INTRUSION DETECTION SYSTEM IN A WSN

Shashank Sharma (101280)

Project Guide –

Dr.Nitin

Submitted in partial fulfilment of the Degree of

Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

Table of Content

Chapter - 4

Future Enhancements & Conclusion

Certificate

This is to certify that the work titled **— "INTRUSION DETECTION SYSTEM**", submitted by Shashank Sharma, partial fulfillment for the award of degree of Bachelor of Technology in Computer Science Engineering to Jaypee University of Information Technology, Waknaghat, 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 ……………………..

Name of Supervisor Dr. Nitin

Designation Associate Professor

Date ……………………..

Acknowledgement

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

First and foremost, I am sincerely thankful to Dr.Nitin, our Project Guide, who has always encouraged me to put in my 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.

I am 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.

Shashank Sharma
Enrollment no. 101280
CSE
Signature ……………………..

Abstract

Wireless sensor networks (WSNs) are vulnerable to different types of security threats that can degrade the performance of the whole network; that might result in fatal problems like denial of service (DoS) attacks, routing attacks, Sybil attack etc. Key management protocols, authentication protocols and secure routing cannot provide security to WSNs for these types of attacks. Intrusion detection system (IDS) is a solution to this problem. It analyzes the network by collecting sufficient amount of data and detects abnormal behavior of sensor node(s). IDS based security mechanisms proposed for other network paradigms such as ad hoc networks, cannot directly be used in WSNs. Researchers have proposed various intrusion detection systems for wireless sensor networks during the last few years. We classify these approaches into three categories i.e. purely distributed, purely centralized and distributed-centralized. In this report, I present a survey of these mechanisms. These schemes are further differentiated in the way they perform intrusion detection.

Chapter-1: Wireless Sensor Networks & Intrusion Detection Systems.

1.1 Introduction

Wireless Sensor Networks (WSNs) are composed of sensor nodes and sinks. Sensor nodes have the capability of self-healing and self-organizing. They are decentralized and distributed in nature where communication takes place via multihop intermediate nodes. The main objective of a sensor node is to collect information from its surrounding environment and transmit it to the sink. WSNs have many applications and are used in scenarios such as detecting climate changed, monitoring environments and habitats, and various other surveillance and military applications. Mostly sensor nodes are used in such areas where wired networks are impossible to be deployed. WSNs are deployed in physical harsh and hostile environments where nodes are always exposed to physical security risks damages. Furthermore, self-organizing nature, low battery power supply, limited bandwidth support, distributed operations using open wireless medium, multihop traffic forwarding, and dependency on other nodes are such characteristics of sensor networks that expose it to many security attacks at all layers of the OSI model.

Many security-related solutions for WSNs have been proposed such as authentication, key exchange, and secure routing or security mechanisms for specific attacks. These security mechanisms are capable of ensuring security at some level; however they cannot eliminate most of the security attacks. An IDS is one possible solution to address a wide range of security attacks in WSNs.

An IDS is also referred to as a second line of defence, which is used for intrusion detection only; that is, IDS can detect attacks but cannot prevent or respond. Once the attack is detected, the IDSs raise an alarm to inform the controller to take action. There are two important classes of IDSs. One is rule-based IDS and the other is anomaly-based IDS . Rule-based IDS is also known as signature-based IDS which is used to detect intrusions with the help of built-in signatures. Rule-based IDS can detect well-known attacks with great accuracy, but it is unable to detect new attacks for which the signatures are not present in intrusion database. Anomaly-based IDSs detect intrusion by matching traffic patterns or resource utilizations. Although anomaly based

IDSs have the ability to detect both well-known and new attacks, they have more false positive and false negative alarms. Some IDSs operate in specific scenarios or with particular routing protocols. Watchers operate with proactive routing protocol to detect routing anomalies. It is implemented on each node, so all the nodes need some sort of cooperation to detect routing intrusions. Some intrusion detection mechanisms also operate with reactive routing protocols. These mechanisms enable the network to select a reliable path from source to destination.

This report presents a review of existing IDSs. It is organized as follows. In Section 2, we examine existing security attacks. In Section 3, we analyze and discuss some already proposed IDSs. We make comparison of existing IDSs on the basis of detection. In Section 4, we highlight some open research issues and directions, and finally in Section 5, I conclude the report.

1.2 Overview of Security in Wireless Sensor Networks

WSNs are vulnerable to many types of security attacks due to open wireless medium, multihop decentralized communication, and deployment in hostile and physically nonprotected areas. Different threat models are discussed in such as mote-class attacks and laptop-class attacks. In mote-class attacks, the attacker compromises few of the sensor nodes inside a WSN. In laptop-class attacks, the attacker has more powerful device(s) to launch more intense attack against WSNs.

Security attacks against WSNs can be classified as active and passive . Passive attacks are silent in nature and are conducted to extract important information from the network. Passive attacks do not harm the network or network resources. Active attacks are used to misdirect, temper, or drop packets. The unique characteristics such as wireless medium, contention-based medium access, multihop nature, decentralized architecture, and random deployment of such networks make them more vulnerable to security attacks at various layers.

Physical layer of WSN is responsible for radio and signals management. Radio jamming is one of the severe attacks against WSN . Another physical layer attack is battery exhaustion attack. In a WSN, battery power of sensor nodes plays an important role and determines the lifetime of the network. Keeping in view the power limitations of WSNs, it is highly desirable to design power efficient mechanisms for sustainable WSNs. Sensor nodes in sleep mode consume less energy as compared to active mode. In energy exhaustion attack, the attacker tries not to allow sensor nodes to switch to sleep mode. This can be done by sending unnecessary data or beacons to sensor nodes to keep them always busy. As WSNs are deployed in hostile environment, it is susceptible to many physical attacks such as node destruction, node replacement, node replication, battery replacement, or reprogramming of node with malicious code . However such attacks need to physically access the network.

Most WSNs use contention based carrier sense multiple access with collision avoidance mechanism (CSMA/CA). This mechanism tries to avoid collision; however it adds more complications in the form of collision, hidden-node problem, MAC selfishness, and unfairness. Possible countermeasures against such kind of attacks are small frames and rate limitations.

Network layer is responsible for appropriate route selection from source to destination. In WSN, the multihop route from source to destination is vulnerable to many active and passive attacks. Active attacks include packet-dropping attacks, packet-misdirecting attacks, rushing attack, Sybil attack, byzantine attack, routing table overflow attack, spoofed routing information, hello flood, and acknowledgement spoofing .

1.3 Intrusion Detection Systems

One of the key features of a WSN is its multihop distributed operations, which add more complexity in terms of security attack detection and prevention. In a multihop distributed environment, it is very difficult to locate attackers or malicious nodes. Many security attack detection and prevention mechanisms are designed for WSNs; however most of the existing solutions are capable of handling only a few security attacks. For example, most secure routing protocols are designed to counter few security attacks . Similarly new media access mechanisms are designed to handle hidden-node problem or selfishness. Encryption mechanisms are designed to protect data against passive attacks. Hence, one can say that there is a need to design mechanisms that are capable enough of detecting and preventing multiple security attacks in WSNs. An Intrusion Detection System (IDS) is one possible solution to it.

An intrusion is basically any sort of unlawful activity which is carried out by attackers to harm network resources or sensor nodes. An IDS is a mechanism to detect such unlawful or malicious activities. The primary functions of IDS are to monitor users' activities and network behaviour at different layers.

A single perfect defence is neither feasible nor possible in wireless networks, as there always exist some architectural weaknesses, software bugs, or design flaws which may be compromised by intruders. The best practice to secure wireless networks is to implement multilines of security mechanisms; that is why IDS is more critical in wireless networks. It is viewed as a passive defence, as it is not intended to prevent attacks; instead it alerts network administrators about possible attacks well in time to stop or reduce the impact of the attack. The accuracy of intrusion detection is generally measured in terms of false positives (false alarms) and false negatives (attacks not detected), where the IDSs attempt to minimize both these terms [3].

There are two important classes of IDSs. One is known as signature-based IDS, where the signatures of different security attacks are maintained in a database. This kind of IDS is effective against well-known security attacks. However, new attacks are difficult to be detected as their signatures would not be present in the database. The second type is anomaly-based IDS. This kind is effective to detect new attacks; however it sometimes misses to detect well-known security attacks. The reason is that anomaly-based IDSs do not maintain any database, but they continuously monitor traffic patterns or system activities.

IDS can operate in many modes, for example, stand-alone operation and cooperative cluster based operation. A standalone IDS operates on every node to detect unwanted activities. Cooperative cluster based IDS are mostly distributed in nature in which every node monitors its neighbours and surrounding nodes activities and operation; in case of any malicious activity detection, the cluster head is informed.

Broadly speaking, IDS has three main components as shown in Figure.(i)Monitoring component is used for local events monitoring as well as neighbours monitoring. This component mostly monitors traffic patterns, internal events, and resource utilization.(ii)Analysis and detection module is the main component which is based on modeling algorithm. Network operations, behavior, and activities are analyzed, and decisions are made to declare them as malicious or not.(iii)Alarm component is a response generating component, which generates an alarm in case of detection of an intrusion.It should be noted that IDSs are passive in nature and can only detect intrusion. They cannot take any preventive action; they only generate an alarm. It is then the administrator’s job to take preventive measures against the attack. Researchers in WSNs are working on two broad categories of IDSs, that is, signature-based and anomaly-based IDSs.

Monitoring component
Analysis and detection
Alarm
Figure 1: Components of IDS.

Chapter2: Literature Survey.

2.1 Types Of Intrusion Detection Systems.

2.1.1 Signature-Based Intrusion Detection Systems

Signature-based IDS, also known as rule-based IDS, has predefined rules of different security attacks. When the network's behaviour shows any deviation from the predefined rules, it is classified as an attack. Signature-based IDSs are well suited for known intrusions; however they cannot detect new security attacks or those attacks having no predefined rules. In this section, we present existing signature-based IDSs for WSNs.

In , a rule-based IDS for WSNs is presented. It is host based in which every node has IDS. The architecture of the proposed IDS has many modules such as packet monitoring, cooperative engine, detection engine, and response unit. The IDS is basically designed for routing attacks and is capable of detecting packet-dropping attacks. An IDS for detection of sink-hole attack is presented in. The proposed IDS is hosted on each sensor node and requires TinyOS with the combination of MintRoute routing protocol. It is an advanced version of with narrow approach; that is, the former can detect many packet-dropping and misdirecting attacks while the latter is only designed for detection of sink-hole attacks. In both approaches, every node monitors and cooperates with neighbours. Intrusion Detection Architecture (IDA) is presented in. IDA is distributed and hierarchical in nature which can operate by cooperation of sensor nodes, cluster head, and central system. IDA generates either passive or active response on the basis of attack nature. However, this work does not present results on the detection rate and false positive and false negative ratios.

In, Intrusion Detection Program (IDP) is proposed, which is capable to detect known attacks. IDP is based on genetic programming (GP) technique and is effective against a variety of attacks such as denial of service (DoS) and unauthorized access. IDA uses three variants of GP such as linear-genetic programming (LGP), multiexpression programming (MEP), and gene-expression programming (GEP). GEP and MEP detection and classification accuracy are greater than 95%. A distributed IDS (DIDS) using soft computing techniques is presented in. It uses few fuzzy rule-based classifiers to identify intrusions. The authors claim that fuzzy classifier provides 100% accuracy for all kinds of intrusions.

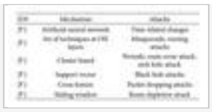
A decentralized rule-based IDS is proposed in. This mechanism has three main phases, namely, data acquisition, rule application, and intrusion detection. The proposed mechanism is capable of detecting many routing attacks such as worm-hole, black-hole, selective-forwarding, and delay attacks. The authors also claim that the proposed solution is capable of detecting jamming attack as well; however they did not explain how jamming attacks are detected as it is a physical layer attack. Spontaneous watchdog IDS and its basic architecture is given in. This architecture consists of local and global agents; however it is not implemented yet. An ant-colony-based IDS in conjunction with machine learning is another rule-based IDS. The proposed IDS perceives behaviour and acts using self-organizing principle initiated with probability values. Different signature-based IDSs are given in Table 1.

Table 1: Signature based IDSs.

2.1.2 Anomaly-Based Intrusion Detection Systems

Anomaly-based IDS monitors network activities and classifies them as either normal or malicious using heuristic approach. Most of anomaly-based IDSs identify intrusions using threshold values; that is, any activity below a threshold is normal, while any condition above a threshold is classified as an intrusion. The main advantage of anomaly-based IDS is its capability to detect new and unknown attacks; however sometimes it fails to detect even well-known security attacks. Many anomaly-based IDSs have been proposed so far. An unsupervised neural network based IDS is capable of learning and detecting unknown attacks. This intelligent system learns the time-related changes using Markov model. When any intrusion occurs, a mobile agent moves to the malicious region of the WSN to investigate. The proposed mechanism can detect time-related changes and events.

A set of intrusion detection techniques at different layers is presented. These techniques are independent of each other. At physical layer, RSSI values are used to detect masquerade, while at network layer, a specialized table driven routing protocol is used to detect routing and authentication attacks. A cluster based IDS for routing attack is proposed. This mechanism is capable of building a normal traffic model, which is used to differentiate between normal and abnormal traffic. The normal traffic model consists of number of packets received and sent, number of route requests received and sent, and so forth. The IDS can detect many attacks such as periodic route error attack and sink-hole attack. A support vector machine based IDS is used to detect routing attacks such as black hole. It is basically cooperation based detection in which nodes communicate and share information about security attacks. A cross feature based anomaly detection mechanism is proposed in . This mechanism monitors and learns normal traffic patterns in order to detect any intrusion in case of deviation. The IDS is capable of detecting packet-dropping and misdirecting attacks. A sliding window based IDS using threshold value is efficient in the detection of few security attacks such as route depletion attacks. Table 2 presents a summary of a number of anomaly-based IDSs.



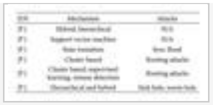
IDS	Mechanism	Attacks
[illegible]	Artificial neural network	Time related changes
[illegible]	Set of techniques at OSI layers	Masquerade, routing attacks
[illegible]	Cluster based	Periodic route error attack, sink hole attack
[illegible]	Support vector	Black hole attacks
[illegible]	Cross feature	Packet dropping attacks
[illegible]	Sliding window	Route depletion attack

Table 2: Anomaly based IDSs.

2.1.3 Hybrid Intrusion Detection Systems

Hybrid IDSs are a combination of both anomaly-based and signature-based approaches. Hybrid mechanisms usually contain two detection modules; that is, one module is responsible of detecting well-known attacks using signatures, while the other is responsible for detecting and learning normal and malicious patterns or monitor network behavior deviation from normal profile. Hybrid IDSs are more accurate in terms of attack detection with less number of false positives. However, such mechanisms consume more energy and more resources. Hybrid IDSs are generally not recommended for a resource constraint networks such as a WSN; however they are still an active research area. A hybrid intrusion detection model is presented in. In this model, sensor nodes are divided into hexagonal regions like cellular networks. Each region is monitored by a cluster node, while cluster nodes are monitored by regional nodes. The base station has the responsibility to monitor all regional nodes. It is hierarchical in nature forming a tree-like structure. Attack signatures are stored in base station and propagated toward the leaf node for attack detection. Similarly the mechanism has predefined specifications of normal and abnormal behaviour. Anomaly detection is done by measuring deviation from defined specifications. The authors did not mention detection rate or false-alarm ratio of their proposed mechanism. Furthermore, it is not clear which security attacks are detected using this mechanism.

Another hybrid IDS using support vector machine (SVM) and misuse detection is proposed in. A distributed learning algorithm is used to train SVM to distinguish normal and malicious patterns. This intrusion detection mechanism is designed to operate in cluster based WSNs, where all nodes monitor their neighbours. The authors claim high detection rate with fewer false positives; however attack types are not described. An IDS that uses state transition analysis and stream flow to detect sync-flood attack against WSNs is presented in. This mechanism monitors three-way handshake of TCP to identify attack pattern; however it is not yet implemented and tested. A cluster based hybrid IDS is given in, where the cluster head is responsible for detecting intrusions. The key idea behind this mechanism is to reduce energy consumption. A further enhanced IDS is proposed in. The enhanced IDS has three modules, that is, anomaly-based detection, signature-based detection, and decision making. A supervised back propagation network is used to learn and identify normal and malicious packets. Another hierarchical hybrid IDS for detection of routing attacks is presented in. It has high accuracy in terms of detection of network layer security attacks such as sink hole and worm hole. Table 3 presents a summary of a few hybrid IDSs.



IDS	Mechanism	Attacks
[illegible]	Hybrid, hierarchical	N/A
[illegible]	Support vector machine	N/A
[illegible]	State transition	Sync flood
[illegible]	Cluster based	Routing attacks
[illegible]	Cluster based, supervised learning, misuse detection	Routing attacks
[illegible]	Hierarchical and hybrid	Sink hole, worm hole

Table 3: Hybrid IDSs.

2.1.4 Cross Layer Intrusion Detection Systems

Cross layer design is a relatively new security technique in which different parameters across OSI layers are exchanged for optimal solutions. Traditional IDS operates at a single layer of the OSI model and hence can monitor and detect intrusions at that particular layer. For example, network layer Intrusion Detection System can detect only routing attacks but cannot respond to MAC, physical, or transport layer anomalies. Cross layer IDSs have the capability to monitor and detect intrusions at multiple layers by communicating and exchanging parameters amongst different layers using cross layer interface. As we know, WSNs have many constraints in terms of computations, memory, and energy. Although cross layer IDS can detect many intrusions at different layers, this technique consumes more energy and computational resources by monitoring, analyzing, and exchanging multilayer parameters.

Cross layer intrusion detection agent (CLIDA) for WSNs is proposed in [47]. CLIDA ensures cross layer information exchange amongst physical, MAC, and network layer. Cross layer data module collects and represents data to all layers. CLIDA is capable of detecting multi-layer security attacks. This architecture has good detection rate; however energy and computational comparison is not given, which could be more interesting. Another cross layer security mechanism for WSN is proposed in, in which the authors have the observations that such mechanism would exhaust the limited resources of sensor nodes. In, a real-time cross layer security mechanisms for large scale flood detection and attack trace-back mechanism is presented. It uses different parameters from MAC and network layers to detect multi-layer flooding attacks. It maintains different profiles for low, medium, and high intensity attacks.

2.2 Comparison and Discussion

Wireless Sensor Networks are distributed in nature using the multihop communication model. These networks are usually deployed in such areas where direct human interaction is either impossible or very difficult. Furthermore, WSNs have limitations in terms of computation, bandwidth, memory, and energy. These limitations are considered while designing any proposal for such networks. Due to the hostile environments of WSNs, security is one of their most important aspects. IDSs are widely used for securing WSNs. IDS has the ability to detect an intrusion and raise an alarm for appropriate action. Due to the energy and computational power limitations, designing appropriate IDS for WSN is a challenging task.

Anomaly-based IDSs are suitable for small-sized WSNs where few nodes communicate with the base station. In small sized WSNs, the traffic pattern is mostly the same, so unusual traffic pattern or changing behaviour can be treated as an intrusion. However such IDS may generate more false alarms and may not be able to detect well-known intrusions. Anomaly-based IDSs are usually lightweight in nature and mostly use statistical, probabilistic, traffic analysis or intelligent techniques.

Signature-based IDSs are suitable for relatively large-sized WSNs, where more security threats and attacks can compromise network operations. Signature-based IDS needs more resources and computations as compared to anomaly-based IDS. One of the important and complex activities is the compilation and insertion of new attack signatures in the databases. Such IDSs mostly use data mining or pattern matching techniques.

Hybrid IDSs are suitable for large and sustainable WSNs. These IDSs have both anomaly-based and signature-based modules, so they require more resources and computations. To reduce the usage of limited resources, such mechanisms are mostly used in cluster based or hierarchical WSNs, in which some parts of the network are used to execute anomaly detection while other parts are accompanied with signature-based detection.

Cross layer IDSs are usually not recommended for a resource constraint networks such as WSNs, as it consumes more resources by exchanging parameters across the protocol suits for attack detection. Table 4 gives the comparison and characteristics of different IDSs.

2.3 Classification of attacks

There are four aspects of a wireless sensor network that security must protect:
1) Confidentiality
2) Data Integrity
3) Service Availability
4) Energy

The first three are addressed by security systems in wired networks and non-energy-constrained wireless networks, but the fourth is unique to the sensor network application.I will thus classify the attacks that can be launched by which of these aspects they attack. In this section I will brieflydescribe the kinds of attacks in each of these categories and the ways in which the nature of the wireless network causes trouble.

2.3.1 Stealing Data (Confidentiality)

When we think of electronic security, this is the first kind of attack that comes to mind. We want to be able to send messages without enemies being able to figure out the contents. Because of the wireless nature of the WSN, it is easy for an attacker to listen in on all the messages sent in the network, so to maintain confidentiality, the network must encrypt all the messages. One of the biggest ideas in encryption today is public-key encryption. This is very powerful because it allows one toreceive encrypted messages without even sharing a secret key with the sender. But this asymmetry comes at a cost. RSA public key encryption involves exponentiating the message, which can be quite computationally expensive, and is not really feasible in WSNs, where the nodes typically are not capable of doing such computations in a time- and energy-efficient way. But symmetric key encryption mechanisms pose the problem that if any node is apprehended by the attacker, he can look in its memory to find the key and effectively be able to masquerade as any other node (compromising the data integrity) and listen in on any other conversation.2

2.3.2 Altering/Generating False Data (Data Integrity)

Because sensor networks are used to monitor some environment, data integrity is even more important than confidentiality. This is because applications may include tracking objects that physically move through the environment and since attackers can typically see the physical environment, the only thing they could gain from listening in on the data is a sense of where the sensors are located. On the other hand, if they are able to alter to make the data collected by the WSN incomplete or incorrect, the deployer of the WSN will not know what is really going on in the environment he is trying to monitor. In other networks, the same asymmetric key system that is used for encryption can be used for digital signatures, but this requires a lot of additional overhead. The signature may consist of a lot of additional bytes of data added on to a transmission (which takes additional energy), and verifying the signature can be very computationally expensive. Clearly, different techniques are needed for WSNs.

2.3.3 Attacks on Service Availability

This class of attacks is not at all concerned with the actual data that is begin sent. Rather, the goal is to make the network not function properly. This can be done by sending bogus routing information (for example advertising a route that does not exist). It can also be done by flooding the network with packets (denial of service attack), or even jamming the frequency at the physical layer. Another interesting type of attack is homing. In a homing attack, the attacker looks at network traffic to deduce the geographic location of critical nodes, such as cluster heads or neighbors of the base station. The attacker can then physically disable these nodes. This leads to another type of attack: the “black hole attack”. In a “black hole” attack, the attacker compromises all the neighbors of the base station, making it effectively a black hole. A final kind of attack on service availability is a de-synchronization attack, where the attacker tries to disrupt a transport-layer connection, by forging packets from either side.

2.3.4 Denial of Sleep Attacks (Energy)

The constrained energy of WSNs adds a new element that can greatly complicate security issues. Because there is a limited amount of energy available and no way to replenish it, it is not sufficient to make sure that bad data is not used. We need to make sure that we do not waste energy listening to or re-transmitting bad packets. This introduces a whole new set of possible attacks. These include constantly sending RTS packets to stop nodes from going to a low power “sleep” state, sending falsified or repeated packets so that nodes waste energy re-transmitting them, or draining the power of a node by forcing it to do excessive computations.

2.4 SOUTIONs

2.4.1SPINS

Many of the confidentiality and data integrity issues can be handled by SPINS. SPINS is a collection of protocols for sensor networks. The key security components are SNEP and μTESLA. SNEP provides a lot of key security features. It provides confidentiality and data integrity for pairwise connections as well as weak freshness. Freshness means that old packets cannot be repeated by an adversary to create confusion and waste energy. Weak freshness means that there are no delay guarantees, but packets cannot be repeated or re-ordered. In SNEP, each pair of nodes shares a pair-wise key κ. This key is used in DES in cipher block chaining (CBC) mode. The cipher block chain provides semantic security (meaning that the same message string will not always encrypt to the same cipher string) through the use of an initialization vector (IV). Rather than sending this IV in the clear along with a message, the IV comes from a shared counter. This alleviates the need to send unnecessary bits. The counter also provides data freshness, because since it is incremented with each transmission, a previous transmission cannot be repeated, and the correct ordering is evident. In addition to being encrypted, messages are also authenticated in SNEP through the use of a message authentication code (MAC), which is a function that takes two arguments and maps them to an 8 byte number. The arguments used are the pairwise key and the concatenation of the count and the encrypted message. Because all of these are available to the receiver, it can calculate the function to verify the signature. SNEP is very good because it provides all of this security for an overhead of merely 8 bytes per message. While SNEP provides pairwise authentication, it cannot provide authentication for broadcasts, because if the key is shared among several nodes, compromising any of them would allow the attacker to masquerade as any node in the group. Broadcastauthentication is therefore handles using μTESLA. Because of the nature of broadcast, asymmetry of information is very important. The sender (typically the base station orcluster head) must be able to generate a signature that the other nodes can verify, but not create on their own. μTESLA accomplishes this using a one-way function F(·). It generates a sequence of keys to be used in the MAC in the following way:

$Kn = F(Kn+1)$. (1)

It is important that the keys are used in reverse order, because this assures that a key cannot be used to predict future keys (F(·) is one-way). In addition, the keys are disclosed periodically, rather than with each packet, because this saves the unnecessary overhead. An example of μTESLA can be seen. In figure we can see that if a node can buffer P1 through P5, it only needs to hear K4 to be able to authenticate all of the previous packets. Having each key represent a given time frame rather than a given packet assures that even if a packet is missed, all received packets can be authenticated.

Chapter-3: Testbed, Experimental Setup & Simulation Outputs.

3.1Proposed Model For a WSN.

Our objective is to detect the sleep deprivation attack in sensor network. In this section, a lightweight model, INSOMNIA MITIGATING INTRUSION DETECTION SYSTEM (IMIDS) is proposed for heterogeneous wireless sensor network (HWSNET) to detect insomnia of stationary sensor nodes. It uses cluster based mechanism in an energy efficient manner to build a five layer hierarchical network to enhance network scalability, flexibility and lifetime. The low energy constraints of WSN necessitate the use of a hierarchical model for IDS. We divide sensor network into clusters which are again partitioned into sectors.

It will minimize the energy consumption by avoiding all the nodes needing to send data to a distant sink node. It uses anomaly detection technique in such a way so that phantom intrusion detection can be avoided logically.

3.1.1 Assumptions

O A sensor can be in any one of the following states:

NEW→MEMBER→ SUSPECTED→MALICIOUS→ISOLATED

↓↕↓

GENUINE → DEAD

O Each sensor node has a unique id in the network.

O Each member node has authentic wake-up token.

O A protocol is used to assign a secure wakeup and sleep schedule for the sensor nodes.

O Sink node is honest gateway to another network.

O The threshold values are pre-calculated and set for the entire network.

O If any of cluster coordinator, forwarding sector head, sector monitor or sector coordinator is found to be compromised, reconfiguration procedure takes place dynamically.

O Sensor nodes excluding leaf nodes and forwarding sector heads in the system participate in intrusion detection process.

O Generally, sector coordinator is responsible for anomaly detection and sector monitor is responsible for detection of intrusion.

O Initially, probability of sleeping schedule and wake-up schedule are same (p=0.5) for any normal node.

O Initially, trust value of each node is represented by a nibble t3 t2 t1 t0 containing all 1's, belief is set to 1.

O SM may be more than one within a sector.

O SN selects CC and CC selects SC, SM, FSH.

O Anomaly can be detected on the basis of energy consumption rate, allotted wakeup schedule, authentic wakeup token, number of packets received within a time interval. Reputation of sensor node needs to be considered during intrusion detection.

3.1.2 Data Definition

▪ Definition 1: Leaf Node LN– A node N is defined to be a Leaf Node if ChildN{ }= {∅} AND ParentN { } ≠ {∅}. Its detection power(DP) ←0.

▪ Definition 2: Setor Coordinator SC – A node N is defined to be a Sector Coordinator if Rem_engN = MAX_ENG {FN[ ]}, where FN[] → follower nodes.

▪ Definition 3: Setor Monitor SM - A node N is defined to be a Sector Monitor if DPN =MAX_DETECT {N [ ]}, where N∉ {CCk, SN} AND DPN→ power required by a node for intrusion detection.

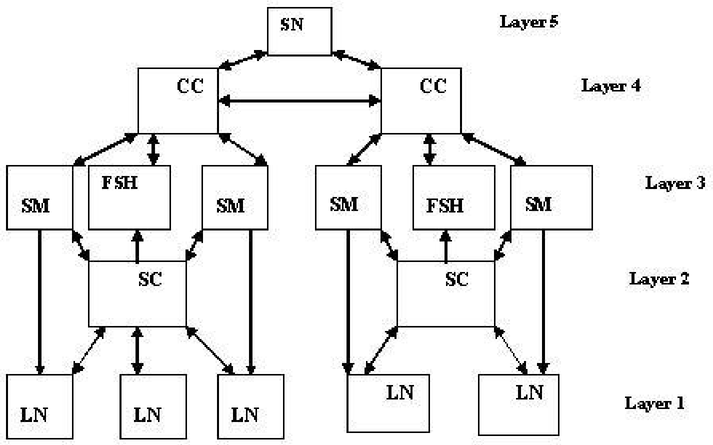
▪ Definition 4: Forwarding Sector Head FSH - A node N is defined to be a Forwarding Sector Head, where hop_distanceN {} = min{hop_ distanceN from CCk},where N ∉CCk. Its detection power (DP) ←0.

▪ Definition 5: Cluster Coordinator CC - A node N is defined to be a Cluster Coordinator, if Rem_engN =MAX_ENG{N[ ]}AND CAPACITYN=MAX(CAPACITYN),where N∉SN AND CAPACITYN = (DEGREEN/ INITIAL_ENGN)*Rem_EngN, DEGREEN→number of nodes within its radio range.

▪ Definition 6: Sink Node SN - A node N is defined to be a Sink Node if ChildN { } ≠ {∅} AND ParentN { } = {∅}.

3.1.3 System Model

Figure 1 describes the main building block of the system model. Here SN–> SINK NODE; CC–>CLUSTER COORDINATOR; SM–>SECTOR MONITOR; FSH–>FORWARDING SECTOR HEAD; SC–>SECTOR COORDINATOR; LN–> LEAF NODE;



Layered model

3.1.4 Description of Each Layer

The five layers of sensor network are described below-

Ø Layer 1: In this lowest layer leaf nodes sense environmental data and send it to its immediate next higher layer i.e. layer 2. Layer 1 has no anomaly detection capacity.

Ø Layer 2: This layer includes sector coordinator (SC) of each sector that collects data from layer 1 and checks for anomaly. Sector coordinator maintains membership list [] of all leaf nodes within a sector, normal profile [] (tuple space that consists of sensor node's attribute) and knowledge base [] (system parameters, application requirements). Suspected nodes are penalized and legitimate nodes are rewarded by SC. Reputation list [] is updated. Suspected node details are inserted in suspected list [] before forwarding to SM and valid packets are forwarded to FSH of layer 3.

Ø Layer 3: This layer includes sector monitor(SM) and forwarding sector head (FSH). Sector monitor maintains suspected list [], normal profile [], knowledge base [], reputation list []. SM can detect intruders, compromised nodes and isolate them by inserting the details into quarantine list [] and forwards the information to cluster coordinator (CC).FSH (nearest neighbor of cluster coordinator) acts as router that inserts valid packet details to forwarding table [] and forwards valid packet of legitimate nodes to CC of layer 4.

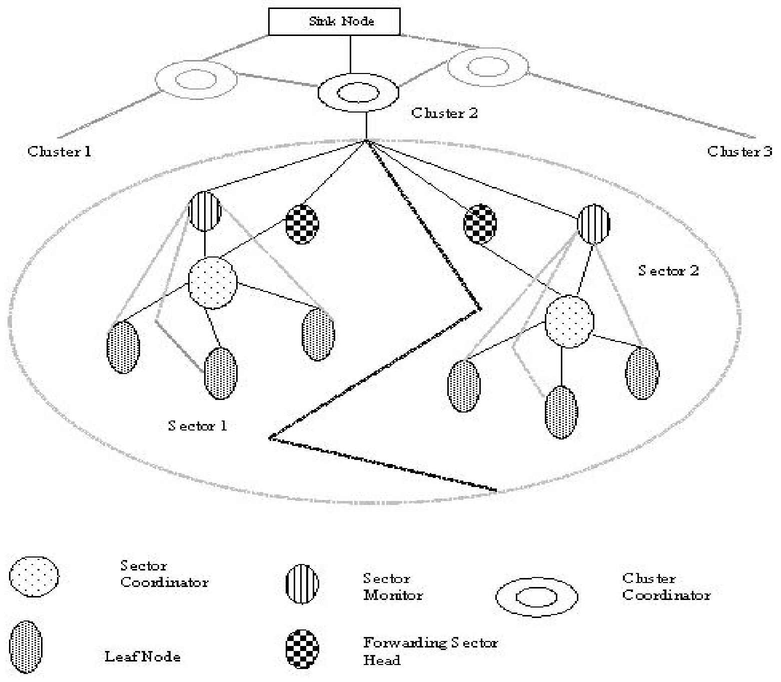
Ø Layer 4: This layer constitutes the cluster coordinator (CC) which controls SM and FSH of each Sector within a cluster. It inserts valid packets details to valid list [] and forwards data to the sink node. Cluster coordinators (CC) can cooperate with each other to form global IDS.CC contains backup copy of its own cluster.

Ø Layer 5: The topmost layer is the sink node that collects data from lower layer and it acts as a gateway between sensor network and other networks or acts as access point. SN contains backup copies of all clusters.

3.2 IMIDS: Insomnia Mitigating Intrusion Detection System

The entire heterogeneous sensor field is divided into overlapping or disjoint clusters like Ck, for k ∈ {1,..,r}, r being the number of clusters in the sensor network. Each cluster consists of its member nodes including a cluster coordinator (CC). Let mem1, mem2, ...,memn be the members of a cluster Ck, which are unaware of their locations and n is the number of members within a cluster excluding CC. Clusters are partitioned into non-overlapping sectors like Sj, for j∈{1,…,m},where m is the number of sectors within a cluster, where r<<m . We assume three types of sensor nodes in this five layered model: (i) leader nodes or LDN (in layer 3 and 4) (ii) follower nodes or FN (in layer 1 and 2) and (iii) sink node or SN (in layer 5). Leader nodes can be equipped with EXIDS (extended IDS), but only the node designated as sector monitor can activate it. Cluster coordinator (CC) and sink nodes (SN) are also using EXIDS for detecting intrusion during its requirement. SIDS (simple IDS) can be loaded in all follower nodes, but can be activated only at sector coordinator of layer 2 for detecting anomaly. Sector coordinator collects sensing data within allotted TDMA time slot of each leaf node in a sector. Sector coordinator (SC) monitors the sensor nodes for detecting anomaly by SIDS. Suspected nodes are

penalized and legitimate nodes are rewarded. Forwarding sector head (FSH) forwards valid packets to CC. Sector monitor (SM) decides whether a suspected node is malicious or not. EXIDS has the responsibility to declare the suspected node as malicious and to drop fake or corrupted packets. To avoid phantom intrusion detection logically, suspected nodes get chance to increase their reputation by SM, if it is not decided as malicious. Intruder/Malicious nodes are isolated in quarantine list; so that no intrusion occurs through these nodes. If HWSNET is considered as a graph G (V, E), any edge E between two nodes ni and njis valid if and only if distance between two nodes Di,j <= Rtr (transmission range).Detection power of LN and FSH are 0%, SC,CC,SN are 50% and SM is 80%.When detection power reaches to minimum threshold, detection capacity is automatically disabled. Reconfiguration procedure takes place dynamically if any node found to be suspected i.e. energy consumption rate greater than normal consumption rate. Each of leader and follower nodes must be included within a cluster. If any node is under more than one CC then RSSI value need to be checked. If there is a tie, it is broken randomly. Anomaly is detected by SC. But there is a possibility of false positive or false negative. If any genuine node is suspected by SC (false positive), SM can detect it and takes final decision. If any compromised node is treated as genuine and forwarded to FSH (false negative), CC can detect it.



Cluster Based Heterogeneous Sensor Network

3.2.1 Procedural Steps of IMIDS

1. Initialization Phase:

Sensor nodes are deployed in the sensor field during this phase. A unique identification number consisting of the geographical position vectors is assigned to each new node. The sink node searches for its neighbors to acquire energy details of all nodes after broadcasting advertised message.

2. Cluster Coordinator Selection and Cluster Formation Phase:

Cluster coordinator is selected among all leader nodes and its coverage area is considered as cluster. The Cluster-head details are broadcasted to all its neighbors. The neighbor nodes collect advertised messages during a given time interval and send a join message to nearest cluster coordinator for all nodes within the range of any specific cluster coordinator. Intersection of two cluster domains may or may not be NULL.

3. Sector Coordinator Selection and Sector Formation Phase:

Sector coordinator is selected among all follower nodes and its detailed information along with node-id is broadcasted to all of its neighbors. Its coverage area is considered as sector. Intersection of domains of two sectors must be NULL. Sector monitors and forwarding sector heads are selected for each sector.

4. IDS Activation Phase:

Activate IDS preinstalled in cluster coordinators, sector monitors and sector coordinators.

5. Reconfiguration Phase:

When cluster coordinator or sector coordinator's behavior deviates from normal, reconfiguration procedure takes place.

6. Data Transfer and Intrusion Detection Phase:

After sector coordinator selection is done each follower node (leaf node) sends data to the sector coordinator that transfers genuine packet to its cluster coordinator through forwarding sector head. Cluster coordinator collects valid data from all sectors within its coverage area and then forwards aggregated packets to sink node.

3.2.2 Selection Procedure

(i) Sink node broadcasts its node-id and query message to acquire current residual energy of each sensor node within its coverage area.

(ii) According to response from sensor nodes, sink node categorized sensor nodes into leader nodes having high energy and follower nodes having comparatively low energy.

(iii)Leader node having minimum distance from sink node, maximum residual energy among all other leader nodes and high reputation, is selected as cluster coordinator.

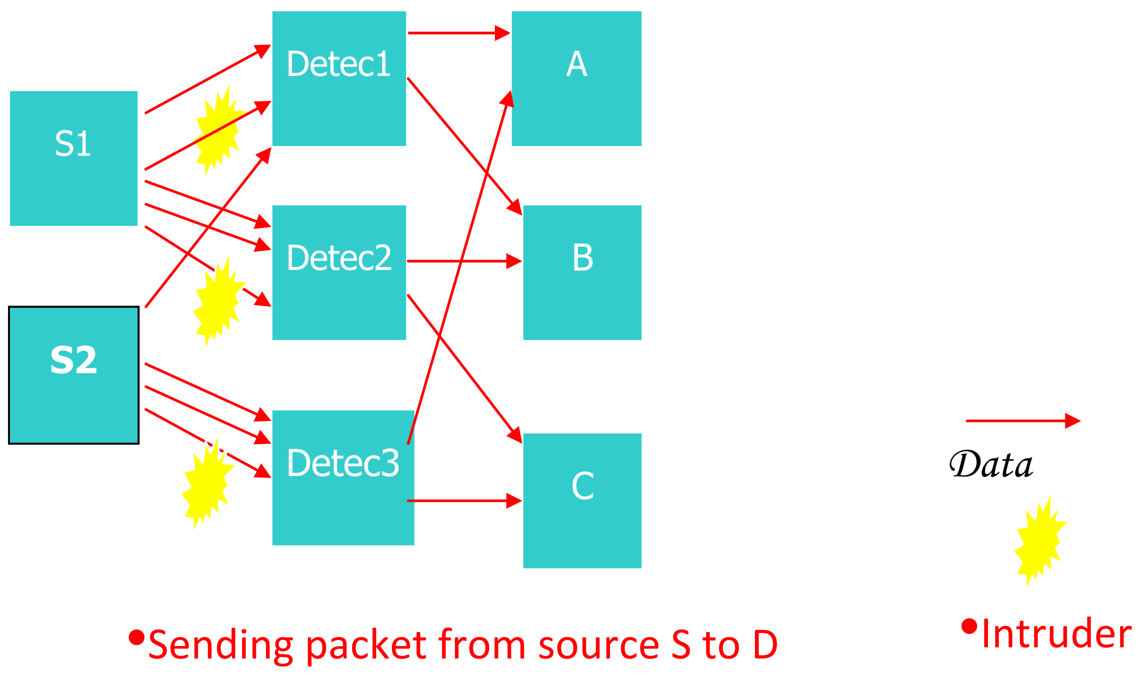
(iv)Remaining leader nodes within a cluster having high detection power is selected as sector monitor,whereas leader nodes having minimum distance from cluster coordinator is selected as forwarding sector head.

(v) Follower nodes having maximum energy among all follower nodes are selected as sector coordinator, other follower nodes are considered as leaf nodes.

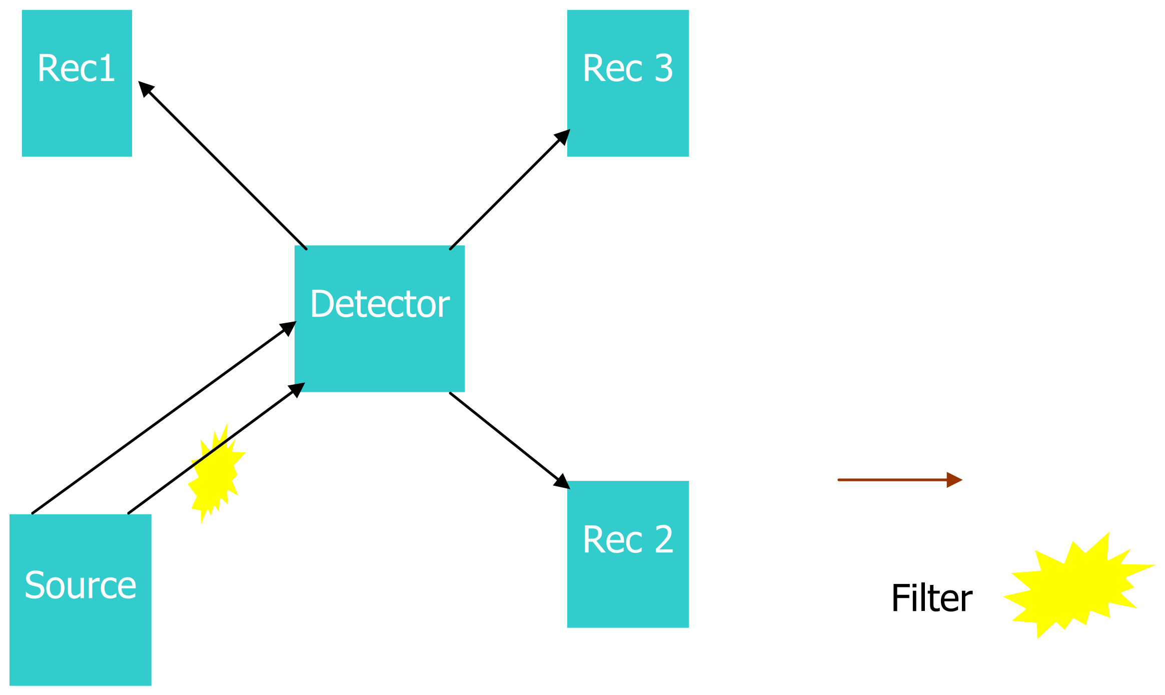
3.3 System Design

3.3.1 Data Flow Diagram

Heterogeneous:



Homogeneous:



3.3.2 Modules:

1 Constructing Sensor Network
2 Packet Creation
3 Find authorized and un authorized port
4 Constructing Inter-Domain Packet Filters
5 Receiving the valid packet

3.3.3 Modules Descriptions:

Module-1:

In this module, we are going to connect the network .Each node is connected the neighboring node and it is independently deployed in network area. And also deploy the each port no is authorized in a node.

Module-2:

In this module, browse and select the source file. And selected data is converted into fixed size of packets. And the packet is send from source to detector.

Module-3:

The intrusion detection is defined as a mechanism for a WSN to detect the existence of inappropriate, incorrect, or anomalous moving attackers. In this module check whether the path is authorized or unauthorized. If path is authorized the packet is send to valid destination. Otherwise the packet will be deleted. According port no only we are going to find the path is authorized or Unauthorized.

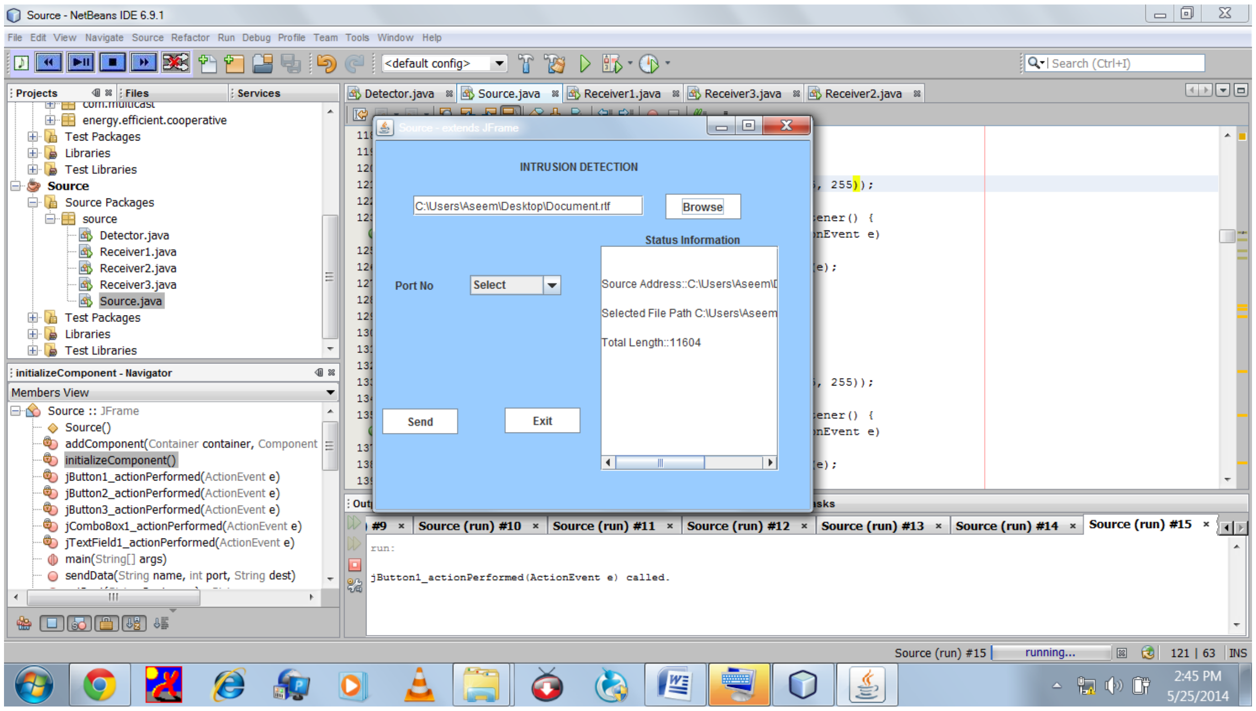
Module-4:

If the packet is received from other than the port no it will be filtered and discarded. This filter only removes the unauthorized packets and authorized packets send to destination.

Module-5:

In this module, after filtering the invalid packets all the valid Packets will reach the destination.

3.4 Output



INTRUSION DETECTION

C:\Users\Aseem\Desktop\Document.rtf

Port No: Select

Status Information

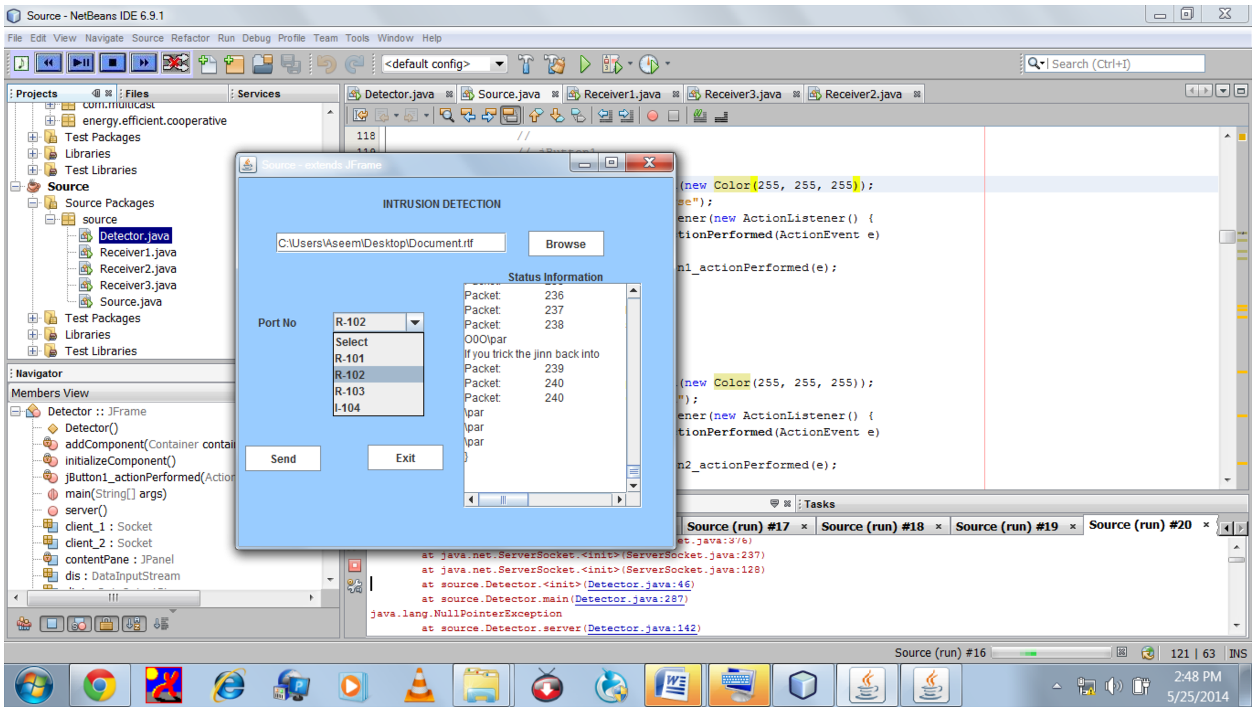
Source Address::C:\Users\Aseem\D

Selected File Path C:\Users\Aseem

Total Length::11604

```
run:

jButton1_actionPerformed(ActionEvent e) called.
```


INTRUSION DETECTION

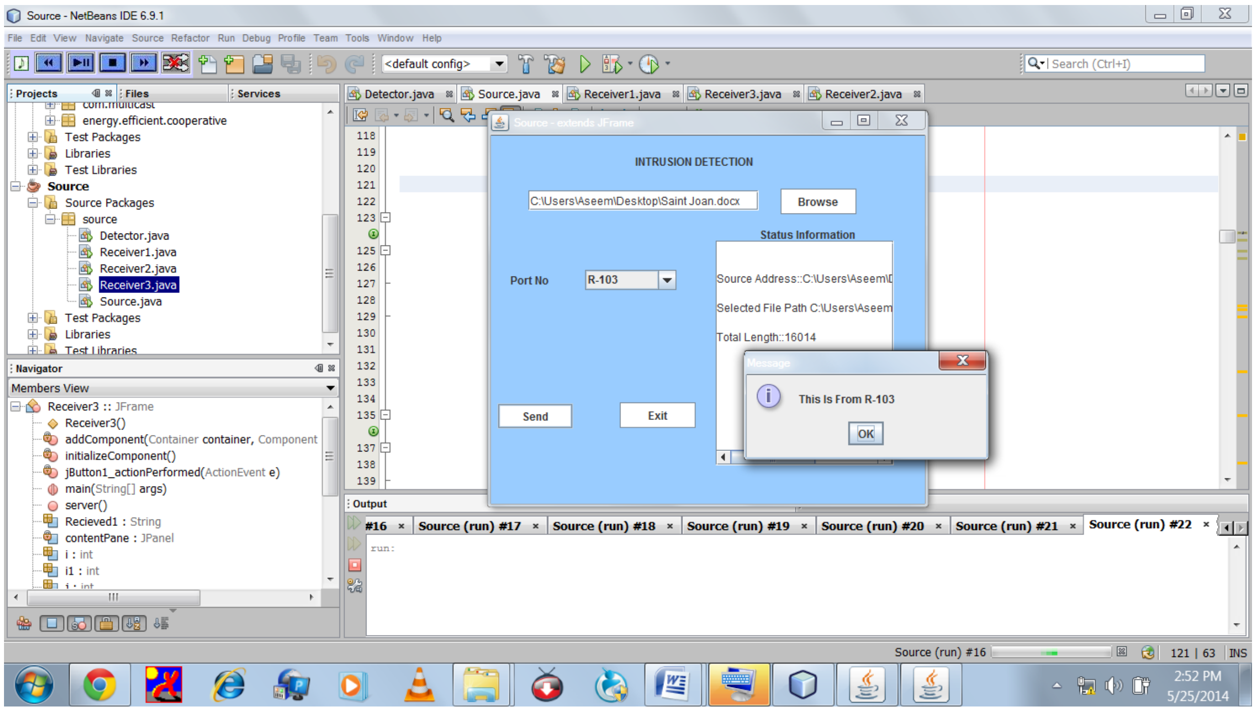
C:\Users\Aseem\Desktop\Document.rtf

Port No: R-102

Status Information

```
Packet:     236
Packet:     237
Packet:     238
OOO\par
If you trick the jinn back into
Packet:     239
Packet:     240
Packet:     240
\par
\par
\par
}
```

```
        at java.net.ServerSocket.<init>(ServerSocket.java:237)
        at java.net.ServerSocket.<init>(ServerSocket.java:128)
        at source.Detector.<init>(Detector.java:46)
        at source.Detector.main(Detector.java:287)
java.lang.NullPointerException
        at source.Detector.server(Detector.java:142)
```


INTRUSION DETECTION

C:\Users\Aseem\Desktop\Saint Joan.docx

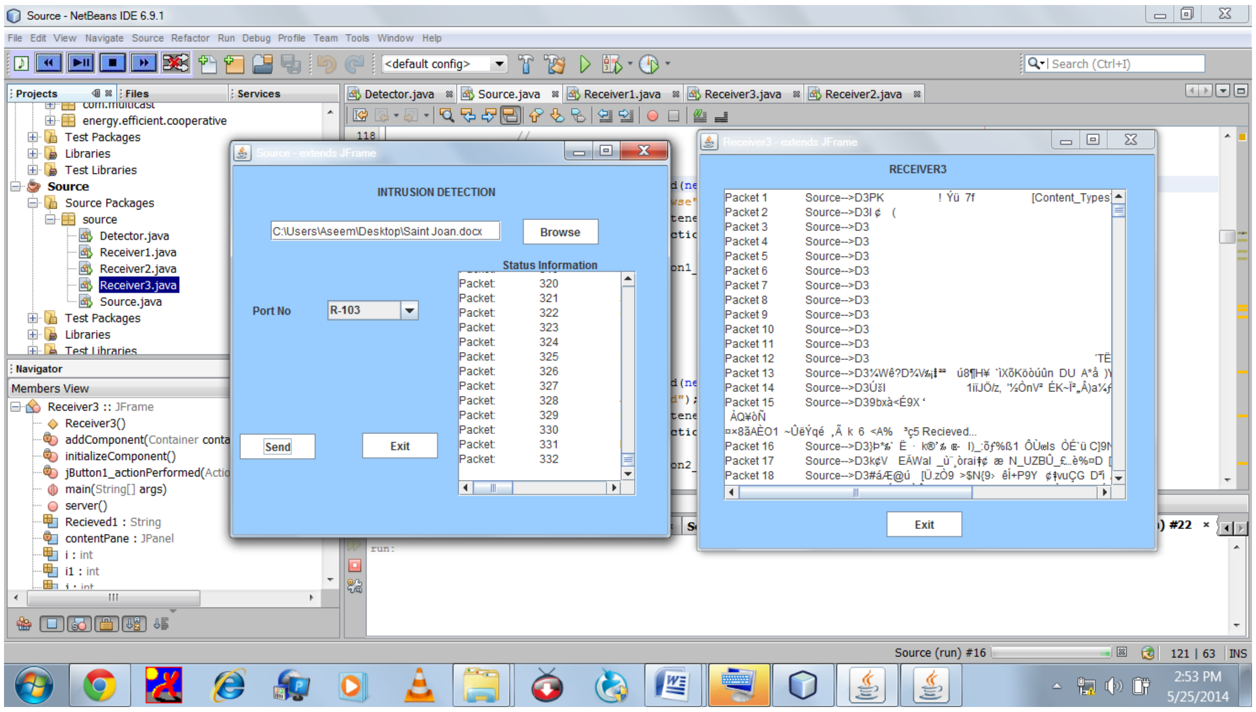
Port No: R-103

Status Information

```
Source Address::C:\Users\Aseem\D
Selected File Path C:\Users\Aseem
Total Length::16014
```

Message: This Is From R-103

```
run:
```



INTRUSION DETECTION

C:\Users\Aseem\Desktop\Saint Joan.docx Browse

Port No R-103

Status Information

```
Packet:    320
Packet:    321
Packet:    322
Packet:    323
Packet:    324
Packet:    325
Packet:    326
Packet:    327
Packet:    328
Packet:    329
Packet:    330
Packet:    331
Packet:    332
```

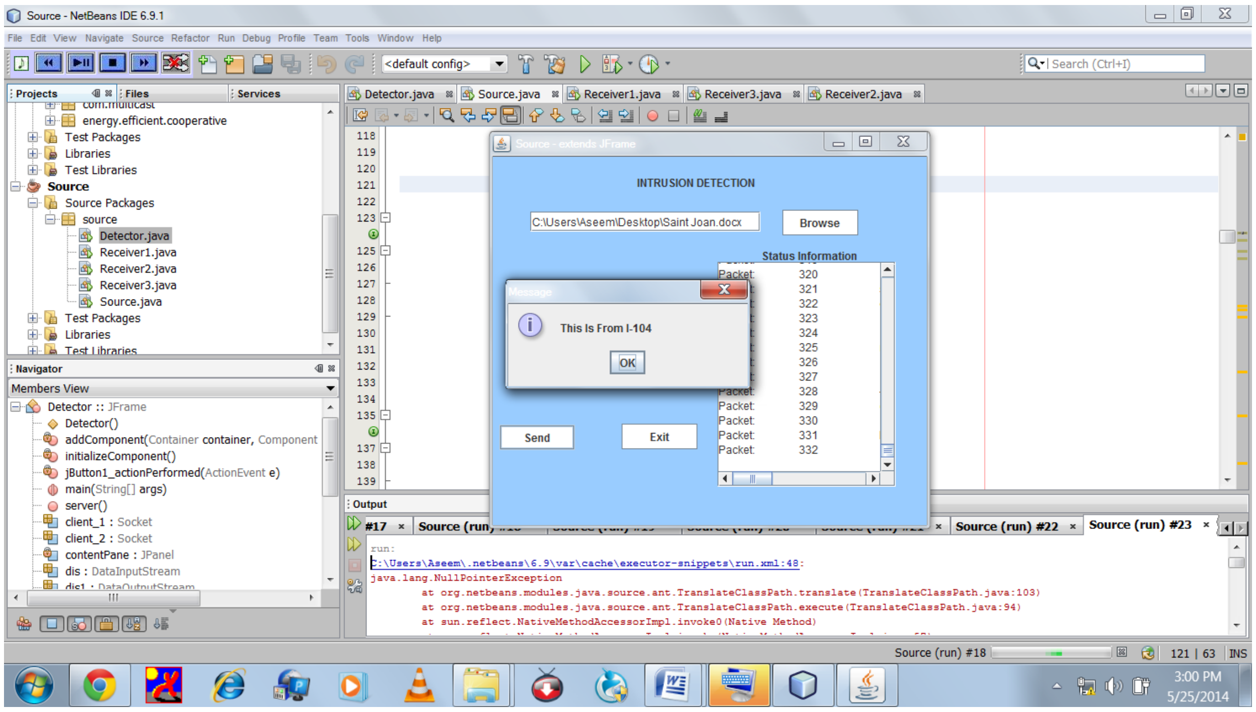
Send Exit

RECEIVER3

```
Packet 1     Source-->D3PK          ! Ýü  7f          [Content_Types]
Packet 2     Source-->D3l ¢   (
Packet 3     Source-->D3
Packet 4     Source-->D3
Packet 5     Source-->D3
Packet 6     Source-->D3
Packet 7     Source-->D3
Packet 8     Source-->D3
Packet 9     Source-->D3
Packet 10    Source-->D3
Packet 11    Source-->D3
Packet 12    Source-->D3                                       ´TË
Packet 13    Source-->D3¼Wê?D¾V‰¡‡ªª  ú8¶H¥ `ìXõKöòúûn DU A*å )Y
Packet 14    Source-->D3Úš l              1iïJÖ/z, '½ÒnVª ÉK~Ï²„Â)a¼f
Packet 15    Source-->D39bxà<É9X ‘
 ÀQ¥òÑ
¤×8ãAÈO1 ~ÛëÝqé ,Ã k 6 <A%  ³ç5 Recieved...
Packet 16    Source-->D3}Þ*‰` Ë · k®'‰ œ- I)_:õƒ%ß1 ÔÙœls ÒÉ`ü C]9N
Packet 17    Source-->D3k¢V  EÄWal _ù¯¸òrai‡¢ æ N_UZBÛ_£…è%¤D [
Packet 18    Source-->D3#áÆ@ú  [Ü.zÒ9 >$N{9› êÍ+P9Y ¢‡vuÇG Dª
```

Source (run) #16 121 | 63 INS

2:53 PM 5/25/2014



INTRUSION DETECTION

C:\Users\Aseem\Desktop\Saint Joan.docx Browse

Status Information

```
Packet:    320
Packet:    321
Packet:    322
Packet:    323
Packet:    324
Packet:    325
Packet:    326
Packet:    327
Packet:    328
Packet:    329
Packet:    330
Packet:    331
Packet:    332
```

This Is From I-104

Send Exit

Output

```
run:
C:\Users\Aseem\.netbeans\6.9\var\cache\executor-snippets\run.xml:48:
java.lang.NullPointerException
        at org.netbeans.modules.java.source.ant.TranslateClassPath.translate(TranslateClassPath.java:103)
        at org.netbeans.modules.java.source.ant.TranslateClassPath.execute(TranslateClassPath.java:94)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

Source (run) #18 121 | 63 INS

3:00 PM 5/25/2014

Chapter 4: Future Enhancements & Conclusion.

4.1 Future Scope

Our Future enhancements are intrusion detections in internet application and parallel computer interconnection network for a Wireless sensor network.

Though the design, development, deployment and testing of proactive network surveillance framework for solving network security problem has been successfully demonstrated by use of different result vectors and threat vectors, but there have been some limitations in the research work carried out in this thesis. One of the limitations, in the proposed framework, is automatic integration of new intruder signature to the Intruder Database. This portion of the framework requires human intervention arid lacks intelligence to incorporate new signatures automatically. Also, at Fifth layer, proposed work used a low interaction Deflect, which could be replaced with medium or high interaction counterpart to achieve better know-how about the unknown threats.
In future this research effort can be put on to a dedicated hardware platform with embedded Linux functionality. As a dedicated out of the box product this effort would fit into Unified Threat management group of devices. Another point which is left to the future development is integration of the database dumps into graph generation engine. In the research effort, we produced graphs manually, which takes a lot of effort and as observed many of the steps are of repeated nature. Thus, this portion of the work can also be automated. Hence, there is a lot of future scope of the research work to be carried out in this vital area of great significance to mankind.

4.2 Conclusions

This report analyzes the intrusion detection problem by characterizing intrusion detection probability with respect to the intrusion distance and the network parameters (i.e., node density, sensing range, and transmission range).The analytical model for intrusion detection allows us to analytically formulate intrusion detection probability within a certain intrusion distance under various application scenarios.

In this report, I present a review of recent works on different approaches of IDS for WSN. It has been observed that these intrusion detection systems are not adequate for protecting WSN from intruders efficiently. The need of the day is an IDS for detecting intrusions accurately in an energy-efficient manner.

Among the different types of prevalent attacks, sleep deprivation attack at link layer has been found to be the most devastating one for sensor nodes, exhausting the battery life very quickly. This paper comes up with the idea of a novel IDS that can mitigate sleep deprivation attack without using MAC based protocols like S-MAC, T-MAC, B-MAC, G-MAC. The outline of layer based approach using cluster technique to design a lightweight IDS capable of detecting insomnia of sensor nodes with less energy consumption has been documented here. The aim of this proposed model is to extend the lifetime of the WSN, even in the face of sleep deprivation attack. Generally, intruder attacks lower layer leaf nodes in HWSNET. In this model, intrusion detection is mainly focused on layer 1 that has no intrusion detection capacity of its own. Simulation proves the effectiveness of proposed model. At present work is on for more detailed analysis of IMIDS in a simulated environment.

Finally, the research contributions reported in this thesis are not only applicable to solve a single network implementation as demonstrated but the recommendation, algorithms, scripts and configuration steps reported are extremely useful in Unified Threat Management scenarios. At last, we wish to conclude that “Network Security” is not a product that one can purchase. It is a process. A long process that needs to be continually updated, improved, and monitored. Security depends upon various crucial components a.s demonstrated in this research like perimetric security, physical security; preventive security measures against unknown attacks and completes information system alerting system to alert about the health of network at regular intervals.

Security is a very difficult topic. Everyone has a different idea of what ``security" is, and what levels of risk are acceptable. The key for building a secure network is to *define what security means to your organization*. Once that has been defined, everything that goes on with the 4
network can be evaluated with respect to that policy. Projects and systems can then be broken down into their components, and it becomes much simpler to decide whether what is proposed will conflict with your security policies and practices. Many people pay great amounts of lip service to security, but do not want to be bothered with it when it gets in their way. It's important to build systems and networks in such a way that the user is not constantly reminded of the

security system around him. Users who find security policies and systems too restrictive will find ways around them. It's important to get their feedback to understand what can be improved, and it's important to let them know *why* what's been done has been, the sorts of risks that are deemed unacceptable, and what has been done to minimize the organization's exposure to them. Security is everybody's business, and only with everyone's cooperation, an intelligent policy, and consistent practices, will it be achievable.

APPENDIX

Module Coding

(A) Source Coding

```
/*                   Source                              */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.net.*;
/**
 * Summary description for Source
 **/
public class Source extends JFrame
{
          // Variables declaration
          private JLabel jLabel1;
          private JLabel jLabel2;
          private JLabel jLabel3;
          private JTextField jTextField1;
          private JComboBox jComboBox1;
          private JTextArea jTextArea1;
          private JScrollPane jScrollPane1;
          private JButton jButton1;
          private JButton jButton2;
          private JButton jButton3;
          private JPanel contentPane;
          String msg="";
```

```
        int flag=1;
        int flag1=1;
        Socket n1_client;
        String destination;
        int limit;
        String a[]={"Select","R-101","R-102","R-103","I-104"};
        int len;
        int packets;
        int rem;
        // End of variables declaration



        public Source()
        {
                super();
                initializeComponent();

                this.setVisible(true);
        }

         * This method is called from within the constructor to initialize the form.
private void initializeComponent()
        {
                jLabel1 = new JLabel();
                jLabel2 = new JLabel();
                jLabel3 = new JLabel();
                jTextField1 = new JTextField();
                jComboBox1 = new JComboBox(a);
                jTextArea1 = new JTextArea();
                jScrollPane1 = new JScrollPane();
                jButton1 = new JButton();
```

```
jButton2 = new JButton();
jButton3 = new JButton();
contentPane = (JPanel)this.getContentPane();

jLabel1.setText("INTRUSION DETECTION");
jLabel2.setText("Port No");
jLabel3.setText("Status Information");
jTextField1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
                jTextField1_actionPerformed(e);
        }

});
jComboBox1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
                jComboBox1_actionPerformed(e);
        }

});
jScrollPane1.setViewportView(jTextArea1);
jButton1.setBackground(new Color(255, 255, 255));
jButton1.setText("Browse");
jButton1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
                jButton1_actionPerformed(e);
        }

});
```

```
jButton2.setBackground(new Color(255, 255, 255));
jButton2.setText("Send");
jButton2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
                jButton2_actionPerformed(e);
        }

});
jButton3.setBackground(new Color(255, 255, 255));
jButton3.setText("Exit");
jButton3.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
                jButton3_actionPerformed(e);
        }

});
//
// contentPane
//
contentPane.setLayout(null);
contentPane.setBackground(new Color(153, 204,255));
addComponent(contentPane, jLabel1, 158,14,136,28);
addComponent(contentPane, jLabel2, 21,149,60,18);
addComponent(contentPane, jLabel3, 294,97,118,23);
addComponent(contentPane, jTextField1, 41,60,252,22);
addComponent(contentPane, jComboBox1, 103,147,100,22);
addComponent(contentPane, jScrollPane1, 246,115,195,246);
addComponent(contentPane, jButton1, 317,58,83,28);
addComponent(contentPane, jButton2, 7,293,83,28);
```

```
                addComponent(contentPane, jButton3, 141,292,83,28);
                //
                // Source
                //
                this.setTitle("Source - extends JFrame");
                this.setLocation(new Point(66, 48));
                this.setSize(new Dimension(483, 435));
        }

        /** Add Component Without a Layout Manager (Absolute Positioning) */
        private void addComponent(Container container,Component c,int x,int y,int
width,int height)
        {
                c.setBounds(x,y,width,height);
                container.add(c);
        }

        private void jTextField1_actionPerformed(ActionEvent e)
        {
                System.out.println("\njTextField1_actionPerformed(ActionEvent e)
called.");
                }

        private void jComboBox1_actionPerformed(ActionEvent e)
        {
                System.out.println("\njComboBox1_actionPerformed(ActionEvent e)
called.");
                Object o = jComboBox1.getSelectedItem();
                destination=""+o;
                flag=0;
```

```
	}

	private void jButton1_actionPerformed(ActionEvent e)
	{
		System.out.println("\njButton1_actionPerformed(ActionEvent e) called.");
		try
		{
			int b;
			msg="";
				FileDialog fd=new FileDialog(this,"Open",FileDialog.LOAD);
				fd.show();
				FileInputStream fos=new
FileInputStream(fd.getDirectory()+fd.getFile());
				jTextField1.setText(fd.getDirectory()+fd.getFile());
				while((b=fos.read())!=-1)
				{
					msg+=(char)b;
				}

			flag1=0;
			int len=msg.length();
			int packets=len/48;
			int rem=len%48;
			packets++;
      String source=jTextField1.getText();
      jTextArea1.append("\n\nSource Address::"+source+"\n\n");
      jTextArea1.append("Selected File Path "+fd.getDirectory()+fd.getFile()+"\n\n");
      jTextArea1.append("Total Length::"+len+"\n\n");

	}
		catch (Exception ex)
		{
			ex.printStackTrace();
		}

	}

	private void jButton2_actionPerformed(ActionEvent e)
	{
		System.out.println("\njButton2_actionPerformed(ActionEvent e) called.");
		String dest;
		if(flag==0)
		{
			if(destination.equalsIgnoreCase(a[0]))
			{
				JOptionPane.showMessageDialog(null,"Select the 
Destination!..");
			}
			else if(destination.equalsIgnoreCase(a[1]))
			{
				JOptionPane.showMessageDialog(null,"This Is From R-101");
				dest=setDest(a[1]);
				sendData("localhost",111,dest);

			}
			else if(destination.equalsIgnoreCase(a[2]))
			{
				JOptionPane.showMessageDialog(null,"This Is From R-102");
				dest=setDest(a[2]);
				sendData("localhost",111,dest);
			}
			else if(destination.equalsIgnoreCase(a[3]))
```

```
                    {
                              JOptionPane.showMessageDialog(null,"This Is From R-103");
                              dest=setDest(a[3]);
                              sendData("localhost",111,dest);

                    }
                    else if(destination.equalsIgnoreCase(a[4]))
                    {
                              JOptionPane.showMessageDialog(null,"This Is From I-104");
                              dest=setDest(a[4]);
                              sendData("localhost",111,dest);
                    }
          }
          else
          JOptionPane.showMessageDialog(null,"Load the File OR Select the
Destination!..");


     }

     private void jButton3_actionPerformed(ActionEvent e)
     {
          System.out.println("\njButton3_actionPerformed(ActionEvent e) called.");

     }

public void sendData(String name,int port,String dest)
     {
          try
          {
```

```
                              n1_client=new Socket(name,port);
                              DataOutputStream out=new
DataOutputStream(n1_client.getOutputStream());

                              if(out!=null)
                              {
                                          out.flush();
                              }
                                          int outgoing=0;
                                          byte buffer[]=msg.getBytes();
                                          int len=buffer.length;
                                          int tlength=buffer.length/48;
                                          int length11=buffer.length%48;
                                          int len1=len;
                                    if(length11!=0)
                                    {
                                          tlength++;
                                    }
                                    out.writeInt(tlength);
                                    out.writeUTF(destination);
                                    int st=0;
                                    int end=48;
                        jTextArea1.append("Packet Length:    "+len+"\n");

                              if(len<=48)
                               {


                                           out.writeUTF(dest+msg);
                                           jTextArea1.append("Packet:
"+"\t"+(++outgoing)+"\t"+msg+"\n");
```

```
                              }
                              else
                               {

                                              jTextArea1.append("Packet:
"+"\t"+(++outgoing)+"\t"+msg.substring(st,end)+"\n");
                                              out.writeUTF(dest+msg.substring(st,end));


                                     while(len1>48)
                                     {
                                              len1-=48;
                                              if(len1<=48)
                                               {

                                              jTextArea1.append("Packet:
"+"\t"+(++outgoing)+"\t"+msg.substring(end,len)+"\n");
                                              out.writeUTF(dest+msg.substring(end,len));



                                               }
                                               else
                                               {

                                                            int sp=end+48;
                                                            jTextArea1.append("Packet:
"+"\t"+(++outgoing)+"\t"+msg.substring(end,sp)+"\n");

      out.writeUTF(dest+msg.substring(end,sp));
                                                            end=sp;
```

```
                                        }
                              }
                     }
          }
          catch (Exception exp)
          {
                    exp.printStackTrace();
          }



}
public String setDest(String Destname)
{
          String Destinationname="";
          if(Destname.equalsIgnoreCase(a[1]))
          {
                    Destinationname="Source-->D1";
          }
          else if(Destname.equalsIgnoreCase(a[2]))
          {
                    Destinationname="Source-->D2";
          }
          else if(Destname.equalsIgnoreCase(a[3]))
          {
                    Destinationname="Source-->D3";
          }
          else if (Destname.equalsIgnoreCase(a[4]))
          {
                    Destinationname="Source-->D4";
```

```
                }
                return Destinationname;

        }




//============================= Testing ================================//
//= The following main method is just for testing this class .=//

        public static void main(String[] args)
        {

                new Source();
        }
//= End of Testing =


}
```

(B) Detector Coding

```
/*                  Detector                              */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.net.*;
import java.lang.*;
public class Detector extends JFrame
{
        private JLabel jLabel1;
        private JTextArea jTextArea1;
        private JScrollPane jScrollPane1;
        private JButton jButton1;
        private JPanel contentPane;
        ServerSocket server_1;
        DataOutputStream dis1;
        DataOutputStream dis2;
        DataInputStream dis;
        Socket socket_1;
        Socket client_1;
        Socket client_2;
        long temp;
        int i=1;
        int length;

        // End of variables declaration


        public Detector()
        {
                super();
                initializeComponent();
                this.setVisible(true);
                try
                {
                        server_1=new ServerSocket(111);

                }
                catch (Exception exp)
                {
                        exp.printStackTrace();
                }

                this.setVisible(true);
        }



        private void initializeComponent()
        {
                jLabel1 = new JLabel();
                jTextArea1 = new JTextArea();
                jScrollPane1 = new JScrollPane();
                jButton1 = new JButton();
                contentPane = (JPanel)this.getContentPane();

                // jLabel1
                jLabel1.setText("INTRUSION DETECTOR");
                // jTextArea1
                // jScrollPane1
                jScrollPane1.setViewportView(jTextArea1);
                // jButton1
                jButton1.setBackground(new Color(255, 255, 255));
                jButton1.setText("Exit");
                jButton1.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e)
```

```
                    {
                            jButton1_actionPerformed(e);
                    }

            });
            // contentPane
            contentPane.setLayout(null);
            contentPane.setBackground(new Color(153, 204, 255));
            addComponent(contentPane, jLabel1, 172,9,133,38);
            addComponent(contentPane, jScrollPane1, 26,37,441,341);
            addComponent(contentPane, jButton1, 204,390,83,28);
            // Detector
            this.setTitle("Detector - extends JFrame");
            this.setLocation(new Point(0, 0));
            this.setSize(new Dimension(505, 462));
    }

    private void addComponent(Container container,Component c,int x,int y,int
width,int height)
    {
            c.setBounds(x,y,width,height);
            container.add(c);
    }

    private void jButton1_actionPerformed(ActionEvent e)
    {
            System.out.println("\njButton1_actionPerformed(ActionEvent e) called.");

    }
```

```
public void server()
        {

                                                try
                                                {

                                                String rr="";
                                                socket_1=server_1.accept();
                                                dis=new
DataInputStream(socket_1.getInputStream());
                                                int length=dis.readInt();
                                                String destination=dis.readUTF();
                                                if(destination.equalsIgnoreCase("R-101"))
                                                        {

        jTextArea1.append("\t**********************************\n");
                                                                jTextArea1.append("\tTHIS IS
FROM PORT I_101\n");

        jTextArea1.append("\t**********************************\n");
                                                                client_1=new
Socket("localhost",101);
                                                                dis1=new
DataOutputStream(client_1.getOutputStream());
                                                                dis1.writeInt(length);
                                                while(length>0)
                                                        {

                                                                rr=dis.readUTF();
                                                                jTextArea1.append("Packet
"+i+"\t"+rr+" Recieved...\n");
```

```
                                                            dis1=new
DataOutputStream(client_1.getOutputStream());
                                                            dis1.writeUTF(rr);
                                                            length--;
                                                            i++;
                                                    }

                                                    }
                                            else if (destination.equalsIgnoreCase("R-102"))
                                            {

        jTextArea1.append("\t**********************************\n");
                                                            jTextArea1.append("\tTHIS IS
FROM PORT R_102\n");

        jTextArea1.append("\t**********************************\n");
                                                            client_1=new
Socket("localhost",102);
                                                            dis1=new
DataOutputStream(client_1.getOutputStream());
                                                            dis1.writeInt(length);

                                            while(length>0)
                                                    {

                                                            rr=dis.readUTF();
                                                            jTextArea1.append("Packet
"+i+"\t"+rr+" Recieved...\n");
                                                            dis1=new
DataOutputStream(client_1.getOutputStream());
                                                            dis1.writeUTF(rr);
```

```
                                        length--;
                                        i++;
                                    }




                                }
                                else if (destination.equalsIgnoreCase("R-103"))
                                {

    jTextArea1.append("\t**********************************\n");
                                        jTextArea1.append("\tTHIS IS
FROM PORT R_103\n");

    jTextArea1.append("\t**********************************\n");
                                        client_1=new
Socket("localhost",103);
                                        dis1=new
DataOutputStream(client_1.getOutputStream());
                                        dis1.writeInt(length);

                                while(length>0)
                                    {

                                        rr=dis.readUTF();
                                        jTextArea1.append("Packet
"+i+"\t"+rr+" Recieved...\n");
                                        dis1=new
DataOutputStream(client_1.getOutputStream());
                                        dis1.writeUTF(rr);
                                        length--;
                                        i++;
                                    }




                                }
                                else if (destination.equalsIgnoreCase("I-104"))
                                {

        jTextArea1.append("\t**********************************\n");
                                        jTextArea1.append("\tTHIS IS
FROM PORT I_104\n");

        jTextArea1.append("\t**********************************\n");
                                        while(length>0)
                                            {

                                            rr=dis.readUTF();
                                            StringBuffer sb=new
StringBuffer(rr);
                                            sb.delete(7,10);

        jTextArea1.append("\t\tPacket "+i+"\t"+rr.substring(4,15)+" Recieved...\n");
                                            length--;
                                            i++;

                                            }

        JOptionPane.showMessageDialog(null,"This Is From Intruder Port");

        JOptionPane.showMessageDialog(null,"The Intruder Has Been Detected");
```

```
                                        }




                                        }
                                        catch (Exception exp)
                                        {
                                                exp.printStackTrace();
                                        }





                }

        //============================ Testing ==========================//
                public static void main(String[] args)
                {
                        Detector r1=new Detector();
                        while (true)
                        {
                                r1.server();
                        }
                }}
```

REFRENCES

1. Yuxin Mao, “A Semantic-based Intrusion Detection Framework for Wireless Sensor Network”, Networked Computing (INC), 6th International Conference, Gyeongju, Korea (South) (2010)

2. Sudip Misra, P. Venkata Krishna and Kiran Isaac Abraham, “Energy Efficient Learning Solution for Intrusion Detection in Wireless Sensor Networks” , COMSNETS'10 Proceedings of the 2nd international conference on Communication systems and Networks (2010)

3. Garth V. Crosby, Lance Hester, and Niki Pissinou, “Location-aware, Trust-based Detection and Isolation of Compromised Nodes in Wireless Sensor Networks”, International Journal of Network Security, Vol.12, No.2, PP.107- 117 (Mar. 2011)

4. Rung-Ching Chen, Chia-Fen Hsieh and Yung-Fa Huang, “An Isolation Intrusion Detection System for Hierarchical Wireless Sensor Network”, Journal of Networks, Vol. 5, Number 3 (March 2010)

5. Rung-Ching Chen, Yung-Fa Huang, Chia-Fen Hsieh, “Ranger Intrusion Detection System for Wireless Sensor Networks with Sybil Attack Based on Ontology”, New Aspects of Applied Informatics, Biomedical Electronics and Informatics and Communications (2010)

6. Mohammad Saiful Islam Mamun, A.F.M. Sultanul Kabir, “Hierarchical Design Based Intrusion Detection System For Wireless Ad Hoc Sensor Network” , International Journal of Network Security & Its Applications (IJNSA), Vol.2, No.3 (July 2010)

7. K.Q. Yan, S.C. Wang, C.W. Liu, “A Hybrid Intrusion Detection System of Cluster-based Wireless Sensor Networks”, Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 , Vol IIMECS 2009 ,Hong Kong (March 18 - 20, 2009)

8. Idris M. Atakli , Hongbing Hu, Yu Chen, Wei-Shinn Ku, Zhou Su “Malicious Node Detection in Wireless Sensor Networks using Weighted Trust Evaluation”, The Symposium on Simulation of Systems Security (SSSS’08), Ottawa, Canada (April 14 –17, 2008)

9. Guangcheng Huo, Xiaodong Wang, "DIDS: A Dynamic Model of Intrusion Detection System in Wireless Sensor Networks", IEEE, International Conference on Information and Automation, Zhangjiajie, China (June 20 –23, 2008)

10. Piya Techateerawat, Andrew Jennings "Energy Efficiency of Intrusion Detection Systems in Wireless Sensor Networks", IEEE2006 WIC (2006)

11. R. Hemenway, R. Grzybowski, C. Minkenberg, and R. Luijten, "Optical-packet-switched interconnect for supercomputer applications,"*OSA J. Opt. Netw.*, vol. 3, no. 12, pp. 900–913, Dec. 2004.

12. C. Minkenberg, F. Abel, P. Müller, R. Krishnamurthy, M. Gusat, P.Dill, I. Iliadis, R. Luijten, B. R. Hemenway, R. Grzybowski, and E.Schiattarella, "Designing a crossbar scheduler for HPC applications,"*IEEE Micro*, vol. 26, no. 3, pp. 58–71, May/Jun. 2006.

13. E. Oki, R. Rojas-Cessa, and H. Chao, "A pipeline-based approach formaximal-sized matching scheduling in input-buffered switches," *IEEE Commun. Lett.*, vol. 5, no. 6, pp. 263–265, Jun. 2001.

14. C. Minkenberg, I. Iliadis, and F. Abel, "Low-latency pipelined crossbar arbitration," in *Proc. IEEE GLOBECOM 2004*, Dallas, TX, Dec. 2004, vol. 2, pp. 1174–1179.

15. C. Minkenberg, R. Luijten, F. Abel, W. Denzel, and M. Gusat, "Current issues in packet switch design," *ACM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 119–124, Jan. 2003.

16. C. Minkenberg, F. Abel, P. Müller, R. Krishnamurthy, and M. Gusat,"Control path implementation of a low-latency optical HPC switch," in*Proc. Hot Interconnects 13*, Stanford, CA, Aug. 2005, pp. 29–35.

17. C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load-balanced Birkhoff-von Neumann switches, part I: One-stage buffering," *Elsevier Comput.Commun.*, vol. 25, pp. 611–622, 2002.

18. A. Tanenbaum, *Computer Networks*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

19. R. Krishnamurthy and P. Müller, "An input queuing implementation for low-latency speculative optical switches," in *Proc. 2007 Int. Conf.Parallel Processing Techniques and Applications (PDPTA'07)*, Las Vegas, NV, Jun. 2007, vol. 1, pp. 161–167.

20. H. Takagi, *Queueing Analysis, Volume 3: Discrete-Time Systems*. Amsterdam: North-Holland, 1993.