P-2574

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DATA AGGREGATION PROTOCOLS FOR SENSOR NETWORKS"** is the bonafide work of **"J.ANUSUYA, R.LAVANYA, and S.MRIDULA"** who carried out the project work under my supervision.

SIGNATURE

Mrs.N.Chitradevi., M.E.

SUPERVISOR

Asst.professor

Information Technology

Kumaraguru College of Technology,

Coimbatore -641006.

SIGNATURE

Dr. Thangasamy, B.E (Hons). , Ph.D

DEAN

Computer Science and Engineering,

Kumaraguru College of Technology,

Coimbatore -641006.

The candidates with the University Register number **71205205005, 71205205029 & 71205205301** was examined by us in the project viva-voce examination held on

INTERNAL EXAMINER

27/4/09

EXTERNAL EXAMINER

DECLARATION

We,

J.ANUSUYA	**71205205005**
R.LAVANYA	**71205205029**
S.MRIDULA	**71205205301**

Declare that the project entitled **"DATA AGGREGATION PROTOCOL FOR WIRELESS SENSOR NETWORKS"**, submitted in the partial fulfillment to Anna university as the project work of Bachelor of Technology(Information Technology)Degree, is a record of original work done by us under the supervision and guidance of **Mrs.N.Chitradevi, M.E.,** Asst.professor, Department of Information Technology, Coimbatore.

Place: Coimbatore

Date: 27/4/09

[J.Anusuya] [R.Lavanya] [S.Mridula]

Project Guided by

[L.S **Jayashree M.E., Ph.D**] [**Mrs.N.Chitradevi, M.E.,**]

ACKNOWLEDGEMENT

We take this opportunity to express our gratitude to all, who have helped us all the way.

We express our sincere thanks to our Chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc., F.I.E.,** Vice Chairman Dr.K.**ArumugamB.E, M.S., M.I.E.,** Correspondent **Shri.M.Balasubramaniam** and Joint Correspondent **Dr.A.Selvakumar** for all their support and ray of strengthening hope extended.

We are immensely grateful to our Vice Principal **Prof.R.Annamalai,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy,** Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project coordinator, **L.S Jayashree M.E., Ph.D.,** Asso.Professor, Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide and extend our gratitude towards **Mrs.N.Chitradevi M.E.,** Asst.professor, and Department of Information Technology for her helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project.

We also thank our parents and friends who helped us to complete this project successfully.

ABSTRACT

Sensor networks have recently emerged as an important platform. One of the main design aspects for sensor networks architecture is energy efficiency to keep the network operational as long as possible. Therefore data aggregation technique are an essential block as they aim to reduce the communication overhead and energy expenditure of sensor nodes during the process of data collection in a sensor network.

This project proposes a scheme for DATA AGGREGATION. The Data Aggregation Protocol which uses a novel grouping technique to dynamically partition the nodes in a tree topology into multiple logical groups of similar sizes, and aggregation is done in the leader node. Then the aggregated data is updated in the base station. Then we proposed Grubb's algorithm which detects the outliers and improves data accuracy. The simulations are done using NS2 and the evaluated performance shows the efficiency of the proposed scheme.

LIST OF FIGURES

LIST OF ABBREVIATIONS

WSN	-Wireless Sensor Network
DAP	-Data Aggregation Protocol
TCP	-Transmission Control Protocol
UDP	-User Datagram Protocol
MAC	-Message Authentication Code
NAM	-Network Animation
Tcl	-Tool Command Language

CONTENTS

CONTENTS

INTRODUCTION

1. INTRODUCTION

1.1 GENERAL

1.1.1 OVERVIEW OF SENSOR NETWORKS

Wireless sensor networks are a new class of networks consisting of individual nodes that are able to interact with their environment by sensing or controlling physical parameters. These nodes have to collaborate to fulfill their tasks as, usually, a single node is incapable of doing so; and they use wireless communication to enable this collaboration.

Wireless sensor networks are envisioned to be economic solutions to many important applications, such as real-time traffic monitoring, military surveillance, and homeland security. A sensor network may consist of hundreds of sensors, each of which acts as an information source, sensing and collecting data from the environment for a given task. One of the major problems of any sensor network is determining the most efficient way of conserving the energy of the power source.

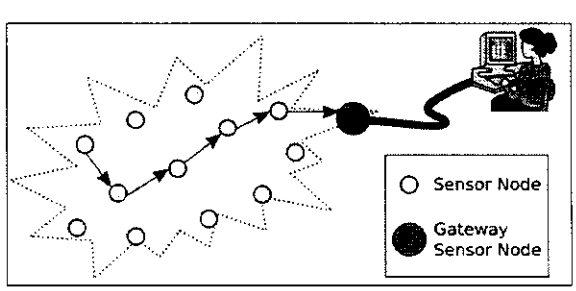


Fig.no:1 Wireless Sensor Networks

COMPONENTS OF SENSOR NETWORKS

A typical sensor node in a sensor network consists of three main components. They are

- Communication component
- Sensing component
- Processing component

Of the above three components the most energy consuming component is the communication component. The next energy consuming component is the sensing component and the least energy consuming component is the processing component.

1.1.2 APPLICATION OF SENSOR NETWORKS

Wireless sensor networks have a wide range of applications such as,

1. Millitary applications
    i. Monitoring friendly forces and equipment.
    ii. Battlefield surveillance.
    iii. Nuclear, biological and chemical attack detection.
2. Environmental Applications
    i. Forest fire detection
    ii. Bio-complexity mapping of the environment
    iii. Flood detection.
3. Health Applications
    i. Tele-monitoring of human physiological data.
    ii. Tracking and monitoring doctors and patients inside a hospital
    iii. Drug administration in hospitals.
4. Home Applications
    i.Home automation.
    ii.Smart environment

1.2 PROBLEM DEFINITION

One of the major problems in sensor network is existing of various potential attacks. In Data Aggregation Protocol, we focus on defending against False Data Injection Attacks where the goal of an attacker is to make the base station to accept false sensor reports. The attacker will inject false values that deviate from the true values in a noticeable scale. In order to detect and eliminate the deviation we proposed GRUBBS' test. Grubbs' test is a hypothesis test for detecting data outliers.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 DATA AGGREGATION PROTOCOL

Data aggregation is a process in which information is gathered and expressed in a summary form, for purposes such as statistical analysis. A common aggregation purpose is to get more information about particular groups based on specific variables such as age, profession, or income.

Data aggregation is a very important technique for reducing the communication overhead and energy expenditure of sensor nodes during the process of data collection in a sensor networks.

Many data aggregation protocols have been proposed to eliminate the data redundancy in sensor data of the network.

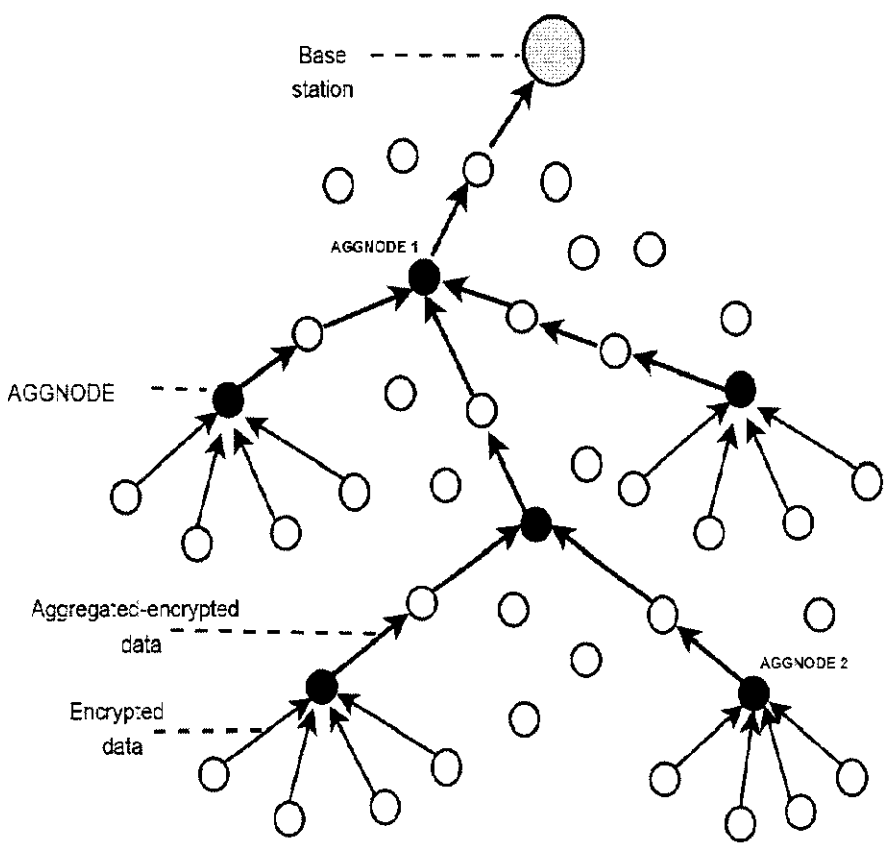


Fig.no:2 Data Aggregation Protocol

DATA AGGREGATION IN A SUBTREE

A sub tree consists of child nodes connected to a cluster head, i.e., the sub tree will have only single hop data packet transmissions. The amount of data packets waiting for service will change according to the arrival rate transition. This will become a critical parameter for any real time application because if the queue size exceeds the buffer size then data loss will occur. The arrival rate which causes data loss is identified and detected.

DATA AGGREGATION FROM THE CLUSTER HEADS TO A SINK NODE

The sink node is connected to cluster heads. The data packets from the cluster heads have to be collected, analyzed and forwarded to the sink node. The number of data packets sent from each cluster head differs with respect to the number of child nodes attached to the cluster head. Therefore, the number of data packets arriving at the sink node from each cluster head will differ. The sink will maintain one queue for each of the cluster heads since it has to handle more than one data packet from each of the queues at a time t.

2.2 OUTLIER DETECTION METHOD

DEFINITIONS

Hawkins (1980) defines an outlier as an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism.

Barnet and Lewis (1994) indicate that an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs.

Johnson (1992) defines an outlier as an observation in a data set which appears to be inconsistent with the remainder of that set of data

TAXONOMY OF OUTLIER DETECTION METHODS

Outlier detection methods can be divided into **univariate methods**, and **multivariate methods** that usually form most of the current body of research. Another fundamental taxonomy of outlier detection methods is between **parametric methods** and **nonparametric methods**.

UNIVARIATE STATISTICAL METHODS

Most of the earliest univariate methods for outlier detection rely on the assumption of an underlying known distribution of the data, which is assumed to be identically and independently distributed .Moreover, many discordance tests for detecting univariate outliers further assume that the distribution parameters and the type of expected outliers . The outlier identification problem is then translated to the problem of identifying those observations that lie in a so-called *outlier region*.

MULTIVARIATE OUTLIER DETECTION

In many cases multivariable observations cannot be detected as outliers when each variable is considered independently. Outlier detection is possible only when multivariate analysis is performed, and the

interactions among different variables are compared within the class of data.

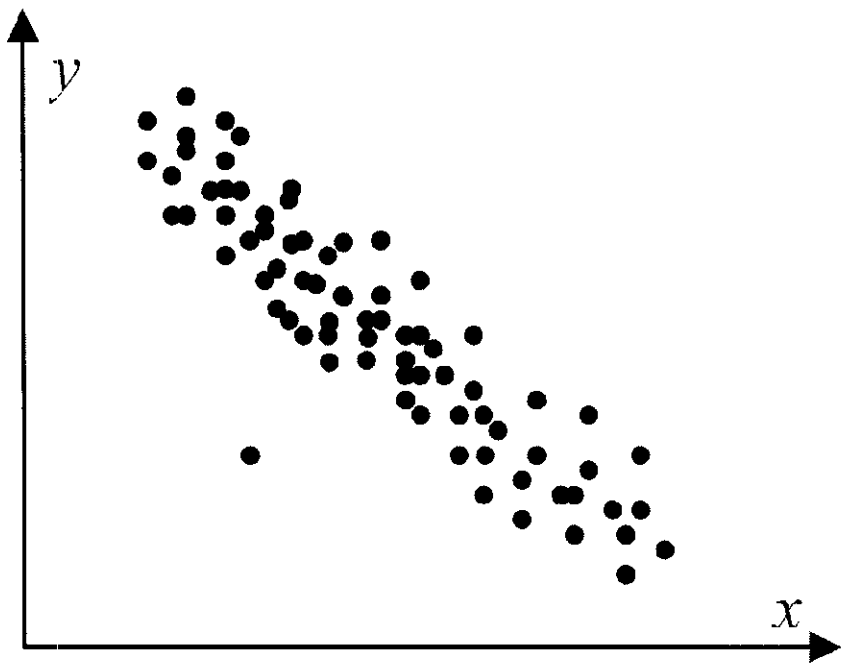


Fig.no:3 Multivariate Outlier Detection

The lower left observation is clearly a multivariate outlier but not a univariate one. When considering each measure separately with respect to the spread of values along the x and y axes, we had seen that they fall close to the centre of the univariate distributions. Thus, the test for outliers must take into account the relationships between the two variables.

DETAILS OF METHODOLOGY EMPLOYED

3. DETAILS OF THE METHODOLOGY EMPLOYED
3.1 MODULES

3.1.1 TREE CONSTRUCTION

In the tree construction module, initially the root broadcasts tree construction beaconing message which includes its own id and its depth to be 0. When a node, say x, receives a broadcast message at its first time from a node y, x assigns its depth to be the depth of y plus one, and its parent to be y. After this, it rebroadcasts the message. This process continues until all nodes have received this message. After constructing the tree, the BS can disseminate the message through this tree.

$$BS \rightarrow\rightarrow :* Fagg, - (1)$$

Where

Fagg refers to a specific aggregation function, such as MEAN, SUM.

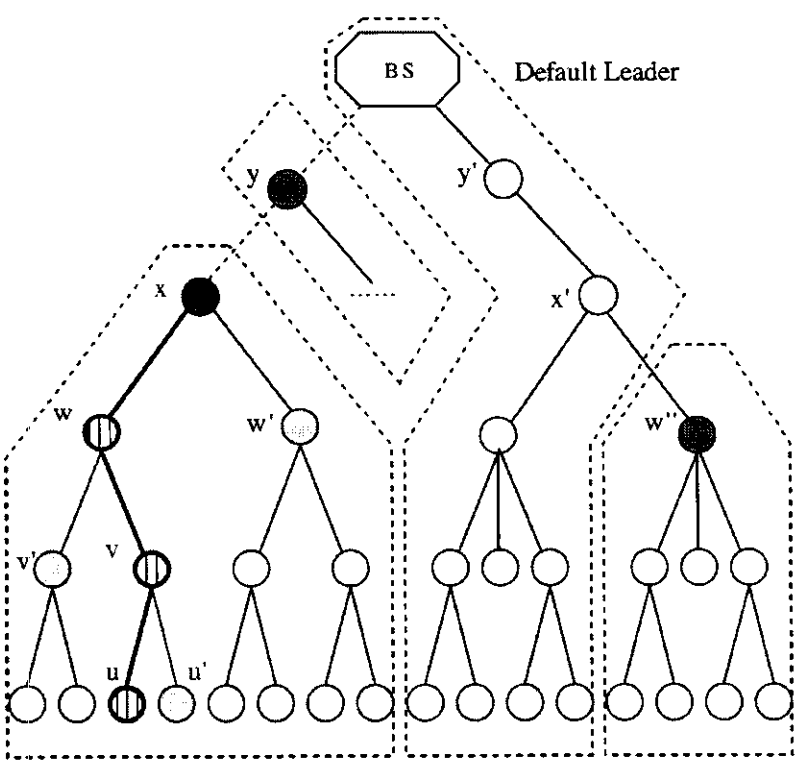


Fig.no:4 Tree Construction

3.1.2 DATA AGGREGATION

Aggregation is defined as the process of aggregating the data by using the aggregating functions such as MEAN,SUM,COUNT.

AGGREGATION

Leaf Node Aggregation

Data aggregation starts from the leaf nodes in the aggregation tree towards the BS. Since a leaf node does not need to do aggregation, it just sends its id, data, and count value to its parent.

Intermediate node aggregation

When an intermediate node receives an aggregate from its Child node, it first checks the flag. If the flag is 0, it keeps a local copy of the aggregates and performs further aggregation; otherwise, the node directly sends the packet to its parent node.

Leader node aggregation

Now suppose that an intermediate node has processed the aggregates from the child nodes and it finds out that it is a group leader like a regular intermediate node, it also computes a new aggregate, keeps local copies of those packets with flag $0.x$.

3.1.3 OUTLIER DETECTION

Outlier data is one which deviates from the original data samples. So it is considered as a false data which needs to be eliminated to obtain the required result. If the outlier is not eliminated, then it will leads to inconsistent data.

To avoid this issue, we proposed GRUBBS' method to detect and eliminate the outliers. Grubbs method is applied to the leader node and

the compromised (faulty sensor) node is identified and the outlier value is detected and eliminated. Finally data accuracy is ensured.

GRUBBS METHOD

Grubbs' test is a hypothesis test for detecting data outliers.

Given a dataset $= \{\chi 1, \chi 2, \cdots, \chi n\}$,

Then the data xi ($1 \leq i \leq n$).

$$Z = | xi - \bar{x} | / S$$

$$S = \sqrt{((xi^2/n) - \bar{x}^2)}$$

Where,

$\bar{x}$ - The sample mean

S- Standard deviation of all the data.

Xi- selected value from the given sample set.

Thus Z value is calculated,

Then it is compared with the following equation to detect the outlier

$$Z > [N-1/\sqrt{N}] * \sqrt{t^2[(\alpha/2N), (N-2)]\backslash N-2+t^2 [(\alpha/2N), (N-2)]}$$

Where,

N - Total number of samples.

α - Predefined Significance level.

t - Value from t-distribution table.

Thus Grubbs' test detects outlier from the dataset and iterates the test over the remaining data until no outliers can be found.

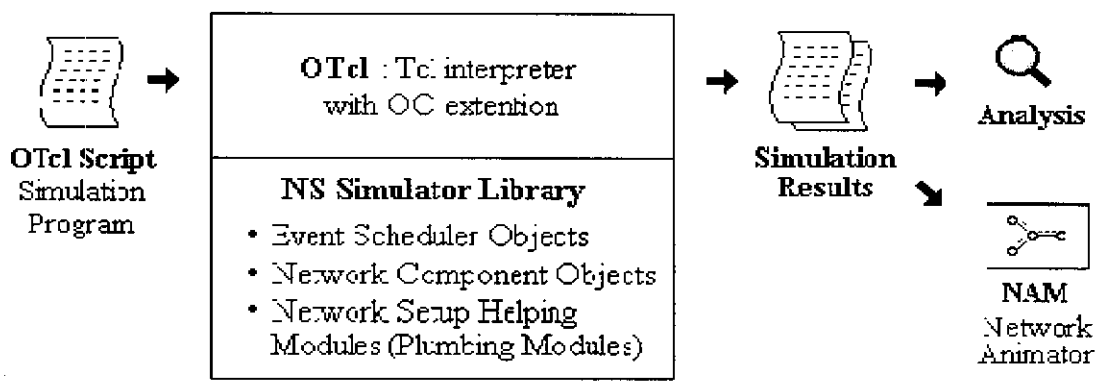
SIMULATION SCENARIO & ENVIRONMENT

4. SIMULATION SCENARIO AND ENVIRONMENT:

INTRODUCTION TO NS2-SIMULATOR

The ns simulator covers a very large number of application protocols, of network types, and of traffic models. The NS simulator is written in two languages: an object oriented simulator, written in c++ and an OTCL (an object oriented extension of TCL) interpreter, used to create user's command script's is a discrete event simulator, where the advance of time depends on timing of events maintained by a scheduler. An event is an object in a C++ hierarchy with a unique ID, a scheduler time and a pointer to an object that handles the event.

NS2 stands for Network Simulator Version 2. NS-2 is a discrete event simulator targeted at network research Focused on modeling network protocols.



COMPONENTS OF NS

Nam, the Network Animator

- Visualize ns (or other) output.
- GUI input simple ns scenarios.

Pre-processing

- Traffic and topology generators.

Post-processing

- Simple trace analysis, often in Awk, Perl, or Tcl.

GOALS OF NS

- Support networking research and education

  - Protocol design, traffic studies, etc.
  - Protocol comparison.

- Provide a collaborative environment

  - Freely distributed, open source.
  - Share code, protocols, models, etc.
  - Allow easy comparison of similar protocols.
  - Increase confidence in results.
  - Models provide useful results in several Situations.

- It covers multiple layers

  Application layer, transport layer, network layer And link layer.

- Supports the simulation of Intserv/diffserv, Multicast, Transport, Applications Wireless (fixed, mobile, satellite).

TWO LANGUAGES

- C++

  - Detailed protocol simulations require systems programming language.
  - Byte manipulation, packet processing, algorithm implementations.
  - Run time speed is important.
  - Turn around time (Run simulation, find bug, fix bug, re-compile) is slower.

- **Tcl**
    - Simulations of slightly varying parameters or configurations.
    - Quickly exploring a number of scenarios.
    - Iteration time (change the model and re-run) is more.

IMPORTANT NS-2 PROGRAMMING LANGUAGES

- **NS-2**

  It is an object oriented simulator, written in C++, with an OTcl (Object Tool Command Language) interpreter as a front-end.

- **Back-end C++**

  - Defining new agents, protocols and framework.
  - Manipulations at the byte/bit levels.
  - If you have to change the behavior of an existing C++ class anyways that weren't anticipated

- **Front-end Otcl**

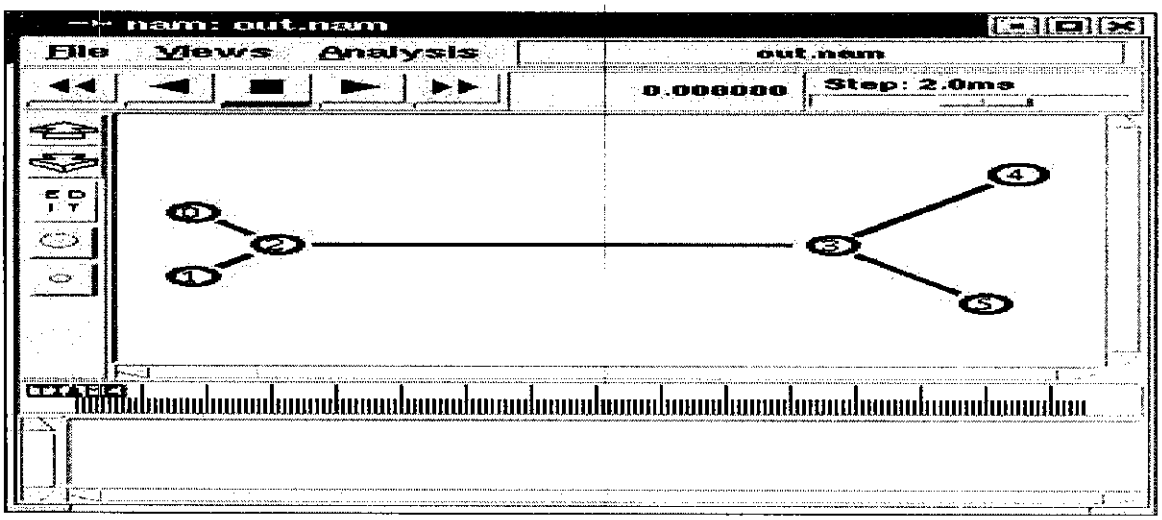
  - Topologies, scenarios, simulations..,
  - Script language (easy topology modifications)
  - If you can do what you want by manipulating existing C++ objects.

Why Two Languages

- Simulator had two distinct requirements
  - Detailed simulation of Protocol (Run-time speed)
  - Varying parameters or configuration (Change model & rerun).
- C++ is fast to run but slower to change.
- OTcl runs much slower but can be changed quickly.

Nam (Network Animator)

"Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces."



X-Graph

- Convert trace output into X-Graph format
- Command line start : "X-Graph"

Creation of Trace Files

- Used to trace packets on all links

  Set tracefd [open simple.tr w]

  $ns_trace-all $tracefd

Format

Event	Time	From node	To node	Pkt type	Pkt size	Flags	Fid	Src addr	Dst addr	Seq num	Pkt id

r: receive (at to node)
+: enqueue (at queue)
- : dequeue (at queue)

d: drop (at queue)

```
r 1.086667 3 5 cbr 1000 ------- 2 1.0 5.0 1 1
r 1.111227 2 3 tcp 40 ------- 1 0.0 4.0 0 2
+ 1.111227 3 4 tcp 40 ------- 1 0.0 4.0 0 2
- 1.111227 3 4 tcp 40 ------- 1 0.0 4.0 0 2
```

WIRELESS SIMULATION

- **Defining wireless options**

```
set val (chan)    Channel/WirelessChannel      ;# channel type
set val(prop)     Propagation/TwoRayGround     ;# radio-propagation model
set val(netif)    Phy/WirelessPhy              ;# network interface type
set val(mac)      Mac/802_11                   ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue      ;# interface queue type
set val(ll)       LL                           ;# link layer type
set val(ant)      Antenna/OmniAntenna          ;# antenna model
set val(ifqlen)   5000                         ;# max packet in ifq
set val(nn)       35                           ;# number of mobilenodes
set val(rp)       AODV                         ;# routing protocol
set val(x)        1500

set val(y)        1000

set ns_           [new Simulator]

set tracefd    [open tree.tr w]

$ns_ trace-all $tracefd

set namtrace [open tree.nam w]

$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
```

- **set up topography object**

```
set topo      [new Topography]

$topo load_flatgrid $val(x) $val(y)
```

- **Create God – General Operations Descriptor**

```
create-god $val(nn)
```

- **Configure node:**

```
$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 OFF \                    -
                      - movementTrace OFF
```

- **creation of nodes**

```
for { set i 0 } { $i < $val(nn) } { incr i } {
global n
set node_($i) [$ns_ node]
}
```

- **To set the random motion**

```
for { set i 0 } { $i < $val(nn) } { incr i } {
$node_($i) random-motion 0
}
```

- **To set the size of the nodes**

```
for { set i 0} {$i < $val(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 60}
```

- **Adding nodes and setting positions**

```
$node_(1) set X_ 185.0
$node_(1) set Y_ 550.0
$node_(1) set Z_ 0.0
```

- **Adding agents and taffic source**

```
set tcp [new Agent/TCP]
$tcp set class_ 1
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $tcp
$ns_ attach-agent $node_(2) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp1
$ns_ at 1.2 "$ftp start"
$ns_ at 2.5 "$ftp stop"
```

- **Tell nodes when the simulation ends**

```
for {set i 0} {$i < $val(nn) } {incr i} {
$ns_ at 10.0 "$node_($i) reset";}
$ns_ at 10.02 "stop"
$ns_ at 10.03 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
global ns_ tracefd
$ns_ flush-trace
close $tracefd
puts "running nam..."
exec nam tree &
        }
 $ns_ run
```

PERFORMANCE EVALUATION

5. PERFORMANCE EVALUATION

5.1 RESULT

NODES SETUP

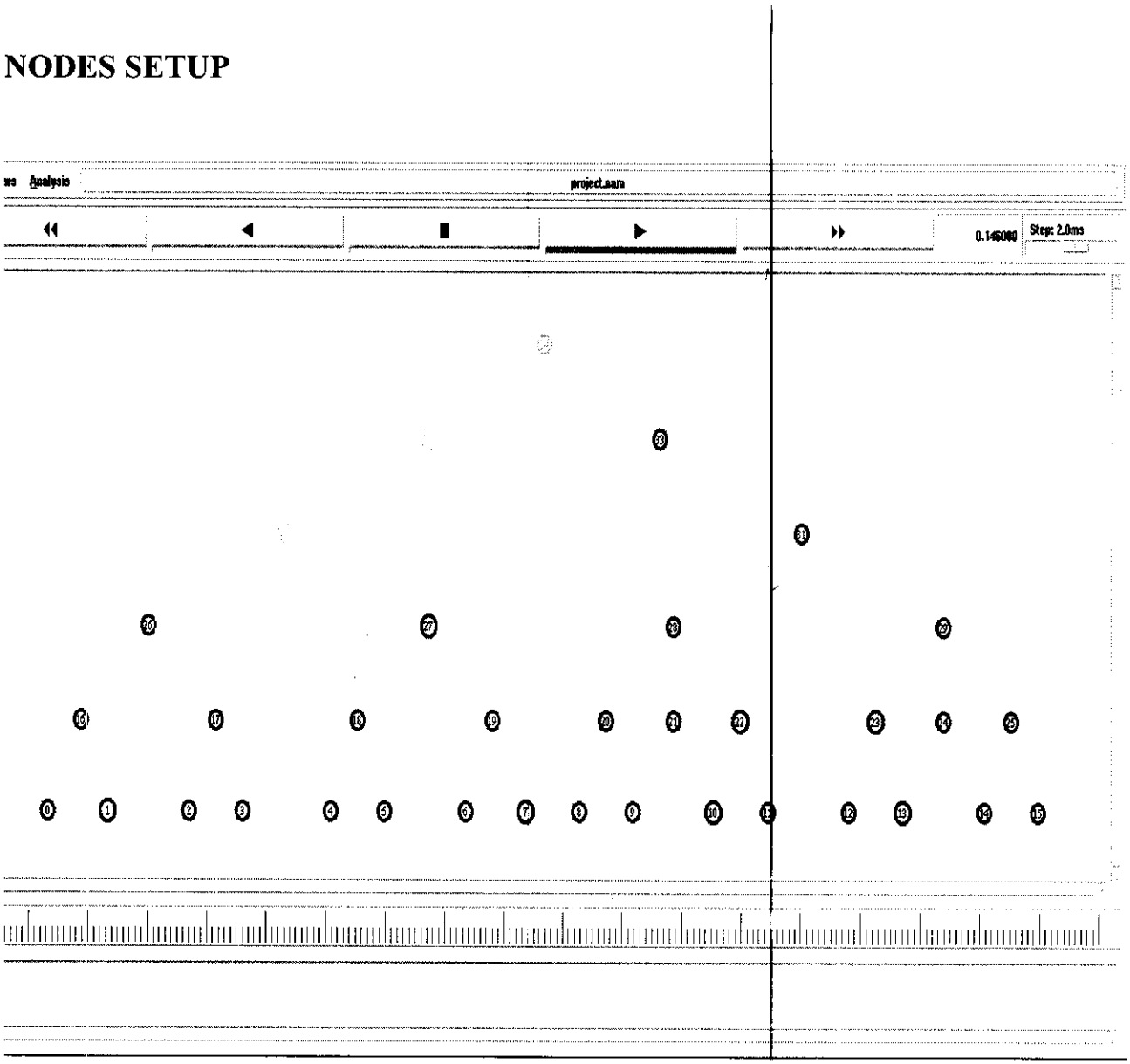


Fig.no:7 Nodes setup

RANGING FOR DATA TRANSMISSION

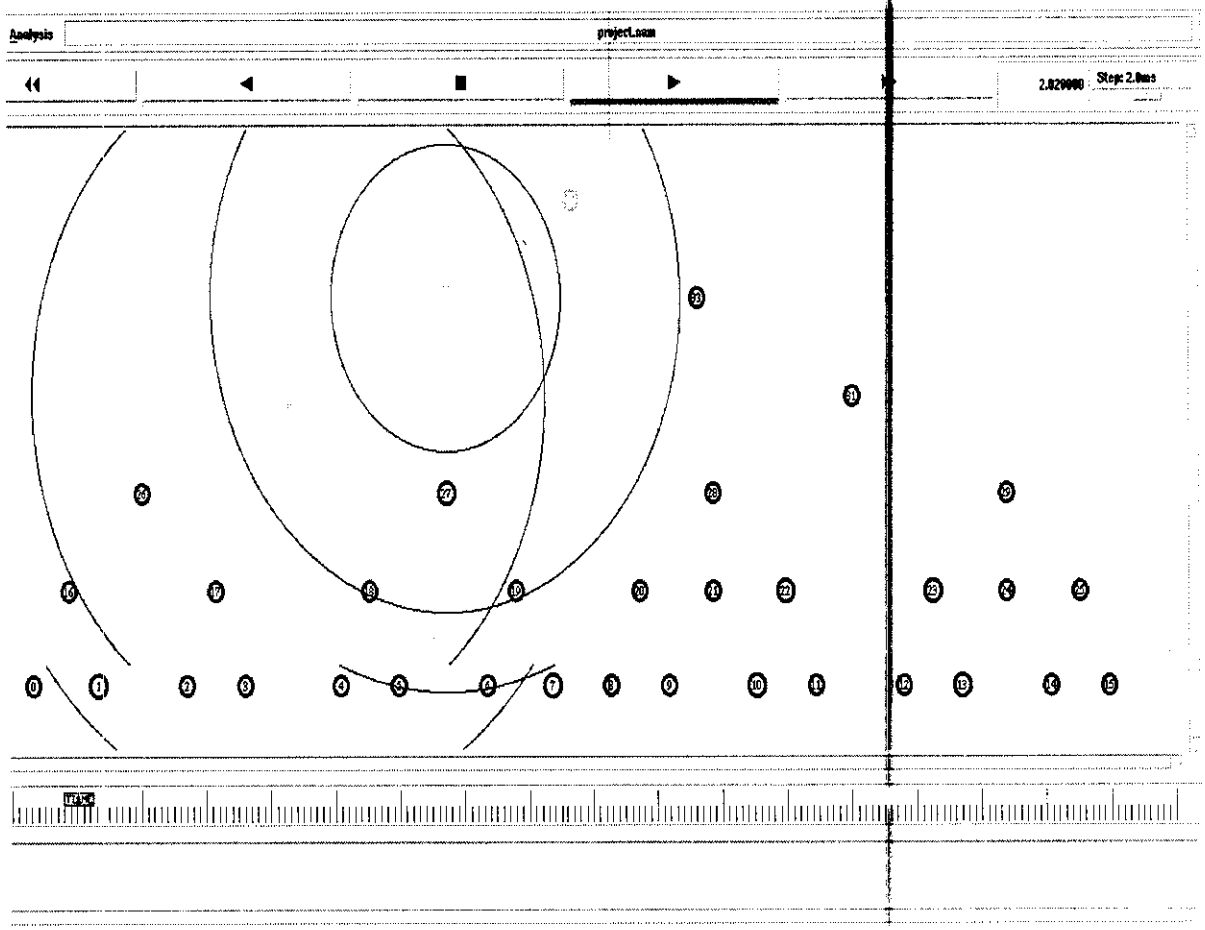


Fig.no:8 Ranging for data Transmission

DATA TRANSMISSION

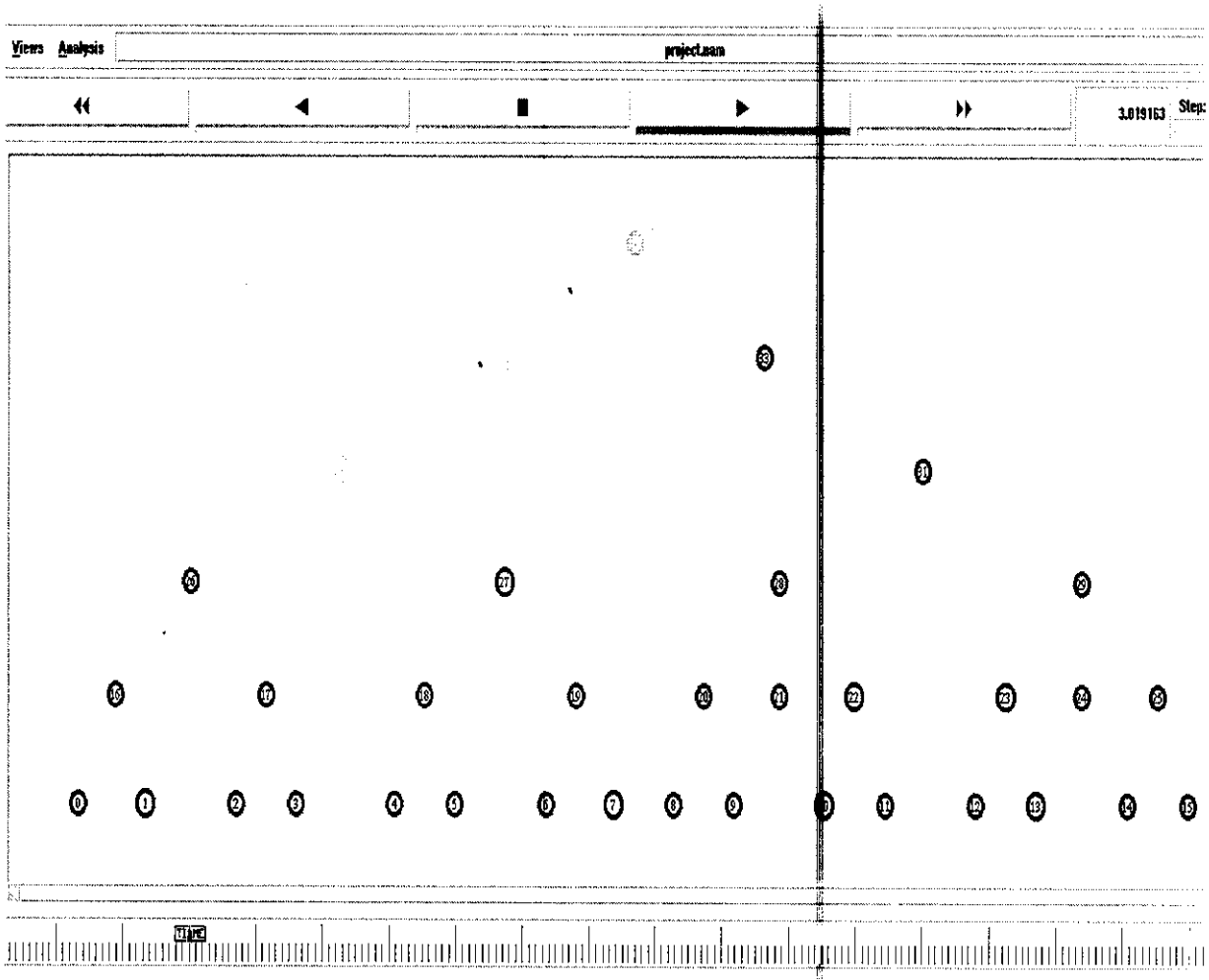


Fig. no: 9 Data transmission

5.2 GRAPH

COMPARISON BETWEEN NAÏVE APPROACH AND GRUBBS' TEST

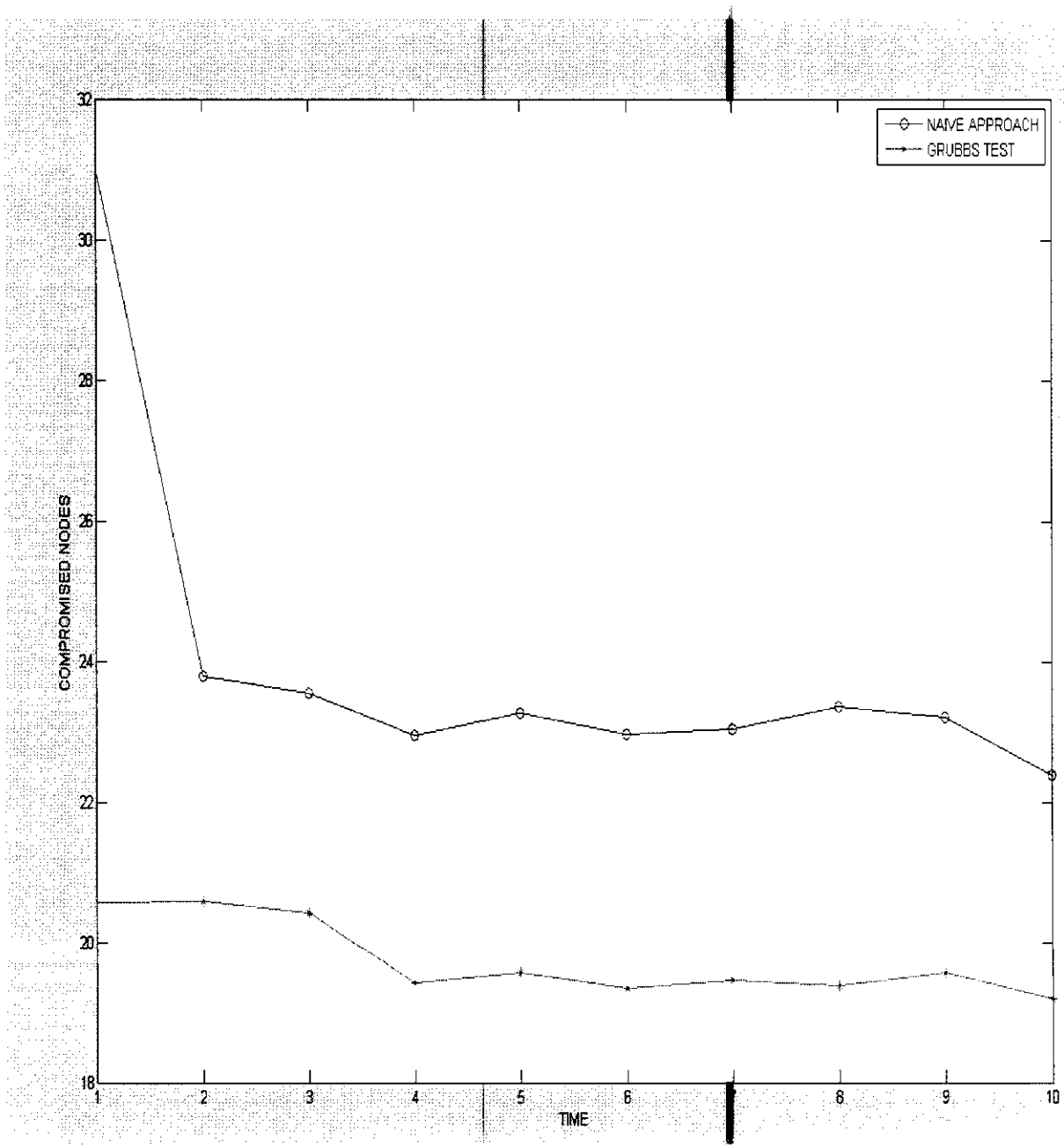


Fig. no: 10 Naïve approach vs. Grubbs' test

DETECTION RATE VS NUMBER OF ATTACKS

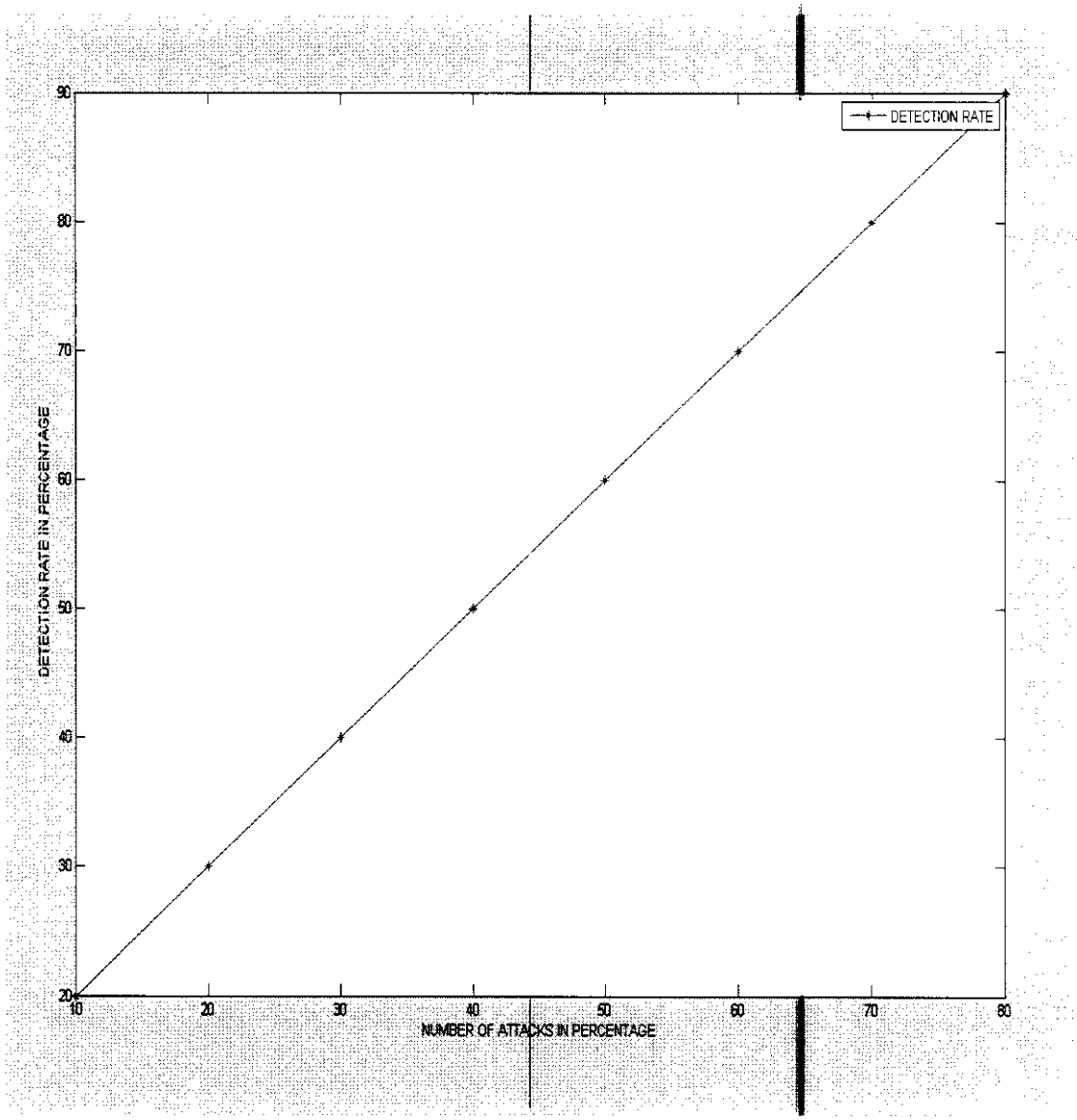


Fig.no:11 Detection rate vs. Number of attacks

FALSE ALARM RATE VS NUMBER OF OUTLIERS

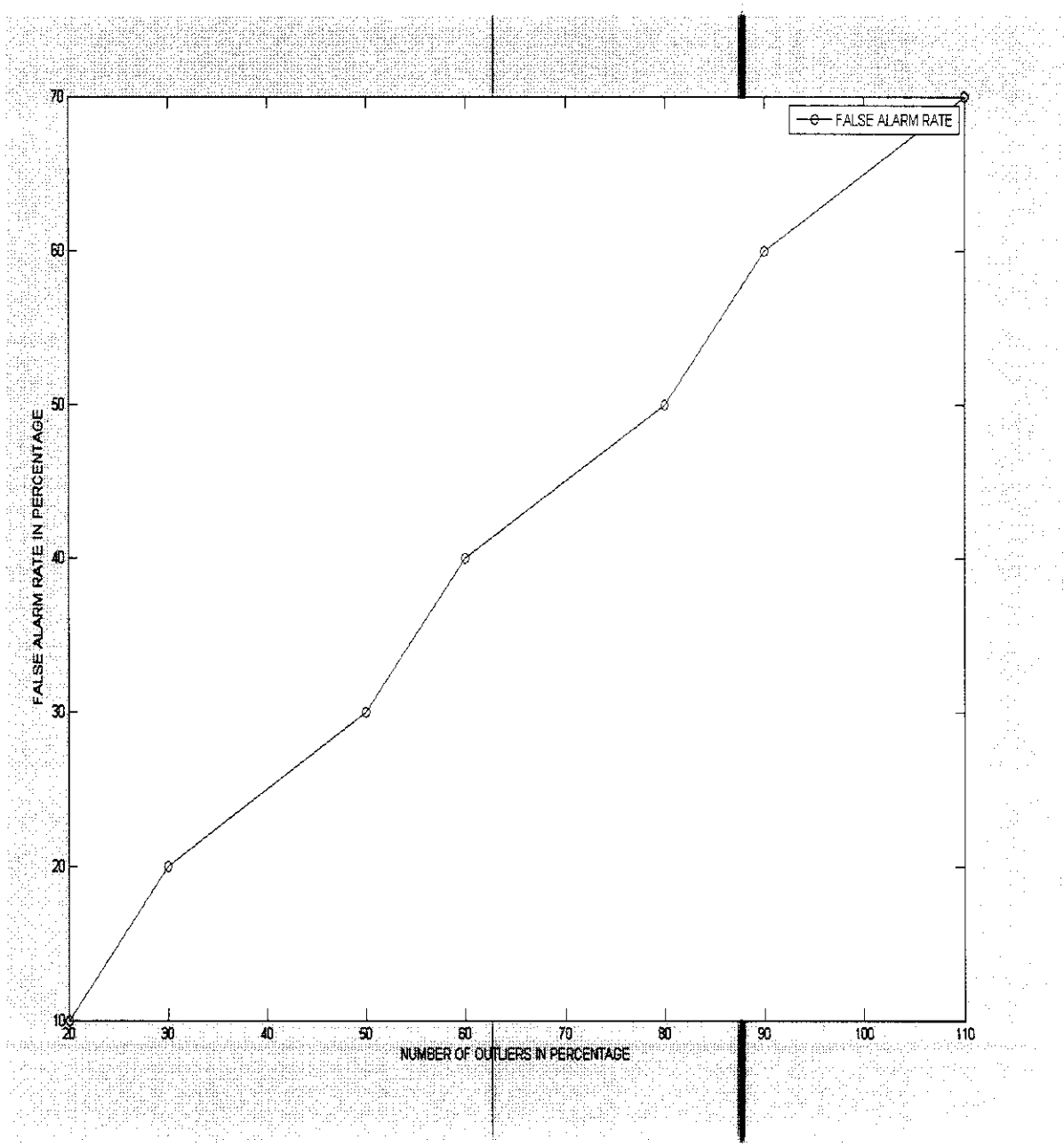
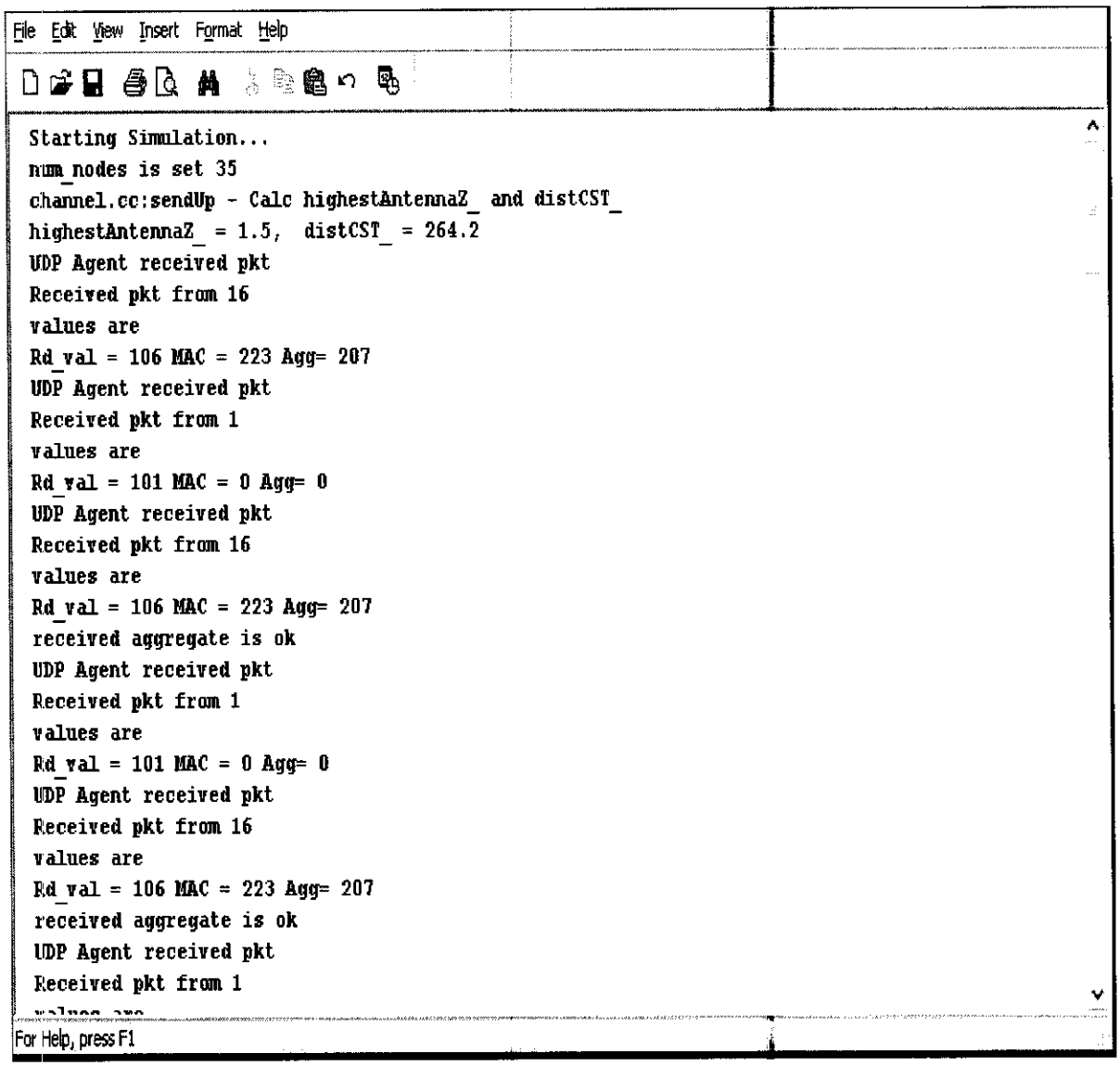


Fig.no:12 False alarm rate vs. number of outliers

5.3 OUTPUT


File Edit View Insert Format Help
Starting Simulation...
num_nodes is set 35
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 264.2
UDP Agent received pkt
Received pkt from 16
values are
Rd_val = 106 MAC = 223 Agg= 207
UDP Agent received pkt
Received pkt from 1
values are
Rd_val = 101 MAC = 0 Agg= 0
UDP Agent received pkt
Received pkt from 16
values are
Rd_val = 106 MAC = 223 Agg= 207
received aggregate is ok
UDP Agent received pkt
Received pkt from 1
values are
Rd_val = 101 MAC = 0 Agg= 0
UDP Agent received pkt
Received pkt from 16
values are
Rd_val = 106 MAC = 223 Agg= 207
received aggregate is ok
UDP Agent received pkt
Received pkt from 1
For Help, press F1

CONCLUSION

6. CONCLUSION

This project proposed Data Aggregation Protocol, which uses a novel grouping technique to dynamically partition the nodes in a tree topology into multiple logical groups of similar sizes, and aggregation is done in the leader node. Then the aggregated data is updated in the base station. Then we proposed Grubb's algorithm which detects the outliers and improves data accuracy. The simulations are done using NS2.

FUTURE ENHANCEMENTS

7. FUTURE ENHANCEMENT

In our proposal, data aggregation is done and outlier is detected for smaller networks and it is tested for reliable transmission. In future, this project can be extended for larger networks and can be tested for unreliable transmission.

APPENDIXS

7. APPENDIXS

7.1 Source code

Project.tcl

```
# Define options
=================================================================
==================
set val(chan)    Channel/WirelessChannel     ;# channel type
set val(prop)    Propagation/TwoRayGround    ;# radio-propagation model
set val(netif)   Phy/WirelessPhy             ;# network interface type
set val(mac)     Mac/802_11                  ;# MAC type
set val(ifq)     Queue/DropTail/PriQueue     ;# interface queue type
set val(ll)      LL                          ;# link layer type
set val(ant)     Antenna/OmniAntenna         ;# antenna model
set val(ifqlen)  5000                        ;# max packet in ifq
set val(nn)      35                          ;# number of mobilenodes
set val(rp)      AODV                        ;# routing protocol

#Phy/WirelessPhy set Pt_ 0.015
=================================================================
==================
# Main Program
#
=================================================================
==================
Agent/AODV set H0 0
#
# Initialize Global Variables
#
set ns_                                              [new Simulator]
#to trace the events generated by MAC, Router and Agent
set tracefd     [open project.tr w]
$ns_ trace-all $tracefd

#to generate animation file
set namfd                                          [open project.nam w]
$ns_ namtrace-all-wireless $namfd 750 260
# set up topography object
set topo       [new Topography]
$topo load_flatgrid 750 260
```

```
# Create God
create-god $val(nn)
# Create the specified number of mobilenodes [$val(nn)] and "attach"
them to the channel.
# configure node
      $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 OFF \                    -
                               movementTrace OFF

         for {set i 0} {$i < $val(nn) } {incr i} {
         set node_($i) [$ns_ node]
         $node_($i) random-motion 0          ;# disable random motion
   }
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 50.0
$node_(1) set Y_ 2.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 110.0
$node_(2) set Y_ 2.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 150.0
$node_(3) set Y_ 2.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 215.0
$node_(4) set Y_ 2.0
$node_(4) set Z_ 0.0
```

```
$node_(5) set X_ 255.0
$node_(5) set Y_ 2.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 315.0
$node_(6) set Y_ 2.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 360.0
$node_(7) set Y_ 2.0
$node_(7) set Z_ 0.0

$node_(8) set X_ 400.0
$node_(8) set Y_ 2.0
$node_(8) set Z_ 0.0

$node_(9) set X_ 440.0
$node_(9) set Y_ 2.0
$node_(9) set Z_ 0.0

$node_(10) set X_ 500.0
$node_(10) set Y_ 2.0
$node_(10) set Z_ 0.0

$node_(11) set X_ 540.0
$node_(11) set Y_ 2.0
$node_(11) set Z_ 0.0

$node_(12) set X_ 600.0
$node_(12) set Y_ 2.0
$node_(12) set Z_ 0.0

$node_(13) set X_ 640.0
$node_(13) set Y_ 2.0
$node_(13) set Z_ 0.0

$node_(14) set X_ 700.0
$node_(14) set Y_ 2.0
$node_(14) set Z_ 0.0

$node_(15) set X_ 740.0
$node_(15) set Y_ 2.0
```

```
$node_(15) set Z_ 0.0

$node_(16) set X_ 30.0
$node_(16) set Y_ 50.0
$node_(16) set Z_ 0.0


$node_(17) set X_ 130.0
$node_(17) set Y_ 50.0
$node_(17) set Z_ 0.0

$node_(18) set X_ 235.0
$node_(18) set Y_ 50.0
$node_(18) set Z_ 0.0

$node_(19) set X_ 335.0
$node_(19) set Y_ 50.0
$node_(19) set Z_ 0.0

$node_(20) set X_ 420.0
$node_(20) set Y_ 50.0
$node_(20) set Z_ 0.0

$node_(21) set X_ 470.0
$node_(21) set Y_ 50.0
$node_(21) set Z_ 0.0

$node_(22) set X_ 520.0
$node_(22) set Y_ 50.0
$node_(22) set Z_ 0.0

$node_(23) set X_ 620.0
$node_(23) set Y_ 50.0
$node_(23) set Z_ 0.0

$node_(24) set X_ 670.0
$node_(24) set Y_ 50.0
$node_(24) set Z_ 0.0

$node_(25) set X_ 720.0
$node_(25) set Y_ 50.0
$node_(25) set Z_ 0.0

$node_(26) set X_ 80.0
```

```
$node_(26) set Y_ 100.0
$node_(26) set Z_ 0.0

$node_(27) set X_ 288.0
$node_(27) set Y_ 100.0
$node_(27) set Z_ 0.0

$node_(28) set X_ 470.0
$node_(28) set Y_ 100.0
$node_(28) set Z_ 0.0

$node_(29) set X_ 670.0
$node_(29) set Y_ 100.0
$node_(29) set Z_ 0.0

$node_(30) set X_ 180.0
$node_(30) set Y_ 150.0
$node_(30) set Z_ 0.0

$node_(31) set X_ 565.0
$node_(31) set Y_ 150.0
$node_(31) set Z_ 0.0

$node_(32) set X_ 288.0
$node_(32) set Y_ 200.0
$node_(32) set Z_ 0.0

$node_(33) set X_ 460.0
$node_(33) set Y_ 200.0
$node_(33) set Z_ 0.0

$node_(34) set X_ 374.0
$node_(34) set Y_ 250.0
$node_(34) set Z_ 0.0

for {set i 0} {$i < $val(nn) } { incr i} {
            set if_($i) [$node_($i) set netif_(0)]
            #$ns_ at 0.0 "$if_($i) set Pt_ 0.005"
}

for {set i 0} { $i < 16} { incr i} {
            #set if_($i) [$node_($i) set netif_(0)]
            $ns_ at 0.0 "$if_($i) set Pt_ 0.0045"
}
```

```
for {set i 16} { $i < $val(nn)} { incr i} {
        #set if_($i) [$node_($i) set netif_(0)]
        $ns_ at 0.0 "$if_($i) set Pt_ 0.015"
}

$ns_ at 0.00 "$node_(30) set is_leader 1"
$ns_ at 0.00 "$node_(32) set is_leader 1"
$ns_ at 0.00 "$node_(29) set is_leader 1"

for {set i 0} {$i < 16} {incr i} {
        $ns_ at 0.00 "$node_($i) set is_leaf 1"
}

for {set i 16} { $i < 29} { incr i} {
        $ns_ at 0.00 "$node_($i) set is_inter 1"
}

$ns_ at 0.0 "$node_(31) set is_inter 1"
$ns_ at 0.0 "$node_(33) set is_inter 1"

for { set i 0} { $i < 8} { incr i} {
        $ns_ at 0.0 "$node_($i) color blue"
}
for { set i 16} { $i < 20} { incr i} {
        $ns_ at 0.0 "$node_($i) color red"
}

$ns_ at 0.0 "$node_(26) color red"
$ns_ at 0.0 "$node_(27) color red"
$ns_ at 0.0 "$node_(30) color gold"
$ns_ at 0.0 "$node_(32) color gold"
$ns_ at 0.0 "$node_(34) color green"

for { set i 0} { $i < 8} { incr i} {
        $ns_ at 0.0 "$node_($i) set Rd_val [expr 100+$i]"
}

for { set i 16} { $i < 20 } { incr i} {
        $ns_ at 0.0 "$node_($i) set Rd_val [expr 90+$i]"
}

$ns_ at 0.0 "$node_(26) set Rd_val 100"
$ns_ at 0.0 "$node_(27) set Rd_val 110"
```

```
$ns_ at 0.0 "$node_(30) set Rd_val 120"
$ns_ at 0.0 "$node_(32) set Rd_val 130"
#$ns_ at 0.0 "$node_(34) set Rd_val 100"

$ns_ at 5.0 "$node_(30) set chg_ 1"

set sink [new Agent/UDP]
$ns_ attach-agent $node_(34) $sink

for {set i 0} {$i < 8 } {incr i} {
set src_($i) [new Agent/UDP]
$ns_ attach-agent $node_($i) $src_($i)
$ns_ connect $src_($i) $sink

set app_($i) [new Application/Traffic/CBR]
$app_($i) attach-agent $src_($i)
$app_($i) set interval_ 1
}
set j 8
for {set i 16} {$i < 20 } {incr i} {
set src_($j) [new Agent/UDP]
$ns_ attach-agent $node_($i) $src_($j)
$ns_ connect $src_($j) $sink

set app_($j) [new Application/Traffic/CBR]
$app_($j) attach-agent $src_($j)
$app_($j) set interval_ 1
set j [expr $j + 1]
}

for {set i 1} {$i < 2 } {incr i} {
  $ns_ at [expr 1.0 + $i] "$app_($i) start"
}
$ns_ at 2.0 "$app_(8) start"

for {set i 0} {$i < $val(nn) } { incr i} {
  $ns_ initial_node_pos $node_($i) 10
}
#
# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
   $ns_ at 20.0 "$node_($i) reset";
}
$ns_ at 20.0 "stop"
```

```
$ns_ at 20.01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
   global ns_ tracefd namfd
   $ns_ flush-trace
   close $tracefd
   close $namfd
}

puts "Starting Simulation..."
$ns_ run
```

Aodv.cc

```
// SDAP starts
// get the sender's id and details
        Node* srcnode = 0 ;
        Node* menode = 0 ;
        int src = Address::instance().get_nodeaddr(ih->saddr());
        srcnode = Node::get_node_by_address(src);
        menode = Node::get_node_by_address(index);
        MobileNode* msrcnode = (MobileNode*)srcnode;
        MobileNode* mmenode = (MobileNode*)menode;
        //if(msrcnode->flag == 0)
        if(ch->flag ==0)
        {
        if(mmenode->is_inter == 1 )
        {
        int rval = msrcnode->Rd_val;
        int myval = mmenode->Rd_val;
//      printf("my_val %d received value %d\n",myval,rval);
        if(rval == 0 || myval == 0 || rval < (myval-(myval/10)) || (rval >
        myval+(myval/10)))
        {
        printf("received value is less than the threshold
```

```
      value\n");
      printf("Me %d dropping packet\n",index);
       drop(p);
       return;
      }
  else
 {
 //    printf("Intermediate node %d received valid value - calculating
 aggregate and MAC\n",index);
 mmenode->Agg = (msrcnode->Rd_val + mmenode->Rd_val);
 mmenode->MAC = 0|3|mmenode-Agg|msrcnodeAgg|index;mmenode-
 >p_Agg[src]=msrcnode->Agg;
mmenode->p_MAC[src]=msrcnode->MAC;
//mmenode->flag = 0;
//     printf("MAC - %d Agg - %d flag - %d\n",mmenode->MAC,
mmenode->Agg, mmenode->flag);
ch->Rd_val = msrcnode->Rd_val;
ch->MAC = msrcnode->MAC;
ch->Agg = msrcnode->Agg;
ch->flag = 0;
}
}//inter checking
if(mmenode->is_leader == 1 )
{
 mmenode->Agg = msrcnode->Agg+mmenode->Rd_val;
 mmenode->MAC = 1|5|mmenode->Agg|msrcnode->Agg|index;
 mmenode->p_Agg[src]=msrcnode->Agg;
 mmenode->p_MAC[src]=msrcnode->MAC;
//mmenode->flag = 1;
//  printf("Leader node %d received valid value - calculating aggregate
```

```
 and MAC\n",index);
//     printf("MAC - %d Agg - %d flag - %d\n",mmenode->MAC,
mmenode->Agg, mmenode->flag);
ch->Rd_val = msrcnode->Rd_val;
 ch->MAC = msrcnode->MAC;
if(mmenode->chg_ == 1)
  ch->Agg = 1000;
else
   ch->Agg = msrcnode->Agg;
   ch->flag = 1;
}//leader checking
}//flag checking
// Added by Parag Dadhania && John Novatnack to handle broadcasting
 if ( (u_int32_t)ih->daddr() != IP_BROADCAST)
  rt_resolve(p);
 else
  {
        Node* snode;
        Node* mnode;
        int src = Address::instance().get_nodeaddr(ih->saddr());
        snode = Node::get_node_by_address(src);
        mnode = Node::get_node_by_address(index);
        MobileNode* msrcnode = (MobileNode*)snode;
        MobileNode* mmnode = (MobileNode*)mnode;
        forward((aodv_rt_entry*) 0, p, NO_DELAY);
}
}
```

REFERENCES

8. REFERENCES

- AKYILDIZ, I., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. Wireless sensor networks:A survey. Comput. Networks 38, 4 (March).

- CHAN, H., PERRIG, A., PRZYDATEK, B., AND SONG, D. 2007. SIA: Secure information aggregation in sensor networks. J. Comput. Secur. Special Issue on Adhoc and Sensor Networks.

- CHAN, H., PERRIG, A., AND SONG, D. 2006. Secure hierarchical in-network aggregation in sensor networks.In Proceedings of the 13th ACM Conference Computer and Communications Security (CCS'06) .

- DU, W., DENG, J., HAN, Y. S., AND VARSHNEY, P. K. 2003b. A witness-based approach for data fusion assurance in wireless sensor networks. In Proceedings of the Global Telecommunications Conference (GLOBECOM'03).

- FRANK, G. 1969. Procedures for detecting outlying observations in samples. Technometrics 11, 1

- HE, W., LIU, X., NGUYEN, H., NAHRSTEDT, K., AND ABDELZAHER, T. 2007. PDA: Privacypreserving data aggregation in wireless sensor networks. In Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'07).

- HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER,D., AND PISTER, K. 2000. System architecture directions for networked sensors. **In** Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'00).

- MADDEN, S., FRANKLIN, M. J., HELLERSTEIN, J. M., AND HONG, W. 2002. TAG: A tiny aggregation service for ad-hoc sensor networks. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI'02).

- MERKLE, R. 1989. A certified digital signature. In Proceedings of Advances in Cryptology - 9th Annual International Cryptology Conference (CRYPTO'89).

- NATH, S., GIBBONS, P., SESHAN, S., AND ANDERSON,Z.2004. Synopsis diffusion for robust aggregation in sensor networks. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04).