IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

*International Advanced Research Journal in Science, Engineering and Technology*
*Vol. 3, Issue 6, June 2016*

# Detection of Misbehaving Nodes in Ad-Hoc Networks

**P. Satyanarayana[1], K.V. Prasad[2], Yeluri. Sri Harsha[3]**

Asst Prof, Department of ECE, V. R Siddhartha Engineering College, Vijayawada[1, 2]

Project Student, Department of ECE, V. R Siddhartha Engineering College, Vijayawada[3]

**Abstract:** In wireless ad-hoc networks, a source node relies on intermediate nodes to forward data packets to a designated destination node. Wireless ad-hoc network is deployed in some hostile or uncontrolled environment where nodes may not behave according to the defined protocol. So, there will be a probability of presence of misbehavior nodes. Nodes may participate in route discovery or route maintenance process but refuse to forward packets due to presence of faulty hardware or software or to save their resources, such as, battery power and bandwidth. To avoid such problems, Detection of Node-Misbehavior using Overhearing is introducing. Detection and isolation of misbehavior nodes are important issues to improve the quality of communication service and to save resources of well behaving wireless nodes. In this work, a neighbor Overhearing based Misbehavior Detection (OMD) scheme is proposed. In OMD, each node overhears the transmissions of its neighbors and calculates packet forwarding ratio of its own as well as its neighbors. Source node uses the calculated information to identify a misbehaving node. The proposed scheme reduces communication overheads and identification delays to detect misbehaving nodes in wireless ad-hoc network. In this project, simulation is done by using ns3 software. Simulation results are presented to evaluate the performance of the proposed OMD scheme.

**Key words:** OMD, Ad-Hoc Networks.

## 1. INTRODUCTION

Wireless ad-hoc network is a decentralized network which is not supported by any pre-existing infrastructure. Nodes in this type of network work collaboratively to realize end to end communication.
Wireless based Networks have changed today's life greatly, while ad-hoc networks provide people more solutions and convenient due to its special property. Ad hoc networks are mainly formed by a group of devices. The devices are called "nodes" in ad hoc network.

## 2. WORMHOLE ATTACKS:

Wormhole attack is also known as tunneling attack. A tunneling attack is where two or more nodes may collaborate to encapsulate and exchange messages between them along existing data routes. This exploit gives the opportunity to a node or nodes to short-circuit the normal flow of messages creating a virtual vertex cut in the network that is controlled by the two colluding attackers. Lack of Co-operation Mobile ad-hoc networks rely on the co-operation of all the participating nodes. The more nodes co-operate to transfer traffic, the more powerful a MANET gets. A selfishness node want to preserve own resources while using the services of others and consuming their resources.

Major contributions of this work are as follows:
• An improved neighbor overhearing based misbehavior detection approach has been proposed to identify misbehaving nodes in the network.
• Simulation results are presented to show the efficacy of the proposed approaches.

## 3. IMPROVED NODE MISBEHAVIOR DETECTION

### NODE MISBEHAVIORS:

Identification of misbehaving nodes in ad hoc networks is critically important to detect security attack in the network. Two types of misbehaving nodes such as selfish and malicious nodes are taken into consideration in [1,2]. Selfish nodes do not intend to directly damage other nodes, but however, do not cooperate, saving battery life for their own communications. But malicious nodes do not give priority to saving battery life, and aim at damaging other nodes. It is introduced that two different types of selfish nodes.
As the nodes in MANETs are battery powered, energy becomes a precious resource, and thus, role of selfish nodes draws more attention. Thus, it is introduced altogether three routing behaviors of nodes in a MANET.
Type-0: well-behaved node: A well behaved node cooperates in the communication well, performs as required by the routing protocol, and equally participates in the communication activities like route discovery, maintenance, packet forwarding and receiving etc.

**Type-1:** active selfish node: Such a node does not participate in packet forwarding, and drops every received packet. It disables the packet forwarding mechanism for the packets which have a destination address, other than this selfish node. In fact, it helps the selfish node to save its own energy, thereby still contributing to network maintenance.

**Type 2:** passive selfish node: Such a node practically does nothing and stays idle in the network. It does not

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
*International Advanced Research Journal in Science, Engineering and Technology*
*Vol. 3, Issue 6, June 2016*

contribute to any of the activities like packet forwarding, receiving, route discovery, network maintenance. With respect to above mentioned misbehaving nodes, we evaluate the performance of DSDV, DSR and AODV routing protocols through extensive simulations, where a certain percentage of nodes behave as active and/or passive selfish nodes with the remaining nodes being well-behaved.

## OVERHEARING BASED MISBEHAVIOR DETECTION:

In this work, the proposed neighbor Overhearing Based Misbehavior Detection (OMD) technique uses the information provided by the neighbors for behavior evaluation of nodes in the path without incurring high communication overhead. An algorithm for overhearing based misbehavior detection is shown in (Algorithm guk1) Source node maintains a matrix φPFR of Packet Forwarding Ratio (PFR) of each node. The matrix contains node ID denoted as i, forwarding ratio calculated by its own denoted as PFRii and forwarding ratio calculated by its neighbors denoted as PFRij,where j denotes all the neighbors of node ni.

Each node calculates PFR of its own and its neighbors and these values are used to update the matrix φPFR of source (refer Algorithm).

IPV4 ROUTTING PROTOCOL**:** The base class defines two packet routing and forwarding. AODV discovers routes on demand. Therefore, the AODV model buffers all packets while a route request packet is disseminated. The AODV implementation can detect the presence of unidirectional links and avoid them if necessary. When source node wants to send a packet to some destination node, it uses the Dynamic Source Routing (DSR) algorithm to find the route to the destination node. In DSR, when source node has a packet for destination node, it checks whether a route exists in its cache.

If route exists in the route cache, source node uses the route to send packets to the destination node. If a route does not exist, source node broadcasts a Route Request (RREQ) packet. This packet contains the source ID, destination ID, and the Time-To-Live (TTL). Any intermediate node that receives the RREQ, appends its ID to the RREQ packet and rebroadcasts it while decreasing the TTL field by one unit. If a receiving node is the destination node, it responds to source node with a Route Reply (RREP) packet containing the entire path PS−>D from source node to destination node.

Source adds the route to route cache for further use. Source node sends the data packets to the destination node using the route determined. Source node continuously monitors the performance of path PS−>D based on the packet delivery ratio η. If it detects that this ratio is less than some predefined threshold value η0 (refer Algorithm), it will send a control packet Req PFR at time 't' to request all the intermediate nodes to calculate Packet Forwarding Ratio (PFR) of their own and their neighbors' based on the number of packets forwarded Ps and number of packets received Pr for a time duration T1.

When this control packet reaches to the destination node, it generates a control packet RepPFR after time duration T1.Destination node calculates PFR of itself and its neighbors and appends this value to the control packet Req PFR and sends the packet in reverse direction to the source node . All intermediate nodes append their calculated PFR value to the packet and send it to their previous node. Source node updates its matrix based on the values in the control packet Rep PFR. Source node checks the matrix and finds the node for which calculated values of Packet Forwarding Ratio conflict with each other and declares that node as misbehaving node.

### 4.2 Algorithm:

The following algorithm is used to detect the misbehavior node:

**Require:** Forwarding ratio PFR of each node, Time epoch t and t1

**Ensure:** Detection of misbehaving node

```
1. Initialize matrix φ PFR of Packet Forwarding Ratio (PFR) with zeros.
2. Selection of path PS−>D using DSR algorithm between source node S and destination node D.
3. Monitoring packet delivery ratio η of the path PS−>D by node S.
4. If η < η0
5. Node S sends a control packet ReqPFR to request all the nodes along the path to calculate PFR of its own PFRI_ii and its neighbors PFR_ji (where j denotes all the neighbors of node i) at time t for a time duration t1.
6. Calculation of PFRii and PFRji by the intermediate node.
7. At time t+t1 node D sends a reply RepPFR along the reverse path and all intermediate nodes append the calculated values to the reply.
8. Node S modifies it's matrix φPFR based on the received RepPFR
9. Node S checks all the entries
10. If PFRii_= PFRij.
11. Node i is misbehaving node.
12. Else
13. Node i is wellbehaving node.
```

## 4. SIMULATION RESULTS

### 4.1 CREATION OF NODES

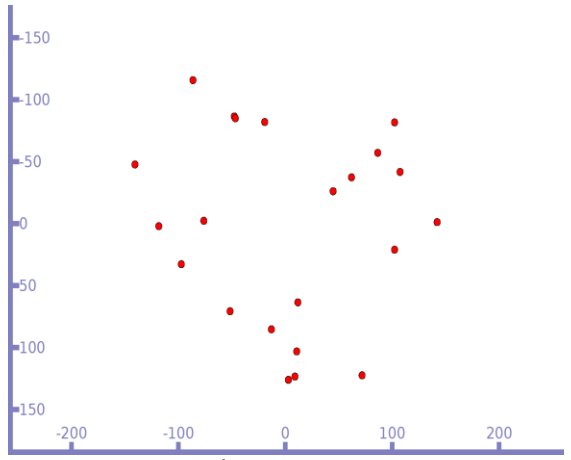
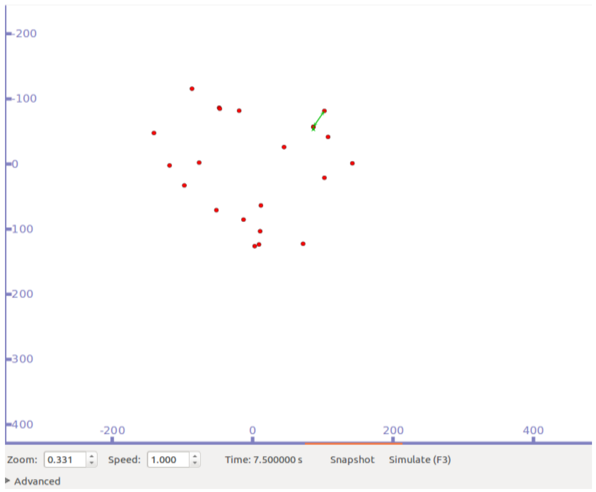


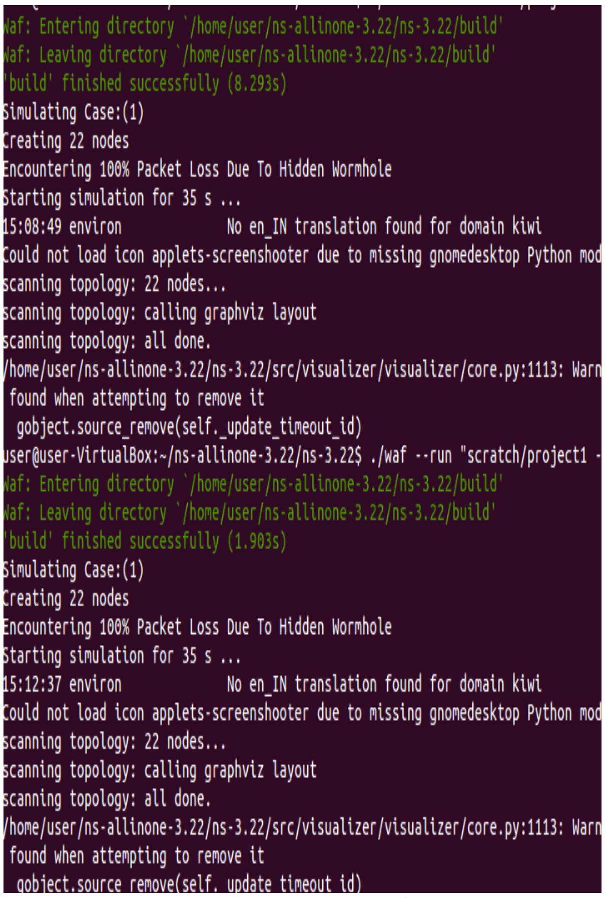
Figure 1: Creation of nodes (red dot indicates nodes).

## 4.2 CASE 1: HIDDEN PASSIVE ATTACK (100% PACKET LOSS)



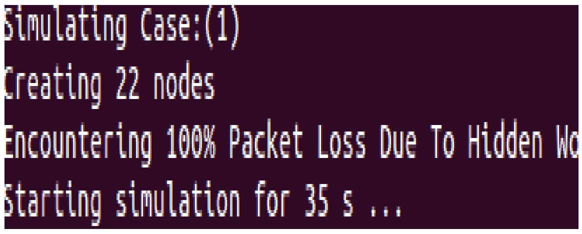
Zoom: 0.331 Speed: 1.000 Time: 7.500000 s Snapshot Simulate (F3)
Advanced

Figure 2: Simulation given to few nodes. (Green line shows packet flow).



```
af: Entering directory `/home/user/ns-allinone-3.22/ns-3.22/build'
af: Leaving directory `/home/user/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (8.293s)
Simulating Case:(1)
Creating 22 nodes
Encountering 100% Packet Loss Due To Hidden Wormhole
Starting simulation for 35 s ...
15:08:49 environ              No en_IN translation found for domain kiwi
Could not load icon applets-screenshooter due to missing gnomedesktop Python mod
scanning topology: 22 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
/home/user/ns-allinone-3.22/ns-3.22/src/visualizer/visualizer/core.py:1113: Warn
 found when attempting to remove it
  gobject.source_remove(self._update_timeout_id)
user@user-VirtualBox:~/ns-allinone-3.22/ns-3.22$ ./waf --run "scratch/project1 -
af: Entering directory `/home/user/ns-allinone-3.22/ns-3.22/build'
af: Leaving directory `/home/user/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (1.903s)
Simulating Case:(1)
Creating 22 nodes
Encountering 100% Packet Loss Due To Hidden Wormhole
Starting simulation for 35 s ...
15:12:37 environ              No en_IN translation found for domain kiwi
Could not load icon applets-screenshooter due to missing gnomedesktop Python mod
scanning topology: 22 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
/home/user/ns-allinone-3.22/ns-3.22/src/visualizer/visualizer/core.py:1113: Warn
 found when attempting to remove it
  gobject.source_remove(self._update_timeout_id)
```

Figure 3: Simulation result for case1



```
Simulating Case:(1)
Creating 22 nodes
Encountering 100% Packet Loss Due To Hidden Wo
Starting simulation for 35 s ...
```

Figure 4: 100% packet loss in Hidden passive attack .

## 4.3 CASE2: EXPOSED PASSIVE ATTACK (100%PACKET LOSS)

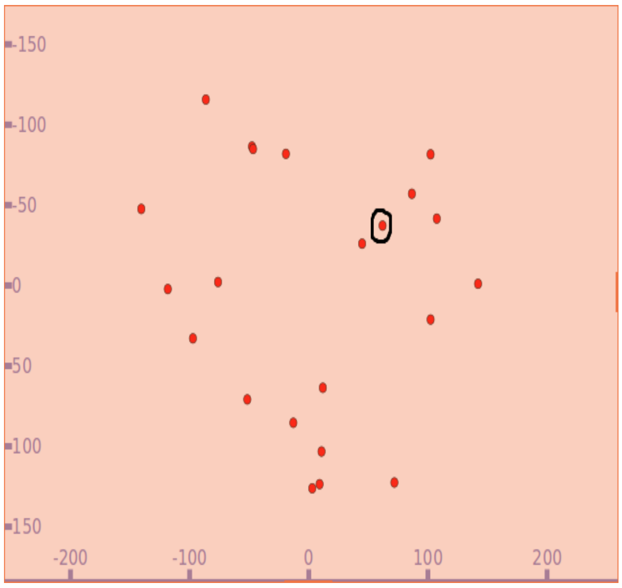
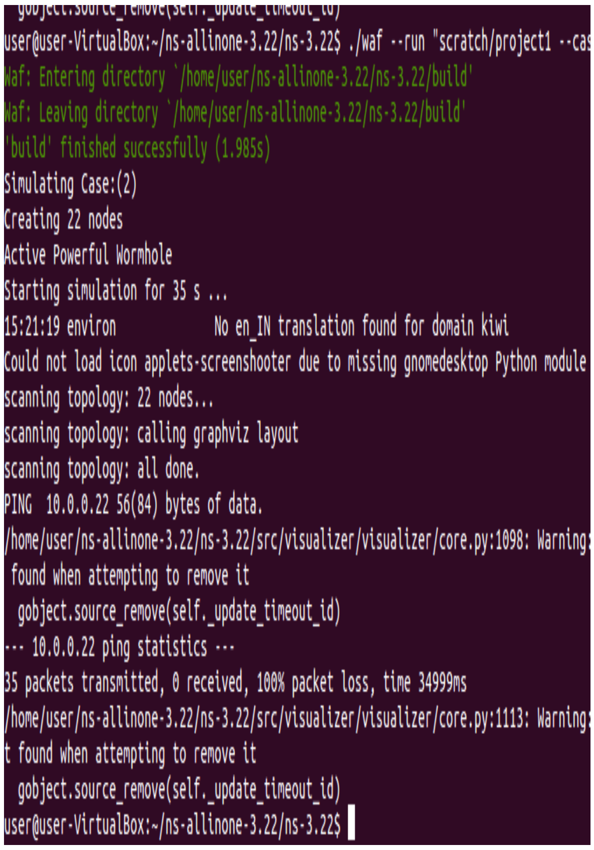
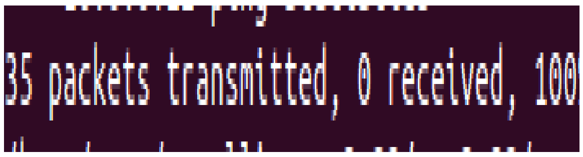


Figure 5: Misbehaving node is exposed with 100% packet loss.



```
user@user-VirtualBox:~/ns-allinone-3.22/ns-3.22$ ./waf --run "scratch/project1 --cas
Waf: Entering directory `/home/user/ns-allinone-3.22/ns-3.22/build'
Waf: Leaving directory `/home/user/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (1.985s)
Simulating Case:(2)
Creating 22 nodes
Active Powerful Wormhole
Starting simulation for 35 s ...
15:21:19 environ              No en_IN translation found for domain kiwi
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 22 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
PING  10.0.0.22 56(84) bytes of data.
/home/user/ns-allinone-3.22/ns-3.22/src/visualizer/visualizer/core.py:1098: Warning:
 found when attempting to remove it
  gobject.source_remove(self._update_timeout_id)
--- 10.0.0.22 ping statistics ---
35 packets transmitted, 0 received, 100% packet loss, time 34999ms
/home/user/ns-allinone-3.22/ns-3.22/src/visualizer/visualizer/core.py:1113: Warning:
t found when attempting to remove it
  gobject.source_remove(self._update_timeout_id)
user@user-VirtualBox:~/ns-allinone-3.22/ns-3.22$
```

Figure 6: Simulation result for case 2.

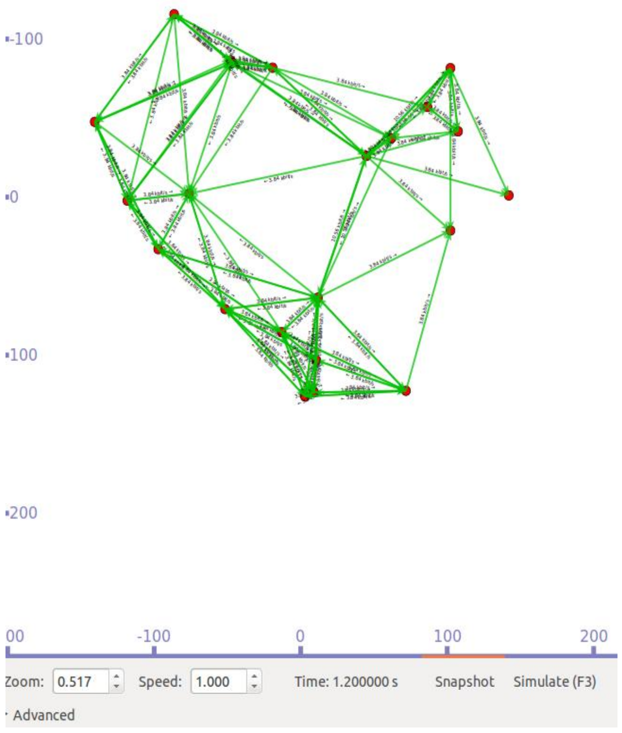


```
35 packets transmitted, 0 received, 100
```

Figure 7: Output terminal shows 100% packet loss in exposed passive attack.

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

## 4.4 CASE 3: HIDDEN ACTIVE ATTACK (CERTAIN % PACKET LOSS)



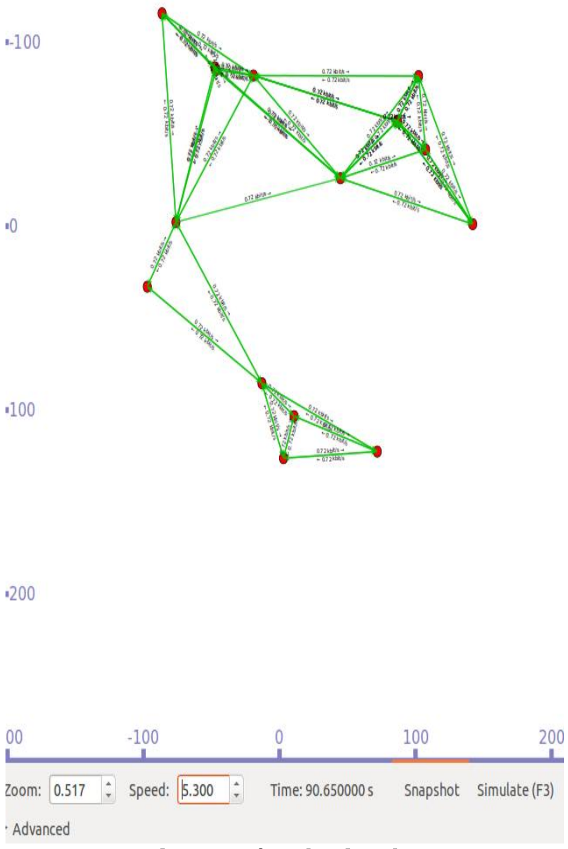
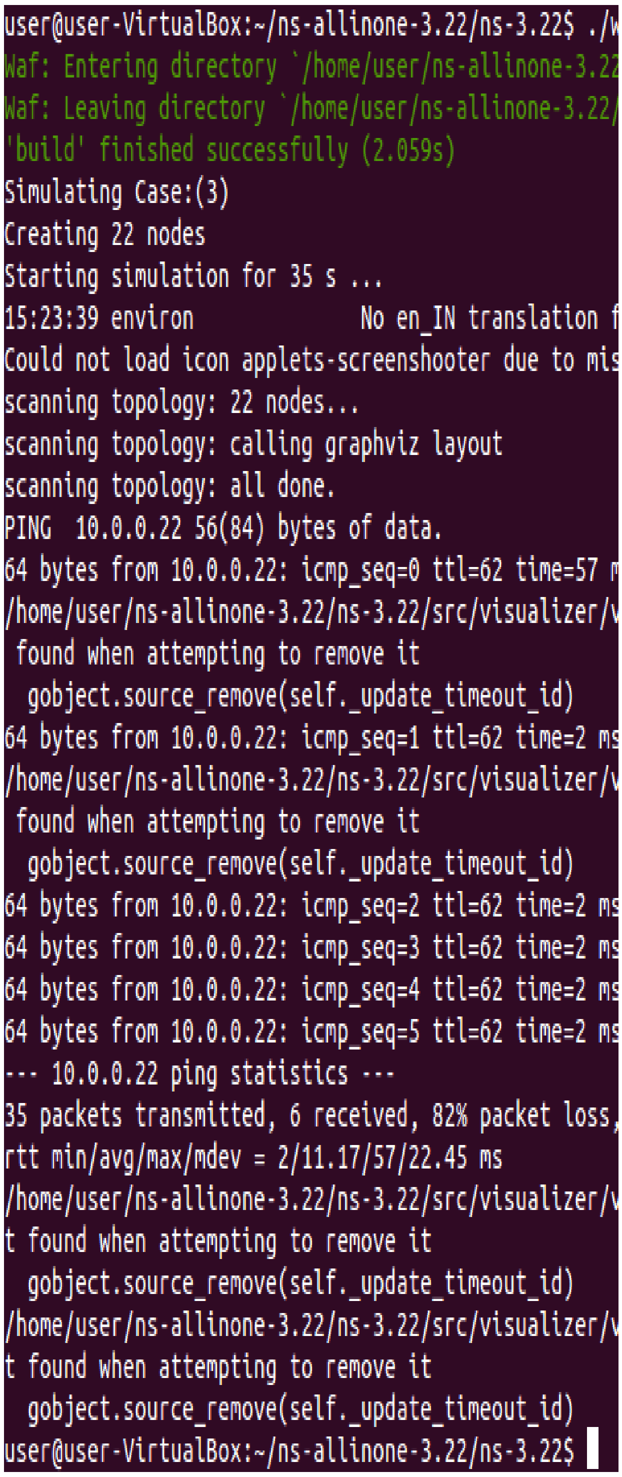
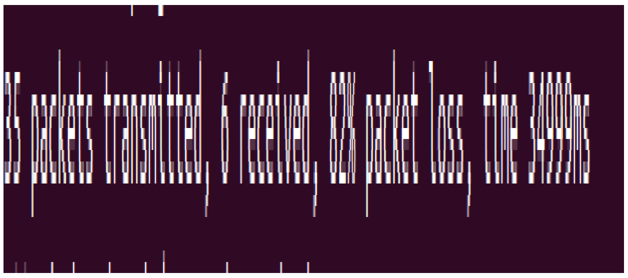


Figure 8: Reduction of packet loss by 18 percent.



```
user@user-VirtualBox:~/ns-allinone-3.22/ns-3.22$ ./w
Waf: Entering directory `/home/user/ns-allinone-3.22
Waf: Leaving directory `/home/user/ns-allinone-3.22/
'build' finished successfully (2.059s)
Simulating Case:(3)
Creating 22 nodes
Starting simulation for 35 s ...
15:23:39 environ              No en_IN translation f
Could not load icon applets-screenshooter due to mis
scanning topology: 22 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
PING  10.0.0.22 56(84) bytes of data.
64 bytes from 10.0.0.22: icmp_seq=0 ttl=62 time=57 m
/home/user/ns-allinone-3.22/ns-3.22/src/visualizer/v
 found when attempting to remove it
  gobject.source_remove(self._update_timeout_id)
64 bytes from 10.0.0.22: icmp_seq=1 ttl=62 time=2 ms
/home/user/ns-allinone-3.22/ns-3.22/src/visualizer/v
 found when attempting to remove it
  gobject.source_remove(self._update_timeout_id)
64 bytes from 10.0.0.22: icmp_seq=2 ttl=62 time=2 ms
64 bytes from 10.0.0.22: icmp_seq=3 ttl=62 time=2 ms
64 bytes from 10.0.0.22: icmp_seq=4 ttl=62 time=2 ms
64 bytes from 10.0.0.22: icmp_seq=5 ttl=62 time=2 ms
--- 10.0.0.22 ping statistics ---
35 packets transmitted, 6 received, 82% packet loss,
rtt min/avg/max/mdev = 2/11.17/57/22.45 ms
/home/user/ns-allinone-3.22/ns-3.22/src/visualizer/v
t found when attempting to remove it
  gobject.source_remove(self._update_timeout_id)
/home/user/ns-allinone-3.22/ns-3.22/src/visualizer/v
t found when attempting to remove it
  gobject.source_remove(self._update_timeout_id)
user@user-VirtualBox:~/ns-allinone-3.22/ns-3.22$
```

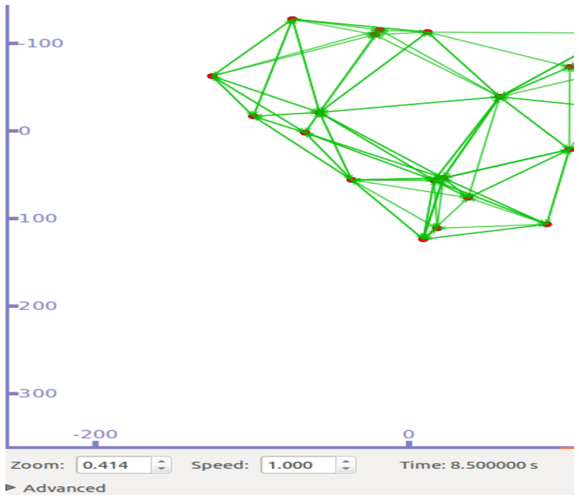
Figure 9: Simulation result for case 3.



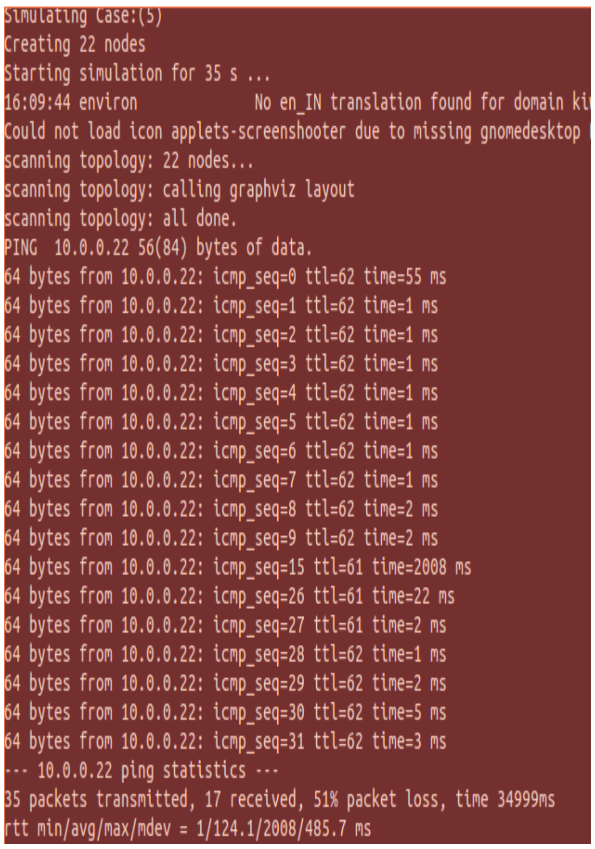
```
35 packets transmitted, 6 received, 82% packet loss, time 34999ms
```

Figure 10: Output terminal shows reduced packet loss by 18 percent.

### 4.5 CASE 4: HANDLES WITH RATE ADAPTION ALGORITHM (LESS PACKET LOSS)



Zoom: 0.414 Speed: 1.000 Time: 8.500000 s



```
Simulating Case:(5)
Creating 22 nodes
Starting simulation for 35 s ...
16:09:44 environ             No en_IN translation found for domain ki
Could not load icon applets-screenshooter due to missing gnomedesktop
scanning topology: 22 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
PING  10.0.0.22 56(84) bytes of data.
64 bytes from 10.0.0.22: icmp_seq=0 ttl=62 time=55 ms
64 bytes from 10.0.0.22: icmp_seq=1 ttl=62 time=1 ms
64 bytes from 10.0.0.22: icmp_seq=2 ttl=62 time=1 ms
64 bytes from 10.0.0.22: icmp_seq=3 ttl=62 time=1 ms
64 bytes from 10.0.0.22: icmp_seq=4 ttl=62 time=1 ms
64 bytes from 10.0.0.22: icmp_seq=5 ttl=62 time=1 ms
64 bytes from 10.0.0.22: icmp_seq=6 ttl=62 time=1 ms
64 bytes from 10.0.0.22: icmp_seq=7 ttl=62 time=1 ms
64 bytes from 10.0.0.22: icmp_seq=8 ttl=62 time=2 ms
64 bytes from 10.0.0.22: icmp_seq=9 ttl=62 time=2 ms
64 bytes from 10.0.0.22: icmp_seq=15 ttl=61 time=2008 ms
64 bytes from 10.0.0.22: icmp_seq=26 ttl=61 time=22 ms
64 bytes from 10.0.0.22: icmp_seq=27 ttl=61 time=2 ms
64 bytes from 10.0.0.22: icmp_seq=28 ttl=62 time=1 ms
64 bytes from 10.0.0.22: icmp_seq=29 ttl=62 time=2 ms
64 bytes from 10.0.0.22: icmp_seq=30 ttl=62 time=5 ms
64 bytes from 10.0.0.22: icmp_seq=31 ttl=62 time=3 ms
--- 10.0.0.22 ping statistics ---
35 packets transmitted, 17 received, 51% packet loss, time 34999ms
rtt min/avg/max/mdev = 1/124.1/2008/485.7 ms
```

Figure 11: Simulation result for case 4 shows 51% packet loss.

## 5. CONCLUSION

The frame work for 22 nodes (assumption) was created in ad-hoc networks using NS3 software. Each node is simulated and a program is written for specific nodes to act as malicious node. Using OMD technique, misbehavior nodes were detected and packet forward ratio is determined using simulation table. The OMD technique reduces the identification delay by transmitting few control packets by identifying the misbehaving nodes in the network. From the experience of this, the following recommendation on future work is suggested: This work can be extended by giving mobility to each node in the network.

## REFERENCES

1. A. Visconti and H. Tahayori, "Detecting misbehaving nodes in manet with an artificial immune system based on type-2 fuzzy sets," International Conference for Internet Technology and Secured Transactions, ICITST'09, pp. 1–2, 2009.
2. H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-securing ad-hoc wireless networks," in In Proc. of ISCC, 2002.R. R. Rout and S. K. Ghosh, "Enhancement of lifetime using duty cycle and network coding in wireless sensor networks," IEEE
3. Transaction on Wireless Communication, vol. 12, no. 2, pp. 656–667, 2013.
4. B. Kannhavong, H. Nakayama, Y. Nemoto, N. Kato, and A. Jamalipour, "A survey of routing attacks in mobile ad hoc networks," IEEE Wireless Communications, vol. 14, no. 5, pp. 85–91, 2007.
5. A. B. Abderrahmane Baadache, "Fighting against packet dropping misbehavior in multi-hop wireless ad hoc networks," J. Network and Computer Applications, vol. 35, no. 3, pp. 1130–1139, 2012.
6. D. Djenouri and N. Badache, "On eliminating packet droppers in manet: A modular solution," Ad Hoc Networks, vol. 7, no. 6, pp. 1243–1258, 2009.
7. F. Kargl, a. Klenk, M. Weber, and S. Schlott, "Advanced detection of selfish or malicious nodes in ad hoc networks," in 1st European Workshop on Security in Ad-hoc and Sensor Networks, Heidlberg, Germany, Aug 5-6, 2004, 2004.
8. K. Liu, J. Deng, P. Varshney, and K. Balakrishnan, "An acknowledgement based approach for detection of routing misbehavior in manets," IEEE Transactions on Mobile Computing, vol. 6, no. 5, pp. 536–550, 2007.
9. J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring, "Modelling incentive for collabration in mobile adhoc network," in Proc. of WiOpt, 2003.
10. S. Ganeriwal, L. Balzano, and M. Srivastava, "Reputation based framework for high integrity sensor networks," ACM Transactions on Sensor Networks, vol. 4, no. 3, pp. 1
11. –37, 2008.