University of Alberta

A Service-Oriented Framework for Sensor-Based Applications

by

Jianzhao Huang

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Examining Committee

Eleni Stroulia, Computing Science, University of Alberta

Pawel Gburzynski, Computing Science, University Alberta

Ken Wong, Computing Science, University of Alberta

Arie Croitoru, Earth & Atmospheric Sciences, University of Alberta

Abstract

Sensor technologies are improving fast: sensors are being developed to record more types of phenomena in improving precisions, and they are becoming less expensive. The vision is that sensor networks deployed across large spaces, attached to important infrastructure, and embedded in our everyday environments, will become an ubiquitous element of the world's information infrastructure. The apparent bottleneck in reaching this vision efficiently and cost effectively is software development. In this thesis, we discuss an integrated service-oriented architecture for collecting, archiving, analyzing and visualizing sensor network data. The framework has been deployed and evaluated in two applications: SensorGIS and SmartCondo, designed for sensor networks deployed in outdoor and indoor spaces.

Acknowledgements

I am heartily thankful to my supervisors, Dr. Eleni Stroulia and Dr. Pawel Gburzynski who provide guidance and support all the way through. I would also like to thank Nicholas Boers for professional advices on wireless sensor network and David Chodos for 3D virtual world technical support. Lastly, I offer my regards and blessings to all those who supported me in any respect during the completion of projects.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview and Motivation

Sensors are tiny devices that observe real-world environmental conditions and convert measurement into signals which can be received by other instruments. They come in different kinds and observe conditions such as motion, heat, pressure or electricity. Nowadays sensor technology is evolving fast: new devices are being invented to record more conditions, precision of detection is being improved, energy efficiency is being ameliorated and sensors are becoming less expensive.

Taking advantage of sensor technology, spatially distributed sensors can be embedded into sensor nodes to form a wireless sensor network (WSN) for cooperatively monitoring environmental conditions. Typically each sensor node consists of a radio transceiver, a micro-controller, an energy source and one or more sensors. With limited number of gateways, WSN can collect sensor readings from nodes.

Potentially, WSNs can be deployed across various spaces and provide data of interest for specific areas or objects. They have found applications in to disaster surveillance, industrial control, civilian monitoring and many other fields. Gradually, they are becoming the ubiquitous information infrastructure of our society.

Although many applications have been built based on WSNs, several basic issues remain unresolved in terms of WSN design and reusability. Among the most important issues is the issue of coupling between applications and WSNs: due to the underlying diversity and complexity of todays WSN technologies, many applications are developed for a particular WSN and are difficult to integrate with new WSNs. Under this circumstance, even the slightest change of network might result in considerable reengineering of the application software. And when essentially the same functionalities need to be implemented for different sensor network applications, little code can be reused. The problem is exacerbated when multiple types of WSNs need to be integrated for a sophisticated task. From the perspective of software development and maintenance, the network-specified approach is not cost-effective and has become the bottleneck for uniting the pervasive WSNs.

In the light of this problem, a middleware is necessary to provide logical abstraction of WSN so that all the low level details of WSN such as routing protocol, operating system of micro-controllers are hidden while the generic, common properties of WSN are preserved. The vision is to have a service API layer working as this piece of middleware. On one hand the service API layer negotiates with WSN and sets up protocol for collecting and interpreting sensor data; on the other, it archives and stores data for user retrieval. The service API layer is supposed to offer extensive application user interfaces (APIs) to facilitate sensor data query and if possible, integrate data fusion components that provide high level answers to questions such as localization. In other words, the service API layer suggests a 3-tier architecture for WSN based applications as shown in Figure 1.1. Based on service API layer, applications can be built with ease and developers can concentrate on data manipulation without worrying about network details.

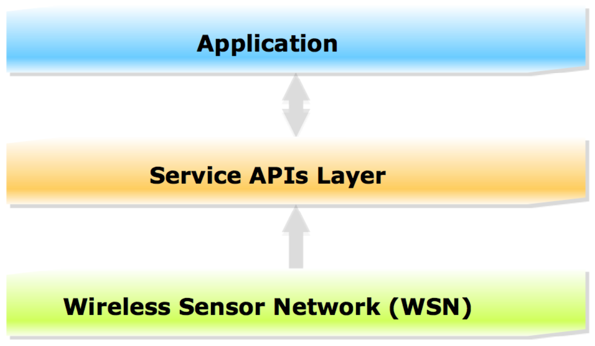


Figure 1.1: The 3-tier Architecture

Up till now, substantial efforts have been devoted into developing applications such as monitoring systems under the 3-tier framework. Among them are Microsoft Sensor Map citeRefSM, Pachube [21], GT Aware Home [1] and Gator Tech Smart House [5]. Microsoft Sensor Map and Pachube mainly aim at outdoor environment monitoring while the other two focus on indoor monitoring and assisted living. Each of them has embedded a flexible service API layer and can accommodate networks changes easily. These applications offer rich build-in APIs and excel in the targeted domain.

Though the four systems mentioned above are powerful, each of them suffers from several limitations. Microsoft Sensor Map and Pachube only supply raw sensor data but do not allow users to answer general high-level questions based on this data. When real-time monitoring or data fusion is necessary this could be a problem. For instance, if the WSN based application needs to monitor and visualize a car moving along the highway additional programming effort has to be put into developing module that infers the car position from raw sensor readings. Compared to having a smart service API layer performing data fusion at the server side and giving the location coordinates of the car directly, this approach fetches large volume of raw sensor readings and increases the network burden significantly. For GT Aware Home and Gator Tech Smart House, data have to be accessed through special protocols. There is no way to request data using general methods like posting a URL link, which infers special client module has to be embedded in WSN based application.

Building on previous work, we have developed an integrated monitoring system based on service APIs. The objectives of this thesis to

1. construct an environment monitoring infrastructure that utilizes WSN and manages collected sensor data effectively;
2. visualize comprehensive, real-time sensor data in both 2D and 3D views, for outdoor and indoor sensor deployments respectively;
3. develop a toolkit which uses potentially or partly available WSN data to help WSN design, deployment and optimization;
4. provide easy-to-use APIs and accessible sensor data to other applications;
5. integrate data fusion components for answering some of the high level questions (currently only localization is implemented).

1.2 The Service-Oriented Architecture Paradigm and its Styles

Service-Oriented architecture is in essence distributed computing using a collection of services. Each of the services modularizes a business logic or individual function and different services can exchange data. Since the services are loosely coupled, applications can compose one to many services without worrying about the underlying complexities.

Currently there are two different styles of implementing service-oriented architecture: Web Service Description Language (WSDL) and Representational State Transfer (REST). WSDL describes a web service by using an XML document that contains information such as data types, possible operations, messaging protocol and etc. REST focuses on the concept of resources and regards every URL as a resource. It finishes the transition from one state of the resources to another by accessing a different URL.

Both WSDL and REST make use of XML technology. While WSDL can use various messaging protocol, REST is limited to HTTP. However, compared to WSDL, REST requires fewer infrastructures and allows caching.

1.3 Software Architecture

The system developed in the context of this thesis consists of five major components as shown in Figure 1.2:

- WSN module
- Service API module

- 2D visualization module
- 3D visualization module
- Wiki-based collaboration module

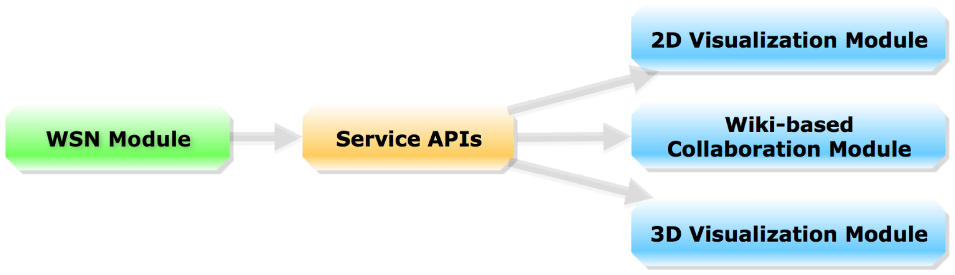


Figure 1.2: Major Modules

The WSN module includes all the sensors and the software that control firmware behaviors. The service API module supports flexible integration of various data streams from WSNs and provides well-designed APIs [1] to fit application-specified needs. All the other modules, except for the WSN module, are built on top of service API module. The 2D visualization module presents sensor data under the context of a multi-layer map and private data overlays. The wiki-based collaboration module works as a platform for sensor-related information exchange. The 2D visualization module and the wiki-based collaboration have the ability to cross-reference each other so that information shown in these two modules can be associated. The 3D module uses virtual reality to represent the interior environment. The most important usage of this module is to undertake near-real-time, privacy-aware monitoring for healthcare purposes.

1.4 Contributions

This thesis makes four contributions to the state-of-the-art in the area of sensor network applications.

First it proposes a REST style service-oriented architecture for applications relying on sensor network data management. The service-orientation paradigm ensures the loose coupling of the application services, and eases their reuse and extendibility to meet application-specific requirements. The architecture enables the integration of multiple wireless sensor networks in the same application, and through a well-designed set of REST APIs, the underlying complexities of accessing the network data are transparent to the application developers. The flexibility of the architecture is demonstrated through the development of two different applications on top of it: SensorGIS and SmartCondo.

Second, it incorporates a 2D multi-layer map view interface that can be accessed on web (not tight to heavy weight client) and works with various public map providers such as Google, NASA,

[1] APIs that are easy to use learn, use even without documentation and are easy to extend.

Yahoo, Microsoft, etc. The map view interface is able to handle multiple map layers at the same time and display private data on overlays. Besides, functionalities are embedded in the map interface so that it can communicate with integrated wiki and provide cross-referencing between sensors on the map and entries in the wiki. The wiki integration and cross-referencing feature work together and make it possible for different researchers to access and share related information efficiently.

Third this thesis proposes a localization algorithm based on overlapping of sensor detection zones. The algorithm is capable of analyzing the incoming sensor readings stream and producing numeric location result (latitude and longitude). It is well designed and offers reasonable accuracy [2] even if radio interference is present and sensor reading transmissions are affected. A service is built to provide localization results generated by the algorithm. Using this service, the SmartCondo application is able to reproduce the activities of the condo occupant. Moreover the SmartCondo application is excellent for online healthcare training because it is accessible on web and supports playback feature.

Fourth this thesis proposes an algorithm to optimize sensor network topology. A toolkit with a graphical user interface (GUI) is built. The toolkit supports easy import and export of network topology and is able to optimize the topology automatically based on user-given parameters. The algorithm defines a set of metrics for optimization and reduces substantially the time needed to optimize sensor network topology. It is especially helpful in improving topology of large sensor network and can serve as the first step for further manual tuning.

1.5 Organization

This thesis is arranged as follows. Chapter 1 introduces the concept of wireless sensor network and gives an overview of the functionalities and architecture of the monitoring system. Chapter 2 explains in detail how different pieces of components in the architecture work together. Chapter 3 discusses how SensorGIS orchestrates various modules and tools to provide web-based monitoring information. Chapter 4 visits every important component in SmartCondo and sheds light on their working mechanisms. Chapter 5 discusses the algorithm used for localization. Section 6 discusses related research, and Chapter 7 lists the potential improvement and future work.

[2] Most of the time match the movements in real world.

Chapter 2

Related Work

As our system has been designed for integration with interior and outdoor environments, with clients a 3D virtual world and a GIS correspondingly, we review both types of applications.

2.1 Integration of GIS with WSNs

2.1.1 SensorMap

SensorMap is a sensor network visualization web service developed by Microsoft. It visualizes real time data generated by sensors deployed across the globe. SensorMap displays sensor data on a Microsoft Virtual Earth Map and provides user with the most recent data recorded by the sensors, as well as live video feeds.

The project SensorMap consists of three major modules: data publishing module, data gathering module and front-end web interface module. The first module allows sensors to register themselves with SensorMap. By applying this module an agreement is established between the sensor and SensorMap on details of communication such as URL where SensorMap can find the sensor data, sensor identity, sensor type, data format including measuring unit, etc. The second module collects the most recent sensor data according to the URL negotiated by data publishing module. After data collection, all the data is stored in GeoDB database. The third module manipulates the available data and is responsible for map visualization.

One of the disadvantages of SensorMap is that it is state-centric instead of sensor-centric. Oftentimes multiple sensors of different types can be installed on a single sensor node to measure the surrounding environment changes. SensorMap is unaware of this possibility and assumes that every node is a specific sensor such as temperature sensor or precipitation sensor. In contrast, a sensor-centric system can handle nodes with more than one sensor. Another drawback of SensorMap is that its analysis tools are quite limited. Neither does it support real-time monitoring or user feedback sharing.

2.1.2 Viewlon

Unlike SensorMap, Viewlon [16] is not a web service. It has a unique feature of adopting a rich ontology for representing the sensor network and the data it records. For instance, it displays sensor relationships such as gateway nodes and regular nodes alongside with how their sensed data are correlated. Nevertheless, Viewlon visualizes sensor network as graphs so it becomes difficult to associate the sensors with their sensed data geographically. Besides Viewlon is not easily extendible because it is associated with a specific sensor network.

2.1.3 Mote-View

Similar to Viewlon, Mote-View [22] is not a web service but it is designed in a client-server architecture style where the Graphic User Interface (GUI) runs on the client side. Unlike SensorMap it is sensor-centric and supports useful queries like getting all the readings from a subset of sensors and readings within a given time period. Furthermore, features to determine the health of sensors are included. Mote-View also represents sensor topology as a graph instead of a map. The major drawback of Mote-View is its dependency on Mica sensor network hardware. Supporting only 9 types of sensor network hardware platforms, it cannot be applied for broader usages.

2.1.4 SNAMP

The Sensor Network Analysis and Management Platform (SNAMP) [23] is a novel multi-sniffer and multi-view visualization platform for pre-deployed indoor wireless sensor network. It consists of 3 major functional modules: sensor network module, data collection network module and visualization module. The network module is built on top of the GAINS-4a sensor nodes. In the data collection module, GAINS-3 sensor nodes work as sniffers and are responsible for collecting various types of information from the running sensor network. By setting up this 2-layer sensor infrastructure, data bottleneck problems can be avoided while at the same time a complete viewer of network activities is preserved. Based on the assumption that multiple kinds of information in wireless sensor network need to be observed and analyzed, a multi-view visualization module is integrated. The visualization includes 5 back-end modules: data dispatcher, topology analysis module, sensing data module, packets analysis module and network measurement module, each of them measuring one type of performance statistics. The front-end part of visualization module can be divided into 4 sub-modules: topology view, sensing chart view, packets view and measurement view. Using SNAMP, developers can monitor the overall performance of wireless sensor network without causing any interference of its activities. Moreover, features such as network activities replay and packet view break point are embedded so that SNAMP can facilitate debugging of sensor network as well. Although SNAMP architecture is flexible, it is still in its baby age. Only indoor version is developed and it is unable to handle outdoor sensor network or network with mobile sensors.

2.1.5 SpyGlass

The SpyGlass project [2] is a modular and extensible visualization framework for wireless sensor networks. Data emitted by individual sensor nodes are collected by gateway software running on a machine in the sensor network. It is then passed on via TCP/IP to the visualization software on a potentially remote machine. SpyGlass is based on Java technology and supports extendible visualization plug-ins such as temperature map plug-in, node painter plug-in, battery plug-in, topology plug-in and position plug-in. The final visualization is displayed on a Java desktop application with a 2D interface. SpyGlass supports network playback thus can facilitate network history monitoring.

2.2 WSNs in Interior Spaces

2.2.1 GTA Aware Home

Several efforts have been devoted to develop a smart home and among them Georgia Techs Aware Home is the most well-known. The Aware Home project is devoted to multidisciplinary exploration of emerging technologies and services based in the home. It seeks to provide services to the homes residents and try to enhance their quality of life or help them to maintain independence as they age. A variety of distinct research activities have been carried out in the context of this project, mostly focusing on usability concerns around a smart space. For example Fetch [9] assists visually impaired people to locate misplaced objects and Cooks Collage [6] assists seniors in following recipes. Most likely, the activity closest to our SmartCondo is the Power Line Positioning (PLP) project [10]. PLP analyzes the frequencies of signals sent over power lines to infer the location of electronic devices within the home. In effect, special modules located at each end of the house serve as sensors for localizing domestic robots (such as the Roomba) and potentially finding missing items, for example keys and wallets.

2.2.2 MavHome

The MavHome project at the University of Texas at Arlington [4] takes a more active approach to helping the smart home occupants. It uses sensors (light, temperature, humidity, motion and door/seat status sensors) to monitor the state of environment and analyzes the collected data to

1. identify lifestyle trends, through sequential pattern mining
2. provide reminders to the home occupants, through prediction of future activities
3. detect anomalies in the current data, when the actual sensed events are considered unlikely according to the systems predictions

MavHomes power line control automates all lights and appliances as well as HVAC, fans and mini blinds. Perception of light, humidity, temperature, smoke, gas, motion and switch settings is performed through a sensor network developed in house.

2.2.3 Gator Tech Smart House

The Gator Tech Smart House at the University of Florida is yet another high-tech house, currently under construction. In this house a variety of sensors will be embedded to assist elderly occupants or patients suffering from diabetes and obesity with the behavioral monitoring (and alteration).

2.2.4 AAL

The ambient Assisted Living (AAL) Laboratory [12] [20] developed by the Fraunhofer Institute for Experimental Software Engineering in Germany is an apartment-like environment for developing, integrating and analyzing ambient intelligence technologies. Currently, the AAL Lab supports the following specific scenarios:

1. monitored drinking via a computerized cup
2. monitored food quality via an RFID system built into the refrigerator
3. item location tracking through motion sensors and RFID tags
4. fall detection

2.2.5 SensorRAUM

The SensorRAUM system [14] seeks to investigate, define, develop and demonstrate user friendly and intuitive user interfaces for wireless sensor networks. The SensorRAUM visualization is based on the Open Croquet [3] virtual world. To project the state of real-world objects into the virtual world, they propose to integrate sensors into objects such as doors, clothes and furniture. Thus, these objects are turned into soft media devices which can communicate with other such devices throughout the environment.

2.2.6 WASP

Although not conceived in the context of a smart home, the WASP architecture [8] focuses on the software infrastructure necessary for effectively integrating a population of wireless sensors to recognize events in a living environment and provide aural feedback. The system requires that the occupant wear active radio-frequency identification (RFID) tag to help localization tasks and uses acceleration sensors to detect doors opening and closing.

2.2.7 Sensorized Elderly Home

The Sensorized Elderly Care Home [11] is a system installed in a nursing home in Tokyo. This work is motivated by the desire to alleviate the routine workload of nursing personnel through automation. In particular, the paper proposes a sensor-based system for localizing patents in a nursing home, monitoring their status and raising alarms as necessary so that nurses do not have to do routine rounds. The system assumes a relatively limited level of activity on the part of the patients. To monitor wheelchair movement, it relies on Ultra Badge transmitters placed on wheelchairs and receivers places in several locations in the nursing home. Furthermore a specially designed placement of transmitters and receivers on the ceiling is meant to monitor the patients head position on and around the bed. This latter functionality is not completely evaluated and in place.

2.3 SensorML

Besides the applications for outdoor WSNs and indoor WSNs, one important related work is the OGC Sensor Model Language (SensorML) [17]. SensorML specifies models and XML encoding that provides a framwork within which the geometric, dynamic, and observational characteristics of sensors and sensor systems can be defined. The SensorML definition supports atomic process models and process chains. All processes and component are encoded as application schema of the Feature model in GML.

2.4 Problems in Existing Applications

In summary, most of the applications discussed above are tight to the network they use and have difficulty extending themselves to new networks. Due to the inherent differences [1] between outdoor WSN and indoor WSN, few application could deal with both of them, not to mention providing unified APIs to manipulate data coming from these two types of networks. Nevertheless, few of the existing applications incorporate the mechanism to expose data services they could provide. Also most of the existing applications are using heavy client so it forces users to install the heavy client. This approach rises the problem of platform compatibility and client version synchronizing.

Table 2.1: Comparison between WSNs in Interior Spaces (Architecture)

	Architecture
GT Aware Home	Software infrastructure to assist with rapid development
MavHome	CORBA
Gator Tech Smart House	Open Services Gateway initiative framework
AAL	Service-oriented architecture
SensorRAUM	Special-purpose, likely driven by Open Croquet
WASP	Service-oriented architecture
Sensorized Elderly Care Home	Not mentioned

[1] Sizes of detection areas, possibility of having detection zone overlap

Table 2.2: Comparison between WSNs in Interior Spaces (User Interface)

	User Interface
GT Aware Home	Not mentioned
MavHome	Specialized interface agent
Gator Tech Smart House	Not mentioned
AAL	Dynamically-rendered UI - likely form based Audio and visual devices for multimodal interaction
SensorRAUM	Open Croquet
WASP	Not mentioned
Sensorized Elderly Care Home	Not mentioned

Table 2.3: Comparison between WSNs in Interior Spaces (Sensors)

	Sensors
GT Aware Home	● Video ● Ultrasonic ● Floor sensors
MavHome	● Door ● Light ● Motion ● Humidity ● Seat status ● Temperature
Gator Tech Smart House	● Smoke detectors ● Security-system motion detectors
AAL	● RFID ● Intelligent appliances (e.g., fridge, cups) ● Vital sensors (pulse, skin temp., skin humid.) ● Ultrasonic and radio-frequency-based motion sensors ● EIB-based home automation for switches, blinds, and power sockets
SensorRAUM	● Temperature sensors ● Ambient light sensors ● Coffee cup with temperature, orientation and switch sensors
WASP	● Wearable sensors including arm band, waist, shoe, and ear-worn sensors ● Ambient sensors including microphones, pressure sensors, RFID tags, electricity and water usage sensors, blob sensor
Sensorized Elderly Care Home	● Ultrasonic sensor (Ultra Badge)

Table 2.4: Comparison between WSNs in Interior Spaces (Analysis / Functionality)

	Analysis / Functionality
GT Aware Home	• Locates/identifies a person • Locates lost objects • Provides several distinct interactions with the home
MavHome	• Detects behavioral patterns via sequential data mining (ED) • Makes recommendations based on prediction (ALZ) • Automates repetitive tasks
Gator Tech Smart House	• Senses the state of both home and resident • Provides remote monitoring and intervention services
AAL	• Monitors daily behavior • Builds histories and medical/activity patterns for the elderly • Assists people in maintaining their well-known daily routine • Recognizes emergency situations (from vital data) • Provides remote care and information systems for relatives and care personal • Incorporates an interactive TV-based video-telephony system • Provides an autonomous transportation platform: a robotic unit for emergency recognition, multimedia interaction, and transportation assistance
SensorRAUM	• Queries the current state (e.g., number of cups in the room) • Uses non-explicit addressing/identification of communication partners • Allows devices to detect their geometric location via a location system
WASP	• Incorporates intelligent data analysis to infer data from multiple sensors, detect patterns across datasets, and identify risks • Uses multi-sensor fusion to obtain better classification rates and decrease ambiguity between activities
Sensorized Elderly Care Home	• Localizes head (when in/around bed) and wheel-chairs • Provides remote monitoring of elderly people • Detects accidents and notifies caregiver

Chapter 3

The SOA Framework for WSN-Based Applications

In this chapter, the service oriented architecture is explored. The motivation for choosing the REST style to construct the SOA architecture is discussed in section 3.1 and the detailed implementation is explained in section 3.2. Current available services are listed in section 3.3.

3.1 REST Style Architecture for WSN-based Applications

To manage sensor network data, two fundamental tasks must be fulfilled. First, data must be collected and stored. Second, the stored data must be accessed and manipulated. However the sensor networks, the management application providing data access APIs and applications built on top of it may be implemented using different technologies. What's more, computing will be performed in a distributed fashion. Raw, unparsed data are first collected by the gateway node in the sensor networks and due to limited storage and computation ability, the parsed data generated by these nodes has to be forwarded to repository hosted on computationally more capable server. The management application and other applications feeding on the managed data may also reside on separate machines. In order to integrate various distributed components, implemented in distinct programming languages and built on potentially different platforms, the REST SOA style is chosen. By encapsulating the management functionalities into loosely coupled services, differently applications can access the services via network and exchange data easily, without worrying about platform, programming language and application-specified technologies.

There are quite a few options for SOA architecture. Among them are SOAP, REST, RPC and CORBA. Each of them has different advantages and fits well for specific purposes. For example, SOAP is language independent and allows the use of different transport protocols. However it is slower compared to CORBA and the support for some of the languages is weak. RPC is good for client-server model and distributed computing nevertheless has to deal with unpredictable network problems which result in failure of procedure invocation. Because sensor data management is inherently data-centric, we believe REST style is more appropriate. Compare to other styles, REST supports caching of representations thus reduces server load and improves response time. What's more, REST style does not require the server to remember session state therefore better server scalability can be achieved by introducing multiple servers to handle different requests in a session, if necessary. In terms of accessibility, REST is able to provide easy HTTP access to services and thus can fit both thin clients such as web browser as well as thick clients.

3.2 Implementation

Figure 3.1 diagrammatically depicts how the SOA architecture is built. The most important component in REST is resources. All of them are stored in databases. Currently MySQL is used for database application and potentially more powerful database products such as Oracle and DB2 can be introduced. In the MySQL databases, there are mainly two types of resources. The first type of resource is the parsed readings collected from the wireless sensor network and the metadata about network topology. This type of resource is generated by the portal. The portal, which connects to the master node of wireless sensor network, is actually a piece of java code that validates incoming readings, populates readings into database and prepares copies of readings for the stream miner. The second type of resource is generated by the stream miner. While the stream miner keeps receiving copies of readings from the portal, it analyzes the readings and tries to infer useful information such as the location of patient moving in a condo. The stream mining mechanism will be further discussed in Chapter 5. Note that while portal is essential for every sensor network, stream miner is not. When new network is introduced into the framework, custom stream miner could be added to dig out network-specified information. In this case, more resources will be stored in the database.

The entity-relationship diagram for the first type of resource is shown in Figure 3.2. The entities are denoted by squares and the relationships diamonds. The ellipses with underline are key attributes. The ellipses without underline are normal attributes. Entity SENSOR corresponds to physical sensor nodes. Each sensor node is equipped with one to multiple sensors and detects one to multiple environmental states. In the case multiple sensors are mounted on the same sensor node, multiple tuples of (SENSOR_ID, STATE_ID) with the same SENSOR_ID will appear in table SENSOR. The entity STATES corresponds to the environmental states which the sensors are measuring. Potentially more than one sensor will be recording the same type of environmental state, for instance temperature. However, what they measure simply correspond to the same tuple in STATES table. Attribute State_ID is the key of STATES table and uniquely identify a specific type of environmental state. The valid range of readings and description of state are also stored in the STATES table. Often location information is necessary to give geography context of the collected sensor data therefore an entity called LOCATION is introduced. Besides attributes X, Y and Z denoting latitude, longitude and altitude of sensor node, a timestamp is also needed. An attribute named Loc_Tstamp is created to handle sensor mobility. When a sensor node is moved to a new location, a row with

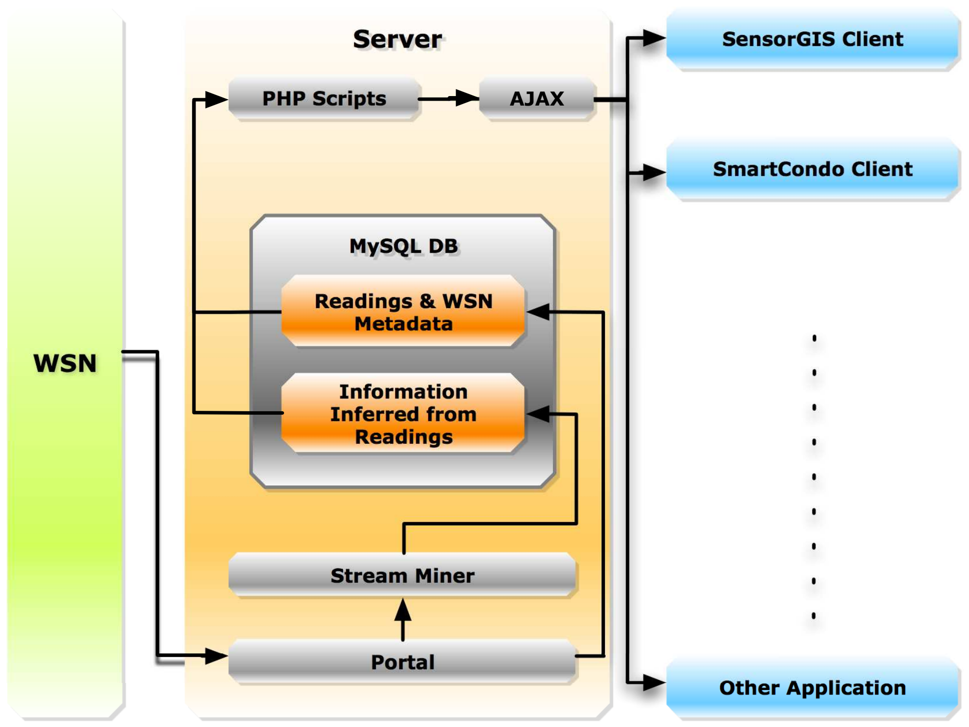


Figure 3.1: SOA Implementation

new timestamp and coordinate will be inserted into LOCATION table. In this way, the history of sensor movement is made traceable. Nevertheless, LOCATION entity is not enough to represent the network topology. A sensor node in the wireless network may either be an ordinary node or a gateway node. The gateway node is able to collect readings from the ordinary node in its vicinity so in terms of network management, they are inherently different and have to be treated separately. Entity ROLES and SENSOR_TYPE are brought in to identify the role of sensor node. The SENSOR_TYPE table stores descriptions and unique identifiers for the 2 different roles of sensor nodes: ordinary and master. And the ROLES table gives a mapping of which sensor node plays which type of role. The most dynamic table in this schema is OBSERVATION. It holds all the parsed readings collected from sensor networks together with the ownership information of the readings.

The entity-relationship diagram for the second type of resource is shown in Figure 3.3. It is comparatively simple and only has one entity called ACTION. The attribute Event_Type and Tstamp work as compound primary key and solely identify the action of the person being monitored at a specific point of time. Coordinate information is generated by the stream miner.

In the REST style architecture, all the resources can be referenced by a URL. A set of PHP scripts are developed to handle user requests. Once the scripts receive an URL, they will extract parameters from the URL and recognize which resource is being requested and what kind of operation should be done. Once result is ready, the scripts will encapsulate the result in a XML file and sends it back

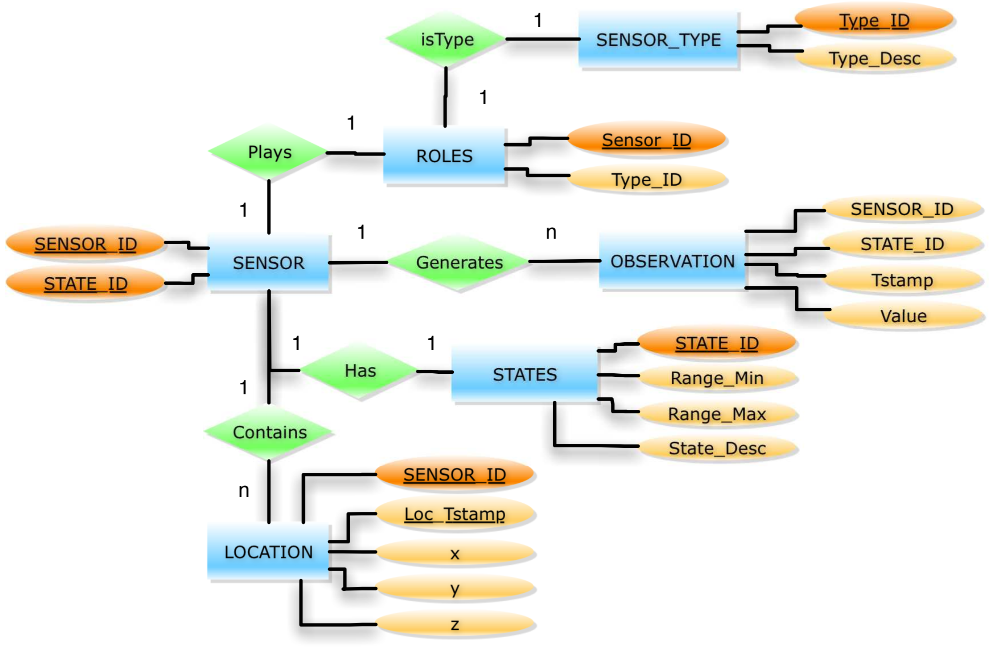


Figure 3.2: ER Diagram for Readings and WSN Metadata

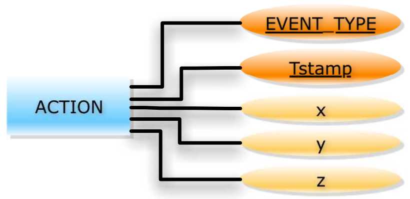


Figure 3.3: ER Diagram for Information Inferred from Readings

to client. In this way, the client and server can communicate using AJAX thus improve performance of interaction.

3.3 Available Services

The services currently enabled are as followed. The first three services can be used to change sensor network topology while the rest can be used to retrieve or insert or reset data of interest.

3.3.1 WSN Management

Add Sensor

This service takes input of an URL which contains a sensor ID and a location. If the sensor being added is not yet deployed in the field then it is initialized as an inactive sensor. Otherwise all the readings of that sensor will be marked accessible in the database.

Move Sensor

This service takes input of an URL which contains a sensor ID and a location. It accommodates the need of moving an existing sensor. Upon invocation of service, the existing sensor will be assigned a new location but all the state values measured previously are still associated with the sensor itself.

Delete Sensor

This service takes input of an URL which contains a sensor location. It is often used together with the first service to manage the sensor network topology. Upon deletion, all the state values measured by the deleted sensor will be marked unavailable.

Set State Value for Sensor

This service takes input of an URL which has three parameters: location, state name and a state value. Usually readings are not supposed to be added by users. However this service is necessary when technicians need to reset the state value(s) of a sensor under certain circumstances. For example, a switch sensor attached to a trap that catches wild animal can report the trap door closed event but it is not able to generate reading saying the trap is reset after the biologist release the animal from the trap. In this case, the biologist can use the service and reset the status of the switch sensor.

3.3.2 Information Retrieval

Get Locations and Roles of all Sensors

This service takes an URL with no user-defined parameter. It aims to provide an overview of sensor network topology. Visualizations in SensorGIS and SmartCondo use this service when startup.

Get State Value of a Specific Type

This service takes input of an URL containing location(s) and state name. The result returned by this service is the latest state values of the sensors located at the given locations. The main purpose of this service is to provide state value comparison among different sensors. However it can also be applied to single sensor. In the case of single sensor, the service returns the latest value of the given state measured by the sensor.

Get Statistics for a Specific Type of State

This service requires parameters of at least two locations, a state name and a time period. Once state type and time period is determined, the service will query for historic data and calculate for each sensor in the group statistics such as minimum value, maximum value, standard deviation, average. Besides statistics for individual sensor, statistics for the sensor group will also be computed and returned.

Get History Values for a Specific Type of State

This service requests an URL with a location and a state name and a time period. It returns all the state values lie within the given time period for a single sensor. Historic readings of a sensor group can be retrieved by using this service multiple times.

Get all State Values Exceeding User-defined Limits

This service requires 3 parameters in the URL: operation, upper bound and lower bound. The possible operations are: smaller than, smaller than or equal to, larger than, larger than or equal to, equal to, larger than A and smaller than $B(A < B)$. If only 1 bounding value is needed for the operation then upper bound and lower bound will be the same. By invoking this service in iteration, any value with the latest timestamp failing the user defined assertion will be returned. This service is designed for near-real-time monitoring.

3.3.3 Information Fusion

Get Action

This service asks for parameter timestamp. It utilizes the second type of resource, which is generated by the stream miner, and returns the location alongside with action of the person being monitored in a condo. The action can be one of the following: move around, open door, close door, sit down and get up. This service is the key for SmartCondo visualization. The avatar in virtual reality will make movement according to the returned result.

Chapter 4

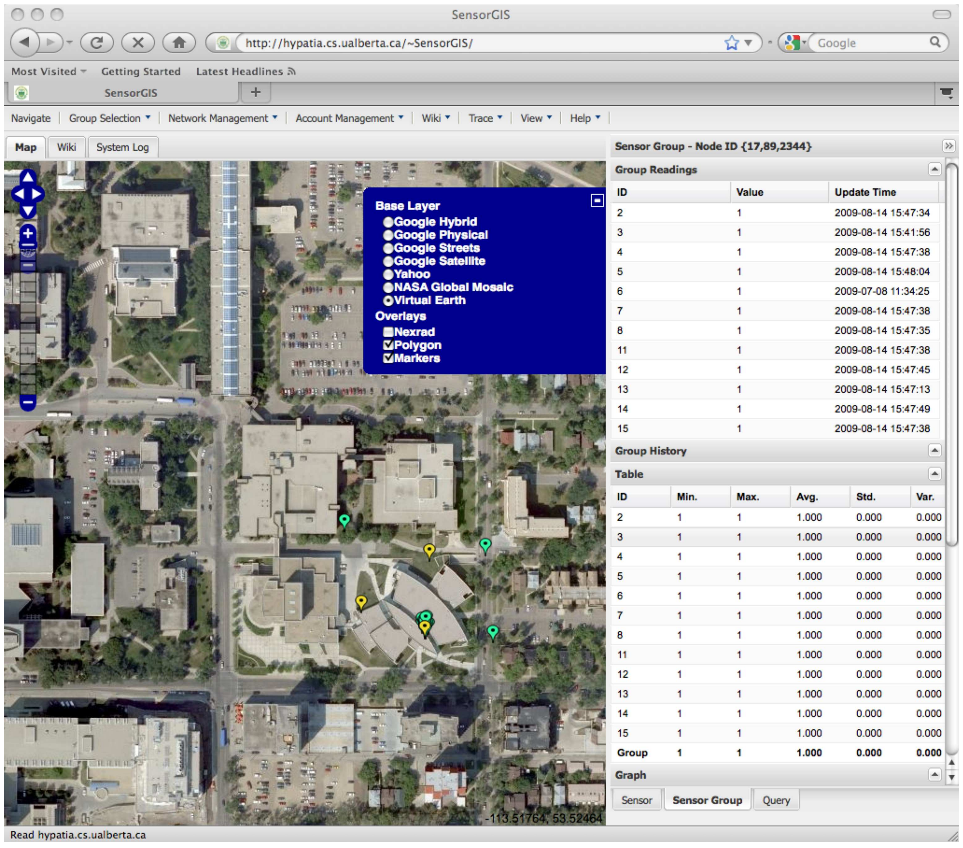
SensorGIS

SensorGIS is a monitoring application designed to enable the collaboration of people interested in information collected from a widespread, outdoor wireless sensor network. It takes advantage of the services offered by the service API module and provides an intuitive geospatial model in which end users can explore the network easily. In the geospatial model a multi-layer map is integrated, using OpenLayers [18]. Compared to maps in other geospatial applications, it is able to switch back and forth among different base maps on the fly thus giving more appropriate information about the context of sensed data. For example, users can choose a Google street map if traffic monitoring is desired, or switch to NASA global map if they are interested in precipitation monitoring done. Besides switching among based maps, private data owned by end users can be loaded and displayed as overlays. This feature is useful when custom view is needed such as the coverage of a hurricane approaching coastline. What's more, a selection tool is embedded. By applying this tool end users can select random sensors as a group and compare historic readings within the group in graphs and tables.

Although exploring sensor data is made convenient by the map, still it might be insufficient to draw a full picture of what's happening within the sensor network coverage. Therefore a wiki is integrated. The wiki works as a platform and communication of sensor-related information can be done on it. Both the wiki and the map can cross reference with other so the end user can quickly associate entries in wiki with actual sensors on map.

4.1 Architecture and Implementation

Built on top of service API module, SensorGIS follow typical REST style architecture. The web browser works as front-end client and takes care of representation of different states. Two different views are included in the interface: map and wiki. The map is built based on open source tool called OpenLayers. With this tool, users can switch to various maps whenever necessary and incorporate private maps. Sensors are shown as markers on the map and user can see the historic data in graph and statistics in table as shown in Figure 4.1. As for the other view, wiki, it is integrated so that



Sensor Group - Node ID {17,89,2344}

Group Readings

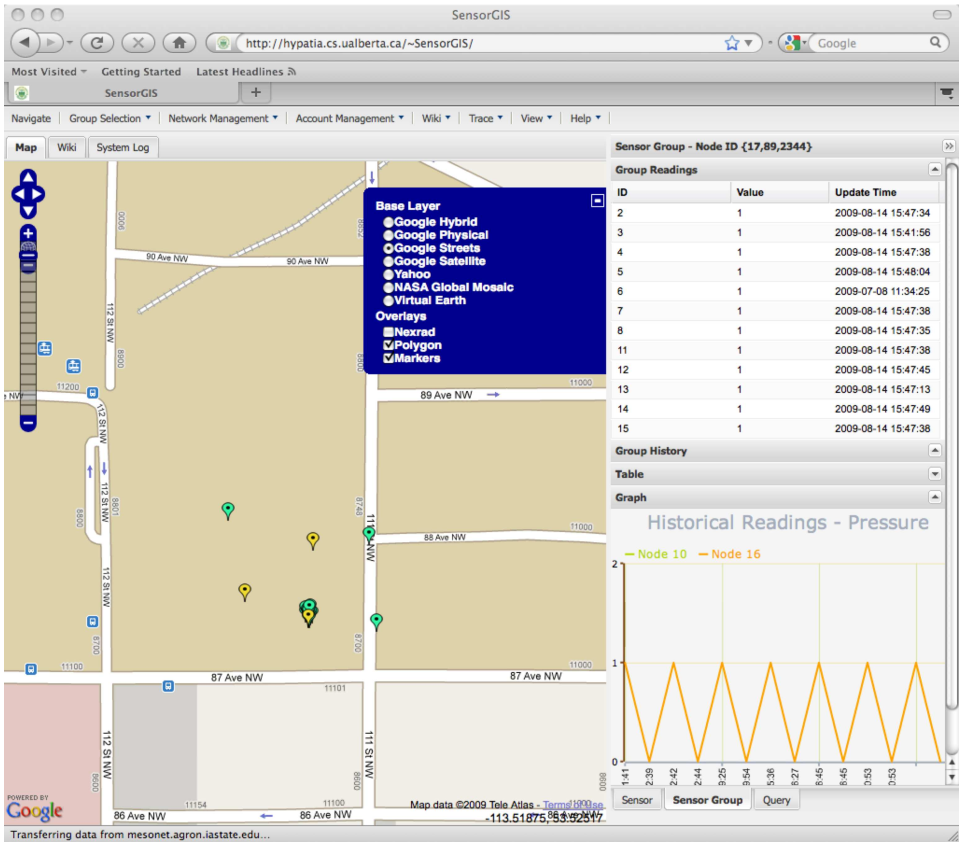
ID	Value	Update Time
2	1	2009-08-14 15:47:34
3	1	2009-08-14 15:41:56
4	1	2009-08-14 15:47:38
5	1	2009-08-14 15:48:04
6	1	2009-07-08 11:34:25
7	1	2009-08-14 15:47:38
8	1	2009-08-14 15:47:35
11	1	2009-08-14 15:47:38
12	1	2009-08-14 15:47:45
13	1	2009-08-14 15:47:13
14	1	2009-08-14 15:47:49
15	1	2009-08-14 15:47:38

Group History

Table

ID	Min.	Max.	Avg.	Std.	Var.
2	1	1	1.000	0.000	0.000
3	1	1	1.000	0.000	0.000
4	1	1	1.000	0.000	0.000
5	1	1	1.000	0.000	0.000
6	1	1	1.000	0.000	0.000
7	1	1	1.000	0.000	0.000
8	1	1	1.000	0.000	0.000
11	1	1	1.000	0.000	0.000
12	1	1	1.000	0.000	0.000
13	1	1	1.000	0.000	0.000
14	1	1	1.000	0.000	0.000
15	1	1	1.000	0.000	0.000
Group	**1**	**1**	**1.000**	**0.000**	**0.000**

Graph



Sensor Group - Node ID {17,89,2344}

Group Readings

ID	Value	Update Time
2	1	2009-08-14 15:47:34
3	1	2009-08-14 15:41:56
4	1	2009-08-14 15:47:38
5	1	2009-08-14 15:48:04
6	1	2009-07-08 11:34:25
7	1	2009-08-14 15:47:38
8	1	2009-08-14 15:47:35
11	1	2009-08-14 15:47:38
12	1	2009-08-14 15:47:45
13	1	2009-08-14 15:47:13
14	1	2009-08-14 15:47:49
15	1	2009-08-14 15:47:38

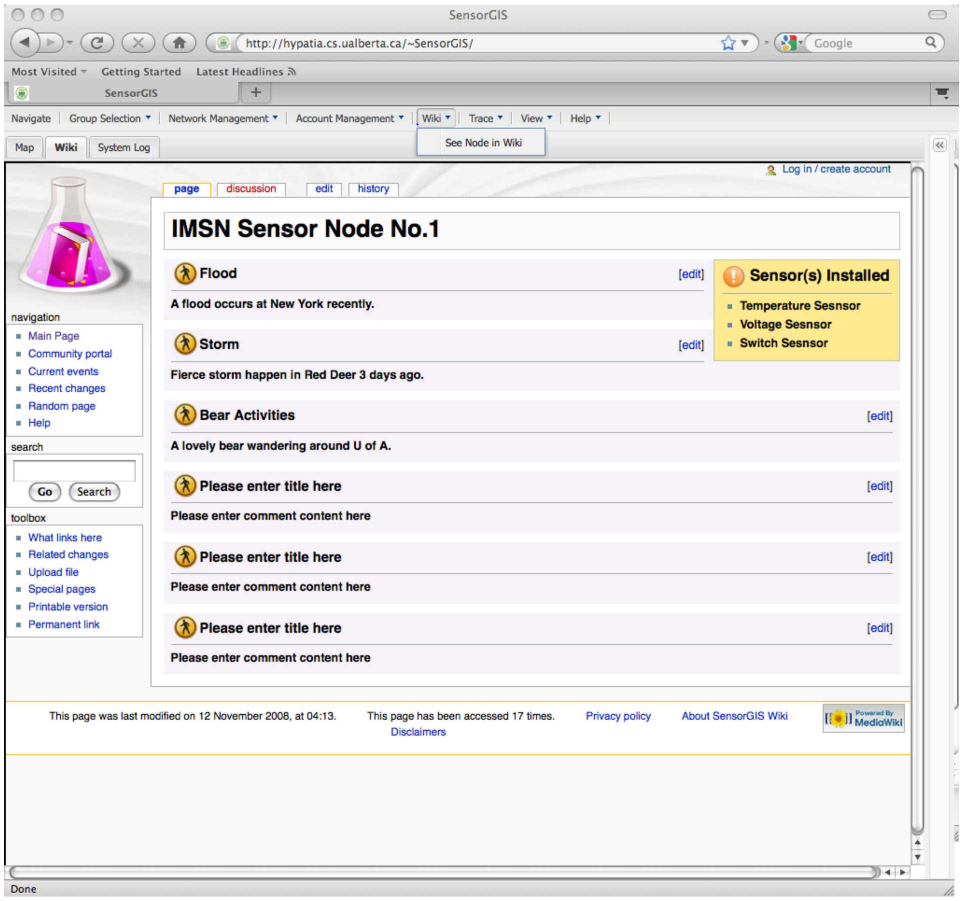
Group History

Table

Graph

Figure 4.1: The Map View in SensorGIS

different researchers can share information and raise discussion about collected data. The wiki integrated is based on MediaWiki [15] and is modified to achieve the cross-reference function. Entries in the wiki are associated with sensors on map so that user can see the sensor by a single click in the interface. On the other hand, user can select sensors and create entries in wiki. The selected sensors will be linked to the wiki entries automatically. To develop JavaScript compatible with different web browsers, EXTJS [7] project is used. When user performs an operation in the web browser, EXTJS invokes browser specified JavaScript codes which in turn performs visualization or issue corresponding request to service API module.



IMSN Sensor Node No.1

Flood [edit]

A flood occurs at New York recently.

Storm [edit]

Fierce storm happen in Red Deer 3 days ago.

Sensor(s) Installed

- Temperature Sesnsor
- Voltage Sesnsor
- Switch Sesnsor

Bear Activities [edit]

A lovely bear wandering around U of A.

Please enter title here [edit]

Please enter comment content here

Please enter title here [edit]

Please enter comment content here

Please enter title here [edit]

Please enter comment content here

This page was last modified on 12 November 2008, at 04:13. This page has been accessed 17 times. Privacy policy About SensorGIS Wiki Disclaimers

Figure 4.2: The Wiki View in SensorGIS

On the server side, service API module is responsible for interpreting requests and retrieving information out of database. Database query result generated by service API is packed as xml by default therefore fulfilling the AJAX contract, which reduces the amount of html content needed to be sent back and at the same time significantly improves the responding time of web interface. Besides xml, the result can also be encapsulated into other formats. A separate component dealing with result format is embedded in the service API module. By providing a proper URL, the query result can be visualized in various ways. What's more, not only the web interface but also other desktop applications can query sensor-related information simply by referencing the same URL in the web browser. To achieve this feature, we decide to use HTTP GET method for posting user request instead of the more secure HTTP POST method because in the POST scenario the URL posted in not replicable.

4.2 The Intelligent Mousetrap Project

To examine the performance of SensorGIS we have hooked it to the Intelligent Mousetrap sensor network (IMSN). IMSN is a network in which sensors are capable of routing messages through other sensors. Routing of sensor messages eventually reaches a second-tier sensor (gateway sensor). The second-tier sensors are able to interact with both the sensors and with legacy IEEE 802.11 networks. The second-tier sensors form a logical mesh over which OLSR routing protocol [19] is used. One of the mesh sensors is linked to a wired backbone. Thus for sensor messages to reach nodes in the backbone, a two-step routing takes place. The first step is over the sensor network and the second is over the mesh sensors. The sensors are capable of broadcasting their existence and joining the network automatically once they are turned on.

In total, there are 3 kinds nodes incorporated in the architecture. The first are regular sensors based on DM2200 platform and they come with a Texas Instruments MSP430 processor, a proprietary RF Monolithic transceiver. The transceivers provide temperature, voltage and received signal strength indication (RSSI). In addition it keeps track of mouse status. There is a trap installed on each regular sensor. When the entrance is crossed a switch is triggered. The DM2200 will generate a reading for mouse status and sends it back to the mesh nodes. Though regular sensors sleep most of the time to save energy, every 30 seconds they awake for 9 seconds and send to the mesh node raw readings. Due to the smart power management strategy of the processor and the short duty cycle the life-span of nodes is in the order of weeks (depending on battery quality).

The second type is mesh nodes. On one hand they act as sensor network peers hence can communicate with the sensor nodes and collect their data. On the other hand they are able to

1. Provide access to roaming user through their IEEE 802.11 access point
2. Use a secondary IEEE 802.11 interface strictly for routing among the mesh nodes using the OLSR protocol

Mesh nodes do not perform sensing tasks of their own but they are the means of communication between legacy network protocols and sensor nodes. This allows the sensor nodes to run custom, even proprietary protocols (in our case PicOS [RefPicOS]). Essentially the mesh nodes act as data collectors. Each mesh node acts as master for a number of the sensors within its communication range.

The third type of nodes is Internet-attached servers. Sensor data collected by mesh nodes are routed among mesh nodes to reach the nearest mesh node, which is connected to the wired backbone. In the current setup only one mesh node is connected to the wired backbone. The mesh nodes are full-fledged x86 Linux platforms and are assumed not to be subjected to the same energy constraints as the sensor nodes. The mesh nodes are responsible for posting updates to SensorGIS system.

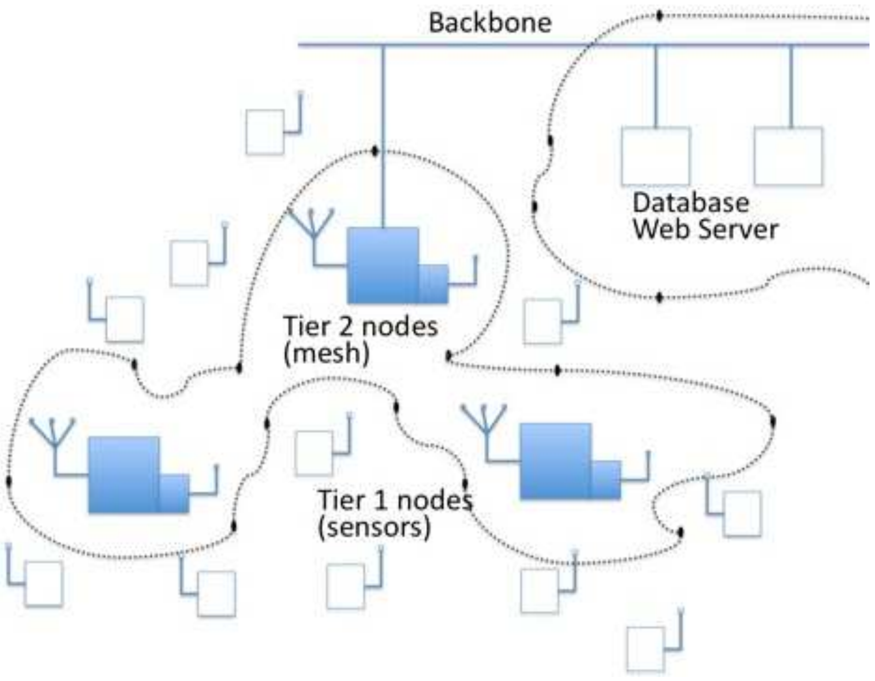


Figure 4.3: The Intelligent Mousetrap Network

Figure 4.3 shows how SensorGIS is integrated with IMSN. The only mesh node attached to backbone (which we call maintenance server) runs scripts and is responsible for network maintenance. The service API module lies on another server on the backbone. When raw readings arrive from the sensors to maintenance server they are parsed by the network-maintenance module and forwarded to the SensorGIS portal. The purpose of this setup is to illustrate the fact that SensorGIS can be integrated with an existing network, preserving all the existing setup without exposing the structure of the sensor network and routing to the backbone services. What's more, due to the double IEEE 802.11 wireless interfaces roaming users can access the SensorGIS from a laptop in the proximity of the mesh nodes, for example, via a pocket PC of PDA.

4.3 Experiment

The IMSN is in an experimental stage: it has not been actually deployed in the field but is used as a testbed for evaluating SensorGIS prototype. Since stability and reliability are the main concerns we conducted 2 test scenarios. The first one is to build IMSN from scratch by adding each individual node through the SensorGIS user interface and see whether SensorGIS provides correct data during the network startup phase. The second scenario involved running IMSN for a long time and assessing whether all the functions behave properly.

In the first test we added new sensors on the map. Irrespective of whether the newly added sensor had already been deployed or not, SensorGIS works correctly and is able to show the latest state values: inactive when the sensor is not deployed yet and correct values after its deployment. We also managed to move existing sensors without any problem. Sensor deletion was also tested. After deletion SensorGIS still behaves properly and the remaining sensors have no problem collecting the desired data after they automatically adjust routing. To test the mousetrap function, we triggered the switch manually. SensorGIS handles the reading of mouse caught properly and the user is able to reset the mousetrap status from the SensorGIS interface.

We also tested the system for a long period of time, which lasted for about half a month. During the test period, we keep retrieving data from the web interface. Operations such as get historic graph, show group readings and statistics are examined. It turns out the SensorGIS has no problem getting the right data when amount of data grows rapidly.

4.4 Hypothesis

The main goal of the IMSN experiment is to test the feasibility of the SOA framework we have proposed: all the APIs should be able to handle the real world situation and produce correct data. And any readings that come into the framework should be saved and audit trail should be kept. By comparing the data provided by the APIs with the raw data generated by the IMSN network at real-time, we prove that all the APIs are functioning correctly.

Chapter 5

SmartCondo

The SmartCondo application is the second application built on our sensor-network based architecture, designed to reflect the status of an indoor space and the behaviors of its occupant. The application is envisioned to support the collection of data of interest to healthcare professionals, monitoring patients who are evaluated regarding their ability to live independently. The SmartCondo application is complementary to the SensorGIS project. While SensorGIS project aims to provide an overall, high level overview of the widely-spread sensor network, SmartCondo focus on the activities of a patient living alone in a limited indoor space.

One of the most important objectives of SmartCondo is to model and instrument healthcare transition facilities, where an individual that have been subjected to a major medical intervention expected to develop autonomy before they are officially released from care. To encourage the development of such autonomy, it is important to limit intervention by others but still be able to monitor their progress over a period of time. Eventually the same technology employed can be subsequently used in the individuals residence.

To accomplish this task several features are needed. First, the monitoring system cannot use videotaping because it raises privacy issues. Second, the infrastructure should be affordable so that SmartCondo can be used for long term monitoring. Moreover the system must be able to accommodate various needs: patients come with different conditions and with time goes by new assistive infrastructure may be introduced.

Based on these assumptions, SmartCondo is built. It takes advantage of sensor network technology and is much less expensive than videotaping: sensors are cheaper than video camera and the sensors send back much less data compared to camera, which mean less network bandwidth and disk storage are required. Moreover it utilizes the service API module and provides a virtual world view (currently, Second Life). The virtual world environment provides an intuitive view of the real indoor space and a virtual world avatar is built to mimic the patients movements. Environmental changes within the indoor space are collected by sensors and sensor readings are used to compute the movement of avatar in virtual reality. Live stream of the patients activity can be alerted to intervene at the occurrence of a harmful event.

5.1 Architecture and Implementation

The architecture of SmartCondo and SensorGIS is similar. Both of them are built on top of service API module and have their front-end client accessing the sensor data through use of APIs. However the font-end client is slightly different from that of SensorGIS. Instead of using the thin, web browser client, SmartCondo requires a thick client which is called Second Life (SL). After the user starts the Second Life client, he can teleport to the SmartCondo space in the virtual world and monitor the patients activities.

The SmartCondo space is built according to the blueprint of a condo. A web-based tool is developed to facilitate the building process. The locations of wall and furniture in virtual reality can be entered via the web-based tool and the tool is able to interface with corresponding database. Once the input process is completed, a program within SL reads the information from the database and creates wall and furniture in the specified locations. At the same time, the wall and furniture locations are used to create a grid-based obstacle map, which guides the path planning algorithm. The path planning algorithm does not compute the patients location directly from sensor reading and only serve to provide better visualization.

The major difference of SmartCondo compared to SensorGIS is the introduction of a toolkit for sensor placement. Originally all the locations of sensor are input in the SensorGIS web interface. Later during the development of SmartCondo we find it is better to have a separate toolkit that can visualize the topology before the real deployment. This is especially useful for the indoor sensor network because it is not uncommon that sensor detection areas overlap with others. By virtually placing the sensors, developer can see clearly how sensor detection area overlap and thus be able improve topology. Another reason for bringing this toolkit into picture is to produce useful information that eventually used while computing the patients activities.

5.1.1 Stream Miner

Though SmartCondo uses several APIs rested on service API module, the most important API for SmartCondo is get action which is discussed in Section 3.3.2. This API relies solely on the results generated by stream miner. This section will shed light on the working mechanism of stream miner and explain how localization is done.

Before diving into the algorithm details, we have to explain the term intersection table because stream mining is all about how to utilize the information store in intersection table. In the intersection table, every entry is stored in the form of $(ID\_combination, coordinate)$. $ID\_combination$ records all the ids of sensors covering the intersection region while $coordinate$ holds latitude and longitude of the intersection center. Each intersection area has a bounding box and by default the center of intersection area is defined as the center of bounding box. However, the network developer who is using the visual toolkit can manually adjust the location of intersection center to better reflect the actual situation.

The stream mining component maintains a sliding window. Every raw reading that comes into service API module is copied in a buffer and the copy is passed to stream miner. This ensures the mining activity does not alter the actual readings that finally go into SensorDB. The reading copies have a soft-coded lifetime in the sliding window. Suppose lifetime is set to X seconds, then the first time stream miner sees a new timestamp it starts counting until X seconds later. At the end of X seconds, it selects all the reading copies with the same or similar timestamps. The selected copies are used to compute the coordinate of the patient living in the condo. If the selected readings have identical timestamp, then the timestamp of the computed coordinate is the same as the timestamp of reading copies. Otherwise the timestamp of the computed coordinate is set to the mean value of the selected timestamps. To give an example of this procedure, suppose the lifetime of copies is 5 seconds. At 12:00:00 the stream miner receives a reading with timestamp 11:59:59, the stream miner looks up its buffer window and finds that no stored copy has newer timestamp than 11:59:59. So the stream miner sets up a timer for the readings with timestamp 11:59:59 and keeps receiving reading copies. 5 seconds later at 12:00:05 the timer ends, all reading copies with timestamp 11:59:59 are consumed and erased from the window buffer. As a result, the patients location at time 11:59:59 is computed. The consumed process is ordered by time so in the example, after 12:00:05 no readings with timestamp older than 11:59:59 will exist in the window buffer. Usually the lifetime is set to a value slightly larger than the maximum transmission delay of the network. In this way by the time stream miner consumes readings with a specific timestamp, it is guaranteed to have all the possible readings generated by sensors. In other words, the mining result will have a delay nearly equal to or larger than the lifetime of reading copies, depending on the time taken to compute the coordinate.

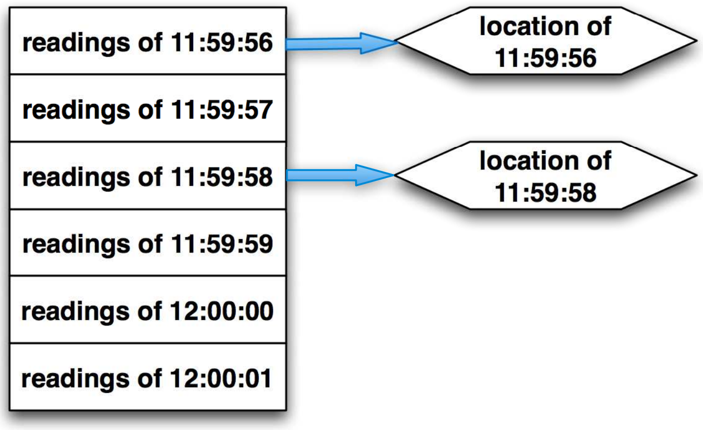


Figure 5.1: Stream Miner using One-second Timeslot

Ideally, no reading is lost and the sensor detection zone size is exactly the same as described in its documentation. When the stream miner extracts the sensor ID from the copies and put them into a combination, the combination should exactly match an entry in the intersection table. This is because, in theory, the intersection table lists all possible combinations of the IDs of sensors that may fire together, based on the sensor placement in the space layout. In this case everything is simple; no other information is needed except for readings with the same timestamp. We call the time slot used here is one second.

Unfortunately, sensors detection is subject to environmental changes. Given that the radio transceiver is cheap and unable to switch channels fast enough when it faces high interference levels, one cell phone call could bring down the simple model we discussed in the paragraph above. Readings can be lost due to interference. To solve this problem, look-ahead and look-back are introduced. In the simplest case no other information is used to infer location except the readings with exactly timestamp, however given the assumption readings can be lost it is helpful to look at the both the previous time slot and the next time slot. By caching the sensor IDs of last time slot and wait for one more time slot, the stream miner can reference these two time slot and make better decision. The advantage of this approach is that the action in previous timeslot and next timeslot usually contains meaningful information, which can be used to make the right judgment. For example, a man walking around the kitchen is detected by multiple sensors. At some point of time the reading of one of the sensor detecting this patient is lost. If the interference just lasts for a short time the patient should be detected by the same sensor again later and this information is useful to deducting the previous location during interference time. In the same way, looking back and referencing history can help determining the patients location. By looking both ahead and backward, we alleviate the impact brought by interference.

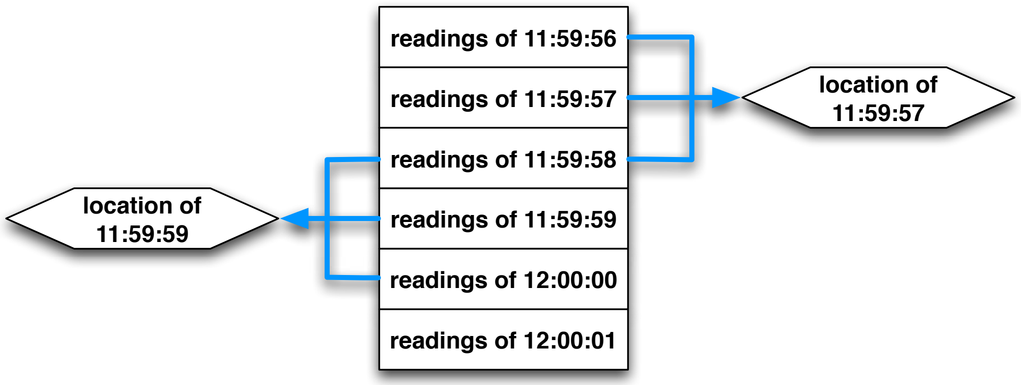


Figure 5.2: Anti-interference Stream Miner

While the stream miner might miss readings due to interference, it may also receive readings more than it expects. Sometimes the sensor detection zone is larger than what is stated in technical documentation due to different reasons. In this case, more readings than expected will come into the stream miner. For instance, many motion sensors have a slightly bigger detection area than the one described in documentation. When someone moves into an intersection region that is covered by three different sensors, in theory, by three different sensors, more than three readings with the same timestamp but different sensor ID may be generated. Besides this, when look-ahead and look-back are used, we might take into account extra readings which are actually not related to the movement. In the light of this, we use Euclidean distance to measure similarity of the sensor ID combination we get from readings and the sensor ID combination stored in intersection table. A sensor ID combination is translated into a multi-dimensional vector and the number of dimension is determined by the number of sensors deployed. Take a network with 3 sensors for example (sensors with ID $1, 2, 3$), the vector in this case is 3 dimensional. A sensor ID combination of $(1, 2, 3)$ and $(1, 3)$ will be translated into vector $(1, 1, 1)$ and $(1, 0, 1)$. The distance between this two vector is $\sqrt{((1-1)^2 + (1-0)^2 + (1-1)^2)} = 1$. The combination got from readings will be matched against each entry in the intersection table. By comparing the Euclidean distance of multi-dimensional vector, we know which entry in the intersection table fits best the sensor ID combination got from readings. Though ID may be missing or more than expected and the ID combination might not match exactly any entry in the intersection table, we will be able to find the match which is most likely the case by taking the combination with minimum Euclidean distance.

5.2 Visual Toolkit for Sensor Placement

The visual toolkit of the SmartCondo was designed to address two problems, namely to provide a platform to place sensors virtually and populate the SensorDB, and second to compute the intersection table used for localization. In this section the workflow of virtual sensor placement is illustrated and the usage of localization table is explained.

For example if a network developer wants to place sensors within a space so that the occupants movements are inferred, he must follow these steps:

1. Import blueprint of the condo
2. Define a reference point and provide its coordinate (latitude, longitude)
3. Define the scale of the blueprint by drawing a line on it and providing the lines length
4. Draw the boundary of the condo
5. Deploy / move / delete the sensors
6. If needed
   (a) Perform intersection analysis
   (b) Manually adjust the intersection result
   (c) Not satisfied with the topology, go back to step 5
7. Populate the database according to the virtual topology

During this process, steps 2 and 3 are used in conjunction to calculate the minimum and maximum latitude and longitude of the imported blueprint. For instance we import a $400 * 600$ jpg file as blueprint, then after step 3 we will know the latitude and longitude of all corners (namely, point (0, 0), (0, 599), (399, 599), (399, 0)). Only after step 3 the scale of the blueprint can be determined and can the network developer start deploying sensors. Step 4 gives a graphical representation of the condo boundary. The boundary is used while computing the intersection table, which is used for localization. In step 5, the developer can deploy any type of the predefined sensors in the condo. To initiate a new sensor, the developer must provide the sensors location (by clicking on the blueprint), Sensor ID and type. The shape of sensor coverage is defined based on the hardware documentations. For example on a 2D pane, a 10 meter infrared motion sensor covers a cone with radius of 10 meters and angle of 110 degree. When this sensor is attached to the condo ceiling, the shape of coverage becomes a circle. When a sensor is visualized on the blueprint, the size of its coverage is adjusted by the blueprint scale. If at any time the network developer re-defines the scale of the blueprint, then all the deployed sensors will be repainted and their sizes will changed correspondingly. The sensor being deployed must have a unique Sensor ID otherwise the deploy process will fail because all sensors must have unique identification.

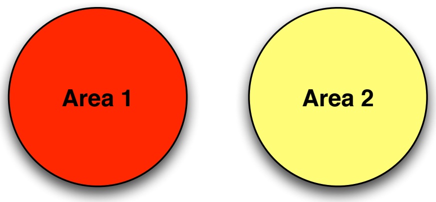


Figure 5.3: Two Sensor without Overlap

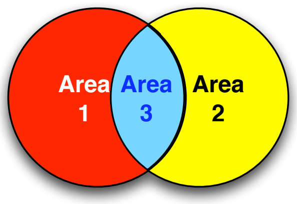


Figure 5.4: Two Sensor with Overlap

At the end of step 5, the virtual sensor network is deployed and the location information can be populated into SensorDB by executing step 7. However in the SmartCondo project, we need to infer from the sensor readings what the patient is doing in the condo. So we need to perform step 6, which generate a comprehensive table containing all the intersections in the current topology. To clarify what is considered as an intersection, we can look at the scenarios in Figure 5.3 and Figure 5.4. In the first scenario, the two sensors have no overlap. In this case we consider each of these two sensor coverage areas is an intersection. In the second scenario, there is a region covered by both two sensors. In this case we consider the yellow, red, blue region as separate intersections. Upon the execution of intersection analysis, the visual toolkit will also generate a center for each of the intersections.

Anytime the occupant is detected in an intersection, then we assume he is doing something at the center. To determine which intersection the patient is in, the stream miner has to get involved and further details are discussed in the following section.

The reason for creating intersections and centers is to improve the precision of localization. Sensors can fire readings if they detect the environmental changes caused by the patient, but there usually have no idea where exactly the changes happen. A motion sensor can generate readings if something is moving in its coverage area but there is no way to tell the exact location of the moving object. By breaking the large coverage areas into smaller intersections, the system can provide finer granularity of localization. In other words the more intersections are there in the condo, the more accurate the localization is. For instance two sensors are deployed in the condo and they do not overlap (case described in Figure 5.3), then the best localization could be done is to tell which sensor coverage area the patient is in. If the patient appears in one of the sensor coverage areas at some point of time and later he moves to another then he will disappear in the system for a short period of time. This is because when the patient exits the coverage area where he starts from and hasnt reached the destination coverage area he is not detected by any of these two sensors. To the contrary, if the two sensors overlap and we have 3 intersections, then there is a chance to see the patient moving from one sensor coverage area to another, if the moving path intersects the blue overlap region shown in Figure 5.4. Even if the patient does not follow this moving path we still manage to improve the localization precision: each of the intersections is smaller than that in the first case and we know the patient is standing somewhere in a comparatively smaller region at a certain point of time.

5.3 Experiment

The SmartCondo application has been deployed in a space of about 850 square feet on the University of Alberta campus. As part of an undergraduate course, teams of Industrial Design and Occupational Therapy (OT) students converted this area into a six-room condominium. Inside each room, they created prototypes for appliances, furniture and other fixtures. Within this space, we have deployed our sensor network which consists of 19 nodes. In terms of motion sensors, we have deployed 6 passive-infrared motion sensors for spot detection (Panasonic AMN43121) and 7 passive infrared motion sensors for wide-area detection (Panasonic AMN44121). Spaced throughout the condominium, the 13 motion sensors give us adequate coverage of the unit to successfully locate an occupant. Two of the chairs within the condo have pressure sensors (FlexiForce A201, 1 pound), which allow us to detect when someone sits on either chair. We attached reed switches to the front door and the door of the microwave, to determine whether they are open (or closed). We attached an accelerometer to the front door to detect knocking. We also placed electric current sensors within the unit to detect whether devices (e.g., a waffle maker) are turned on. After placing sensors within the unit, we worked with our colleagues in occupational therapy (OT) to evaluate our work. A test subject followed a number of scripts within the unit while the OTs evaluated the virtual representation. The SmartCondo contains a number of video cameras (which were installed in the room for a different project) which we used to verify inferences about the location of the occupant. Our OT colleagues matched the script with their observations of both the video feed and the virtual representation. The scripts included such actions as (a) moving from room to room, (b) sitting on chairs, and (c) opening and closing doors. The initial reaction of our colleagues was positive: they were very satisfied with the fidelity of the virtual representation with respect to the real world, and felt that this would be a useful tool for monitoring patients undergoing rehabilitation.

5.4 Hypothesis

The SmartCondo experiment is design to demonstrate that the occupant of a condo is tracked with reasonable precision in terms of space and time. Even though the precision is dependent on the WSN topology, the locations computed by the application should always be close to the real ones: if the occupant is in the detection zone of a sensor, then the computed location should be in the same detection zone or at least in the closest or adjacent detection zone. By comparing the locations shown by the 3D interface and the real locations of the occupant, we prove that the precision requirement is met.

On the other hand, we optimize the sensor placement interactively using the visual toolkit so that computation precision is improved. Based on the previous sensor placement and data, we are able to adjust the network topology so that better precision is achieved.

Furthermore, in terms of computation delay, SmartCondo achieve satisfying performance. Through out the test, it has been producing the computed location within 1.5 seconds. In other words if a sensor network has a network delay of maximum 5 seconds, then within 6.5 seconds right after the occupant's movement we are guaranteed to have the computed location.

Chapter 6

Towards Optimal Sensor Placement

Deploying wireless sensor network is a fairly straightforward task, but it is difficult to deploy the network in such a way that it meets all possible needs. This is especially true when talking about sensor network deployment for a monitoring application. Usually there are multiple types of constrains for sensor network placement: number of sensors is limited, network coverage needs to be optimized, accuracy must be guaranteed, certain important regions must be covered, obstacles must be avoided, so on and so forth. For example in the SmartCondo project, the number of sensors is limited and we need to detect as much area as possible. At the same time we want to ensure regions such as the living room and the bedroom are well covered since they have higher chance to detect the patients activities. All the requirements for sensor placement mix together and add great complexity to the problem of what is the best network topology. What's more, when the number of sensors increases complexity of the problem increase dramatically and its not intuitive for human being to figure out what is a good network placement. On the other hand, sensor network performance is subject to various environmental changes, it is nearly impossible to give a best answer or near-to-best answer if all possible elements have to be considered.

In the light of this, an algorithm called VFA Alpha is developed to help solve the optimization problem. It aims to provide a helpful, optimized layout of WSN so that the network developer can perform manual optimization based on the result generated by the algorithm. In trivial, intuitive cases that contain only a few sensors the algorithm might not be crucial, however for large scale, sophisticated deployment it greatly improves productivity by providing a starting point for further optimization.

6.1 VFA Alpha

VFA Alpha is inspired by the Virtual Force Algorithm (VFA) [13], which builds a decent model for optimal sensor placement. The key concept in VFA is virtual force. Virtual force is something similar to universal gravity and can be either negative or positive, pushing a sensor to different direction. The assumption of VFA is that sensors are placed randomly at first and their original placement acts as initial input for the algorithm. Based on the input topology VFA is executed multiple times, each time giving virtual force for each sensor. The main task of VFA is to define metrics for several aspects of sensor network performance and based on the defined metrics, to come up with a formula that indicates how each sensor should move to make the sensor network topology better. In other words, VFA is all about how to calculate virtual force based on limited types of metrics.

VFA Alpha is built based on similar assumptions but introduces the notion of preferential area and obstacle. Compared to VFA, VFA Alpha can steer away from the obstacles and at the same time cover as much preferential areas as possible. The VFA Alpha algorithm can be applied after the initial deployment of sensor network or even right before any sensor is actually placed. Upon each execution, a topology is taken as input and the output topology is passed to next iteration of algorithm as input. After iterating the algorithm dozens of times the topology will gradually converge to an optimal topology, depending on the parameters passed to the algorithm. When the user is satisfied with the topology, he can go and rearrange the network or start deploying the network according to the output of VFA alpha.

The major goals of VFA Alpha can be described as:

- Maximize total coverage of sensor network
- Maximizing the coverage of preferential area
- Keeping the sensors away from obstacles

There are lots of attributes worthy of considering but in the sense of sensor network for monitoring application, these 3 attributes might be the most interesting.

To fulfill the 3 different tasks, we must define metric for each of them and for each metric generate a virtual force formula. In VFA Alpha, the virtual force is split into 3 parts and can be described as

$$\overrightarrow{VF} = \overrightarrow{VF(s)} + \overrightarrow{VF(p)} + \overrightarrow{VF(o)} \tag{6.1}$$

In the formula, $\overrightarrow{VF}$ denotes the total virtual force that eventually exerted on a sensor. $\overrightarrow{VF(s)}$ is the force between a sensor and the rest of sensors in the network. $\overrightarrow{VF(p)}$ is the force between a sensor and all the preferential areas and $\overrightarrow{VF(o)}$ is the force between a sensor and all the obstacles. The 3 different forces will be discussed in the following sections.

6.1.1 Virtual Force between Sensors

The virtual force $\overrightarrow{VF(s)}$ is used to adjust the distance among sensors so that maximum total coverage can be achieved. Sensor detection range is subject to environmental changes and they have a probability of failure. For example, the radius of detection decreases when the temperature exceeds a certain threshold. Obstacles also decrease dramatically the detection radius. In general, places closer to the sensor are better covered and has a smaller chance of detection failure. Different sensors have different detection zones and the model of detection failure varies, therefore in VFA Alpha a parameter called closeness threshold is set. This parameter enables the network developer to define how close should sensors be so that if the sensors overlap, the overlapping region can be detected with a satisfying probability.

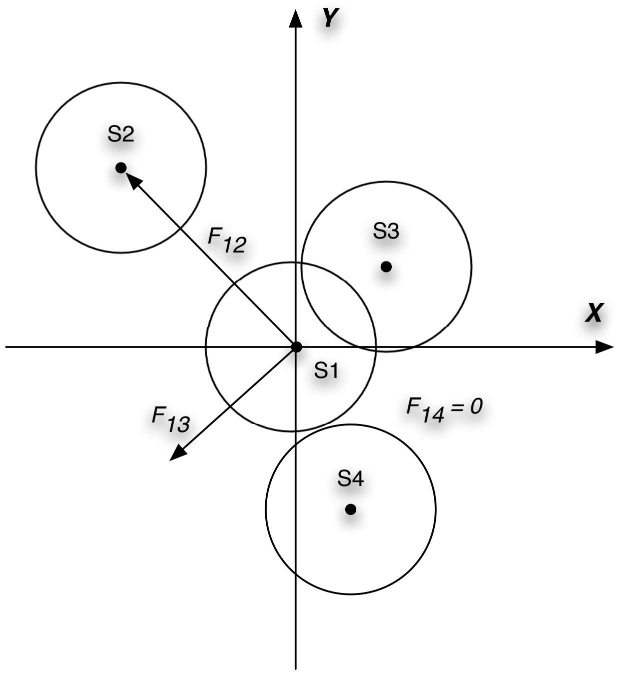


Figure 6.1: Virtual Forces

Figure 6.1 illustrates how $\overrightarrow{VF(s)}$ is computed. Suppose we are calculating $\overrightarrow{VF(s)}$ for sensor S1, then $\overrightarrow{VF(s)}$ equals to the sum of 3 different forces: the force between S1 and S2 ($\overrightarrow{F_{12}}$), the force between S1 and S3 ($\overrightarrow{F_{13}}$) and the force between S1 and S4 ($\overrightarrow{F_{14}}$). In this figure, all the sensors are of the same type and the circles denote detection zones of sensors. Closeness threshold is set to exactly the same as circle diameter. In this case, closeness is defined as the Euclidean distance between 2 circle centers. The magnitude and orientation of virtual force between S1 and other sensors is decided by closeness. ***When closeness between 2 sensors is larger than closeness threshold the virtual force is positive. When closeness is equal to threshold virtual force becomes zero and when closeness is smaller than threshold it is negative***. For instance, the closeness of S1 and S2 is larger than threshold, so the orientation of virtual force exerted on S1 is the arrow, which points from S1 to S2. This indicates sensor S1 should move along the virtual force direction and get closer to sensor S2. To the contrary, sensor S1 and sensor S3 is too close to each other and the virtual force is negative, pointing from S3 to S1. In other words, S3 is expelling S1. For sensor S1 and sensor S4, the closeness equals to threshold hence the virtual force equals to zero.

Besides deciding the orientation for each component of $\overrightarrow{VF(s)}$, we also need to determine the magnitude so that all the components can be added together to form $\overrightarrow{VF(s)}$. Note that there is a tradeoff between accuracy and computation time here: if the magnitude is defined to be a comparatively big value then the network topology converges more quickly, however the sensors are pulling or pushing each other too hard so that around the converging point the sensors keeps bouncing back and forth. In contrast, defining a comparatively small magnitude results in slow convergence and requires more iteration of VFA Alpha. Nevertheless, around the converging point the topology is more stable and the magnitude of bouncing is much smaller. In VFA Alpha, magnitude is expressed as followed:

$$M = \begin{cases} M * (\frac{Threshold}{Closeness})^2 & \text{if Closeness} > \text{Threshold} \\ M * \sqrt{\frac{Threshold}{Closeness}} & \text{if Closeness} < \text{Threshold} \\ 0 & \text{if Closeness = Threshold} \end{cases}$$

The advantage of such definition is that different sensors will draw closer to each if they are in vicinity otherwise stay still. This ensures sensors detecting different regions do not cluster due to execution of the algorithm. And when sensors get close enough or too close to others, they tend to expel each other so that coverage is not wasted. The closer they are, the bigger the force is. This simple approach enables proper convergence and improves the stability around converging point.

6.1.2 Virtual Force between Sensor and Preferential Areas

$\overrightarrow{VF(p)}$ is used to draw sensors closer to preferential areas so that most part of preferential areas are well covered. In VFA Alpha, a simple pre-process is performed before calculating $\overrightarrow{VF(p)}$. The algorithm first checks whether the number of sensors is bigger than the number of preferential areas. If this statement is true then proceed otherwise a subroutine is applied to chop preferential area into smaller pieces. The procedure is as followed:

1. Get the number of sensors and the number of preferential areas
2. If number of sensor is smaller or equal to that of preferential areas exit subroutine, otherwise go to step 3
3. Select the preferential area with biggest size
4. Divide the selected area into two areas. If the width of area is greater than height, divide vertically otherwise horizontally
5. Go back to step 1

In this way, before $\overrightarrow{VF(p)}$ is calculated, the number of sensors and the number of preferential areas are made the same. What's more, preferential areas are divided into areas with similar size. Later on $\overrightarrow{VF(p)}$ is computed in such a way that every preferential area is occupied by one sensor. Compared with the approach of not dividing preferential areas, this approach places higher priority on preferential area coverage than total network coverage. Most of the time it works well without decreasing total coverage. More discussion about the performance of this approach is revealed in the experiment section.

Coming back to the problem of how to define the orientation and magnitude of $\overrightarrow{VF(p)}$, it is obvious that the force should always point to one of the preferential area. All the preferential areas have two states: covered and uncovered. To clarify what is a covered area, the concept center of preferential area and a parameter called covered threshold are introduced. By the term center of preferential area, it means a point where a sensor can be placed so that the biggest coverage of the preferential area can be achieved. Ideally lots of algorithms can be applied to locate the center of preferential area. However in the VFA Alpha prototype, based on grids, we simply do a scanning. If the area has width bigger than height then the scanning will be vertical otherwise horizontal. When the number of point scanned equals to half the size of the area, the point is considered center of preferential area. Covered threshold is used in combination with center of preferential area to determine whether a sensor is covering an area. If the Euclidean distance between sensor and the center of preferential area is smaller or equal to covered threshold the preferential area is regarded covered otherwise uncovered.

If an area is covered, then it exerts no virtual force on sensors, including the sensor which is covering it. If an area is uncovered, then it generates positive force for the nearest sensor. ***The orientation of the virtual force is pointing from the sensor to the center of preferential area*** and the magnitude of force is expressed as

$$M = Distance * Covered\,Threshold \tag{6.2}$$

In the equation above, $Distance$ is the Euclidean distance between the sensor and the center of preferential area. Upon each iteration of VFA Alpha, the sensor attempts to move closer to an uncovered area. Obviously, each sensor has one-to-one correspondence with a preferential area.

6.1.3 Virtual Force between Sensor and Obstacles

$\overrightarrow{VF(o)}$ is the force that drives sensor away from all obstacles and it is always negative or zero. When the detection of a sensor does not intersect with any of the obstacles, $\overrightarrow{VF(o)}$ is zero. If the detection zone intersects with one to more obstacles, then each obstacle exerts a negative force on the sensor and tries to push it away.Similar to preferential area, each obstacle has a center and the center is defined in exactly the same way of that of preferential area. ***The orientation of each component of*** $\overrightarrow{VF(o)}$ ***is always pointing from the center of an obstacle to the sensor***. If the sensor itself is in the obstacle, then the magnitude of repulsive force equal to Euclidean between the sensor and the center of obstacle, as shown in Figure 6.2. If the sensor is not in obstacle but part of its detection zone is in obstacle, as shown in 6.3, then the magnitude equals to the length of line segment which is inside the obstacle.

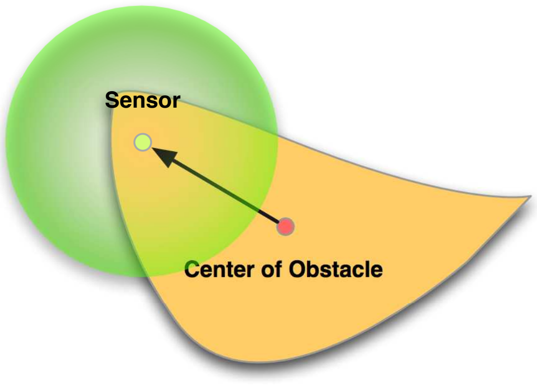


Figure 6.2: Sensor inside Obstacle

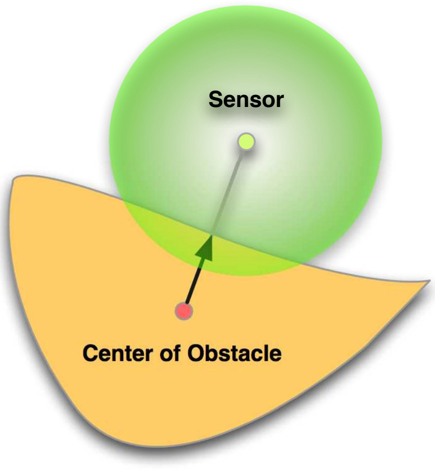


Figure 6.3: Sensor outside Obstacle

6.2 Experiment

An experiment is set up to test the performance of the VFA Alpha algorithm. The algorithm is embedded into a graphical user interface written in Java. The interface is able to perform experiment in batch mode and generate snapshots of the initial layout and final layout. To generate an initial layout, user can define the number of sensors, obstacles and preferential areas and the interface produces layout automatically according to user input. All the sensors deployed are of the same type and have a circle shape detection zone. All the obstacles are chosen randomly from a predefined obstacle set (obstacles of 7 different shapes and sizes, all of them are designed according to real furniture in a real condo). All the preferential areas are rectangles, which are the same size for a single test case and different size for different test cases. The sensors, obstacles and preferential areas are all placed in random locations.

A sample of 60 test cases is chosen for the experiment, which use 3 to 5 sensors, 1 to 5 preferential areas and 1 to 5 obstacles. The quality of final layouts is marked by a specialist. Three items are evaluated: 1) The algorithm sensibly divides the areas of interest; 2) The suggested topology avoids the obstacles; 3) The suggested topology adequately covers the preferential areas. Each item is marked using a 5-point schema:

- **1 Point** - Strongly disagree
- **2 Point** - Disagree
- **3 Point** - Neither disagree nor agree
- **4 Point** - Agree
- **5 Point** - Strongly agree

The summary of marks is presented in table 6.2. According to the domain expert, under some circumstances, the performance of the algorithm can be substantially improved: sometimes preferential areas are not divided equally and sensor fails to find its path to proper locations because too many obstacles are blocking its way. What's more, coverage of preferential areas can be further improved. However, on the whole the algorithm produces meaningful results. It provides helpful information of how preferential area should be divided and how sensors should be placed. Besides, the standard deviation of the results are small, which means the performance of the algorithm is stable.

Table 6.1: Evaluation of Experiment Result

	Min Mark	**Mean Mark**	**Max Mark**	**Std. Deviation**
Divide Region	3	4.67	5	0.68
Avoid Obstacles	1	4.53	5	1.21
Cover Areas	1	3.98	5	1.35

Chapter 7

Contributions and Future Work

7.1 Contributions

In this thesis, we propose and implement a reusable SOA framework for WSNs. This framework provide abstraction of WSNs, both outdoor and indoor. By reusing the APIs we developed, extra services and features could be built so that specific requirements could be met.

In Chapter 3, we proposed a REST style service-oriented architecture. This architecture provides a set of flexible, extendible, loosely-coupled services that are used to construct sensor network management applications. The architecture implementation discussed in this chapter ensures the underlying complexities of sensor network are transparent to the applications built on top of services.

In Chapter 4 we discussed the SensorGIS application. It has a 2D map view interface that offers the flexibility to work with multiple public maps and private overlays at the same time. Moreover a wiki view is incorporated. Cross-reference feature between map view and wiki view is provided so that sensor related information can be retrieved efficiently.

In Chapter 5 we discussed algorithm for localization alongside with SmartCondo application that provides a 3D view of interior space. Making use of the localization results, the SmartCondo application monitors the condo occupant and reproduces his activities near-real-time. Monitoring playback is enabled on web and it offers excellent opportunity for online healthcare training.

In Chapter 6 we proposed an algorithm for optimizing sensor placement. An automatic optimizing process was introduced. This process greatly reduces the time needed for manual optimization. The automatic process works as the cornerstone for further manual tuning and it enables the network developer to import and export network topology easily.

7.2 Future Work

There are a number of improvements that can be made for the applications and algorithms presented in this thesis. This section discusses the potential improvements for each of the components.

7.2.1 SensorGIS

A marker group manager is essential to improve the 2D visualization of large number of sensors, at different levels of geospatial detail, so that markers can be aggregated when the user zooms out. This technique is especially useful in representing the relationship among markers and, at the same time, prevents filling the map with a large number of markers.

In SensorGIS, a marker group would correspond to a sensor group. While the Google Maps marker group manager mainly deals with appearance based on zoom level, the marker group manager in SensorGIS must take care of both appearance and all the existing query or management operations. For instance when a marker denoting a sensor group is selected, group query functions should apply on this marker. A user interface for managing marker groups should also be provided so that users can change, delete, or add members in a group. This interface must communicate with the server and manage the database tables containing the group member list.

7.2.2 SmartCondo

Much of the future work in SmartCondo will be bringing radio frequency identity (RFID) into the system. Currently SmartCondo can only handle a single patient. When multiple residents are in the same condo the data stream miner do not have enough information to associate the sensors readings with the right person. However it is not unusual we need to monitor more than one patient in the condo, say a family with inherited disease. In this case RFID can be applied. A tiny RFID can be attached to each of the patient and sends out identity information. Given that identity is clarified, we can tell which sensor readings belong to which patient and generate meaningful results, which can be used in the 3D visualization.

Another thing that can be improved is the representation of different types of sensor readings within a virtual environment. For example, when a pressure sensor registers a reading corresponding to someone in the real world sitting down, this causes the avatar in the virtual environment to sit down, as well. However, one can easily imagine sensor readings that correspond to more complex real-world actions or activities, and the virtual environment will need to convey these actions realistically and effectively. One example is a heat sensor located over a stove element. Depending on the type of stove, this could be represented by glowing burner coils for an electric stove or a circular flame for a gas stove. Another important area that we plan to explore is two-way communication between the virtual environment and the real world. At present, the virtual environment is used to receive and display information. However, the patient could also be a participant in the virtual environment, and could be provided with tools to interact with medical staff or family members who are logged into the system. SL has a number of built-in communication tools, including the ability to chat using either a keyboard or a microphone and headset.

7.2.3 Optimal Sensor Placement

In the current algorithm, if the number of preferential area is larger than the number of sensors then the regions are divided until the 2 number are the same. Each time the biggest area is divided into 2 sub-areas. This approach works provides excellent result if the region needs to be divided into 2x sub-areas. However if this is not the case then the regions are divided well enough. Potentially a look-ahead can be introduced: based on the size of the areas the difference between number of area and number of sensors, we can predict how many times the biggest area should be divided until all the regions left have similar size. Based on the prediction, we can apply different dividing rules for the selected area.

Moreover, the issue of blocking obstacles needs to be solved: when a sensor is moved by virtual force it is actually trying to find a path to the proper location, oftentimes covering a preferential area. However when there are too many obstacles in its way the sensor might be blocked because the obstacles are exerting a greater negative force than the positive force exerted by preferential area. This could be solved by developing a adequate formula that results in greater $\overrightarrow{VF(p)}$, which is the positive force between sensor and preferential area.

Bibliography

[1] S. Meyer A. Rakotonirainy. A survey of research on context-aware homes. *ACSW Frontiers*, pages 159–168, 2003.

[2] Carsten Buschmann Dennis Pfisterer Stefan Fischer Sandor Fekete Alexander Kroller. Spyglass: A wireless sensor network visualizer. In *ACM SIGBED Review*, volume 1, page 2, 2005.

[3] The Coquet consortium. http://www.opencroquet.org.

[4] D. J. Cook. Health monitoring and assistance to support aging in place. *J. UCS*, 12(1):15–29, 2006.

[5] S. Helal W. C. Mann H. El-Zabadani J. King Y. Kaddoura E. Jansen. The gator tech smart house: A programmable pervasive space. In *IEEE Computer*, volume 38, pages 50–60, March 2005.

[6] Q. Tran G. Calcaterra E. Mynatt. Cooks collage: Deja vu display for a home kitchen. In *Proceedings of HOIT: Home-Oriented Informatics and Telematics*, pages 15–32, 2005.

[7] EXTJS. http://extjs.com/.

[8] L. Atallah B. Lo Y. Guang-Zhong F. Siegemund. Wirelessly accessible sensor populations (wasp) for elderly care monitoring. In *Second International Conference on Pervasive Computing Technologies for Healthcare*, pages 2–7, 2008.

[9] J. A. Kientz S. N. Patel A. Z. Tyebkhan B. Gane J. Wiley G. D. Abowd. Where's my stuff?: Design and evaluation of a mobile system for locating lost items for the visually impaired. In *Proceedings of the 8th international ACM SIGACCESS Conference on Computers and Accessibility*, pages 103–110, New York, October 2006.

[10] S. N. Patel K. N. Truong G. D. Abowd. Powerline positioning: A practical sub-room-level indoor location system for domestic use. In *Proceedings of UbiComp 2006: Ubiquitous Computing*, pages 441–458, 2006.

[11] T. Hori Y. Nishida H. Aizawa S. Murakami H. Mizoguchi. Sensor network for supporting elderly care home sensors. In *Proceedings of IEEE*, volume 2, pages 575–578, 2004.

[12] E. Ras M. Becke J. Koch. Engineering tele-health solutions in the ambient assisted living lab. In *AINAW 07: 21st International Conference on Advanced Information Networking and Applications Workshops*, volume 2, pages 804–809, May 2007.

[13] Y Zou K Chakrabarty. Sensor deployment and target localization based on virtual forces. In *Proc. IEEE INFOCOM Conf.*, volume 2, pages 1293–1303, 2003.

[14] F. Hupfeld M. Beigl. Spatially aware local communication in the raum system. In *Proceedings of the 7th international Workshop on interactive Distributed Multimedia Systems and Telecommunication Services*, volume 1905, pages 285–296, London, October 2000. Springer-Verlag. Lecture Notes In Computer Science.

[15] MediaWiki. http://www.mediawiki.org/wiki/MediaWiki/.

[16] Masayuki Furuyama Jun Mukai Michita Imai. *Viewlon: Visualizing Information on Semantic Sensor Network*. Springer Berlin / Heidelberg, 2007.

[17] OGC. http://www.opengeospatial.org/standards/sensorml.

[18] OpenLayers. http://openlayers.org/.

[19] T Clausen P Jacquet. Optimized link state routing protocol. In *Internet RFCs*, 2003.

[20] T. Kleinberger M. Becker E. Ras A. Holzinger P. Mller. *Ambient Intelligence in Assisted Living: Enable Elderly People to Handle Future Interfaces*, volume 4555. Springer Berlin / Heidelberg, 2007.

[21] Pachube. http://www.pachube.com/.

[22] Martin Turon. Mote-view: A sensor network monitoring and management tool. In *The Second IEEE Workshop on Embedded Network Sensors*, pages 11–18, May 2005.

[23] Yu Yang Peng Xia Liang Huang Quan Zhou Yongjun Xu Xiaowei Li. Snamp: A multi-sniffer and multi-view visualization platform for wireless sensor networks. In *1st IEEE Conference on Industrial Electronics and Applications*, pages 1–4, May 2006.