

# A Sensor Web Middleware with Stateful Services for Heterogeneous Sensor Networks

Tomasz Kobialka<sup>1</sup>, Rajkumar Buyya<sup>2</sup>, Christopher Leckie<sup>1</sup>, Ramamohanarao Kotagiri

<sup>1</sup>NICTA Victoria Laboratory, GRIDS Lab<sup>2</sup>

Department of Computer Science and Software Engineering

The University of Melbourne, Australia

{tkob, raj, caleckie, rao}@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 Consortium's 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 Oriented Architecture (SOA). A 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 includes specifications of interfaces,

protocols and encodings that enable discovering, accessing, and 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) [15] – 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 work well in a Research and Development environment their capabilities fall short of expectations when the user load and complexity of observational queries is increased. These services are based on earlier development work on the NOSA core services [7].

Several groups are working on related work in the Sensor Web field. Microsoft has recently released the MSR SenseWeb Project [22] which allows users to publish their sensor data on a portal web site. Microsoft has implemented its own XML ontology along with a set of querying and tasking mechanisms. Support is provided for sensors running TinyOS and devices such as webcams. Microsoft is not affiliated with the OGC Consortium and there is no support for Linux based operating systems. The current application of SenseWeb is limited to publishing data, with little support for post processing, although it is likely that this will change as the project matures. The GeoICT group at York University [23] have built an OGC SWE compliant Sensor Web infrastructure. They have developed a Sensor Web client capable of visualising geospatial data, and a set of stateless Web Services called GeoSWIFT [24]. The GeoSWIFT

Sensing Server implements all the interfaces of a typical SCS and is capable of communicating with Webcams. There is no implementation of a WNS or SPS, although some work has been done on a Registry Service. The Sensorweb Research Laboratory at Washington State University is focusing its efforts on building a Volcano Monitoring Sensor Network and is working on a NASA funded in-situ Space Sensor Network. Their research is largely on low-level network topology and routing running on top of TinyOS. It is unclear as to the level of sophistication of their Services layer.

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, from TinyOS running on Mica2 and TinyDB to an in-house sensor running Linux called NICTOR [21] developed by NICTA, (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 for heterogeneous wireless sensor networks. The overall structure of NOSA is outlined in Fig. 1. Services interface with low-level components and provide tools for creating applications, along with management of the lifecycle of data captured through sensor networks.

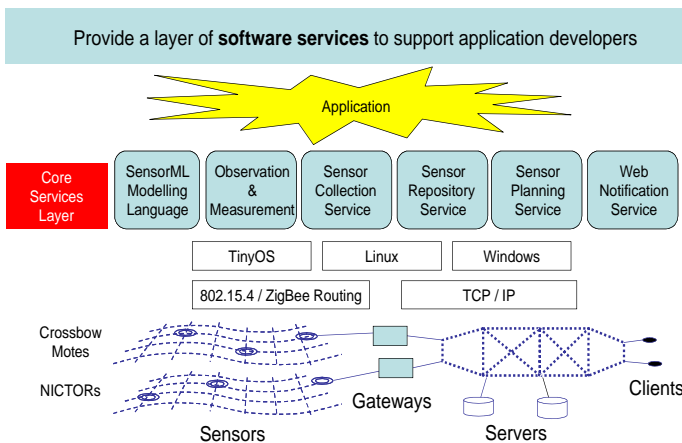


Fig. 1: High-level view of NICTA Open Sensor Web Architecture.

The primary focus of the design and implementation of NOSA has been on the SWE core services including the 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 a sensor network. Fig. 2 illustrates an example of a typical collection request on a sensor network and the resulting invocations on NOSA services.

A client sends an XML observation request to a SPS instance. The request includes the sensors the client is interested in obtaining data from, the type of data, the duration of the query and any other relevant metadata. The SPS assesses the feasibility of the client's requirements and registers the user details with the WNS. If the SPS can fulfil the requirements it forwards the observation onto the SCS which retrieves the observation data from the appropriate sensor network. Upon completion the SPS will notify the WNS and the WNS will forward this onto the client. Observational data may be stored in a repository which can then be retrieved at the user's convenience.

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

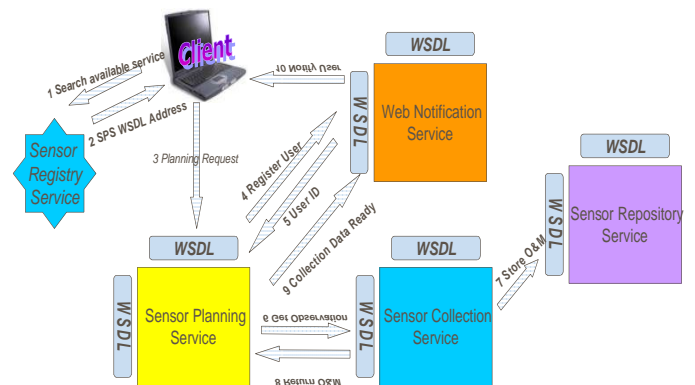


Fig. 2: A typical invocation for Sensor Web client.

### A. Sensor Collection Service

The SCS is responsible for communicating directly with the sensor networks. 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]. It collects observational data produced by the sensor networks and encodes it into XML using the O&M specification. The SCS is deployed on an 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. A major shortcoming of using Web Services is that they are nominally stateless, i.e. they retain no data between invocations, and this limits their usefulness. For a typical Web Service an incoming query, such as retrieving the observations from a sensor network for any extended period of time and containing a regular periodic update frequency, is difficult to implement. Cookies can be used to manage session state but this 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 the sampling period at which results are produced decreases, the resulting communication traffic grows in proportion to the ratio of the query duration and the sampling period, making this solution unacceptable. A second major shortcoming of the SCS is that it is limited to working with sensors which are running TinyOS. This restricts the range of hardware vendors and limits the usefulness of NOSA.

#### *B. Sensor Planning Service*

The SPS is a scheduling service which interfaces between a client or service and the SCS. It accepts XML based plans which contain a listing of observational requirements along with a duration value and update period. 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 if it is capable of fulfilling the request. A scheduler composes the user's plan into a collection request and invokes an observation call on the SCS. The resulting observation data is archived in a database and a notification is sent to the WNS indicating the outcome of the collection request. The client is free to retrieve the observational data once it is notified by the WNS. A shortcoming of the SPS is its inability to manage the scheduling of more than one user plan at a time. The level of user plan sophistication is limited to very simple requests, which further restricts the usefulness of the service.

#### *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. When a SCS returns an observation result to the SPS, the WNS is notified of the operation. The WNS, in turn, notifies the registered user of the operation. The architecture is flexible enough to include a variety of communication plug-ins, however only the email protocol has been implemented. Similarly to the SPS and the SCS the WNS can only process one user request at any one time.

As a Web Services platform, the NOSA core services work well in a single user, single sensor network environment, where all sensors run the same Operating System and resource limits are low and can be easily anticipated. 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 SENSORWEB SERVICES 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 access to 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 as defined by the OASIS standards body. Stateful Web Services provide all the benefits of Web Services including standardization and interoperability. At the time of development there exist two major implementations of the WSRF protocol, these implementations were the most mature in their completeness of the specifications.

Apache WSRF [13] 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 [14].

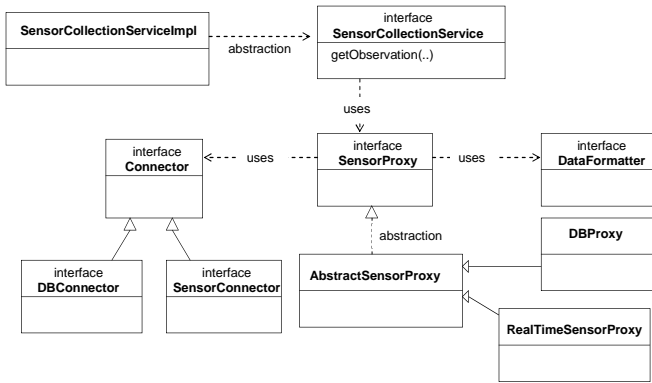
Java WS Core is a component of the Globus Toolkit [15]. The Globus Toolkit is a set of software components for building distributed systems and it is a popular Grid middleware platform. The Java WS Core comes as a set of JAR API files, which are deployed in conjunction with the service in a Tomcat container. The Globus Java WS Core was chosen over the Apache implementation because it required the least amount of rework to be done to existing services. 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 also provides additional components 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 necessary to implement these.

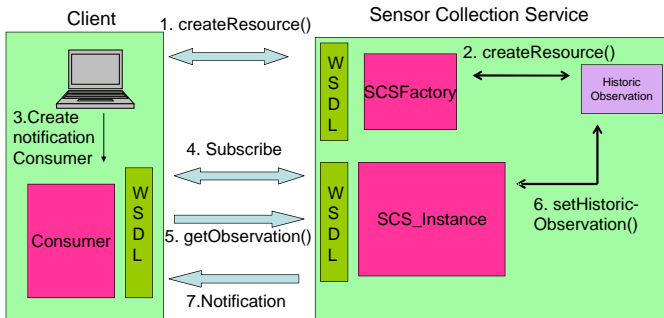
#### *A. Sensor Collection Service*

The SCS is the largest and most important service in the NOSA architecture. Its core classes are illustrated in Fig 3. When implemented as a Web Service, the network traffic involved in executing observational queries becomes unacceptable on a sensor network that requires the SCS to provide regular, near real-time updates to a connecting client or service. WSRF offers a solution to this problem with the introduction of the WS-BaseNotification [16] 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 [17].

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 3: Core classes which make up the SCS and their relationships.**



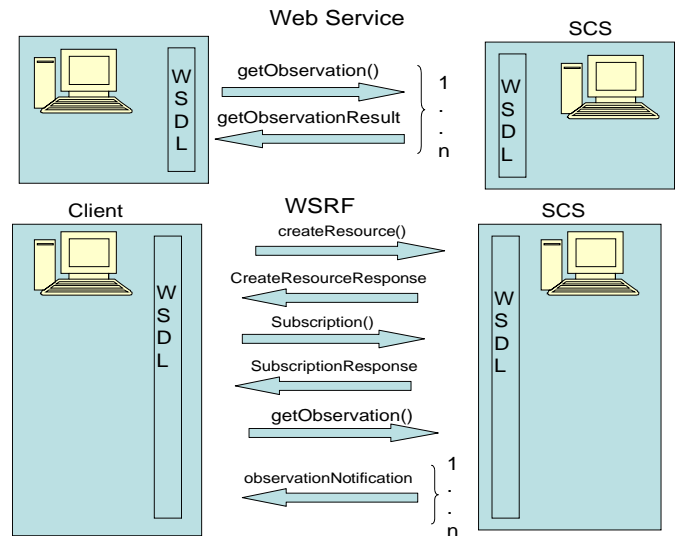
**Fig 4: Subscription to WSRF service**

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 of a typical Web Service and notification subscription pattern of the newly implemented WSRF.

Notification subscription permits connections to exist for a predefined duration or until the client terminates the connection. The responsibility of the SCS in 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 directly subscribe to the SCS for a given duration and update period. Previously this level of scheduling would have been handled by a scheduling service, such as the SPS.

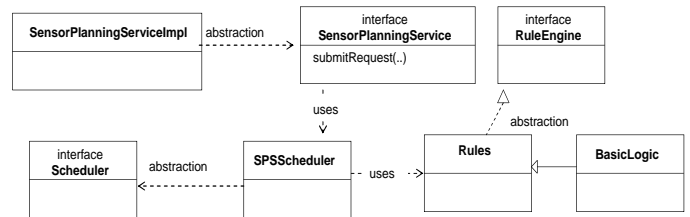
The introduction of WSRF has additional advantages in that it facilitates the transition of the SCS from a single-user towards a multi-user system. A resource factory handles the assignment of unique identifiers for each connection. Multithreading is handled automatically by the Tomcat container. For each new connection Tomcat creates a new instance of the SCS and manages its memory. Concurrency is implemented within the SCS to ensure that shared data does not interfere between SCS instances. The unique identifiers assigned to each connection by the WSRF libraries facilitate multithreading and help ensure concurrency. The move towards operating in a heterogeneous sensor network environment is another key development of the SCS. The SCS has been expanded from being capable of only connecting to sensors running TinyDB and TinyOS on

Crossbow Mica2 motes, to include a Linux based operating system running on NICTA developed sensors called NICTORS.



**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.**

NICTORS interface with the SCS through a MySQL database. In addition, development work has been done to completely implement all interfaces as defined by the SCS specification [18], making the SCS an OpenGIS standards compliant implementation. This includes an implementation of the SensorML specification which has been developed to satisfy our minimum requirements.



**Fig 6: Core classes of the SPS and their relationships**

### B. Sensor Planning Service

The introduction of WSRF to the SPS expands the service's capabilities to include the processing of multiple user plans simultaneously. The core classes which comprise the SPS are illustrated in Fig 6. 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. Architectural improvements to 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 are notified periodically when data is ready to collect. With the offload of some responsibilities from the SPS to the SCS, the SPS now has the potential to execute plans which were previously not possible. Such plans 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 [19]. 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 can also use historic observational data to reduce duplicate observation results. The implementations of some of these higher-level scheduling options still remain for future work.

### 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 historic observation results. No OpenGIS specification exists to this date for a SRS service. It is our intention to feedback our experiences to the OGC consortium in the near future. Additional work in defining the query interface and parameters necessary to facilitate these requests remains to be done.

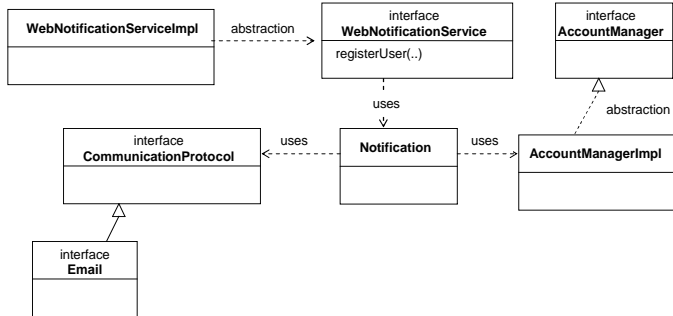


Fig 7: Core classes of the WNS and their relationships

### D. Web Notification Service

The WNS has been implemented as a WSRF service in a similar fashion to the SCS and SPS. The core classes are illustrated in Fig 7. 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 8 illustrates the experimental setup.

The experimental platform was developed using Crossbow's MOTE-KIT4x0 MICA2 Basic Kit [20] which consists of 3 Mica2 Radio boards, 2 MTS300 SensorBoards, and a MIB510 programming and serial interface board.

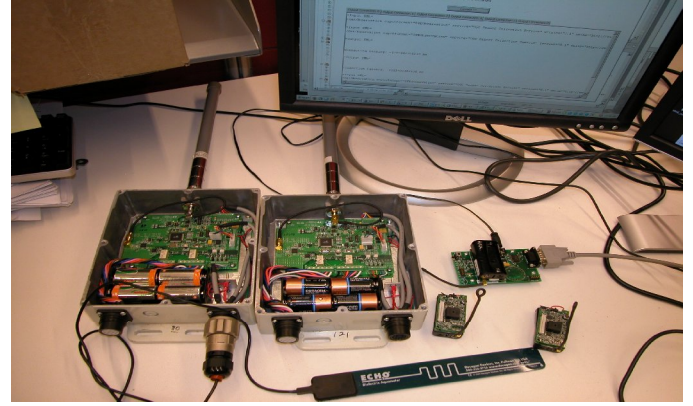
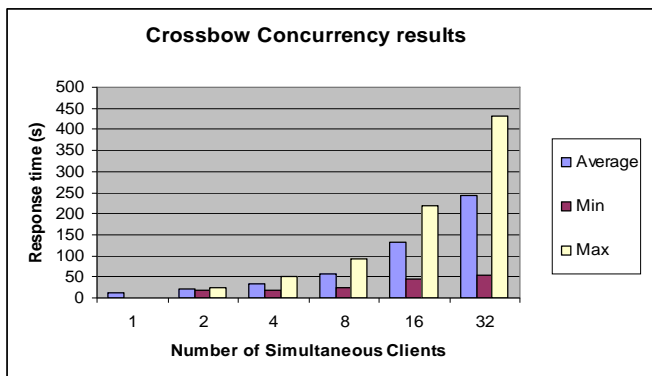


Fig 8: NICTOR sensors with their case removed and Crossbow mica2 motes connected to a workstation with client simulation software running in the background.

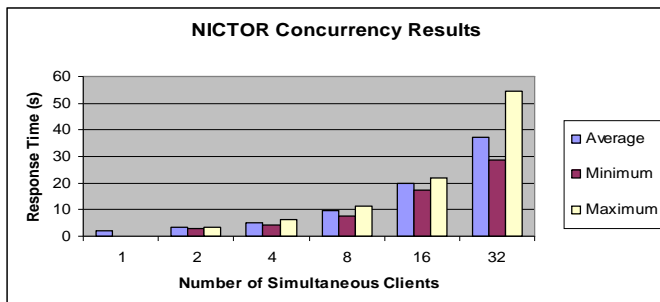
Two Mica2 motes were setup for sensing observational data. Light values were recorded at a sampling period of 100ms for a duration of 10 seconds. Once values were recorded the getObservation query would terminate. The numbers of simultaneous connections tested were {1, 2, 4, 8, 16, 32}. The relative response time is illustrated in Fig 9. The results illustrate 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 the SCS does not scale well. Degradation in the performance is an outcome of the limited concurrency available at the sensor network level. 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 were recorded for a duration of 2 seconds at a sampling period of 20ms. Once observations were retrieved the connection was terminated. Fig 10 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 wait till results are returned from the current query before a new query is sent. Implementing a caching mechanism could further improve the scalability of the SCS when dealing with NICTOR sensors.



**Fig 9: Duration of time recorded for simultaneous clients sending a getObservation() request to the SCS Mica2 notes.**



**Fig 10: 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.

## REFERENCES

- [1] OGC Consortium: <http://www.opengeospatial.org/>
- [2] Botts M, Percivall G, Reed C, Davidson J, OGC® Sensor Web Enablement: Overview And High Level Architecture, OpenGIS Consortium Inc, 2006.
- [3] SensorML: <http://vast.nsstc.uah.edu/SensorML/>
- [4] Cox S, Observations and Measurements OGC 05-087r3, Open Geospatial Consortium Inc, 2006.
- [5] Simonis I, Sensor Planning Service OGC 05-089r1, Open GIS Consortium Inc, 2005.
- [6] Simonis I, Wytzisk A, Web Notification Service OGC 03-008r2, Open GIS Consortium Inc, 2003.
- [7] Chu X, Kobiakka 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, Bangalore, India, 2006.
- [8] Hill J, Szewczyk R, Woo A, Hollar S, Culler D, Pister K, System architecture directions for network sensors, ASPLOS 2000, Cambridge, 2000.
- [9] Madden S, Franklin M, Hellerstein J, Hong W, TinyDB: An Acquisitional Query Processing System for Sensor Networks, ACM TODS, 2005.
- [10] Apache Tomcat: <http://tomcat.apache.org/>
- [12] Banks T, Web Services Resource Framework (WSRF) – Primer v1.2, OASIS, 2006.
- [13] Apache WSRF: <http://ws.apache.org/wsrfl/>
- [14] XML Beans: <http://xmlbeans.apache.org/overview.html>
- [15] Foster I, Globus Toolkit Version 4: Software for Service-Oriented Systems, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LMCS 3779, 2005.
- [16] Graham S, Murry B, Web Services Base Notification 1.2 (WS-BaseNotification), OASIS, 2004.
- [17] Gawor, J and Meder S, GT4 WS Java Core Design, Globus Alliance, 2004.
- [18] McCarty T, Sensor Collection Service OGC 03-023r1, Open GIS Consortium Inc, 2003.
- [19] 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, October, 2006, Singapore.
- [20] <http://www.xbow.com/Products/productsdetails.aspx>
- [21] NICTA Water Information project: [http://www.nicta.com.au/research/projects/water\\_information\\_networks](http://www.nicta.com.au/research/projects/water_information_networks)
- [22] Suman N, Jie L, Feng Z, Challenges in Building a Portal for Sensors World-Wide, First Workshop on World-Sensor-Web: Mobile Device Centric Sensory Networks and Applications (WSW'2006), Boulder CO, 2006.
- [23] Sensor Web: <http://sensorweb.geomatics.ucalgary.ca/>
- [24] Liang S, Coritoru, A., Tao, C. A Distributed Geo-Spatial Infrastructure for Smart Sensor Webs, Journal of Computers and Geosciences Vol.31, 2005.