A Collaborative Sensor Grids Framework

Rui Wang, Geoffrey Fox, and Alex Ho
Anabas, Inc
Suite 106C, 501 North Morton Street
Bloomington, IN 47404
812-856-1212
rui.wang@anabas.com, gcf@indiana.edu, alexho@anabas.edu

Abstract

Integrating sensors and grid computing provides a way to gather, process and model real-time information from the physical world for right, timely decisions and actions to be taken in responding to events over a large scale. We create a sensor grid including various types of sensors, use a grid builder tool for discovering and managing remote, distributed sensors, and provide a unified interface for displaying real-time data from those sensors. Also, we used advanced collaborative technology (i.e., Impromptu software) to enable distributed users to have a consistent view of the displayed information.

1. Introduction

Increased use of sensors in commercial and military environments is being driven by the need for better intelligence data and by advancement in technology, which provides smaller, less costly and more capable sensors. It is, however, not sufficient and in many situations not productive to just provide lots of sensor data to decision-makers at all levels for their missions on hand. It is valuable to have a framework that supports seamless integration of loosely-coupled and custom-developed sensor data management, visualization and presentation tools, and real-time collaboration capability for sharing situational awareness. Thus, real-time information about the physical world can be gathered, processed, correlated and shared to facilitate quick, reliable decision making on a large scale.

This framework directly integrates sensors with grid computing, which is essentially the federation of large-scale, heterogeneous computational servers on top of high-speed network connections. Middleware technologies such as Globus and Gridbus [1] enable secure and convenient sharing of resources (e.g., CPU, memory, storage, content and databases) by distributed users and applications. There has been an increasing demand for applying grid computing in the fields of bioinformatics, drug design, engineering design, business, manufacturing and logistics.

Our research is motivated to design and develop an enabling framework to support the development, deployment, management and real- time visualization and presentation of collaborative, geo-coded sensor grid applications with flexibility, extensibility and scalability. The Collaborative Sensor Grids (CSG) framework underlying technology is based on an event-driven (publish/subscribe) model that utilizes a "publish and subscribe" communication paradigm over a distributed NaradaBrokering (NB) message-based transport network. A key capability component is the Grid Builder sharedlet which supports the assemblage of grid of grids - a compositional model of assembling a multitude of subgrids into a mission-specific grid application. For the CSG framework, the Grid Builder sharedlet facilitates the assemblage of two important subgrids, namely a real-time multimedia collaboration grid and hierarchical, executable sensor grid.

In the next section, we describe some physical sensors, our system architecture and how it was implemented. Then, section 3 explains how the Grid Builder tool facilitates the discovery and management of distributed sensors. Visualization and presentation for displaying sensor data in the collaboration grid are illustrated in section 4. Finally, section 5 summarizes the paper with conclusions and future work.

2. Collaborative Sensor Grids

2.1. Sensors

Sensor grids are a combination of sensor networks and grid computing. As a result of recent advances in electronic circuit miniaturization and micro-electromechanical systems (MEMS), small sensor nods that integrate several kinds of sensors, a Real Time Data (RTD) server and a wireless transceiver can be created to form the sensor network. Basically, a sensor is a type of transducer which uses one type of energy, a signal of some sort, and converts it into a reading for the purpose of information transfer. We used a small-sized tablet computer (i.e., Nokia N800) as the RTD server, which has considerable computing power with supports for both Wi-Fi and Bluetooth connections. Different types of sensors have been provided in sensor grids. They provide various sources of real-time information as follows:

1) GPS sensor: It is a portable GPS receiver, which receives geospatial location information (e.g., latitude, longitude, etc) from satellites and transfers such information to the RTD server via Bluetooth connection.

2) Video/Audio: Those sensors, which are actually the built-in webcam and microphone in the Nokia N800 tablet, provide video/audio streams that can be broadcasted over the Internet.

3) RFID sensor: This sensor includes both RFID tags and a reader. The reader senses information about RFID tags such as the signal strength, motion, temper and panic and encapsulates the information in tag event messages. The RTD server gets and processes those messages via Bluetooth

4) Lego Robot: There are four sensors equipped in the Lego Robot, which are sound, light, touch and ultrasonic sensors. Their detected information is sent to the RTD server via the Bluetooth module in the robot. The robot can also act as instructed by taking programmable commands from a computer (e.g., the RTD server).

Table 1. Sensor Types and Attributes

Sensor Type	*Attributes*
GPS	- Time - Latitude - Longitude - ID
Video/Audio	- Video stream - Audio stream
RFID	- Tag ID - Group code - Motion or stationary - Signal strength
Lego Robot	- Sound - Light - Touch - Ultrasonic

2.2. System Architecture

Our approach uses a scalable, hierarchical and collaborative architecture. Hybrid, large-scale distributed sensor nodes are loosely coupled by a message-oriented middleware system called NaradaBrokering (NB) [2], which is a content distribution infrastructure based on the publish/subscribe paradigm.

The NaradaBrokering substrate itself comprises a distributed network of cooperating broker nodes. One reason for choosing NaradaBrokering as the middleware fabric for Sensor Grids is that it places no restrictions on the *type* of the content: it has been deployed in systems where the content has been GIS data, multimedia codecs, images, text, bit maps and objects among others. The substrate places no constraints on the size, rate and scope of the interactions encapsulated within the streams, or on the number of entities within the system. Also, NaradaBrokering has other useful features such as: secure end-to-end delivery of streams, robust stream disseminations, efficient ordering and synchronization of streams, support for rich Quality of Services, support for multiple transport protocols and support for Web Services, etc.

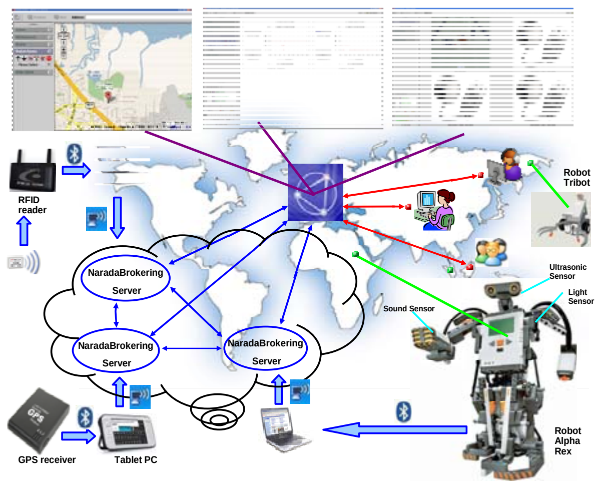


Figure 1. Collaborative Sensor Grids system architecture

The system architecture of Collaborative Sensor Grids is depicted in Figure 1. There are different types of sensors distributed globally. Each sensor (GPS, Video/Audio, RFID, etc) keeps gathering real-time information that it is sensitive to from the environment. A RTD server being connected with the sensor processes the incoming raw data and publishes refined information encoded in certain message format to a NaradaBrokering server. Messages are transported among brokers in NaradaBrokering. Running on a computer for presenting such information, a simple client program can subscribe to a specific topic of NaradaBrokering in order to receive relevant messages. Different information from different types of sensors will be presented to users through Anabas collaboration software Impromptu. In Section 4, we will describe the Impromptu User Interface that includes multiple Sharedlets and explain how to use it to share such information among remote, distributed users.

High-speed, reliable physical network is always a key requirement for this system. There are two types of network connections in the architecture: Bluetooth and WiFi. Bluetooth is used for the connection between a sensor and a RTD server since most sensors are Bluetooth enabled. It is efficient and reliable for transporting data in a short range. For the connection between RTD servers and NaradaBrokering, WiFi is desired because a RTD server usually needs to be portable with sensors and its distance to a NaradaBrokering server may be varying and long.

The overall Sensor Grid system consists of sensor units arranged hierarchically. The sensed information of each sensor is processed by one RTD server. Also, there are sensor grid servers which take the role of brokers. It communicates with a collaboration grid server where meeting sessions reside as well as sensors. Primary functions of a sensor grid server are to manage and broker sensor message flows. It optimizes bandwidth usage that by forwarding messages should only be forwarded to necessary nodes that needed them only. This server is stateful and has memory of connections of sensor and meeting sessions. An overview of the sensor data flow architecture is shown in Figure 2. This structure makes the system scalable in a wide-area deployment. It can be scaled to manipulate hundreds or thousands of sensors in real time, no matter where their physical locations are. With the integration of collaboration grid, users also have a consistent view of extracted, processed and organized information from all those sensors.

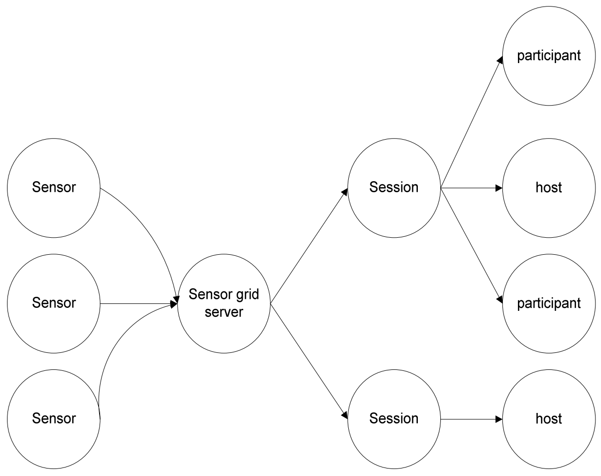


Figure 2. Overview of the sensor data flow

2.3. Implementation of Sensor Grids

We adopted a general process to implement Sensor Grids based on the architecture described before. The major steps include:

1) Establishing the Bluetooth connection between a sensor and a RTD server;
2) On the RTD server, a filter service extracts useful information from raw data through the established connection;
3) Refined data is encoded in messages and published to an NB topic;
4) On a computer for presentation, a client program subscribes to the specific NB topic to receive relevant messages;
5) Retrieved messages are parsed to get information that users are interested;

6) The information is presented in a defined collaborative sharedlet.

We take implementing the GPS sensor grid as a concrete example here:

First, we use a portable tablet Nokia N800 as the RTD server, which connects a GPS receiver device (I-Blue) via Bluetooth. A client program extracts useful information such as latitude, longitude and time from incoming raw data. Then it publishes encoded messages to a NB topic (*Streams/GPS/Location1*) with a certain frequency (e.g. every 3 seconds).

Next, on the computer for presenting GPS information to users, a NB client subscribes to the same topic (*Streams/GPS/Location1*) to retrieve real-time data of the GPS location. We use the Google Maps API to show the GPS current location on the map based on its geospatial location information (latitude and longitude). Also, the GPS ID is marked on the map to distinguish it from other GPS sensors.

Finally, the map is integrated with Impromptu as a sharedlet. A user can have a global view of all GPS sensors or select a single GPS location to view its details. The sharedlet also provides functions specific to geographic information such as zoom in, zoom out, and panning the map to different directions.

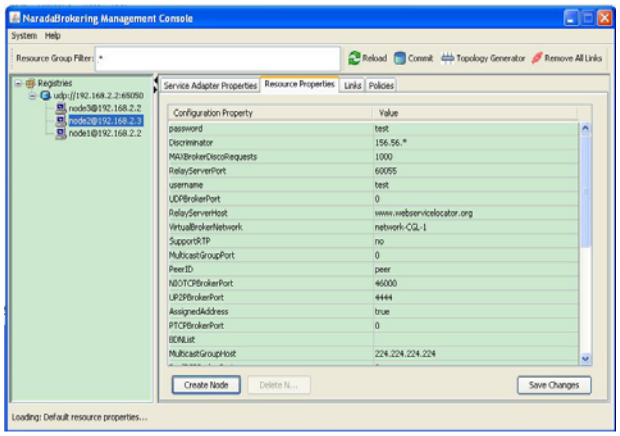
For other sensors like Video/Audio, RFID and Lego Robot, they are implemented following the same pattern. Major differences between implementing different types of sensors lie in the filter for extracting sensed information from raw data and the presentation interface for such information.

One issue worthwhile to mention is that several filters may process sensed information collaboratively as a workflow. For example, after the RTD server receives raw data from the stations it generates and broadcasts messages through a port in a binary format called RYO. One filter on the client side can decode captured RYO messages into text format. Another filter can convert text format to Geography Markup Language (GML) format, since different users may be interested in geographic information in different formats. Further research is ongoing to study the issue of workflow in Sensor Grids.

3. Sensor Grid Management

Sensors services need to be managed over the network so that the user can find out new sensors monitor the status of existing sensors and recover from failure by replacing malfunctioning sensors. We extended a generic management framework that is capable of managing any type of service with modest external state [3] for Sensor Grids. Web service based protocols are implemented in the framework to provide interoperable management. Existing management systems can be effectively integrated. A hierarchical bootstrapping mechanism is deployed to scale the management framework over a wide-area. The framework is tolerant to failures within the framework itself while service failure is handled by executing user-defined failure handling policies. The unit of management framework consists of a set of manageable services, their associated service mangers, message nodes (to provide a scalable messaging substrate) and a scalable, fault-tolerant data structure called registry.

In order to make a sensor service manageable by the management tool, we implemented a specific sensor service adapter and a sensor manager following WS Management. The interface of the Grid Builder tool that implements the management framework is shown in Figure 3. The user can easily view a list of discovered sensor services in the left panel and select one to get more detailed information about it such as its location, the host address and its UUID, etc. The user can also deploy a sensor service on a remote sensor node through the application. Thus, in case one necessary sensor service becomes unavailable due to some reasons, it is still possible to find an alternative sensor to start the same service.



Configuration Property	Value
password	test
Discriminator	156.56.*
MAXBrokerDiscoRequests	1000
RelayServerPort	60055
username	test
UDPBrokerPort	0
RelayServerHost	www.webservicelocator.org
VirtualBrokerNetwork	network-CGL-1
SupportRTP	no
MulticastGroupPort	0
PeerID	peer
NIOTCPBrokerPort	46000
UP2PBrokerPort	4444
AssignedAddress	true
PTCPBrokerPort	0
BDNList	
MulticastGroupHost	224.224.224.224

Loading: Default resource properties...

Figure 3. Grid Builder tool interface

The process of finding new sensors is based on WS_Discovery specification, in which the message exchanging between a sensor service adapter and a sensor manager is described as below:

1) Initially a sensor sends a multicast Hello message when it joins a network;

2) The sensor manager multicasts a Probe message with Type and/or Scope to discover existing sensors;

3) A sensor may receive a multicast Probe message and send a unicast Probe Match (PM) if it matches that Probe;

4) The sensor manager multicasts a Resolve message with Name to discover existing sensors;

5) A sensor may receive a multicast Resolve message and send a unicast Resolve Match (RM) if it matches that Resolve;

6) When a sensor leaves a network, it sends a multicast Bye message

7) The sensor manager will add/remove a sensor when it receives a multicast Hello/Bye message.

The Grid Builder tool follows the idea of constructing grid of grids, which assembles a multitude of subgrids into a mission-specific grid application. In the CSG framework, the Grid Builder tool facilitates the assemblage of two important subgrids, namely a real-time multimedia collaboration grid and hierarchical, executable sensor grid. For example, many types of sensors have controls for changing their respective behavior and to perform actions. However, the specific action and its corresponding control operation vary from sensors to sensors. To facilitate the control of sensors in an end-user application, the CSG client will support sets of customable controls to be defined by users with Grid Builder. This information will be transmitted to the respective sensor sharedlets in order to construct appropriate buttons for sensor control. We provided some initial sensor sharedlet controls for certain generic sensor types, which will be shown in the next section.

4. Visualization and Presentation

A particular design objective for the CSG client is to provide an intuitive user interface for enabling UDOP (User Defined Operation Picture) and COP (Common Operation Picture) capabilities, which are essential for agile formulation and sharing of visual, situational awareness and effective decision-support.

To have the basic support of sensors, users must be able to visualize sensors' data and to control sensors. Some basic principles for visualizing sensor are:

1) Groups of geo-located sensors can be visualized by their geospatial locations;
2) Each sensor can be visualized particularly based on its property;
3) Sensors can be grouped and visualized on a single canvas
4) Multiple canvas can be viewed concurrently
5) The state can be captured

Also, each sensor should accept control which can be none. And those control functions should be generated according to sensor's controllable capability. Multiple sensors can be controlled simultaneously.

We used Anabas Impromptu as the collaboration solution for sharing information from sensors among distributed users. It is a system for real time shared display, audio, and chats. Its portal consists of multiple sharedlets, each of which can be a standalone, shared application. Different sharedlets were developed for handling and displaying different types of real time information from sensors in Impromptu.

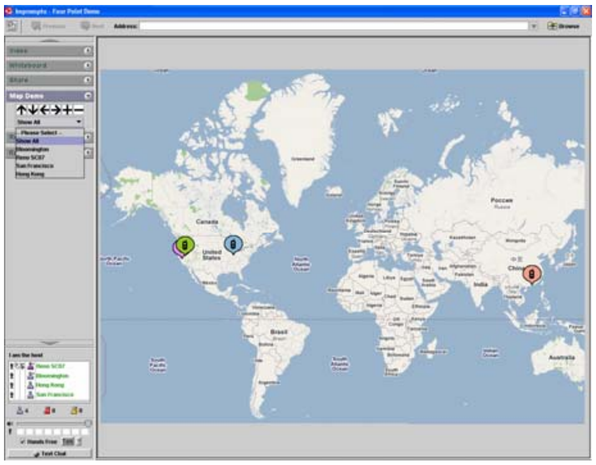


Figure 4. GPS Sensors sharedlet interface

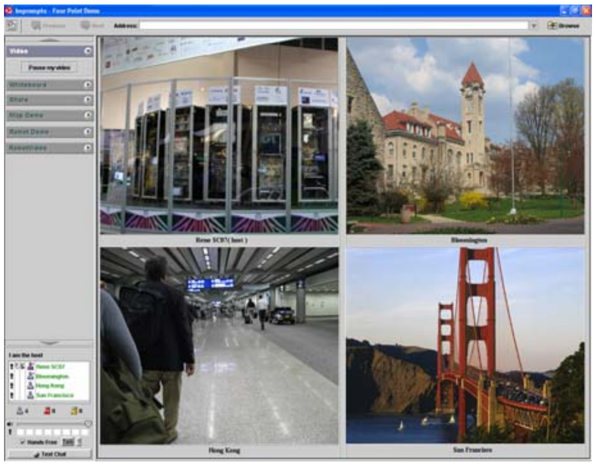


Figure 5. Video Sensors sharedlet interface

Figure 4 and Figure 5 show the display for GPS and Video sensors, respectively. The GPS sharedlet includes a Google Map with four different GPS locations marked on. Users can use the supported control to zoom in, zoom out, and move the map around via the sensor control tool menu. In the Video sharedlet, there are four windows streaming real time videos from those locations without any user control.

Figure 6 presents a case that a user wants to control a Lego Robot and see its sensor information. The sensor information includes not only its built-in sensors such as light intensity, ultrasonic and voice sensors, but also GPS, RFID and video sensors. Thus, it is a composite sensor consisting of multiple primitive sensors.

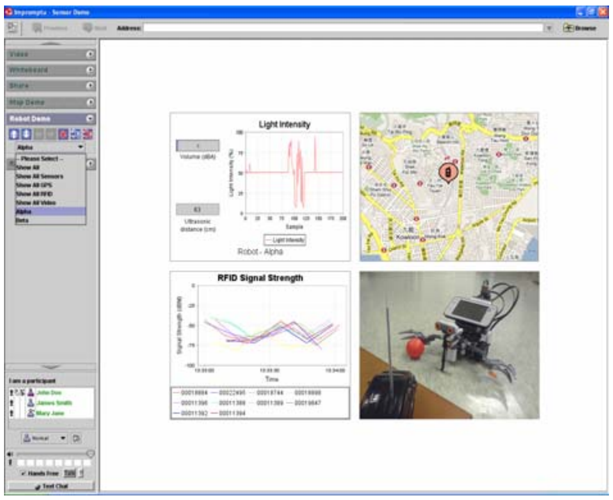


Figure 6. Lego Robot and RFID sensors

5. Conclusions and Future Work

We have developed a framework to support the development, deployment, management, visualization of collaborative, geospatially located sensor grids with flexibility, extensibility and scalability. Based on the framework, we created sensor grid applications for various types of sensors, used a Grid Builder tool for discovering and managing those sensors, and provided a rich, informative user interface for presenting real-time sensor data to users. Also, we used advanced collaborative technology built on top of message-oriented middleware to enable distrusted users to have a consistent view of such information.

There are still unfinished issues for future work. First, currently the Grid Builder tool is standalone. It is expected to be integrated with Impromptu as a sharedlet. Not only it will make the interface more consistent but also it can link the collaboration grid with the sensor grid better. Second, it is expected that a user can customize controls for new types of sensors. Predefined controls may not be suitable for those. More interactions between users and the system can be useful. Last, information from different sensors can be synthesized through a workflow to make it more meaningful to users.

References

[1] The Gridbus Project. http://www.gridbus.org
[2] The NaradaBrokering Project. http://www.naradabrokering.org

[3] Harshwardhan Gdgil, Geoffrey Fox, Shrideep Pallickara, and Marlon Pierce, "Scalable, Fault-tolerant Management of Grid Services", in Proceedings of IEEE Cluster, 2007.