VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum, Karnataka-590014

A Project Report
On

"Low-Priced and Energy-Efficient Detection of Replicas for Wireless Sensor Networks"

Submitted in partial fulfillment of the requirements for the award of the degree

Bachelor of Engineering
In
Information Science and Engineering

Submitted By

PARAS RAJ PAHARI	**1SJ11IS062**
RAJASHEKAR G	**1SJ11IS069**
SUNIL KUMARA M.A	**1SJ11IS096**
SAIRAM K	**1SJ11IS122**

Under The Guidance Of
Prof. NAGARAJA G
Associate Professor
Department Of ISE, SJCIT

S.J.C. INSTITUTE OF TECHNOLOGY
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
CHICKABALLAPUR-562101

2015

S.J.C INSTITUTE OF TECHNOLOGY

CHICKBALLAPUR-562101

CERTIFICATE

This is to certify that the project work entitled "**Low-Priced and Energy-Efficient Detection of Replicas for Wireless Sensor Networks"** is bonafide work carried out by **PARAS RAJ PAHARI, RAJASHEKAR G, SUNIL KUMARA M.A, SAIRAM K** bearing **USN: 1SJ11IS062, 1SJ11IS069, 1SJ11IS096, 1SJ11IS122** are students of **S J C Institute of Technology** in partial fulfilment for the award of Bachelor of Engineering in Information Science and Engineering of **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belgaum** during the year **2015**. It is certificated that all corrections / suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Signature of Guide	**Signature of HOD**	**Signature of Principal**
Prof. NAGARAJA G.	**Prof. SATEESH CHANDRA REDDY**	**Dr. T.MUNIKENCHE GOWDA**
Associate Professor	Associate Professor & HOD	Principal
Dept. Of ISE, SJCIT	Dept. of ISE, SJCIT	SJCIT

Name of the Examiners: **Signature with Date:**

1……... ……..

2. ……. …….

ABSTRACT

Wireless sensor nodes are embedded in the daily lives of users—poses new challenges to security and end-user privacy. One of the major challenge is preventing sensor nodes from replica attacks. Sensor node can be physically captured, reprogrammed and replicated into a multiple number of replicas to maliciously occupy the network. Several schemes have been proposed for detection of replicas; however, most of them require expensive hardware such as a global positioning system. In general, the ideal price for a sensor node is as low as one dollar, and thus, it is equipped with limited resources; hence, it is not practical to employ additional devices. In this paper, we propose a low-priced and efficient solution for replica detection in static wireless sensor networks. Although the proposed solution does not need any additional hardware, it exhibits similar or better performance, as compared to existing schemes. Through simulation experiments, we show that the proposed solution provides comparable performance in terms of the replica detection ratio and the time required to detect replicas. Furthermore, we show that the proposed solution saves more energy in comparison to existing schemes in most of our simulations.

ACKNOWLEDGEMENT

The completion of project work brings with a sense of satisfaction, but it is never complete without thanking the person responsible for its successful completion.

First and foremost we would like to express our heartfelt gratitude and sincere thanks to **Dr. T. MUNIKENCHE GOWDA**, **Principal of SJCIT** for providing us the infrastructure and facilities for the completion of our project.

We would also thank to **Prof. Sateesh Chandra Reddy**, Professor & Head of Information Science and Engineering Department, **SJCIT** for his immense guidance which helped us in the completion of our project.

We are extremely thankful to our guide **Prof. Nagaraja G., Associate Professor, Department of ISE** for his valuable guidance, support and advice during the course of project.

We also wish to thank all our teachers and staff of department of ISE, **SJCIT** for their constant help and encouragement which has helped a lot in the successful completion of our project.

Finally, we would like to thank our parents, brother, sister, friends and all those who were directly or indirectly involved in successful completion of the project.

PARAS RAJ PAHARI **1SJ11IS062**

RAJASHEKAR G. **1SJ11IS069**

SUNIL KUMARA M.A **1SJ11IS096**

SAIRAM K **1SJ11IS122**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 An Overview

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.

The main characteristics of a WSN include:

- Power consumption constrains for nodes using batteries or energy harvesting
- Ability to cope with node failures
- Mobility of nodes
- Communication failures
- Heterogeneity of nodes
- Scalability to large scale of deployment
- Ability to withstand harsh environmental conditions
- Ease of use

Wireless Sensor Networks have grown its applications widely, they also have faced various security challenges, as compared to traditional networks, because sensor nodes generally lack hardware support for tamper-resistance and are often deployed in physically insecure environments, where they are vulnerable to capture and compromise by attackers. A harmful consequence of a node compromise attack is that once an attacker has acquired the credentials of sensor, he/she can fabricate replicas with these credentials and insert them at selected target positions within the network. These replicas can be used to launch various stealth attacks depending on the attacker's motives, such as eavesdropping on network communications or controlling the target areas. This type of attack is called a **Replica Attack**.

Our challenges would be to detect such replica attack low priced and energy efficient way in Wireless Sensor Network.

Figure 1.1 Replication of Nodes

1.2 Objective

Wireless sensor network have been using the expensive methodologies to maintain the security threats and issues. The methods we are approaching aims in maintaining the security aspects of wireless sensor networks in cost effective way. In compared to existing schemes where they have higher consumption of energy our approach aims in less energy consumption. It further provide assistance in scalability for large scale wireless sensor networks. It also aims in providing better detection ratio of replica nodes.

1.3 Scopes of the Project

Wireless sensor networks can achieve scalability, better detection of attacker nodes with less energy consumption at most cost effective way. It can also further enhance the throughput and overcome the communications overhead. With Only use of Bloom Filter Algorithm we can implements Low Priced Energy Efficiency Wireless Sensor Networks. We show how this is achieved through simulation.

1.4 Motivation

In spite of the low price of a sensor node, most existing schemes in static Wireless Sensor Networks assume that a sensor node has expensive hardware, i.e., a global positioning system (GPS) receiver for acquiring the location information of a sensor node, which is used as proof of identification. The intensive approach used in existing schemes greatly increases the unit price of a sensor; hence, it is not suitable for resource-limited sensor applications.

Thus existing schemes being very costly, our motivation has been to design a low priced replica detection approach for wireless sensor networks which can provide solution just by using simple probabilistic memory efficient data structure algorithm “Bloom Filter”.

1.5 Problem Statement

To design a Wireless Sensor Network that provides energy efficient and cost effective detection of replica nodes in wireless sensor networks, the proposed system uses Bloom filter based approach to replace the functions of expensive hardware. With the use of Bloom filter approach the proposed system can as well deliver the scalability and better detection ratio of replicas in wireless sensor network.

1.6 Report Organization

Chapter 1: Introduction

It gives the overall information about the project. The Background knowledge needed for the project, problem definition, and main objectives of the project, scopes, and challenges.

Chapter 2: Literature Survey

This includes previous methods or study of various Replica Detection for Wireless Sensor Networks.

Chapter 3: System Analysis

This Chapter will give detailed information of existing and proposed system

Chapter 4: System Requirement Specification

This chapter will give functional, non-functional, hardware and software requirements necessary to develop project

Chapter 5: System Design

This chapter displays system architecture, to give a wholesome overview of undertaken concept. It will also discuss some prime data flow diagram, usecase and sequence diagrams for the proposed system.

Chapter 6: Implementation

This chapter deals with the steps involved in the creation of the project work. It is defined with the assistant of code explanation for the ease of reader

Chapter 7: System Testing

This chapter mainly deals with various types of test cases such as unit testing and integration testing to prove the validity of the project.

Chapter 8: Performance Analysis

This chapter deals with the performance comparison of existing schemes with proposed system and why proposed system has better performance.

Chapter 9: Conclusion and Future Enhancement

This chapter mainly deals with the summary of the entire project development and it also suggest some of the enhancement idea which couldn't be covered up due to constraint.

Chapter 2

LITERATURE SURVEY

a) "Security and Privacy Challenges in the Internet of Things,"

AUTHORS: C.P. Mayer

In the past decade, internet of things (IoT) has been a focus of research. Security and privacy are the key issues for IoT applications, and still face some enormous challenges. In order to facilitate this emerging domain, we in brief review the research progress of IoT, and pay attention to the security. By means of deeply analyzing the security architecture and features, the security requirements are given. On the basis of these, we discuss the research status of key technologies including encryption mechanism, communication security, protecting sensor data and cryptographic algorithms, and briefly outline the challenges.

b) "Distributed Detection of Node Replication Attacks in Sensor Networks,"

AUTHORS: B. Parno, A. Perrig, and V. Gligor

The low-cost, off-the-shelf hardware components in unshielded sensor-network nodes leave them vulnerable to compromise. With little effort, an adversary may capture nodes, analyze and replicate them, and surreptitiously insert these replicas at strategic locations within the network. Such attacks may have severe consequences; they may allow the adversary to corrupt network data or even disconnect significant parts of the network. Previous node replication detection schemes depend primarily on centralized mechanisms with single points of failure, or on neighborhood voting protocols that fail to detect distributed replications. To address these fundamental limitations, two new algorithms based on emergent properties (Gligor (2004)),

i.e., properties that arise only through the collective action of multiple nodes. Randomized multicast distributes node location information to randomly-selected witnesses, exploiting the birthday paradox to detect replicated nodes, while line-selected multicast uses the topology of the network to detect replication. Both algorithms provide globally-aware, distributed node-replica detection, and line-selected multicast displays particularly strong performance characteristics. These emergent algorithms represent a promising new approach to sensor network security; moreover, their results naturally extend to other classes of networks in which nodes can be captured, replicated and re-inserted by an adversary.

c) "Distributed Detection of Clone Attacks in Wireless Sensor Networks,"

AUTHORS: M. Conti, R.D. Pietro, L. Mancini, and A. Mei

Wireless Sensor Networks (WSNs) are often deployed in hostile environments where an adversary can physically capture some of the nodes, first can reprogram, and then, can replicate them in a large number of clones, easily taking control over the network. A few distributed solutions to address this fundamental problem have been recently proposed. However, these solutions are not satisfactory. First, they are energy and memory demanding: A serious drawback for any protocol to be used in the WSN-resource-constrained environment. Further, they are vulnerable to the specific adversary models. The contributions of this work are threefold. First, analyze the desirable properties of a distributed mechanism for the detection of node replication attacks. Second, shows that the known solutions for this problem do not completely meet our requirements. Third, a new self-healing, Randomized, Efficient, and Distributed (RED) protocol for the detection of node replication attacks, shows that it satisfies the introduced requirements. Finally, extensive simulations shows that this protocol is highly efficient in communication, memory, and computation; is much more effective than competing

solutions in the literature; and is resistant to the new kind of attacks introduced in this paper, while other solutions are not.

d) "Set: Detecting Node Clones in Sensor Networks,"

AUTHORS: H. Choi, S. Zhu, and T.F.L. Porta

Sensor nodes that are deployed in hostile environments are vulnerable to capture and compromise. An adversary may obtain private information from these sensors, clone and intelligently deploy them in the network to launch a variety of insider attacks. This attack process is broadly termed as a clone attack. Currently, the defenses against clone attacks are not only very few, but also suffer from selective interruption of detection and high overhead (computation and memory). In this paper, a new effective and efficient scheme, called SET, to detect such clone attacks. The key idea of SET is to detect clones by computing set operations (intersection and union) of exclusive subsets in the network. First, SET securely forms exclusive unit subsets among one-hop neighbors in the network in a distributed way. This secure subset formation also provides the authentication of nodes' subset membership. SET then employs a tree structure to compute non-overlapped set operations and integrates interleaved authentication to prevent unauthorized falsification of subset information during forwarding. Randomization is used to further make the exclusive subset and tree formation unpredictable to an adversary. We show the reliability and resilience of SET by analyzing the probability that an adversary may effectively obstruct the set operations. Performance analysis and simulations also demonstrate that the proposed scheme is more efficient than existing schemes from both communication and memory cost standpoints.

e) "DHT-Based Detection of Node Clone in Wireless Sensor Networks,"

AUTHORS: Z. Li and G. Gong

Wireless sensor networks are vulnerable to the node clone attack because of low-cost, resource-constrained sensor nodes, and uncontrolled environments where they are left unattended. Several distributed protocols have been proposed for detecting clone. However, some protocols rely on an implicit assumption that every node is aware of all other nodes' existence; other protocols using a geographic hash table require that nodes know the general network deployment graph. Those assumptions hardly hold for many sensor networks. In this paper, we present a novel node clone detection protocol based on Distributed Hash Table (DHT). DHT provides good distributed properties and our protocol is practical for every kind of sensor networks. It analyzes the protocol performance theoretically.

Chapter 3

SYSTEM ANALYSIS

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with the problem. Analysis involves a detailed study of current system, leading to specifications of new system, various operations performed by a system and their relationship within the outside the system. During the analysis, data are collected on transactions handled by present system. Keeping in the view the problem and new requirements and working out pros and cons including the new areas of the system it becomes easy to draw the exact boundary of the system under consideration. In the dynamic world, the subject system analysis mainly deals with the software development activities.

3.1 Existing System

Most existing schemes of wireless sensor networks that has been out there has assumed to be using expensive hardware. In order to prevent attacks, for every sensor nodes deployed in the field, expensive tamper resistance are deployed. In order to generate location proof id costly device such as global positioning system (i.e. GPS) are found to be used. These hardware used in Existing schemes greatly increase the unit price of a sensor. For Example Sensor Nodes with GPS is 50 euros and Sensor nodes without GPS is 5 euros. The use of such expensive hardware increases the unit price of sensor, hence, it’s safe to assume that it is not suitable for resource limited sensor applications.

3.1.1 Disadvantages of Existing System:

- Generates Heavy Traffic by transmitting several GPS location proof id for every sensor nodes
- Since it uses tamper resistance hardware and Global positioning System devices for every sensor nodes deployed in field, it's going to cost very high
- It has to generate GPS location proof id and has to broadcast the proof, it need n^2 broadcasting messages thus congesting the traffic in network. It requires to heavy processing power to generate and compute each proof.
- Use of expensive hardware makes it inapplicable to use in large scale wireless sensor networks.
- It consumes high energy and also need to be provide heavy battery power for sensor nodes to harvest energy source.

3.2 Proposed System

Despite low priced sensor nodes existing Wireless Sensor Networks scheme assume that a sensor node has expensive hardware, i.e., a global positioning system (GPS) receiver for acquiring the location information of a sensor node, which is used as proof of identification. The intensive approach used in existing schemes greatly increases the unit price of a sensor; hence, it is not suitable for resource-limited sensor applications.

Accordingly, without using additional hardware, we design a low-priced and energy efficient replica detection solution for static WSNs by using Bloom filter approaches. The proposed solution uses neighboring node IDs, instead of location information, in order to detect replicas. Neighboring node IDs are presented with a constant size using a Bloom filter. The Bloom filter output (BFO) is used as a proof. A newly deployed node generates different proofs according

to the collected neighboring node IDs, until collecting the entire neighboring node IDs. The proofs are delivered to a randomly selected node in the network. Here, the delivery frequency increases proportionally to the number of the collected neighboring node IDs. The strategy slowly increases traffic between the neighboring nodes and their randomly selected nodes; however, existing schemes generate heavy traffic by transmitting several proofs from the beginning.

3.2.1 Aim to achieve

- Develop effective Low Priced Approach: Replacing the expensive hardware used in existing schemes with just simple Bloom Filter Probabilistic data structure we hope to eliminate cost overheads.
- Energy Efficient Detection of Replicas: Since we don't use expensive hardware in proposed system there's no need for heavy power battery source to save energy source and with much more less energy consumption we hope to detect replica attacks in energy efficient way.
- Support Large scale WSN: We only use of inexpensive sensor nodes in wireless sensor networks, thus we can deploy them in large numbers and hopes to provide the scalability.

Chapter 4

SYSTEM REQUIREMENT SPECIFICATIONS

A System requirement specification (SRS) is a complete description of the behavior of the system to be developed. It includes a set of use cases that describes all the interactions the user will have with software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional requirements. Non-functional requirements are requirements which imposes constraints on the design or implementation.

4.1 Functional Requirements

The Functional Requirements Specification documents the operations and activities that a system must be able to perform.

Functional Requirements should include:

- Descriptions of data to be entered into the system
- Descriptions of operations performed by each screen
- Descriptions of work-flows performed by the system
- Descriptions of system reports or other outputs
- Who can enter the data into the system
- How the system meets applicable regulatory requirements

The Functional Requirements Specification is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document. The plan for implementing functional requirements is detailed in the system design.

Our Project "Low Priced Energy Efficient Detection of Replica "requires the following

Functional Requirements:

- Users must be able to send message from source nodes to destination node securely
- It should allow validated node to be deployed
- If attacker deploys replica nodes in field, it must be able to detect the replica node
- Replica nodes should be revoked access and shouldn't be able to perform data transmission
- Valid nodes can perform data transmission

4.2 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing non-functional requirements is detailed in the system architecture.

Portability Requirement: Since the product is developed in JAVA which is known for its portability characteristics, it can be executed on any platform with minor or no modifications

Usability Requirement: As the complete product is being developed as small chunks, it gives a large scope for reusing these chunks for similar requirements. This feature drastically reduce the development cost.

Cost Effective: Since it plans to use no additional hardware, it generally is going cost very less in effective way.

4.3 Hardware Requirements

This section enlists the hardware required to implement the product:

- System : I3 Processor
- Hard Disk : 200 GB
- Monitor : ANY
- Mouse : ANY
- Ram : 1GB

4.4 Software Requirements

This section enlists the software, platform, operating system, coding languages required to implement the product.

- Operating System : Windows OS
- Coding Language : JAVA/J2EE
- IDE : NET BEANS 8.0

Chapter 5

SYSTEM DESIGN

System Design is process through which requirements are translated into a representation of software. During the design process different system models at different levels of abstraction are developed. The goal of this document is to produce a model of our project, which is used as blueprint for the development which is used as blueprint for the development of the project. At this level of design focus is on designing which modules are integrated. System design is a modeling process, it is a solution for systematic approach to create new system. This phase acts as a bridge between the required specification and the implementation phase of the project.

5.1 Design Considerations

There are many aspects to consider in the design of a piece of software. The importance of each should reflect the goals the software is trying to achieve. Some of these aspects are:

- **Compatibility:** The software is able to operate with other products that are designed for interoperability with another product. For example, a piece of software may be backward compatible with an older version of itself.
- **Extensibility:** New capabilities can be added to the software without major changes to underlying architecture.
- **Fault tolerance:** The software is resistant to and able to recover from component failure.
- **Maintainability:** The software can be restored to a specified condition within a specified period of time. For example, antivirus software may include the ability to periodically receive the virus definition updates in order to maintain the software's effectiveness.

- **Modularity:** The resulting software comprises well defined, independent components. That leads to better maintainability. The components could then be implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.
- **Packaging:** Printed material such as the box and manuals should match the style designated for the target market and should enhance the usability. All components required for use should be included in the package or specified as a requirement on the outside of package.
- **Reliability:** The software is able to perform a required function under stated conditions for specified period of time.
- **Reusability:** The modular components designed should capture the essence of the functionality expected out of them and no more or less. This single minded purpose renders the components reusable wherever there are similar needs in other designs.
- **Robustness:** The software is able to operate under stress or tolerates unpredictable or invalid input. For example, it can be designed with resilience to low memory conditions.
- **Security:** The software should be able to withstand hostile acts and influences.
- **Usability:** The software user interface must be intuitive and often aesthetically pleasing to its target user/audience. Default values for the parameters must be chosen so that they are good choice for majority of the users.

5.2 System Architecture

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

A system architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

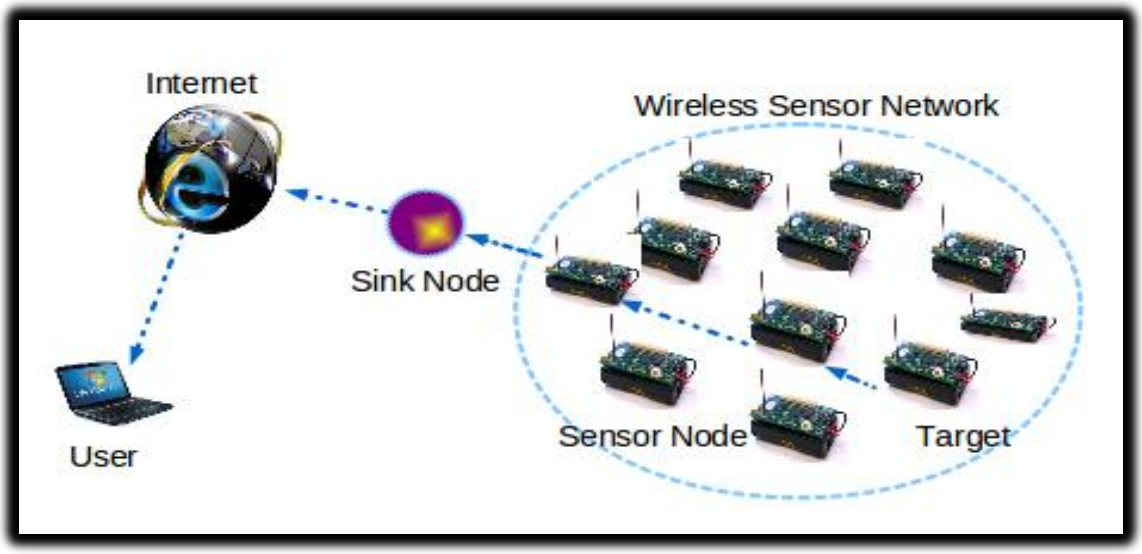

Figure 5.1 System Architecture for WSN

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external

antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.

5.3 Modular Description

List of modules present are:

- Node Formation
- Node Validation
  - BFO Proof Generation
  - BFO Node Registration
- Bloom Filter Detection
- Information Transmission

5.3.1 Node Formation

Nodes are presented with a constant size using a Bloom filter. The Bloom filter output (BFO) is used as a proof. A newly deployed node generates different proofs according to the collected neighboring node IDs, until collecting the entire neighboring node IDs. The proofs are

delivered to a randomly selected node in the network. Here, the delivery frequency increases proportionally to the number of the collected neighboring node IDs. The strategy slowly increases traffic between the neighboring nodes and their randomly selected nodes.

5.3.2 Node Validation

Here, it process through several stages:

- Proof Generation: Updates newly Created node by generating BFO output
- Proof Registration: Checks whether neighboring nodes are registered and validates it

5.3.3 Bloom Filter Detection

We assume, attacker steals the credentials of original nodes and use these credentials to form attacker node. With this, attacker creates multiple replicas by maliciously reprogramming nodes to perform various stealth attack and eaves dropping in WSN. Then attacker deploy these replica nodes in WSN field of network to compromise the network and provide false information.

Here, Bloom Filter constantly updates and generates it BFO outputs with every neighboring nodes in the group. If it finds that certain nodes is not the member of the Group, it identifies the node as replica node.

5.3.4 Information Transmission

Here after Nodes are validated, we focus on transmission of information from one nodes to another securely. With creation of security key in source node, it shares same security key with destination node and transmits the information from source node to destination nodes securely.

5.4 FLOW CHART

A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. Flowcharts are used in designing and documenting simple processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help understand a process, and perhaps also find flaws, bottlenecks, and other less-obvious features within it. There are many different types of flowcharts, and each type has its own repertoire of boxes and notational conventions. The two most common types of boxes in a flowchart are:

- A processing step, usually called activity, and denoted as a rectangular box
- A decision, usually denoted as a diamond.

A flowchart is described as "cross-functional" when the page is divided into different swim lanes describing the control of different organizational units. A symbol appearing in a particular "lane" is within the control of that organizational unit. This technique allows the author to locate the responsibility for performing an action or making a decision correctly, showing the responsibility of each organizational unit for different parts of a single process.

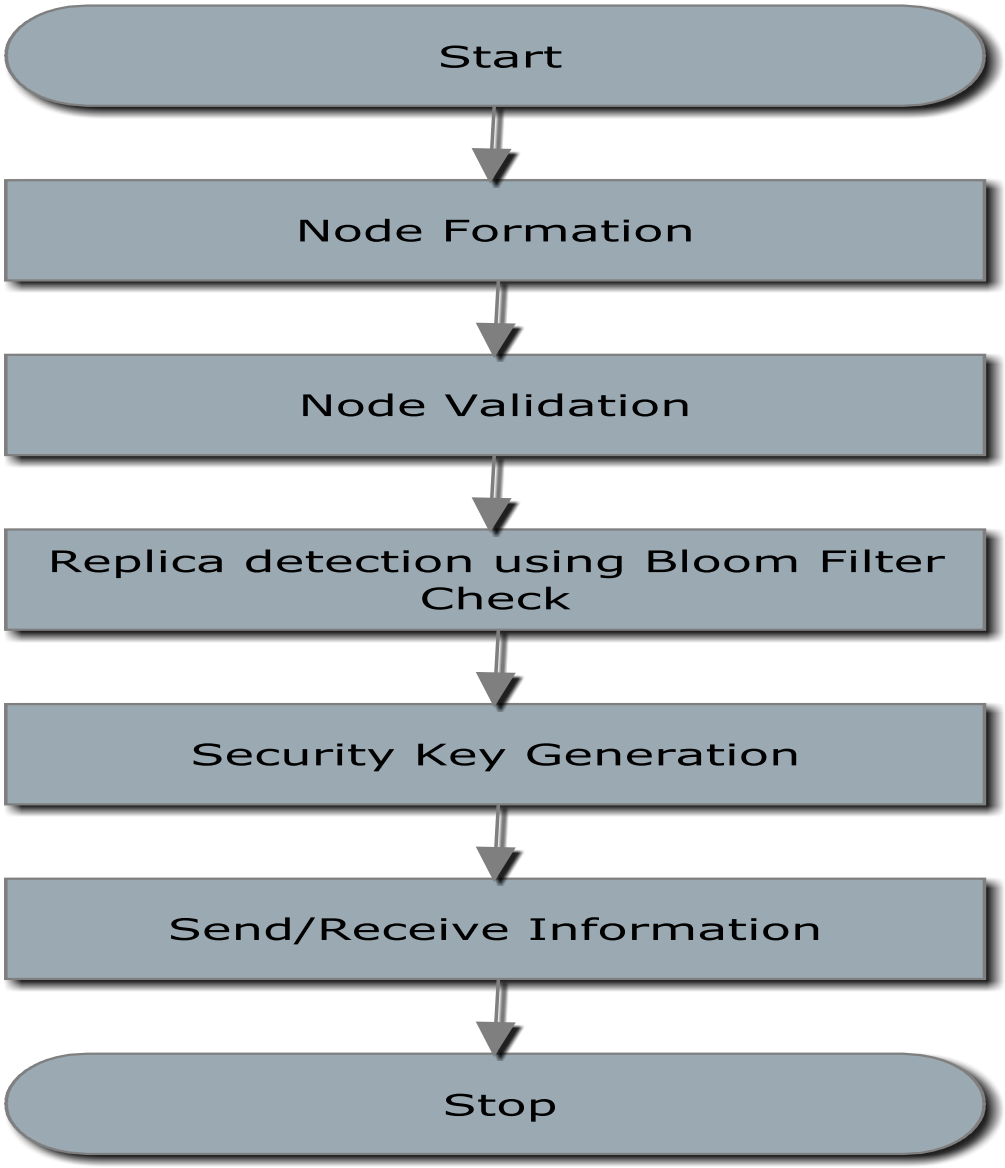

Figure 5.2 FlowChart Diagram

5.5 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.

- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

5.5.1 USECASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Here, we have several use cases and they interacts with source node, destination nodes and routers.

Several use cases used are:

- Deploy Node: Both Source and destination nodes can be deployed
- Proof Generation: Both Nodes can generate BFO proof and update it to neighbor
- Validate Node: Both Nodes can be validated
- Detection Of Replica: Router and other Nodes can identify the replica nodes with checking of BFO proof bits generated
- Socket Connection: Router and Nodes medium of interface is socket connection
- Key Generation: Security Key is generated for information transmission
- Send Data: Source nodes sends information to Destination node
- Routing: Routers need to forward data packets to Destination node

- Receive Data: Destination nodes receive Data Transmitted by Source Node

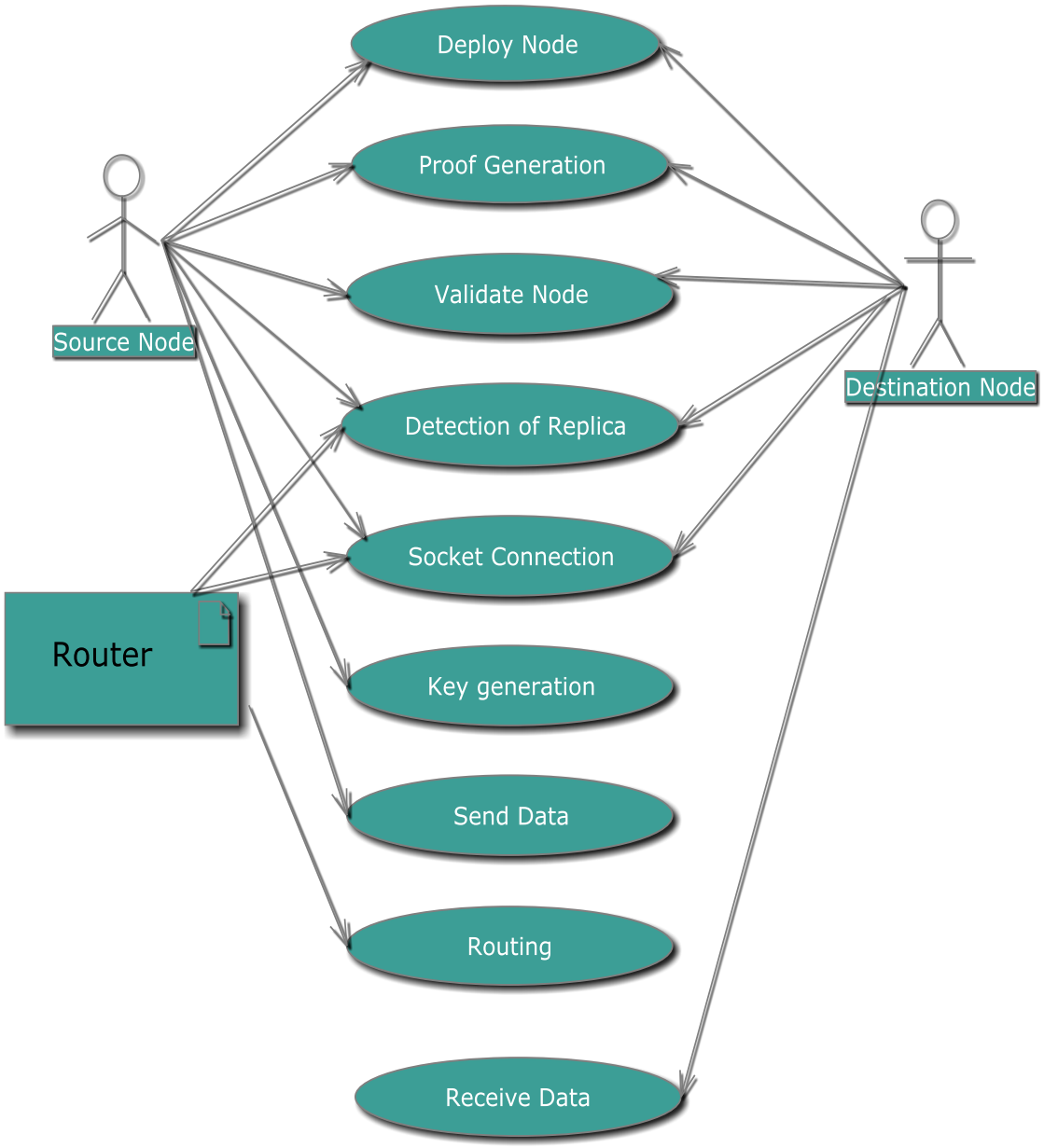

Figure 5.3 Use Case Diagram

5.5.2 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

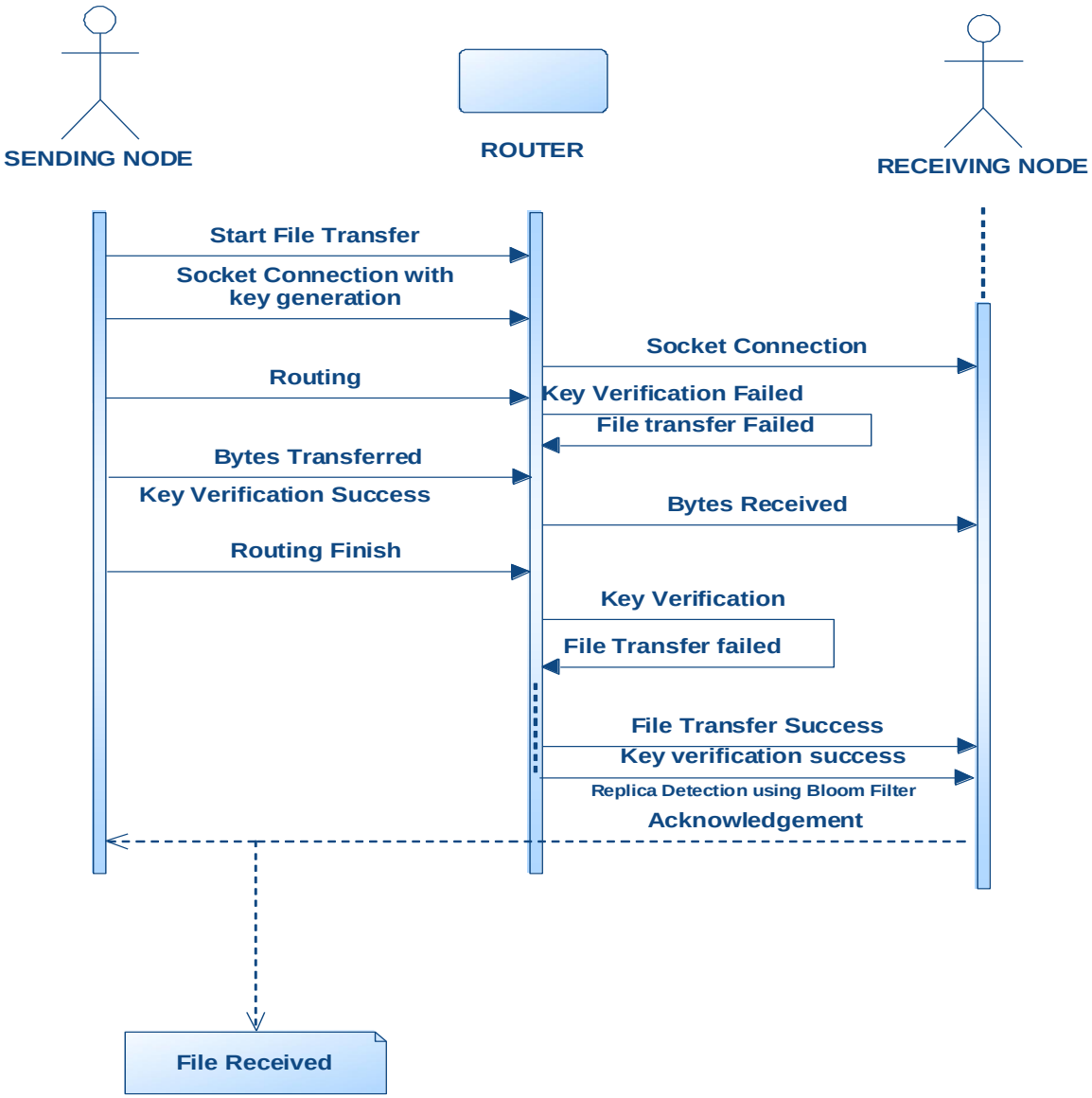

Figure 5.4 Sequence Diagram

Chapter 6

IMPLEMENTATION

The implementation phase of any project development is the most important phase as it yields the final solution, which solves the problem at hand. The implementation phase involves the actual materialization of the ideas, which are expressed in the analysis document and developed in the design phase.

Implementation should be perfect mapping of the design document in a suitable programming language in order to achieve the necessary final product. Often the product is ruined due to incorrect programming language chosen for implementation or unsuitable method of programming.

It is better for the coding phase to be directly linked to the design phase in the sense if the design is in terms of object oriented terms then implementation should be preferably carried out in an object oriented way. The factors concerning the programming language and platform chosen are described in the next couple of sections.

The implementation stage in a system project in its own right. It involves

- Careful planning
- Investigation of the current system and the constraints on implementation.
- Training of staff in the newly developed system.

Implementation of any software preceded by important decisions regarding selection of the platform, the language used, etc. These decisions are often influenced by several factors such as real environment in which the system works the speed that is required, the security concerns, and other implementation specific details.

There are three major implementation decisions that have been made before the implementation of this project .They are as follows:

- Selection of the platform (Operating system).
- Selection of the programming language/ IDE softwares for development of the application.
- Coding guideline to be followed.
-

6.1 Selection of Platform

We have selected Windows operating system where our projects is going to be hosted. We run several applications and programming environment in windows.

6.2 Software Environment

6.2.1 Java

Java technology is both a programming language and a platform.

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded

- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

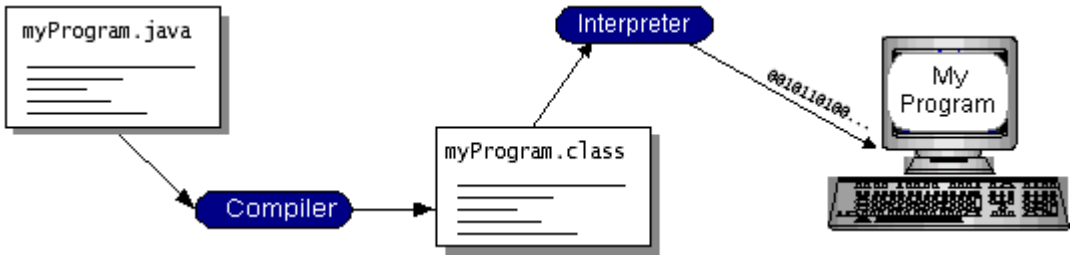

Figure 6.1 Compilation and Interpretation in Java

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

Setting up Java Environment variables:

- Right click My Computer icon

- Choose Properties from the context menu
- Click Advanced tab (Advanced system settings link in Vista)
- In the Edit windows, modify **PATH** by adding the location of the class to the value for **PATH** (i.e "C:\Program Files\Java\jdk1.8.0_25\bin").If you do not have the item **PATH**, you may select to add a new variable and add **PATH** as the name and the location of the class as the value.
- Reopen Command prompt window, and run your java code.
- Also we need to set JAVA_HOME variable , we can add its value as :

  "C:\Program Files\Java\jdk1.8.0_25"

6.2.2 Netbeans IDE

NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others. The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform (including the NetBeans IDE itself) can be extended by third party developers.

Framework for simplifying the development of Java Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications; no additional SDK is required. Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally signed upgrades and new

features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management
- Wizard framework (supports step-by-step dialogs)
- NetBeans Visual Library
- Integrated development tools

6.3 Coding

6.3.1 Node Formation

```
public Node node_form()
{
design_node(); //Design Nodes
set availability_node();
return nodes_formed;
}
```

6.3.2 Validation of Node and Bloom Filter Detection of Replicas

Our approach is mainly based in Bloom Filter Algorithm. So it is important to get insight on what is Bloom filter algorithm and how it helps to in detection of replica in energy efficient and cost effective way in Wireless Sensor networks.

Bloom Filter Explanation

A Bloom filter is a data structure designed to tell you, rapidly and memory-efficiently, whether an element is present in a set. The price paid for this efficiency is that a Bloom filter is a probabilistic data structure: it tells us that the element either definitely is not in the set or may be in the set. The base data structure of a Bloom filter is a Bit Vector.

An empty Bloom filter is a bit array of m bits, all set to 0. There must also be k different hash functions defined, each of which maps or hashes some set element to one of the m array positions with a uniform random distribution.

To add an element, feed it to each of the k hash functions to get k array positions. Set the bits at all these positions to 1.

To query for an element (test whether it is in the set), feed it to each of the k hash functions to get k array positions. If any of the bits at these positions is 0, the element is definitely not in the set – if it were, then all the bits would have been set to 1 when it was inserted

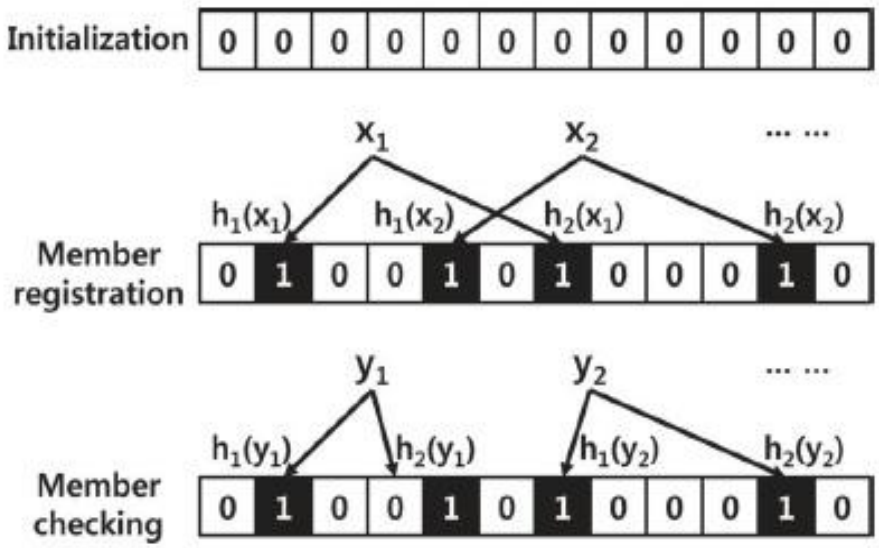

Figure 6.2 Bloom Filter Algorithm

public BitArray bits = new BitArray(32);// ***Generates 32-bit Bloom Filter Proof Bits***

public hash1(node)// ***hash method 1***

{

value=hashmethod1.node();

return value;

}

public hash2(node) // ***hash Method 2***

{

value=hashmethod2.node();

return value;

}

public add(node)//***Proof Generation and Registering nodes***

{

point1=this.hash1();

point2=this.hash(2);

this.bits[point1]=true;

this.bits[point2]=true;

}

public check(node)// ***Bloom Filter Detection for Replicas***

{

point1=this.hash1();

point2=this.hash(2);

if(this.bits[point1]&& this.bits[point2]=true) return true; // ***i.e Node is Legitimate***

else return false; // ***i.e Replica Node***

6.3.3 Information Transmission

data_send ()

{

source=action.getnode();

destination=action.getnode();

generate_key();

port=action.destination.getPort();

request_connection(port);

send(data);

```
}

data_receive()
{
receive.connection(port);
action.source.openPort();
verify_key();
receive(data);
}
```

6.4 Execution

After writing the code we can compile and run the java code to create and run jar applications through cmd line.

We can add command:

"java -jar C:\Users\Realnapster\Documents\NetBeansProjects\lowcost\dist\lowcost.jar" and execute jar file.

Else we can create the "lowcost.bat" file in desktop with command line as text content in:

"java -jar C:\Users\Realnapster\Documents\NetBeansProjects\lowcost\dist\lowcost.jar" and click it to execute every time through desktop easily.

Chapter 7

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1 Types of Testing

- **Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

- **Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were Individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

- **Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

- **System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- **White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

- **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

7.2 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Table 7.1 Node Formation Testing

Module : Node Formation	
Input	Nodes
Output	Expected Result : Nodes are formed
	Actual Result : Same as Expected
Test case: pass/fail	Pass

Table 7.2 Node Validation Testing

Module : Node Validation	
Input	Formed Nodes
Output	Expected Result : Node's Proof is generated
	Actual Result : Same as Expected
Test case: pass/fail	Pass

Table 7.3 Bloom Filter Detection Testing

Module : Bloom Filter Detection	
Input	Replica Node
Output	Expected Result : Replica Node is alerted
	Actual Result: Same as Expected
Test case: pass/fail	Pass

Table 7.4 Information Transmission

Module : Information Transmission	
Input	Data to be sent to Destination
Output	Expected Result: Data Successfully Sent
	Actual Result : Same as Expected
Test case: pass/fail	Pass

7.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Table 7.5 Integration Testing

Module	**Test case/constraints**	**Expected Results**	**Testcase Pass/Fail**
Node Formation	Node creation	Nodes Deployed	Pass
Validation Of Node	Proof Generation	Node Validated	Pass
Bloom Filter Detection	Replica Nodes	Alert for Replica Nodes	Pass
Information Transmission	Transmission of data	Validating Node, Sending Data, Verifying Key, Receiving Data	Pass

Chapter 8

PERFORMANCE ANALYSIS

The purpose of the performance analysis of this application is to cross-check the accuracy rate of all the logic used to design the product. Understanding the results obtained with help in the further analysis and research of this product.

We distinguish three basic steps in the performance analysis process: data collection, data transformation, and data visualization.

Data collection is the process by which data about program performance are obtained from an executing program. Data are normally collected in a file, either during or after execution, although in some situations it may be presented to the user in real time. Three basic data collection techniques can be distinguished:

- **Profiles** record the amount of time spent in different parts of a program. This information, though minimal, is often invaluable for highlighting performance problems. Profiles typically are gathered automatically.
- **Counters** record either frequencies of events or cumulative times. The insertion of counters may require some programmer intervention.
- **Event traces** record each occurrence of various specified events, thus typically producing a large amount of data. Traces can be produced either automatically or with programmer intervention.

The raw data produced by profiles, counters, or traces are rarely in the form required to answer performance questions. Hence, **data transformations** are applied, often with the goal of reducing total data volume. Transformations can be used to determine mean values or other higher-order statistics or to extract profile and counter data from traces. For example, a profile

recording the time spent in each subroutine on each processor might be transformed to determine the mean time spent in each subroutine on each processor, and the standard deviation from this mean. Similarly, a trace can be processed to produce a histogram giving the distribution of message sizes. Each of the various performance tools described in subsequent sections incorporates some set of built-in transformations; more specialized transformation can also be coded by the programmer.

Parallel performance data are inherently multidimensional, consisting of execution times, communication costs, and so on, for multiple program components, on different processors, and for different problem sizes. Although data reduction techniques can be used in some situations to compress performance data to scalar values, it is often necessary to be able to explore the raw multidimensional data. As is well known in computational science and engineering, this process can benefit enormously from the use of data visualization techniques. Both conventional and more specialized display techniques can be applied to performance data. As we shall see, a wide variety of data collection, transformation, and visualization tools are available. When selecting a tool for a particular task, the following issues should be considered:

- **Accuracy**: In general, performance data obtained using sampling techniques are less accurate than data obtained by using counters or timers. In the case of timers, the accuracy of the clock must be taken into account.

- **Simplicity**: The best tools in many circumstances are those that collect data automatically, with little or no programmer intervention, and that provide convenient analysis capabilities.

- **Flexibility**: A flexible tool can be extended easily to collect additional performance data or to provide different views of the same data. Flexibility and simplicity are often opposing requirements.

- **Intrusiveness**: Unless a computer provides hardware support, performance data collection inevitably introduces some overhead. We need to be aware of this overhead and account for it when analyzing data.

- **Abstraction**: A good performance tool allows data to be examined at a level of abstraction appropriate for the programming model of the parallel program. For example, when analyzing an execution trace from a message-passing program, we probably wish to see individual messages, particularly if they can be related to send and receive statements in the source program. However, this presentation is probably not appropriate when studying a data-parallel program, even if compilation generates a message-passing program. Instead, we would like to see communication costs related to data-parallel program statements.

8.1 Performance Comparison

Here we compare our performance study of existing schemes such as Random Multicast (RM) scheme, Line Selected Multicast (LSM), and Randomized-Efficient and Distributed (RED) schemes with our Proposed System.

8.1.1 Average Energy Consumption

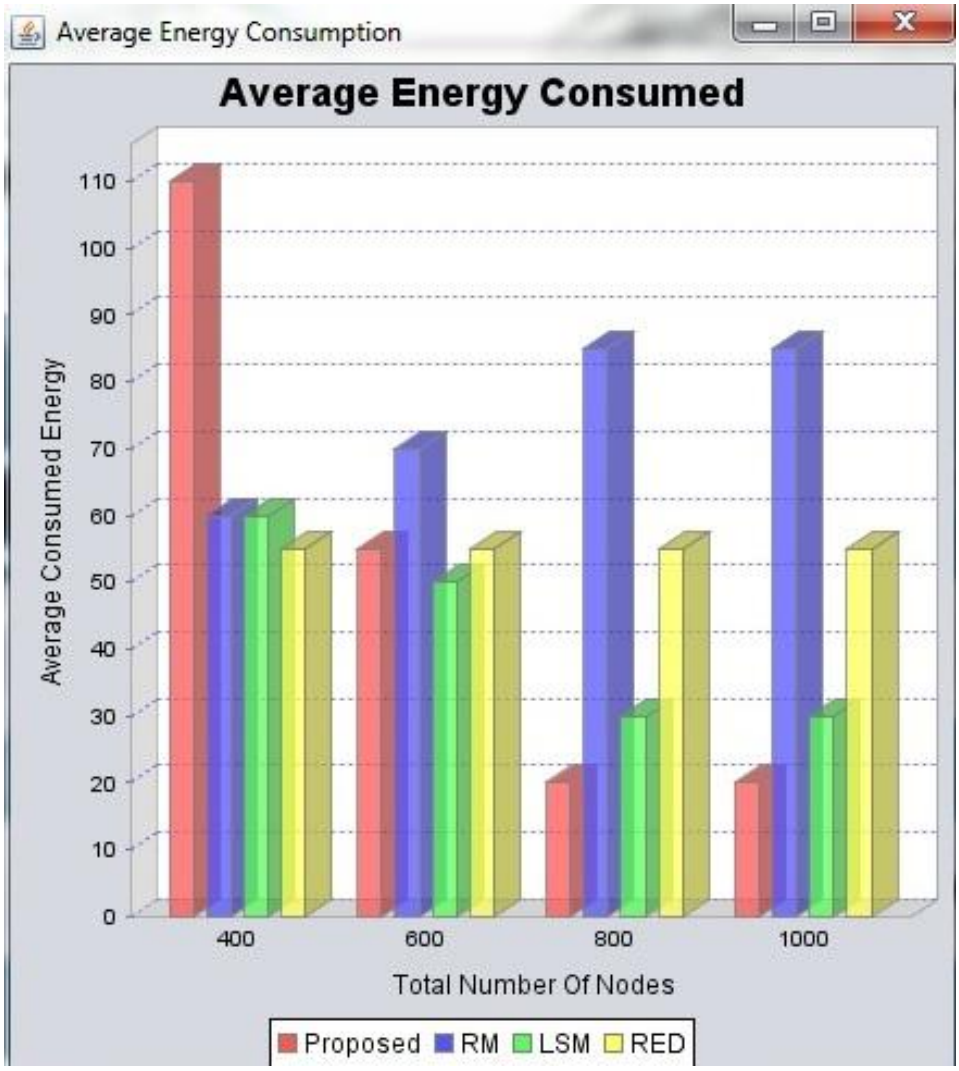

Figure 8.1 Average Energy Consumption

In Figure 8.1, it is shown that average energy consumed by all genuine nodes. We consider only sum of communication and computational costs for energy cost, because most of the energy consumed in the communication and computation processes in wireless sensor network. As shown in figure, proposed solution consumes lots of energy in small scale network with 400 nodes. This high energy consumption is caused by collection of neighboring node IDs in the proof generation step. However proposed solution shows low energy consumption in a network with 600 nodes and moreover, it shows lowest energy consumption in large scale network with 800 to 1000 nodes. On the other hand, existing schemes allow all neighboring nodes to deliver proofs with probability defined individually. From above evaluation Energy efficiency of proposed solution is definitely superior in compared to other existing schemes in large scale networks.

The proposed solution requires more information exchange between a newly deployed node and the neighboring nodes in the first stage, compared to existing schemes. The energy required for the initial communication overhead is relatively higher compared to the entire energy required throughout the completion time in small-scale networks than in large-scale networks, because a small-scale network has a shorter distance between neighboring nodes and their witness nodes, as compared to large-scale networks. However, in networks consisting of more than 600 nodes, the proposed solution consumes least energy compared to existing schemes hence we can say that proposed solution is an energy efficient schemes in large WSN.

8.1.2 Detection Ratio of Replicas

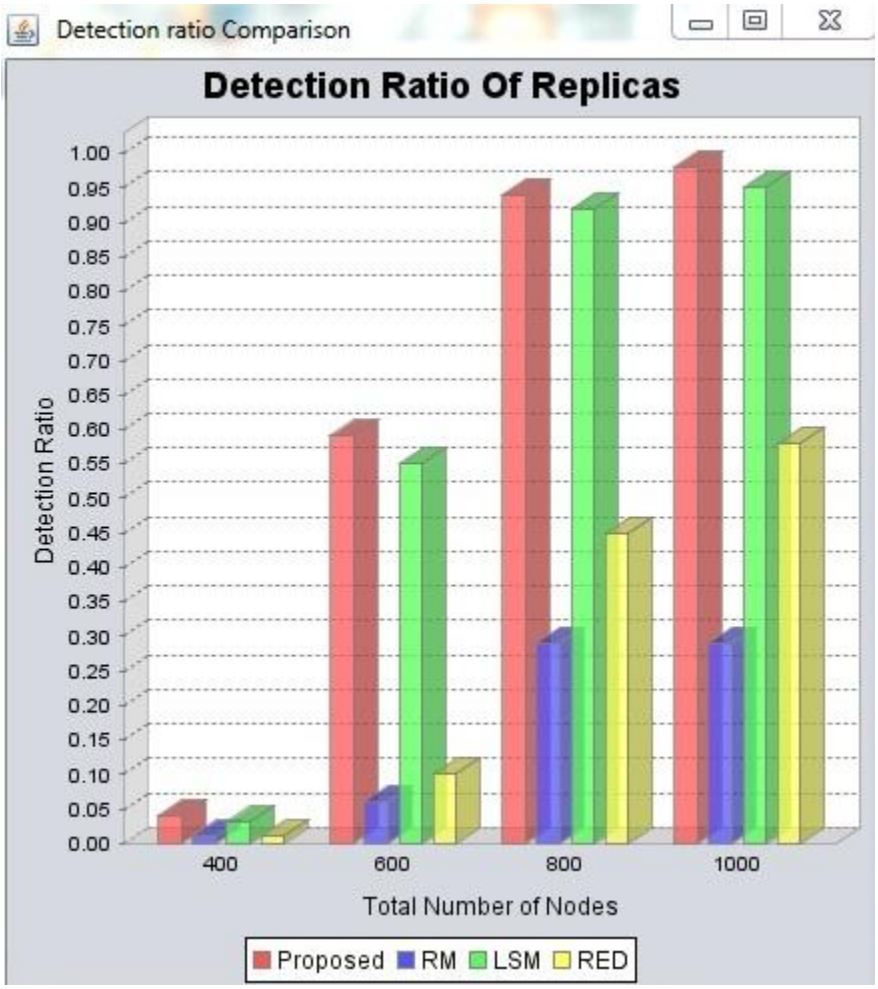

Figure 8.2 Detection Ratio of Replicas

Above figure 8.2 shows the replica detection ratio of Proposed System and Existing Schemes. In the graph, the proposed solution mostly shows similar or higher detection ratios compared to existing schemes. LSM scheme has given the nearest best detection ratio competition to proposed system. Considering that all existing schemes need expensive hardware such as GPS in order to detect replicas, the proposed solution can be sufficiently competitive for application to WSNs consisting of low-priced sensor nodes.

Chapter 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

Our approach for low cost and energy efficient solution was based on simple Bloom Filter algorithm which replaces expensive hardware's from existing schemes to detect the replica attacks. Proposed solution does not need any additional hardware. Nevertheless the proposed solution could succeed to exhibit similar or better performance than existing schemes; it detection ratio of replica is good and energy consumption is less with increase in scale of wireless sensor network. This means we can develop the low priced energy efficient detection of replicas in Wireless Sensor Network.

9.2 Future Enhancement

Although this Bloom Filter Based Low Priced WSN answers to the problems of energy and Cost effective solution compared to existing solution. Still it has challenges:

In long run for large number nodes, the traffic on network might get congested and slow down the network.

In small scale network the energy consumption considerably higher and detection ratio of replicas was lower.

Such as these challenges need to be addressed. In future there's still room for development and enhancement for Wireless Sensor Networks. There's still wide range of scopes that in future need to be addressed.

BIBLIOGRAPHY

[1] C.P. Mayer, "Security and Privacy Challenges in the Internet of Things," Electronic Comm. EASST, vol. 17, pp. 1-12, 2009.

[2] B. Parno, A. Perrig, and V. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks," Proc. IEEE Symp. Security and Privacy, pp. 49-63, 2005

[3] M. Conti, R.D. Pietro, L. Mancini, and A. Mei, "Distributed Detection of Clone Attacks in Wireless Sensor Networks," IEEE Trans. Dependable and Secure Computing, vol. 8, no. 5, pp. 685-698, Sept. 2011.

[4] H. Choi, S. Zhu, and T.F.L. Porta, "Set: Detecting Node Clones in Sensor Networks," Proc. Third Int'l Conf. Security and Privacy in Comm. Networks and the Workshops (SecureComm'07), pp. 341-350, 2007.

[5] Z. Li and G. Gong, "DHT-Based Detection of Node Clone in Wireless Sensor Networks," Proc. First Int'l Conf. Adhoc Networks, pp. 240-255, 2009.

[6] Kwantae Cho, Byung-Gil Lee, and Dong Hoon Lee," Low-Priced and Energy-Efficient Detection of Replicas for Wireless Sensor Networks," IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 11, NO. 5, SEP/OCT 2014

[7] Y. Zeng, J. Cao, S. Zhang, S. Guo, and L. Xie, "Random-Walk Based Approach to Detect Clone Attacks in Wireless Sensor Networks," IEEE J. Selected Areas Comm., vol. 28, no. 5, pp. 677-691, June 2010

[8] Michael Blaha, James Rumbaugh: Object-Oriented Modeling and Design with UML,2nd Edition, Pearson Education,2005

[9] http://en.wikipedia.org/wiki/Wireless_sensor_network

APPENDIX A

The snapshots of the outcome after rigorous testing for accuracy and efficiency at our level were taken to get the summary of all the above results of this project.

Figure A.1 Main Form

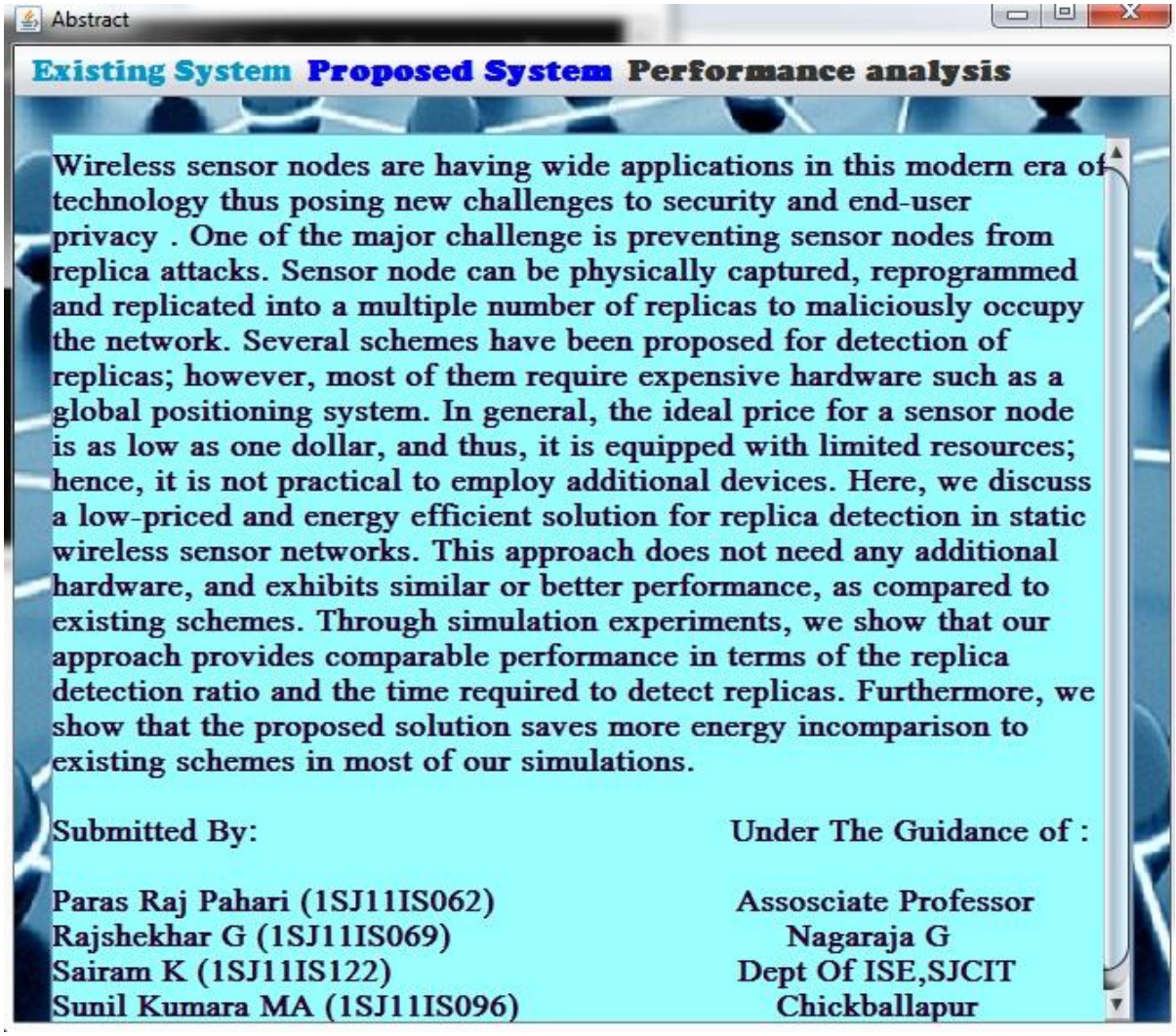

Figure A.2 Abstract

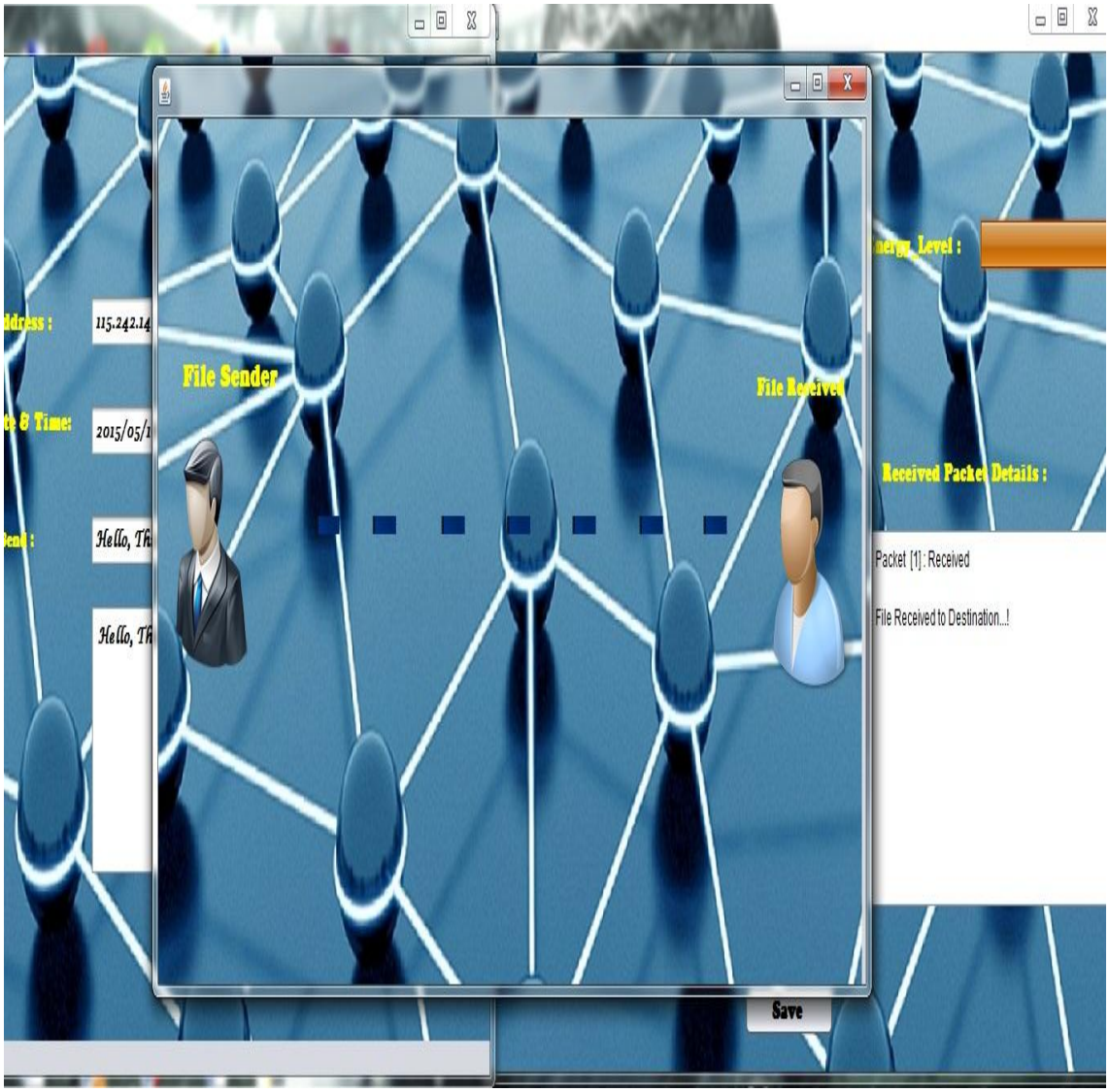

Figure A.3 Existing System Demo

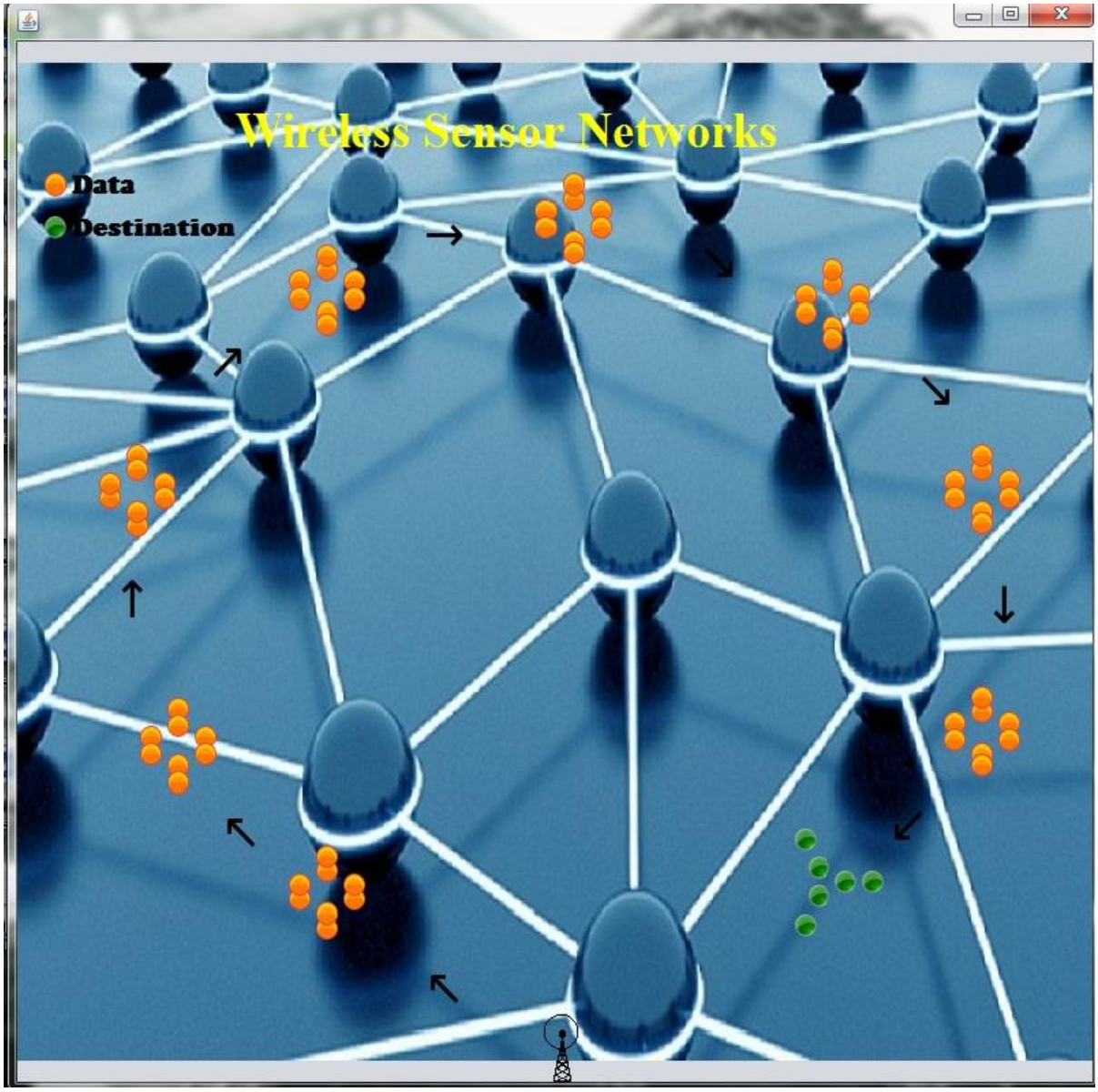

Figure A.4 Proposed System Node Deployment and Validation

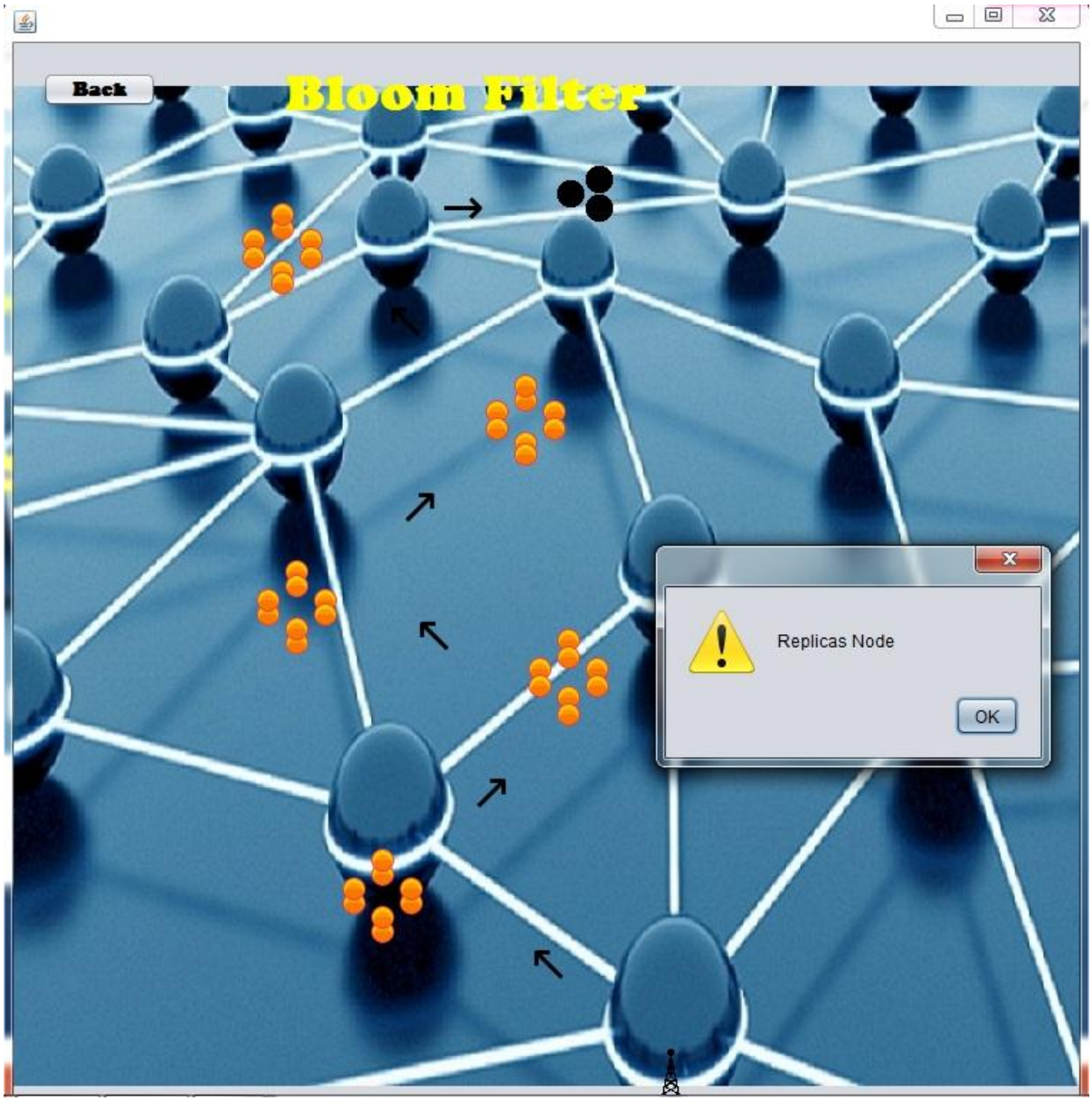

Figure A.5 Bloom Filter Detection of Replicas

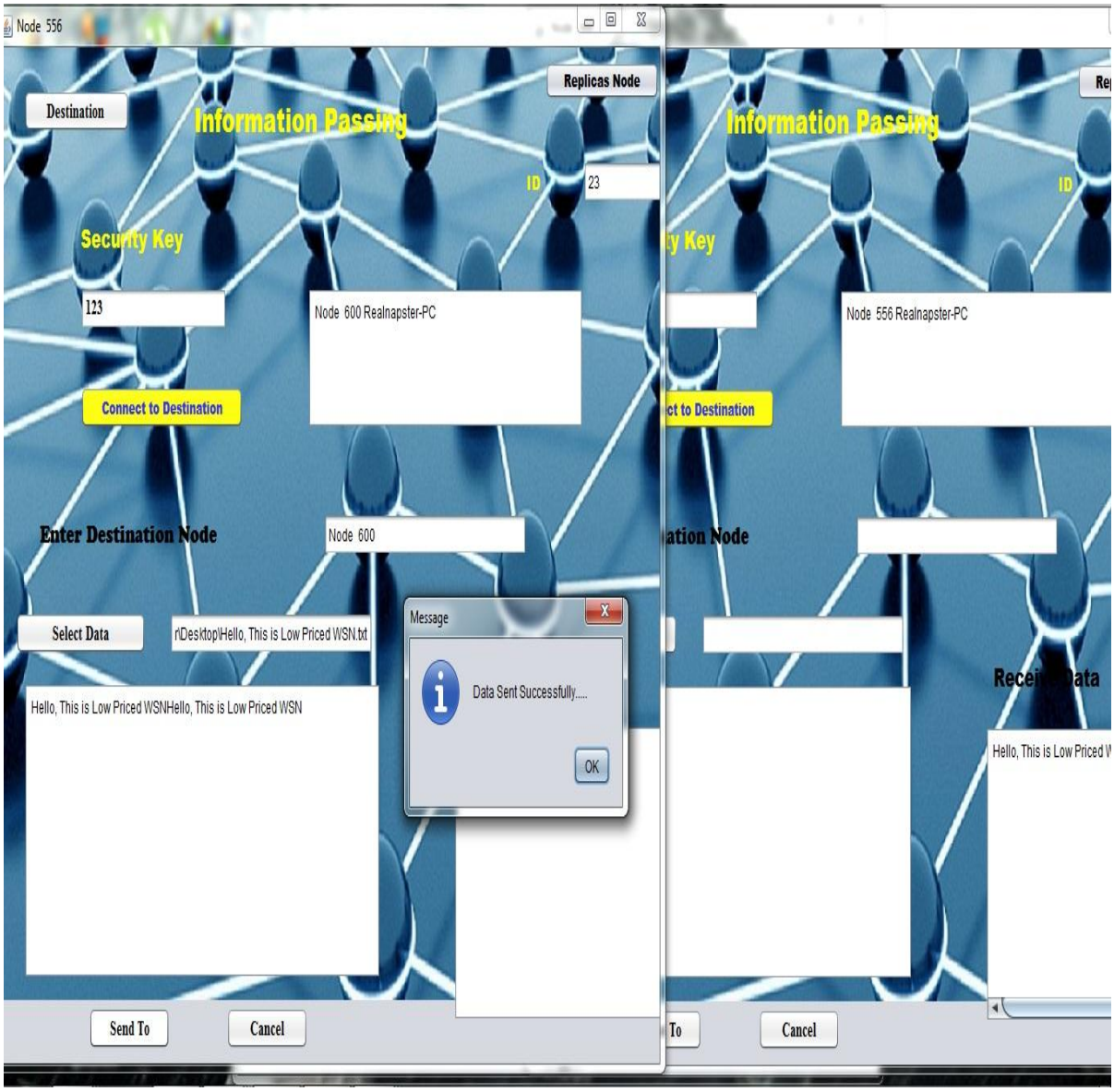

Figure A.6 Information Passing

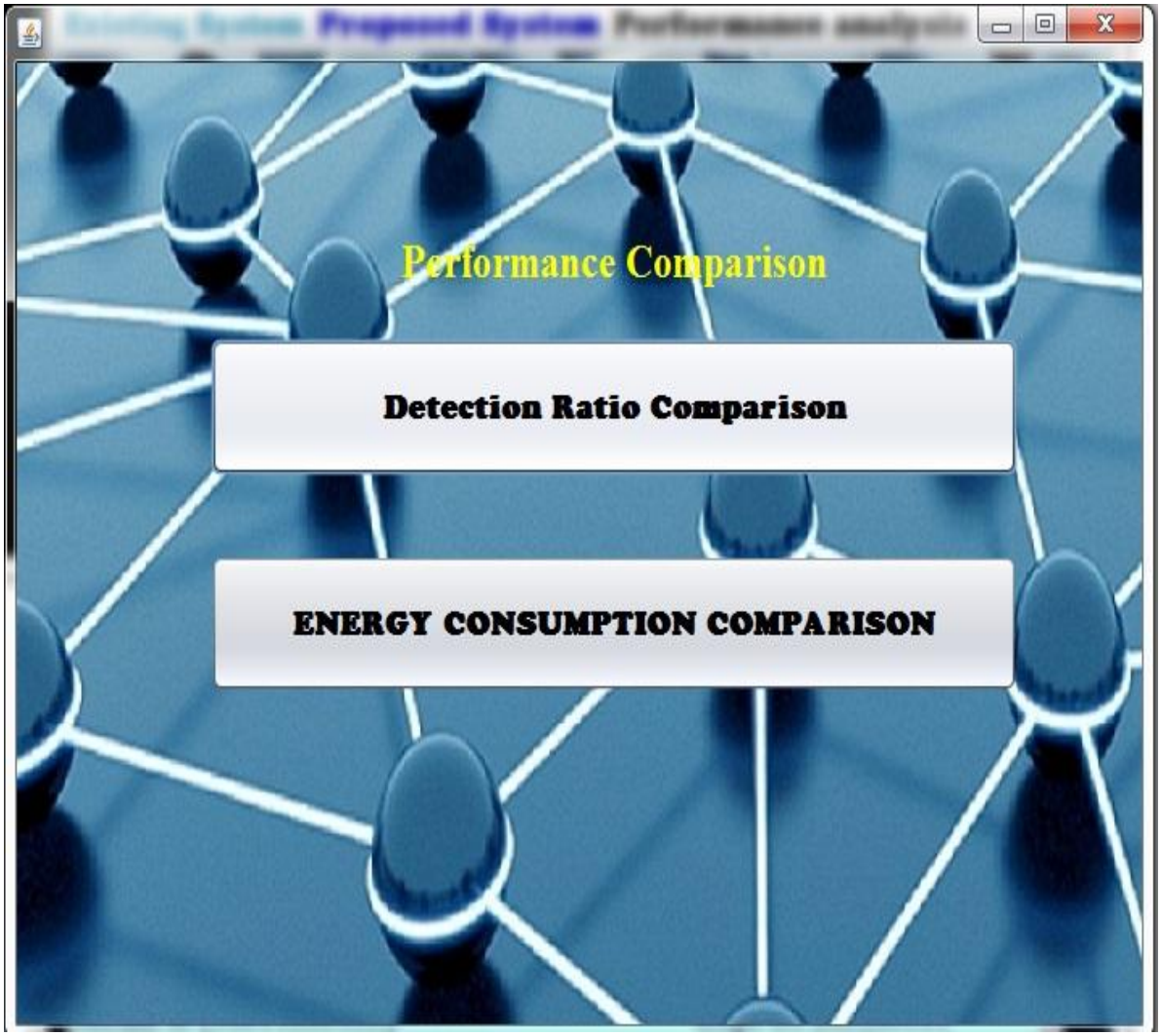

Figure A.7 Performance Comparison

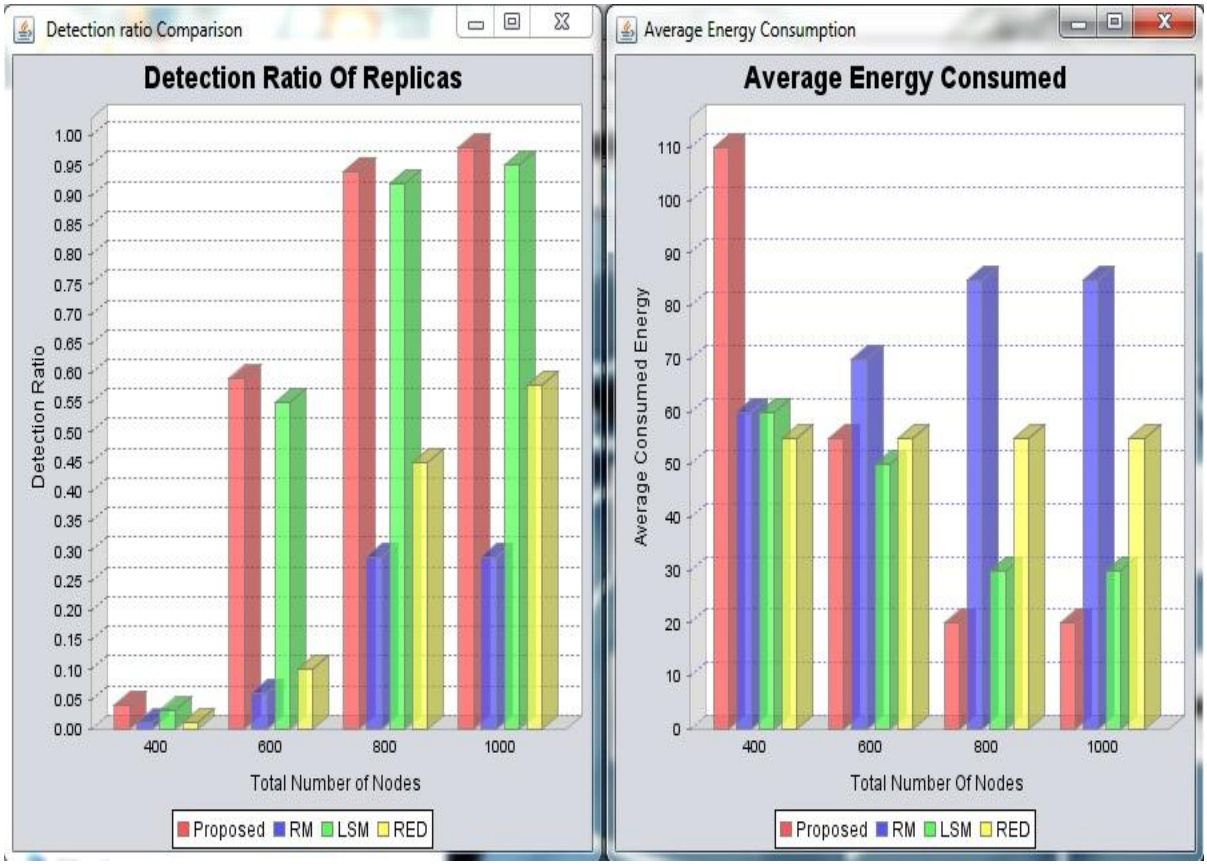

Figure A.8 Performance Chart

APPENDIX B

Certificate included here shows that we have presented our Bloom filter approach for Low Priced Energy Efficient Detection of Replicas in Wireless Sensor Network at National Level Conference held in Gitam University, Bengaluru.