Open Sensor Web Architecture: Stateful Web Services

Tom Kobialka [1], Rajkumar Buyya [2], Christopher Leckie [1]
[1] NICTA Victoria Laboratory and GRIDS Lab[2]
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{tkob, raj, caleckie}@csse.unimelb.edu.au

Abstract

As sensor networks become more pervasive there emerges a need for interfacing applications to perform common operations and transformations on sensor data. Web Services provide an interoperable and platform independent solution to these needs. A key challenge of using Web Services in this context is how to support ongoing sensor queries that persist over an extended period of time. In this paper we introduce Web Service Resource Framework (WSRF) mechanisms into the core services implementation of the NICTA Open Sensor Web Architecture (NOSA). NOSA is a suite of middleware services for sensor network applications which are built upon the OpenGIS Consortiums Sensor Web Enablement standard. WSRF expands the functionality of our services to handle simultaneous observational queries to heterogeneous Sensor Networks. It facilitates the adoption of a multi-user, multi-threaded service environment. Using components from the Globus Middleware platform, NOSA takes a major step forward to achieving the vision of a Sensor Grid.

1. INTRODUCTION

The identification of common data operations and transformations on sensor data has fuelled the birth of the Sensor Web paradigm. Sensor Web combines sensors and sensor networks with a Service Orientated Architecture (SOA). The SOA allows us to discover, describe and invoke services from a heterogeneous software platform using XML and SOAP standards. When interlinked, geographically distributed services form what is called a Sensor Grid; this is a key step in the integration of sensor networks and the distributed computing platforms of SOA and Grid Computing. Services are defined for common operations including data query, retrieval and aggregation, resource scheduling, allocation and discovery. Sensor networks can be discovered, accessed and controlled over the World Wide Web.

The NICTA Open Sensor Web Architecture [7] is built upon a uniform set of operations and standard sensor data representations as defined by the Open Geospatial Consortium [1] (OGC). The OGC is a geospatial standards authority that has defined a Sensor Web Enablement (SWE) method [2] which encompasses specifications for interfaces, protocols and encodings that enable discovering, accessing, obtaining sensor data as well as sensor-processing services. The following are the five core specifications of the SWE which are implemented in the NOSA core services:

1. Sensor Model Language (SensorML) [3] – Information model and XML encodings that describe either a single sensor or sensor platform in regards to discovery, query and control of sensors.
2. Observation and Measurement (O&M) [4] – Information model and XML encodings for observations and measurement.
3. Sensor Collection Service (SCS) [16] – Service to fetch observations, which conform to the O&M information model, from a single sensor or a collection of sensors. It is also used to describe the sensors and sensor platforms by utilizing SensorML.
4. Sensor Planning Service (SPS) [5] – Service to help users build a feasible sensor collection plan and to schedule requests for sensors and sensor platforms.
5. Web Notification Service (WNS) [6] – Service to manage client sessions and notify the client about the outcome of the requested service using various communication protocols.

Although the core services worked well in a Research and Development environment their capabilities fell short of expectations as the user load and complexity of observational queries was increased, and additional sensor networks where added to the SCS. These services are based on earlier development work on the NOSA core services [7].

A key challenge of NOSA is how to support ongoing sensor queries which persist over time to heterogeneous sensor networks. This challenge is addressed by the following improvements to the NOSA architecture; (i) all services are implemented as stateful Web Services (WSRF), (ii) the SCS works with many different types of sensors extending TinyOS running on Mica2 and TinyDB to include an in-house sensor running Linux developed by NICTA called NICTOR [22], (iii) services are capable of handling concurrent requests from multiple users, (iv) a repository service has been added to store historic observation results.

In Section 2 we provide background information on core services along with an analysis of their shortcomings. Section

3 introduces stateful Web Services and describes the improved architecture and design of NOSA. Section 4 provides a performance evaluation of the SCS and presents recommendations for future development work.

2. Background: Core Services

The aim of NOSA is to provide a software infrastructure that simplifies the task of application development on heterogeneous wireless sensor networks. The overall structure of NOSA is outlined in Fig. 1. Four layers have been defined, namely Fabric, Services, Development and Application. Fundamental services are provided by low-level components whereas higher-level components provide tools for creating applications and management of the lifecycle of data captured through sensor networks. NOSA provides the following sensor services:

1. Sensor notification, collection and observation;
2. Data collection, aggregation and archival;
3. Sensor co-ordination and data processing;
4. Faulty sensor data correction and management, and;
5. Sensor configuration and directory service

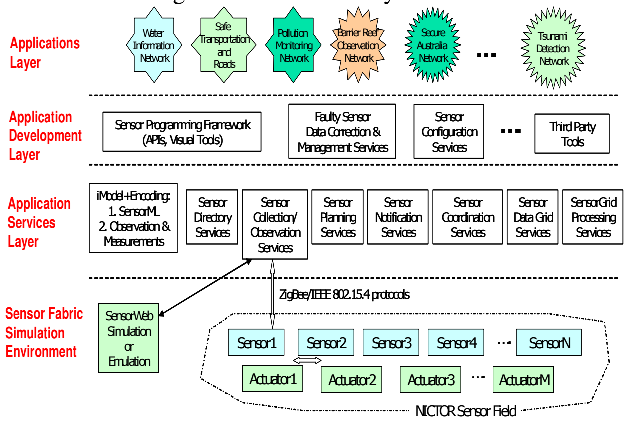
Applications Layer
Water Information Network
Safe Transportation and Roads
Pollution Monitoring Network
Barrier Reef Observation Network
Secure Australia Network
...
Tsunami Detection Network
Application Development Layer
Sensor Programming Framework (APIs, Visual Tools)
Faulty Sensor Data Correction & Management Services
Sensor Configuration Services
...
Third Party Tools
Application Services Layer
iModel+Encoding: 1. SensorML 2. Observation & Measurements
Sensor Directory Services
Sensor Collection/ Observation Services
Sensor Planning Services
Sensor Notification Services
Sensor Coordination Services
Sensor Data Grid Services
SensorGrid Processing Services
ZigBee/IEEE 802.15.4 protocols
Sensor Fabric Simulation Environment
SensorWeb Simulation or Emulation
Sensor1
Sensor2
Sensor3
Sensor4
...
SensorN
Actuator1
Actuator2
Actuator3
...
ActuatorM
NICTOR Sensor Field

Fig. 1: High-level view of NICTA Open Sensor Web Architecture.

Fig. 2 depicts a prototype instance of NOSA, The implementation concentrates on the Service Layer and Sensor Layer as well as the XML encoding and the communication between the sensors and external networks.

The primary focus of the design and implementation of NOSA has been on the SWE core services including SCS, WNS, and SPS (which extend from the SWE) as well as the Sensor Repository Service (SRS) which provides a persistent data storage mechanism for sensor and observation data. O&M XML specifications have been implemented and are used to encode observational data recorded and retrieved from the sensor network by the SCS. Fig. 3 illustrates an example of a collection request and the invocations between related services. Once a client knows the location of an SPS instance it will send an XML observational plan containing specifics of the observation request. These may include the sensors it is interested in obtaining observational data from, the type of data, the duration of the query and any other relevant metadata.

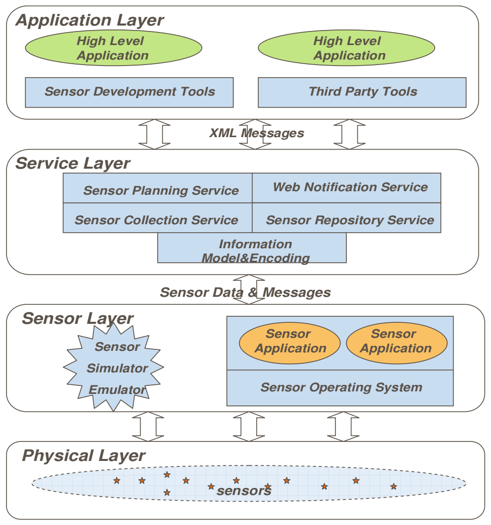

Fig. 2: A prototype instance of NOSA

The SPS will assess the feasibility of the client's requirements and register the user details with the WNS. If the SPS can fulfil the requirements it will forward the observation onto the SCS which will retrieve the observation data from the appropriate sensor network. The SPS will notify the WNS of the completed plan and the WNS will forward this onto the client. Meanwhile the SCS may store the observational data in the repository.

The following subsections describe in detail the core set of NOSA services, SCS, SPS and WNS, and their limitations.

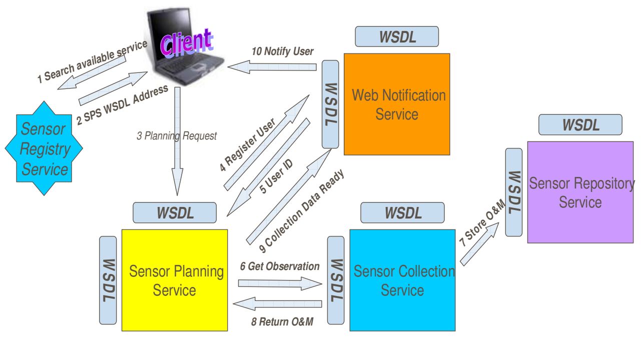

Fig. 3: A typical invocation for Sensor Web client.

A. Sensor Collection Service

The SCS is responsible for communicating directly with the sensor networks. It receives incoming SOAP requests invoking the getObservation call. It interfaces with the sensor network via a proxy, collecting the resulting query information and translating the raw observational data into a XML O&M encoding. It then returns the encoded observation data to the connecting client. The SCS provides a proxy interface to both streaming data and query based sensor applications that are built on top of TinyOS [8] and TinyDB [9]. The SCS is implemented as a Web Service, deployed on the Apache Tomcat [10] servlet container.

The design of the SCS is currently limited to executing one query request at a time from a single connecting client application. A major shortcoming of using Web Services is that they are nominally stateless, i.e. they retain no data between invocations. This limits their usefulness, although workarounds exist — such as having the Web Service read from a database, or using session state by way of cookies or WS-Session [11]. For a typical Web Service, an incoming getObservation query from a client or SPS, such as retrieving the observations from a sensor network for any extended period of time with a regular periodic update frequency, is difficult to implement. Using cookies to manage session state essentially ties down the connecting client to being a web browser and limits the connectivity of the Web Service to console applications and services. A Web Service is incapable of sending automatic updates to a connecting client. It is the responsibility of the client to follow a request-response communication pattern. As the duration of an observational query increases and sampling period at which results are produced decreases, the resulting communication traffic grows in proportion to the ratio of the duration and the sampling period making this solution unacceptable. For example a SPS plan with a duration of 10 minutes and a sampling period of 10 seconds would require a request-response communication pattern to occur between the SPS and SCS services approximately 60 times, which would mean a total of 120 messages sent across the network.

B. *Sensor Planning Service*

The SPS is a scheduling service which interfaces between a client or service and the SCS. A connecting client sends a SOAP message to the SPS containing an XML based plan listing observational requirements. The plan can include the getObservation query request itself e.g. Light intensity values that are above a given threshold from all sensors, along with a duration and update period, e.g. for a duration of 10 minutes with an update period of 10 seconds. The SPS will check the feasibility of the plan utilizing a rule engine. Predefined rules are used to check the boundary conditions of observation query values, whether a SCS service exists and is capable of fulfilling the request e.g. the SCS records light values. The SPS provides an immediate response to the client on the feasibility of the plan. A scheduler is implemented in the SPS as a separate thread. It composes the user's plan into a collection request, invokes the getObservation call on the SCS, stores the resulting data in a DataCollector database and sends a notification to the WNS indicating the outcome of the collection request. The client is free to retrieve the observational data from the DataCollector after it has been notified by the WNS. The SPS falls short in its inability to manage the scheduling of more than one user plan at a time. The level of execution plan sophistication is limited to very simple requests which only require the SPS to manage duration time of one query request.

C. *Web Notification Service*

The WNS service performs the basic functionality of acting as a communication relay between services and clients. Clients register with the WNS through the SPS. Once the SCS returns a getObservation result to the SPS, the SPS notifies the WNS of the operation. The WNS will then send a message to the registered user informing them of the operation. The architecture is flexible enough to include a variety of communication plug-ins, however only the email protocol has been implemented.

As a Web Services platform, the NOSA core services work well in a single user, single sensor network environment where resource limits are low and easily anticipated. However, such an environment is very little use outside a research laboratory. To overcome these limitations, we have extended our Sensor Web implementation to include stateful Sensor Web services.

3. STATEFUL WEB SERVICES: ARCHITECTURE AND DESIGN

The introduction of stateful Web Services into the NOSA architecture aims to provide a cohesive solution to the shortcomings experienced in development of the core services. Stateful Web Services provide the ability of users to access and manipulate state, i.e., data values that persist across and evolve as a result of Web Service interactions. The Web Service Resource Framework (WSRF) [12] defines conventions for managing state so that applications discover, inspect and interact with stateful resources in standard and interoperable ways [13] as defined by the OASIS standards body. Stateful Web Services provide all the benefits of Web Services including standardization and interoperability, with the advantages of state.

At the time of development there existed two major implementations of the WSRF protocol, these implementations where the most mature in their completeness of the specifications.

Apache WSRF [14] is an Apache Axis-based web application. It uses an XML schema to compile Java interfaces and classes that can then be used to access and modify XML instance data [15]. The second implementation is the Java WS Core which is a component of the Globus Toolkit [16]. The Globus Toolkit is a set of software components for building distributed systems and it is a popular Grid middleware platform. The Java WS Core comes as a set of JAR API files which are called by both the client and server and deployed in conjunction with the service in a Tomcat container. The Globus Java WS Core was chosen due to the reduced amount of work necessary to convert existing services to WSRF in comparison to the Apache version of the specification. Under Globus, minor modifications were made to existing WSDL and Java files, whereas, the Apache method of generating Java from WSDL was error prone when OpenGIS schemas were introduced into the WSDL. The Globus toolkit provides software components (Security, Data

Management and Information discovery) which in future development works may prove useful.

In the following sections we detail the advantages of WSRF to the NOSA core services and discuss the architectural modifications made necessary to implement these.

A. Sensor Collection Service

The SCS is the largest and most important service in the NOSA architecture. When implemented as a Web Service the network traffic involved in executing observational queries on a sensor network which require the SCS to provide regular, near real-time updates to a connecting client or service becomes unacceptable. WSRF offers a solution to this problem with the introduction of the WS-BaseNotification [17] specification which defines a notification subscription interaction pattern. Using the Globus WS-Notification and WS-Resource APIs the SCS defines a resource. A resource is an entity that encapsulates the state of a stateful Web Service [18]. When the SCS receives an incoming SOAP request it creates a resource object and generates a unique resource key. This resource key is returned to the client and all further communication exchanges between the client and service include the resource key. When the client subscribes to the service resource, the subscription ensures that any changes of state which occur on the resource e.g. new observational results received from a sensor network, will automatically be forwarded to the client . This process is illustrated in Fig 4.

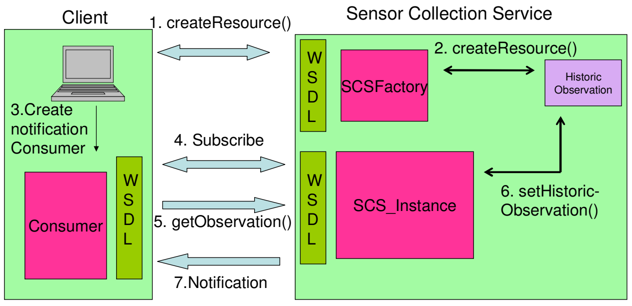

Fig 4: Subscription to WSRF service

The benefits of notification subscription become apparent when the number of simultaneous connecting clients is increased because the network traffic overhead is reduced by close to half. Fig 5 compares the network traffic between the response-request and notification subscription patterns.

Notification subscription permits connections to exist for a predefined duration or until the client terminates the connection. This responsibility of the SCS managing the duration of its own connections facilitates the offload of some responsibilities handled by the SPS to the SCS. Clients can now choose to subscribe to the SCS for a given duration and update period. Previously this level of scheduling would have been handled by a scheduling service namely the SPS.

The introduction of WSRF has additional advantages in simplifying the move of the SCS towards a multi-user system. A resource factory handles the assignment of unique identifiers for each connection. Multithreading is handled automatically by the Tomcat container. For each new connection Tomcat will create a new instance of the SCS and manage its memory. Concurrency is implemented within the SCS to ensure that shared data does not interfere among SCS instances and shared memory remains consistent. The unique identifiers assigned to each connection by the WSRF libraries facilitate multithreading and help ensure concurrency.

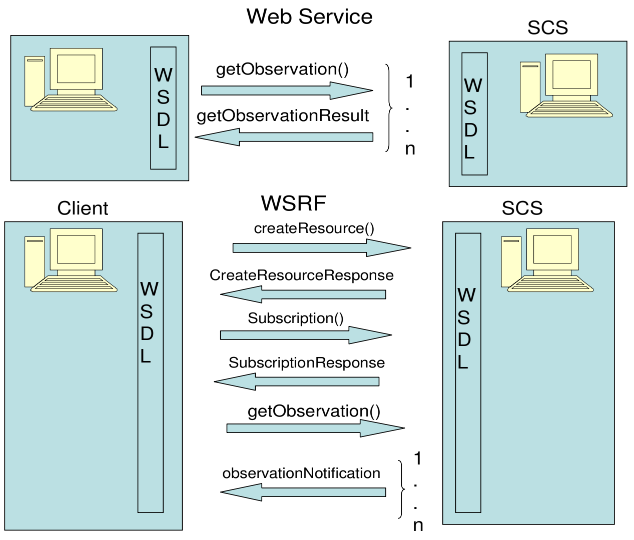

Fig 5: Network communication traffic, of a typical Web Service vs. a Web Service implemented with WSRF, between a client (typically SPS) and the SCS.

The move towards heterogeneous sensor network capabilities is another key development for the SCS. The type of sensor networks the SCS is capable of interacting with has been expanded from TinyDB and TinyOS running on Crossbow Mica2 motes to include a Linux based operating system running on NICTA developed sensors called NICTORs. NICTORs interface with the SCS through a MySQL database. Additional development work has been done to completely implement all interfaces as defined by the SCS specification [19], making the SCS an OpenGIS standards compliant implementation. This includes an implementation of SensorML to satisfy the minimum requirements.

B. Sensor Planning Service

The introduction of WSRF to the SPS expands its capabilities to include the processing of more than one connecting client at a time. Synchronization constructs have been added to ensure concurrency and prevent shared memory corruption. Multi threading is handled by Tomcat in much the same way as for the SCS. The improvements made to the architecture of the SPS are very similar to those made to the SCS. The SPS implements a notification subscription interaction pattern. Once clients subscribe to the SPS they can be notified periodically when data is ready to collect. This functionality can also be provided by the WNS for more complex execution plans. Because some of the SPS responsibilities have been offloaded to the SCS, the SPS has the potential to execute plans which it previously lacked the functionality to perform. Possible plans may include a finer level of scheduling, e.g. Query the SCS every 100ms for 5 minutes, every hour for a period of 3 months. An operator may be deployed on the SPS capable of performing transformations on observational data, which may include operations such as distributed anomaly detection [20]. Users may be interested in the retrieval of observational data from historic queries, which can be facilitated by the SPS with calls to the SRS. Plans could also use historic observational data to reduce duplicate observation results. The implementations of these higher-level scheduling options remain for future work. Modifications to the SPS have also included improvements to concurrency making it possible for the SPS to handle the execution of multiple user plans simultaneously.

C. Sensor Repository Service

In previous versions of the SPS a DataCollector interface was implemented whose task was to store observational data collected from the SCS. This interface has since been decoupled from the SPS and implemented as the SRS. The SRS is as a WSRF service which implements one routine saveObservation() that accepts O&M XML encoded data.

The SRS maintains a store of historical data which can be used by the SPS to further improve the quality of service provided to connecting services. Historical data stored in the SRS could be used by advanced SPS execution plans which require aggregate or statistical information on previously acquired observation results. No OpenGIS specification exists to this date for a SRS service. It is our intention to feedback our experiences in the near future. Additional work in defining the query interface and parameters necessary to facilitate these requests remains to be done.

D. Web Notification Service

The WNS has been implemented as a WSRF service in a similar fashion to the SCS and SPS. Besides this, however, no major modifications have been made to the baseline WNS code.

4. PERFORMANCE EVALUATION

The evaluation of service capabilities has been limited to the SCS. This service has had the most improvements made to it and its performance is the cornerstone of good performance for all the other services. Two experiments were carried out: in the first the focus was on the Crossbow sensors, in the second on the NICTOR sensors. Both experiments test the robustness of the SCS in a multi-user, multi-threaded heterogeneous sensor network environment. Fig 6 illustrates the experimental setup.

The experimental platform was developed using Crossbow's MOTE-KIT4x0 MICA2 Basic Kit [21] which consists of 3 Mica2 Radio boards, 2 MTS300 SensorBoards, a MIB510 programming and serial interface board.

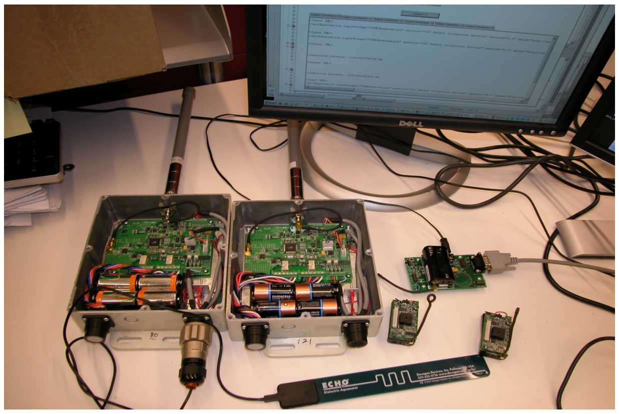

Fig 6: NICTOR sensors with their case removed and Crossbow mica2 motes connected to a workstation with client simulation software running in the background.

Two Mica2 motes were setup for sensing observational data. Light values were recorded at a sampling period of 100ms for a duration of 10 seconds. Once values where recorded the getObservation query would terminate. The numbers of simultaneous connections tested were {1, 2, 4, 8, 16, 32}. The relative response time is illustrated in Fig 7. The results show that as the number of simultaneous clients steadily increases so does the response time. The minimum and maximum response time values for each group of simultaneous connections tend to vary greatly from the average. As the number of clients is increased this difference grows larger, in the case of Crossbow sensors our SCS does not scale well. Degradation in the performance is an outcome of the limited concurrency available at the sensor network level of the service. Only one query can be processed by the sensor network at a time and so consecutive queries must be placed in a queue until the current query returns its results. A possible solution to this problem could be the introduction of a query cache. In this approach, if a new query is received that is similar to one which has recently been executed, then the result can be anticipated as being the same and resulting observational data could then be pulled from the cache. This would reduce the number of duplicate queries sent to the sensor network and improve the scalability.

The second experiment was performed using 2 NICTOR sensors. One was programmed to act as the base station; the other was responsible for recording observations. A soil moisture sensor was connected to the sensing node. Observations where recorded for a duration of 2 seconds at a sampling period of 20ms. Once observations were retrieved the connection was terminated. Fig 8 illustrates the results. As the number of simultaneous clients increases so does the overall average response time of each connection. The minimum and maximum response times fall closer to the average when compared to results produced by the Crossbow sensors. The primary reason behind this is that the NICTOR sensors are interfaced through a MySQL database, which does not require the results of the previous query to be returned before a new query is sent. Hence we see an improvement in the scalability of the SCS. This is in contrast with the Crossbow sensors which must wait till the results are returned for the current query before a new query can be sent. Implementing a caching mechanism could further improve the scalability of the SCS when dealing with NICTOR sensors. The existing MySQL interface would serve as a base upon which caching functionality could be built upon.

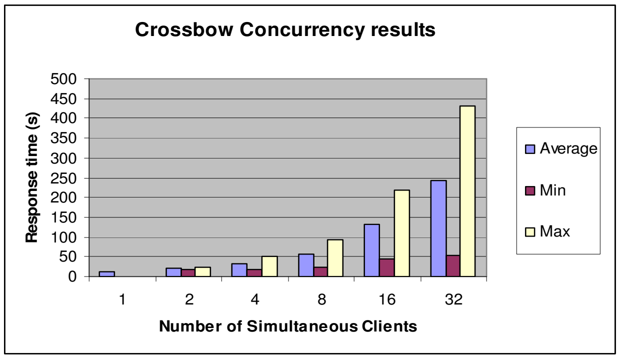

Fig 7: Duration of time recorded for simultaneous clients sending a getObservation() request to the SCS Mica2 motes.

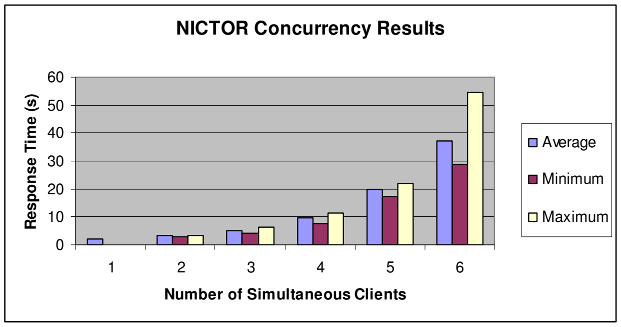

Fig 8: Duration of time recorded for simultaneous client connections performing a getObservation() request on the NICTOR sensors.

5. Conclusion

The move from Web Services to WSRF addresses the limitations of the NOSA core services. WSRF facilitates the ability of the SCS to handle simultaneous observational queries to heterogeneous sensor networks. The functionality of the SCS is extended to include tasks previously assigned to the SPS. This allows the SPS to concentrate on executing plans of increased detail and complexity. WSRF improves the performance of services by reducing the amount of network traffic, using the notification subscription communication model. Additional services have been added in the form of the SRS, which separates data storage tasks from the SPS. The adoption of Globus Middleware for the WSRF implementation is a significant step forward towards achieving a Sensor Grid. Potential for future work exists in the implementation of operators for the SPS and improvement of scheduling capabilities, and the implementation of a caching mechanism to improve performance of the SCS.

Acknowledgement

We would like to thank Ronald Wong for his contribution to the development of a test client. We thank all members of the NOSA project especially those involved in advancing Sensor Web into the future.

References

[1] http://www.opengeospatial.org/
[2] Botts M, Percivall G, Reed C, Davidson J, (2006) OGC® Sensor Web Enablement: Overview And High Level Architecture, OpenGIS Consortium Inc
[3] http://vast.nsstc.uah.edu/SensorML/
[4] Cox S, (2006) Observations and Measurements OGC 05-087r3, Open Geospatial Consortium Inc, http://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/index.php?artifact_id=14034
[5] Simonis I, (2005) Sensor Planning Service OGC 05-089r1, Open GIS Consortium Inc
[6] Simonis I, Wytzisk A, (2003) Web Notification Service OGC 03-008r2, Open GIS Consortium Inc
[7] Chu X, Kobialka T, Durnota B, and Buyya R. "Open Sensor Web Architecture: Core Services". In Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing (ICISIP 2006, IEEE Press, Piscataway, New Jersey, USA, ISBN 1-4244-0611-0) pp. 98-103, Dec. 15-18, 2006, Bangalore, India.
[8] http://www.tinyos.net
[9] http://telegraph.cs.berkeley.edu/tinydb
[10] http://tomcat.apache.org/
[11] http://en.wikipedia.org/wiki/Web_service
[12] Banks T, (2006) Web Services Resource Framework (WSRF) – Primer v1.2, OASIS, http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf
[13] http://www.globus.org/wsrf/
[14] http://ws.apache.org/wsrf/
[15] http://xmlbeans.apache.org/overview.html
[16] http://www.globus.org/
[17] Graham S, Murry B, (2004) Web Serivuces Base Notification 1.2 (WS-BaseNotification), OASIS,http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf
[18] Gawor, J and Meder S, (2004) GT4 WS Java Core Design, Globus Alliance, http://www.globus.org/toolkit/docs/4.0/common/javawscore/developer/JavaWSCoreDesign.doc
[19] McCarty T, (2003) Sensor Collection Service OGC 03-023r1, Open GIS Consortium Inc.
[20] Rajasegarar S, Leckie C, Palaniswami M and Bezdek J. Distributed Anomaly Detection in Wireless Sensor Networks. To appear in Proceedings of Tenth IEEE International Conference on Communications Systems (IEEE ICCS 2006), 30 October-1 November 2006, Singapore.
[21] http://www.xbow.com/Products/productsdetails.aspx
[22]http://www.nicta.com.au/research/projects/water_information_networks