INDENTIFYING PACKET DROPPERS AND MODIFIERS IN WIRELESS SENSOR NETWORKS

A PROJECT REPORT

Submitted by

SURYA PRABA .E

in partial fulfillment for the requirement of award of the degree of

MASTER OF ENGINEERING

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE 641 049

(An Autonomous Institution Affiliated to Anna University, Chennai)

APRIL 2013

IDENTIFYING PACKET DROPPERS AND MODIFIERS IN WIRELESS SENSOR NETWORKS

A PROJECT REPORT

Submitted by

SURYA PRABA .E

in partial fulfillment for the requirement of award of the degree of

MASTER OF ENGINEERING
in

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE 641 049

(An Autonomous Institution Affiliated to Anna University, Chennai)

APRIL 2013

BONAFIDE CERTIFICATE

Certified that this project work titled **"IDENTIFYING PACKET DROPPERS AND MODIFIERS IN WIRELESS SENSOR NETWORKS"** is the bonafide work of Ms. SURYA PRABA, E. (1120108019) who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other students.

Prof. N. Jayapathi M.Tech.,	Mr. K. Sivan Arul Selvan M.E., (Ph.D.),
HEAD OF DEPARTMENT,	SUPERVISOR
PROFESSOR,	Associate Professor,
Computer Science and Engineering,	Computer Science and Engineering,
Kumaraguru College of Technology,	Kumaraguru College of Technology,
Coimbatore-641 049.	Coimbatore-641 049.

Submitted for the Project Viva-Voce examination held on ____________.

--------------------------------	----------------------------
Internal Examiner	External Examiner

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project.

I express my profound gratitude to our **Chairman Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E., Co-Chairman Dr. B. K. Krishnaraj Vanavarayar, B.Com., B.L., Correspondent Mr. M. Balasubramaniam, M.Com., M.B.A., and Joint Correspondent Mr. K. Shankar Vanavarayar**, **M.B.A., PGDIEM** for giving his great opportunity to purse this course. I would like to thank **Principal Dr.S.Ramachandran, Ph.D.,** for providing the necessary facilities to complete my thesis.

I take this opportunity to thank **Prof.N.Jayapathi, M.Tech**., Head of the Department, Department of Computer Science and Engineering, for his support and timely motivation. Special thanks to my Project Coordinator **Dr.V.Vanitha, Ph.D.**, Senior Associate Professor, Department of Computer Science and Engineering, and project committee members for arranging brain storming project review sessions.

I register my sincere thanks to my Guide **Mr. K. Sivan Arul Selvan, M.E., (Ph.D.),** Senior Associate Professor, Department of Computer Science and Engineering, my thesis advisor. I am grateful for his support, encouragement and ideas. I would like to convey my honest thanks to all **Teaching** and **Non Teaching Staff** members of the department and my classmates for their support.

I dedicate this project work to my **Parents** for no reasons but feeling from bottom of my heart that without their love, this work wouldn't be possible.

-SURYA PRABA E

TABLE OF CONTENT

ABSTRACT

In Wireless Sensor Network, sensors at different locations can generate streaming/ discrete data, which can analysed in real-time/Non real-time to identify events of interest. A sensor network is often deployed in an unattended and hostile environment to perform monitoring and data collection tasks. When it is deployed in such an environment, it lacks physical protection and is subject to node compromise. After compromising one or multiple sensor nodes, an adversary may launch various attacks to disrupt the in-network communication. In this project, the node which acts as the packet droppers or modifiers are identified based on the delay and energy consumption. Based on the information available in the routing table, the consumed energy at each node is calculated. A compromised node consumes more energy for dropping or modifying the packet than other nodes and the delay is increased. Hence the compromised nodes are identified and the packets are routed in the alternate path.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC	Analog and Digital Converter
BS	Base Station
CDMA	Code Division Multiple Access
CSMA	Carrier Sense Multiple Access
DPDSN	Detection Of Packet Dropping Attacks for Sensor Network
DSR	Dynamic Source Routing
LEACH	Low-Energy Adaptive clustering Hierarchy
MAC	Medium Access control
PNM	Probabilistic Nested Marking
QOS	Quality of Service
RF	Radio Frequency
SEF	Statistical En-route Filtering
UNPF	Unified Network Protocol Framework
WSN	Wireless Sensor Network

CHAPTER 1

INTRODUCTION

A Wireless Sensor Network (WSN) is a self-configuring network of small sensor nodes communicating among themselves using radio signals, and deployed in quantity to sense, monitor and understand the physical world. It consists of spatially distributed autonomous sensors (Figure 1.1) to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location.

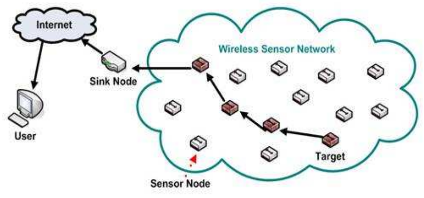


Figure1.1 WSN Architecture

The more number of modern networks are bi-directional, enables to control the activity of the sensors. 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.

1.1 SENSOR NODE

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 wireless sensor node is composed of four basic components as shown in Figure 1.2: a sensing unit, a processing unit (microcontroller), a transceiver unit and a power unit.

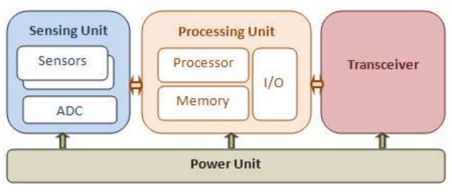


Figure 1.2 Components of Sensor Node

In addition to the above units, a wireless sensor node may include a number of application-specific components, for example a location detection system or mobiliser; for this reason, many commercial sensor node products include expansion slots and support serial wired communication.

Sensing Unit: The main function of the sensing unit is to sense or measure physical data from the target area. The analog voltage or signal is generated by the sensor corresponding to the observed phenomenon. The continual waveform is digitized by an Analog-to-Digital Converter (ADC) and then delivered to the processing unit for further analysis. The sensing unit is a current technology bottleneck because the sensing technologies are much slower than those of the semi-conductors.

Processing Unit: The processing unit which is generally associated with a small storage unit manages the procedures that make the sensor nodes collaborate with the other nodes to carry out the assigned sensing tasks.

Transceiver: There are three deploying communication schemes in sensors which include optical communication (Laser), Infrared, and Radio-Frequency (RF). Laser consumes less energy than radio and provides high security, but requires line of sight and is sensitive to atmospheric conditions. Infrared, like laser, needs no antenna but is limited in its broadcasting capacity. RF is the most easy to use but requires antenna. Various energy consumption reduction strategies have been developed such as modulation, filtering, and demodulation. Amplitude and frequency modulation are standard mechanisms. Amplitude modulation is simple but susceptible to noise.

Power Unit: One of the most important components of a sensor node is the power unit. Every sensor node is equipped with a battery that supplies power to remain in active mode. Power consumption is a major weakness of sensor networks. Any energy preservation schemes can help to extend sensor’s lifetime. Batteries used in sensors can categorized into two groups; rechargeable and non-rechargeable. Often in harsh environments, it is impossible to recharge or change a battery.

1.2 WIRELESS SENSOR NETWORKS ARCHITECTURE

The architecture of Wireless Sensor Networks is classified into two types.

1.2.1 Layered Architecture

In this type of architecture there is a single powerful base station (BS) and layers of sensor nodes are formed around BS, based on their hop count distance to reach BS. Therefore, in general layer i denote all nodes that are i-hop away from BS. Layered architecture is depicted in Figure 1.3.

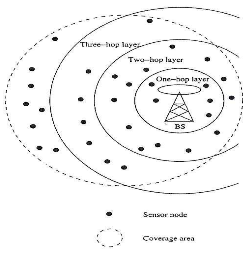


Figure1.3 Layered Architecture

Unified Network Protocol Framework (UNPF)

It is a type of layered architecture with a set of protocols that integrates the following operations:

- Network Initialization & Maintenance Protocol

Unified Network Protocol Framework (UNPF)

It is a type of layered architecture with a set of protocols that integrates the following operations:

- Network Initialization & Maintenance Protocol
- Medium Access Control Protocol
- Routing Protocol

Network Initialization & Maintenance Protocol:

- BS broadcasts its ID using Code Division Multiple Access(CDMA) common control channel (BS reaches all nodes in one hop)
- Nodes record BS ID & send beacon signal with their own IDs at their low default power levels
- All nodes ,the BS can hear are at 1-hop distance

The BS broadcasts a control packet with all layer one node IDs

- All nodes send a beacon signal again
- The layer one nodes record the IDs they hear-layer 2
- The layer one nodes inform the BS of the layer 2
- The BS broadcasts the layer2 nodes IDs
- **To maintain**: periodic beaconing updates are required.

MAC protocol

- A Time Division CDMA (TCDMA) protocol for spatial bandwidth reuse.
- Ensures a scheduling scheme for fair access.

Routing Protocol

- Downlink from the BS is by direct broadcast on the control channel
- Enables multi-hop data forwarding to the BS
- The remaining energy is considered when forwarding to the next hop (layer)
- Only the nodes of the next layer need to be maintained in the routing table.

1.2.2 Clustered Architecture

In this type of architecture sensor nodes are organized into clusters and each cluster is governed by a cluster-head. Only cluster heads send messages to a BS. This architecture is suitable for data fusion and is self-organizing in nature. This is depicted in Figure 1.4.

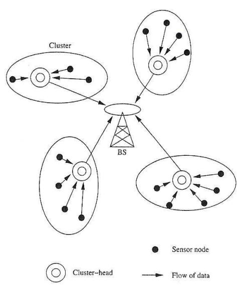


Figure1.4 Clustered Architecture

Low-Energy Adaptive Clustering Hierarchy (LEACH)

It is a self-organizing and adaptive clustering protocol which evenly distributes the energy expenditure among the sensors. It performs data aggregation where cluster heads act as aggregation points.
There are two main phases in this architecture.

- Setup phase: organizing the clusters
- Steady-state phase: deals with the actual data transfers to the BS.

Setup phase:

- Each sensor chooses a random number m between 0 and 1
- If m < $T(n)$ for node n, the node becomes a cluster-head where

$$T(n) = \begin{cases} \dfrac{P}{1 - P[r * \mathrm{mod}(1/P)]} & \text{if } n \in G \\ 0 & \text{otherwise}, \end{cases}$$

P: the desired percentage of cluster heads
r: the round number.
G: the set of nodes that have not been cluster heads during the last 1 / P rounds.

- A cluster head advertises its neighbours using a CSMA MAC.
- Surrounding nodes decide which cluster to join based on the signal strength of these messages.
- Cluster heads assign a TDMA schedule for their members.

Steady-state phase

- All source nodes send their data to their cluster heads.
- Cluster heads perform data aggregation/fusion through local transmission.
- Cluster heads send them back to the BS using a single direct transmission.
- After a certain period of time, cluster heads are selected again through the set-up phase.

Merits

- Accounting for adaptive clusters and rotating cluster heads.
- Opportunity to implement any aggregation function at the cluster heads.

Demerits:

- Highly dynamic environments.
- Continuous updates
- Mobility

1.3 DESIGN CHALLENGES OF WSN

1. Scalable and flexible architecture- the network must preserve its stability. Introducing more nodes into the network means that additional communication messages will be exchanged, so that these nodes are integrated into the existing network.

2. Error prone wireless medium- The wireless medium can greatly affected by noisy environments.

3. Fault tolerance and adaptability- Fault tolerance means to maintain sensor network functionalities without any interruption due to failure of sensor node because in sensor network every node have limited power of energy so the failure of single node doesn't affect the overall task of the sensor network.

4. Infrastructure- Sensors network are infrastructure less in which nodes can communicate directly with base station.

5. Node Deployment- Sensor network be deployed randomly in geographical area. After deployment, they can maintain automatically without human presence.

6. Real –Time- Achieving Real-Time in WSN is difficult to maintain. It must support maximum bandwidth, minimum delay and several QOS parameters.

7. Dynamic changes- As in sensor network, nodes are deployed without any topology and they are adaptable to changes due to addition of new nodes or failure of nodes.

8. Power Consumption- Wireless sensor node is a microelectronic device means it is equipped with a limited number of power source. Nodes are dependent on battery for their power. Hence power conservation and power management is an important issue in wireless sensor network.

9. Production cost- As the name suggests production cost, in the sensor network there are large no of nodes deployed, so if a single node cost is very high then the cost of overall network will be very high.

10. Short Range Transmission- In WSNs, here the short transmission range should be considered in order to reduce the possibility of being eavesdropped.

11. Hardware design- While designing any hardware of sensor network, it should be energy-efficient.

12. Limited computational power and memory size- It is another factor that affects WSN in the sense that each node stores the data individually and sometime more than one node stored same data and transferred to the base station which is waste of power and storage capacity of nodes so have to develop effective routing schemes and protocols to minimize the redundancy in the network.

13. Quality of Service- It means data should be delivered within time period.

14. Security- Security is very important parameter in sensor network. Since sensor networks are data centric so there is no particular ID associated with sensor nodes and attacker can easily inserted himself into the network and stole the important data by becoming the part of network without the knowledge of sensor nodes of the network. So it is difficult to identify whether the information is authenticated or not.

For many applications in wireless sensor networks, users may want to continuously extract data from the networks for analysis later. However, accurate data extraction is difficult – it is often too costly to obtain all sensor readings, as well as not necessary in the sense that the readings themselves only represent samples of the true state of the world. In order to enable reliable and efficient observation and initiate right actions, physical phenomenon features should be reliably detected/estimated from the collective information provided by sensor nodes. Moreover, instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data.

1.4 ATTACKS

Wireless Sensor networks are vulnerable to security attacks due to the broadcast nature of the transmission medium. Figure 1.5 shows various attacks in Wireless Sensor Networks. Basically attacks are classified into two types. They are

1.4.1 Passive Attacks

The monitoring and listening of the communication channel by unauthorized attackers are known as passive attack. The Attacks against privacy is passive in nature.

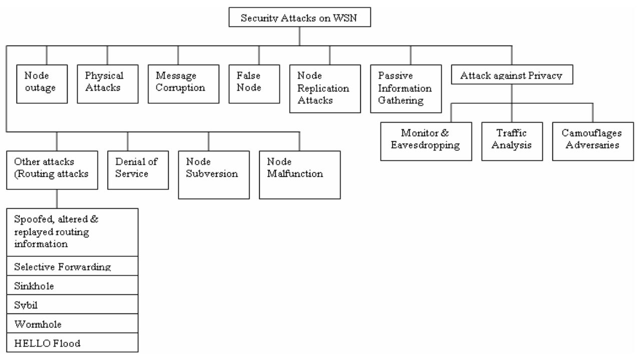


Figure1.5 The Attacks Classification on WSN

1.4.2 Active Attacks

The unauthorized attackers monitors, listens to and modifies the data stream in the communication channel are known as active attack.

1.5 SECURITY MECHANISMS

The security mechanisms are actually used to detect, prevent and recover from the security attacks. A wide variety of security schemes (Figure 1.6) can invent to counter malicious attacks and these can categorized as

1.5.1 Low-level mechanism
1.5.2 High-level mechanism

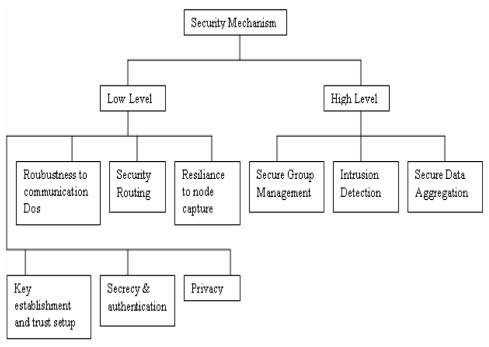


Fig.1.6 The Order of Security Mechanisms.

1.5.1 Low-Level Mechanism

Low-level security primitives for securing sensor networks includes,

a. Key establishment and trust setup
b. Secrecy and authentication
c. Privacy
d. Robustness to communication denial of service
e. Secure routing
f. Resilience to node capture

a. Key establishment and trust setup

The primary requirement of setting up the sensor network is the establishment of cryptographic keys. Key-establishment techniques need to scale to networks with hundreds or thousands of nodes. In addition, the communication patterns of sensor networks differ from traditional networks; sensor nodes may need to set up keys with their neighbours and with data aggregation nodes. The disadvantage of this approach is that attackers who compromised sufficiently and many nodes could also reconstruct the complete key pool and break the scheme.

b. Secrecy and authentication

Most of the sensor network applications require protection against eavesdropping, injection, and modification of packets. For point-to-point communication, end-to-end cryptography achieves a high level of security but requires that keys be set up among all end points and be incompatible with passive participation and local broadcast. Link-layer cryptography with a network wide shared key simplifies key setup and supports passive participation and local broadcast, but intermediate nodes might eavesdrop or alter messages.

c. Privacy

Like other traditional networks, the sensor networks have also to force privacy concerns. Initially the sensor networks are deployed for legitimate purpose might subsequently be used in unanticipated ways.

d. Robustness to communication denial of service

An adversary attempts to disrupt the network's operation by broadcasting a high-energy signal. If the transmission is powerful enough, the entire system's communication could be jammed. More sophisticated attacks are also possible; the adversary might inhibit communication by violating the 802.11 medium access control (MAC) protocol by, transmitting while a neighbour is also transmitting or by continuously requesting channel access with a request-to send signal.

e. Secure routing

Routing and data forwarding is a crucial service for enabling communication in sensor networks. Unfortunately, current routing protocols suffer from many security vulnerabilities. The simplest attacks involve injecting malicious routing information into the network, resulting in routing inconsistencies. Simple authentication might guard against injection attacks, but some routing protocols are susceptible to replay by the attacker of legitimate routing messages.

f. Resilience to node capture

One of the most challenging issues in sensor networks is resiliency against node capture attacks. In most applications, sensor nodes are likely to be placed in locations easily accessible to attackers. Such exposure raises the possibility that an attacker might capture sensor nodes, extract cryptographic secrets, modify their programming, or replace them with malicious nodes under the control of the attacker. Algorithmic solutions to the problem of node capture are preferable.

1.5.2 High-Level Mechanisms

High-level security mechanisms for securing sensor networks, includes

a. Secure group management
b. Intrusion detection
c. Secure data aggregation

a. Secure group management

Each and every node in a wireless sensor network is limited in its computing and communication capabilities. However, interesting in-network data aggregation and analysis can perform by groups of nodes. The actual nodes comprising the group may change continuously and quickly. Many other key services in wireless sensor networks are also performed by groups. Consequently, secure protocols for group management are required, securely admitting new group members and supporting secure group communication. The outcome of the group key computation is normally transmitted to a base station. The output must be authenticated to ensure it comes from a valid group.

b. Intrusion detection

Wireless sensor networks are susceptible to many forms of intrusion. Wireless sensor networks require a solution that is fully distributed and inexpensive in terms of communication, energy, and memory requirements. The use of secure groups may be a promising approach for decentralized intrusion detection.

c. Secure data aggregation

One advantage of a wireless sensor network is the fine grain sensing that large and dense sets of nodes can provide. The sensed values must be aggregated to avoid overwhelming amounts of traffic back to the base station. All aggregation locations must be secured.

1.6 LITERATURE SURVEY

In this section, the papers related to Packet dropping and modifications are discussed. These are two common attacks that can launch by an adversary to interrupt communication in wireless multi-hop sensor networks.From that, an effective scheme is proposed to identify misbehaving forwarders that drop or modify packets.

1.6.1 DPDSN: Detection of Packet-Dropping Attacks for WSN

A lightweight solution (Bhuse 2005) is proposed to identify paths that drop packets (Figure 1.7) by using alternate paths. Alternate paths are found during route discovery and it incurs no additional cost because one of the alternate paths is utilized for all subsequent communication. DPDSN does not require monitoring individual nodes, making it feasible for WSNs. It formulates the probability of success and failure of DPDSN in the presence of malicious nodes that drop packets. This is approach compared with existing techniques. This analysis found that the overhead of DPDSN is at most O (N) for a two-dimensional grid network of N nodes. The simulations show that the overhead of DPDSN for a WSN with 100 nodes is less than 3% of energy consumed on route discovery when using DSR or Directed Diffusion routing protocols.

Finding of compromised paths

The process of finding an alternate path is embedded in the route discovery phase of routing protocols like DSR and Directed Diffusion. It is assumed that source and destination nodes are trustworthy. Ideally, the alternate path does not have any node in common with the original path.

```
Detect_compromised_path(s, d)
    begin
        Get $n_s$, $n_r$.
        while (TRUE)
            if $n_s - n_r > 0$ then
                Guess that packets are being dropped by
                malicious nodes on the source-to-destination
                path.
                return TRUE
            else
                return FALSE
            Wait till next verification cycle.
    end
```

Fig.1.7 Detection of a Packet-Dropping Path

Embed alternate path discovery in route discovery

A straightforward solution is to perform route discovery using DSR and mark the edges of the original path. It incurs significant cost due to flooding. A better heuristic approach would be to keep two route requests at every node when a node receives multiple route requests one of the route requests is used for establishing the path and second one will be used for alternate path.

Advantage

1. Continuous monitoring of each and every node is not feasible for resource constrained WSNs especially when extending lifetime is the main goal in the design of WSNs. DPDSN avoids continuous monitoring of every node.

Disadvantage

1. DPDSN succeeds whenever an alternate path does not have any malicious nodes that drop packets.

1.6.2 Mitigating Routing Misbehavior in Mobile Ad Hoc Networks

There are two techniques to improve throughput in an ad hoc network in the presence of nodes that agree to forward the packets but fail to do so. The nodes can categorise based upon their dynamically measured behaviour. It introduces two extensions to Dynamic Source Routing (DSR) to mitigate the effects of routing misbehaviour: the watchdog and the pathratcr (Marti 2000). A **watchdog** that identifies misbehaving nodes and a **pathrater** that helps routing protocols avoid these nodes.

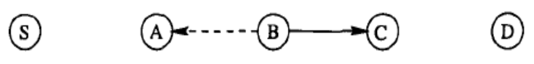


Figure 1.8 Watchdog Mechanism

The watchdog method detects misbehaving nodes. Figure 1.8 illustrates how the watchdog works. Suppose there exists a path from node S to D through intermediate nodes A, B, and C. Node A cannot transmit all the way to node C, but it can listen in on node B's traffic. Thus, when A transmits a packet for B to forward to C, A can often tell if B transmits the packet. If encryption is not performed separately for each link, which be expensive, then A can also tell if B has tampered with the payload or the header. The watchdog is implemented by maintaining a buffer of recently sent packets and comparing each overheard packet with the packet in the buffer to see if there is a match. If so, the packet in the buffer is removed and forgotten by the watchdog, since it has been forwarded on.

When B forwards a packet from S toward D through C, A can overhear B's transmission and can verify that B has attempted to pass the packet to C. The solid line represents the intended direction of the packet sent by B to C, while the dashed line indicates that A is within transmission range of B and can overhear the packet transfer. If a packet has remained in the buffer for longer than a certain timeout, the watchdog increments a failure tally for the node responsible for forwarding on the packet. If the tally exceeds a certain threshold bandwidth, it determines that the node is misbehaving and sends a message to the source notifying it of the misbehaving node. Each node runs pathrater in the network, combines knowledge of misbehaving nodes with link reliability and data to choose the route most likely to be reliable. Each node maintains a rating for every other node it knows about in the network. It calculates a path metric (Marti 2000) by averaging the node ratings in the path.

Advantages

1. DSR with the watchdog has the advantage that it can detect misbehaviour at the forwarding level and not just the link level.
2. The two techniques to increase the throughput by 17% in presence of 40% misbehaving nodes, while increasing the percentage of overhead transmissions from the standard routing protocol's 9% to 17%.
3. During extreme mobility, watchdog and pathrater can increase network throughput by 27%, while increasing the overhead transmissions from the standard routing protocol's 12% to 24%.

Disadvantages

1. Watchdog’s weaknesses are that it might not detect a misbehaving node in the presence of ambiguous collisions, receiver collisions, limited transmission power, false misbehaviour, collusion and partial dropping.
2. The watchdog method requires nodes to buffer the packets and operate in the promiscuous mode, the storage overhead and energy consumption may not be affordable for sensor nodes.

1.6.3 Statistical En-route filtering of Injected False Data

Statistical En-route Filtering (SEF) mechanism (Ye 2004) can detect and drop those false reports. SEF requires that each sensing report be validated by multiple keyed message authentication codes (MACs), each generated by a node that detects the same event. As the report is forwarded, each node along the way verifies the correctness of the MACs probabilistically and drops those with invalid MACs at earliest points. The sink further filters out remaining false reports that escape the en-route filtering. In SEF there is a global key pool. The sink has the knowledge of the entire pool. Each sensor stores a small number of keys that are drawn in a randomized fashion from the global key pool before deployment. Once a stimulus appears in the field, multiple detecting nodes elect a Centre-of-Stimulus (CoS) node that generates the report. Each detecting node produces a keyed MAC for the report using one of its stored keys.

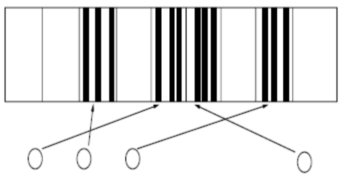


Figure 1.9 Global Key Pool

A **Bloom filter** is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. False positive retrieval results are possible, but false negatives are not; i.e. a query returns either "inside set (may be wrong)" or "definitely not in set". Elements be added to the set, but not removed (though this can address with a counting filter). The more elements that are added to the set, the larger the probability of false positives. Using the Bloom filter, instead of a list of MACs, greatly reduces the packet size. These multiple MACs collectively act as the proof that a report is legitimate. A report with an insufficient number of MACs will not be forwarded. An example of a global key pool(Figure 1.9) with n = 9 partitions and 4 nodes, each of which has k = 3 keys randomly selected from one partition. In a real system, k, n may be much larger.

The sink serves as the final goal-keeper for the system. When it receives reports about an event, the sink verifies every MAC carried in the report because it has complete knowledge of the global key pool.

Advantage

1. With an overhead of 14 bytes per report, SEF is able to drop 80-90% injected false reports by a compromised node within 10 forwarding hops, and reduce energy consumption by 50% or more in many cases

Disadvantage

1. SEF also does not address the issues of how to identify compromised nodes or revoke compromised keys. For identification, neighbour nodes may overhear the channel to detect unusual activities of compromised nodes such as high traffic volume and notify the sink. After the nodes are identified, the user may deploy new nodes and the sink could flood instructions to revoke compromised keys and propagate new ones

1.6.4 An Interleaved Hop-by-Hop Authentication Scheme

An interleaved hop-by-hop authentication scheme (Zhu 2004) that guarantees that the base station will detect any injected false data packets when no more than a certain number t nodes are compromised. Further, our scheme provides an upper bound B for the number of hops that a false data packet could be forwarded before it is detected and dropped, given that there are up to t colluding compromised nodes. The scheme involves the following five phases.

1. In the node initialization and deployment phase, the key server loads every node with a unique id, as well as necessary keying materials that allow the node to establish pairwise keys with other nodes. After deployment, a node first establishes a one-hop pairwise key with each of its neighbours.

2. In the association discovery phase, a node discovers the ids of its associated nodes. This process may be initiated by the base station periodically, or by a node that detects the failure of a neighbour node.

3. In the report endorsement phase, t + 1 node generate a report collaboratively when they detect the occurrence of an event of interest. More specifically, every participating node computes two MACs over the event, one using its key shared with the BS, and the other using its pair wise key shared with its upper associated node. Then it sends the MACs to its cluster head. The cluster head collects MACs from all the participating nodes, wraps them into a report, and then forwards the report towards BS.

4. In the en-route filtering phase, every forwarding node verifies the MAC computed by its lower association node, and then removes that MAC from the received report. If the verification succeeds it then computes and attaches a new MAC based on its pair wise key shared with its upper associated node. Finally, it forwards the report to the next node towards the BS.

5. In the base station verification phase, the BS verifies the report after receiving it.

Advantage

1. Our scheme attempts to filter out false data packets injected into the network by compromised nodes before they reach the base station, thus saving the energy for relaying them.

Disadvantage

1. The number of hops before an injected data packet is detected and discarded should be as small as possible.

1.6.5 Catching "Moles" in Sensor Networks

False data injection is a severe attack that compromised sensor nodes ("moles"') can launch. These moles inject large amount of bogus traffic that can lead to application failures and exhausted network resources.

Probabilistic Nested Marking (PNM) Scheme

A Probabilistic Nested Marking (PNM) scheme (Ye 2007) that is secure against such colluding attacks. No matter how colluding moles manipulate the marks, PNM can always locate them one by one. Here proved that nested marking is both sufficient and necessary to resist colluding attacks. Here it locates such moles within the framework of packet marking, when forwarding moles collude with source moles to manipulate the marks. The packet marking is used to discover the true origin of packets: A node marks its identity in the packets it forwards. By collecting such marks, the sink can infer the route, thus the origin location of the traffic. Due to the nested marking, any tampering with the previous IDS, or MACs, or their order, will make the MAC invalid. Probabilistic marking requires an additional feature, anonymity of IDS, to defeat selective dropping attacks.

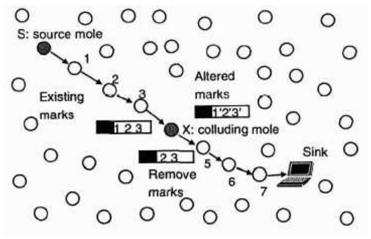


Figure 1.10 False Data Injection

The challenge for an effective marking scheme is, a colluding mole X along the forwarding path may tamper the marks arbitrarily (see Figure 1.10). It can hide both its location and the source mole's location, or even trick the sink trace to innocent nodes. Hiding their locations allows continuous injection without being punished. This is needed for the injection to cause significant damage. Leaking any of their locations will lead to punishment such as network isolation or physical removal. Tricking the sink trace to innocent nodes is extra bonus: the sink may punish these nodes, thus denying legitimate resource and service to itself. Mole S and X work together to cover their traces for injecting attack**.** S injects bogus reports. X receives a packet with nodes 1, 2, 3**'**s marks. X may manipulate the marks in various ways, such as altering these marks to l ', 2', 3'**,** or remove the mark of node 1**.** The moles' goal is to hide their locations, or lead the sink trace to innocent nodes.

Advantage

1. Probabilistic Nested Marking is the first work that can locate moles despite colluding attack. Combined with physical removal or network isolation, PNM can used to actively fight back
2. PNM also has fast-trace back: within about 50 packets, it can track down a mole up to 20 hops away from the sink. This virtually prevents any effective data injection attack: moles will be caught before they have injected any meaningful amount of bogus traffic.
3. While public-key cryptography can implement in such low-end devices, it is too expensive in energy consumption. Thus considered only efficient symmetric cryptography (e.g., secure hash functions) in our design.

Disadvantage

1. In PNM scheme, modified packets should not be filtered out en-route because they should be used as evidence to infer packet modifiers; hence, it cannot be used together with existing packet filtering schemes.

1.6.6 Catching Packet Droppers and Modifiers in Wireless Sensor Networks

Packet dropping and modification are common attacks that can launch by an adversary to disrupt communication in wireless multi-hop sensor networks. Many schemes have been proposed to mitigate or tolerate such attacks but very few can effectively and efficiently identify the intruders. To address this problem, a simple effective scheme (Chuang Wang 2012), this can identify misbehaving forwarders that drop or modify packets. In this scheme, a routing tree rooted at the sink is first established. When sensor data is transmitted along the tree structure towards the sink, each packet sender or forwarder adds a small number of extra bits, which is called packet marks, to the packet. The format of the small packet marks is deliberately designed such that the sink can obtain very useful information from the marks. Specifically, based on the packet marks, the sink can figure out the dropping ratio associated with every sensor node, and then runs our proposed node categorization algorithm to identify nodes that are droppers/modifiers for sure or are suspicious droppers/modifiers. As the tree structure dynamically changes every time interval, behaviours of sensor nodes can observe in a large variety of scenarios. As the information of node behaviours has been accumulated, the sink periodically runs our proposed heuristic ranking algorithms to identify most likely bad nodes from suspiciously bad nodes. This way, most of the bad nodes can gradually identify with small false positive.

Packet Sending and Forwarding

Each node maintains a counter Cp which keeps track of the number of packets that it has sent so far. When a sensor node u has a data item D to report, it composes and sends the following packet to its parent node Pu. Cp MOD Ns is the sequence number of the packet. Ru (0 ≤ Ru ≤ Np − 1) is a random number picked by node u during the system initialization phase, and Ru is attached to the packet to enable the sink to find out the path along which the packet is forwarded. {X}Y represents the result of encrypting X using key Y. Padding padu,0 and padu,1 are added to make all packets equal in length, such that forwarding nodes cannot tell packet sources based on packet length.

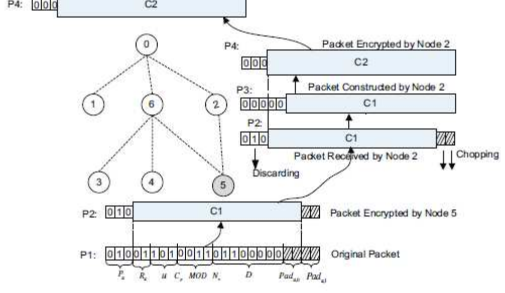

Packet Received by Node 0
P4:
C2
Packet Encrypted by Node 2
P3:
Packet Constructed by Node 2
C1
P2:
Packet Received by Node 2
Discarding
Chopping
Packet Encrypted by Node 5
P1:
Original Packet

Figure 1.11 Packet Sending, Forwarding

Figure 1.11 shows an example sensor network with 7 nodes, nodes 0−6. Node ID is represented by 3 bits. Suppose the maximum packet sequence number Ns are 16 and 4 bits are used to represent the counter Cp. Np, the maximum number of parents that each sensor node should record during the tree establishment, is 4. Assume that the length of sensory data LD is 8 bits. The figure illustrate the following procedure: node 5, which is 2 hops away from the sink, generates sensory data 96; the data is sent to the sink node 0. Assume data from node 5 follows path 5->2->0 and Cp = 3. Node 5 constructs packet .The plain-text of the packet is shown in the figure as P1.Specifically, Pu = 2(010), Ru = 1(01), u = 5(101), Cp =3(0011), and D =96(01100000).The cipher-text is represented by C1 and the encrypted packet P2 is constructed accordingly. P2 is sent to node 2. When node 2 receives packet P2, it first chops off the rightmost logNp bits, which are 2 bits of the paddings. Next, node 2 constructs packet P3 by adding its parent ID and the random number R2 to the front of cipher-text C1. Note that the packet length is kept the same since the rightmost 2 bits are chopped off, and a random number R2 with 2 bits is added. Next, node 2 uses its secret key K2 to encrypt information {R2, C1} in packet P3 and generates packet P4. P4 is then sent to the sink. After the sink receives the packet P4 from its children, the sink tries to figure out the sender. The sink tries to decrypt the cipher-text C2 by using its children's secret keys one by one. The sink finds that the packet is from node 2 after C2 is decrypted by using K2. The sink also recovers the decrypted C2 which does not start with {R2, 2}.The sink concludes that node 2 is an intermediate node. It continues this process and finds out the source of the data is node 5.

Advantages

1. Low communication and energy overheads
2. Compatible with existing false packet filtering schemes

CHAPTER 2

PROJECT DESCRIPTION

2.1 PROBLEM DEFINITION

Wireless Sensor networks consist of large number of small sensor nodes which has limited computation capacity, restricted memory space, limited power resource, and short-rage radio communication device. With a widespread deployment of these devices, one can precisely monitor the environment. Basically, sensor networks are application dependent and sensor nodes monitor the environment, detect events of interest, produce data, and collaborate in forwarding the data toward a sink, which could be a gateway, base station, storage node, or querying user.For example wireless sensor networks are used in environmental monitoring, military surveillance, etc. Farmers can use sensor network to collect information about how much water is needed, what pesticide to use, what fertilizer to be used. A soldier in a battle field may need to know the location of the nearest enemy tank.

Wasteful energy consumption is due to idle listening in the network, retransmitting due to packet collisions, overhearing and generating or handling control packets. Nodes in sensor network have very limited computational resources and they are energy constrained. Sensor nodes generate a large volume of data and they must be processed in order to respond to the queries given. Processing of these queries has to be carried out energy efficiently. Sensor nodes are often battery powered and it is very difficult to change batteries when they get down. A sensor network is often deployed in an unattended and hostile environment to perform the monitoring and data collection tasks. When it is deployed in such environment, it lacks physical protection and is subject to node compromise. After compromising one or multiple sensor nodes, an adversary may lunch various attacks to disrupt the in-network communication. This work deals with two common attacks, dropping packets and modifying packets which can launch by compromised nodes.

2.2 ROUTING PROCESS

The main objective is to identify compromised nodes and to prolong network lifetime via an energy efficient routing algorithm and contributions are listed as below:

(1) Given the source to sink node distance d, the optimal multi-hop number and the corresponding individual distance d can determined based on the theoretical analysis of energy consumption under time based traffic model.

(2) Based on (1), a Routing algorithm is proposed which consists of route setup and route maintenance phases. The distance factor is treated as the first parameter during the routing process and the residual energy factor is the second parameter to be considered. The algorithm can balance energy consumption for all sensor nodes and consequently prolong the network lifetime.

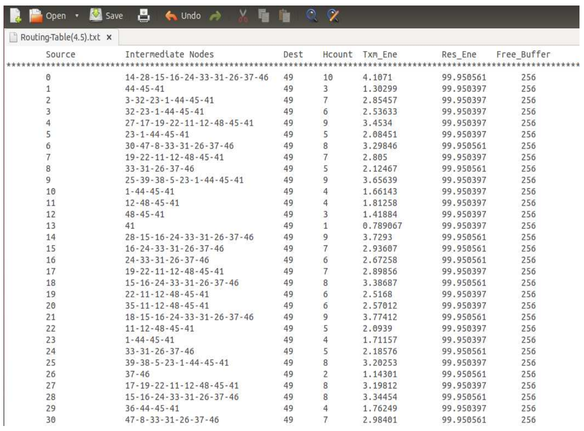
Distance Calculation

There are N nodes randomly scattered in a two dimensional square field [X, Y]. There exists a link E (i, j) between node I and node j if the Euclidean distance d (i, j) is not larger than the radio transmission radius *R*, namely d (i, j) ≤ R. In the Euclidean plane, if p = (p_1, p_2) and q = (q_1, q_2) then the distance is given $\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2}$.

Routing Tables

Sensor node has two tables. One is the routing table which contains information like source node, previous and next hop node, etc. The other table is the neighbour table which contains neighbours information like distance between them, distance to the sink node, residual energy, node degree, etc. Thus, each node can make intelligent decisions about the next hop based on this algorithm and the algorithm is easy to implement for practical engineering applications.

The routing table is generated by source whenever it takes new path to destination. The routing will include residual energy. Before routing a packet through a particular path, source will have a knowledge about intermediate nodes, hop count, free buffer space, etc. Then energy of each node in path is calculated. Figure 2.1 shows routing table created for first path.



```
      Source          Intermediate Nodes              Dest    Hcount  Txm_Ene         Res_Ene     Free_Buffer
*****************************************************************************************************************
      0               14-28-15-16-24-33-31-26-37-46   49      10      4.1071          99.950561       256
      1               44-45-41                        49      3       1.30299         99.950397       256
      2               3-32-23-1-44-45-41              49      7       2.85457         99.950397       256
      3               32-23-1-44-45-41                49      6       2.53633         99.950397       256
      4               27-17-19-22-11-12-48-45-41      49      9       3.4534          99.950397       256
      5               23-1-44-45-41                   49      5       2.08451         99.950397       256
      6               30-47-8-33-31-26-37-46          49      8       3.29846         99.950561       256
      7               19-22-11-12-48-45-41            49      7       2.805           99.950397       256
      8               33-31-26-37-46                  49      5       2.12467         99.950561       256
      9               25-39-38-5-23-1-44-45-41        49      9       3.65639         99.950397       256
      10              1-44-45-41                      49      4       1.66143         99.950397       256
      11              12-48-45-41                     49      4       1.81258         99.950397       256
      12              48-45-41                        49      3       1.41884         99.950397       256
      13              41                              49      1       0.789067        99.950397       256
      14              28-15-16-24-33-31-26-37-46      49      9       3.7293          99.950561       256
      15              16-24-33-31-26-37-46            49      7       2.93607         99.950561       256
      16              24-33-31-26-37-46               49      6       2.67258         99.950561       256
      17              19-22-11-12-48-45-41            49      7       2.89856         99.950397       256
      18              15-16-24-33-31-26-37-46         49      8       3.38687         99.950561       256
      19              22-11-12-48-45-41               49      6       2.5168          99.950397       256
      20              35-11-12-48-45-41               49      6       2.57012         99.950397       256
      21              18-15-16-24-33-31-26-37-46      49      9       3.77412         99.950561       256
      22              11-12-48-45-41                  49      5       2.0939          99.950397       256
      23              1-44-45-41                      49      4       1.71157         99.950397       256
      24              33-31-26-37-46                  49      5       2.18576         99.950561       256
      25              39-38-5-23-1-44-45-41           49      8       3.20253         99.950397       256
      26              37-46                           49      2       1.14301         99.950561       256
      27              17-19-22-11-12-48-45-41         49      8       3.19812         99.950397       256
      28              15-16-24-33-31-26-37-46         49      8       3.34454         99.950561       256
      29              36-44-45-41                     49      4       1.76249         99.950397       256
      30              47-8-33-31-26-37-46             49      7       2.98401         99.950561       256
```

Figure 2.1 Routing Table

2.3 IDENTIFICATION OF COMPROMISED NODE

Here the compromised node be a dropper or modifier, such nodes are identified by data delivery ratio and energy. Here delay is also calculated to identify the compromised node.

Delay

With the help of routing table, path taken by source node is identified. Source has the knowledge of whole path to reach the destination. And delay of each packet is calculated. When delay is increased, there is a chance for modification. The delay is calculated for each packet form one source node to destination. Here delay is calculated for entire path.

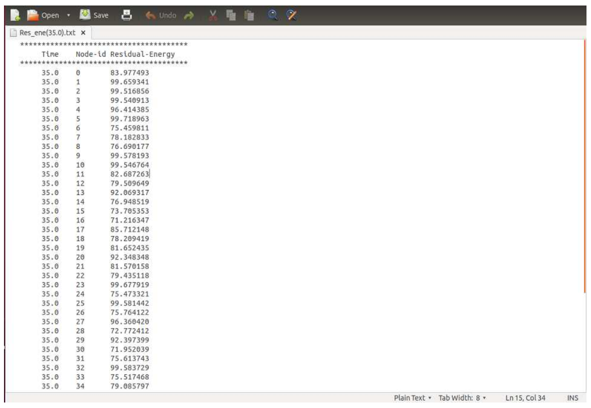
Source will send the packets in a path, if the particular path is delayed, then it will conclude that there is a compromised node. So it will check the routing table and take alternate route to reach the same destination. Again it will check delay to second path. The algorithm is implemented to calculate delay of each packet. The timer value is set to 5ms. At each node the delay is calculated for every 5ms.and energy is also calculated as same for delay.

Algorithm

```
Initialize $N_s$ =0, $N_r$ = 0, $S_t$ =0, $E_t$ =0
Wait until a packet is received
Record PACKET receipt time $E_t$
$S_t = t_1$, $E_t = t_2$, $N_s = 1$, $N_r$ =1
Start ‘DROPPER or MODIFIER TIMER ‘
While ‘DROPPER or MODIFIER TIMER ‘has not expired
    Wait until next PACKET is received or ‘DROPPER or MODIFIER TIMER
    ‘is expired
    If a PACKET i is received
        Record PACKET receipt time $t_i$
        Reset ‘DROPPER or MODIFIER TIMER ‘
    End if
End while
```

Energy

Energy consumption in a sensor node that is due to either useful sources or wasteful sources. Useful energy consumption is due to transmitting or receiving data, processing query request, and forwarding queries or data to neighbouring nodes. To identify the energy loss in each node, the residual energy is calculated.



```
  *************************************
       Time     Node-id Residual-Energy
  *************************************
       35.0     0       83.977493
       35.0     1       99.659341
       35.0     2       99.516856
       35.0     3       99.540913
       35.0     4       96.414385
       35.0     5       99.718963
       35.0     6       75.459811
       35.0     7       78.182833
       35.0     8       76.690177
       35.0     9       99.578193
       35.0     10      99.546764
       35.0     11      82.687263
       35.0     12      79.509649
       35.0     13      92.069317
       35.0     14      76.948519
       35.0     15      73.705353
       35.0     16      71.216347
       35.0     17      85.712148
       35.0     18      78.209419
       35.0     19      81.652435
       35.0     20      92.348348
       35.0     21      81.570158
       35.0     22      79.435118
       35.0     23      99.677919
       35.0     24      75.473321
       35.0     25      99.581442
       35.0     26      75.764122
       35.0     27      96.360420
       35.0     28      72.772412
       35.0     29      92.397399
       35.0     30      71.952039
       35.0     31      75.613743
       35.0     32      99.583729
       35.0     33      75.517468
       35.0     34      79.085797
```

Figure 2.2 Residual Energy

Here routing is used to find the path between each source and destination pair. While finding a path, residual energy is also calculated for each node in the path Figure 2.2 shows residual energy for each node. The energy is calculated at each node in the path for every 5ms. When the energy utilization is more, there will be a chance for modification. When the energy utilization is more in the node, it is declared as modifier.

CHAPTER 3

RESULTS AND ANALSIS

3.1 SOFTWARE DESCRIPTION

NS2 is one of the most popular open source network simulators. The original NS is a discrete event simulator targeted at networking research. First and foremost, NS2 is an object-oriented, discrete event driven network simulator which was originally developed at University of California-Berkely. The programming it uses is C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT). The usage of these two programming language has its reason. The biggest reason is due to the internal characteristics of these two languages. C++ is efficient to implement a design but it is not very easy to be visual and graphically shown. It’s not easy to modify and assemble different components and to change different parameters without a very visual and easy-to- use descriptive language. The event scheduler and the basic network component objects in the data path are written and compiled using C++ to reduce packet and event processing time. OTcl happens to have the feature that C++ lacks. So the combination of these two languages proves to be very effective.

3.2 HARDWARE REQUIREMENTS

Processor	: 3rd generation Intel core i5 – 3210 M
Clock speed	: 2.5 GHz
Hard Disk	: 500 GB
RAM	: 4 GB
Cache Memory	: 3 MB
Monitor	: Color Monitor
Keyboard	: 104Keys
Mouse	: 3Buttons

3.3 SOFTWARE REQUIREMENTS

Operating System	: Ubuntu
Language	: Network Simulator 2.34

3.4 IMPLEMENTATION

The proposed work is implemented using Network Simulator NS2. For the evaluation purpose, 30 or 50 sensor nodes are deployed randomly. The routing operation is performed to identify the path for each source and their destination. The routing table is created and updated periodically. A path is chosen from the routing table. While identifying the path, residual energy is also calculated for each node. The routing table consist of source node, destination node, intermediate nodes, hop count, residual energy and buffer size of each node. The packets are routed through two different paths.

Initially, packets are forwarded through a path and delay and energy calculations are made. The energy and delays calculations are made for each 5ms. Here values obtained for energy and delay are found to high in a particular node. So it will be declared as compromised node. Later an alternate path is taken with help of routing for same source and destination. Again the energy and delay calculations are made for this particular path. The values obtained for energy and delay calculations are found to be normal. So there is no compromised node found in this alternate path. The code is written in awk file and tcl file.

3.5 ANALYSIS

3.5.1 Energy Consumption

In sensor networks, consumed energy is the primary performance measure. Here energy consumption is considered as an important factor to identify the compromised nodes in the network. Below graph (Figure 3.1) compares the energy consumption for path with compromised node and alternate path with no compromised nodes.

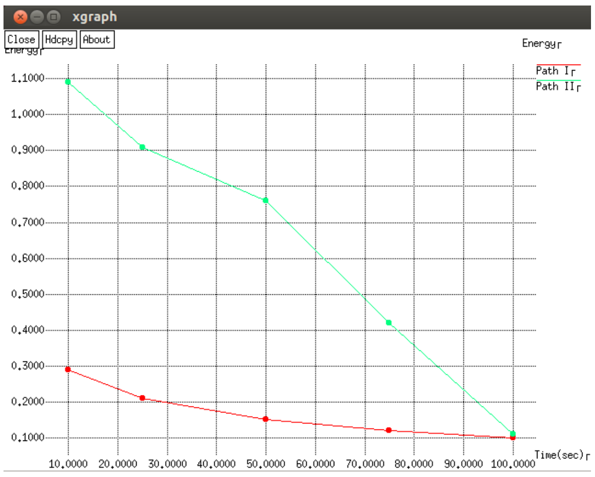


Figure3.1 Energy Consumption

3.5.2 Packets Received

The packets dropped are identified by calculating total number of packets received by sink. The graph below (Figure 3.2) shows packets received with respect to time. When the received packet is less compared to transmitted packets, then it may be dropped.

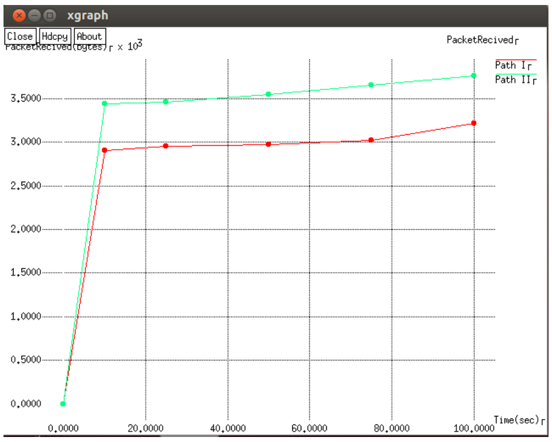


Figure 3.2 Packet Received

3.5.3 Delay

Delay of each packet is calculated. When delay is increased, there is a chance for modification. The delay is calculated for each packet form one node to another. The graph below (Figure 3.3) compares the delay for path with compromised node and alternate path with no compromised nodes.

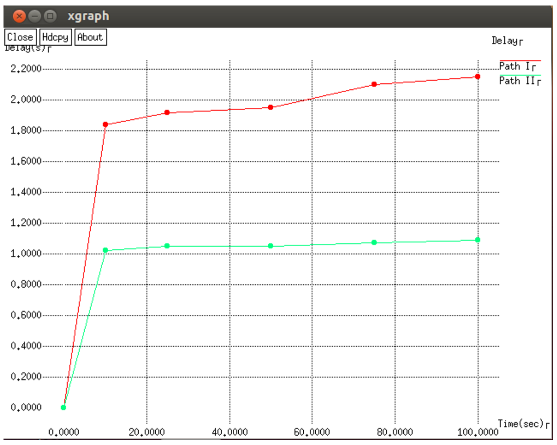
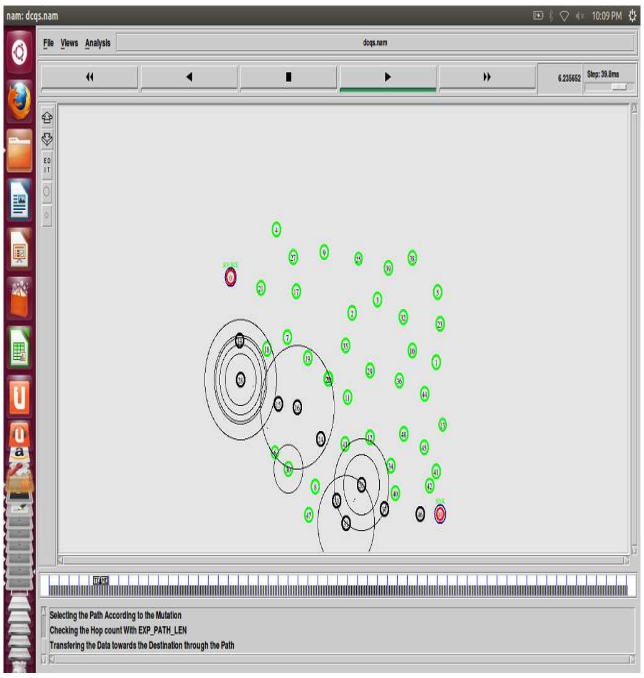


Figure3.3 Delay

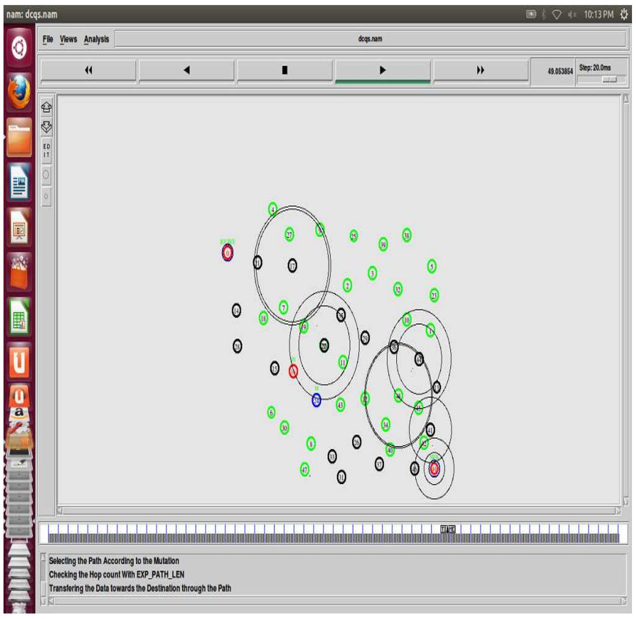
3.6 Snapshots



Selecting the Path According to the Mutation
Checking the Hop count With EXP_PATH_LEN
Transfering the Data towards the Destination through the Path

Figure 3.4 First Path

The packets are transmitted through the first path. The source and sink nodes are labeled. Intermediated nodes are denoted by dark black color from other nodes in the network.



Selecting the Path According to the Mutation
Checking the Hop count With EXP_PATH_LEN
Transfering the Data towards the Destination through the Path

Figure 3.5 Alternate Path

Here the nodes which drop and modify the packets are identified in the first path. The compromised nodes in the first path are labeled, and then source transmits the packets through the alternate path.

3.7 CONCLUSION AND FUTURE ENHANCEMENT

A simple effective method is proposed to identify the packet droppers and modifiers in the network. The routing is performed to identify the path between each source node and their destination, and residual energy is calculated for each node in the network. The routing table is created and updated periodically. The packet will be transmitted through a path chosen from routing table. Delay and energy calculations are made for each path to identify the compromised nodes. Analysis and simulations are conducted.

In future, this is anticipated to extend better scalability and improve to identify the modifier by next honest node and alternate path will be chose by intermediate nodes.

APPENDIX

SOURCE CODE

```
Packet.tcl
# Environmental Settings
 set val(chan)          Channel/WirelessChannel        ;# channel type
 set val(prop)          Propagation/TwoRayGround       ;# radio-propagation model
 set val(ant)           Antenna/OmniAntenna            ;# Antenna type
 set val(ll)            LL                             ;# Link layer type
 set val(ifq)          Queue/DropTail/PriQueue         ;# Interface queue type
 set val(ifqlen)       256                              ;# max packet in ifq
 set val(netif)        Phy/WirelessPhy                 ;# network interface type
 set val(mac)         Mac/802_11                        ;# MAC type
 set val(rp)            DSDV                           ;# Routing Protocol
 set val(nn)           50                              ;# number of mobilenodes
 set val(x)           1500
 set val(y)           1500
 set opt(energymodel)  EnergyModel                      ;# Energy model
 set opt(radiomodel)   RadioModel                       ;# Radio model
 set opt(initialenergy)  100                             ;# Initial energy Joules
 set val(sc)            setdest-100.tr
 set r                  250
 set s_thres            250
 set qlen               256
 set beta               2
# Default parameters
 Agent/TCP/RFC793edu set rto_ 250              ; #250 m Transmission range
 Agent/TCP set packetSize_ 1024                ; # Packet size (data +
                                                        overhead)
 Phy/WirelessPhy set Pt_ 0.015                 ; # Transmission Power
 Phy/WirelessPhy set RXThresh_ 2.025e-12       ; #500m radius.Receving
                                               Threshold
 Phy/WirelessPhy set CSThresh_ [expr 0.9*[Phy/WirelessPhy set RXThresh_]]
                                                       ; # Carrier Sence Threshold
 Phy/WirelessPhy set CPThresh_ 10.0            ; # Carrier Power
 Phy/WirelessPhy set bandwidth_ 2e6            ; # Bandwidth
 Phy/WirelessPhy set freq_ 914e+6              ; # Frequency

# Simulator Object Creation
set ns_ [new Simulator]
# Trace File to record all the Events
set f [open Link-mdp.tr w]
$ns_ trace-all $f
$ns_ use-newtrace
# NAM Window creation
set namtrace [open dcqs.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
# Topology Creation
set topo [new Topography]
$topo load_flatgrid 1500 1500
# General Operational Director
create-god $val(nn)
# Node Configuration
$ns_ node-config  -adhocRouting $val(rp) \
                        -llType $val(ll) \
                  -macType $val(mac) \
                  -ifqType $val(ifq) \
                  -ifqLen $val(ifqlen) \
                  -antType $val(ant) \
                  -propType $val(prop) \
                  -phyType $val(netif) \
                  -channelType $val(chan) \
                  -topoInstance $topo \
                  -agentTrace ON \
                  -routerTrace ON \
                  -macTrace ON \
                  -movementTrace ON \
                  -idlePower 0.012 \
                        -rxPower 1.5 \
                        -txPower 2.0 \
                        -sleepPower 0.00015 \
              -initialEnergy $opt(initialenergy) \
                  -energyModel $opt(energymodel)
# Node Creation
set god_ [create-god $val(nn)]
for {set i 0} {$i < $val(nn) } {incr i} {
set node_($i) [$ns_ node]
$node_($i) random-motion 0
$god_ new_node $node_($i)
}
for {set i 0} {$i < $val(nn)} {incr i} {
          $ns_ initial_node_pos $node_($i) 50
          $node_($i) set X_ 750.0
          $node_($i) set Y_ 0.0
          $node_($i) set Z_ 0.0
          $node_($i) color black
}

source $val(sc)

# subclass Agent/MessagePassing to make it do flooding
Class Agent/MessagePassing/Flooding -superclass Agent/MessagePassing
Agent/MessagePassing/Flooding instproc recv {source sport size data} {
    $self instvar messages_seen node_
    global ns BROADCAST_ADDR
    # extract message ID from message
    set message_id [lindex [split $data ":"] 0]
    #puts "\nNode [$node_ node-addr] got message $message_id\n"
    if {[lsearch $messages_seen $message_id] == -1} {
          lappend messages_seen $message_id
          $self sendto $size $data $BROADCAST_ADDR $sport
    } else {
           }}

Agent/MessagePassing/Flooding instproc send_message {size message_id data port} {
    $self instvar messages_seen node_
    global ns MESSAGE_PORT BROADCAST_ADDR
    lappend messages_seen $message_id
    $self sendto $size "$message_id:$data" $BROADCAST_ADDR $port
}
set t [$ns_ now]
for {set i 0} {$i<50} {incr i} {
set sink$i [new Agent/LossMonitor]
}
$ns_ attach-agent $node_(0) $sink0
$ns_ attach-agent $node_(1) $sink1
$ns_ attach-agent $node_(2) $sink2
$ns_ attach-agent $node_(3) $sink3
$ns_ attach-agent $node_(4) $sink4
$ns_ attach-agent $node_(5) $sink5
$ns_ attach-agent $node_(6) $sink6
$ns_ attach-agent $node_(7) $sink7
$ns_ attach-agent $node_(8) $sink8
$ns_ attach-agent $node_(9) $sink9
$ns_ attach-agent $node_(10) $sink10
$ns_ attach-agent $node_(11) $sink11
$ns_ attach-agent $node_(12) $sink12
$ns_ attach-agent $node_(13) $sink13
$ns_ attach-agent $node_(14) $sink14
$ns_ attach-agent $node_(15) $sink15
$ns_ attach-agent $node_(16) $sink16
$ns_ attach-agent $node_(17) $sink17
$ns_ attach-agent $node_(18) $sink18
$ns_ attach-agent $node_(19) $sink19
$ns_ attach-agent $node_(20) $sink20
$ns_ attach-agent $node_(21) $sink21
$ns_ attach-agent $node_(22) $sink22
$ns_ attach-agent $node_(23) $sink23
$ns_ attach-agent $node_(24) $sink24
$ns_ attach-agent $node_(25) $sink25
$ns_ attach-agent $node_(26) $sink26
$ns_ attach-agent $node_(27) $sink27
$ns_ attach-agent $node_(28) $sink28
$ns_ attach-agent $node_(29) $sink29
$ns_ attach-agent $node_(30) $sink30
$ns_ attach-agent $node_(31) $sink31
$ns_ attach-agent $node_(32) $sink32
$ns_ attach-agent $node_(33) $sink33
$ns_ attach-agent $node_(34) $sink34
$ns_ attach-agent $node_(35) $sink35
$ns_ attach-agent $node_(36) $sink36
$ns_ attach-agent $node_(37) $sink37
$ns_ attach-agent $node_(38) $sink38
$ns_ attach-agent $node_(39) $sink39
$ns_ attach-agent $node_(40) $sink40
$ns_ attach-agent $node_(41) $sink41
$ns_ attach-agent $node_(42) $sink42
$ns_ attach-agent $node_(43) $sink43
$ns_ attach-agent $node_(44) $sink44
$ns_ attach-agent $node_(45) $sink45
$ns_ attach-agent $node_(46) $sink46
$ns_ attach-agent $node_(47) $sink47
$ns_ attach-agent $node_(48) $sink48
$ns_ attach-agent $node_(49) $sink49
proc attach-CBtraffic { node sink } {
   #Get an instance of the simulator
   set ns_ [Simulator instance]
   #Create a CBR  agent and attach it to the node
   set udp [new Agent/UDP]
   $ns_ attach-agent $node $udp
   set cbr [new Application/Traffic/CBR]
   $cbr attach-agent $udp
   $cbr set packetSize_ 256  ;#sub packet size
   $cbr set interval_ 0.048
   #Attach CBR source to sink;
   $ns_ connect $udp $sink
   return $cbr
  }
proc attach-CBR-traffic { node sink } {
   #Get an instance of the simulator
   set ns_ [Simulator instance]
   set udp [new Agent/UDP]
   $ns_ attach-agent $node $udp
   #Create a CBR  agent and attach it to the node
   set cbr [new Application/Traffic/CBR]
   $cbr attach-agent $udp
   $cbr set packetSize_ 256  ;#sub packet size
   $cbr set interval_ 0.048
   $cbr set random_ 1
   $cbr set maxpkts_ 5000
   #Attach CBR source to sink;
   $ns_ connect $udp $sink
   return $cbr
  }
#                    ~~~~~~~~~~~~~~~~~~~~For                    route
discovery~~~~~~~~~~~~~~~~~``~~~`~~~~~~
set init [attach-CBR-traffic $node_(0) $sink21]
set init1 [attach-CBR-traffic $node_(0) $sink21]
set init2 [attach-CBR-traffic $node_(0) $sink44]
$ns_ at 3.0 "$init start"
$ns_ at 3.01 "$init stop"
$ns_ at 37.8 "$init1 start"
$ns_ at 37.801 "$init1 stop"
$ns_ at 35.8 "$init2 start"
$ns_ at 35.801 "$init2 stop"

set dis [open E-Distance.txt w]
puts$dis
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~"
puts $dis "\tsource-Node\tDest-Node\tSX-cor\tSY-Cor\tE-Distance(d)"
puts$dis
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~"
close $dis
set nbr [open Neighbour w]
puts$nbr
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~”
puts $nbr "\tsource-Node\tNeighbour-Node\tH-Distance(d)"
puts$nbr
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~"
close $nbr
#~~~~~~~~~~~~~ For Calculation of Euclidean distance~~~~~~~~~~~~~~~~~~~~
proc distance { n1 n2 nd1 nd2 fl} {
global r
set dis [open E-Distance.txt a]
set nbr [open Neighbour a]
set x1 [expr int([$n1 set X_])]
set y1 [expr int([$n1 set Y_])]
set x2 [expr int([$n2 set X_])]
set y2 [expr int([$n2 set Y_])]
set d [expr int(sqrt(pow(($x2-$x1),2)+pow(($y2-$y1),2)))]
if {$nd2>=$nd1} {
if {$fl == 49} {
puts $dis "\t$nd1\t\t$nd2\t\t$x1\t$y1\t$d"
}}
if {$d<250} {
if {$nd2!=$nd1} {
puts $nbr "\t$nd1\t\t$nd2\t\t$d"
}}
close $dis
close $nbr
}
#~~~~~~~~~For Calculating Energy~~~~~~~~~~~~~~~~~~
proc energy {stnode etnode stime etime} {
set etp [open etmp w]
puts $etp "$stnode $etnode $stime $etime"
close $etp
exec awk -f energy.awk etmp Link-mdp.tr
}
# ~~~~~~~~~~~~~ For  Routing~~~~~~~~~~~~~~~~~~~~~~~~~~~
proc routing { tme stnode etnode snk qlen s_thres lc beta stme etme} {
set rtmp [open rtmp w]
puts $rtmp "$tme $stnode $etnode $snk $qlen $s_thres $lc $beta $stme $etme"
close $rtmp
exec awk -f routing.awk rtmp Res_ene($tme).txt E-Distance.txt
set rt [open route.txt r]
set rout [read $rt]
puts "$rout"
close $rt
}
```

```
#                    ~~~~~~~~~~~~~~~~~~~For                    function                    Calling
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
for {set i 0} {$i<=49} {incr i} {
for {set j 0} {$j<=49} {incr j} {
$ns_ at 3.5 "distance $node_($i) $node_($j) $i $j 49"
}}
$ns_ at 4.51 "energy 0 49 0 4.5"
$ns_ at 36.0 "energy 0 49 5 35.0"
$ns_ at 4.6 "routing 4.5 0 48 49 $qlen $s_thres $lc $beta 5.0 35.0"
$ns_ at 36.6 "routing 35.0 0 48 49 $qlen $s_thres $lc $beta 40.0 70.0"
set src Trans.tcl
$ns_ at 4.7 "source $src"
$ns_ at 36.7 "source $src"

proc fenergy { timee } {
global ns_ t i sink0
set i [expr $i+5]
set time 5
set now [$ns_ now]
for {set i 0} {$i <=49} {incr i} {
set tmp [open temp1.tr w]
puts $tmp "[expr $timee-5] $timee $i"
close $tmp
exec awk -f renergy.awk temp1.tr Link-mdp.tr
}
set tme [expr $now+$time]
$ns_ at $tme "energy $tme"
}
set i 0
set t 4
# For energy of each node
proc nenergy { t } {
global ns_
set cnt 0
set tot 0
set tm [$ns_ now]
set int 5
set rt [open route.txt r]
while {!([eof $rt])} {
set hp [gets $rt]
set tmp [open temp1.tr w]
puts $tmp "$t $hp"
close $tmp
set cnt [expr $cnt+1]
exec awk -f nenergy.awk temp1.tr Link-mdp.tr
```

```
set tmp2 [open temp2.tr r]
set e [gets $tmp2]
set tene($cnt) $e
close $tmp2
set en [open ene($hp).tr a]
puts $en "$t\t$e"
close $en
puts "Node - $hp\tTime - $t\tEnergy - $e"
}
for {set k 1} {$k<=$cnt} {incr k} {
set tot [expr $tene($k)+$tot]
}
set avg [expr $tot/$cnt]
set pe_ene [open P-PriResEnergy.xg a]
set se_ene [open P-SecResEnergy.xg a]
if {$tm <=35} {
puts $pe_ene "$t $avg"
} elseif {$tm >=40 && $tm<=70 }  {
puts $se_ene "$t $avg"
}
close $pe_ene
close $se_ene
set tim [expr int($tm+$int)]
if {$tim <=70 } {
$ns_ at $tim "nenergy $tim"
} }
# For Delay Calculation
set pdely [open pridelay w]
set sdely [open secdelay w]
close $pdely
close $sdely
#set tm 35
proc delay { start end send rec id} {
global ns_ dely rno
set tm $start
set t [open tdly.tr w]
puts $t "$start $end $send $rec"
close $t
exec awk -f delay.awk tdly.tr Link-mdp.tr
set pdely [open pridelay a]
set sdely [open secdelay a]
set dly [open tmp$send r]
set value [gets $dly]
if {$tm<=30 && $id==1} {
puts $pdely "$value"
```

```
} elseif {$tm>=30 && $tm<=60 && $id==2} {
puts $sdely "$value"
}
close $pdely
close $sdely
set tm [expr $tm+5]
if {$tm<=30 && $id==1 } {
delay $tm [expr $tm+5] $send $rec $id
} elseif {$tm>=40 && $tm<70 && $id==2} {
delay $tm [expr $tm+5] $send $rec $id
} else {
return
}}
set ptp [open P-PriThroughput.xg w]
set stp [open P-SecThroughput.xg w]
set pdp [open P-PriDrop.xg w]
set sdp [open P-SecDrop.xg w]
set ppdr [open P-PriPDR.xg w]
set spdr [open P-SecPDR.xg w]
puts $spdr "35 0"
set pdly [open P-PriDelay.xg w]
set sdly [open P-SecDelay.xg w]
set pe_ene [open P-PriResEnergy.xg w]
set se_ene [open P-SecResEnergy.xg w]
close $pe_ene
close $se_ene
close $pdly
close $sdly
proc record { } {
global ns_ sink0 sink14 sink28 sink15 sink16 sink24 sink33 sink31 sink26 sink37
sink46 sink49 ptp pdp ppdr
set t [$ns_ now]
set itval 5.0
set r0 [$sink0 set npkts_]
set r14 [$sink14 set npkts_]
set r28 [$sink28 set npkts_]
set r15 [$sink15 set npkts_]
set r16 [$sink16 set npkts_]
set r24 [$sink24 set npkts_]
set r33 [$sink33 set npkts_]
set r31 [$sink31 set npkts_]
set r26 [$sink26 set npkts_]
set r37 [$sink37 set npkts_]
set r46 [$sink46 set npkts_]
set r49 [$sink49 set npkts_]
```

```
set                                        rec                                        [expr
($r0+$r14+$r28+$r15+$r16+$r24+$r33+$r31+$r26+$r37+$r46+$r49)/12]
set b0 [$sink0 set bytes_]
set b14 [$sink14 set bytes_]
set b28 [$sink28 set bytes_]
set b15 [$sink15 set bytes_]
set b16 [$sink16 set bytes_]
set b24 [$sink24 set bytes_]
set b33 [$sink33 set bytes_]
set b31 [$sink31 set bytes_]
set b26 [$sink26 set bytes_]
set b37 [$sink37 set bytes_]
set b46 [$sink46 set bytes_]
set b49 [$sink49 set bytes_]
setbyt[expr($b0+$b14+$b28+$b15+$b16+$b24+$b33+$b31+$b26+$b37+$b46+$b
4)/12]
set pdr 0
if {$rec!=0} {
set pdr [expr ($rec+0.0)/($rec+$los)]
}
set tput [expr ($byt*8.0)/($itval*1000000)]
puts $ptp "$t\t$tput"
puts $pdp "$t\t$los"
puts $ppdr "$t\t$pdr"
set inter [expr $t+5.0]
if {$inter <=35} {
$ns_ at $inter "record"
}
$sink0 set bytes_ 0
$sink14 set bytes_ 0
$sink28 set bytes_ 0
$sink15 set bytes_ 0
$sink16 set bytes_ 0
$sink24 set bytes_ 0
$sink33 set bytes_ 0
$sink31 set bytes_ 0
$sink26 set bytes_ 0
$sink37 set bytes_ 0
$sink46 set bytes_ 0
$sink49 set bytes_ 0
}
proc record1 { } {
global ns_ sink0 sink21 sink17 sink20 sink35 sink29 sink36 sink44 sink13 sink41
sink49 stp sdp spdr
set t [$ns_ now]
```

```
set itval 5.0
set r0 [$sink0 set npkts_]
set r21 [$sink21 set npkts_]
set r17 [$sink17 set npkts_]
set r20 [$sink20 set npkts_]
set r35 [$sink35 set npkts_]
set r29 [$sink29 set npkts_]
set r36 [$sink36 set npkts_]
set r44 [$sink44 set npkts_]
set r13 [$sink13 set npkts_]
set r41 [$sink41 set npkts_]
set r49 [$sink49 set npkts_]
set rec [expr ($r0+$r21+$r17+$r20+$r35+$r29+$r36+$r44+$r13+$r41+$r49)/11]
set l0 [$sink0 set nlost_]
set l21 [$sink21 set nlost_]
set l17 [$sink17 set nlost_]
set l20 [$sink20 set nlost_]
set l35 [$sink35 set nlost_]
set l29 [$sink29 set nlost_]
set l36 [$sink36 set nlost_]
set l44 [$sink44 set nlost_]
set l13 [$sink13 set nlost_]
set l41 [$sink41 set nlost_]
set l49 [$sink49 set nlost_]
set los [expr ($l0+$l21+$l17+$l20+$l35+$l29+$l36+$l44+$l13+$l41+$l49)/11]
set b0 [$sink0 set bytes_]
set b21 [$sink21 set bytes_]
set b17 [$sink17 set bytes_]
set b20 [$sink20 set bytes_]
set b35 [$sink35 set bytes_]
set b29 [$sink29 set bytes_]
set b36 [$sink36 set bytes_]
set b44 [$sink44 set bytes_]
set b13 [$sink13 set bytes_]
set b41 [$sink41 set bytes_]
set b49 [$sink49 set bytes_]
set                                   byt                                   [expr
($b0+$b21+$b17+$b20+$b35+$b29+$b36+$b44+$b13+$b41+$b49)/11]
set pdr 0
if {$rec!=0} {
set pdr [expr ($rec+0.0)/($rec+$los)]
}
set tput [expr ($byt*8.0)/(2*$itval*1000000)]
puts $stp "$t\t$tput"
puts $sdp "$t\t$los"
puts $spdr "$t\t$pdr"
set inter [expr $t+5.0]
if {$inter >=40 && $inter <= 70} {
$ns_ at $inter "record1"
}
$ns_ at 5.0 "record"
$ns_ at 40.0 "record1"
$ns_ at 5.0 "nenergy 5"
# Finish Procedure to exec NAM Window
proc finish {} {
        global ns_ namtrace ptp pdp ppdr stp sdp spdr
        $ns_ flush-trace
      close $namtrace
        close $ptp
        close $pdp
        close $ppdr
        close $stp
        close $sdp
        close $spdr
        exec awk -f adelay.awk pridelay
        exec awk -f adelay.awk secdelay
        exec nam -r 5m dcqs.nam &
                exec  xgraph  P-PriDrop.xg  -geometry  800x400  -t  "  PacketDrop"  -x
"Time" -y "Avg Drop" &
        exec  xgraph  P-PriResEnergy.xg  -t  "EnergyConsumption  "  -x  "Time"  -y
"Energy(mj)" -ly 50,100 &
     exec xgraph -m -P -bg white  q1.tr -geometry 640x480 &
     exec xgraph -m -P -bg white  q2.tr -geometry 640x480 &
     exec xgraph -m -P -bg white  q3.tr -geometry 640x480 &
     exec xgraph -m -P -bg white  q4.tr -geometry 640x480 &
        exit 0
        }
$ns_ at 101.0 "finish"
puts "Start of simulation.."
$ns_ run
```

Adelay.awk

```
BEGIN {
i=1
}
{
if(FILENAME=="pridelay" || FILENAME=="secdelay") {
fle=FILENAME
d[i,1]=$1
d[i,2]=$2
i++
}}
END {
if(fle=="pridelay") {
for(k=10;k<=35;k=k+5) {
for(j=1;j<i;j++) {
if(k==d[j,1]) {
t[k]=t[k]+d[j,2]
}}
print k" "t[k] > "P-PriDelay.xg"
}}
if(fle=="secdelay") {
for(k=45;k<=70;k=k+5) {
for(j=1;j<i;j++) {
if(k==d[j,1]) {
t[k]=t[k]+d[j,2]
}}
print k" "t[k] > "P-SecDelay.xg"}}}
```

Delay.awk

```
BEGIN {
i=1
}
{
if(FILENAME=="pridelay" || FILENAME=="secdelay") {
fle=FILENAME
d[i,1]=$1
d[i,2]=$2
i++
}}
END {
if(fle=="pridelay") {
for(k=10;k<=35;k=k+5) {
for(j=1;j<i;j++) {
if(k==d[j,1]) {
t[k]=t[k]+d[j,2]
}}
print k" "t[k] > "P-PriDelay.xg"
}}
if(fle=="secdelay") {
for(k=45;k<=70;k=k+5) {
for(j=1;j<i;j++) {
if(k==d[j,1]) {
t[k]=t[k]+d[j,2]
}}
print k" "t[k] > "P-SecDelay.xg"
}}}
```

Energy.awk

```
BEGIN {
stnode=0
etnode=0
stime=0
etime=0
n=-1
t=0
nd=-1
ene=0
}
{
if(FILENAME == "etmp") {
stnode=$1
etnode=$2
stime=$3
etime=$4
}
if(FILENAME != "etmp") {
n=$1
t=$3
nd=$5
ene=$7
if($1 == "N") {
for(i=stnode;i<=etnode;i++) {
if(t>=stime && t <= etime) {
if(nd==i) {
node[i,1]=i
node[i,2]=ene
}}}} }}
END {
print "   ***************************************" > "Res_ene("etime").txt"
print "\tTime\tNode-id\tResidual-Energy" > "Res_ene("etime").txt"
print "   ***************************************" > "Res_ene("etime").txt"
for(i=stnode;i<=etnode;i++) {
print "\t"etime"\t"node[i,1]"\t"node[i,2] > "Res_ene("etime").txt"
}
```

REFERENCES

1. V. Bhuse, A. Gupta, and L. Lilien, “DPDSN: Detection of Packet- Dropping Attacks for Wireless Sensor Networks,” Proc. Fourth Trusted Internet Workshop, 2005.

2. H. Chan and A. Perrig, “Security and privacy in sensor networks,” IEEE Computer, Digital Object Identifier vol. 36, no. 10, pp. 103-105, Oct 2003.

3. Chuang Wang, Taiming Feng, Jinsook Kim, Guiling Wang, Member, IEEE, and Wensheng Zhang, Member “Catching Packet Droppers and Modifiers in Wireless Sensor Networks” in IEEE Trans. on Parallel and Distributed Systems, vol. 36, no. 5, May 2012.

4. T.H. Hai and E.N. Huh, “Detecting Selective Forwarding Attacks in Wireless Sensor Networks Using Two-Hops Neighbour Knowledge,” Proc. IEEE Seventh Int’l Symp. Network Computing and Applications (NCA ’08), 2008.

5. M. Just, E. Kranakis, and T. Wan, “Resisting Malicious Packet Dropping in Wireless Ad Hoc Networks,” Proc. Int’l Conf. Ad-Hoc Networks and Wireless (ADHOCNOW ’03), 2003.

6. M. Kefayati, H.R. Rabiee, S.G. Miremadi, and A. Khonsari, “Misbehavior Resilient Multi-Path Data Transmission in Mobile Ad-Hoc Networks,” Proc. Fourth ACM Workshop Security of Ad Hoc and Sensor Networks (SASN ’06), 2006.

7. I.Krontiris, T. Giannetsos, and T. Dimitriou, “LIDeA: A Distributed Lightweight Intrusion Detection Architecture for Sensor Networks,” Proc. Fourth Int’l Conf. Security and Privacy in Comm. Netowrks (SecureComm ’08), 2008.

8. W. Li, A. Joshi, and T. Finin, “Coping with Node Misbehaviors in Ad Hoc Networks: A Multi-Dimensional Trust Management Approach,” Proc. 11th Int’l Conf. Mobile Data Management (MDM ’10), 2010.

9. K. Liu, J. Deng, P.K. Varshney, and K. Balakrishnan, “An Acknowledgment Based Approach for the Detection of Routing Misbehavior in Manets,” IEEE Trans. Mobile Computing, vol. 6, no. 5, pp. 536-550, May 2007.

10. S. Marti, T. Giuli, K. Lai, and M. Baker, “Mitigating Routing Misbehavior in Mobile Ad Hoc Networks,” Proc. ACM MobiCom, 2000.

11. H. Song, S. Zhu, and G. Cao, “Attack-Resilient Time Synchronization for Wireless Sensor Networks,” Ad Hoc Networks, vol. 5, no. 1, pp. 112-125, 2007.

12. B. Xiao, B. Yu, and C. Gao, “Chemas: Identify Suspect Nodes in Selective Forwarding Attacks,” J. Parallel and Distributed Computing, vol. 67, no. 11, pp. 1218-1230, 2007.

13. H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, “Toward Resilient Security in Wireless Sensor Networks,” Proc. Sixth ACM Int’l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc ’05), 2005.

14. F. Ye, H. Luo, S. Lu, and L. Zhang, “Statistical En-Route Filtering of Injected False Data in Sensor Networks,” Proc. IEEE INFOCOM, 2004.

15. F. Ye, H. Yang, and Z. Liu, “Catching Moles in Sensor Networks,” Proc. 27th Int’l Conf. Distributed Computing Systems (ICDCS ’07), 2007.

16. S. Zhu, S. Setia, S. Jajodia, and P. Ning, “An Interleaved Hop-by- Hop Authentication Scheme for Filtering False Data in Sensor Networks,” Proc. IEEE Symp. Security and Privacy, 2004.

LIST OF PUBLICATIONS

1. E. Surya Praba and K. Sivan Arul Selvan, “Identification of Packet Droppers and Modifiers in Wireless Sensor Network” International Journal of Societal Applications of Computer Science, In Volume II and Issue I January 2013, ISSN 2319 – 8443.

2. E. Surya Praba and K. Sivan Arul Selvan, “Detection of Packet Droppers and Modifiers in Wireless Sensor Network”, DRDO sponsored National Conference on “Innovations in Information Technology(NCIIT 2013)” , Bannari Amman Institute of Technology, 21st and 22nd February 2013.