5-2006

Service oriented architecture for real time data fusion.

Derek Ray Massey
University of Louisville

Service Oriented Architecture for Real Time Data Fusion

By

Derek Ray Massey
B.S. University of Louisville, 2004

A Thesis
Submitted to the Faculty of the
University of Louisville
Speed Engineering School
in Partial Fulfillment of the Requirements
for the Professional Degree of

MASTER OF ENGINEERING

Department of Computer Engineering and Computer Science
University of Louisville

May 2006

Service Oriented Architecture for Real Time Data Fusion

By

Derek Ray Massey
B.S., University of Louisville, 2004

A Thesis Approved on

May 2006

By the following Thesis Committee:

____
Dr. Rammohan K. Ragade, CECS Dept. Thesis Director

____
Dr. Dar Jen Chang, CECS Dept.

____
Dr. Suraj M. Alexander, IE Dept.

Dedication

I dedicate this project and my accomplishments to everyone who has been in support of me. I dedicate this to all my family and friends. To Billy and Annine Massey who are the best parents anyone could ask for. And most of all to God who leads me in the right direction.

ACKNOWLEDGMENTS

I would like to acknowledge Dr. Rammohan K. Ragade who was my thesis director. He has continued to push and motivate me throughout my college career. I would also like to acknowledge my academic counselor Brenda Hart for being there for me from the start of my academic career until the end. I would also like to recognize Dr. Suraj M. Alexander on my committee whom also served as my mentor coming into college. No matter how busy he was, he never was too busy to talk to me. Also on my committee Dr. Dar Jen Chang who has continued to be one of my favorite professors at Speed Engineering School. Lastly I would like to acknowledge Brian Carter for giving me direction in this project.

ABSTRACT

This project will provide a service-oriented architecture to handle sensor data in real time as the information comes in. There are two types of sensors we're implementing into our project, mobile sensors and stationary sensors. These sensors attach unto motes to gather data about temperature, light and acoustics. The fusion part of the topic is taking both types of sensors, bringing the data together and storing the data in a SQL Database. This project will focus on the gathering, storing and preprocessing of the data. The data from the sensors is stored every three minutes using the BULK INSERT command. We found that storing the data every three minutes is about the most efficient for our implementation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
INTRODUCTION

The project involves providing a service-oriented architecture to handle sensor data in real time as the information comes in. The data fusion part of the project addresses many sensor platforms that use different protocols such as Surge Reliable, Zigbee, Sensornet, ect. It also addresses different types of sensors such as mobile sensors and stationary sensors. Data fusion is a method designed to compute data from multiple sensors and to use the redundancy to improve the quality of the information produced in terms of the quality of the monitored data and alarm management. The data fusion part of the topic is taking both types of sensors and bringing them together in a data warehouse. Then an analysis of some performance issues of this architecture will be conducted. The real purpose of this project is irrelevant to the measurements themselves that provide an accurate estimation of the environment, but to the timely fashion of how the information is gathered, stored and preprocessed. The goal of the architecture is to maximize efficiency of these operations.

1.2 Introduction to Architecture

Providing an efficient architecture for real time data fusion is useful for a complex environment when many factors depend on the backbone of having a good infrastructure. For this project the environment consists of the sensors being dispersed throughout the atmosphere gathering inputs from the environment. The end result of the project is running the reports and getting a sense of the project through visualization (graphs, measurements).

A good architecture is needed to assist that there be a smooth transaction between gathering the inputs and dispersing the outputs. A good architecture is needed for any type of environment that is driven from data. Intensive care units in the medical industry use these practices to diagnose and treat many types of disease in a timely fashion. "...Most of the studies are seeking to reduce the number of false alarms (those with no clinical significance) by using multiparametric approaches: most of the time it is the simultaneous variation of several parameters that is characteristic of an event..." (Chambrin, 2001).

Cases were an efficient architecture is needed crosses over into many domains. Suppose someone needs heart surgery in the next 48 hours, there needs to be a good architecture in that the problem can be diagnosed correctly, located and a solution can be concluded in a timely fashion. If there is a problem on any end of handling the information from inputs to output then the patient is at risk. Or suppose our sensor network was setup up to detect a certain phenomena in a chemical making process in that all the variables that are needed were being collected and handled by our architecture. If there are problems with the timing of getting the information and displaying the results we could miss the phenomena that the whole network was set up to detect. This is why issues with the real time architecture cannot be overlooked.

1.3 Our Project in a Logical Sense

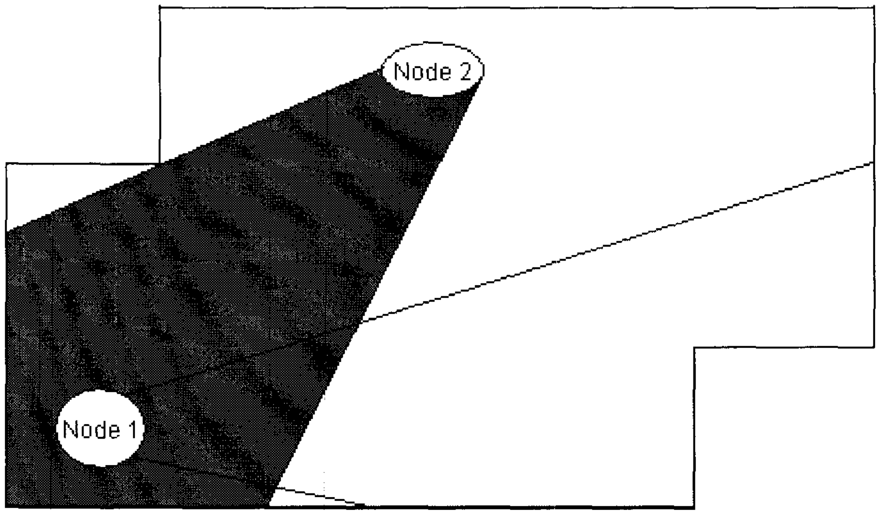
The project will be accomplished using the following tools: a base sensor, stationary sensors (motes), mobile sensors (robot dogs), Windows 2003 server based distributed network using alchemist, crossbow's software for the sensors, SQL Server 2003, Java, ASP.NET and Visual Basic.NET. A general explanation of how the project will work is that the sensors will make data and send it out to a text file located on the server. On the server will be running a VB.NET program that will provide all the queries and necessary instructions for our program. The program takes the text file and dumps the lines of the text file into a SQL database. From there we shrink the size of the table by taking unnecessary information out and storing the data into a new table. Lastly there is an ASP.NET webpage that allows filtering through the data.

1.4 Physical Hardware Components

Generally the sensors communicate through wireless technology. The base sensor is the sensor that is attached via serial cable to the server and all other sensors communicate with the base sensor through wireless technology. There are a few different types of sensors. One that this project will be using is an uncoordinated mobile node which is a sensor whose mobility or direction is not directed for any specific sensing activity. Instead, it is a node that independently observes a cross section of the field along its own path. "The limited observation of a node can be greatly extended via information exchange among collaborating nodes coming across each other (Wang). Figure 1 shows two nodes traversing through an atmosphere. Node1's sensor and travel direction is colored in yellow. Node2's sensor and travel direction is colored in green. At the point where they both cover the same area in the atmosphere is colored in red.

Figure 1

Node Traversal



This project being a service-oriented architecture for real time sensor data fusion will cover analysis of time and space in data collection. It will also do analysis on the raw encoded measurements (temperature, light and acoustics) from the sensors. The overall goal of this project is to maximize efficiency of real-time operations. We plan to accomplish this using our chosen methods and tools.

CHAPTER II
LITERATURE REVIEW

There have been many projects and research in sensor data fusion. The University of Rochester, TIMC and scholars at Mississippi State have similar projects as ours but they're all different in they're own respect (Carvalho, 2002). The University of Rochester project focuses on adaptation, a Unified Modeling Language and resource management, while the TIMC laboratory uses a CAN network for multichannel sound acquisition (Virone, 2003). Lastly the scholars at Mississippi State focus on an Intelligent Intrusion Detection System (Siraj, 2004). None of the projects read address different types of sensor networks (PNP, Zigbee, ect.). The goal of our project (SNITTER) is to accommodate heterogeneous types of sensor networks (Carter and Ragade, 2005).

This section will review the following projects from:

- The University of Rochester
- TIMC laboratory
- Mississippi State University

A project comparison table is given in table 2.1.

Table 2.1

Project Comparisons Chart

Comparisons							
School	PNP	Uncertainties	Reliability	Adaptation	High Level Fusion	Low Level Fusion	Network Protocol
SNITTER	yes	No	yes	yes	yes	no	Surge Reliable
University of Rochester	no	Yes	yes	yes	yes	yes	PHMS
TIMC	yes	No	yes	no	yes	yes	CAN
Mississippi State	no	Yes	yes	yes	no	no	IIDS

2.2 The University of Rochester Project

At the University of Rochester, scholars have used a general approach for data fusion architecture along with adaptation, UML (Unified Modeling Language) and resource management. The architecture they developed is like our architecture such that it can run on a network-based dynamic distributed system or conventional stand alone systems. Their main focus was to develop a data fusion model that would include adaptation being able to change the sensors depending on conditions in the environment (Carvalho, 2002). They felt that "... as the state of the system being monitored and available resources change, the general data fusion framework should change dynamically based on the current environment and available resources in the system ..." (Carvalho, 2002).

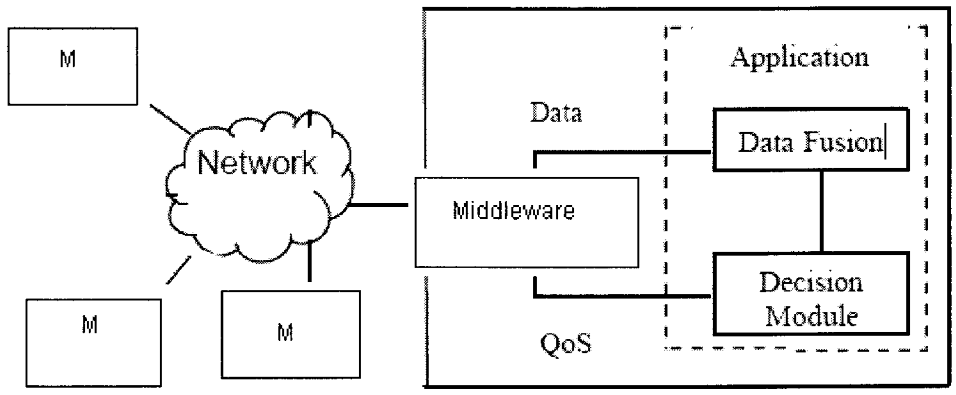
The reason for their project is that they felt none of the frameworks before dealt with the uncertainties of the system. Sensor measurements have problems with noise which is the electrical interference amongst circuit parts and the environmental factors. Also sensors are prone to errors. Sensor data can also suffer from incompleteness. Incompleteness happens when there aren't enough sensors to develop enough detailed data for a complete view of the world. This then turns into the reliability of a sensor or the amount of confidence with the data being output from the sensor. These are all issues that the scholars felt had not been addressed before.

2.2.1 University of Rochester's Architecture

Carvalho et al develop their architecture which is based on their belief that a network based system needs all of its components to deal with dynamic changes in the availability of resources and changes in the environment. Their network based system is broken down into three parts: network, middleware and applications. They feel the system should adapt to availability of sensors and also to the changes in the measurements. Below is a visual blueprint of their architecture combining the physical network layer, the middleware layer and lastly an application layer (Carvalho, 2002).

Figure 2.2.1

University of Rochester Architecture



Network, middleware, and application

As one can see in figure 2.2.1 the sensor nodes represent the network layer; in this paper they call the nodes service suppliers or data sources and they call the application layer service consumers. The middleware's job is connecting the sensors to the application over the network. The application contains two parts, a data fusion model and a decision module. The application sends requirements to the middleware and the middleware sends sensor data to the application.

2.2.2 University of Rochester's Data Fusion

The data fusion part of their project can be divided into three main types of fusion: data oriented, task oriented (variable) and a mixture of both. The difference between data and variable is that data is a measurement of the environment that is generated by a sensor. Variable is determined by analyzing the data or feature extraction. The variable determination can be answered by using data mining techniques such as a data analysis algorithm or a neural network (Kantardzic, 2003). Determining these three levels of data fusion is based on whether the fusion is before data analysis (raw data), after (variable) or both. This is also referred to as high level and low level data fusion respectively. Carvalho et al's architecture is dynamic enough so that applications can have only low level data fusion if that is desired, high level data fusion or both as appropriate (Carvalho, 2002).

2.2.3 Applications for the University of Rochester

Further, Carvalho et al present their data fusion architecture by mapping to different applications in various domains. In military applications they can achieve both levels of fusion to detect biological agents used in war. In one application soldiers can carry reliable sensors that advise them when a biological agent is deployed in the environment. In another application the same soldier can have another sensor that detects as before and also detect modifications in the soldier's body. A neural network can be used on the input to determine if an infection resulting from a biological agent has occurred.

Another domain of applications for their data fusion architecture is in robot navigation. One can imagine a simple application in that sensors evaluate the environment and give feedback to the robot to which path he should try and take. The robot can take account many types of different variables: robot position, distance from the objects around the robot and types of surrounding surfaces. It can then take these variables, fuse them and make a navigational decision.

Another domain of applications for their data fusion architecture could be found in home security systems. These systems could have many different types of sensors such as: sound, video cameras, ultrasound, temperature, smoke, vibration or infra-red. These different sensors can work together to detect intruders, fires and other problems. The data fusion architecture of their project could perform low-level data fusion for redundant data from some sensors while algorithms run to extract variables for other data. For instance if the vibration sensor detects a broken window and the motion sensor detects a person it can be concluded that there is an intruder.

2.2.4 Summary of the University of Rochester's Work

The project with scholars from the University of Rochester has a project similar to ours that focuses on dynamic modification of their system according to different states of the environment. Their future goals are to develop a communication between the middleware layer and the application to allow them to achieve the quality of service of the whole system when there is more than one application running. Their project still does not address different sensor networks being able to work together.

2.3 TIMC Laboratory Project

A separate project in the TIMC laboratory for the remote monitoring of elderly patient's health status a system based on multichannel sound acquisition has been used. The goal of the team was to eliminate video taping of elderly patients because of privacy issues (Virone, 2003). So instead, they turned to a CAN network (Controller Area Network) linked to volumetric, physiological and environment sensors. A CAN network is a serial bus system that comprises the data link layer of the network that makes hardware capable of sending and requesting of messages (Nilsson, 2001). The multichannel sound system will allow them to detect if a person is in distress.

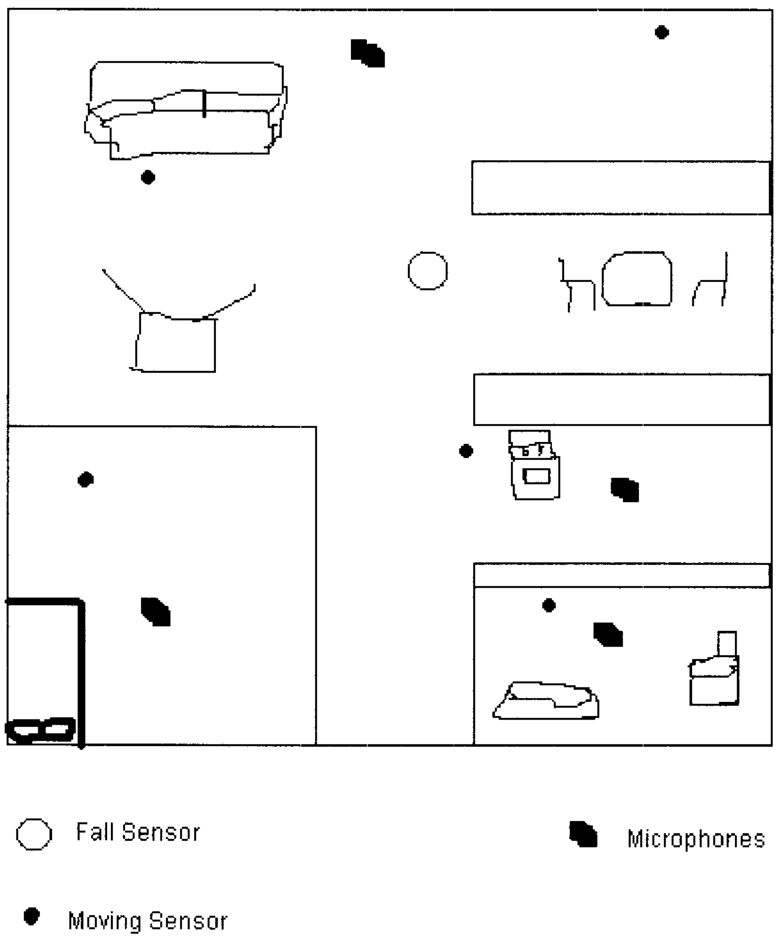
The reason for their project was looking at the increasing number of people over the age of 60. According to their statistics by 2030 the percentage of people over the age of 60 is expected to be twenty percent. There are around 23 million people who serve as caregivers to the elderly which is proof that there is a need for new techniques to provide social and healthcare services. These services should be technologically up to date such that caregivers are advised of problems as quickly as possible. This in turn has formed a new type of residential care called Health Smart Homes (HSH). One of the main goals of their project was replacing video cameras with a system of multichannel sound acquisition. The multichannel sound acquisition performs an analysis of a real time sound environment of the home to detect abnormal noises such as helps or moans (Virone, 2003).

2.3.1 Architecture of the TIMC Project

The physical environment of the architecture is a 30m squared apartment that consists of two rooms and a kitchen. It also contains a technical area for the development of technologies to ensure the security and quality of life for its patients. The physical environment uses smart sensors like volumetric, audio, physiologic and environmental sensors linked to a master PC using the CAN protocol spoke of earlier. There are to be eight microphones which are linked to a slave PC which compromises all the audio sensors. These sound sensors monitor the patient's position and sound activity within the environment. A blueprint of the physical environment chosen by them is in Figure 2.3.1.

Figure 2.3.1

TIMC Architecture



One element that's found in their architecture is that each sensor delivers its information at its own rate. The rate at which a sensor delivers messages is usually constant. When a patient tries to bring about or affect the readings on a sensor, the time lapse between two messages can be long which makes it easy to detect an abnormal event. This is called a

Producer-Consumer model because both the producer (sensor) and the consumer (patient) use data at their own rates. The CAN network aids this process by providing a common protocol between the sensors, actuators and the computer (Virone, 2003).

Another element in their architecture is called Information Diffusion. This is where each data frame constituting a message from a sensor is endowed with a bit identifier. This lets you know which sensor is sending the information. All the nodes connected receive the message and can decide to handle it based on its identifier. Enabling the system to share information among all the sensors is also called distributed intelligence. Furthermore distributed intelligence allows for "Plug and Play" capabilities being able to add or remove sensors in an easy manner (Virone, 2003). Our SNITTER project that we are currently working on allows Plug and Play capabilities (Carter, 2006).

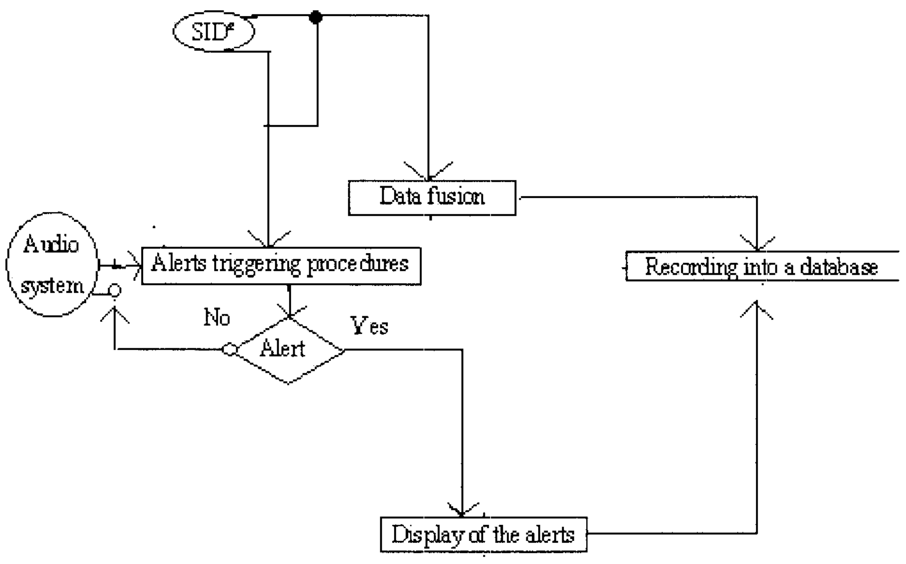
Another element of their architecture is the roles of the master and slave PC's. The only task of the slave PC is to serve as the single sound gathering machine. The slave machine is actually hooked up to 8 different microphones, 8 signal conditioning boards and a data acquisition board. All of this together handles the incoming sound. The master PC is in charge of receiving data from the CAN network protocol. It connected with a PCI-CAN/2 board linked to a software application. The software application is capable of receiving incoming messages from all the other sensors and saves them into a database file or XML (Virone, 2003).

2.3.2 Data Fusion of the TIMC Project

Alert triggering procedures are made from two types: short and long-term alerts. Short term alerts are instantaneously triggered and long term alerts are obtained after an analysis period. Short term alerts happen when there has been a reception from the environment or the sound system. Long term alerts extract disease scenarios from a database that has been built based on all possible combinations of alerts. Figure 2.3.2 shows how this part works (Virone, 2003).

Figure 2.3.2

Alert Triggering Procedures



2.3.3 Conclusion of the TIMC Project

The Health Smart Home application is part of a new wave of technology brought on by sensor networks and data fusion. Their application presents a communication between the home health monitoring system and sound systems to increase the data processing of sharing information to correctly alert caregivers. This application is a lot like ours in that it supports Plug and Play. One issue that it still doesn't cover is networks beyond Plug and Play (Virone, 2003).

2.4 The Mississippi State Project

In the Intrusion Detection Sensor Data Fusion project that Ambareen Siraj et al at Mississippi State University have come together to develop a secure multi-sensor system for intrusion detection. They felt there was a need for more secured systems in this area in order to achieve trustworthiness. In the past, most intrusion detection systems employ multiple sensors to maximize their trustworthiness. The scholars at Mississippi State came up with what they called a Decision Engine for an Intelligent Intrusion Detection System (IIDS). The system fuses information from different intrusion detection sensors using a form of artificial intelligence. This method is accomplished using Fuzzy logic, Fuzzy Cognitive Maps (FCMs), fuzzy rule-bases for causal knowledge acquisition and the causal knowledge reasoning process (Siraj, 2004).

The project from Mississippi State can be thought of in a different way than our project. The sensors used for this project (anomaly detection sensors and misuse sensors) are generally software coded modular sensors. The sensors used for our project are actual circuit board sensors. The reason we included this project to briefly talk about was that even though the physical part of both projects differ, some of the logic and concepts of gathering, storing and pre-processing remain similar.

2.4.1 Mississippi State's Architecture

The Intelligent Intrusion Detection System (IIDS) is characterized by a few unique features. One key feature is that it has a distributed and network based modular architecture to monitor activities across a network. Another feature is that it can detect anomaly and misuse situations. An anomaly situation is where it detects deviations from normal behavior conditions. Misuse detection is where it has found a known pattern of attack (Siraj, 2004). S Udupa Ratish at the University of Louisville accomplished misuse detection with USE Case Modeling. His Use Case Modeling introduced policies that would send you to different system states to detect misuse (Ratish, 2005). Another feature of the IIDS is that it is intelligent accomplished by employing artificial intelligence techniques for anomaly detection and alert fusion from the different types of sensors. Lastly the IIDS is equipped with a graphical presentation tool to monitor security status and the IIDS is adaptive to changing network or user behaviors (Siraj, 2004).

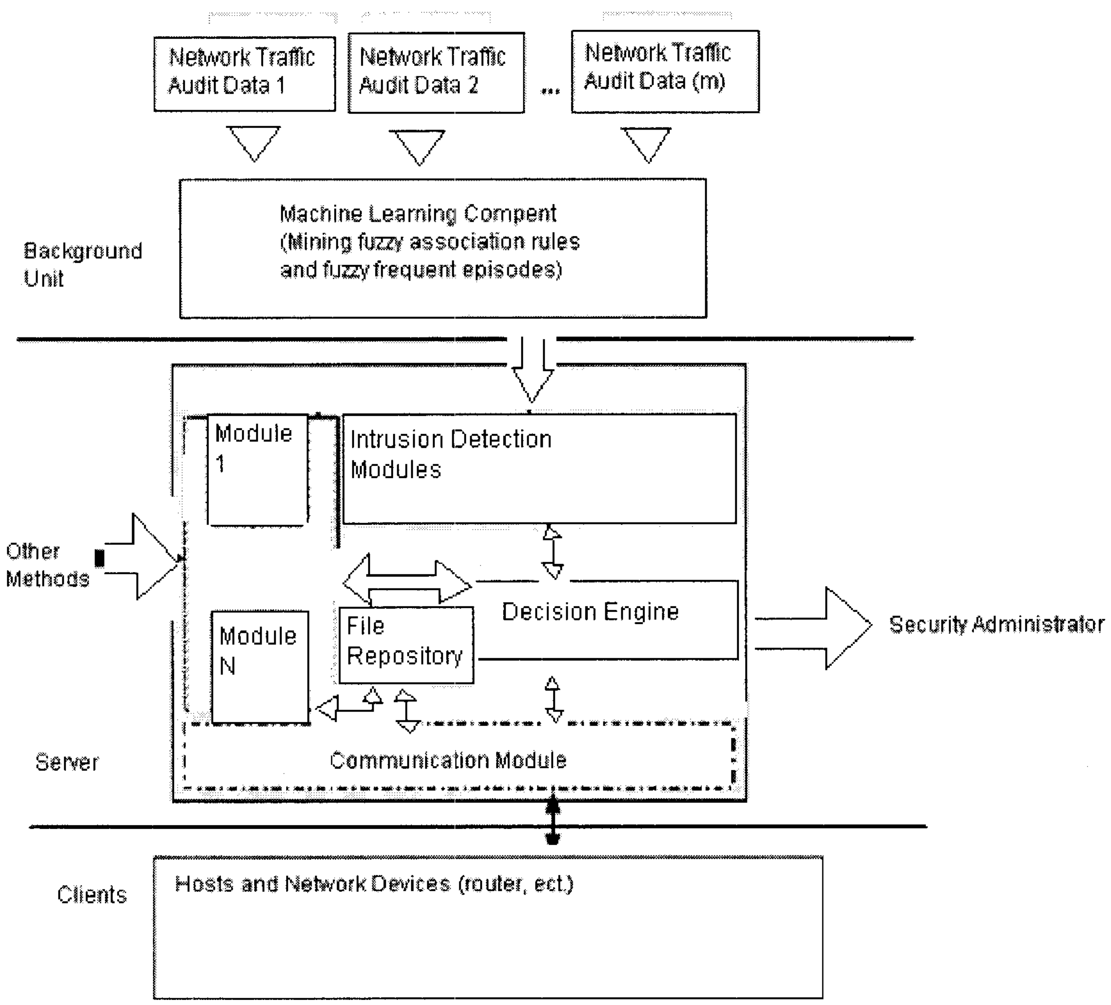
Even though the sensors are software based, the architecture of the system is very similar. The two types of software sensors they use are anomaly detection and misuse sensors. These sensors perform an audit of the network traffic monitoring the network and applications. The information passed from these two types of sensors are fused then stored in a database.

The next part of the system, Machine Learning Component, performs data mining techniques to the data. In this component the system uses Fuzzy Cognitive Maps which is a form of data mining to extract fuzzy association rules and fuzzy frequent episodes. In fuzzy logic there's no defined system. If you take two inputs from a source that are the same there maybe be different outcomes, this is the basis behind fuzzy logic. In this case Fuzzy Cognitive Maps are made up of nodes and directions to other nodes. Each node is a state and each direction edge sends you to another state in the Intrusion Detection system (Siraj, 2004).

The output of the data mining techniques is then sent to the core component or server where the decision engine decides what to do with the information. Here it can check its file repository for information on how it should handle its input and decide whether it needs to alert. A diagram of the architecture is shown in Figure 2.4.1 (Siraj, 2004).

Figure 2.4.1

Mississippi State Architecture



Network Traffic Audit Data 1
Network Traffic Audit Data 2
...
Network Traffic Audit Data (m)
Background Unit
Machine Learning Compent (Mining fuzzy association rules and fuzzy frequent episodes)
Module 1
Intrusion Detection Modules
Other Methods
Decision Engine
File Repository
Module N
Security Administrator
Server
Communication Module
Clients
Hosts and Network Devices (router, ect.)

2.4.2 Conclusion of the Mississippi State Project

The Intrusion Detection project seemed to work fine. It was tested on a test set of data that the Mississippi Scholars provided which can only be so accurate considering that it was only tested for attacks that have happened or that it was supposed to find. It will only be interesting to see if it can detect and alert on situations that previously haven't happened. Yet this project is very different from ours the gathering storing and pre-processing is similar. It shows that some parts of our project can be applied to any domain (Siraj, 2004).

CHAPTER III
IMPLEMENTATION

3.1 Requirements

In gathering information about this project we have to do research on existing projects like the ones talked about earlier in this paper. There is a lot of work that is similar to what we are trying to accomplish. Most of the work has features that we are not interested in while other works have features that we want to add. This part of the project consists of gathering, storing and preprocessing data; our group has requirements that pertain to these three concepts.

For gathering data the primary focus is doing it in an efficient process. The gathering process should be done as quickly as possible for other procedures later. In the plug and play network the gathering process should be simple no matter how many sensor devices are attached to our sensor network. In that case it can be said that one goal in the gathering data process is to make a way of gathering data so that a large number of sensors wouldn't make a difference or slow the process.

For storing data we want to achieve an efficient process. We also want to store the data as quickly as possible for preprocessing the data in our next step. The data to be stored is to be done in a SQL database. It is to be stored in the SQL database in a manner that it will be easy to query in a GUI at a later time.

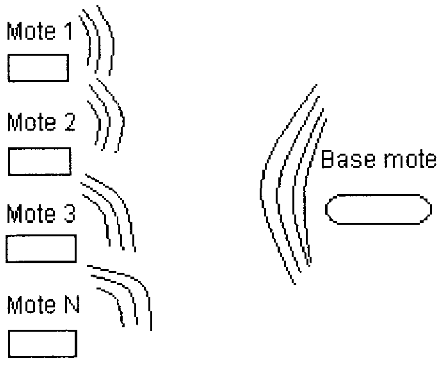
Lastly for preprocessing the data we are going to use some data mining techniques. This part of the project doesn't focus on this too much, since data mining techniques would have a big effect on sensor coverage. But for the most part we want to reduce the features that are of no use to us.

3.2 Architecture

We use wireless sensors as a part of our physical architecture. The wireless sensors have to be assembled from pieces. First we take the motes which are cable of the mesh networking firmware and assemble those first. These motes come built in with wireless technology for networking. First we find the mote labeled as Base_###_0, this is the mote that connects to our server through a RS-232 straight-through serial cable. All other motes connect to our network by communicating directly with the base mote wirelessly as shown in Figure 3.2.1 (Crossbow, 2004).

Figure 3.2.1

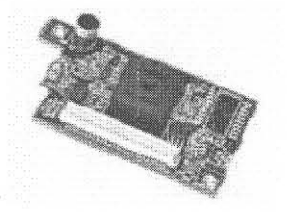
Wireless Network



In order to make these motes cable of gathering sensor data we must attach a sensor board to it. The sensor boards attach to the motes through a 19 pin connector residing on both the mote and sensor boards. Figure 3.2.2 is a picture of a MTS 300/310 sensor board shown below (Crossbow, 2004).

Figure 3.2.2

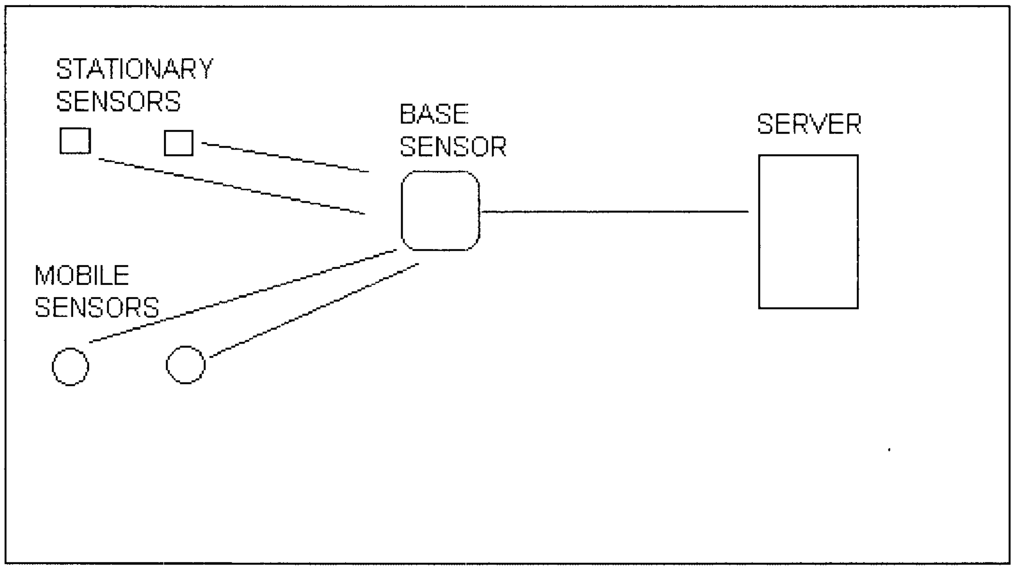
MTS 300/310



After connecting the base sensor to the serial port on our server, we now have the backbone to our plug and play network. As shown in Figure 3.2.3 the stationary and mobile sensors communicate with the base sensor which is connected to our server through a serial cable.

Figure 3.2.3

Network Drawing

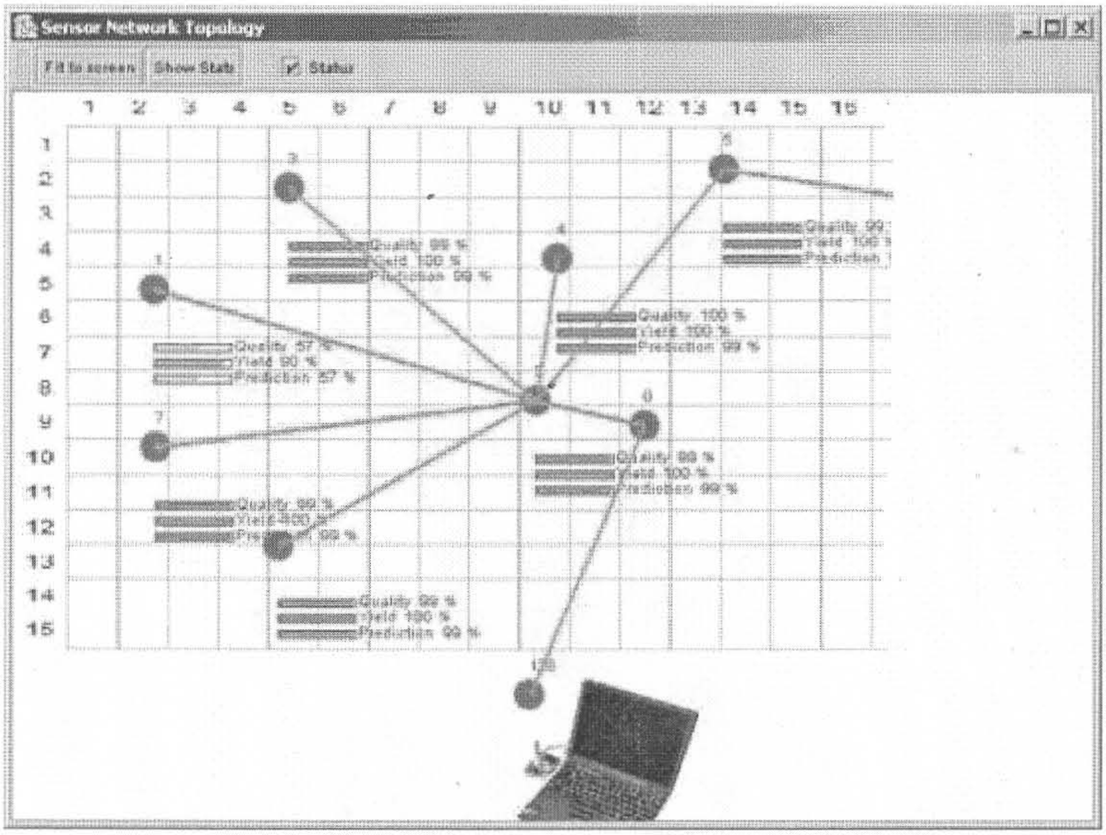


The next step in setting up our sensor network is to configure the serial port using the serial forwarder GUI that came with the motes. It is prominent that the serial port speed be set to 57600. This is the speed that best suites our sensor network (Crossbow, 2004).

Our sensor network is consistent no matter the number of sensors that can become attached. All sensors in our Plug and Play network connect wirelessly with the base sensor. You can think of the base sensor serving as a router or gateway. All of the other sensors, mobile and stationary, can be thought of as workstations that must communicate directly to the gateway to get out. The Surge GUI provides us with a graphical display to show our sensor network. From here we can see how many sensors are connected to our base station dedicated by their respective node numbers. We also can see each nodes quality, yield and prediction. Figure 3.2.4 is a topology of a sensor network connected to a server.

Figure 3.2.4

Sensor Network Topology



3.3 Implementation: Gathering Process

After configuring the serial port for the right line speed we're now able to run the Surge GUI for the implementation of our gathering data process. The Surge GUI is the program that actually starts the sensors on the motes to recording data which outputs to the command prompt. The Surge GUI has a feature that allows one to output the data real-time onto a text file. This can be invoked at the command prompt as follows where sensorstest is the text file that it outputs the data to.

```
Surge 125 > sensorstest
```

The text file sensorstest.txt can be found in the same directory as the executable for Surge 125. An example of the pound sign delimited text file is shown in Figure 3.3.1 where the first column represents the node number (Crossbow, 2004).

Figure 3.3.1

Sensortext Data

```
0#1#09/20/2005 05:12:11
PM#1127250731682#1844#126#0.5422993492407809#41#0#0#390#0#0#0.0#0#0#0.0#0#0#0.0#0#0#0.0#
0#0#0.0#102#83#53#107#0#0#

2#1#09/20/2005 05:12:14
PM#1127250734698#500#0#2.0#29#0#0#423#0#0#0.0#0#0#0.0#0#0#0.0#0#0#0.0#0#0#0.0#136#244#14
#204#255#255#
```

One good feature for gathering data in our plug and play sensor network is that a new sensor node can be powered on at any time. A new sensor node can connect at anytime with it's output being seen in the text file without stopping and restarting the Surge 125 executable.

3.4 Implementation: Storing Process

The storing process of the data is the next step in our project. Our main goal for storing data is efficiency. We want to find the best way to store the data without taking up a whole lot of space. We want to store it in a way that it can be queried and viewed easily. Most of all we want the process to be fast.

Our options for initial data storage are pretty open but we want to keep it a simple process as well. In order to convert the data from the text file to a table in a SQL database we first have to dump the whole text file to a table in SQL. The database table at this time is no special entity but merely a column for column clone of the text file.

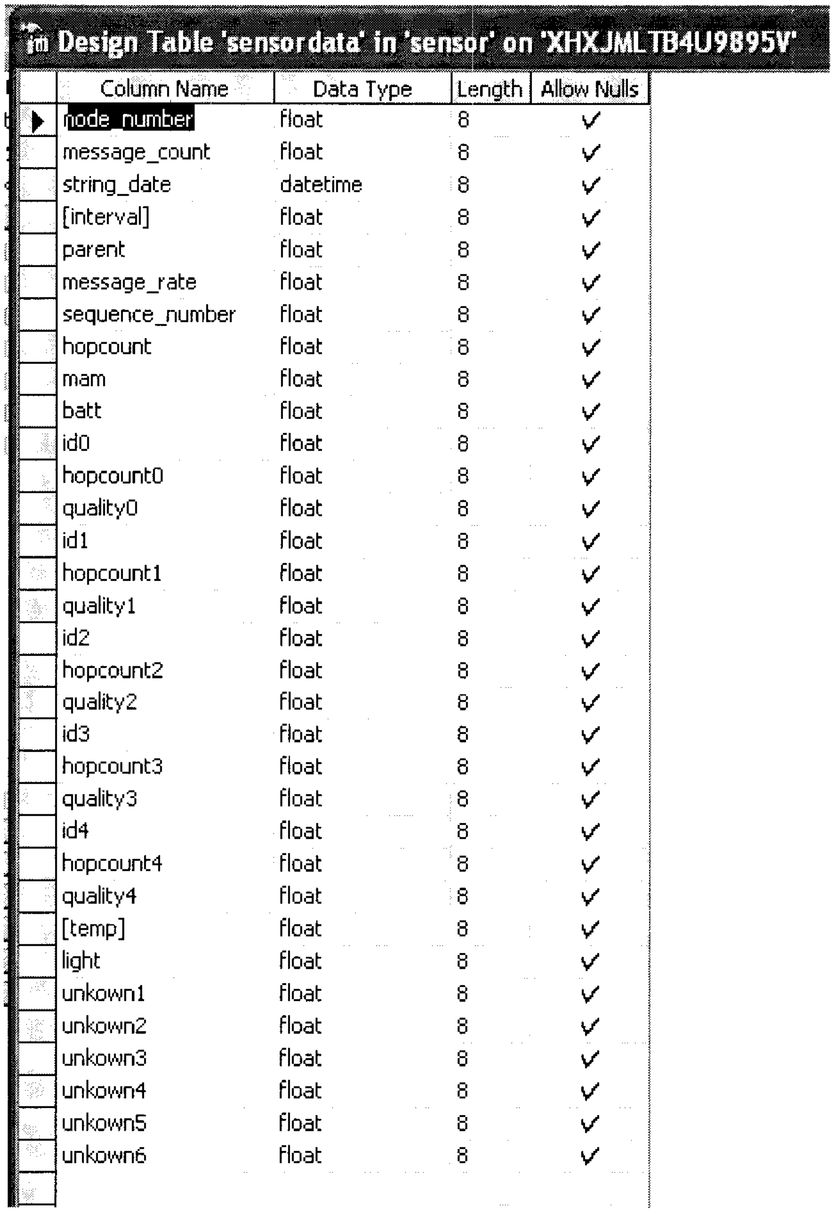
In the text file of the sensor data most of the columns have no interest to us, most of these columns are for doing specific tasks with the devices themselves. Our primary interest is the node number, date, time, temperature, light and acoustics. Here is the complete header column list for the text file.

```
node Number#Message Count#String Date#Time#interval#parent#Message Rate#Sequence
Number#hopcount#mAm#Batt#id 0#hopount 0#quality 0#id 1#hopount 1#quality 1#id 2#hopount
2#quality 2#id 3#hopount 3#quality 3#id 4#hopount 4#quality 4#Temp#Light#Acoustic
```

On the SQL side of things the initial super table (sensordata) we created for converting from the text file column for column to the database is made of 32 float columns and one datetime column. The structure of the table is shown in Figure 3.4.1.

Figure 3.4.1

Sensordata Database Table


Design Table 'sensordata' in 'sensor' on 'XHXJMLTB4U9895V'

Column Name	Data Type	Length	Allow Nulls
node_number	float	8	✓
message_count	float	8	✓
string_date	datetime	8	✓
[interval]	float	8	✓
parent	float	8	✓
message_rate	float	8	✓
sequence_number	float	8	✓
hopcount	float	8	✓
mam	float	8	✓
batt	float	8	✓
id0	float	8	✓
hopcount0	float	8	✓
quality0	float	8	✓
id1	float	8	✓
hopcount1	float	8	✓
quality1	float	8	✓
id2	float	8	✓
hopcount2	float	8	✓
quality2	float	8	✓
id3	float	8	✓
hopcount3	float	8	✓
quality3	float	8	✓
id4	float	8	✓
hopcount4	float	8	✓
quality4	float	8	✓
[temp]	float	8	✓
light	float	8	✓
unkown1	float	8	✓
unkown2	float	8	✓
unkown3	float	8	✓
unkown4	float	8	✓
unkown5	float	8	✓
unkown6	float	8	✓

After creating the sensordata table we are ready for the initial conversion from the text file to the database table. To accomplish this function our group did some querying research and found a command called Bulk Insert. A Bulk Insert takes a delimited text file and inserts all rows of the text file into a database table. The command works with different types of delimiters because it allows you to select the one you're going to use. Below is the syntax for the Bulk Insert command that is used (SQLteam.com, 2004).

```
"BULK INSERT sensordata FROM 'c:\thesis\sensorstest.txt' WITH
(FIELDTERMINATOR = '#')"
```

3.4.2 Implementation: Storing Complexity

One key concern about our Bulk Insert is speed. So in this matter our goal for a large scale sensor network is to do a Bulk Insert about every 8 to 10 seconds. The Bulk Insert query is a fast command but we feel the need to use it often not letting too many rows of text add up for this would tremendously slow the command down. For example, if we had ten sensor motes that each spit out a new reading every 8 seconds, after an hour that would be 4500 records that the Bulk Insert command would have to insert at one time. By reducing the time to every 8 seconds between using the Bulk Insert for new text data we cut down the number of inserts at one time to one insert per sensor and in our example 10 sensors = 10 inserts. Here is a Bulk Insert formula for our applications where N represents the number of sensors and T is time in seconds.

Bulk insert formula: N * [Floor(T/8)] = inserts

Bulk insert every 1 minute 10 * [Floor(60/8)] = 70 inserts

Bulk insert every 1 hour 10 * (3600/8) = 4500 inserts

Bulk insert every 8 seconds 10 * (8/8) = 10 inserts

In a large scale sensor network by keeping our Bulk inserts to one every 8 seconds our complexity stays at O(n) otherwise our complexity becomes time dependent as well.

O(n * T) Time dependent

For our application and simulation we can do a Bulk Insert every three minutes considering we only have two sensors. Three minutes is a small enough interval in our case that the Bulk Insert command does not slow down.

3.5 Implementation: Preprocessing Process

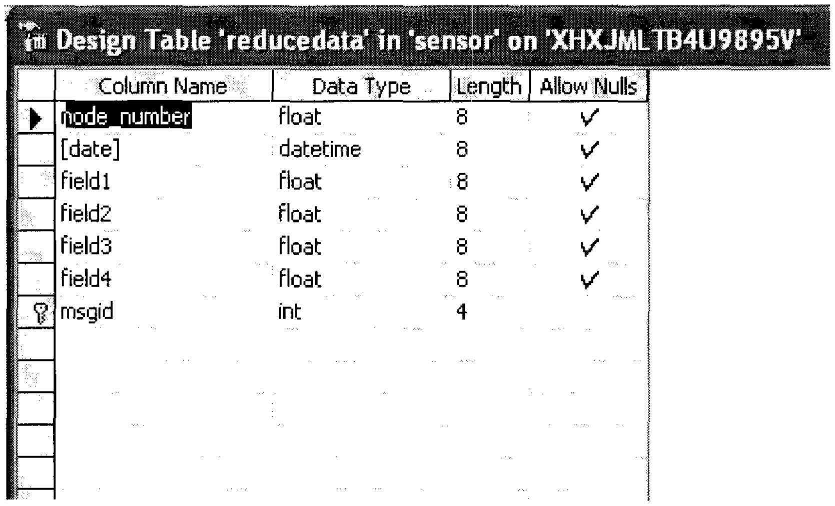
After converting the data from the text file to the initial SQL table there is a need for some preprocessing on the data. Our group decided to use a data mining technique that Mehmed Kantardzic talks about in his book (Kantardzic, 2003). The data mining technique we used is called feature reduction. Feature reduction is simply removing the columns of a data set that are irrelevant or useless. The goal of feature reduction is to deal with relevant features alone so that in the future if we choose to implement another data mining concept such as a learning algorithm we can achieve maximum performance with minimum measurement and processing efforts. In other words there would be less data so that the data mining algorithm could learn faster, the model would generalize better from data which results in a higher accuracy of a mining process. Also simpler results would make the output of the mining process easier to understand and use (Kantardzic, 2003).

Using feature reduction we removed columns that had no use for our project. We are only interested in columns that contained measurements, a date with a time and a node number. All other columns contain data for other purposes that don't have anything to do with our requirements. So these are the column features we removed.

After using feature reduction on the initial SQL table of 32 columns we end up with a table called reducedata that contains 7 columns. Six of the 7 columns we use from the initial table, one column we added as a primary key to the new table. Our resulting table is smaller, all of its information is relevant and easier to understand. The resulting table from using feature reduction is shown in Figure 3.5.1 where in the diagram Field1 = temperature, Field2 = Light and Field3 = Acoustic.

Figure 3.5.1

Reducedata Database Table


Design Table 'reducedata' in 'sensor' on 'XHXJMLTB4U9895V'

Column Name	Data Type	Length	Allow Nulls
node_number	float	8	✓
[date]	datetime	8	✓
field1	float	8	✓
field2	float	8	✓
field3	float	8	✓
field4	float	8	✓
msgid	int	4	

To put the data from the initial table to the reduced data table using the feature reduction concept we used a simple query. The query is shown below.

INSERT INTO reducedata SELECT node_number, string_date, light, unkown1, unkown2, unkown3 FROM sensordata

3.6 Unit Testing

To make sure that all of the major parts of the program (gathering, storing and preprocessing) work we tested the modules individually. Testing code through modules is also known as Unit Testing (Ellims, 2004). Our unit testing for our major three modules (gathering, storing and preprocessing) is done in three parts.

- Test the gathering process
- Test the storing process
- Test the preprocessing process

To validate the gathering process we make sure that we have a text file of data from the sensors. Now that a text file is visible we can validate the storing process when we convert the data from the text file to our initial table, sensordata. Once we have our sensordata table we can validate the preprocessing process by applying feature reduction and having the relevant data put into the reducedata table. All of these validations conclude our unit testing phase.

3.7 Integration

After implementation of the modules for gathering, storing and preprocessing its now time to focus on integrating the three parts. The environment we are working in is Visual Basic.NET (VB.NET). We use the Microsoft Visual Studio.NET 2003 to implement our VB.NET program. We also created a webpage that contains VB.NET scripts running on an ASP.NET page. Using ASP.NET allows us to be able to log onto our website and run queries simultaneously. Although the meat of our implementation runs through the VB.NET program, using ASP.NET gives us capabilities for the future to run many scripts using different languages.

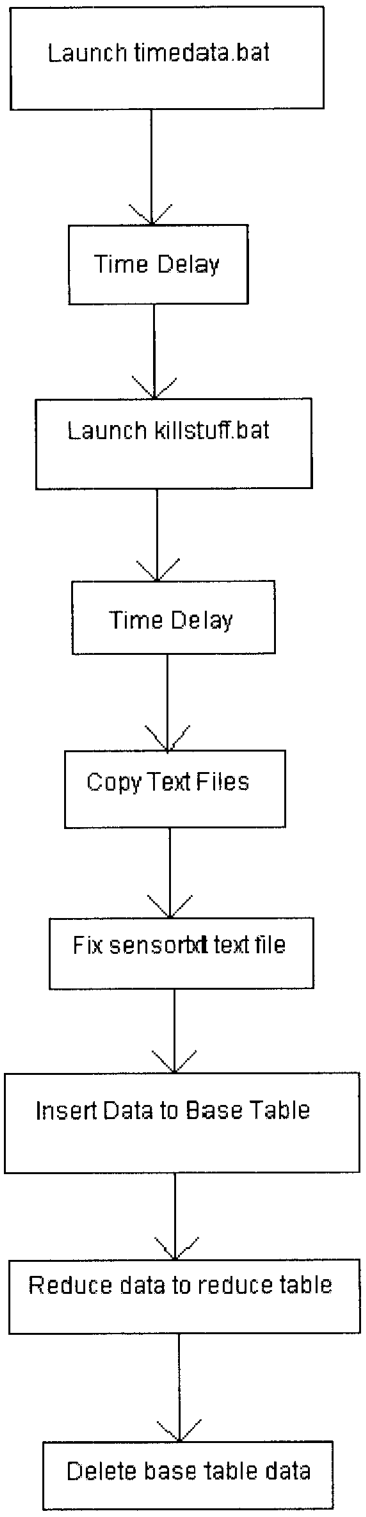
In our program the processing flow from start to finish (gathering to viewing processed data) is something that we want to keep simple. There are three parts to the integration of our implementation.

- timedata.bat (launches surge 125 and text file)
- VB.NET Program (used for queries and all other processes)
- killstuff.bat (kills the surge.exe process)

The killstuff.bat batch program includes the Command Line Process Utility from Beyondlogic.org. The program allows users to view and kill processes on your machine from the command prompt (Peacock, 2005). The reason we use this utility is that the process killing components built inside Visual Basic.NET will not stop the surge.exe process when we need.

To better understand how things are put together Figure 3.7 is a Flow Chart showing the logic steps of the program.

Figure 3.7 VB.NET Flowchart

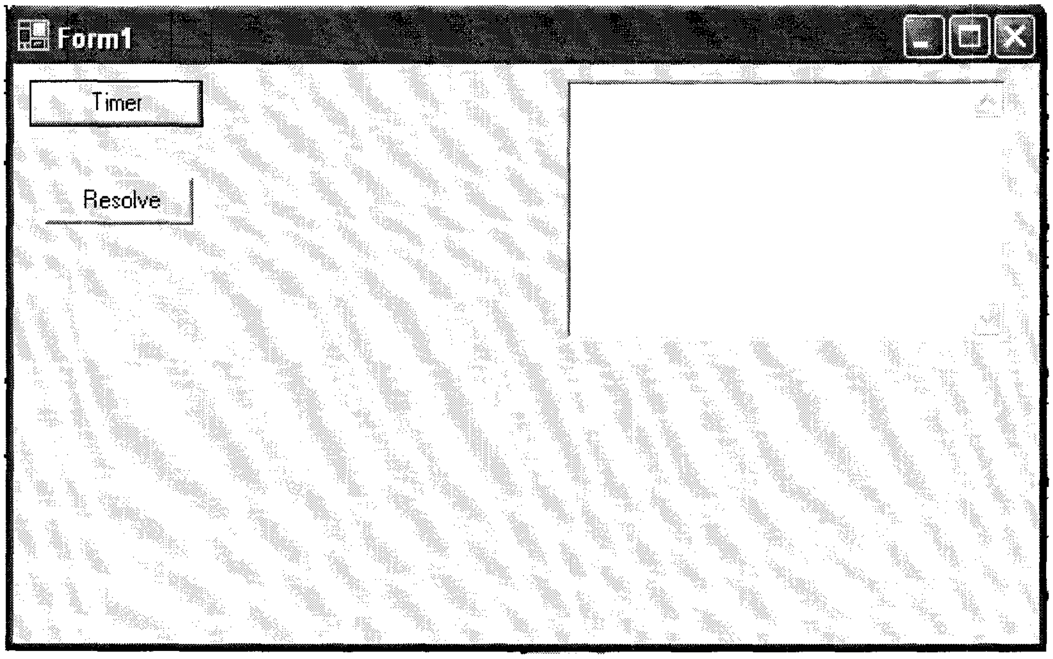


3.8 Compiled Program

Here is what our VB.NET program looks like. The program has two buttons (Timer and Resolve). The Timer button is the button that runs the flowchart found in figure 3.7. And the Resolve button resolves any issues with the text file if the program gives an error message and stops. The error message happens with the SQL driver, which is trying to insert data from the text file into the sensordata table. When the user sees the error message they can click on continue in the error message window and then they can click on the resolve button on our VB.NET program which will correct the issue of the error. The resolve button will filter out unwanted syntax in the text file. Figure 3.8 is a screenshot of the VB.NET program showing the Timer and Resolve buttons.

Figure 3.8

VB.NET Program



3.9 Summary

After looking over the requirements, the implementation method that the group chose covers the requirements of gathering, storing and preprocessing data. In the gathering process being able to output data to a text file through surge.exe is a quick way of getting data. It is also a simple way if one decided to add additional sensors while the program is running. In the storing process, getting the data to SQL through the Bulk Insert query command allows us to insert many lines of text from the text file into our database at one time. Since there are so many messages being output the Bulk Insert command inserts efficiently. In the preprocessing process being able to reduce features in the table really saves our group a lot of space. Now that all of the unwanted features our gone it is easier to run more data mining algorithms to increase the coverage of the sensors.

CHAPTER IV
RESULTS

4.1 Introduction

The project works as expected as far as gathering, storing and preprocessing the data. We used two sensors to conduct a simulation in a bedroom of an apartment over a time period of almost 24 hours. One sensor (Node 0) was placed on carpet floor and the other (Node 1) was placed in the nearby window seal. We conducted our implementation into segments throughout the 24 hour period. The segments were broke up so that we could cover different parts of the day such as: nighttime, morning, noon and evening.

The readings coming from the sensors are in a raw form (digital encoded form). They're not represented in a fashion at this point that a biologist or chemist could relate to. It is possible at this point to see increases and decreases in the output of the sensors making it easy for observers to draw observations.

Brian Carter (Carter, 2006) has an accepted paper titled "Message Transformation Services for Wireless Sensor Networks", at the 2006 International Conference on Wireless Networks (ICWN'06: June 26-29, Las Vegas, USA). In this paper he talks about transforming the encoded raw data from the sensors to a more visible format. The reason for his research on this topic was that manufactures want consumers to buy their extra programs to handle such conversions and transformations. So in the paper he focused on a scheme for transformation of the raw data and integrating other platforms at the message level (Carter and Ragade, 2006).

4.2 Space and Time Results

At the end of our simulation, it was found that our program inserted 1474 records into our reduced database table. The total space that SQL used over that time for our sensor database was 109mb. If our simulation hadn't been broken into segments it could as well have been three or four hundred megabytes. Each of the two sensors in our simulation output a message every 8 seconds for each segment. Table 4.2.1 gives a breakdown of the space and time for the data.

Table 4.2.1

Space and Time

Sensor Data Space and Time				
	Node #	Number of Records	Space used in mb's	Total Running Time min.
	0	743	54.94mb	99
	1	731	54.05mb	97
Totals	2	1474	109mb	196

4.3 Temperature Results

The temperature results at the end of our simulation were as expected. The sensor on the carpet floor (Node 0) stayed pretty constant throughout the 24 hour day which reflects insulation, whereas the sensor in the window seal (Node 1) had less insulation and fluctuated with the temperature outside. When it was night time node 1 seemed to output a cooler temperature than in the heat of the day around noon. Table 4.3.1 is an average read of both sensors around the time that the read is matched to.

Table 4.3.1

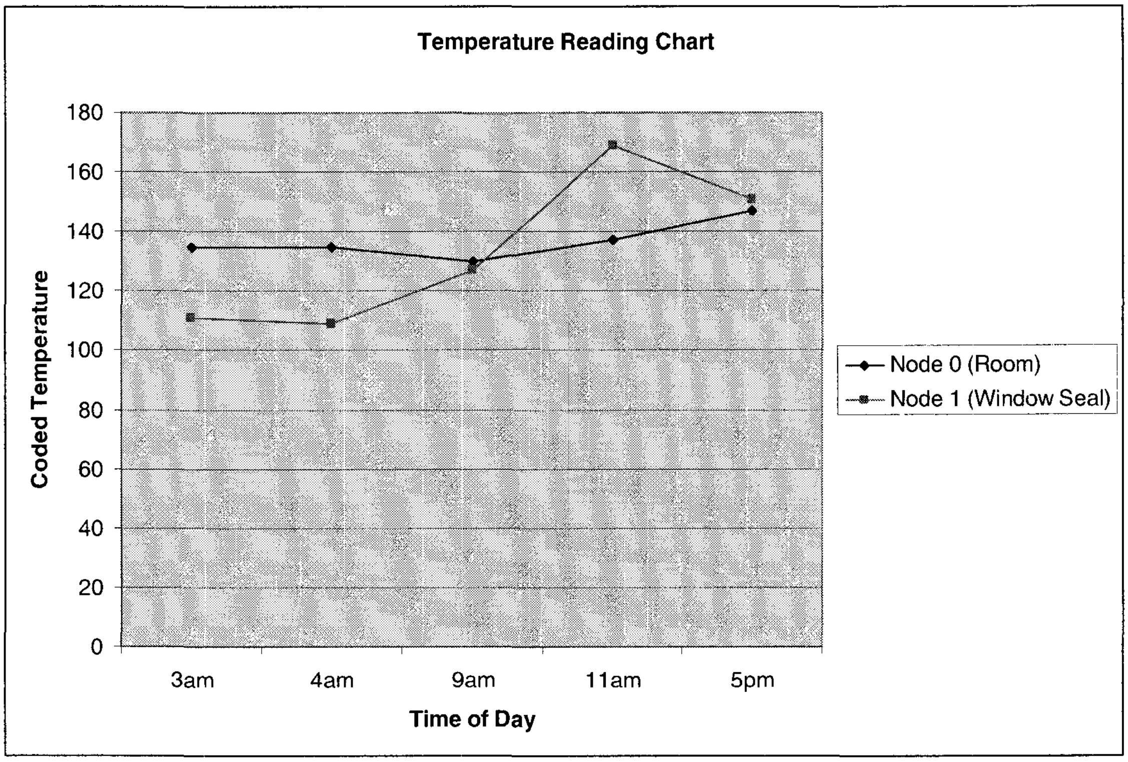
Temperature

		Sensor Data Readings Temperature (Field1)	
Time	Node 0 (Room)	Node 1 (Window Seal)	
3am	135	111	
4am	135	109	
9am	130	127	
11am	137	169	
5pm	147	151	

Figure 4.3.2 is a chart where you can see more of a constant line for Node 0 and more movement in Node 1.

Figure 4.3.2

Temperature Reading Chart



4.4 Light Results

The lighting results in our simulation were also as expected. At 3am in the room the only light that was on was coming from the computer monitor. At 4am all lights were turned off and it was pitch black. The sun rose around 7am making the light reading higher than it was at 3am or 4am. There was a higher light reading on Node 1 by the window seal than inside the room. This is the result of the sun refracting its rays through the window pane. Table 4.4.1 shows the data from the light sensors.

Table 4.4.1

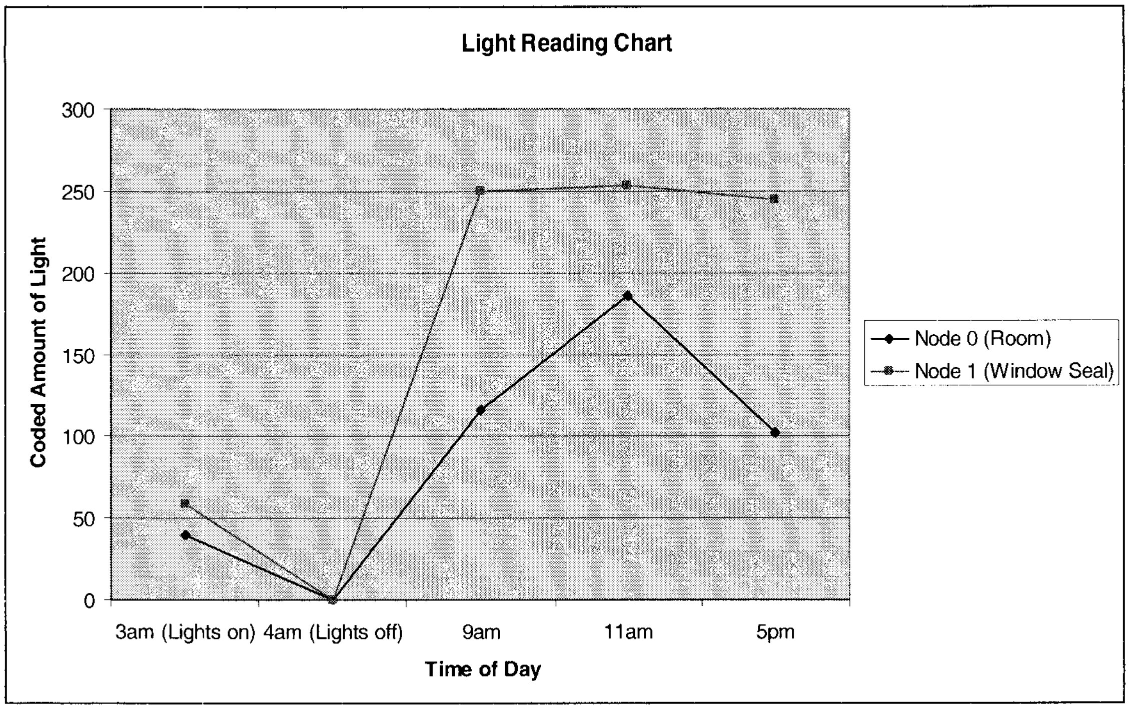
Light

	Sensor Data Readings Light (Field2)		
Time	Node 0 (Room)	Node 1 (Window Seal)	
3am (Lights on)	40	59	
4am (Lights off)	0	0	
9am	116	250	
11am	186	253	
5pm	102	245	

Figure 4.4.2 gives a good comparison of light readings between the two nodes. As you may notice in the diagram at 4am all lights were off.

Figure 4.4.2

Light Reading Chart



4.5 Acoustic Results

The acoustic or sound results at the end of the simulation represent a quiet place. In the results by being in the window seal, Node 1, heard less noise than being inside the room next to the computer and an ongoing floor fan. The averages of Node output around the following times are visible in Table 4.5.1.

Table 4.5.1

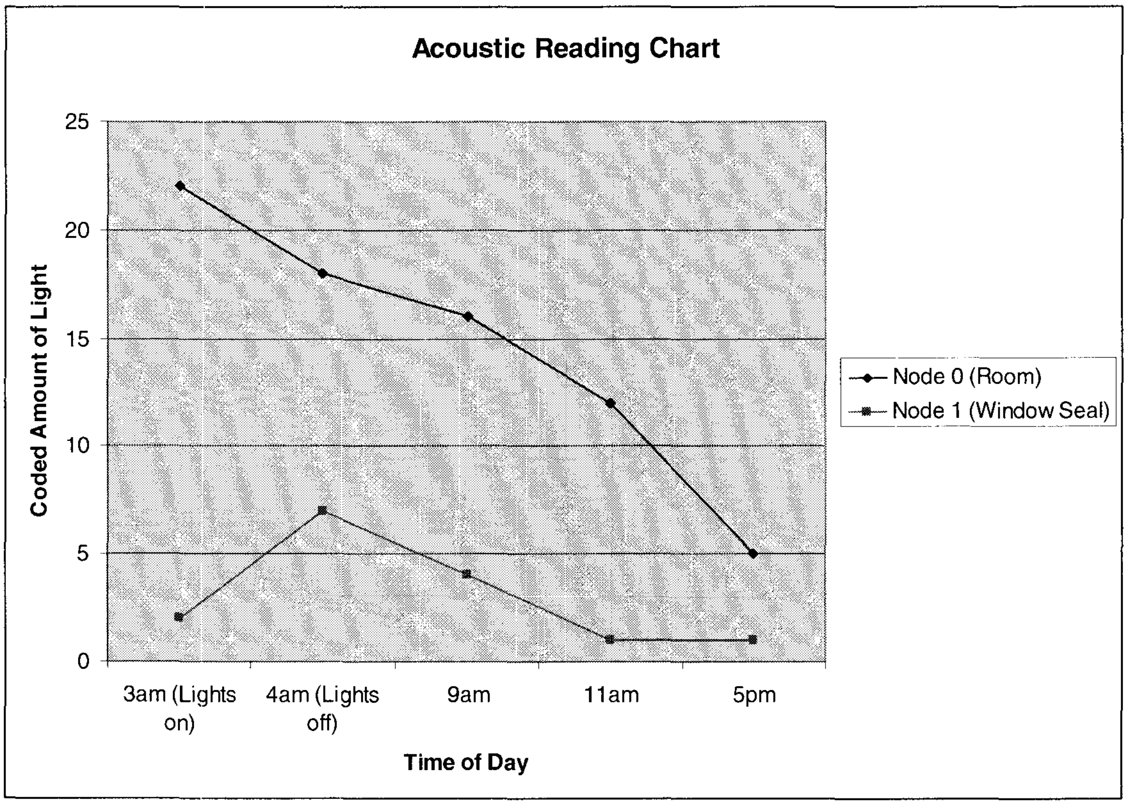
Acoustic

		Sensor Data Readings Acoustics (Field3)	
Time	Node 0 (Room)	Node 1 (Window Seal)	
3am (Lights on)	22	2	
4am (Lights off)	18	7	
9am	16	4	
11am	12	1	
5pm	5	1	

Figure 4.5.2 lets you clearly see that the outside noise was quieter than the inside noise. The only time they were close was when no one was present in the apartment around 5pm and the floor fan was turned off.

Figure 4.5.2

Acoustic Reading Chart



CHAPTER V
CONCLUSIONS AND FUTURE RESEARCH

5.1 Future Works for Improvement

Even though we got this project to work there are some improvements that could be applied to this implementation. One improvement could be based on the amount of space that the data output consumes. As we found out in our simulation two sensors in 24 hours generated over 100mb's in our database. And in the manner the simulation was conducted, the sensors didn't run continuously during that 24 hour period. For example, instead of focusing on recording every 8 second output from a sensor in each segment, one might be able to just grab the averages per segment and only store the average.

Another improvement that could be sought out in our implementation was how our VB.NET program started and stopped the output from the sensors. To accomplish this feat we kept killing and restarting the surge.exe process. It would be as efficient to simply be able to delete the text file that surge.exe outputs to and surge.exe being able to recreate the file on the fly but surge isn't able to do that which is a constraint.

Another improvement that Brian Carter is working on transforms the data at the message level which in turn allows many different sensor platforms a chance to integrate with our SNITTER PNP network. It might be able to integrate different sensor networks such as

Sensornet, Zigbee, surge Reliable, ect. Also this implementation is a lighter resource user than the surge program (Carter and Ragade, 2006).

5.2 Overall Project Outcome

The overall project of providing a service-oriented architecture to handle sensor data in real time as the information comes in with a focus on implementing an efficient way of gathering, storing and preprocessing data was a success. It was a nice project to be a part of. The project was implemented as planed all the way from the requirements to the results. The project could help implement many applications in different domains.

REFERENCES

Carter, Brian and Ragade, Rammohan K. (2006). Message Transformation Services for Wireless Sensor Networks (MTS-WSN). University of Louisville, Speed School of Engineering, Louisville, KY.

Carvalho, Hervaldo S. (2003). A General Data Fusion Architecture. Information Fusion, 2003. Proceedings of the Sixth International Conference. University of Rochester. Rochester, NY. Vol. 2. pp 1465-1472.

Chambrin, Marie-Christine (2001). Alarms in the intensive care unit: how can the number of false alarms be reduced? University of Lille. Lille, France. Crit Care. May 23, 2001. pp 184-188.

Crossbow. (2004) Wireless Sensor Networks: Getting Started Guide. Rev. B.

Ellims, Michael. (2004). Unit Testing in Practice. IEEE Computer Society. http://doi.ieeecomputersociety.org/10.1109/ISSRE.2004.44.

Nilsson, Staffan. (2001). Controller Area Network – CAN Information. www.algonet.se/~staffann/developer/CAN.htm.

Peacock, Craig. (2005). Command Line Process Viewer/Killer/Suspender for Windows NT/2000/XP. http://www.beyondlogic.org/solutions/processutil/processutil.htm.

Ratish, S. Udupa. (2005). USE Case Model Approach to Study and Understand Peoplesoft ERP Security. Department of Computer Engineering & Computer Science, University of Louisville. Louisville, KY.

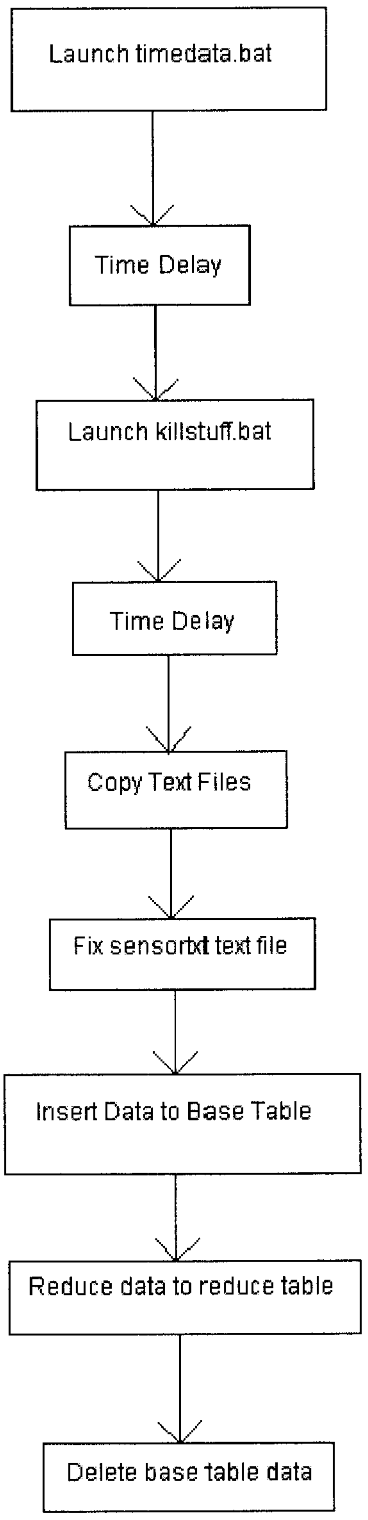
Siraj, Ambareen. (2004). Intrusion Sensor Data Fusion in an Intelligent Intrusion Detection System Architecture. System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on. Mississippi State University. January 5-8, 2004. pp 279-288.

SQLteam. (2004). Using BULK INSERT to Load a Text File. http://www.sqlteam.com/item.asp?ItemID=3207.

Virone, G. (2003). First Steps in Data Fusion between Multichannel Audio Acquisition and an Information System for Home Healthcare. Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE. La Tronche, France. Vol. 2 pp 1364-1367

Wang, Kuang-Ching. "Collaborative Sensing Using Sensors of Uncoordinated Mobility." Department of Electrical and Computer Engineering, Clemson University. Clemson, SC.

APPENDIX I. VB.NET Flowchart



VITA

Derek Massey

5102-3 Quail Ct.

Louisville, KY 40213

My name is Derek Ray Massey. The son of Billy Massey and Annie Savage Massey. I have one sister Deana Islas. I was born in Franklin, Kentucky on September 25th 1982. I attended Franklin Simpson High School from Fall 1996 to Spring 2006. During high school I took an interest to computers, programming and hardware. Thus I decided to come to the University of Louisville in Louisville, Kentucky. I came to the University of Louisville on a Woodford R. Porter academic scholarship and decided to major in Computer Engineering and Computer Science. As a freshman I became a member of the National Society of Black Engineers. During my college career I co-oped at Louisville Gas & Electric (LG&E) where I worked in the Project Engineering Department. There I assisted Engineers and kept up the department web page coded in HTML. My last two co-ops were done at Noahtek which was a IT consulting business. At Noahtek I developed server client applications using Clarion integrated with SQL. I also designed and implemented websites with ASP.NET integrated with SQL. As well as installing server client networks and desktop support. In December 2004 I received my B.S. in Computer Engineering & Computer Science and May 2006 I'm receiving my Master of Engineering in Computer Engineering & Computer Science. I'm currently employed by Jefferson County Public School System in their IT department.