Middleware-Based Sensor Web Integration

Yudong Tian, James V. Geiger, Jr., Hongbo Su, Sujay V. Kumar, and Paul R. Houser

***Abstract*—The Earth observation sensor web enables multiple-way interaction between earth observing sensors, sensor networks, Earth science models, and decision support systems. To achieve this goal, flexible and reliable integration between these disparate components is needed. In this study, a middleware-based, message-driven integration paradigm is proposed and implemented with the Land Information Sensor Web (LISW), to link a high-performance land surface modeling system with sensor simulators and other sensor web components, under a service-oriented architecture. OGC Sensor Web Enablement standard is adopted for interoperability. The middleware played a key role in enabling an integrated real-time sensor web with demonstrated simplicity, resilience and flexibility. We recommend that middleware-based integration should be adopted as a standard model in a wide range of sensor web applications, to replace the conventional point-to-point, client-server model.**

***Index Terms*—Distributed information systems, enterprise application integration, land information system, messaging, middleware, sensor web, service-oriented architecture.**

I. Introduction

THE Earth observation sensor web establishes a new paradigm to enable real-time, multiple-way interaction and collaboration between many disparate Earth observing and modeling applications. In contrast to the conventional one-way data flow from sensors to Earth system models, a sensor web provides an integrated framework within which Earth-observing sensors, sensor networks, sensor simulators, decision support systems, and Earth system modeling and data assimilation applications can communicate and collaborate with one another on the fly. An Earth system model, for example, no longer passively receives data from sensors; instead, based on its modeling and forecasting outcome, it can instruct sensors to reconfigure their sampling frequency and coverage for optimal scientific return and cost effectiveness.

Manuscript received October 15, 2009; revised March 29, 2010; accepted April 28, 2010. Date of publication June 21, 2010; date of current version December 15, 2010. This work was supported by the National Aeronautics and Space Administration Advanced Information Systems Technology Program under Grant AIST-05-0093.

Y. Tian is with the Earth System Science Interdisciplinary Center, University of Maryland, College Park, MD 20740 USA (e-mail: yudong.tian@nasa.gov).

J. V. Geiger, Jr. is with the Goddard Space Flight Center, NASA, Greenbelt, MD 20771 USA.

H. Su is with the Center for Research on Environment and Water, Calverton, MD 20705 USA.

S. V. Kumar is with Science Applications International Corporation, Beltsville, MD 20705 USA.

P. R. Houser is with the Department of Geography and Geoinformation Science, George Mason University, Fairfax, VA 22030 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTARS.2010.2049732

Many sensor webs have been developed recently, covering a wide range of Earth observing applications. For example, a volcano monitoring sensor web has been created by the Jet Propulsion Laboratory (JPL) to link *in situ* seismic sensors, spaceborne sensor platforms and physical models to form a fully autonomous system [1]. Mandl *et al.* [2] developed a sensor web for wildfire monitoring and emergency response with integrated observational data from multiple satellite platforms, such as Earth Observing-1 (EO-1).

The Land Information Sensor Web (LISW) [3] is one of the pilot efforts to showcase the sensor web's potential benefits in monitoring the land surface water and energy fluxes, and other hydrological applications, such as flood monitoring. LISW links a conventional high-performance land surface model to sensor simulators, visualization systems and other third-party applications within the integrated sensor web framework, and facilitates real-time, *ad hoc* data exchange between the components. The land surface model, the Goddard Space Flight Center Land Information System (LIS) [4]–[6], will not only be able to assimilate data from land surface-observing sensors, but also to instruct sensors to adjust their measurement strategy based on LIS's capability to predict the time and location of phenomena of greater interest. Additionally, third-party applications, such as a visualization system, can tap into the services provided by LISW to display scientific results.

LISW adopts the service-oriented architecture (SOA) [7], [8], and embraces the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) standards [9]. SOA provides a uniform framework to seamlessly integrate disparate applications by exposing their functionalities as services while hiding the implementation details of each service. For example, with LISW's SOA, a user can request the soil moisture data at specified locations from LIS via a standard service interface, without being concerned with the operation of LIS's modeling and data assimilation machinery in producing the requested data. OGC SWE ensures interoperability between the services within LISW, and across other sensor webs with SWE compliance.

This study concerns the optimal integration strategy for sensor webs. Currently the typical sensor web implementations with SOA are simply extending the point-to-point web application model, based on the client-server architecture. Such a model requires tight coupling between the various entities and synchronous communication, and lacks of resilience to failures. We explored a middleware-based integration strategy for LISW. Middleware-based integration has been widely adopted in areas such as enterprise application integration [10], [11] to support complex, distributed applications. There are many distinctive advantages in this strategy. Instead of a point-to-point communication model, we employed a message-based, publish/subscribe paradigm, and adopted an asynchronous, loosely coupled model. This model provides a communication substrate highly compatible with the SOA architecture, and as a result, making the implementation and integration of heterogeneous sensor web services considerably simpler and more reliable.

II. Basic Concepts

For web application-style sensor web implementation based on the conventional client-server architecture, the interacting entities are generally classified into clients and services, the communication between a client and a service is via a point-to-point channel, and data exchange is done in a synchronous, request-response interaction. This model generally works well when the entities are relatively few and the interactions are simple. But when there are many entities, and one client needs to contact many services, such as the case with sensor webs in which a client needs to receive data from many satellite platforms, the topology becomes increasingly complex [Fig. 1(a)]. Additionally, the tight coupling makes reliability a challenge, especially for real-time applications in a sensor web. For example, if the connection from one of the clients to a service is interrupted, the data supposed to be received by the client will be lost during this interruption, unless complex logic to deal with caching, session management and error recovery on the service side is implemented. Moreover, the synchronous nature of the communication ties up resources to handle the interaction, and prevents efficient implementation of interactive applications such as asynchronous JavaScript and Extensible Markup Language (XML) (AJAX)-based web interfaces [12].

A middleware-based integration pattern with a publish/subscribe model eliminates most of these deficiencies. As shown in Fig. 1(b), the use of middleware as an intermediary enables loosely coupled interaction between the entities, which act as publishers and subscribers. A publisher simply publishes a message to the middleware, and the generic middleware, a message broker, is responsible for routing a copy of the message to all the applications having subscribed to receive the data. The middleware mediates the interaction between the heterogeneous components, and ensures the reliability and persistence of the messages. This frees up the publishers from handling the tedious work, and greatly simplifies the implementation and logic of service providers. A crashed subscriber to services can recover the data missed during the offline period as the middleware maintains the persistence of the data.

III. Middleware

The middleware is the most critical component in constructing LISW's SWE framework. It can greatly simplify the implementation of the high-level web services and provide reliability, persistency and transparency for the data exchange.

There are basically four types of middleware [10]:

1) transaction processing monitors (TPM);
2) remote procedure call (RPC);
3) message-oriented middleware (MOM); and
4) object request brokers (ORB).

Table I summarizes some of the differences between these middleware types.

For sensor web integration under SOA architecture, message-oriented middleware (MOM) is the perfect choice, because the message-based data exchange maps well with web services and the publish/subscribe model fits the service-oriented architecture. Moreover, the asynchronous, loosely-coupled communication model is appealing due to its simplicity and reliability: a service provider and a consumer can run at their own pace; either one does not have to be ready when a service call is initiated. MOM's built-in store-and-forward mechanism guarantees the delivery of the message.

A critical assumption for MOM-based implementation is that the data are modeled as atomic, stateless messages. This will impose restrictions on complex transactions. However, for sensor web interactions such as those defined by SWE, the services

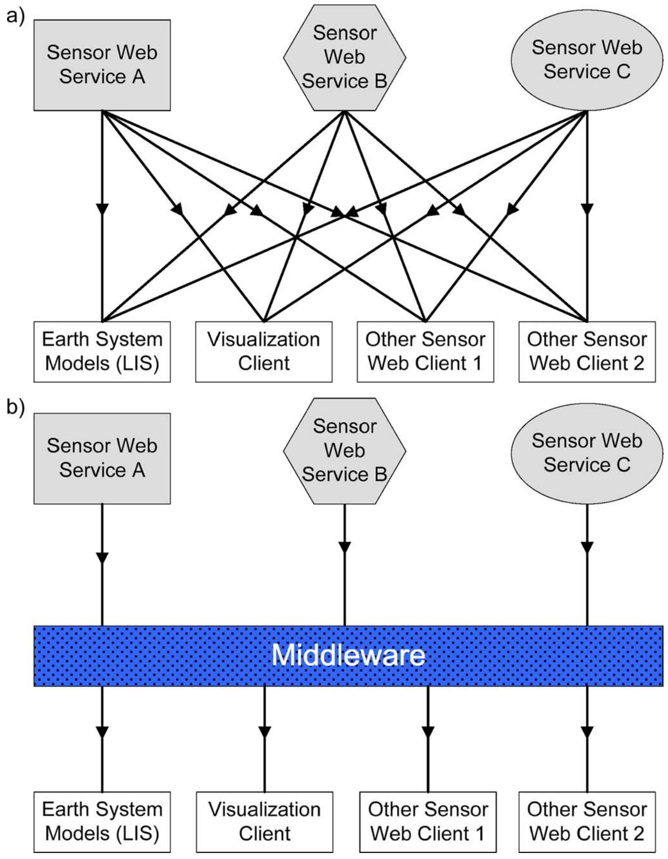


Fig. 1. (a) Point-to-point integration versus (b) middleware-based integration. When there are many interacting components such as the case with sensor web, the conventional point-to-point integration pattern leads to considerable complexity, both in communication channel topology and protocol semantics. Middleware eliminates these deficiencies and greatly simplifies the integration.

TABLE I
Comparison of Four Middleware Types

Category	TPM	RPC	MOM	ORB
Target	transactions	procedures	messages	objects
Coupling	tight	tight	loose	tight
Synchronization	sync	sync	async	sync
Typical product	IBM CICS	eCube NXTera	Microsoft MSMQ	BEA ObjectBroker

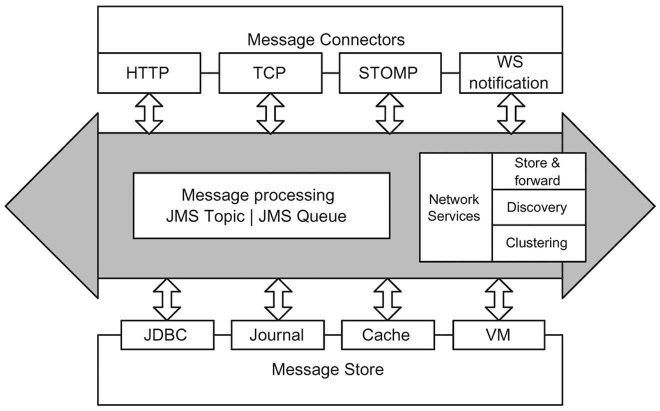
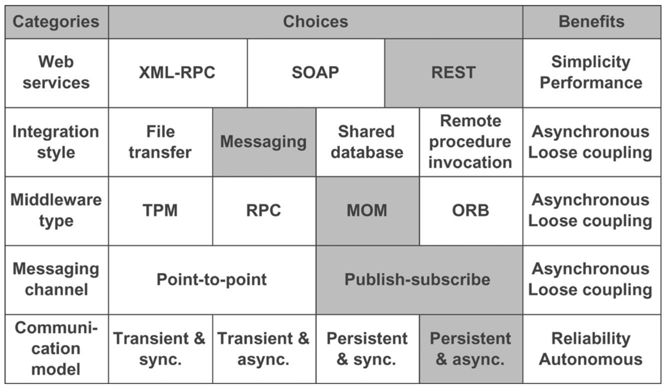


Fig. 2. ActiveMQ architecture.



Categories	Choices				Benefits
Web services	XML-RPC	SOAP	REST		Simplicity Performance
Integration style	File transfer	Messaging	Shared database	Remote procedure invocation	Asynchronous Loose coupling
Middleware type	TPM	RPC	MOM	ORB	Asynchronous Loose coupling
Messaging channel	Point-to-point		Publish-subscribe		Asynchronous Loose coupling
Communi-cation model	Transient & sync.	Transient & async.	Persistent & sync.	Persistent & async.	Reliability Autonomous

Fig. 3. Design choices for sensor web integration. Land Information Sensor Web (LISW) selects the shaded options for its service-oriented architecture. These decisions are critical for LISW's simplicity, flexibility and resilience.

are entirely based on exchanges of atomic, stateless XML messages, therefore MOM is more preferable to more complex middleware options.

For LISW integration, we choose the open-source ActiveMQ from the Apache Software Foundation [13] as our middleware, due to cost, openness, community support and maturity considerations. Nevertheless, our integration does not rely on any particular product; a drop-in replacement of the middleware can be easily performed.

ActiveMQ is a Java Messaging Service (JMS)-based MOM. It supports a publisher/subscriber programming model and reliable, asynchronous message delivery. Thus, it serves as a perfect messaging substrate to construct a loosely-coupled and robust system to integrate distributed applications.

Fig. 2 shows the software architecture of ActiveMQ [13]. It supports two types of JMS messages, topics and queues. JMS topics, with their associated publish/subscribe model, are particularly an optimal communication mechanism for the implementation of service-oriented architecture. In addition, ActiveMQ provides an HTTP connector which allows simple message delivery and receiving for web service-based applications. It also has several built-in, user-configurable mechanisms as message stores to ensure data persistency.

The asynchronous nature of the message delivery is particularly useful for the interaction between the web service providers and consumers. It allows loose interaction between the two parties and enhances performance, reliability and resilience of the whole system. Specifically, a message producer (such as a sensor simulator) does not need to engage in a lock-step send-acknowledge-send cycle with a message consumer (such as LIS), and it will not be tied down by a crashed consumer. The middleware will handle the message delivery reliably. This also paves the way for highly interactive web applications based on AJAX technology (see below), which can take full advantage of the middleware's asynchronous messaging characteristics.

IV. Implementation

A. Design Choices

Once the middleware is chosen, it is straightforward to implement the service-oriented architecture for LISW. Fig. 3 shows a series of design choices, anchored by the message-oriented middleware. At lower levels, the MOM middleware provides the publish/subscribe messaging channel and persistent, asynchronous communication model. At higher levels, the middleware dictates the integration of the disparate applications via messaging. For web service implementation, LIS chose REpresentational state transfer (REST) style [14], [15] for its simplicity. However, MOM does not exclude other web service implementations, such as Simple Object Access Protocol (SOAP). Though OGC is inclined to SOAP-based web services, this is not critical since REST-style web services can be used to deliver SOAP messages, and we expect REST-style web services will gain more acceptance within sensor web community as they do now in other sectors of web service providers, such as Amazon [16].

B. SWE Web Services

We implemented the SWE Sensor Observation Service (SOS) and Sensor Planning Service (SPS) web services to demonstrate the multiple-way interaction between the heterogeneous applications within LISW. Whereas other web services could also be implemented (e.g., Sensor Alert Service), these two web services are sufficient to demonstrate the reconfiguration and optimization capabilities of a sensor web.

These services will enable data retrieving, data assembling, data collection request handling, as well as the multiple-way communication between LIS and the other applications with this sensor web. These services would allow the sensor web to optimize a variety of objectives. At the most basic level, the sensor web should provide only the most important observations or information needed to minimize model error or to make the optimal decision, thereby minimizing observation, transmission, and data processing costs.

C. LISW Service-Oriented Architecture

Fig. 4 shows the implemented LISW service-oriented architecture. The middleware, ActiveMQ, is the center piece, responsible for message routing, data persistency and loose coupling. The middleware provides a built-in REST-style web service interface to its underlying JMS semantics, making it very simple for conventional applications, such LIS, to publish or subscribe

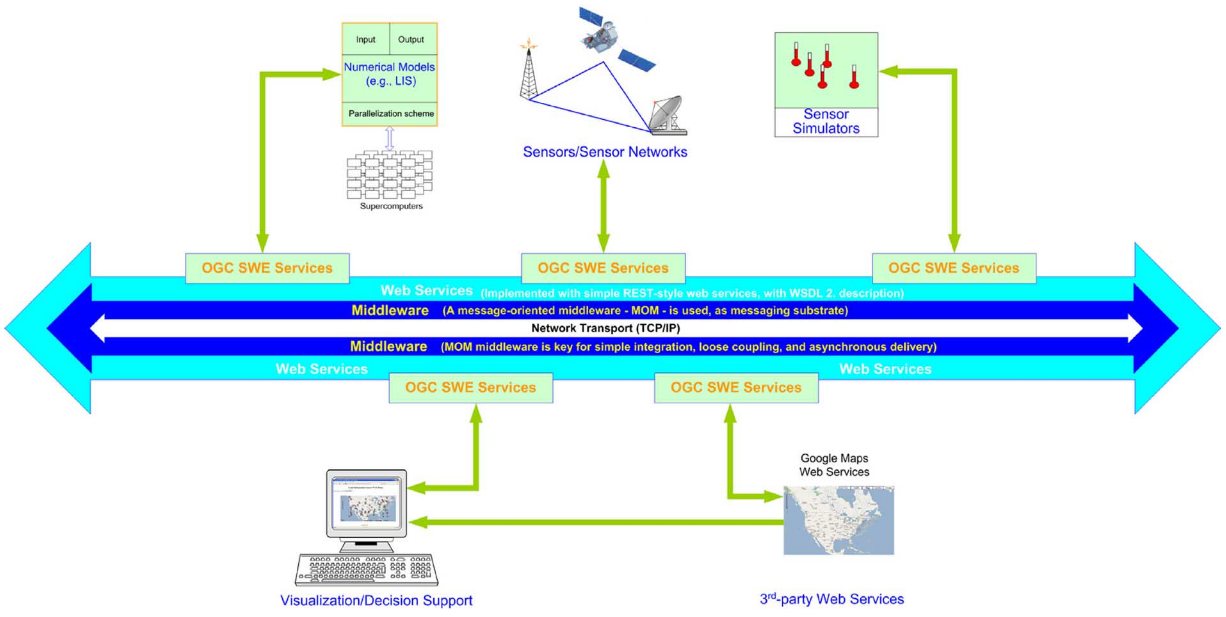

Input
Output
Numerical Models (e.g., LIS)
Parallelization scheme
Supercomputers
Sensors/Sensor Networks
Sensor Simulators
OGC SWE Services
OGC SWE Services
OGC SWE Services
Web Services (Implemented with simple REST-style web services, with WSDL 2. description)
Middleware (A message-oriented middleware - MOM - is used, as messaging substrate)
Network Transport (TCP/IP)
Middleware (MOM middleware is key for simple integration, loose coupling, and asynchronous delivery)
Web Services
Web Services
OGC SWE Services
OGC SWE Services
Google Maps Web Services
Visualization/Decision Support
3rd-party Web Services

Fig. 4. Land Information Sensor Web (LISW) service-oriented architecture. The message-oriented middleware plays the central role in providing a unified communication substrate, and in enabling *ad hoc* service provision and consumption.

various services with XML-encoded data. Confirming to different web service standards, such as OGC SWE, is just a matter of adopting the corresponding XML schema.

LIS can now act as both a service provider and consumer. When it subscribes to one of the SOS services provided by a particular sensor platform, such as a sensor simulator or third-party SWE-compliant real-time sensors, it will receive XML-encoded data from these SOS service providers as soon as they publish them. But LIS does not have to know any details of these SOS implementations or the providers. After LIS assimilates the sensor data and produces an uncertainty outlook to guide the reconfiguration of relevant sensor platforms, it simply publishes to an SPS service, and whichever sensor platforms have subscribed to this SPS will receive the data and can adjust themselves accordingly.

Fig. 5 illustrates the data flow within LISW for typical sensor web applications. LIS's land surface simulation and data assimilation capability provides a wide range of land surface states and fluxes data, such as surface temperature and soil moisture, with their associated uncertainties, in near real-time. These data can be made available to third-party applications, such as wildfire risk assessment [2] via the implemented SWE SOS service. Meanwhile, LIS can also assimilate field observations as a consumer of SOS services provided by third parties, to improve its forecasts accuracy. Such an end-to-end, autonomous data flow is a powerful paradigm to enable third-party applications to fully tap into LIS's data resources without any prior arrangements.

D. Integration With Third-Party Web Services

To visually illustrate LISW services, interactivity, and interoperability, we developed a web front-end as a consumer of LISW SOS service, and a publisher of SPS service. Additionally, the web front-end taps into a third-party web service, Google Maps, to get geospatial information. The result is an interactive and feature-rich web-based geospatial application.

Fig. 6 is a screenshot of this web front-end. The map information is retrieved from Google Maps with its specified web service application programming interfaces (APIs), with the usual user controls such as switching between "Map", "Satellite" and "Hybrid" views and zooming. Then we developed AJAX code to subscribe to an LISW SOS service, and publish an SPS service. The subscribed SOS service publishes temperature data over the continental U.S., generated by a sensor simulator in this case within LISW. Other land surface data, such as soil moisture, can be similarly obtained and displayed on the web interface easily. A user can click at a region over the continental U.S., to generate an SPS message for the temperature simulator, for example, instructing it to change its sampling frequency over this region from 1 sample every 10 seconds to 1 sample every 1 second.

The asynchronous nature of the underlying messaging system (ActiveMQ) is critical to the interactivity of our web front-end with AJAX. The communication between the front-end and the LISW SOS and SPS services is performed via a callback mechanism in the background. After the web front-end sends out an SOS request, it returns immediately to handle front-end user interaction without waiting for a response from the SOS service provider. When a response does come, a callback function is invoked to handle the service, including decoding the data and displaying them on Google Maps. Therefore, even if a service provider is slow to respond, or temporarily offline, it does not affect the front-end's responsiveness to user inputs.

V. Summary

The Earth observation sensor web establishes a new framework to support the integration of heterogeneous, distributed Earth observing sensors and models, with unified service-based interfaces for interoperability. LISW is an effort to link a conventional land surface modeling and data assimilation system, LIS, to sensors or sensor simulators, to demonstrate

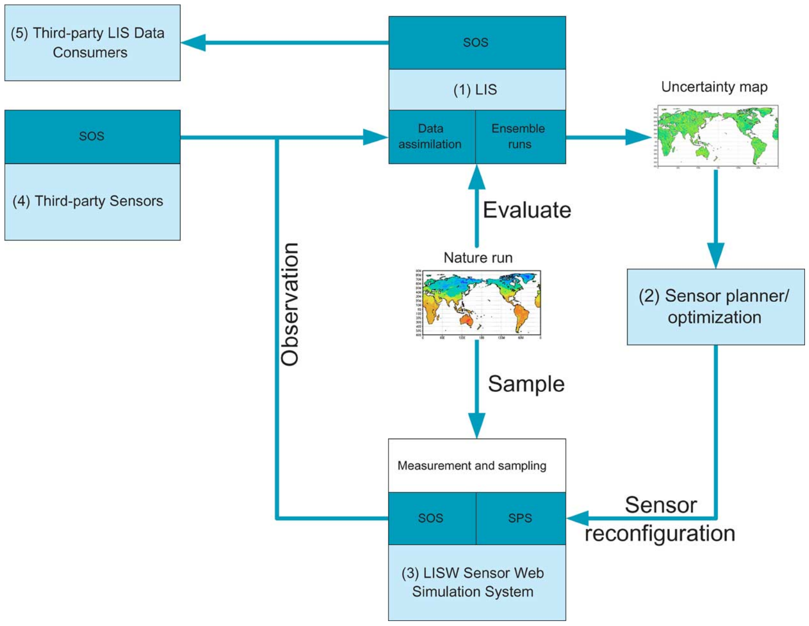
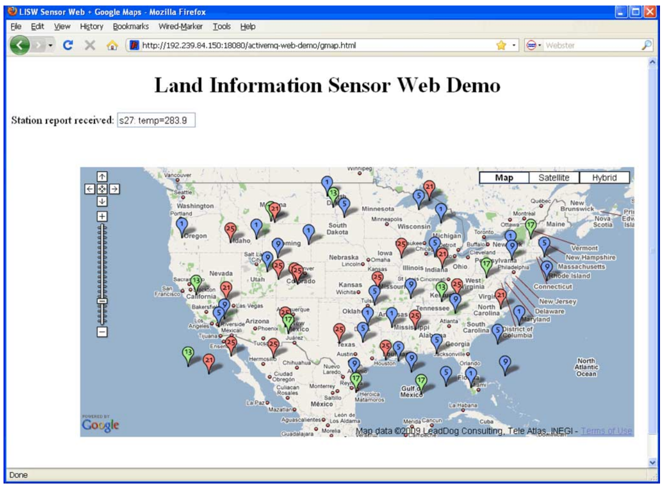


Fig. 5. End-to-end data flow for typical land surface simulation and data assimilation applications, with LISW SOA as shown in Fig. 4. LIS (1) can assimilate both simulated data from LISW simulators (3) and observations from third-party, SWE-compliant services (4). Estimates of uncertainties will also be produced to support decision making as well as sensor planning and optimization (2). The land surface data from LIS (1), such as surface temperature and soil moisture, provided via SWE SOS services, can support third-party applications such as wildfire risk assessment (5).


LISW Sensor Web - Google Maps - Mozilla Firefox
File Edit View History Bookmarks Wired-Marker Tools Help
http://192.239.84.150:18080/activemq-web-demo/gmap.html
Map Satellite Hybrid
Map data ©2009 LeadDog Consulting, Tele Atlas, INEGI - Terms of Use
Done

Land Information Sensor Web Demo

Station report received: s27: temp=283.9

Fig. 6. Displaying LISW's simulated temperature data over the continental U.S. on Google Maps. The temperature data were obtained via LISW's SOS in real-time. A user can also instruct the sensor simulator to change its sampling frequency by sending an SPS request.

the scientific and cost benefits resulting from the real-time, multiple-way interaction between the components. The analysis and forecast capability of an Earth system model, such as LIS, coupled with the on-demand re-configurability of modern sensors, such as EO-1, provides new opportunities to create intelligent, autonomous sensor webs for critical applications, such as flood monitoring.

One of the major technical challenges for sensor web development is the integration of the disparate and distributed components. LISW adopted a new, middleware-based approach, which significantly reduced the complexity and enhanced the reliability. In particular, our study shows that message-oriented middleware (MOM) maps well to the service-oriented architecture now widely adopted by the sensor web community. The middleware provides a transparent, coherent messaging substrate for all the components to communicate with one another reliably and asynchronously. With the middleware as an intermediary, direct, tight coupling between the components is eliminated, and one component's failure has much less impact on others than the conventional point-to-point direct coupling.

LISW embraces the OGC Sensor Web Enablement (SWE) specifications, and implements two principal SWE web services, Sensor Observation Service (SOS) and Sensor Planning Service (SPS). These services are essential in enabling the land surface models to obtain observations from sensors and to send re-configuration instructions to them, based on the models' forecast of significant events, such as flash floods. The middleware-based integration makes the implementation of these web services, or any *ad hoc* services, rather straightforward.

A web-based front-end was built to illustrate all the key advantages of the LISW integration. Simple AJAX code can be used to receive the sensor web services LISW provides. The front-end also exploits the asynchronous communication mechanism enabled by the middleware to ensure responsiveness to user inputs. It also provides an efficient and reliable way for a user to send SPS requests to LISW to re-configure a temperature simulator.

Based on many of the significant benefits obtained, we recommend the middleware-based integration pattern against the conventional, still widely adopted point-to-point pattern, for a wide range of sensor web applications. In particular, message-oriented middleware proves to be the best option for a simple, reliable and flexible service-oriented architecture due to the middleware's support of the persistent, reliable and asynchronous communication model, and its native enablement of the loose coupling between the components.

Acknowledgment

Y. Tian thanks Dr. Zhiwei Xu and his research group at University of Michigan, Dearborn, for assistance with software validation and verification. Helpful comments from two anonymous reviewers are greatly appreciated.

References

[1] S. Chien *et al.*, "An autonomous Earth-observing sensorweb," *IEEE Intel. Sys.*, vol. 20, pp. 16–24, May/Jun. 2005.

[2] D. Mandl *et al.*, "A space-based sensor web for disaster management," in *Proc. Int. Geoscience and Remote Sensing Symp.*, Boston, MA, Jul. 6–11, 2008, pp. 294–297.

[3] Y. Tian, P. R. Houser, H. Su, S. V. Kumar, and J. V. Geiger, Jr., "Integrating sensor webs with modeling and data-assimilation applications: An SOA implementation," in *Proc. 2008 IEEE Aerospace Conf.*, Big Sky, MT, Mar. 1–8, 2008, 7 pp.

[4] S. V. Kumar *et al.*, "Land information system—An interoperable framework for high resolution land surface modeling," *Environ. Modelling & Software*, vol. 21, pp. 1402–1415, 2006.

[5] C. D. Peters-Lidard *et al.*, "High-performance Earth system modeling with NASA/GSFC's land information system," *Innovat. Syst. Software Eng.*, vol. 3, pp. 157–165, 2007.

[6] Y. Tian *et al.*, "High performance land surface modeling with a Linux cluster," *Comput. Geosci.*, vol. 34, pp. 1492–1504, 2008.

[7] X. Chu and R. Buyya, "Service oriented sensor web," in *Sensor Network and Configuration: Fundamentals, Standards, Platforms, and Applications*, N. P. Mahalik, Ed. Berlin, Germany: Springer-Verlag, 2007, pp. 51–74.

[8] L. Di, G. Yu, and N. Chen, "A service-oriented general framework for facilitating the interaction and dynamic coupling between sensor web and Earth system models," in *Proc. Int. Geoscience and Remote Sensing Symp.*, Boston, MA, Jul. 6–11, 2008.

[9] M. Botts, G. Percivall, C. Reed, and J. Davidson, Eds., "OGC sensor web enablement: Overvieew and high level architecture," in *Open Geospatial Consortium*, 2007, p. 14 [Online]. Available: http://portal.opengeospatial.org/files/?artifact_id=25562

[10] P. A. Bernstein, "Middleware: A model for distributed services," *Commun. ACM*, vol. 39, pp. 86–97, 1996.

[11] G. Hohpe and B. Woolf*, Enterprise Integration Patterns*. New York: Addison-Wesley, 2004.

[12] N. C. Zakas, J. McPeak, and J. Fawcett*, Professional AJAX*. Indianapolis, IN: Wiley, 2006.

[13] ActiveMQ. Apache Software Foundation [Online]. Available: http://activemq.apache.org/

[14] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. California, Irvine, CA, 2000.

[15] L. Richardson and S. Ruby*, RESTful Web Services*. Sebastopol, CA: O'Reilly, 2007.

[16] Amazon Web Services. [Online]. Available: http://www.amazon.com/webservices

Yudong Tian received the Ph.D. degree in atmospheric sciences from the University of California at Los Angeles (UCLA) in 1999.

He conducted research in climate dynamics and geophysical fluid dynamics at UCLA. He was also one of the developers at UCLA for the popular Advanced Spectral Analysis SSA-MTM Toolkit. Between 2000 and 2002, he worked in the Internet industry, holding such positions as systems manager and chief technology officer. He joined GEST and NASA Goddard Space Flight Center in 2002, and has been working on various advanced information systems development for NASA and US Air Force. He is a core developer of GSFC's Land Information System, NASA's 2005 Software of the Year. He is now with the Earth System Science Interdisciplinary Center at the University of Maryland and NASA GSFC.

James V. Geiger, Jr. received the B.S. degree in mathematics from Old Dominion University, Charlottesville, VA, in 1993.

Since 1992, he has been a civil servant at NASA's Goddard Space Flight Center. He is currently a member of the Advanced Data Management and Analysis Branch and the Hydrological Sciences Branch. His research interests include land surface modeling, mathematics, computer science and parallel software development.

Hongbo Su received the Ph.D. degree in civil and environmental engineering from Princeton University, Princeton, NJ, in 2006.

He is currently with the Center for Research on Environment and Water, Calverton, MD. He has worked on quantitative remote sensing, including land surface temperature retrieval and terrestrial evapotranspiration estimation. His research currently focuses on integration of satellite observations into land surface modeling to study the water cycle and surface energy balance at regional or global scales.

Sujay V. Kumar received the B.Tech. degree in civil engineering from the Indian Institute of Technology, Bombay, India, in 1996, and the M.S. degree in civil engineering from North Carolina State University, Raleigh, in 1998. He continued at North Carolina State University for the Ph.D. in civil engineering with emphasis on computer-aided high-performance computing applications. He finished the Ph.D. degree in January 2002.

He is currently with Science Applications International Corporation, Beltsville, MD.

Paul R. Houser received the Ph.D. degree in hydrology and water resources from the University of Arizona, Tucson, in 1996.

He joined the NASA-GSFC Hydrological Sciences Branch and the Data Assimilation Office in 1997, served as manager of NASA's Land Surface Hydrology Program from 1999 to 2000, and served as branch head of the Hydrological Science Branch from 2000 to 2005. In 2005, he joined the George Mason University Climate Dynamics Program as the Professor of Global Hydrology, and formed the Center for Research for Environment and Water.