CSS497 Winter/Spring 2012 final report By Dmitry Zavyalov

Contents

[Summary 2](#_Toc326449748)

[Activities for this period 2](#_Toc326449749)

[Detailed description 2](#_Toc326449750)

[Sensor Simulator (FakeSensors) 2](#_Toc326449751)

[Purpose 2](#_Toc326449752)

[VikingX protocol 2](#_Toc326449753)

[Design and implementation 3](#_Toc326449754)

[Configuration file format 3](#_Toc326449755)

[Delivery 4](#_Toc326449756)

[Modifications in Connector 4](#_Toc326449757)

[initSensor() Connector call 5](#_Toc326449758)

[Sensors matrix support 5](#_Toc326449759)

[Modifications in Sensor Server (VikingX) 7](#_Toc326449760)

[Support for sensor aliases 7](#_Toc326449761)

[Refactoring the main class 8](#_Toc326449762)

[Separating communication layer from main class 8](#_Toc326449763)

[Tracing support 8](#_Toc326449764)

[New UDP listening support 8](#_Toc326449765)

# Summary

## Activities for this period

* Gained access to dslab servers/working folders
* Made myself familiar with Connector and VicingX sensor server code base
* Added alias support functionality into VikingX sensor map file
* Refactored VikingX code by moving sensor map file related functionality to separate class
* Was able to run, trace, and debug sensor server code with connector either on local machine and remote server. Note: I am using Eclipse 3.7.1 on local machine.
* Made myself familiar with VikingX sensor UDP protocol.
* Delivered the sensor simulator, which could emulate a number of sensors via playing previously recoded data files. Simulator also able to broadcast pre-configured GPS location information and sensor’s nick name.
* Was able to run sensor simulator, sensor server, and connector based test application on one machine for debugging.
* Implemented sensors matrix support in Connector library, which involves GPS coordinates/location support.
* Refactored VikingX code for separating transport protocol from the main code
* Sensor server now is not depending on listener code; it uses event-driven class to listen for variety of UDP broadcasts from sensors by running threads in the same process
* VikingX supports location data broadcasts (MACID\_BROADCAST) from sensors; this information is available for applications and sensors matrix in Connector

# Detailed description

## Sensor Simulator (FakeSensors)

### Purpose

Sensor simulator is designed for imitating a number of VikingX sensor servers in local network by replaying previously recorded files from real sensors. Also, The records can be generated using some mathematical modeling.

### VikingX protocol

The protocol is implemented via multicast UDP datagrams sent over the local network. Integer data should be presented in little-endian format. No padding between fields. The header should be sent first following by a payload.

Table 1: Header format

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Description** | **Note** |
| Length | 16 bit integer | Header+payload length |  |
| FrameType | 8 bit integer | COMMAND\_XXX |  |
| MacAddress | 24 bit integer |  |  |
| TransactionID | 24 bit integer | Increment it every new sample |  |
| Flags | 8 bit integer | No information |  |
| Interface | 8 bit integer | COMMAND\_INTERFACE\_XXX | A bit field |

Current payload is a comma-separated ASCII string contains the following fields:

<Payload Type>,<MAC Address>,<Temperature in F>,<Battery Voltage in Volts>,<RSSI in dB>

Also, sensors simulator can broadcast MACID\_BROADCAST packets, which contain sensor location information:

<Nick Name>, <MAC Address>, lat: <Latitude>, lng: <Longitude>, elv: <Elevation>

### Design and implementation

Sensor simulator is a multithread java application, which may be executed on variety of platforms supporting Java virtual machine. However, it was tested under Windows and Linux platforms only.

The FakeSensor class reads the data file line by line and transmit the response packets. It does not analyze or alter given payload. At the end of the data file, the stream is rewind to beginning.

FakeSensorMain class provides a basic command line interface, which might be used for future extensions of the simulator functionality like on the fly modifications or specific sample injections.



### Configuration file format

The application configuration is stored in **fakesensors.cfg** file, which has the following format.

Sensors = 3

Sensor1Data = 0x227bdb.csv

Sensor1Enabled = 1

Sensor1ReportPeriod = 10

Sensor1Mac = 0x227bdb

Sensor1Lat = 47.759444

Sensor1Lng = -122.191111

Sensor1Elv = 0

Sensor1Nick = FakeSens1

Sensor2Data = 0x227c3c.csv

Sensor2Enabled = 1

Sensor2ReportPeriod = 10

Sensor2Mac = 0x227c3c

Sensor2Lat = 47.759454

Sensor2Lng = -122.191112

Sensor2Elv = 1

Sensor2Nick = FakeSens2

Sensor3Data = 0x227c6a.csv

Sensor3Enabled = 1

Sensor3ReportPeriod = 10

Sensor3Mac = 0x227c6a

Sensor3Lat = 47.759464

Sensor3Lng = -122.191113

Sensor3Elv = 2

Sensor3Nick = FakeSensor3

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Data type** | **Description** | **Note** |
| Sensors | Numeric | Number of sensors | Required |
| SensorXData | FileName | File to read responses for Xth sensor | X should be between 1 and Sensors |
| SensorXEnabled | Boolean(1/0) | Allows to disable Xth sensor | Default = 0 |
| SensorXReportPeriod | Numeric | Report interval in seconds | Default = 0 |
| SensorXMac | Numeric | Mac address that sensor will be reporting | Use 0xNNNNNN for hexadecimal |
| SensorXLat | Float | Sensor latitude |  |
| SensorXLng | Float | Sensor longitude |  |
| SensorXElev | Float | Sensor elevation |  |
| SensorXNick | Text | Sensor nick name for MACID\_BROADCAST | Max 10 characters |

### Delivery

The sensor simulator source files were uploaded to Hercules machine under dslab account.

dslab/SensorGrid/FakeSensors

To compile, just type “make” in this folder. To execute, use **start.sh** script.

## Modifications in Connector

There are two modifications to Connector were done beside minor bugfixes

1. Add support for sensor aliases
2. Sensors matrix support

### initSensor() Connector call

The SensorDataSplitter class implements Runnable interface, so Connector may start a splitter thread for each sensor servicing.

The SensorDataSplitter helper class implements Readable interface, so this class can feed output Scanners.



### Sensors matrix support





Lat1, Lng1 – GPS coordinates of the [0,0] point in the orchard

Lat2, Lng2 – GPS coordinates that sensor reports

Interval – a distance between each sensor in the logical grid

Then, it is following the algorithm above. For calculating a distance between two GPS coordinates it uses well-known haversine formula, which is optimal for small distances.

1. Calculate angular distance between two points



1. Convert to linear distance



Where R is the Earth radius.

Please see the following references for more information:

<http://www.math.montana.edu/frankw/ccp/cases/Global-Positioning/spherical-coordinates/learn.htm>

<http://mathforum.org/library/drmath/view/51879.html>

## Modifications in Sensor Server (VikingX)



The following changes were made to sensor server

1. Add support for sensor aliases
2. Move out the sensor map support to separate class
3. Separate and abstract communication layer
4. Add tracing support
5. Switch to event-driven threaded class for UDP broadcast listening, which is replacing listener2.java
6. LIST ftp command provides actual sensors listing including location data and aliases
7. Privileged FTP commands cannot be executed without authentication
8. Minor bug fixes
9. Graceful server shutdown is possible

### Support for sensor aliases

The map file (sensor-file.txt) now is supporting new command [alias]. This command allows describing a number of sensors using their implicit names or regexp expressions as a wildcard under a single alias. Besides future cloud applications, this functionality is used in Connector’s sensor matrix, which is selecting required sensor from the list by its geographic location. The alias feature will allow to limit a search scope within a given sensors set.

### Refactoring the main class

Before digging in into the sensor map file support extensions, I did some refactoring by moving sensor map file related functionality into the separate class (SensorMap.java.) This refactoring is improving coding style, and increases the code readability. Also, some dead code was identified during refactoring.

### Separating communication layer from main class

Each transport protocol class should implement IServerTransport interface. From other side, main SensorServer class is implementing IServerCallbacks interface in order to provide necessary service level for the transports. This separation is improving stability and extensibility of the server. Besides better coding style, it allows another developer working on communication layer improvements without interfering with SensorServer developer.

### Tracing support

Errors, events, and debug messages could be saved to file, printed, or transmitted to remote host by ITracer interface derivative classes. Currently, the FileTracer class saves those messages to a plain text file. Good tracing is feasible for diagnosing low-reproducing problems.

### New UDP listening support

In the previous version, a hacked modification of Todd Elliot’s listener.java application (known as listener2) was started in separate process for piping out the temperature readings from sensors. This method had number of issues, but it also was not allowed to extend sensor server for capturing location information broadcasts. Also, it was required to run separate copy of the process for each application connected.

New SensorUDPListener class is running in the same process context and generating events for each client application connected. There are two types of events supported. First, it is the same temperature, signal strength, and battery voltage reading. Second event contains sensor location information. This type of events is not delivering to the applications directly, but caching in SensorMap to be returned by request.

