# Building a Sensor Grid for Real Time Global Positioning System Data

Galip Aydin[1,2], Zhigang Qi[1,2], Marlon E. Pierce[1], Yehuda Bock[3], and Geoffrey C. Fox[1,2]

[1]Community Grids Lab
[2]Department of Computer Science
Indiana University
501 North Morton Street
Suite 224
Bloomington, IN 47404

[3]Cecil H. and Ida M. Green Institute of Geophysics and Planetary Physics
Scripps Institution of Oceanography
La Jolla, CA 92093

{gaydin, zqi ,mpierce, gcf}@indiana.edu, ybock@ucsd.edu

**Abstract.** We describe the architecture of our streaming sensor grid system. Using a topic-based publish/subscribe methodology, we are able to build a scalable system for managing real-time data streams produced by the California Real Time GPS Network. The architecture is based on atomic, extensible elements called *filters* that receive, modify, and republish 1 Hz GPS data streams in our deployment. Our filter approach can be extended to include sophisticated data analysis and event detection applications.

**Keywords:** Real time data streams, global positioning system, sensor webs, publish/subscribe middleware, message-oriented middleware.

## 1. Introduction

Recent advancements in sensor technologies such as micro-circuitry, nano-technology and low-power electronics have allowed sensors to be deployed in a wide variety of environments [1-6]. The trend in this field shows that in the near future thousands of sensor nodes will be deployed either individually or as part of sensor networks in a large variety of application domains. Environmental monitoring, air pollution and water quality measurements, detection of the seismic events, and understanding the long-term motions of the Earth crust are example areas where the extent of the deployment of sensor networks can easily be seen. Extensive use of sensing devices

and deployment of the networks of sensors that can communicate with each other to achieve a larger sensing task will fundamentally change information gathering and processing [7].

Our work on developing a common Grid infrastructure for Geographic Information Systems has led us to the conclusion that this new type of data source is capable of producing very large amounts of observational data that potentially may help us obtain detailed knowledge about the environment we live in. Although the most common type of geographic data are kept in various types of data stores such as databases, the real-time sensor measurements will become the dominant type of data sources and have the capacity to produce tremendous amount of measurements. This data deluge may be more than the traditional systems can handle in normal operation. For instance Southern California Integrated GPS Network (SCIGN) [8] has deployed 250 continuously-operating GPS stations in Southern California whereas several hundred non-real time stations are operating elsewhere in the United States. The GPS Earth Observation Network System or GEONET in Japan consists of an array of 1200 GPS stations that cover the entire country with an average spacing of about 20 km [9]. These networks are capable of producing terabytes of measurements per year.

We see an important research problem in investigating appropriate mechanisms for managing the real-time data streams that are produced by these devices. We thus conduct systems research into implementing overlay networks to organize and transform sensor grid streams and investigate the mechanisms for delivering this data to applications.

## 2. Real Time Data Grid Components

### 2.1 Filters and Web Services

Filters in a Sensor Grid context are data processing applications, mostly operating in real-time. Similar to filters in electronics, which accept processes and output certain types of signals, a real-time software filter accepts transforms and presents messages. Depending on the task they are designed to accomplish, filters may be small applications such as format converters or very complex applications like simulation software. They may be expected to run in real-time by immediately processing and presenting the results or in near-real time by accumulating certain amount of data and executing the processing afterwards.


**Fig.** 1. Simple Filter concept includes a signal generator unit, actual data processing filter unit and the output signal.

The filters in our real-time data Grid architecture have three major parts corresponding to the three components depicted in Figure 1. The first component is the data input unit which is responsible for obtaining the data from the sources. In a real-time data Grid the data sources are sensors, or proxy servers which disseminate the real-time sensor measurements. The input unit must have the capability to access and present the data to the actual filter. The second component is the actual data processing unit. And the last component is a data output unit. Depending on the type of the Grid or client applications, the output unit may be implemented to support various data transfer protocols.

While designing real-time applications one obvious principle to remember is the importance of keeping the data flow from sources to destinations alive. Data processing in general requires multiple steps. For instance in a simplistic case, three steps would be required: accessing the data, converting them into application specific formats, and executing the actual processing. However in real world applications many more steps might be required to create a data processing flow. One must remember that any kind of interruption at some step of the flow will disrupt the entire process and possibly cause major breakdown because in the real-time systems the data are likely to be streamed continuously. For this reason it is wise to break down the data processing applications into as many small filters as possible and allow them to be accessed and controlled through standard interfaces. This approach helps creating robust real-time data flows because it allows distribution of the processing components and in turn integrating failsafe measures. For instance backup services could be used to replace any failed services thus allowing keeping the flow alive.

We adopt a Web Service approach to create standard control interfaces for the filters. Every filter in the system implements the same Web Service interface. The Filter Web Service interface provides capabilities such as *start*, *stop* and *resume*. It also provides a unique identifier for each filter, which can be useful for creating distributed chains. Another important feature of this service is to provide metadata descriptions of the filters.

In the larger picture each domain specific Grid has a specific filter chains. It is desirable for the Grid services to have access to the filter-chains at a given time for different reasons. For instance at the time of any server failures, it may be expected that the Sensor Grid restarts all the filter services in some chains. To be able to do this we need to keep metadata about the running, or potentially useful chains. For this reason we have developed an XML Schema for describing filter chains, as depicted in Figure 2.

Creating distributed systems requires successful orchestration of multiple remote resources. Minimizing human interference in this orchestration is also important for fast and accurate operation. However, to do this, the resources must be well defined. There must be a standard way for resources to communicate for successful integration. Web Services present us with a useful way for defining the service capabilities so that coupling the resources can be done automatically. In the filters case we also need additional metadata about each filter for creating filter chains. The metadata documents should contain unique properties of the filters such as its ID, input –output requirements, dependencies and a short description of the processing it is responsible for.

In Sensor Grid the filters are deployed as Web Services and in most cases run in a workflow as part of complex processes. Each filter is designed to execute a particular task, which means it accepts and produces certain type of messages. This introduces dependencies between the filters. Defining these dependencies in a filter specific metadata document is useful for checking if these dependencies are satisfied at the time of the deployment. This way the users will see which other filters must be deployed as well.

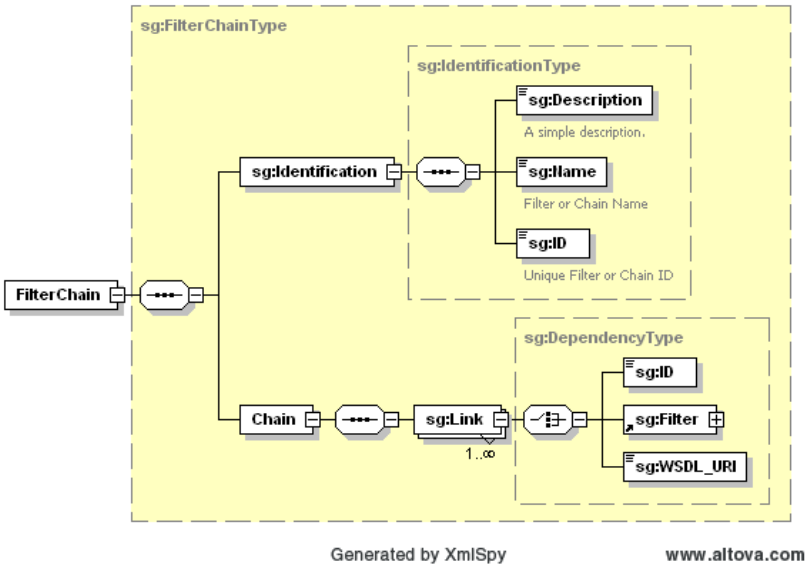


**Fig.** 2. XML Schema for the Filter Chains provides a simple data model for describing a collection of filters into a filter chain.

Figure 2 shows the metadata schema for the SensorGrid filters. It should be noted that the OGC SensorML [15] specification provides schemas for describing the sensor metadata and it could have been used here. However because of the complexity of the SensorML schemas and our system's requirement for providing additional schema to describe filter chains, we have decided to create a simple schema instead. For a fuller critique, see [16].

As described above, complex data processing tasks require multiple steps. In our architecture we use distributed filters for realizing such complex tasks. The standard Web Service interfaces for the filters allow remote creation and management of filter chains. We provide base classes for creating new filters and filter Web Services, which each particular filter extends. The filters are deployed as message sources and sinks in a publish-subscribe messaging system which federates the distributed resources and allows hierarchical operation of the filters.

Depending on the type of the processing the filters may be chained in parallel or serial modes. If the input data can be processed by different filters at the same time

and the results of them are merged after these independent processes are complete then the parallel operation is appropriate.

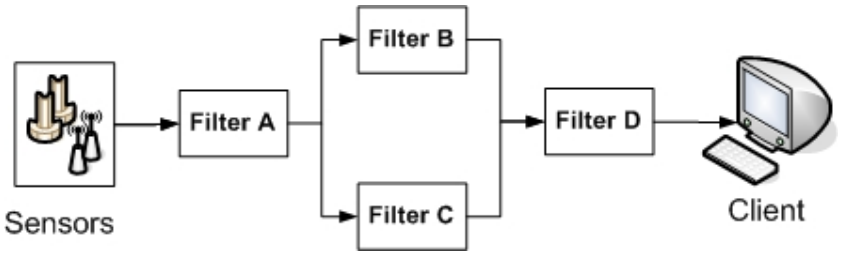


**Fig.** 3. Parallel operation of the filters allows endpoint filter D to aggregate content.

Figure 3 show the output of the Filter A is shared as input by Filters B and C, while the Filter D merges the outputs from both B and C and does the final processing. A simple example of this would be a filter that aggregates specific GPS sources into a personalized collection for an end user.

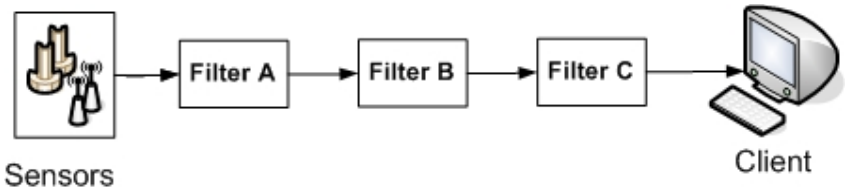


**Fig.** 4. Serial operation of the filters is the typical use case for GPS data streams.

However in our implementation, serial operations are dominant. A filter requires output from another as input, which also provides its output to the successive one as input. Figure 4 show three filters processing sensor messages in serial connection. As we will discuss below, both of these scenarios map directly to network-enabled publish/subscribe methods.

### 2.2 Managing Real Time Data Streams

To process GPS sensor streams in real-time we have developed several filters as Web Services to make real-time position messages available to scientific applications. In summary, the core of the system is to implement filter chains that convert or otherwise process the incoming data streams. These filters serve as both subscribers (data sinks) and publishers (data sources) for publish/subscribe middleware. NaradaBrokering [17] topics are used to organize different data stream sources into hierarchies. NaradaBrokering is a general topic-based publish/subscribe system that

supports the Java Messaging Service API as well as its own more extensive programming interfaces. The software's brokering scheme allows for multiple brokers in a distributed environment to be arranged hierarchically for efficient message routing and to operate as a single, logical broker. Currently the filters are being used to support 8 networks with 85 GPS Stations maintained by SOPAC [10].

In our architecture filters are small applications designed to perform simple tasks such as transforming or aggregating messages. We have developed an abstract filter interface which can be extended to create new filters. A basic filter is consisted of three parts: a NaradaBrokering subscriber, a publisher and a data processing unit. The entire system is illustrated in Figure 5.

The abstract filter interface provides subscriber and publisher capabilities. Typically a filter subscribes to a specified NaradaBrokering topic to receive streaming messages, process the received data and publishes the results to another topic. However outputs need not be always published. For instance, an Archiving Database Filter may only receive the station positions to insert into the database. We use this approach to optionally archive the streams, which may be replayed later.

The first filters we have developed are format converters that present original binary messages in different formats since applications require different representations of geographic data. Since the data provided by the RTD server is in a binary format, we developed filters to decode and present it in different formats. Once we receive the original binary data we immediately publish this to a NaradaBrokering topic. Another filter that converts the binary message to ASCII subscribes to this topic and publishes the output message to another topic. Another filter application subscribes to ASCII message topic and publishes a GML representation of the position messages to a different topic. This approach allows us to keep the original data intact and different formats of the messages accessible by multiple clients in a streaming fashion. The topic-based approach allows interested data subscribers to connect to the system at any point along the chain.

The GML Schema we developed for messages is based on the RichObservation type, which is an extended version of the GML 3 Observation model [13]. This model supports Observation Array and Observation Collection types which are useful in describing SOPAC Position messages since they are collections of multiple individual station positions. We follow strong naming conventions for naming the elements to make the Schema more understandable to the clients.

We used Apache XML Beans [14] for data binding purposes and created an application that reads ASCII position messages and generates GML instances. SOPAC GML Schema and sample instances are available at http://www.crisisgrid.org/schemas.

Station messages collected from GPS stations have several sub-sections. We have developed several filters that simplify or convert the messages since not all the parts of a position message are needed by most clients. Figure 5 shows the entire system including the GPS networks, proxy server, filters and the broker. The RTD Server is a real time proxy server that collects and streams GPS sensor sub-network position data on well-defined TCP/IP port numbers. This data typically is encoded in a binary format (RYO) and contains several different stations' data in a single message.

Reformatting, de-multiplexing, and analyzing the data is the job of our filter-based Sensor Grid system (right side of Figure 5).

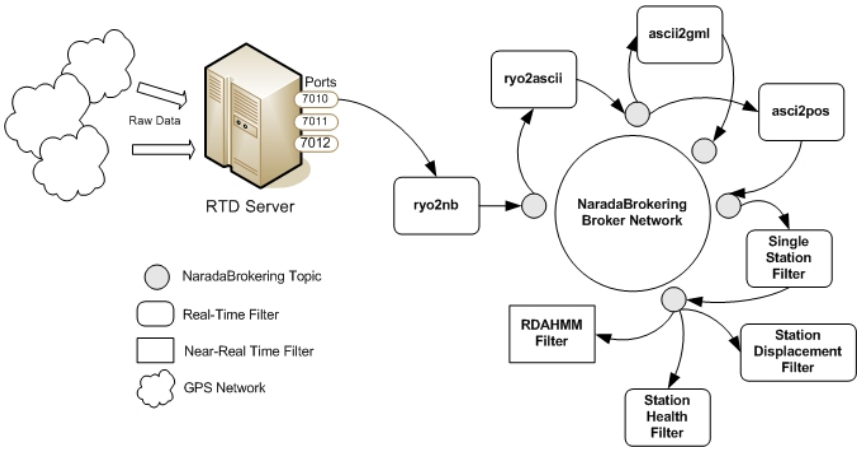


**Fig.** 5. Real-Time Filters for processing real-time GPS streams. See text for a full description.

### 2.3 System Performance

System tests for various specific cases are documented thoroughly in [19], and we summarize here. Basic tests were conducted with 24 hours of archived data for one of the California Real Time Network (CRTN)'s sub-networks (Parkfield). Stations on this network publish data at the rate of 1 Hz.

First, we demonstrated that the system's performance for simple GPS message transformation is stable for 24 hour periods of continuous operation with up to 1000 publishers or subscribers. The actual system has been deployed since August 2006, so these tests quantify what we observe anecdotally. Such results are not unexpected but are an effective way for uncovering slow memory leaks and related problems that downgrade the system's performance over time.

The overhead associated with publication is approximately 5 milliseconds even under heavy loads of up to 1000 simultaneous publishers (i.e. ryo2nb filters in Figure 5). Multiple subscribers place a longer delay on the system, which is an artifact of the underlying brokers rather than our filters. 1000 simultaneous subscribers to a single GPS topic introduce a delay of approximately 35-40 milliseconds on each message.

The limit of 1000 publishers or subscribers is an operating system limit. By using NaradaBrokering's ability to create logical brokers out of broker networks, we can exceed this operating system limit and have tested system performance for up to 1500 simultaneous publishers or subscribers. We expect additional scaling. The current CRTN consists of 8 live sub-networks, so we are well able to handle the current system demands.

## Conclusions

We have presented an overview of our sensor grid architecture for supporting real time data streams. In this paper, we have concentrated on describing the system components and metadata models that we use for describing our filter chains. This system can be used to make filter chains of varying complexity. The simplest filters are used to simply reformat the data into alternative formats (that is, convert binary to ASCII, convert ASCII to various XML formats, and so on). Each of these formats is associated with distinct topics in the underlying publish/subscribe system. In addition to formatting filters, more sophisticated filters such as event detectors can be integrated into the system following our approach. These filters can be combined into filter chains of various types to generate specific outputs. We describe these effort in more detail in [18] and [19]. In addition, thorough tests on the scalability and performance of this system are more thoroughly described in [19].

This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. This work was supported by the Advanced Information Systems Technology Program of NASA's Earth-Sun System Technology Office and by NASA SENH grant NAG5-13269 (Scripps) and NASA ACCESS grant NNG06GG94A (Indiana University). Use of the Geodetics, Inc. RTD software was provided by the University of California, San Diego.

## References

1. Akyildiz, I.F., et al., A Survey on Sensor Networks. IEEE Communications Magazine, 2002.
2. Akyildiz, I.F., et al., Wireless sensor networks: a survey. 2002, Elsevier. p. 393-422.
3. Chong, C.Y., S.P. Kumar, and B.A. Hamilton, Sensor networks: evolution, opportunities, and challenges. Proceedings of the IEEE, 2003. 91(8): p. 1247-1256.
4. Delin, K.A., The Sensor Web: A Macro-Instrument for Coordinated Sensing. Sensors, 2002 2002. 2(1): p. 270–285.
5. Delin, K.A. and S.P. Jackson, The Sensor Web: A New Instrument Concept. 2001. p. 20-26.
6. Martinez, K., J.K. Hart, and R. Ong, Environmental sensor networks. 2004. p. 50-56.
7. Estrin, D., et al., Next century challenges: scalable coordination in sensor networks. 1999, ACM Press New York, NY, USA. p. 263-270.
8. Hudnut, K.W., et al., The Southern California Integrated GPS Network (SCIGN). 2002. p. 167–189.
9. Yamagiwa, A., Y. Bock, and J. Genrich. Real-time monitoring of crustal deformation using large GPS geodetic networks-Japanese GEONET's potential as a natural hazards mitigation system. in American Geophysical Union, Fall Meeting 2004, abstract #SF53A-0724. 2004.
10. Bock, Y., et al., Scripps Orbit and Permanent Array Center (SOPAC) and Southern California Permanent GPS Geodetic Array (PGGA). 1997, National Academy Press. p. 55–61.

11. Hudnut, K.W., et al. THE SOUTHERN CALIFORNIA INTEGRATED GPS NETWORK (SCIGN). in The 10th FIG International Symposium on Deformation Measurements. 2001. Orange, California, USA.
12. Bock, Y., L. Prawirodirdjo, and T.I. Melbourne, Detection of arbitrarily large dynamic ground motions with a dense high-rate GPS network. GEOPHYSICAL RESEARCH LETTERS, 2004. 31.
13. Cox, S. (2003) Observations and Measurements. Volume, DOI: OGC 03-022r3
14. Apache, X.M.L., Beans Project, http://xmlbeans.apache.org/. 2003.
15. Botts, M., Sensor model language (SensorML) for in-situ and remote sensors specification. 2002, discussion paper 02-026r4, Open GIS Consortium. http://www. opengis. org/techno/discussions/02-026r4. pdf.
16. Aydin, G.: An Overview of the Open Geospatial Consortium Standards and their use in Community Grids Laboratory. Community Grids Laboratory Internal Technical Report.
17. Pallickara, S., Fox, G.C.: NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Middleware 2003: 41-61.
18. Granat, R., Galip Aydin, Marlon Pierce and Zhigang Qi, Analysis of Streaming GPS Measurements of Surface Displacement Through a Web Services Environment, 2007 IEEE Symposium on Computational Intelligence and Data Mining, April 1-5 2007, Honolulu, HI, USA. Preprint available from http://grids.ucs.indiana.edu/ptliupages/publications/GranatPierceAydinQi_IEEE_CIDM07_submitted.pdf.
19. Adyin, G., Qi, Z., Pierce, M. E., Fox, G. C., and Bock, Y. : Architecture, Performance, and Scalability of a Real-Time Global Positioning System Data Grid. Submitted to Physics of the Earth and Planetary Materials.