

Jaypee University of Information Technology
Waknaghat, Distt. Solan (H.P.)

Learning Resource Center

CLASS NUM:

BOOK NUM.:

ACCESSION NO.: SP08023 / SP0812023

This book was issued is overdue due on the date stamped below. If the book is kept over due, a fine will be charged as per the library rules.

Due Date	Due Date	Due Date

Achieving QoS in Wireless Sensor Networks Using Gur Game Approach

PROJECT

Submitted in partial fulfillment
of the requirements for the degree of

BACHELOR OF TECHNOLOGY
Department of Computer Science Engineering

By

Priya Jain 081262

Rachit Singhal 081310

Under the supervision of

Dr.Nitin



JAYPEE UNIVERSITY OF
INFORMATION TECHNOLOGY



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT
SOLAN, HIMACHAL PRADESH, INDIA
2012

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT
SOLAN, HIMACHAL PRADESH**

Date:

CERTIFICATE

This is to certify that the work titled — **“Achieving Qos in Wireless Sensor Networks Using Gur Game Approach”**, submitted by **Priya Jain, Akhil Sahni, Rachit Singhal** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science Engineering to Jaypee University of Information Technology, Wagnaghat, Solan has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor *Nitin*

Name of Supervisor Dr. Nitin

Designation Associate Professor

Date ..31/.5/.2012.....

Acknowledgement

As we conclude our project with the God's grace, we have many people to thank; for all the help, guidance and support they lent us, throughout the course of our endeavor.

First and foremost, we are sincerely thankful to Dr.Nitin, our Project Guide, who has always encouraged us to put in our best efforts and deliver a quality and professional output. His methodology of making the system strong from inside has taught us that output is not the END of project. We really thank him for his time & efforts.

We are deeply indebted to all those who provided reviews and suggestions for improving the materials and topics covered in our project, and we extend our apologies to anyone we may have failed to mention.

Priya Jain

Enrollment no. 081262

CSE

Date .. 31/5/2012

Signature 

Rachit Singhal

Enrollment no. 081310

CSE

Date 31/05/2012

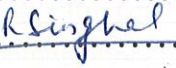
Signature 

Table of Content

Page

CERTIFICATE I	
ACKNOWLEDGEMENT II	
TABLE OF CONTENTS III	
LIST OF FIGURES IV	
ABSTRACT V	

Chapter – 1

Wireless Sensor Networks

1.1 Introduction	8
1.2 Individual Wireless Sensor Node Architecture	9
1.3 Wireless Sensor Networks Architecture.....	10
1.3.1 Star Network (Single Point-to-Multipoint).....	10
1.3.2 Mesh Network.....	11
1.3.3 Hybrid Star – Mesh Network.....	12
1.4 Challenges.....	13
1.5 Power Consideration in Wireless Sensor Networks.....	14
1.6 Applications of Wireless Sensor Networks.....	15
1.7 Future Developments	17
1.8 Summary.....	18

Chapter - 2

Literature Survey

2.1 Introduction	19
2.2 Introduction to Wireless Sensor Networks.....	19
2.2.1 Sensor Networks : An Overview.....	19
2.2.2 Theoretical and practical aspects of military wireless sensor networks...	19
2.2.3 Routing Techniques in Wireless Sensor Networks: A Survey.....	20
2.3 Quality of Service in Wireless Sensor Networks	20
2.3.1 Quality of Service in Wireless Sensor Networks	20
2.3.2 Simulation for Energy Expenditure Estimation of WSN for comparison of Network Routing Protocols.....	21
2.4 Introduction to Gur Game.....	21
2.4.1 QoS Control For Sensor Networks.....	21
2.4.2 Using Finite State Automata to Produce Self Optimization and Self Control.	22
2.5 Applying Distributed Algorithms on Sensor Networks.....	22
2.5.1 On Employing Distributed Algorithms and Evolutionary Algorithms in Managing Wireless Sensor Networks.....	22

2.5.2 Distributed Ants Algorithm.....	22
2.5.3 Gur Game Algorithm.....	23
2.5.4 A Dynamic Energy-Aware Algorithm for Self Optimizing.....	24
Wireless Sensor Networks	
2.6 Summary.....	24

Chapter - 3

Testbed, Experimental Setup and Simulation Outputs

3.1 Introduction.....	25
3.2 Selecting an Irregular area.....	25
3.3 Scanning the Irregular Area.....	28
3.4 Selecting a nxn matrix.....	29
3.5 Deployment of Sensors.....	30
3.6 Final Results and Conclusion.....	31

Chapter - 4

Insight Into the Program

4.1 Variables Descriptions.....	33
---------------------------------	----

Chapter - 5

Insight Into User Interface

5.1 Introduction.....	36
5.2 Status Panel.....	36
5.3 Drawing Panel.....	38
5.4 Control Panel.....	38
5.5 Snapshots of GUI.....	40
5.6 Gur Game Output.....	46

Chapter - 6

Future Directions

6.1 Future Scope	49
------------------------	----

APPENDIX – I51

APPENDIX – II63

REFERENCES71

List of Figures

Figures

Page

1.1: Wireless sensor node functional block diagram.....	10
1.2: Star network topology.....	11
1.3: Mesh network topology.....	12
1.4: Hybrid star-mesh network topology.....	13
1.5: Power consumption of a 5000-ohm strain gauge wireless sensor node.....	15
1.6: Industrial application of wireless sensors.....	16
1.7: Ben Franklin Bridge.....	17
1.8: Bridge strain data.....	18
2.1 Gur Game(Markov Chain).....	23

ABSTRACT

Sensor Networks has been one of the most dynamic technologies in the world for the last two decades. Recent technological improvements have made the deployment of small, inexpensive, low-power, distributed devices, which are capable of local processing and wireless communication, a reality. Such nodes are called as sensor nodes. Each sensor node is capable of only a limited amount of processing. But when coordinated with the information from a large number of other nodes, they have the ability to measure a given physical environment in great detail. Thus, a sensor network can be described as a collection of sensor nodes which co-ordinate to perform some specific action. Unlike traditional networks, sensor networks depend on dense deployment and co-ordination to carry out their tasks. But there are certain limitations with these sensor networks. One of the main limitations, which we tried to overcome in this project, is of energy conservation in WSN and to achieve best Quality of Service.

Our aim in this Project is to develop software, which simulates real life deployment of wireless sensors in such a way that it achieves best Quality of Service. We have used Gur Game Approach in our Project for achieving the required Quality of Service. We have used Java as the programming language in our project. The software designed is user friendly and any new user can work on it without any difficulty.

CHAPTER 1

WIRELESS SENSOR NETWORKS

1.1 INTRODUCTION

Sensors integrated into structures, machinery, and the environment, coupled with the efficient delivery of sensed information, could provide tremendous benefits to society. Potential benefits include: fewer catastrophic failures, conservation of natural resources, improved manufacturing productivity, improved emergency response, and enhanced homeland security. However, barriers to the widespread use of sensors in structures and machines remain. Bundles of lead wires and fiber optic "tails" are subject to breakage and connector failures. Long wire bundles represent a significant installation and long term maintenance cost, limiting the number of sensors that may be deployed, and therefore reducing the overall quality of the data reported. Wireless sensing networks can eliminate these costs, easing installation and eliminating connectors.

The ideal wireless sensor is networked and scaleable, consumes very little power, is smart and software programmable, capable of fast data acquisition, reliable and accurate over the long term, costs little to purchase and install, and requires no real maintenance.

Selecting the optimum sensors and wireless communications link requires knowledge of the application and problem definition. Battery life, sensor update rates, and size are all major design considerations. Examples of low data rate sensors include temperature, humidity, and peak strain captured passively. Examples of high data rate sensors include strain, acceleration, and vibration.

Recent advances have resulted in the ability to integrate sensors, radio communications, and digital electronics into a single integrated circuit (IC) package. This capability is enabling networks of very low cost sensors that are able to communicate with each other using low power wireless data routing protocols. A wireless sensor network (WSN) generally consists of a base station (or "gateway") that can communicate with a number of wireless sensors via a radio link. Data is collected at the wireless sensor node, compressed, and transmitted to the gateway directly or, if required, uses other wireless sensor nodes to forward data to the gateway. The transmitted data is then presented to the system by the gateway connection. The purpose of this chapter is to provide a brief technical introduction to wireless sensor networks and present a few applications in which wireless sensor networks are enabling.

1.2 Individual Wireless Sensor Node Architecture

A functional block diagram of a versatile wireless sensing node is provided in Figure 1.1. A modular design approach provides a flexible and versatile platform to address the needs of a wide variety of applications [2]. For example, depending on the sensors to be deployed, the signal-conditioning block can be re-programmed or replaced. This allows for a wide variety of different sensors to be used with the wireless sensing node. Similarly, the radio link may be swapped out as required for a given applications' wireless range requirement and the need for bidirectional communications. The use of flash memory allows the remote nodes to acquire data on command from a base station, or by an event sensed by one or more inputs to the node. Furthermore, the embedded firmware can be upgraded through the wireless network in the field.

The microprocessor has a number of functions including:

- 1) Managing data collection from the sensors
- 2) Performing power management functions
- 3) Interfacing the sensor data to the physical radio layer
- 4) Managing the radio network protocol

A key feature of any wireless sensing node is to minimize the power consumed by the system. Generally, the radio subsystem requires the largest amount of power. Therefore, it is advantageous to send data over the radio network only when required. This sensor event-driven data collection model requires an algorithm to be loaded into the node to determine when to send data based on the sensed event. Additionally, it is important to minimize the power consumed by the sensor itself. Therefore, the hardware should be designed to allow the microprocessor to judiciously control power to the radio, sensor, and sensor signal conditioner.

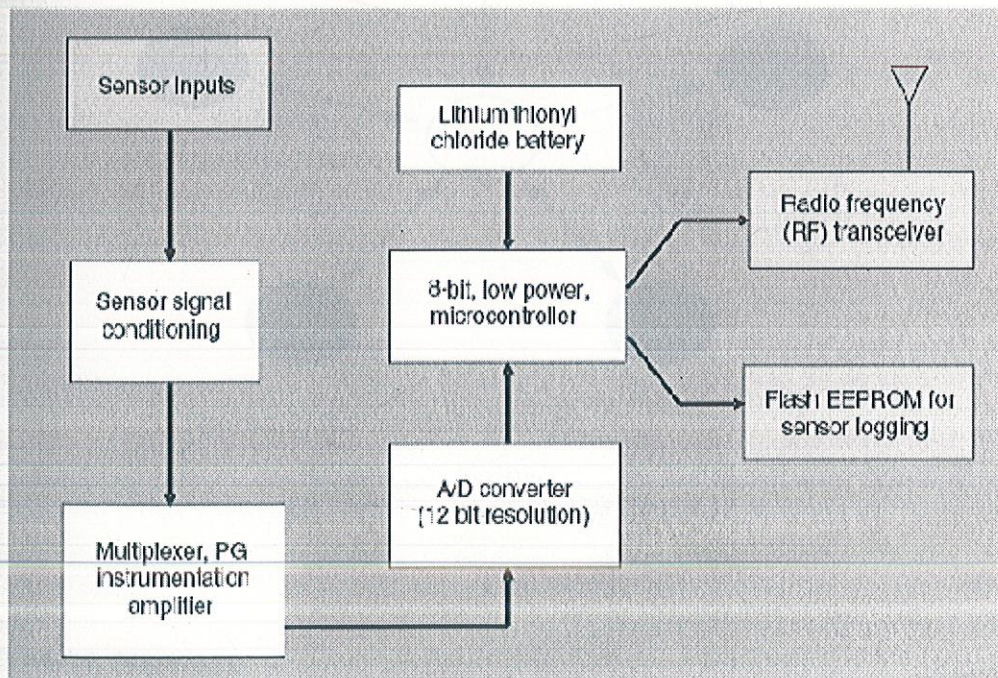


Figure 1.1: Wireless sensor node functional block diagram

1.3 Wireless Sensor Networks Architecture

There are a number of different topologies for radio communications networks. A brief discussion of the network topologies that apply to wireless sensor networks are outlined below.

1.3.1 Star Network (Single Point-to-Multipoint)

A star network (Figure 22.3.1) is a communications topology where a single base station can send and/or receive a message to a number of remote nodes. The remote nodes can only send or receive a message from the single base station, they are not permitted to send messages to each other. The advantage of this type of network for wireless sensor networks is in its simplicity and the ability to keep the remote node's power consumption to a minimum. It also allows for low latency communications between the remote node and the basestation. The disadvantage of such a network is that the basestation must be within radio transmission range of all the individual nodes and is not as robust as other networks due to its dependency on a single node to manage the network.

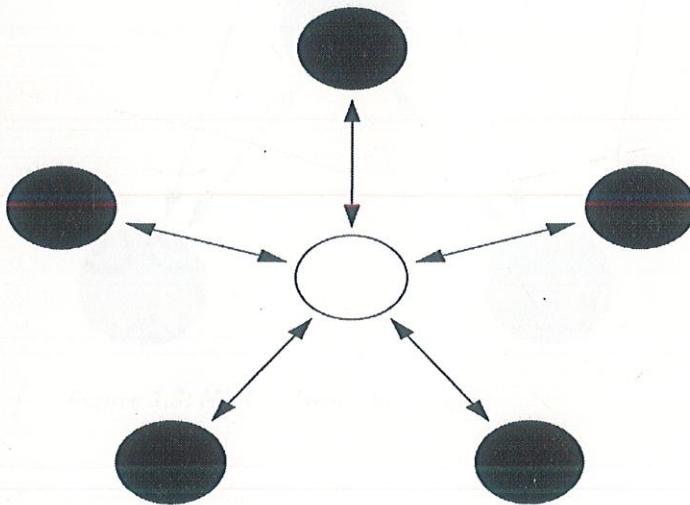


Figure 1.2: Star network topology

1.3.2 Mesh Network

A mesh network allows for any node in the network to transmit to any other node in the network that is within its radio transmission range. This allows for what is known as multihop communications; that is, if a node wants to send a message to another node that is out of radio communications range, it can use an intermediate node to forward the message to the desired node. This network topology has the advantage of redundancy and scalability. If an individual node fails, a remote node still can communicate to any other node in its range, which in turn, can forward the message to the desired location. In addition, the range of the network is not necessarily limited by the range in between single nodes, it can simply be extended by adding more nodes to the system. The disadvantage of this type of network is in power consumption for the nodes that implement the multihop communications are generally higher than for the nodes that don't have this capability, often limiting the battery life. Additionally, as the number of communication hops to a destination increases, the time to deliver the message also increases, especially if low power operation of the nodes is a requirement.

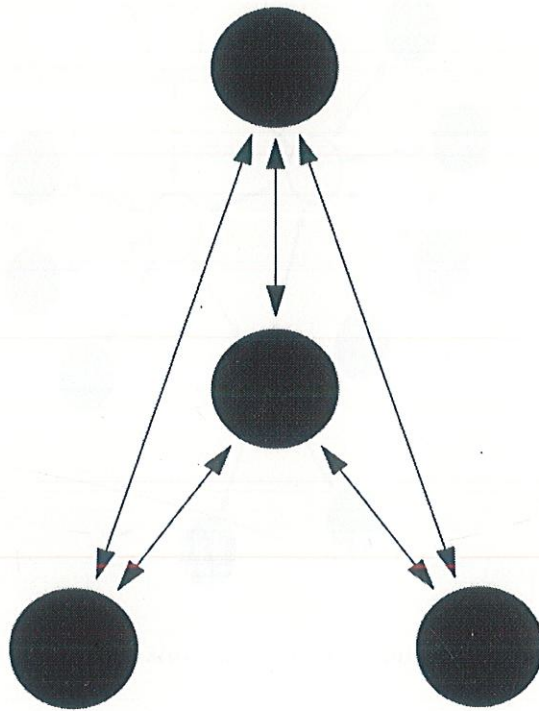


Figure 1.3: Mesh network topology

1.3.3 Hybrid Star – Mesh Network

A hybrid between the star and mesh network provides for a robust and versatile communications network, while maintaining the ability to keep the wireless sensor nodes power consumption to a minimum. In this network topology, the lowest power sensor nodes are not enabled with the ability to forward messages. This allows for minimal power consumption to be maintained. However, other nodes on the network are enabled with multihop capability, allowing them to forward messages from the low

power nodes to other nodes on the network. Generally, the nodes with the multihop capability are higher power, and if possible, are often plugged into the electrical mains line. This is the topology implemented by the up and coming mesh networking standard known as ZigBee.

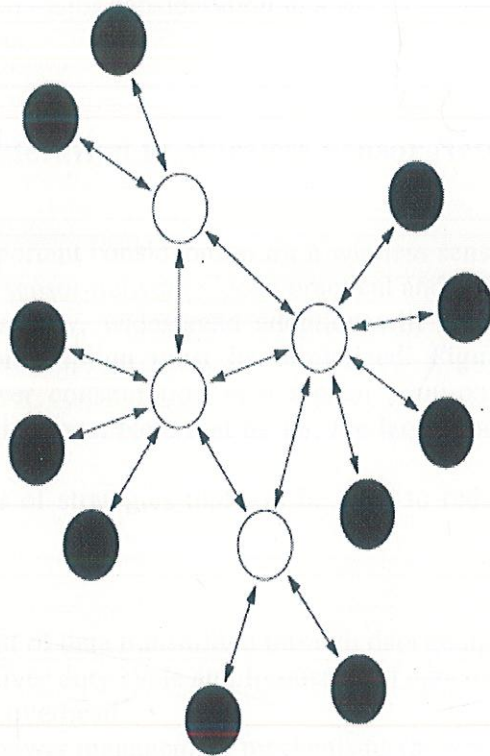


Figure 1.4: Hybrid star-mesh network topology

1.4 Challenges

In spite of the diverse applications, sensor networks pose a number of unique technical challenges due to the following factors:

>> **Ad hoc deployment:** Most sensor nodes are deployed in regions which have no infrastructure at all. A typical way of deployment in a forest would be tossing the sensor nodes from an aeroplane. In such a situation, it is up to the nodes to identify its connectivity and distribution.

>> **Unattended operation:** In most cases, once deployed, sensor networks have no human intervention. Hence the nodes themselves are responsible for reconfiguration in case of any changes.

>> **Untethered:** The sensor nodes are not connected to any energy source. There is only a finite source of energy, which must be optimally used for processing and communication. An interesting fact is that communication dominates processing in energy consumption. Thus, in order to make optimal use of energy, communication should be minimized as much as possible.

>> **Dynamic changes:** It is required that a sensor network system be adaptable to changing connectivity (for e.g., due to addition of more nodes, failure of nodes etc.) as well as changing environmental stimuli. Thus, unlike traditional networks, where the focus is on maximizing channel throughput or minimizing node deployment, the major consideration in a sensor network is to extend the system lifetime as well as the system robustness.

1.5 Power Consideration in Wireless Sensor Networks

The single most important consideration for a wireless sensor network is power consumption. While the concept of wireless sensor networks looks practical and exciting on paper, if batteries are going to have to be changed constantly, widespread adoption will not occur. Therefore, when the sensor node is designed power consumption must be minimized. **Figure 1.5** shows a chart outlining the major contributors to power consumption in a typical 5000-ohm wireless strain gage sensor node versus transmitted data update rate. Note that by far, the largest power consumption is attributable to the radio link itself.

There are a number of strategies that can be used to reduce the average supply current of the radio, including:

- Reduce the amount of data transmitted through data compression and reduction.
- Lower the transceiver duty cycle and frequency of data transmissions.
- Reduce the frame overhead.
- Implement strict power management mechanisms (power-down and sleep modes).
- Implement an event-driven transmission strategy; only transmit data when a sensor event occurs.

Power reduction strategies for the sensor itself include:

- Turn power on to sensor only when sampling.
- Turn power on to signal conditioning only when sampling sensor.
- Only sample sensor when an event occurs.
- Lower sensor sample rate to the minimum required by the application.

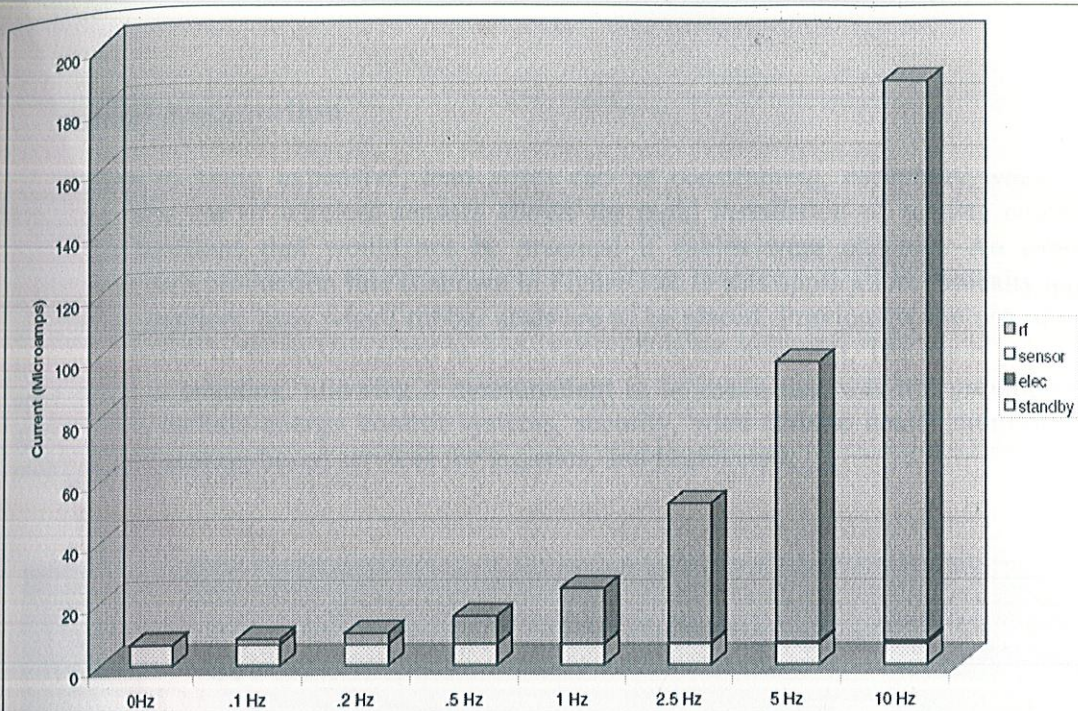


Figure 1.5: Power consumption of a 5000-ohm strain gauge wireless sensor node

1.6 Applications of Wireless Sensor Networks

Structural Health Monitoring – Smart Structures

Sensors embedded into machines and structures enable condition-based maintenance of these assets. Typically, structures or machines are inspected at regular time intervals, and components may be repaired or replaced based on their hours in service, rather than on their working conditions. This method is expensive if the components are in good working order, and in some cases, scheduled maintenance will not protect the asset if it was damaged in between the inspection intervals. Wireless sensing will allow assets to be inspected when the sensors indicate that there may be a problem, reducing the cost of maintenance and preventing catastrophic failure in the event that damage is detected. Additionally, the use of wireless reduces the initial deployment costs, as the cost of installing long cable runs is often prohibitive.

In some cases, wireless sensing applications demand the elimination of not only lead wires, but the elimination of batteries as well, due to the inherent nature of the machine, structure, or materials under test. These applications include sensors mounted on continuously rotating parts, within concrete and composite materials, and within medical implants.

Industrial Automation

In addition to being expensive, lead wires can be constraining, especially when moving parts are involved. The use of wireless sensors allows for rapid installation of sensing equipment and allows access to locations that would not be practical if cables were attached. An example of such an application on a production line is shown in Figure 1.6. In this application, typically ten or more sensors are used to measure gaps where rubber seals are to be placed. Previously, the use of wired sensors was too cumbersome to be implemented in a production line environment. The use of wireless sensors in this application is enabling, allowing a measurement to be made that was not previously practical. Other applications include energy control systems, security, wind turbine health monitoring, environmental monitoring, location-based services for logistics, and health care.

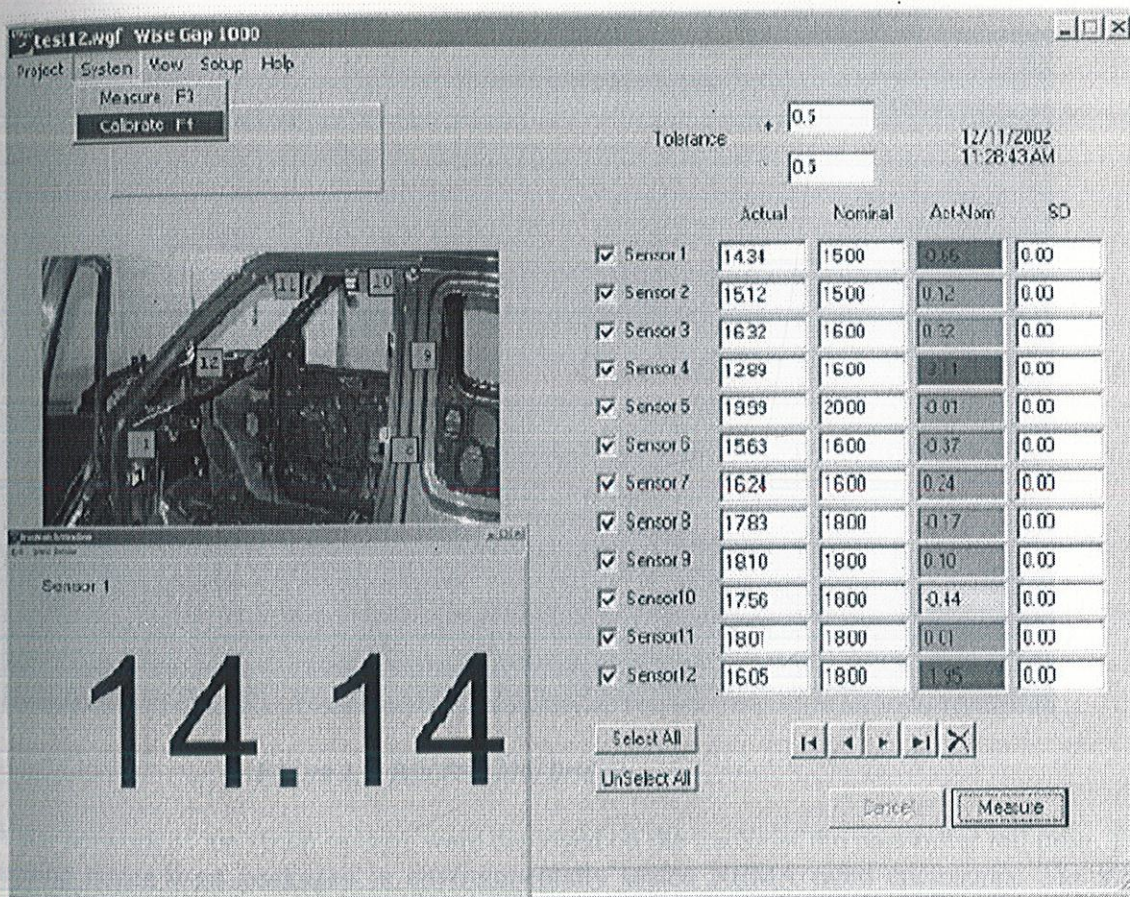


Figure 1.6: Industrial application of wireless sensors

Application Highlight – Civil Structure Monitoring

One of the most recent applications of today's smarter, energy-aware sensor networks is structural health monitoring of large civil structures, such as the Ben Franklin Bridge (Figure 1.7), which spans the Delaware River, linking Philadelphia and Camden, N.J. The bridge carries automobile, train and pedestrian traffic. Bridge officials wanted to monitor the strains on the structure as high-speed commuter trains crossed over the bridge.



Figure 1.7: Ben Franklin Bridge

A star network of ten strain sensors were deployed on the tracks of the commuter rail train. The wireless sensing nodes were packaged in environmentally sealed NEMA rated enclosures. The strain gauges were also suitably sealed from the environment and were spot welded to the surface of the bridge steel support structure. Transmission range of the sensors on this star network was approximately 100 meters.

The sensors operate in a low-power sampling mode where they check for presence of a train by sampling the strain sensors at a low sampling rate of approximately 6 Hz. When a train is present the strain increases on the rail, which is detected by the sensors. Once detected, the system starts sampling at a much higher sample rate. The strain waveform is logged into local Flash memory on the wireless

sensor nodes. Periodically, the waveforms are downloaded from the wireless sensors to the base station. The base station has a cell phone attached to it which allows for the collected data to be transferred via the cell network to the engineers' office for data analysis. This low-power event-driven data collection method reduces the power required for continuous operation from 30 mA if the sensors were on all the time to less than 1 mA continuous. This enables a lithium battery to provide more than a year of continuous operation. Resolution of the collected strain data was typically less than 1 micro strain. A typical waveform downloaded from the node is shown in Figure 1.8. Other performance specifications for these wireless strain sensing nodes have been provided in an earlier work .

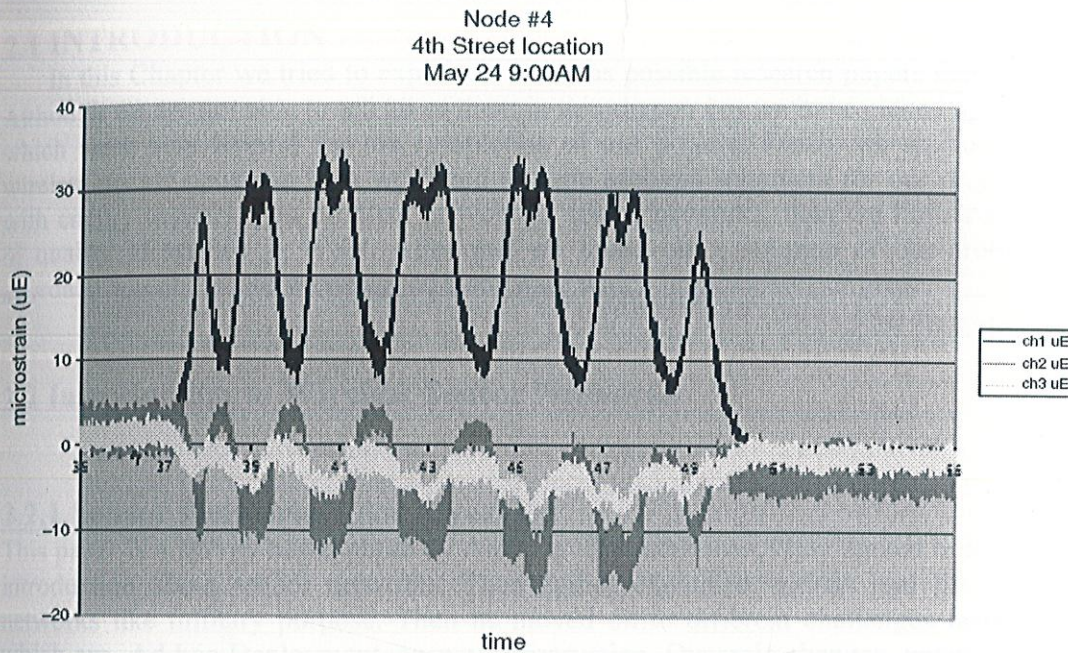


Figure 1.8: Bridge strain data

1.7 Future Developments

The most general and versatile deployments of wireless sensing networks demand that batteries be deployed. Future work is being performed on systems that exploit piezoelectric materials to harvest ambient strain energy for energy storage in capacitors and/or rechargeable batteries. By combining smart, energy saving electronics with advanced thin fi lm battery chemistries that permit infinite recharge cycles, these systems could provide a long term, maintenance free, wireless monitoring solution .

1.8 Summary

Wireless sensor networks are enabling applications that previously were not practical. As new standards-based networks are released and low power systems are continually developed, we will start to see the widespread deployment of wireless sensor networks.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

In this Chapter we tried to explain as many as possible research papers that we have gone through. Although we are not able to tell all of them in this chapter but we have picked up some important papers which were very helpful for the completion of our project. Firstly we studied about introduction to wireless sensor networks then we found out one problem statement for our project. Basically we dealt with energy conservation problem in wireless sensor networks, then we further moved on to achieving of quality of service in WSN. After that we found out a solution of our problem as GUR GAME approach and at the end we designed a software which implements gur game.

2.2 Introduction to Wireless Sensor Networks

2.2.1 Sensor Networks : An Overview

This paper is a survey based paper by Archana Bharathidasan, Vijay Anand. In this paper author gave a introduction about sensor networks. Then author explained various real life applications of sensor networks like military purpose. Then he moved on to different challenges faced by sensor networks which are Ad hoc Deployment, energy conservation, Dynamic changes, untethered, localization. Next he explained about different architectures of sensor networks. After that he again explained in deep about energy conservation problem in sensor networks. Next author discussed about localization techniques and different routing techniques. In localization technique we got the idea of dividing a field into square grids which we used and implemented afterwards in our project.

2.2.2 Theoretical and practical aspects of military wireless sensor networks

This paper is basically based on applications of wireless sensor networks and problems faced while deploying these sensors in real life examples. This is written by Michael Winkler and Kester Hughes. Author explained about Wireless sensor networks can be used by the military for a number of purposes such as monitoring militant activity in remote areas and force protection. Being equipped with appropriate sensors these networks can enable detection of enemy movement, identification of enemy force and analysis of their movement and progress. The focus of this article networks. Based on the main networking characteristics and military use-cases, insight into specific military requirements is given in order to facilitate the reader's understanding of the operation of these networks in the near to medium term (within the next three to eight years). The article structures the evolution of military sensor networking devices by identifying three generations of sensors along with their capabilities. Existing developer solutions are presented and an overview of

some existing tailored products for the military environment is given. The article concludes with an analysis of outstanding engineering and scientific challenges in order to achieve fully flexible, security proved, ad hoc, self-organizing and scalable military sensor networks. is on the military requirements for flexible wireless sensor.

2.2.3 Routing Techniques in Wireless Sensor Networks: A Survey

In this paper authors are Jamal N. Al-Karaki, Ahmed E. Kamal. They tried to explain about the different challenges faced by wireless sensor networks and how by using different routing techniques we can overcome these difficulties. They also explains about

Wireless Sensor Networks (WSNs) consist of small nodes with sensing, computation, and wireless communications capabilities. Many routing, power management, and data dissemination protocols have been specifically designed for WSNs where energy awareness is an essential design issue. The focus, however, has been given to the routing protocols which might differ depending on the application and network architecture. In this paper, we present a survey of the state-of-the-art routing techniques in WSNs. We first outline the design challenges for routing protocols in WSNs followed by a comprehensive survey of different routing techniques. Overall, the routing techniques are classified into three categories based on the underlying network structure: flat, hierarchical, and location-based routing. Furthermore, these protocols can be classified into multipath-based, query-based, negotiation-based, QoS-based, and coherent-based depending on the protocol operation. We study the design tradeoffs between energy and communication overhead savings in every routing paradigm. We also highlight the advantages and performance issues of each routing technique. The paper concludes with possible future research areas.

2.3 Quality Of Service in Wireless Sensor Networks

2.3.1 Quality of Service in Wireless Sensor Networks

This paper is given by Hwee Xian-Tan. Author tried to explain about what actually Quality of Service and how it comes into paly in wireless sensor networks. He explains about Wireless Sensor Networks (WSNs) comprise of groups of tiny sensor nodes that are deployed for collaborative missions such as environmental monitoring, target tracking and surveillance. Due to the miniature size of the nodes, they are typically deployed in large numbers and communicate via multiple hops through a wireless shared communication channel. The successful implementation of such networks is dependent on the enabling technologies (such as digital electronics and wireless communications), as well as the provisioning of Quality of Service (QoS) in the network. While there have been many efforts in QoS provisioning in traditional networks such as the Internet and Mobile Ad Hoc Networks (MANETs), these networks have very different characteristics from that of WSNs. Consequently, the QoS models and protocols that have been designed for the Internet and MANETs cannot be directly applied to WSNs. In this paper, we look at some of the existing QoS mechanisms in the networking literature, and the inherent characteristics of WSNs which make it challenging to provision for QoS in the network. We then identify some key performance metrics for WSN QoS and outline some mechanisms to achieve QoS in the sensor network. Finally, we propose **WISER** – a framework to enhance QoS in **Wireless SENsoR** networks.

2.3.2 Simulation for Energy Expenditure Estimation of WSN for comparison of Network Routing Protocols

This paper is given by Sanjeev Gupta, Sanjeev Sharma, Mayank Sharma. They tried to explain about the energy expenditure estimation of WSN for different routing protocols.

They also explained that the low power wireless communication technology has enabled the development of next generation low cost wireless sensor network. One important performance index of routing mechanism in sensor networks is sensor life span. How we utilize limited energy in wireless sensor network, directly affects the lifetime and cost of the network. As sensors are energy constrained by batteries and located far away from the access point. A sensor dies if its battery dies. Since the radio transmission consumes a lot of energy, we propose network routing protocols to save energy to prolong the sensor lifespan.

2.4 Introduction to Gur Game

2.4.1 QoS Control For Sensor Networks

This paper is written by Ranjit Iyer and Leonard Kleinrock. In this paper the author tried to explain how to control quality of service in sensor networks using the Gur Game approach.

They also explain that sensor networks are distributed networks made up of small sensing devices equipped with processors, memory, and short-range wireless communication. They differ from conventional computer networks in that they have severe energy constraints, redundant low-rate data, and a plethora of information flows. Many aspects of sensor networks, such as routing, preservation of battery power, adaptive self-configuration, etc., have already been studied in previous papers. However, to the best knowledge of the authors, the area of sensor network quality of service (QoS) remains largely open. This is a rich area because sensor deaths and sensor replenishments make it difficult to specify the optimum number of sensors (this being the service quality that we address in this paper) that should be sending information at any given time. In this paper we present an amalgamation of QoS feedback and sensor networks. We use the idea of allowing the base station to communicate QoS information to each of the sensors using a broadcast channel and we use the mathematical paradigm of the Gur Game to dynamically adjust to the optimum number of sensors. The result is a robust sensor network that allows the base station to dynamically adjust the resolution of the QoS it receives from the sensors, depending on varying circumstances.

2.4.2 Using Finite State Automata to Produce Self Optimization and Self Control

This paper is written by Brian Tung and Leonard Kleinrock. In this paper they give a deep overview of the Gur game and how it can be used for self optimization and self control in wireless sensor networks. They also discuss about that a simple game provides a framework within which agents can spontaneously self organize. In this paper the author presents this game and develops basic theory underlying this robust method for distributed combination based on this game. This method makes use of finite state automata, one associated with each agent, which guides the agent. They give a new general method of analysis of these systems which previously had been studied only in limited cases. They also provide a physical example which should hint at the problems resolvable using this method.

2.5 Applying Distributed Algorithms on Sensor Networks

2.5.1 On Employing Distributed Algorithms and Evolutionary Algorithms in Managing Wireless Sensor Networks

This paper is written by Syed Imran Nayer and Hesham H. Ali. This paper presents two new algorithms for managing the activation process in sensor networks. The objective of the algorithms is to maximize the coverage of the sensor area while conserving energy consumed by sensor nodes. This is achieved via carefully activating/deactivating the sensors while maximizing coverage. The two algorithms are based on evolutionary technique and the distributed Ant Algorithm (AA). The results of employing each algorithm is evaluated by a comparative study with random Gur Game based approach, introduced recently in other papers. Obtained results showed that the distributed algorithm performs best with regard to coverage and convergence time. The Constructive Genetic Algorithm (CGA) performs best under dynamic environments when the structure of the network changes dynamically.

2.5.2 Distributed Ants Algorithm

WSN is treated as a graph i.e. All the sensor nodes are treated as the vertex of the graph.

1. Randomly activate/ deactivate nodes.
2. Randomly select agents i.e ants on different nodes.
3. Calculate **Violation** = (same status nodes) – (different status nodes) for each vertex.
4. Each ant move to its adjacent vertex, changes its status.
5. Recalculate the violation for itself and adj. vertices.
6. Once each ant has moved, repeat the process until coverage to near optimal solution(QOS. =0.4) is attained.

2.5.3 Gur Game Algorithm

1. Gur Game can be described as a function which is random but working in a greedy fashion to achieve global optimization.
 2. This is a trial and error method.
 3. Key in Gur Game is reward function which is responsible to measure the performance of whole system.
- N –Players(sensors nodes) and 1 Refree(Base Station or Gateway).
 - In each iteration each sensor sends a stimuli to the Base Station.
 - On receiving the stimuli Base Station calculates F(fraction), which is the no. of active sensors to the total no of sensors ($0 \leq F \leq 1$).
 - Here the fraction F is denoted by X_t .

REWARD FUNCTION :

$R(X_t) = 0.2 + 0.8e^v$ Where $v = (-0.002) * (X_t * 100 - 40)^2$
 X_t is the ratio between regions covered and active sensor nodes.

$X = (\text{\# of regions covered by active sensors}) / (\text{\# of active sensors})$

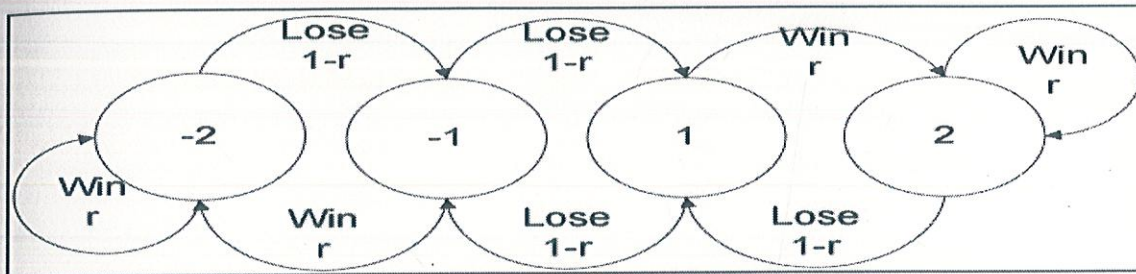


Fig 2.1 Gur Game(Markov Chain)

Reward probability : $R(X_t)$ – is calculated as above

- Nodes are rewarded with prob. r or penalized with the prob. $(1-r)$.
- Iteration is carried until acceptable level of redundancy $X=0.4$ is achieved.

2.5.4 A Dynamic Energy-Aware Algorithm for Self Optimizing Wireless Sensor Networks

This paper is written by Syed Imran Nayer and Hesham H. Ali. They told that Wireless sensor networks are distributed ad-hoc networks that consist of small sensor devices collecting and transmitting data. In such devices, optimizing power consumption due to sensing and routing data is an open area for researchers. In earlier works, approaches based on the Gur game, ant algorithms and evolutionary algorithms were introduced. These approaches have been employed to control the number of active sensors in wireless networks required to guarantee high performance parameters while taking energy conservation into consideration. These studies though ignored the coverage redundancy, which is a key issue in sensor networks. In this paper, we use the mathematical paradigm of the Gur Game in order to achieve the optimal assignment of active sensors while maximizing the number of regions covered by sensor nodes. We use a dynamic clustering algorithm that employs the concept of connected dominating sets. The proposed algorithm addresses this problem by playing Gur Game among the cluster nodes. We also further develop the earlier developed ants algorithm and genetic algorithm to take into consideration

node addition and deletion. A simulation study was used to test the proposed algorithms under different network scenarios.

2.6 Summary

These were certain important papers which helped us a lot in completing our project. Information provided by these papers was very helpful for us.

CHAPTER 3

Testbed, Experimental Setup and Simulation Outputs

3.1 Introduction:

We have used Java Technology (i.e. JDK 1.6) for simulating and this software is running on top of the HP System, running with Windows Vista Home Basic operating system. We have made use of Java Applets and all the panels have been incorporated on the same applet. The tool can be used in the following different phases:

- Selecting an irregular area on the screen by means of an irregular polygon. The area selected will be shown in blue.
- Scanning the whole area selected with the help of polygon through a function called Get Area which selects all the homogeneous grids, i.e. the grids which totally lie inside the selection. The area selected will be shown in red.
- Once the grids have been selected, we can select a $n \times n$ grid structure inside these homogeneous grids where the sensors are needed to be deployed. The area selected will be shown in turquoise blue and the number of grids selected will be shown in the Area field.
- Once this process of selecting the required grid size is finished, we can increase the number of sensors in the field given and use the Deploy function to deploy them.
- The tool will separately ask for each sensor that whether it is active or not and based upon our input the Gur Game Algorithm will run in the background finally telling whether QoS was achieved or not.

3.2 Selecting an Irregular Area :

The tool is designed to make the process of selecting grids simple and more realistic and so that we can relate it to the real world. So the first step is to select an approximate area where we wish to deploy our sensors. The grids on the screen would themselves divide the selected area.

The user can click on the Grids option any time while using the tool to divide the screen into grids or to highlight the division if it's already divided. Clicking on the Irregular Polygon button sets the user to select an approximate area on the screen in the form of an irregular polygon. The user just needs to click on the screen where the vertices of the polygon are required and the polygon will start taking its shape from the third vertex onwards as shown in Fig.1.

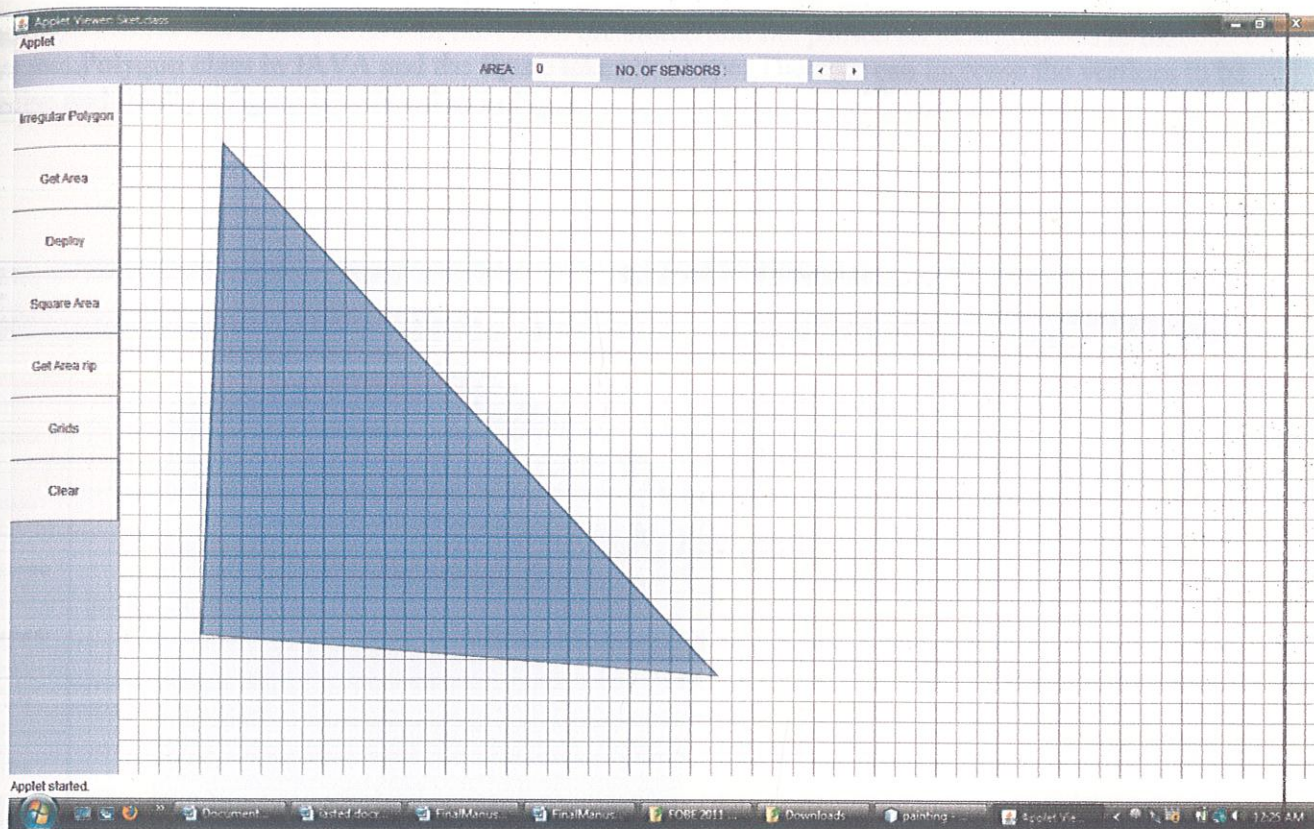


Figure 1 Irregular Polygon after 3 clicks

The logic behind this is simple and all the clicks are stored and the passed on as arguments to the `java.awt.Polygon` class in JAVA and the figure takes its shape. The user can increase the vertices to be precise and finally a figure is selected as in Fig.2.

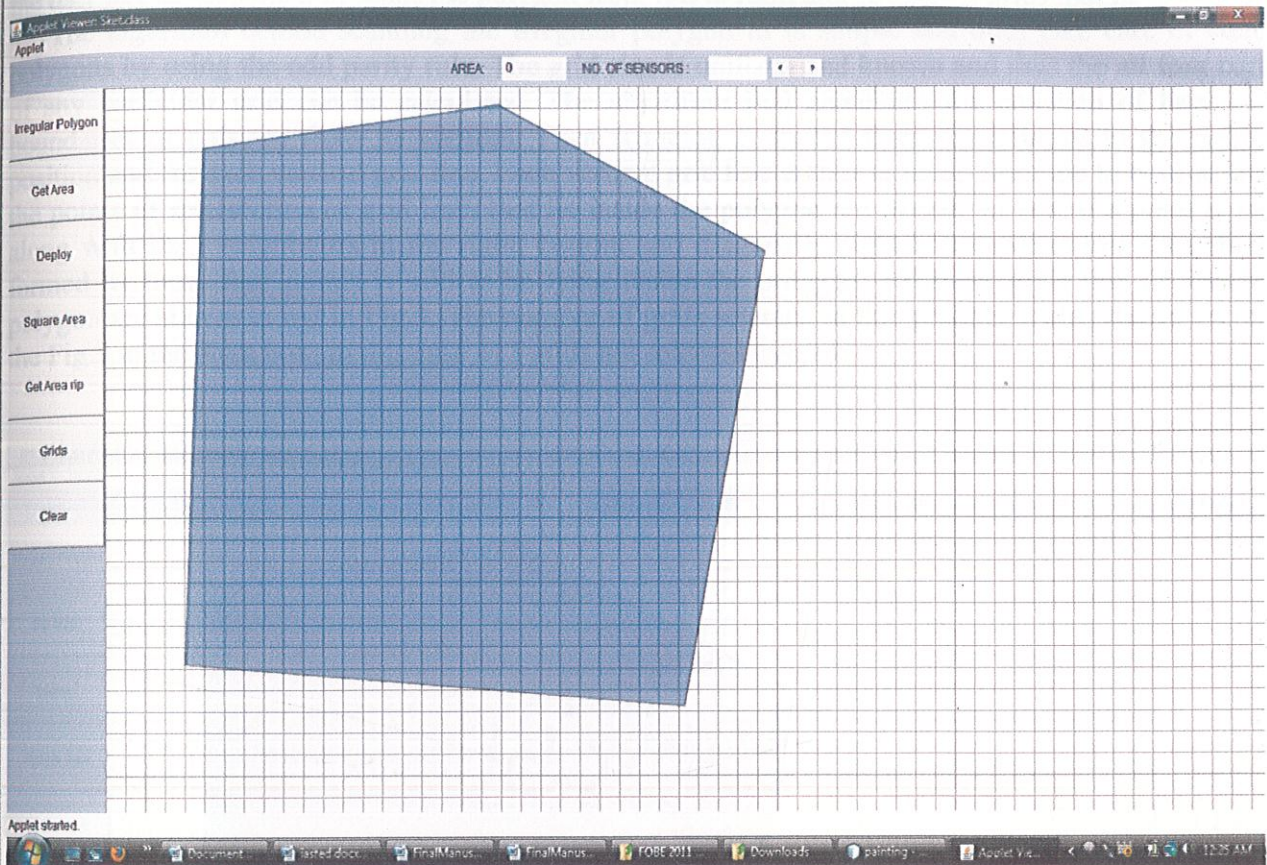


Figure 2 Approximate area selected through irregular polygon

3.3 Scanning the Irregular Area

The approximate area selected is needed to be refined and that is done through scanning the homogeneous grids in the approximate selection. Only the grids that totally lie in the region are needed and rest have to be discarded. The user has to click on the Get Area button for this functionality. Once the user has clicked the tool scans the polygon from left to right and selects all grids that lie totally in it.

The algorithm behind scanning the irregular polygon is simple and does take care of concave polygons by using the odd parity rule. The grid size is uniform and known and thus the all four corners of any particular grid can be calculated. The algorithm first calculates the equation of lines of the boundaries of polygon from its vertices. Then scanning is started from the leftmost to the rightmost position and for each vertical grid line. Each vertical grid line is then scanned from top to bottom and all the points of intersection of grid lines that lie inside the polygon are stored. At last, if a point is inside along with its 3 neighbours in the right, bottom and diagonally right-bottom direction, then the grid formed by these four points is said to be lying inside the polygon. All these grids that lie inside the polygon are shown in red in Fig.3. The number of grids selected in this process is also shown on top of the Fig.3 in the Area box; in this case 602 grids are selected overall.

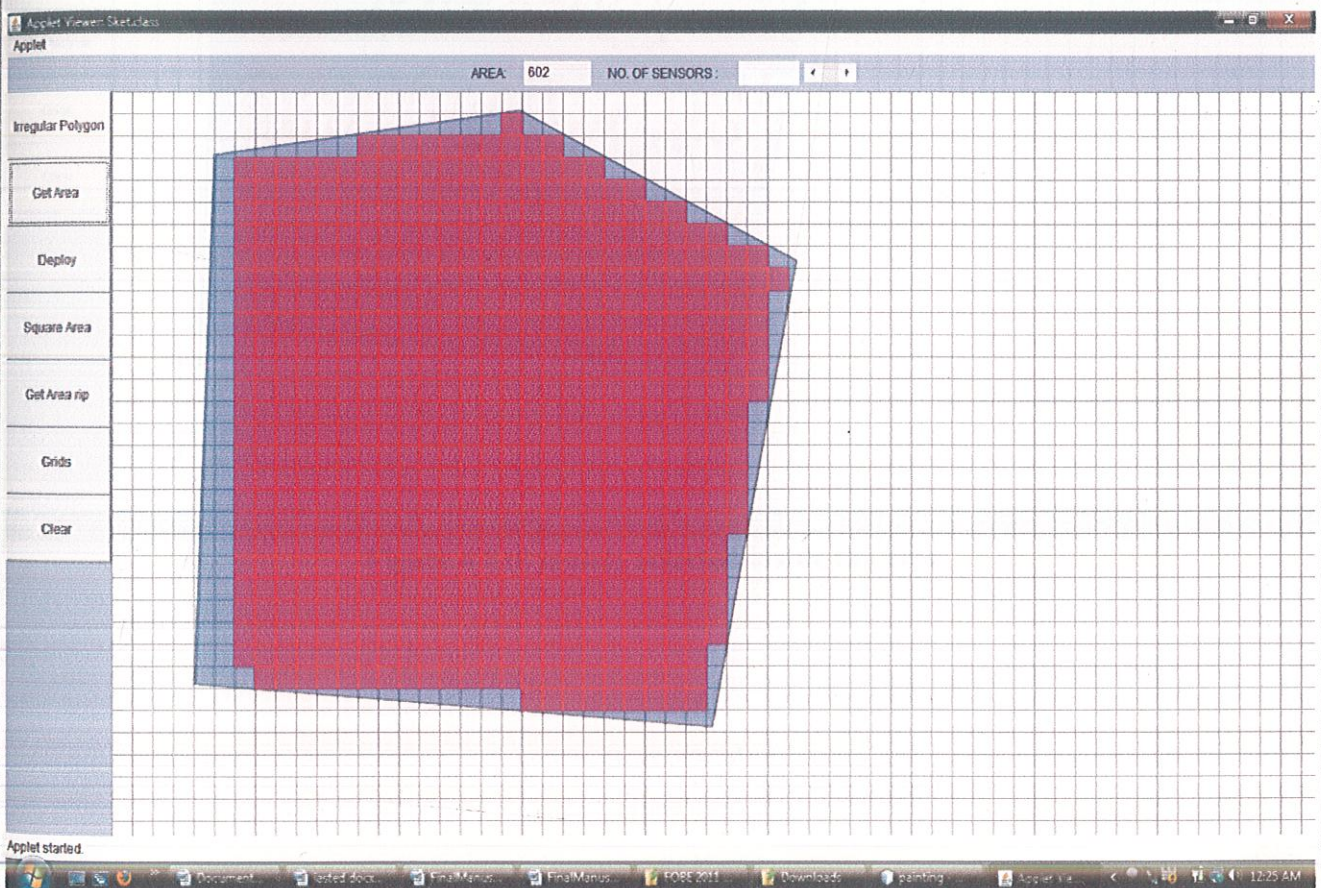


Figure 3 Grids which totally lie inside the polygon in red after scanning

3.4 Selecting a $n \times n$ matrix

The actual need of a Gur Game algorithm is that it requires a homogenous matrix where the sensors can be deployed. So there is a need of selecting a $n \times n$ matrix out of these grids which lie inside. The user need to click on the Square Area button for this purpose and then clicking and dragging the cursor can select a $n \times n$ matrix. Fig. 4 shows that a 8×8 matrix has been selected and the number of grids is shown in the Area field. We can select any size of square matrix that fits in the polygon. The tool restricts the user to extend this matrix beyond the polygon boundaries.

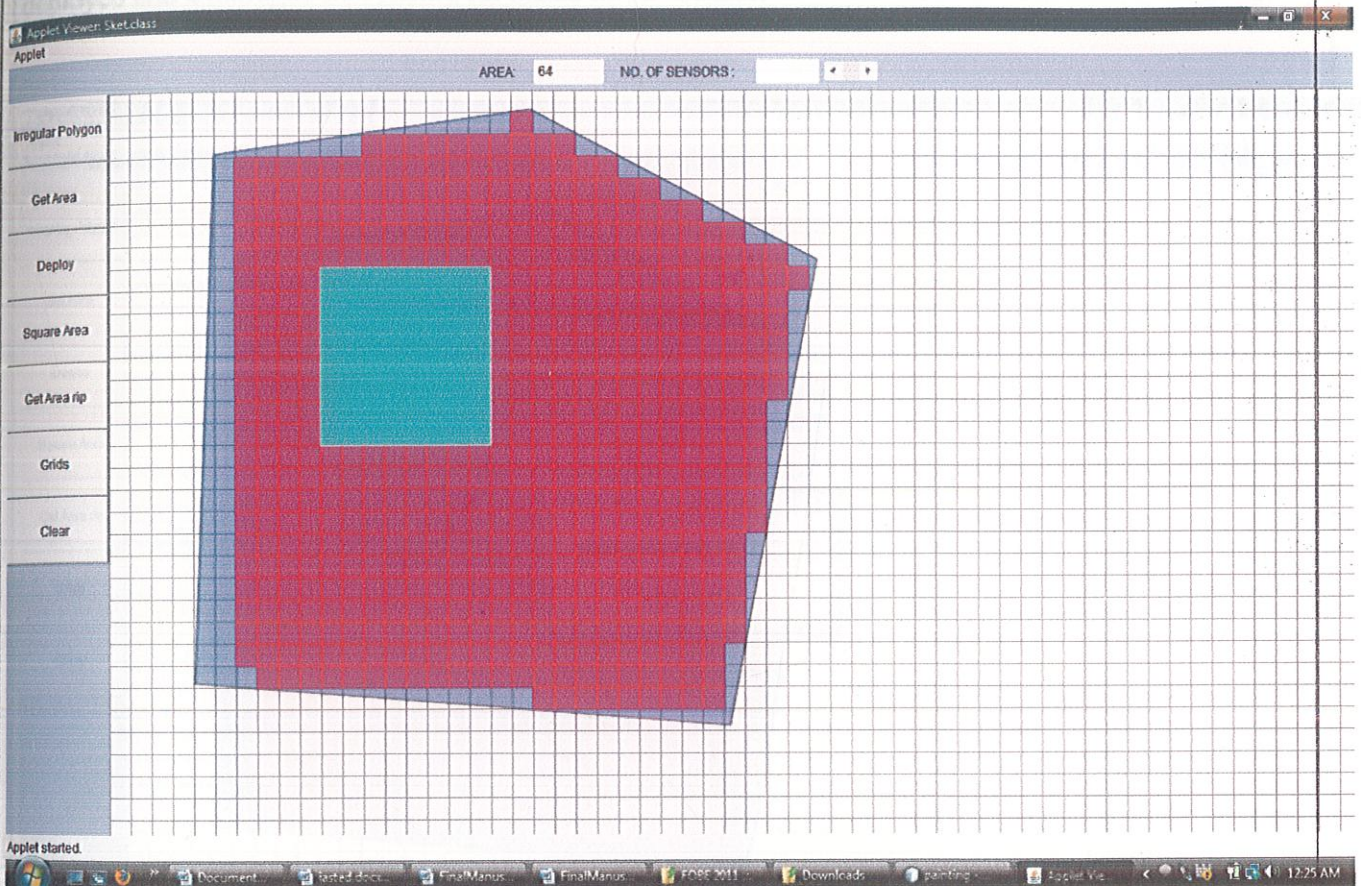


Figure 4 8×8 matrix selected inside the area.

3.5 Deployment of Sensors

The Gur Game Algorithm requires the number of regions and the number of sensors to be deployed as input along with the initial status of sensors. The square matrix provides the algorithm with the number of grids (in this case 64) and now the tool needs to fill in for sensors. The user needs to increase the number of sensors in the No. of Sensors field and then has to just click on the Deploy button to deploy these sensors in the region selected. The tool will ask the user to provide the initial status of each sensor that whether it is active or not. Once the user provides the necessary inputs the Gur Game Algorithm runs in the background to check if Quality of Service is achieved or not. Fig. 5 shows 160 sensors being deployed and a dialogue box asking for status of 78th sensor.

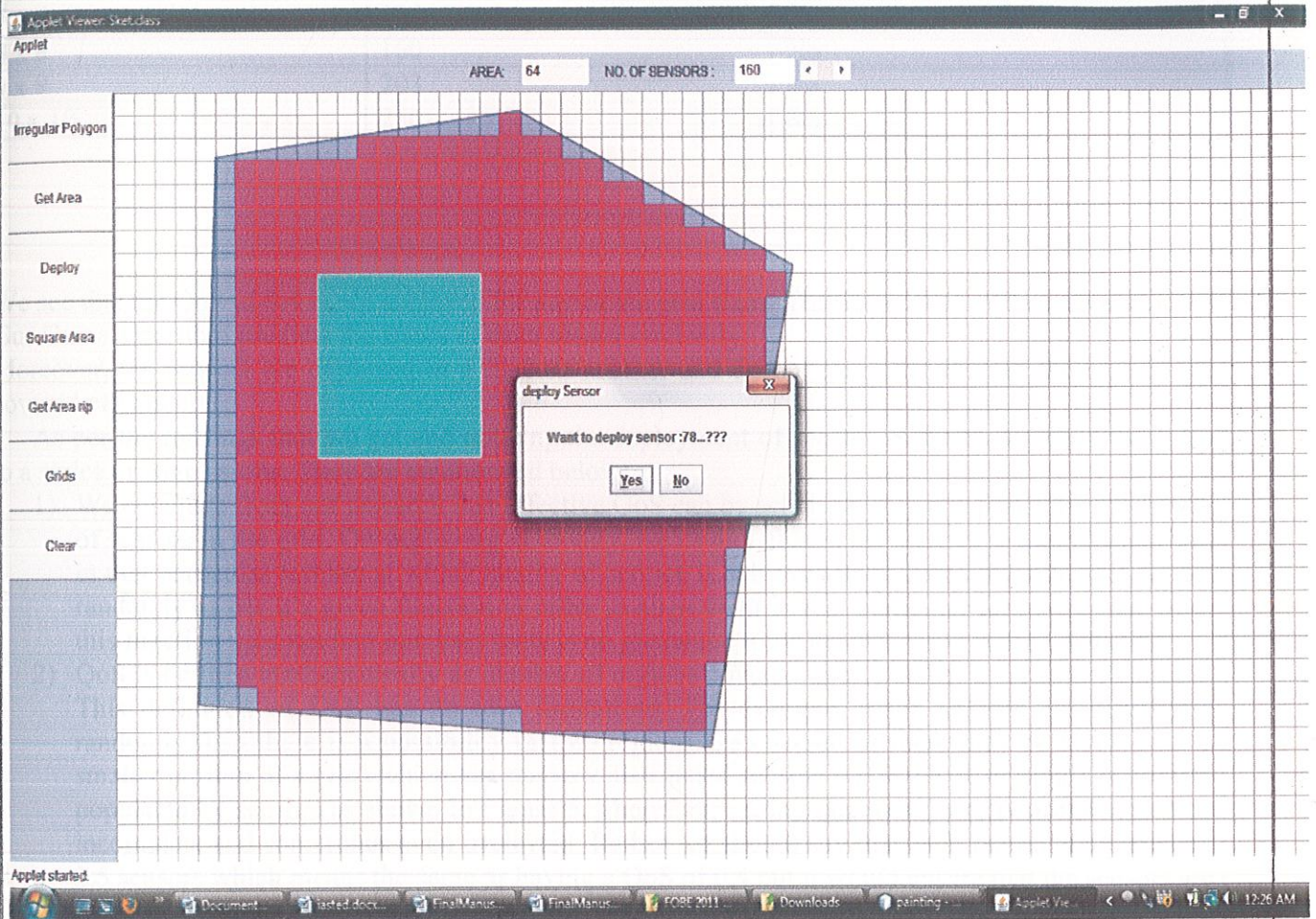


Figure 5 Sensors being deployed and tool asking for status of sensors

3.6 Results and Conclusion

We have run the simulation for different grid sizes and the results are as shown in table 1.

Table 1
Results of simulation

Grid Size (n x n)	No. of sensors deployed	QoS of 0.4 achieved?
4 x 4	40	No
5 x 5	63	No
6 x 6	90	No
7 x 7	123	No
8 x 8	160	No
9 x 9	203	No
10 x 10	250	No

We see that the QoS of 0.4 is not achieved for any of the grid sizes. This is because of the reason that the Gur Game algorithm predicts the status of each sensor randomly. Moreover, the sensors are deployed in a random manner and do not guarantee that all the regions are covered effectively.

As per the readings that we get and noticing the deployment of sensors at the background we come to a series of conclusions. They are enumerated below:

- 1) We conclude from our research that effective QoS can be achieved only at a particular placement of sensors in the grid. On deploying randomly, we cannot ensure the sensors to align themselves in that particular fashion at which quality of service is achieved. Gur Game rewards each sensor randomly as per its given function in order to align them in that particular formation, but since this function uses random numbers there is no guarantee if the desired result will be achieved.
- 2) QoS is defined mathematically as number of regions covered upon number of sensors deployed. This definition of QoS has its own limitations. For example, when sensors are distributed randomly then there is a possibility that no regions are covered or there may be a sub-grid (a smaller portion of a grid) which has sparsely distributed sensors while the other region is densely populated by sensors as shown in Figure 6. Therefore, calling quality of service at 0.4 would be incomplete and sometimes even irrelevant. Rather we should say that each region is covered by 2.5 sensors which means the same as having a QoS of 0.4 but it would ensure that the sensors are evenly distributed also.

3	3	5	2
1	2	3	4
1	1	3	5
1	1	2	3

Figure 6 A 4 x 4 grid with number of sensors being displayed

- 3) From our research we also conclude that the QoS can effectively be achieved for a $n \times n$ grid if and only if n is an even number. If n is an odd number then number of regions in an $n \times n$ grid would also be an odd number and as we know quality of service is achieved at each region being monitored by 2.5 sensors which would not be practically feasible as we would be requiring a fraction of sensors to cover the whole grid. Table 2 illustrates this phenomenon for various $n \times n$ grid and the number of sensors required for each grid to achieve the QoS of 0.4.

Table 2
Achievable QoS of 0.4 in grids of $n \times n$, where n is even

Grid Size ($n \times n$)	No. of Sensors deployed	QoS
4 x 4	40	0.4
6 x 6	90	0.4
8 x 8	160	0.4
10 x 10	250	0.4
12 x 12	360	0.4

CHAPTER 4

INSIGHT INTO THE PROGRAM

4.1 Variable Description:

These are some of the important variables, methods and their descriptions used in our program code in implementing the Gur Game algorithm in our GUI.

VARIABLES	DESCRIPTION
Public int i	Counter
Intxpoint[100]	For storing x-coordinates of points
Intypoints[100]	For storing the y-coordinates of points
Intnpoints	For keeping a counter of no. of points
Int yes	Check for active sensor
Int no	Check for in-active sensor
Intsenv	Counter for no. of sensors deployed
Ints[100]	For storing status of sensors
Intsqflag	Checks if we have selected Square option
Public static Random rm	To generate Random Nos.
Private ArrayList<Integer> generate	For generating a list of regions where an active sensor is deployed
Float r	Reward Function
Float county	To keep a count of active sensors
Intprevar	To keep a count of the area previously selected
ArrayList<Shape> shapes	Stores the grids selected
Point startDrag	To keep the value of the first point when we start dragging
Point endDrag	To keep the value of the last point when we have finished dragging
Intxreg[100]	Stores the x-coordinates that lie in the polygon between the min and max values
Intyreg[100][100]	Stores the y-coordinates of the points in the polygon for each x-coordinate.
Intxp[]	For calculation during checking whether a point is inside a polygon or not.
Int start	Converting startDrag value to nearest grid intersection
Int end	Converting endDrag value to nearest grid intersection.
Intar	Stores the area of the selection in total no.

	of grids
int xlim1[100]	For storing the x-coordinates when we encounter a reflex polygon
Intpoints[1000][2]	For storing the points of intersection of grids that lie inside polygon
Int entry	Keeps a count for the no. of entries in points[[]].
Intxpoly[4]	For storing the 4 x-coordinates of the square grid
Intypoly[4]	For storing the y-coordinates of the square grid
Int flag	Acts as a flag
Intbeginsq	Set when we can begin with making of the Square Area
Intpolflag	Set if Irregular Polygon has been made
Intxmin	For calculating the minimum value of x-coordinate in the Polygon
Intxmax	For calculating the maximum value of x-coordinate in the Polygon
Intymin	For calculating the minimum value of y-coordinate in the Polygon
Intymax	For calculating the maximum value of y-coordinate in the Polygon
Inta,b,c,x,y	For counters and internal calculations

METHODS	DESCRIPTIONS
Public void init()	For initialization of the applet
Public void actionPerformed(ActionEvent e)	Called whenever an action is performed on the Applet
Public void background()	For drawing the background with grid lines
Public void clearpnt()	For clearing the previous values of drawing
Public void polOperation(MouseEvent e)	When mouse is clicked for drawing irregular polygon
Public void fillp()	To draw the polygon and fill it
Public void irpOperation(MouseEvent e)	When mouse is clicked for selecting a square area
Public irprel(MouseEvent e)	When mouse is released while selecting a square area
Public void irpdrg(MouseEvent e)	When mouse is dragged while selecting a square area

Void area(int s)	Calculates area for a given side of a square
Public void paint()	Drawing the selected square area
Public void rpareaOperation()	To calculate the area of the selected Irregular Polygon
Public void search(int a, int b)	For searching whether the given x,y values is a grid intersection inside the given Irregular Polygon
Private void sorty(intpos, int size)	For sorting the y-coordinates in case of a reflex polygon
Public void clearlist()	To clear the points for new selection
Public void clearPanel(Panel p)	This is to refresh the working frame
Public void deploy()	For deploying sensors in the selected area of the given field and to apply Gur Game technique to check if QoS is achieved
Public void rewardFunc()	This is used to reward the sensors for the Gur Game.

CHAPTER 5

INSIGHT INTO USER INTERFACE

5.1 Introduction

Its user interface consists of three panels: status panel, control panel, drawing panel.

Method	Description
private Panel controlPanel = new Panel(new GridLayout(1,2,0,0));	To make a control panel, that contains the buttons.
private Panel drawPanel = new Panel();	To make a drawing panel, where all the drawing polygon takes place.
private Panel statusPanel = new Panel();	To make a status panel that gives the information about the status of the program.
add(statusPanel, "North"); add(controlPanel, "West"); add(drawPanel, "Center");	To add the panels in the positions as mentioned.

5.2 STATUS PANEL

This panel contains information about the status of the program.

This contains AREA LABEL window where the area of the polygon is shown.

NO. OF SENSOR LABEL that allows a user to enter the number of sensors that have to be deployed .

VARIABLES	DISCRIPTION
private Label areaLabel = new Label(" AREA:");	To set a new label that says area in the status panel.

private Label senLabel = new Label(" NO. OF SENSORS :");	To print a new label that says No. of sensors.
private TextField areaStatusBar = new TextField(6);	This is a textfield used to show the area.
private TextField senStatusBar = new TextField(5);	This is a textfield used to show the no. of sensors.
private Scrollbar senSlider = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 255);	This is scroll bar used to take the input of number of sensors from the user.
getarea = String.valueOf(ar);	This variable used to get the area of the polygon from the rpareaOperation(described later).

METHOD	DESCRIPTION
areaStatusBar.setText(getarea);	To set the area status bar to the area of the polygon.
public void updatearea()	To update the area status bar for the new polygon and set it to zero until the rpareaOperation function is called again.
public void adjustmentValueChanged(AdjustmentEvent e)	This is used to invoke updatesen() when the event takes place at the scrollbar.
Updatesen()	This function is used to update the values as the scroll bar is pressed.


```

private final int NO_OP      = 0;
private final int POL_OP    = 1;
private final int IRP_OP    = 2;
private final int RPAREA_OP = 3;
private final int PAINT_OP  = 4;
private final int IRPAREA_OP = 5;
private final int CLEAR_OP  = 6;
private final int DEPLOY_OP = 7;

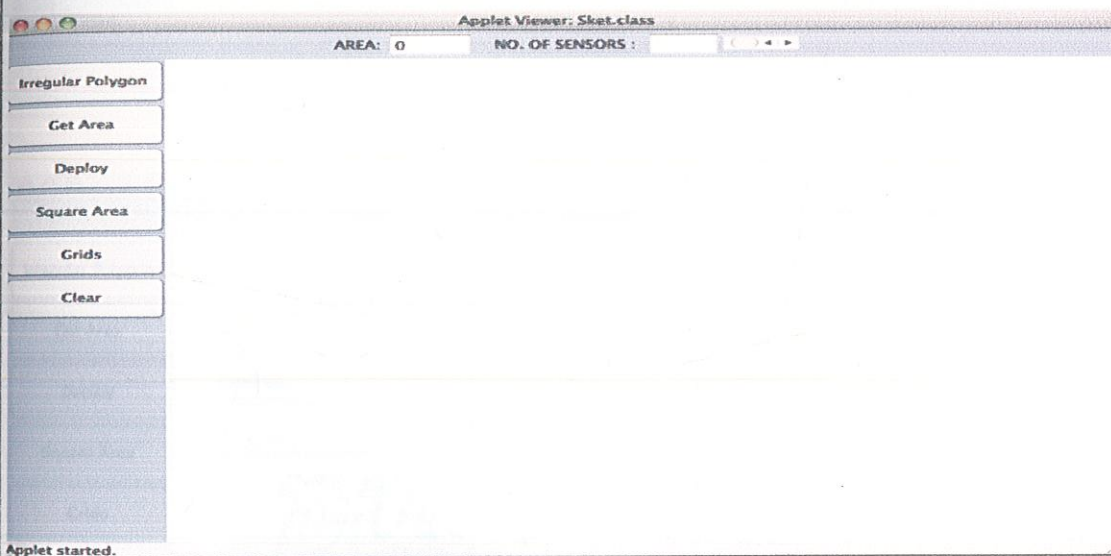
```

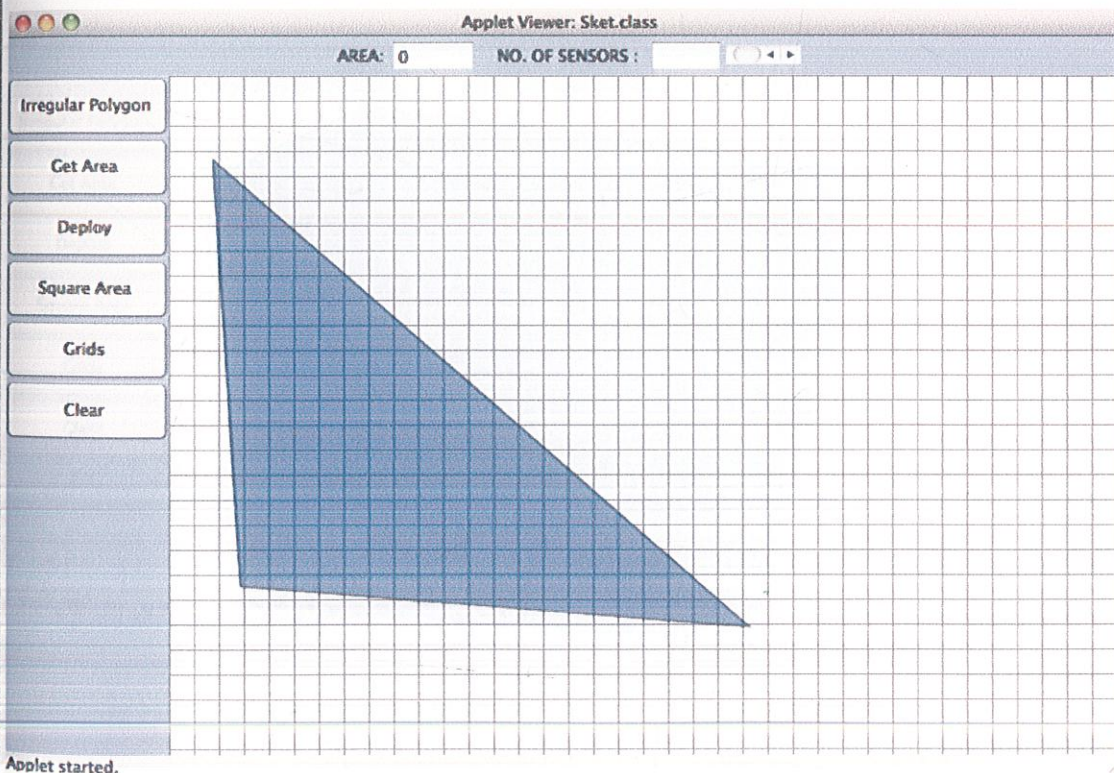
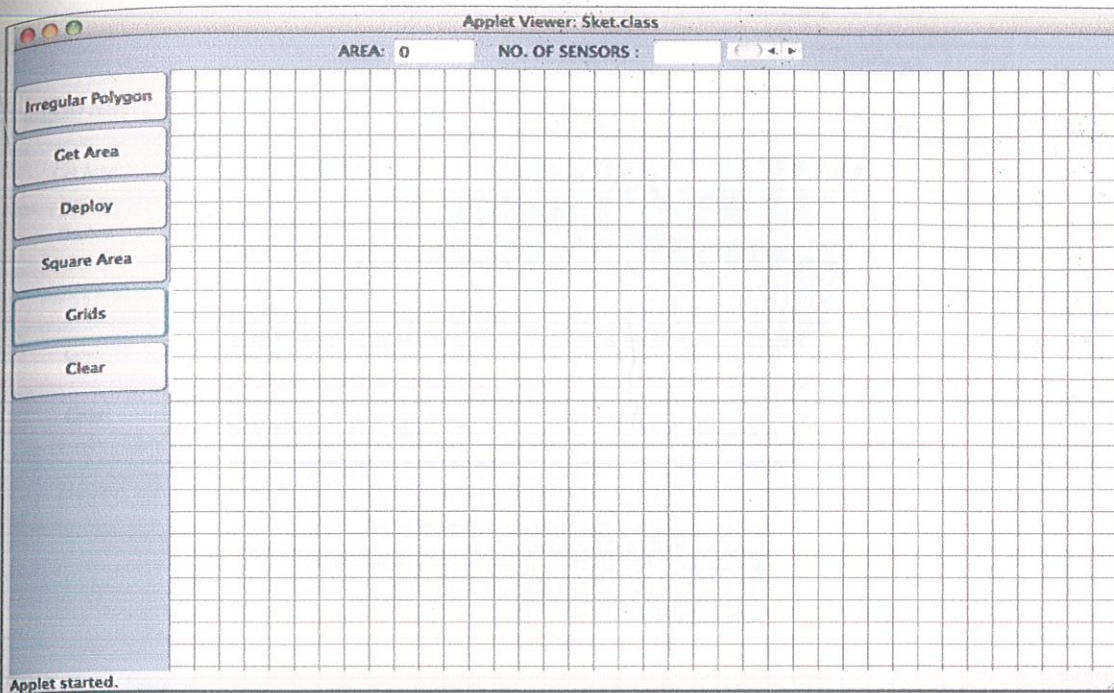
These variables are set to keep a count of which button is selected and whose functions should be invoked.

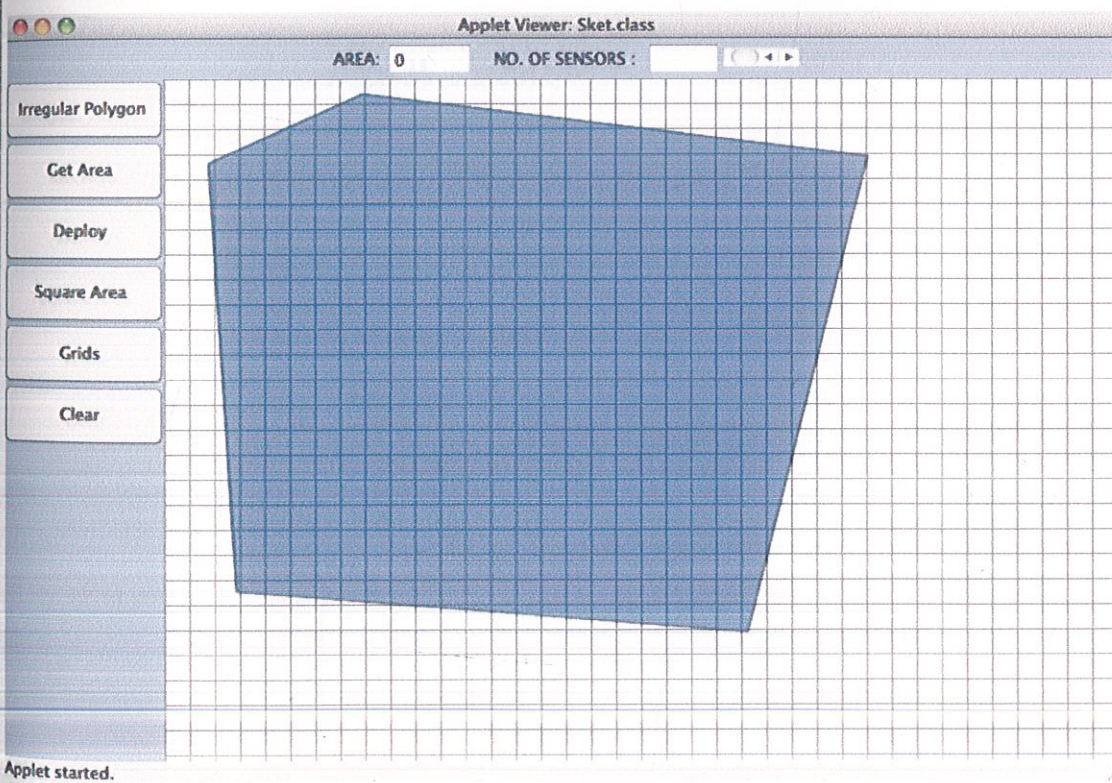
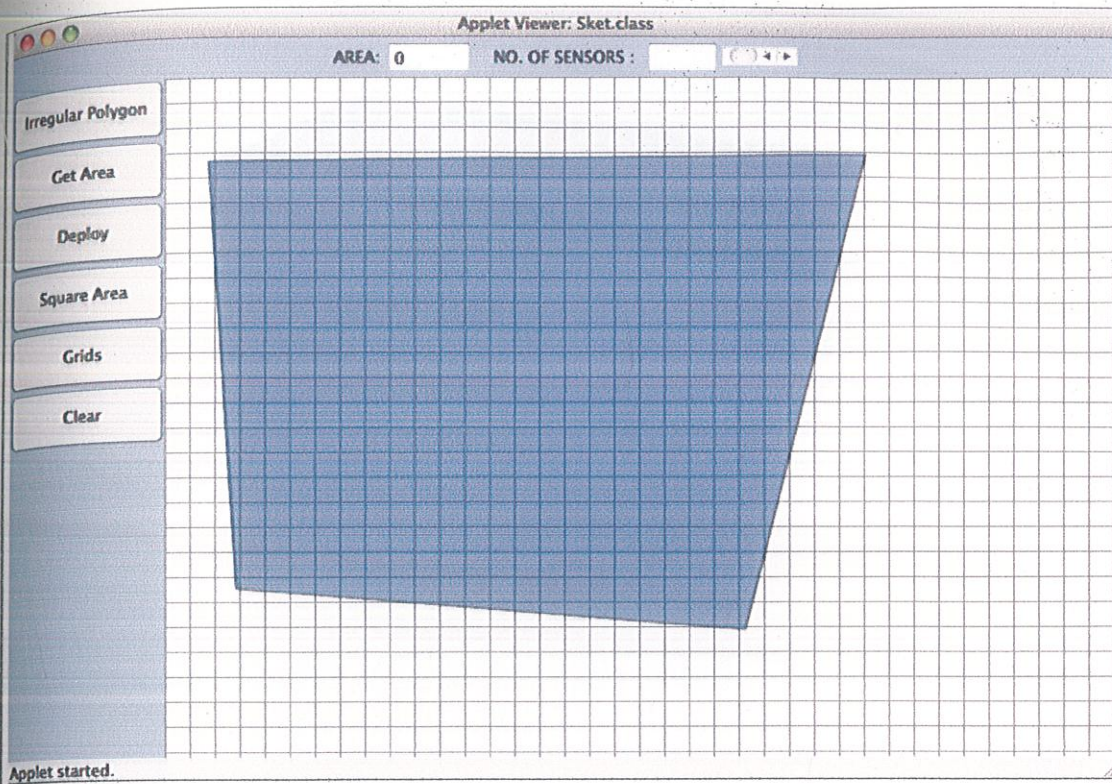
```
private int opStatus;
```

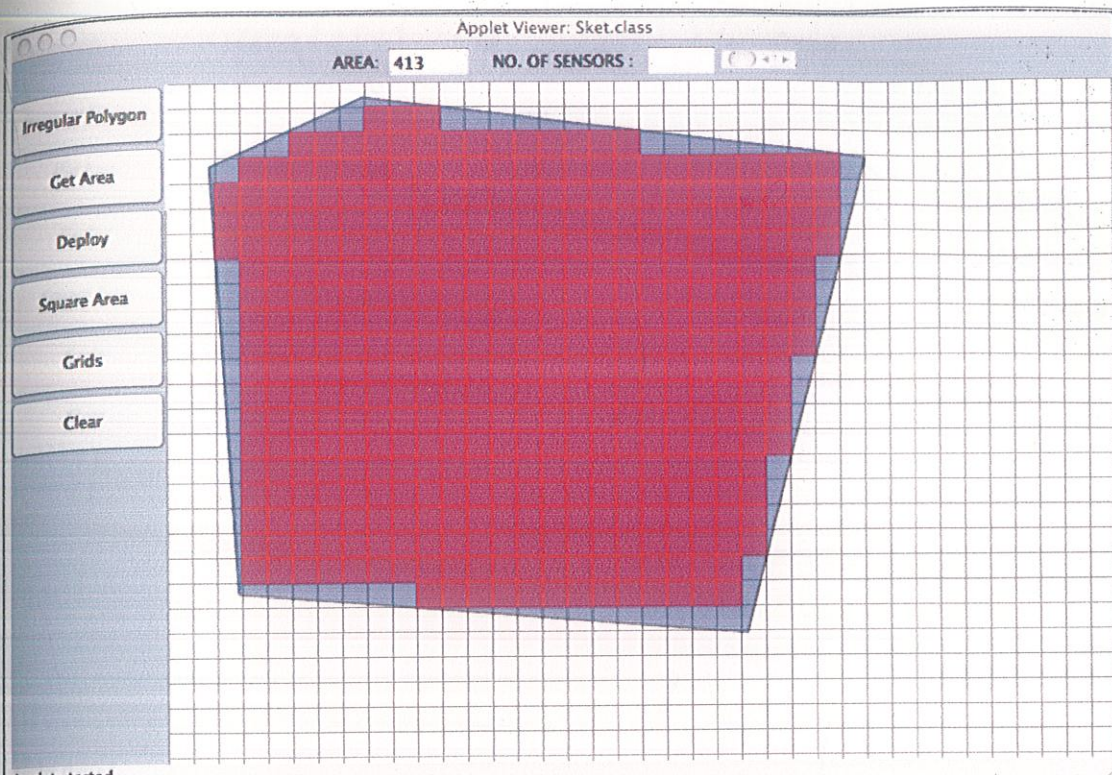
This is the variable that saves the status about the button that is currently clicked.

5.5 Snapshots of GUI

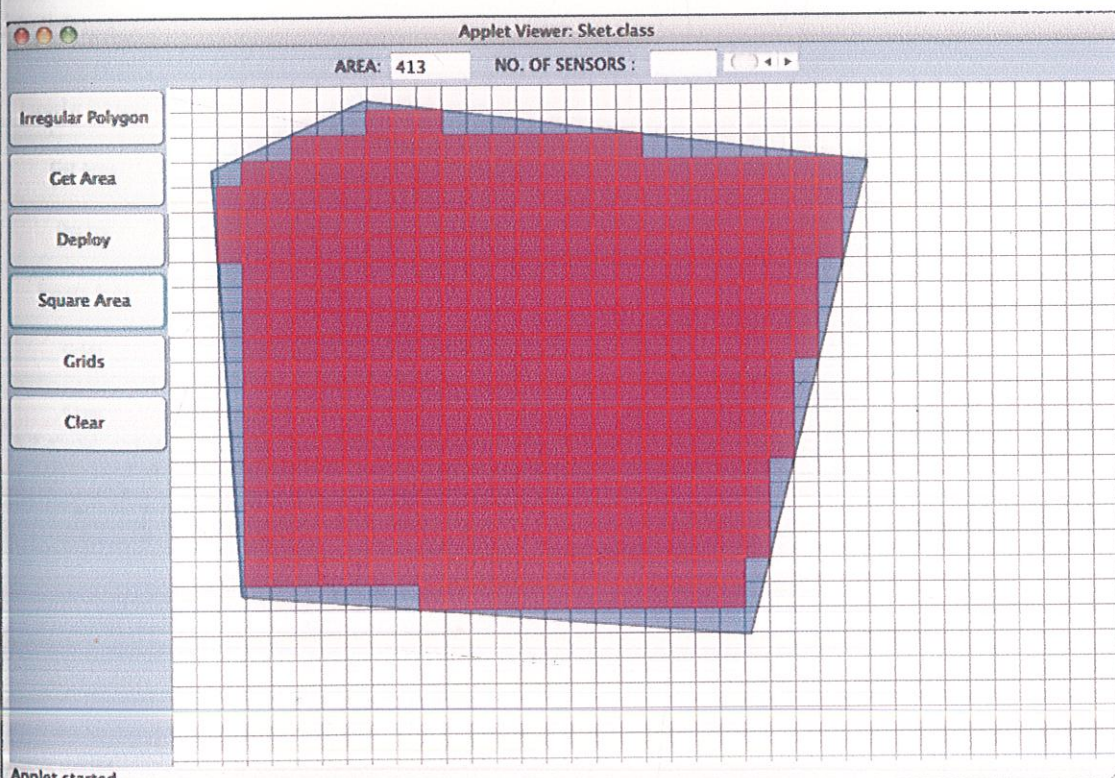




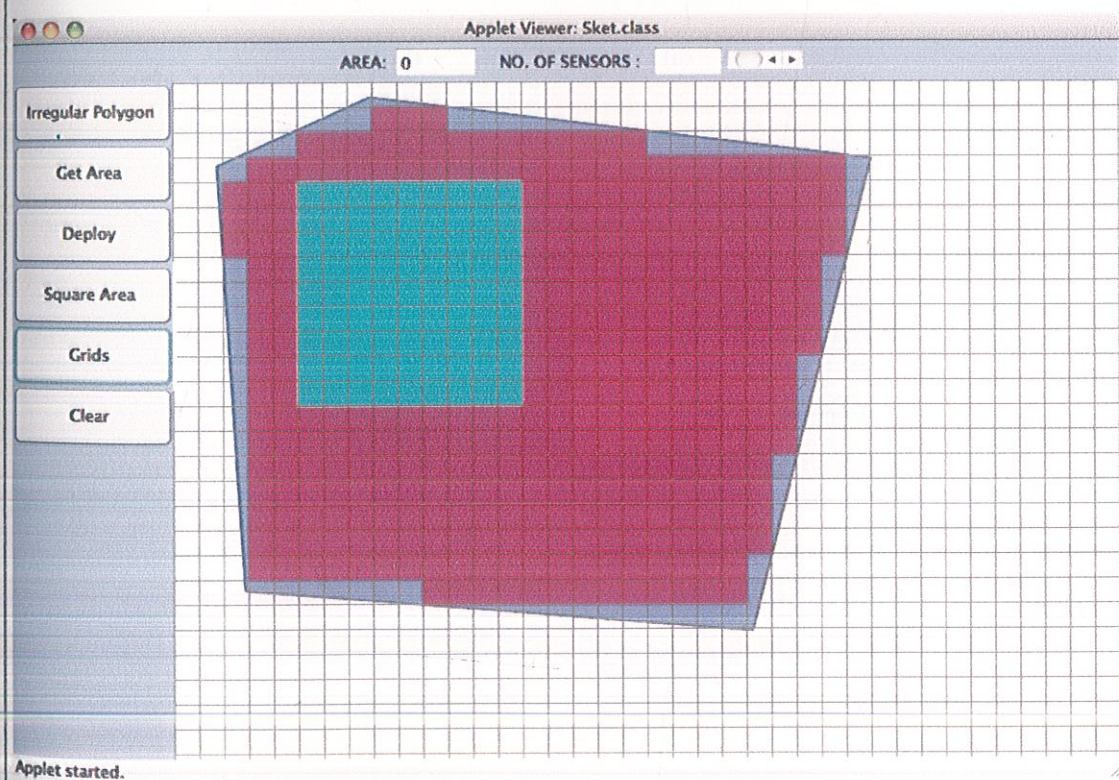
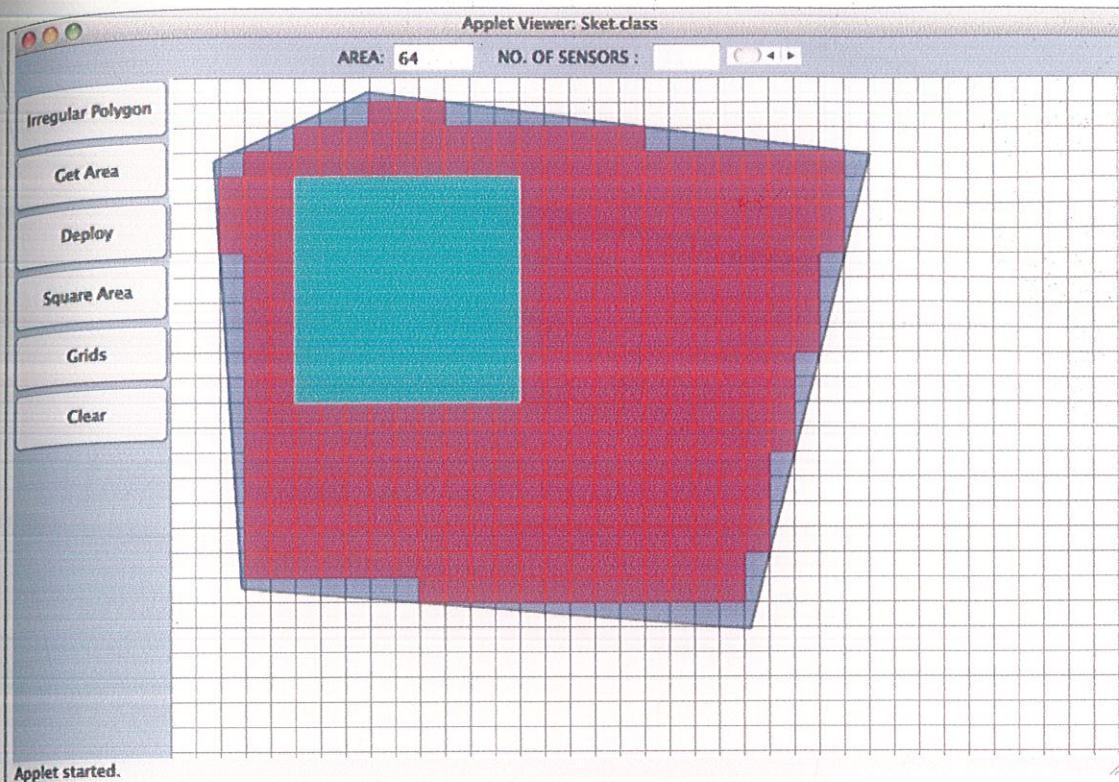


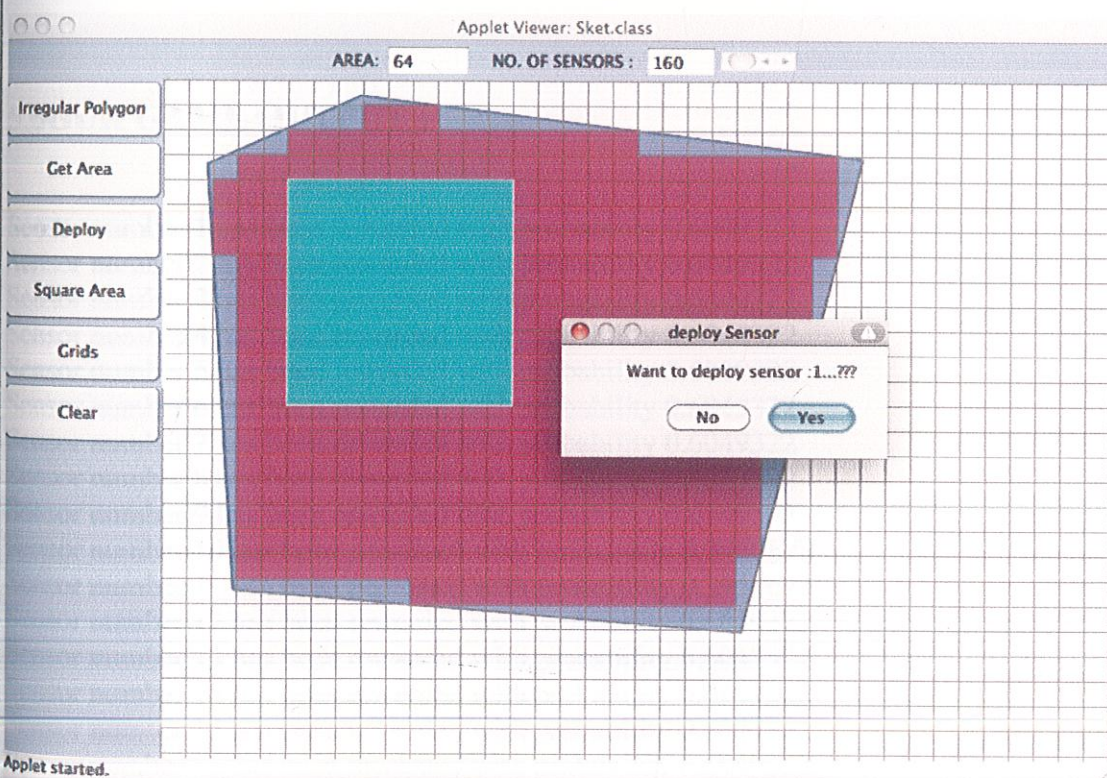
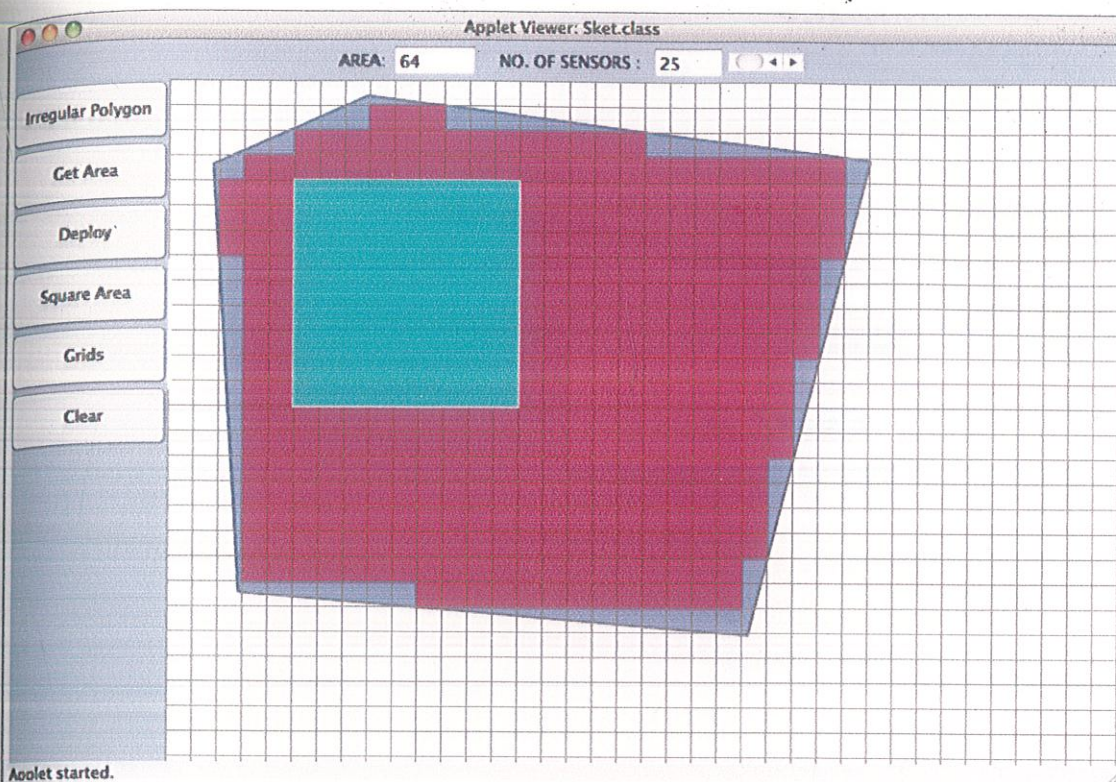


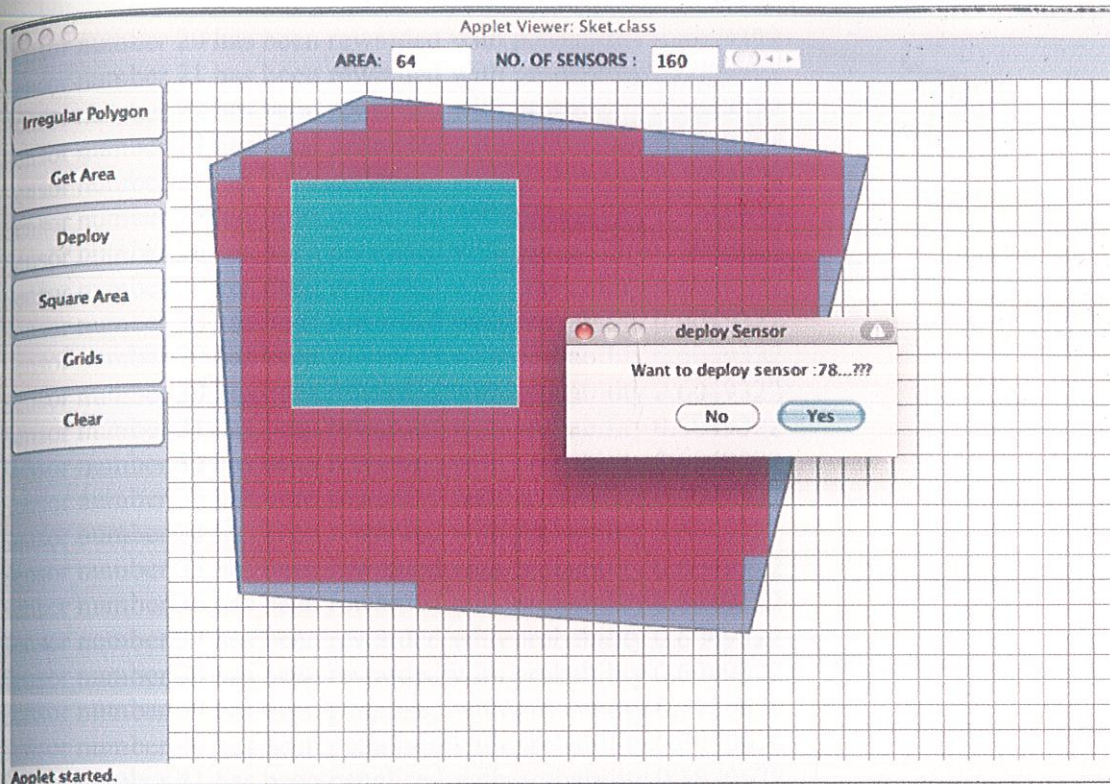
Applet started.



Applet started.







5.6GUR GAME OUTPUT :

Sensor number 1 has been rewarded with probability 0.6049322
 Sensor number 2 has been rewarded with probability 0.6049322
 Sensor number 3 has been rewarded with probability 0.6049322
 Sensor number 4 has been rewarded with probability 0.6049322
 Sensor number 5 has been rewarded with probability 0.6049322
 Sensor number 6 has been rewarded with probability 0.6049322
 Sensor number 7 has been rewarded with probability 0.6049322
 Sensor number 8 has been rewarded with probability 0.6049322
 Sensor number 9 has been rewarded with probability 0.6049322
 Sensor number 10 has been penalised with probability 0.3950678
 Sensor number 11 has been penalised with probability 0.3950678
 Sensor number 12 has been rewarded with probability 0.6049322
 Sensor number 13 has been rewarded with probability 0.6049322
 Sensor number 14 has been rewarded with probability 0.6049322
 Sensor number 15 has been rewarded with probability 0.6049322
 Sensor number 16 has been rewarded with probability 0.6049322
 Sensor number 17 has been rewarded with probability 0.6049322
 Sensor number 18 has been rewarded with probability 0.6049322
 Sensor number 19 has been rewarded with probability 0.6049322

Sensor number 68 has been rewarded with probability 0.6049322
Sensor number 69 has been rewarded with probability 0.6049322
Sensor number 70 has been rewarded with probability 0.6049322
Sensor number 71 has been rewarded with probability 0.6049322
Sensor number 72 has been rewarded with probability 0.6049322
Sensor number 73 has been rewarded with probability 0.6049322
Sensor number 74 has been rewarded with probability 0.6049322
Sensor number 75 has been rewarded with probability 0.6049322
Sensor number 76 has been rewarded with probability 0.6049322
Sensor number 77 has been rewarded with probability 0.6049322
Sensor number 78 has been rewarded with probability 0.6049322
Sensor number 79 has been rewarded with probability 0.6049322
Sensor number 80 has been rewarded with probability 0.6049322
Sensor number 81 has been rewarded with probability 0.6049322
Sensor number 82 has been rewarded with probability 0.6049322
Sensor number 83 has been rewarded with probability 0.6049322
Sensor number 84 has been rewarded with probability 0.6049322
Sensor number 85 has been penalised with probability 0.3950678
Sensor number 86 has been rewarded with probability 0.6049322
Sensor number 87 has been rewarded with probability 0.6049322
Sensor number 88 has been penalised with probability 0.3950678
Sensor number 89 has been rewarded with probability 0.6049322
Sensor number 90 has been rewarded with probability 0.6049322
Sensor number 91 has been rewarded with probability 0.6049322
Sensor number 92 has been rewarded with probability 0.6049322
Sensor number 93 has been rewarded with probability 0.6049322
Sensor number 94 has been rewarded with probability 0.6049322
...SORRY QoS IS NOT ACHIEVED as X is not equal to 0.4...

CHAPTER 6

FUTURE DIRECTIONS

6.1 Future Scope

The future vision of WSNs is to embed numerous distributed devices to monitor and interact with physical world phenomena, and to exploit spatially and temporally dense sensing and actuation capabilities of those sensing devices. These nodes coordinate among themselves to create a network that performs higher-level tasks.

Although extensive efforts have been exerted so far on the routing problem in WSNs, there are still some challenges that confront effective solutions of the routing problem. First, there is a tight coupling between sensor nodes and the physical world. Sensors are embedded in unattended places or systems. This is different from traditional Internet, PDA, and mobility applications that interface primarily and directly with human users. Second, sensors are characterized by a small foot print, and as such nodes present stringent energy constraints since they are equipped with small, finite, energy source. This is also different from traditional fixed but reusable resources. Third, communications is primary consumer of energy in this environment where sending a bit over 10 or 100 meters consumes as much energy as thousands-to-millions of operations (known as R4 signal energy drop-off) [36].

Although the performance of these protocols is promising in terms of energy efficiency, further research would be needed to address issues such as Quality of Service (QoS) posed by video and imaging sensors and real-time applications. Energy-aware QoS routing in sensor networks will ensure guaranteed bandwidth (or delay) through the duration of connection as well as providing the use of most energy efficient path. Another interesting issue for routing protocols is the consideration of node mobility. Most of the current protocols assume that the sensor nodes and the BS are stationary. However, there might be situations such as battle environments where the BS and possibly the sensors need to be mobile. In such cases, the frequent update of the position of the command node and the sensor nodes and the propagation of that information through the network may excessively drain the energy of nodes. New routing algorithms are needed in order to handle the overhead of mobility and topology changes in such energy constrained environment. Future trends in routing techniques in WSNs focus on different directions, all share the common objective of prolonging the network lifetime. We summarize some of these directions and give some pertinent references as follows:

- **Exploit redundancy:** typically a large number of sensor nodes are implanted inside or beside the phenomenon. Since sensor nodes are prone to failure, fault tolerance techniques come in picture to keep the network operating and performing its tasks. Routing techniques that explicitly employ fault tolerance techniques in an efficient manner are still under investigation.
- **Tiered architectures (mix of form/energy factors):** Hierarchical routing is an old technique to enhance scalability and efficiency of the routing protocol. However, novel techniques to network clustering which maximize the network lifetime are also a hot area of research in WSNs.
- **Exploit spatial diversity and density of sensor/actuator nodes:** Nodes will span a network area that might be large enough to provide spatial communication between sensor nodes. Achieving energy efficient communication in this densely populated environment deserves further investigation. The dense deployment of sensor nodes should allow the network to adapt to unpredictable environment.

- Achieve desired global behavior with adaptive localized algorithms (i.e., do not rely on global interaction or information). However, in a dynamic environment, this is hard to model.

- Leverage data processing inside the network and exploit computation near data sources to reduce communication, i.e., perform in-network distributed processing. WSNs are organized around naming data, not nodes identities. Since we have a large collections of distributed elements, localized algorithms that achieve system-wide properties in terms of local processing of data before being sent to the destination are still needed. Nodes in the network will store named data and make it available for processing. There is a high need to create efficient processing points in the network, e.g., duplicate suppression, aggregation, correlation of data. How to efficiently and optimally find those points is still an open research issue.

- Time and location synchronization: energy-efficient techniques for associating time and spatial coordinates with data to support collaborative processing are also required [20].

- Localization: sensor nodes are randomly deployed into an unplanned infrastructure. The problem of estimating spatial-coordinates of the node is referred to as localization. Global Positioning System (GPS) cannot be used in WSNs as GPS can work only outdoors and cannot work in the presence of any obstruction. Moreover, GPS receivers are expensive and not suitable in the construction of small cheap sensor nodes. Hence, there is a need to develop other means of establishing a coordinate system without relying on an existing infrastructure. Most of the proposed localization techniques today, depend on recursive trilateration/multilateration techniques (e.g., [38]) which would not provide enough accuracy in WSNs.

- Self-configuration and reconfiguration is essential to lifetime of unattended systems in dynamic, and constrained energy environment. This is important for keeping the network up and running. As nodes die and leave the network, update and reconfiguration mechanisms should take place. A feature that is important in every routing protocol is to adapt to topology changes very quickly and to maintain the network functions.

- Secure Routing: Current routing protocols optimize for the limited capabilities of the nodes and the application specific nature of the networks, but do not consider security. Although these protocols have not been designed with security as a goal, it is important to analyze their security properties. One aspect of sensor networks that complicates the design of a secure routing protocol is in-network aggregation. In WSNs, in-network processing makes end-to-end security mechanisms harder to deploy because intermediate nodes need direct access to the contents of the messages.

- Other possible future research for routing protocols includes the integration of sensor networks with wired networks (i.e. Internet). Most of the applications in security and environmental monitoring require the data collected from the sensor nodes to be transmitted to a server so that further analysis can be done. On the other hand, the requests from the user should be made to the BS through Internet. Since the routing requirements of each environment are different, further research is necessary for handling these kinds of situations.

APPENDIX I

PROGRAM OF INTERFACE

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.awt.event.MouseMotionListener;
import java.awt.event.MouseEvent;
import java.lang.Math.*;
import java.text.*;
import java.io.*;
import java.util.*;
import java.awt.Scrollbar.*;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.AlphaComposite;
import java.awt.BasicStroke;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.RenderingHints;
import java.awt.Shape;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;
import java.awt.Polygon;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JTextField;
import java.awt.event.MouseListener;
import javax.swing.JTextField;
```



```

public class Sket1 extends Applet implements ActionListener,MouseListener, MouseMotionListener
{
    /* Operation Constants */
    private final int NO_OP      = 0;
    private final int POL_OP     = 1;
    private final int IRP_OP     = 2;
    private final int RPAREA_OP  = 3;
    private final int PAINT_OP   = 4;
    private final int IRPAREA_OP = 5;
    private final int CLEAR_OP   = 6;
    private final int FILL_OP    = 7;
    private Color barc   = new Color(166,166,255);

    /* Operation Button definitions */
    private Button polButton      = new Button("Regular polygon");
    private Button irpButton      = new Button("Irregular polygon");
    private Button rpareaButton   = new Button("Get Area ");
    private Button paintbButton   = new Button("Grids");
    private Button irpareaButton  = new Button("Get Area.rip");
    private Button clearButton    = new Button("Clear");
    private Button fillButton     = new Button("Fill");

    /* Assorted status values for different variables */
    private TextField areaStatusBar = new TextField(20);

    /* Labels for operation and color fields */
    private Label areaLabel      = new Label(" AREA:");
    //private Label rpLabel      = new Label(" REGULAR POLYGON :");
    //private Label irpLabel     = new Label(" IRREGULAR POLYGON :");

    private int  opStatus        = PAINT_OP;
    public int   height;
    public int   width;
    public int i;
    int xpoints[] = new int[100];
    int ypoints[] = new int[100];
    int npoints = 0;
    String getarea;

    ArrayList<Shape> shapes = new ArrayList<Shape>();
    ArrayList<Point> drag = new ArrayList<Point>();
    Point startDrag, endDrag;

```



```

int yreg[][] = new int[100][100];
int xreg[] = new int[100];
int xp[] = new int[100];
int start = 0;
int end = 0;
int ar = 0;
double m[] = new double[100];
double c[] = new double[100];
int xlim1[] = new int[100];
int xlim2[] = new int[100];
int points[][] = new int[10000][2];
int entry = 0;
int xpoly[] = new int[4];
int ypoly[] = new int[4];
int flag = 0;
ArrayList<Polygon> scanned = new ArrayList<Polygon>();
int xmin=10000,xmax=0,a,b,xval,temp,z,ymin=10000,ymax=0;

```

```

private Panel controlPanel    = new Panel(new GridLayout(11,2,0,0));
private Panel drawPanel      = new Panel();
//private Panel udefdemcolPanel = new Panel();
private Panel statusPanel    = new Panel();

```

```

public void init()
{
    setLayout(new BorderLayout());

    /* setup color buttons */
    controlPanel.add(polButton);
    controlPanel.add(rpareaButton);
    controlPanel.add(irpButton);
    controlPanel.add(irpareaButton);
    controlPanel.add(paintbButton);
    controlPanel.add(clearButton);
    controlPanel.add(fillButton);

    /* Add label and color text field */
    statusPanel.add(areaLabel);
    statusPanel.add(areaStatusBar);
    areaStatusBar.setEditable(false);

    statusPanel.setBackground(barc);
    controlPanel.setBounds(0,0,100,300);
}

```



```

controlPanel.setBackground(barc);
drawPanel.setBackground(Color.white);
add(statusPanel, "North");
    add(controlPanel, "West");
add(drawPanel, "Center");

polButton.addActionListener(this);
irpButton.addActionListener(this);
rpareaButton.addActionListener(this);
paintbButton.addActionListener(this);
irpareaButton.addActionListener(this);
clearButton.addActionListener(this);
fillButton.addActionListener(this);

drawPanel.addMouseMotionListener(this);
drawPanel.addMouseListener(this);
this.addMouseListener(this);
this.addMouseMotionListener(this);
}

public void actionPerformed(ActionEvent e)
{
    if (e.getActionCommand() == "Regular polygon")
        opStatus = POL_OP;

    if (e.getActionCommand() == "Irregular polygon")
        opStatus = IRP_OP;

    if (e.getActionCommand() == "Get Area ")
        opStatus = RPAREA_OP;

    if (e.getActionCommand() == "Grids")
        opStatus = PAINT_OP;

    if (e.getActionCommand() == "Get Area rip")
        opStatus = IRPAREA_OP;

    if (e.getActionCommand() == "Clear")
        opStatus = CLEAR_OP;

    if (e.getActionCommand() == "Fill")
        opStatus = FILL_OP;

    switch (opStatus)
    {

```



```

        case PAINT_OP : updatearea();
                        background();
                        break;

        case CLEAR_OP : updatearea();
                        clearPanel(drawPanel);
                        break;

        case POL_OP : updatearea();
                        clearpnt();
                        break;

        case RPAREA_OP : rpareaOperation();
                        clearlist();
                        break;

    }

}

public void updatearea()
{
    areaStatusBar.setText("0");
}

public void background()
{
    width= drawPanel.getBounds().width;
    height= drawPanel.getBounds().height;
    //System.out.println(width+" "+height);

    Graphics g = drawPanel.getGraphics();
    //Graphics2D g2 = (Graphics2D) g;
    g.setColor(Color.GRAY);
    for (i = 0; i < width; i += 20) {
        //Shape line = new Line.Float(i, 0, i,h);
        g.drawLine(i, 0, i, height);
    }

    for (i = 0; i < height; i += 20) {
        //Shape line = new Line.Float(0, i,w, i);
        g.drawLine(0, i,width, i);
    }
}

```



```

public void clearpnt()
{
    //int xpoints[] = new int[100];
    //int ypoints[] = new int[100];
    npoints = 0;
}

```

```

public void polOperation(MouseEvent e)
{
    //npoints=0;
    xpoints[npoints]=e.getX();
    ypoints[npoints]=e.getY();
    npoints++;
    System.out.print("Mouse clicked"+npoints);
    repaint();
    fillp();
}

```

```

public void fillp()
{
    Panel p1 = drawPanel;
    Graphics g = p1.getGraphics();
    g.setColor(Color.white);
    g.fillRect(0,0,p1.getBounds().width,p1.getBounds().height);
    repaint();
    Graphics2D g2 = (Graphics2D) g;
    repaint();
    background();
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
    RenderingHints.VALUE_ANTIALIAS_ON);
    g2.setStroke(new BasicStroke(2));
    g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.50f));
    //if(npoints>2) {
        Polygon p = new Polygon(xpoints,ypoints,npoints);
        g2.setPaint(Color.black);
        g2.draw( p );
        g2.setPaint(Color.blue);
        g2.fill( p );
    //}
}

```

```

public void irpOperation(MouseEvent e)
{

```



```

/*ArrayList<Shape> shapes = new ArrayList<Shape>();
ArrayList<Point> drag = new ArrayList<Point>();
Point startDrag, endDrag;*/
startDrag = new Point(e.getX(), e.getY());
endDrag = startDrag;
drag.add(endDrag);
repaint();
}

public void irprel(MouseEvent e)
{
    drag.add(new Point(e.getX(),e.getY()));
    startDrag = null;
    endDrag = null;
    repaint();
}

public void irpdrg(MouseEvent e)
{
    endDrag = new Point(e.getX(), e.getY());
    drag.add(endDrag);
    repaint();
    paint();
}

public void paint() {
    Panel p1 = drawPanel;
    Graphics g = p1.getGraphics();
    Graphics2D g2 = (Graphics2D) g;
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
    RenderingHints.VALUE_ANTIALIAS_ON);

    Color[] colors = { Color.YELLOW, Color.MAGENTA, Color.CYAN , Color.RED, Color.BLUE,
    Color.PINK};
    int colorIndex = 0;
    int x,y;

    g2.setStroke(new BasicStroke(2));
    //g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.50f));

    for(Point p : drag) {
        g2.setPaint(Color.black);
        x=(int) p.getX();
        y=(int) p.getY();
        g2.fillOval(x,y,7,7);
    }
}

```



```

public void rpareaOperation()
{
    for(a=0;a<npoints;a++) {
        if(xmin>xpoints[a])
            xmin = xpoints[a];
        if(xmax<xpoints[a])
            xmax = xpoints[a];
        if(ymin>ypoints[a])
            ymin = ypoints[a];
        if(ymax<ypoints[a])
            ymax = ypoints[a];
        m[a] = ((ypoints[(a+1)%npoints]-ypoints[a])*1.0)/(xpoints[(a+1)%npoints]-xpoints[a]);
        c[a] = ypoints[a] - (m[a]*xpoints[a]);
        if(xpoints[(a+1)%npoints]>xpoints[a]) {
            xlim1[a] = xpoints[a];
            xlim2[a] = xpoints[(a+1)%npoints];
        }
        else {
            xlim2[a] = xpoints[a];
            xlim1[a] = xpoints[(a+1)%npoints];
        }
        System.out.println("--"+m[a]+"--"+c[a]);
    }
    start = xmin-(xmin%20)+20;
    end = xmax-(xmax%20);
    ymin = ymin-(ymin%20)+20;
    ymax = ymax-(ymax%20);
    xval=0;
    temp = start;
    while(temp<=end) {
        b=0;
        xreg[xval] = temp;
        for(a=0;a<npoints;a++) {
            if(temp>=xlim1[a] && temp<=xlim2[a])
            {
                yreg[xval][b++] = (int)((m[a]*temp)+c[a]);
                //System.out.println(xreg[xval]+"x--y"+yreg[xval][b-1]);
            }
        }
        System.out.println();
        xp[xval] = b;
        //System.out.println("--"+b);
        sorty(xval,b);
        temp+=20;
    }
}

```



```

        xval++;
    }
    entry=0;
    for(a=0;a<xval;a++) {
        for(b=0;b<xp[a];b++) {
            if(b%2==0) {
                yreg[a][b] = yreg[a][b]-(yreg[a][b]%20)+20;
            }
            else {
                yreg[a][b] = yreg[a][b]-(yreg[a][b]%20);
            }
            //System.out.println(xreg[a]+"x--y"+yreg[a][b]);
        }
        for(b=0;b<xp[a];b+=2) {
            for(z=yreg[a][b];z<=yreg[a][b+1];z+=20) {
                points[entry][0] = xreg[a];
                points[entry++][1] = z;
            }
        }
        //System.out.println();
    }
    for(a=0;a<entry;a++) {
        System.out.println("(" + points[a][0] + "," + points[a][1] + ")");
    }

    for(a=start;a<=end;a+=20) {
        for(b=ymin;b<=ymax;b+=20) {
            //flag=0;
            if(search(a,b) && search(a+20,b) && search(a+20,b+20) && search(a,b+20)) {
                xpoly[0] = xpoly[3] = a;
                xpoly[1] = xpoly[2] = a+20;
                ypoly[0] = ypoly[1] = b;
                ypoly[2] = ypoly[3] = b+20;
                /*for(int s=0;s<4;s++) {
                    System.out.println(xpoly[s]+"---"+ypoly[s]);
                }*/
                Polygon p = new Polygon(xpoly,ypoly,4);
                scanned.add(p);
                flag=1;
                repaint();
                ar+=1;
            }
        }
    }

    /*for(a=0;a<=500;a+=20) {
        for(b=0;b<=500;b+=20) {

```



```

        if(search(a,b)) {
            if(search(a,b+20) && search(a+20,b) && search(a+20,b+20))
                ar+=20*20;
        }
    }
}*/
System.out.println("Area is : "+ar);
/*for(temp=start;temp<=end;temp+=20) {
    for(b=0;b<xp[temp/20];b++)
        System.out.print(yreg[temp/20][b]+"---");
}*/
Panel p1= drawPanel;
Graphics g = p1.getGraphics();
Graphics2D g2 = (Graphics2D) g;
g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
g2.setStroke(new BasicStroke(2));
g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.50f));
if(npoints>2) {
    //Polygon p1 = new Polygon(xpoints,ypoints,npoints);
    //g2.setPaint(Color.black);
    //g2.draw(p1);
    //g2.setPaint(Color.blue);
    //g2.fill(p1);
    if(flag==1) {
        for(Polygon p : scanned) {
            g2.setPaint(Color.red);
            g2.draw(p);
            g2.fill(p);
        }
    }
}
getarea = String.valueOf(ar);
areaStatusBar.setText(getarea);
}

public void clearlist()
{
    shapes.clear();
    drag.clear();
    ar=0;
    xmin=10000;
    xmax=0;
    a=0;
    b=0;
    xval=0;
    temp=0;
}

```



```

        z=0;
        ymin=10000;
        ymax=0;
        scanned.clear();
    }

    private boolean search(int a, int b) {
        int in;
        for(in=0;in<entry;in++) {
            if(points[in][0]==a && points[in][1]==b)
                return true;
        }
        return false;
    }

    private void sorty(int pos, int size) {
        int b,temp;
        for(int a=0;a<size;a++) {
            for(b=0;b<size;b++) {
                if(yreg[pos][a]<yreg[pos][b]) {
                    temp = yreg[pos][b];
                    yreg[pos][b] = yreg[pos][a];
                    yreg[pos][a] = temp;
                }
            }
        }
        /*for(int a=0;a<size;a++)
            System.out.println(xreg[pos]+"x--y"+yreg[pos][a]);
        System.out.println();*/
    }
}

```

```

public void irpareaOperation(MouseEvent e)
{
}

```

```

public void clearPanel(Panel p)
{
    //Panel p= drawPanel;
    Graphics g = p.getGraphics();
    g.setColor(Color.white);
    g.fillRect(0,0,p.getBounds().width,p.getBounds().height);
}

```



```

public void setGraphicalDefaults(MouseEvent e)
{
    width= getWidth();
    height= getHeight();
    System.out.println(width+" "+height);
}

```

```

public void mouseClicked(MouseEvent e)
{
    switch (opStatus)
    { case POL_OP : polOperation(e);
      break;

      case IRP_OP : irpOperation(e);
      break;

      /*case RPAREA_OP : rpareaOperation(e);
      break;*/

      case IRPAREA_OP : irpareaOperation(e);
      break;

    }
}

```

```

public void mouseEntered(MouseEvent e)
{

}

```

```

public void mouseExited(MouseEvent e)
{

    //updateMouseCoordinates(e);
}

```

```

/*
Method updates mouse coordinates if mouse has moved
*/
public void mouseMoved(MouseEvent e)
{
    //updateMouseCoordinates(e);
}

```



```

public void mouseReleased(MouseEvent e)
{
    switch (opStatus)
    {
        case IRP_OP : irprel(e);
            break;
    }
}

```

```

public void mouseDragged(MouseEvent e)
{
    switch (opStatus)
    {
        case IRP_OP : irpdrg(e);
            break;
    }
}

```

```

/*
Method updates mouse coordinates if mouse has been pressed
*/

```

```

public void mousePressed(MouseEvent e)
{
    //updateMouseCoordinates(e);
}
}

```


APPENDIX II

Research paper based on the project

A Graphical Tool Designed to Deploy Wireless Sensors in Homogeneous Grid Selected from Irregular Polygon

Rachit Singhal, Mayank Gaur and Nitin

Department of CSE and ICT

Jaypee University of Information Technology,

P.O. Wahnaghat, Solan-173234, Himachal Pradesh

rachitsinghal23@gmail.com

ABSTRACT

In this paper, we present a graphical tool, which will help us with algorithms that are used for finding Quality of Service. This tool relates these algorithms to the real world and thus provides a better insight into the actual problems of deploying sensors in a field and finding out whether QoS would be achieved or not. We have used Mathematical Game and its application to deploy the sensors in the homogeneous are, which is selected from the Irregular Polygon.

KEY WORDS

Wireless Sensor Networks, Game, Quality of Service, Homogeneous Grid, Irregular Polygon.

1. Introduction

A WSN consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants [1-3] and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, enabling also to control the activity of the sensors. The development of WSNs was motivated by military applications such as Battlefield Surveillance; today such networks are used in many industrial and consumer application, such as Environmental Monitoring, Habitat Monitoring, Acoustic Detection, Seismic Detection, Military Surveillance, Inventory Tracking, Medical Monitoring, Smart Spaces and Process Monitoring [1-5]. Sensors are low-powered devices with very little computational capability. WSNs allow sensors to work collectively for the purpose of exhaustive covering of the wide area. In any conventional application, data is collected with the help of sensor nodes of a WSN spread across a region.

1.1 Limitations of WSNs

- **Hostile Environment:** Unfriendly or remote environments like battlefields can also serve as locations for the deployment of sensor networks. Since these locations are easily accessible to anyone, there is a threat of physical attacks in these regions, from which it is difficult or may be impossible to protect the nodes.
- **Random topology:** It is a difficult task to deploy a sensor network in a hostile or remote location and thus sensors are usually distributed randomly in such regions usually with the help of an aeroplane. Because of the random distribution, it becomes difficult to know the sensor networks' topology. It is likely that some sensors may be covering the same region and they may gather similar data, resulting in too much redundancy of data sent back to base station. This is good to some extent but will result in sensors dying rapidly and will require frequent re-deployment, which is very difficult in many of the scenarios in which sensors are used. In order to avoid this it is required that sensors can act together and co-operate in a way that they cover a maximum area with a minimum number of nodes active at a given time (Maximum Cover with Minimum Node: MCMN). [6]
- **Limited Resources:** Under this category, we have Power restrictions, Limited Computational Power and Storage Restrictions. The limitation of power in sensor networks is mainly attributed to its puny physical design and its actual nature of being wireless. Sensor nodes are driven by batteries as there is no constant supply power through wires. It is practically not feasible to recharge or replace the batteries of the sensor nodes because sensor networks are usually huge containing hundreds to thousands or even more nodes and also because WSNs are usually deployed in hostile

environments. The need for power in a sensor node is to carry out its various operations like information processing, communication of data and running of nodes.

Computational power is closely linked to the available amount of power with the sensor nodes and thus the limitation is induced. However communication of data requires more power than that required for computational process. There are limitations on storage as well as it also consumes power. The limitation affects the storage and usage of the cryptographic keys also.

- **Design Challenges:** Under this category, we have Scalable and flexible architecture, Error Prone Wireless Medium and Fault-tolerance and adaptability: A need may arise for the extension of the network size and thus the network should be flexible and scalable to accommodate the changes. Deployment of more nodes should not affect the clustering and routing of the network and thus the protocols should be designed specifically. The protocols are required to adapt to any topology and should preferably be generalised. The diversity of the situations in which the sensor networks may be deployed also poses problems as the requirements may change from region to region. The noise in the environment can also affect the wireless medium and thus must be considered and dealt with.

Node failure is also a potential threat to the sensor network if it is not dealt properly. The condition may arise due to some technical failure of if the battery gets exhausted. Thus sensor networks are needed to be designed in a way such that they may handle such problems and work with new links.

In this paper, we present a graphical tool, which will help us with algorithms that are used for finding Quality of Service. This tool relates these algorithms to the real world and thus provides a better insight into the actual problems of deploying sensors in a field and finding out whether QoS would be achieved or not. In order to find out whether QoS is achieved or not we have used Gur Game Algorithm at the back end of this tool and made the front end more realistic. We will also draw conclusions from the results that we obtain about the QoS when sensors are deployed in various sizes of homogeneous grids.

1.2 Problem Description

WSNs are employed for area monitoring, environmental sensing and industrial monitoring and there are many more applications. The sensors are needed to be deployed across battlefields to surfaces of glaciers. "In any sensor network, we want to accomplish two things: (1) maximize the lifetime of the sensor network by powering down sensors periodically, and (2) have enough sensors powered-up to cover maximum area." [6] To achieve this we simulate the results before actually deploying the sensors through some Quality of Service finding algorithms. These algorithms are rigorously tested for optimal outputs for different sensor count for a specific area. Talking specifically for the homogeneous algorithms like Gur Game Algorithm we need to specify a region size in terms of a matrix. Simulating the region size at a remote location is a difficult task and there is no surety about it while entering that in numbers. We can never be sure about a particular location by just assuming it. There is a need for these algorithms to be related to the real-world such that simulations can be more specific and accurate thus eventually resulting in optimal deployments. There is a need of linking these simulations with reality. We will run this simulation and try to infer from the results the limitations and the actual need and meaning of quality of service.

1.3 Contributions

Quality of Service is a major aspect while employing WSNs. Maximum output and efficiency is required out of the WSNs as deploying sensors in most of the regions is a very difficult and expensive task. Constant need of sensors would lead to increased costs. Thus before employing simulations are carried out so that to reduce future costs and there is always a pressure that these simulations should be more and more realistic. In [6, 7] an interesting approach based on a random technique called the Gur Game was introduced for simulating and finding whether a particular configuration is effective or not. We have added a paradigm to this simulation by providing this algorithm a graphical outlook. Gur Game Algorithm is used here at the back end to check for QoS and at the front end the tool provides a more realistic and effective way of choosing our area where we need to deploy. The grid size used by the tool can be scaled to that on the map and thus we can select our region to deploy sensors in a better way and not just by guessing. Gur Game Algorithm requires a $n \times n$ matrix as a region where sensors can be deployed. In real physical world we don't have such matrices. Thus this tool simulated the real physical world and out of it we can choose the square matrix required by the algorithm. This will enhance the simulations and we can get better outputs and thus effectively deploying sensors which would eventually be cost effective.

2. Preliminaries and Background

The sensors are volatile in nature and on other hand; the WSNs are dynamic in nature. This will make difficult to understand and predict the behaviour of Sensor Networks. In order to achieve the primary goal of energy conservation

numerous techniques have been proposed for node placement and the management of self optimized WSNs. Many researchers have presented their work in designing the specialized energy-aware routing protocols that consume less energy than others while routing data. [8-10, 15] However, such techniques lack proper management of real-time traffic, maximum area coverage or desired redundancy of sensors. Moreover, every protocol has its own advantages and disadvantages.

The authors of [11] proposed that some protocols are for real-time environments where routing of data in real time concern a lot than preserving energy. On the similar lines, the authors of [12] proposed a management technique to maximize object detection or coverage instead of energy conservation. Incorporating fault tolerance in managing self optimized WSNs was studied in [13, 14]. The authors of [13] presented an adaptive scheme called ASCENT "Adaptive Self Configuring Sensor Network" in which initially every node discovers its neighbour and then required sensors to join network and others will wait while probing if they are required to join the network in the future.

The authors of [6] have presented a hierarchical Gur Game. It signifies that Instead of playing the Gur Game between all the sensor nodes simultaneously, the sensor nodes are clustered in the form of a tree and then super-nodes (roots of each sensor tree) are used to play the Gur Game in each cluster. The output of the research is excellent as it converges more quickly.

A random and greedy mathematical paradigm of the Gur Game has been proposed. This technique allow the nodes to communicate with the base station at a regular interval of time, and base station send feedback in order to manage nodes on basis of packets received and the result is a robust sensor network that allows the base station to dynamically adjust the resolution of a network based on feedback it receives from the sensors in the network. [6, 7, 15] Recently self optimizing algorithms to regulate the process of activating sensors while maximizing the number of regions covered by sensor nodes have been introduced in which a dynamic clustering algorithm that employs the concept of connected dominating sets was proposed and the earlier Ants Algorithm and Genetic Algorithm to take into consideration the dynamic nature of WSNs was also improved. [15, 16]

3. Testbed, Experimental Setup and Simulation Outputs

We have used Java Technology (i.e. JDK 1.6) for simulating and this software is running on top of the HP System, running with Windows Vista Home Basic operating system. We have made use of Java Applets and all the panels have been incorporated on the same applet. The tool can be used in the following different phases:

- Selecting an irregular area on the screen by means of an irregular polygon. The area selected will be shown in blue.
- Scanning the whole area selected with the help of polygon through a function called Get Area which selects all the homogeneous grids, i.e. the grids which totally lie inside the selection. The area selected will be shown in red.
- Once the grids have been selected, we can select a $n \times n$ grid structure inside these homogeneous grids where the sensors are needed to be deployed. The area selected will be shown in turquoise blue and the number of grids selected will be shown in the Area field.
- Once this process of selecting the required grid size is finished, we can increase the number of sensors in the field given and use the Deploy function to deploy them.
- The tool will separately ask for each sensor that whether it is active or not and based upon our input the Gur Game Algorithm will run in the background finally telling whether QoS was achieved or not.

3.2 Selecting an Irregular Area

The tool is designed to make the process of selecting grids simple and more realistic and so that we can relate it to the real world. So the first step is to select an approximate area where we wish to deploy our sensors. The grids on the screen would themselves divide the selected area.

The user can click on the Grids option any time while using the tool to divide the screen into grids or to highlight the division if it's already divided. Clicking on the Irregular Polygon button sets the user to select an approximate area on the screen in the form of an irregular polygon. The user just needs to click on the screen where the vertices of the polygon are required and the polygon will start taking its shape from the third vertex onwards as shown in Fig.1.

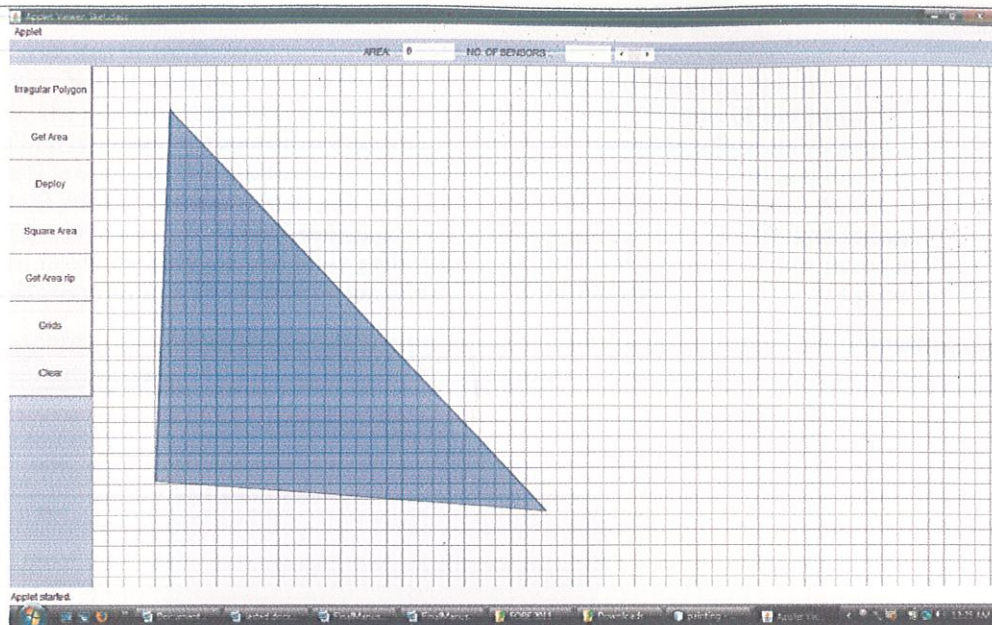


Figure 1. Irregular Polygon after 3 clicks

The logic behind this is simple and all the clicks are stored and the passed on as arguments to the `java.awt.Polygon` class in JAVA and the figure takes its shape. The user can increase the vertices to be precise and finally a figure is selected as in Fig.2.

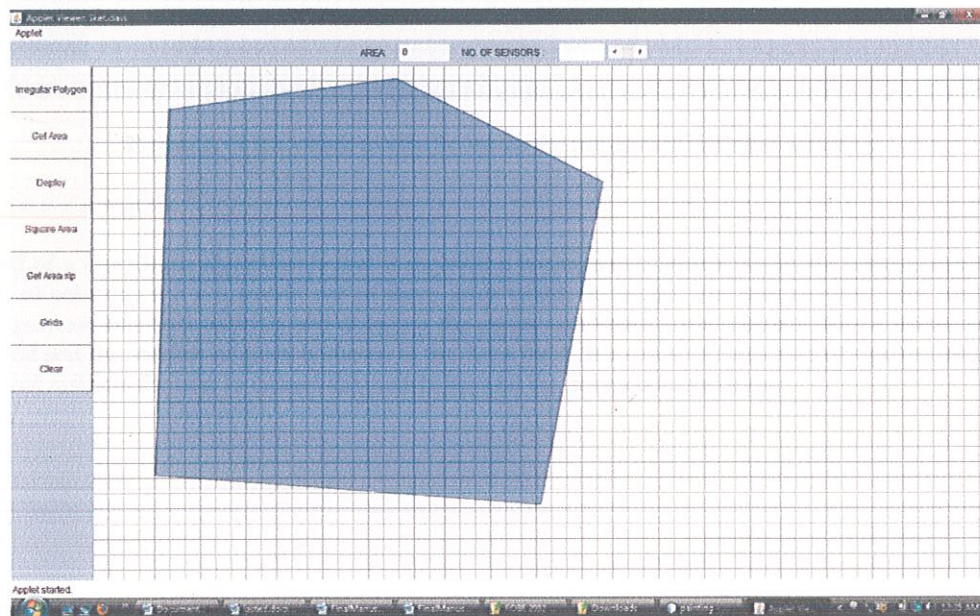


Figure 2. Approximate area selected through irregular polygon

3.2 Scanning the Irregular Area

The approximate area selected is needed to be refined and that is done through scanning the homogeneous grids in the approximate selection. Only the grids the totally lie in the region are needed and rest have to be discarded. The user has to

click on the Get Area button for this functionality. Once the user has clicked the tool scans the polygon from left to right and selects all grids that lie totally in it.

The algorithm behind scanning the irregular polygon is simple and does take care of concave polygons by using the odd parity rule. The grid size is uniform and known and thus the all four corners of any particular grid can be calculated. The algorithm first calculates the equation of lines of the boundaries of polygon from its vertices. Then scanning is started from the leftmost to the rightmost position and for each vertical grid line. Each vertical grid line is then scanned from top to bottom and all the points of intersection of grid lines that lie inside the polygon are stored. At last, if a point is inside along with its 3 neighbours in the right, bottom and diagonally right-bottom direction, then the grid formed by these four points is said to be lying inside the polygon. All these grids that lie inside the polygon are shown in red in Fig.3. The number of grids selected in this process is also shown on top of the Fig.3 in the Area box; in this case 413 grids are selected overall.

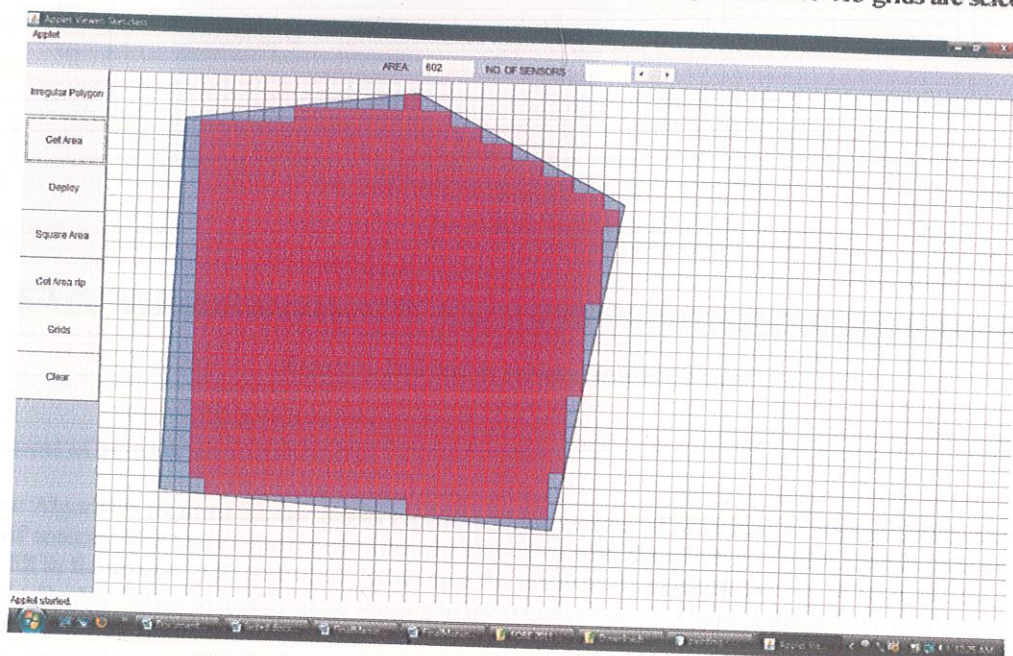


Figure 3. Grids which totally lie inside the polygon in red after scanning

3.2 Selecting a $n \times n$ matrix

The actual need of a Gur Game algorithm is that it requires a homogenous matrix where the sensors can be deployed. So there is a need of selecting a $n \times n$ matrix out of these grids which lie inside. The user need to click on the Square Area button for this purpose and then clicking and dragging the cursor can select a $n \times n$ matrix. Fig. 4 shows that a 8×8 matrix has been selected and the number of grids is shown in the Area field. We can select any size of square matrix that fits in the polygon. The tool restricts the user to extend this matrix beyond the polygon boundaries.

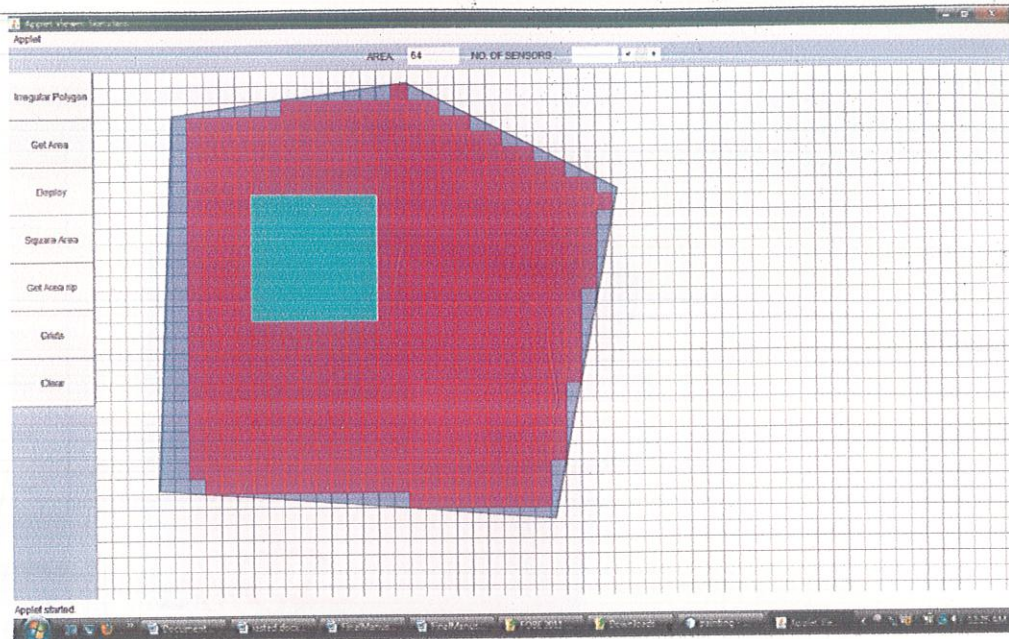


Figure 4. 8x8 matrix selected inside the area.

3.2 Deployment of Sensors

The Gur Game Algorithm requires the number of regions and the number of sensors to be deployed as input along with the initial status of sensors. The square matrix provides the algorithm with the number of grids (in this case 64) and now the tool needs to fill in for sensors. The user needs to increase the number of sensors in the No. of Sensors field and then has to just click on the Deploy button to deploy these sensors in the region selected. The tool will ask the user to provide the initial status of each sensor that whether it is active or not. Once the user provides the necessary inputs the Gur Game Algorithm runs in the background to check if Quality of Service is achieved or not. Fig. 5 shows 160 sensors being deployed and a dialogue box asking for status of 78th sensor.

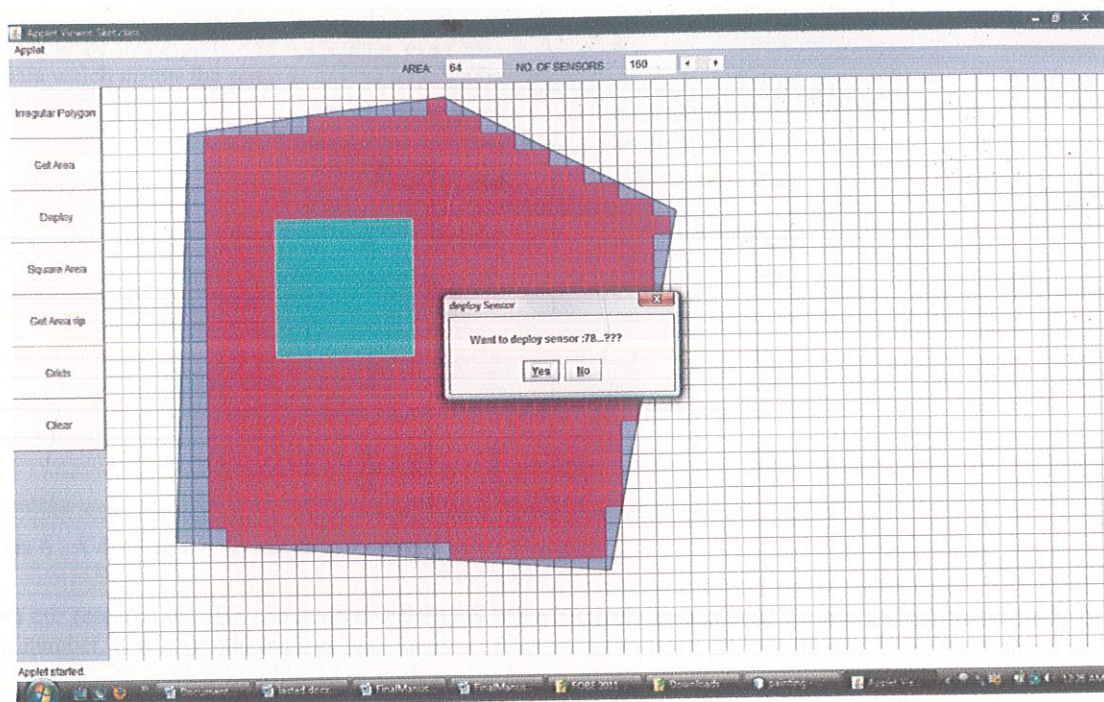


Figure 5. Sensors being deployed and tool asking for status of sensors

4. Results and Conclusion

We have run the simulation for different grid sizes and the results are as shown in table 1.

Table 1
Results of simulation

Grid Size (n x n)	No. of sensors deployed	QoS of 0.4 achieved?
4 x 4	40	No
5 x 5	63	No
6 x 6	90	No
7 x 7	123	No
8 x 8	160	No
9 x 9	203	No
10 x 10	250	No

We see that the QoS of 0.4 is not achieved for any of the grid sizes. This is because of the reason that the Gur Game algorithm predicts the status of each sensor randomly. Moreover, the sensors are deployed in a random manner and do not guarantee that all the regions are covered effectively.

As per the readings that we get and noticing the deployment of sensors at the background we come to a series of conclusions. They are enumerated below:

- 1) We conclude from our research that effective QoS can be achieved only at a particular placement of sensors in the grid. On deploying randomly, we cannot ensure the sensors to align themselves in that particular fashion at which quality of service is achieved. Gur Game rewards each sensor randomly as per it's given function in order to align them in that particular formation, but since this function uses random numbers there is no guarantee if the desired result will be achieved.
- 2) QoS is defined mathematically as number of regions covered upon number of sensors deployed. This definition of QoS has its own limitations. For example, when sensors are distributed randomly then there is a possibility that no

regions are covered or there may be a sub-grid (a smaller portion of a grid) which has sparsely distributed sensors while the other region is densely populated by sensors as shown in Figure 6. Therefore, calling quality of service at 0.4 would be incomplete and sometimes even irrelevant. Rather we should say that each region is covered by 2.5 sensors which means the same as having a QoS of 0.4 but it would ensure that the sensors are evenly distributed also.

3	3	5	2
1	2	3	4
1	1	3	5
1	1	2	3

Figure 6 . A 4 x 4 grid with number of sensors being displayed

- 3) From our research we also conclude that the QoS can effectively be achieved for a $n \times n$ grid if and only if n is an even number. If n is an odd number then number of regions in an $n \times n$ grid would also be an odd number and as we know quality of service is achieved at each region being monitored by 2.5 sensors which would not be practically feasible as we would be requiring a fraction of sensors to cover the whole grid. Table 2 illustrates this phenomenon for various $n \times n$ grid and the number of sensors required for each grid to achieve the QoS of 0.4.

Table 2
Achievable QoS of 0.4 in grids of $n \times n$, where n is even

Grid Size ($n \times n$)	No. of Sensors deployed	QoS
4 x 4	40	0.4
6 x 6	90	0.4
8 x 8	160	0.4
10 x 10	250	0.4
12 x 12	360	0.4

REFERENCES

- [1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, Wireless Sensor Networks: A Survey, Computer Networks, 38 (4), pp. 393-422, 2002.
- [2] R. Kay and F. Mattern, The Design Space of Wireless Sensor Networks, IEEE Wireless Communications 11 (6), pp. 54-61, 2004.
- [3] L. Wang, Survey on Sensor Networks, Department of Computer Science and Engineering Michigan State University.
- [4] A. Tiwari, Energy-efficient Wireless Sensor Network Design and Implementation for Condition-based Maintenance, ACM Transactions on Sensor Networks 3 (1), pp. 1-23, 2007.
- [5] S. Hadim and M. Nader, Middleware Challenges and Approaches for Wireless Sensor Networks, IEEE Distributed Systems Online 7(3), pp. 1-23, 2006.
- [6] R. Iyer and L. Kleinrock, QoS Control For Sensor Networks, In Proceedings of the IEEE International Conference on Communications, pp. 517-521, 2003.
- [7] B. Tung and L. Kleinrock, Using Finite State Automata to Produce Self-Optimization and Self-Control, IEEE Transactions on Parallel and Distributed Systems 7 (4), pp. 439-448, 1996.
- [8] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, In Proceedings of the MOBICOM, pp. 56-67, 2000.
- [9] Q. Li, J. Aslam, and D. Rus, Online Power-aware Routing in Wireless Ad-hoc Networks, In Proceedings of the MOBICOM, pp. 97-107, 2001.
- [10] S. Dulman, T. Nieberg, P. Havinga and P. Hartel, Multipath Routing for Data Dissemination in Energy Efficient Sensor Networks, In Proceedings of CTIT-02-20, pp. 6-163, 2002.
- [11] T. He, J. Stankovic, C. Lu and T. Abdelzaher, SPEED: A Real-Time Routing Protocol for Sensor Networks, In Proceedings of the IEEE Distributed Computing Systems, pp. 46-55, 2003.
- [12] B. Krishnamachari, Y. Mourtada and S. Wicker, The Energy-Robustness Tradeoff for Routing in Wireless Sensor Networks, In Proceedings of the IEEE ICC, pp. 1833-1837, 2003.
- [13] A. Cerpa and D. Estrin, Ascent: Adaptive Self-configuring Sensor Networks Topologies, IEEE Transactions on Mobile Computing, 3 (3), pp. 1-14, 2004.
- [14] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava, Topology Management for Sensor Networks: Exploiting Latency and Density, In Proceedings of the 3rd ACM International Symposium on Mobile Adhoc Networking & Computing, pp. 135-145, 2002.
- [15] S.I. Nayer and H. Ali, A Dynamic Energy-Aware Algorithm for Self-Optimizing Wireless Sensor Networks, Self Organizing Systems, Lecture Notes in Computer Science 5343, pp. 262-268, 2008.
- [16] S.I. Nayer and H. Ali, On Employing Distributed Algorithms and Evolutionary Algorithms in Managing Wireless Sensor Networks, Proceedings of the International Workshop on Theoretical and Algorithmic Aspects of Wireless Ad Hoc, Sensor, and P2P Networks, Chicago, Illinois, 2004.