, i.e. the search of the sensors is sent directly to the SIR. It is important to note that the client searches for observable properties that are registered in all virtual machines configured for the implementation of validation. In first experiment, the SIR service returns a list of 12 possibilities of services to the clients. Then, the threads select a service to submit requests randomly. Otherwise, the second experiment takes into account Broker utilization in the service discovery process. In this case, a QoS parameter is sent together with the search message, and unlike the first experiment, the return message from the Broker reports only one service. The Broker returns only the service that meets client QoS criteria. Twenty experiments were executed considering ten amounts of threads and versions with and without Broker. Each experiment is replicated 30 times to obtain a statistical validity. It is important to note that obtained response times in the experiments consider interaction between clients and providers, after the service selection. Section 4.2 discusses the obtained results.

4.2 Results

The results obtained for the completion of the validation presented in this section are represented in two types of charts. Response time charts present the variation of average response times in relation to increasing of workloads. The confidence intervals are calculated using an alpha of 0.05 (95% confidence interval). In turn, Pareto charts show the influences of each of the factors in the tests. In contrast to response time charts, the Pareto charts show an analysis considering a workload only to 10 and 100 threads. This analysis consists in calculating a linear regression model that considers two factors with two levels, such as: **Broker utilization** (with and without Broker) and **workload** (10 and 100 threads). The method of calculation is presented in [3].

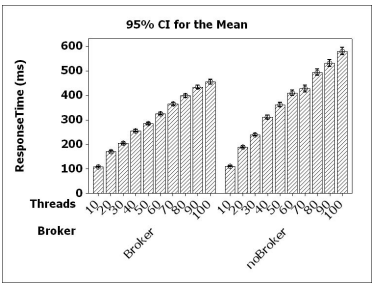

Fig. 2. SES: Response Times

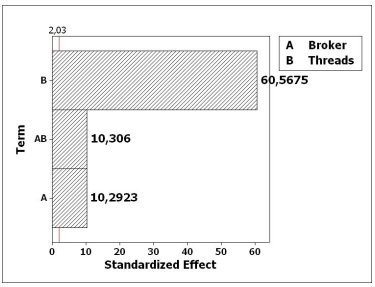

Fig. 3. SES: Factor Influence

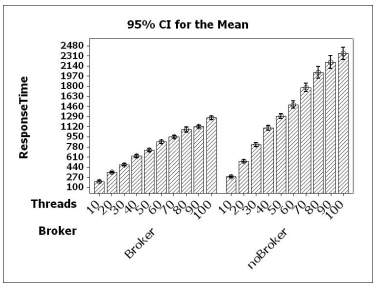

Fig. 4. SOS: Response Times

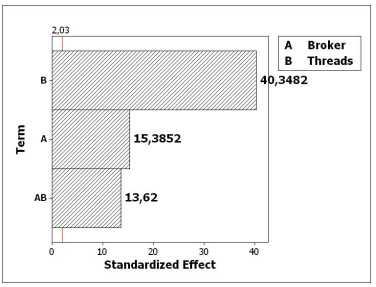

Fig. 5. SOS: Factor Influence

The chart in Figure 2 shows the results of experiments conducted on the SES service. The Broker intermediation improves response times for up to 21.6% on experiments of SES service. The Figure 3 shows a Pareto chart that defines the number of threads as the most influential factor in the tests, followed by the Broker factor. The use of the threads has significant influence in experiments since the values for the Broker in Pareto chart overtook vertical line. So, the experiments demonstrate that the Broker provides better performance when there is an increase on workload service. In the experiments conducted on the SOS service (Figure 4), it is also possible to observe that the improvement in response times, using the Broker intermediation, is higher regarding SES services. Furthermore, the Broker, used as an intermediary in the selection of the SOS service, decreases response times in approximately 46% compared to its non-use. Regarding to the influence, demonstrated in Figure 5, it is noticeable that amount of threads is the most influential factor. However, the Broker intermediation has a considerable influence. Thus, important information that must be highlighted for both SOS and SES services is related to the confidence interval obtained on Broker experiments. The standard deviation of the experiments without the Broker is higher when compared to the standard deviation of the experiment including the Broker. So, these results show that the use of the Broker makes the service access more stable.

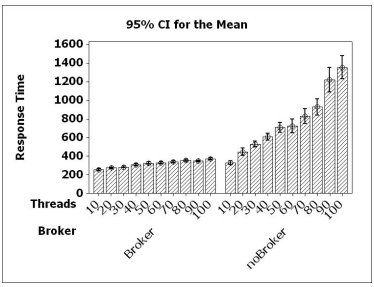

Fig. 6. Broker: Response Times

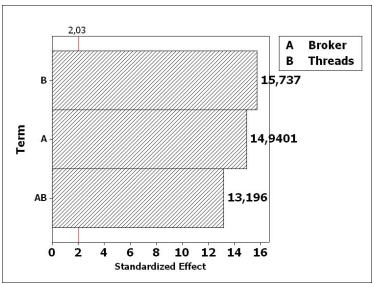

Fig. 7. Broker: Factor Influence

Finally, experiments to verify influence of the Broker in service select were conducted. These experiments consider the response time in service selection through Broker and through SIR directly. In this case, it is not considered the time access on SOS/SES services. It is considered only the direct searches on SIR and search on Broker. The Broker has another cache system that optimizes searches on SIR. This is made by means of a storing search messages in memory. Every sensor system has identification in SIR service. So, a search message can return one or more sensor IDs. The identifications are stored in an array that is associated with some search message. When a search message reaches to the Broker, and this message is stored in cache, the Broker changes this message for a search by ID. ID searches are faster than searches by other criteria. However, we considered in the Broker experiments a cache hit rate of 40%. The chart of Figure 6 demonstrates the response times for the selection service. It can be observed that response times in Broker experiments are lower and more stable. Additionally, Pareto chart presented in Figure 7 shows the broker with almost the same influence of the amount of threads. That is, the Broker impacts significantly on response times of service selection. Section 5 presents the conclusion and future work of our approach.

5 Conclusion and Future Work

This paper presents architecture to provide QoS support in SWE context, highlighting specifically the SOS and SES services. Therefore, the proposed architecture has, as a main element, a Broker that periodically monitors the QoS parameters on SWE services. The QoS provisioning was implemented by means of an insertion of a quality of service element in a SIR search message, and this QoS element is used to guide the Broker in a service selection that meets application with quality of service constraints. The validation of the architecture presented in this paper evaluates response times on service access using a Broker as intermediary component in SWE services selection. Furthermore, it also evaluates response times in service access through experiments planning that indicates statically the architecture efficiency. Additionally, the insertion of

the Broker in the process of service selection improves significantly the response times in access to the considered services.

Future work should add functionality to improve the specifications of quality of service parameters. Thus, the implementation of this feature may be used more formal mechanism for the determination of QoS parameters such as WSLA (Web Service Level Agreement). The use of this type of specification can assist in maintaining interoperability of SWARCH with the protocols and languages defined in SWE. In addition, it is intended to extend the tool for use in cloud computing. In this case, the Broker would have function to check the QoS parameters to manage the elasticity of the resources available in the processing of the SWE services configured in a cloud infrastructure.

Acknowledgements. The authors would like to thank the financial support of FAPESP (São Paulo Research Foundation), FAPEMIG (Minas Gerais Research Foundation), and IFSULDEMINAS/Campus Inconfidentes (Federal Institute of Education, Science and Technology of Southern of Minas Gerais).

References

1. Akyildiz, I., Vuran, M.C.: Wireless Sensor Networks. John Wiley & Sons, Inc., New York (2010)
2. Bröring, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T., Stasch, C., Liang, S., Lemmens, R.: New generation sensor web enablement. Sensors 11(3), 2652–2699 (2011), http://www.mdpi.com/1424-8220/11/3/2652
3. Jain, R.K.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley (April 1991)
4. Laukkarinen, T., Suhonen, J., Hännikäinen, M.: A survey of wireless sensor network abstraction for application development. IJDSN 2012 (2012), http://dblp.uni-trier.de/db/journals/ijdsn/ijdsn2012.html#LaukkarinenSH12
5. Mohamed, N., Al-Jaroodi, J.: Service-oriented middleware approaches for wireless sensor networks. In: 2011 44th Hawaii International Conference on System Sciences (HICSS), pp. 1–9 (2011)
6. Neto, F.C., Ribeiro, C.M.F.A.: Dynamic change of services in wireless sensor network middleware based on semantic technologies. In: International Conference on Autonomic and Autonomous Systems, pp. 58–63 (2010)
7. OGC: Ogc standards and specifications (December 2013), http://www.opengeospatial.org/standards (last access: May 06, 2013)
8. OGC: Why is the ogc involved in sensor webs? (2013), http://www.opengeospatial.org/domain/swe (last access: May 09, 2013)
9. Parhi, M., Acharya, B.M., Puthal, B.: Discovery of sensor web registry services for wsn with multi-layered soa framework. In: 2011 2nd International Conference on Computer and Communication Technology (ICCCT), pp. 524–530 (2011)
10. Wang, M., Cao, J., Li, J., Das, S.K.: Middleware for wireless sensor networks: A survey. J. Comput. Sci. Technol. 23(3), 305–326 (2008)
11. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. Comput. Netw. 52(12), 2292–2330 (2008)