A POWER EFFICIENT METHOD TO PREVENT SYBIL ATTACK IN MOBILE AD-HOC NETWORK USING REPUTATION SYSTEM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology
in
Computer Science and Engineering

By:

Sanjeeb Kumar Padhan

108CS027

Soumya Ranjan Mund

108CS062

Under the guidance of

Prof. M.N. Sahoo

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Department of Computer Science and engineering
May, 2012

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "**A POWER EFFICIENT METHOD AGAINST MISBEHAVING NODE IN REPUTATION SYSTEM TO PREVENT SYBIL ATTACK IN MOBILE AD-HOC NETWORK**" submitted by **Sanjeeb Kumar Padhan(108CS027)** and **Soumya Ranjan Mund(108CS062)** in partial fulfillment of the requirements for the completion of Bachelor of Technology Degree in Computer Science and Engineering at the National Institute of Technology, Rourkela is an authentic work carried out by them under my supervision and guidance .To the best of my knowledge, Neither this thesis or any part or it has been submitted for any degree or diploma award elsewhere.

Date:
Place: NIT Rourkela

Prof. Manmath Narayan Sahoo
Dept. of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela – 769008

Acknowledgement

With great satisfaction and pride we present our thesis on the project under the “Research Project” paper during Final Year, for partial fulfillment of our Bachelor of Technology degree in Computer Science and Engineering at NIT Rourkela.

We are thankful to Prof. Manmath Narayan Sahoo for being the best guide and advisor for this research work in every field we have taken to complete our requirement. His ideas and inspirations have helped us make this nascent idea of ours into a fully-fledged project. Without presence of we may be researchers till now.

Again we are thankful to our batch-mates and Mr. Alekh Kumar Mishra to support us in our implementation part sometime. We are also grateful to all the professors of our department especially for being a constant source of inspiration and motivation during the course of the project.

Last but not the least we dedicate this project to our parents and family members, without their moral support, motivation and inspiration we could not complete this any way.

So we thankful again to all who are being a part of our Final year research project.

Sanjeeb Kumar Padhan(108CS027)
Soumya Ranjan Mund(108CS062)
Department of Computer Science and Engineering
NIT Rourkela

Abstract

Mobile ad-hoc network has become a very important field of study for students and researchers owing to its wide application. In mobile ad-hoc network all nodes are responsible for routing and forwarding of packets, hence all nodes are required to act selflessly for proper functioning of mobile ad-hoc network. The presence of selfish behavior in a node can degrade the performance of the mobile ad hoc network to a large extent. Several works have been done for identification and punishment of the misbehaving nodes in mobile ad hoc network. We propose here a method where some selected neighbors are participated in detecting misbehaving nodes in power effective manners. These neighbors participating in selfish node detection are chosen randomly. It also alerts all other nodes about the misbehaving links in the network. The simulation studies show that this does the job efficiently with less power consumption in the network. The power effectiveness of the algorithm also reduces the number of misbehaving nodes because many nodes show misbehavior to save their power.

CONTENT

LISTS OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction to MANET

Mobile Ad-Hoc Networks (referred to as MANETs)), are wireless networks for communication. These are increasingly utilized in the Commercial, Military, and Private sector as portable wireless computers have become more and more accessible. Mobile Ad-Hoc Networks allow users for access and exchange of information independent of their geographic location or proximity to infrastructure [7]. Opposed to the infrastructure networks, all nodes in MANETs are mobile and they have dynamic connections. Unlike others mobile, MANETs do not need a fixed infrastructure. This offers a decentralized character to the network. Because of decentralization, the networks are more flexible and robust.

MANETs are widely ranging used in many critical situations: search and rescue operation is an ideal application. Such cases have characteristics of lack of installed infrastructure for communications. This may be due to all of the equipment's were destroyed, or may be because the region is too remote. Rescuers must be able to communicate between them in order to make the best use of their energy, also to maintain safety. By automatic establishment of a data network with the equipment's for communications that the rescuers are carrying already, their job becomes easier.

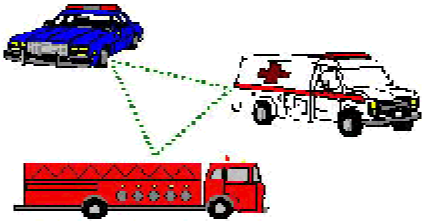


Fig 1.1 Mobile ad hoc network among Vehicles

A commercial application for MANETs includes ubiquitous computing. By using computers to forward data for others, data networks can be extended far beyond the usual reach of any installed infrastructure. Networks can be made more widely available and easier to use.

Another application of MANETs is sensor networks. This is a network made up of a very large number of small sensors. These can be used to detect various properties of an area. For example: temperature, pressure, toxins, pollutions, etc. The capabilities of each sensor are very limited, and each relies on others to forward data to a central computer. Individual sensors are constrained in their computing capability and are prone to failure and loss. Mobile ad-hoc sensor networks could be the key to future homeland security.

However MANETs are not perfect. The challenges of scalability, mobility, bandwidth limitations, and power constraints of these networks have not been completely eliminated to date.

At the center of these difficulties with MANETs are issues concerning the determination of the rules (protocols) governing the communication between the entities (nodes) in the network, One important question is how to facilitate the dynamic discovery of the most efficient route between two nodes within the network. It is important to take care of the mobility of the nodes and the lack of a fixed topology in the network.

1.2 Introduction to Sybil attack

The **Sybil attack** in computer security is an attack wherein a reputation system is subverted by forging identities in peer-to-peer networks. It is named after the subject of the book *Sybil*, a case study of a woman with multiple personality disorder. The name was suggested in or before 2002 by Brian Zill at Microsoft Research. The Sybil attack can be used to attack several types of protocols in wireless ad-hoc networks as described in varieties of literatures.

It is possible to control Internet polls by using multiple IP addresses to submit votes, to gain advantage in any results of a chain letter- a well-known and potentially major problem in real-world elections. A Sybil attack is also used by companies to increase the Google PageRank rating of the pages of their customers. Some particular search terms can be linked to unexpected results for political gain. Reputation systems are usual targets for Sybil attackers including real-world systems like eBay .Spammers can use Sybil attack to gain access to multiple accounts on free email systems. Peer-to-peer computing systems using voting to verify correct answers, like SETI@home, are also prone to accept false solutions from a Sybil attacker. Ad hoc mobile network routing can be manipulated when a Sybil attacker appears to be many different mobile nodes at once. In systems that provide anonymity between peers, as Tor, the Sybil attack is generally able to reveal the initiator of a connection and there is no defence against this attack. It also allows free riding in services in cooperative file storage systems such as Pastiche.

CHAPTER 2

REPUTATION SYSTEM AS A SOLUION TO SYBIL ATTACK

2.1 Existing Solutions to Sybil Attack

Several approaches have been proposed in various research papers against Sybil attack. The following graph shows the summery [10].

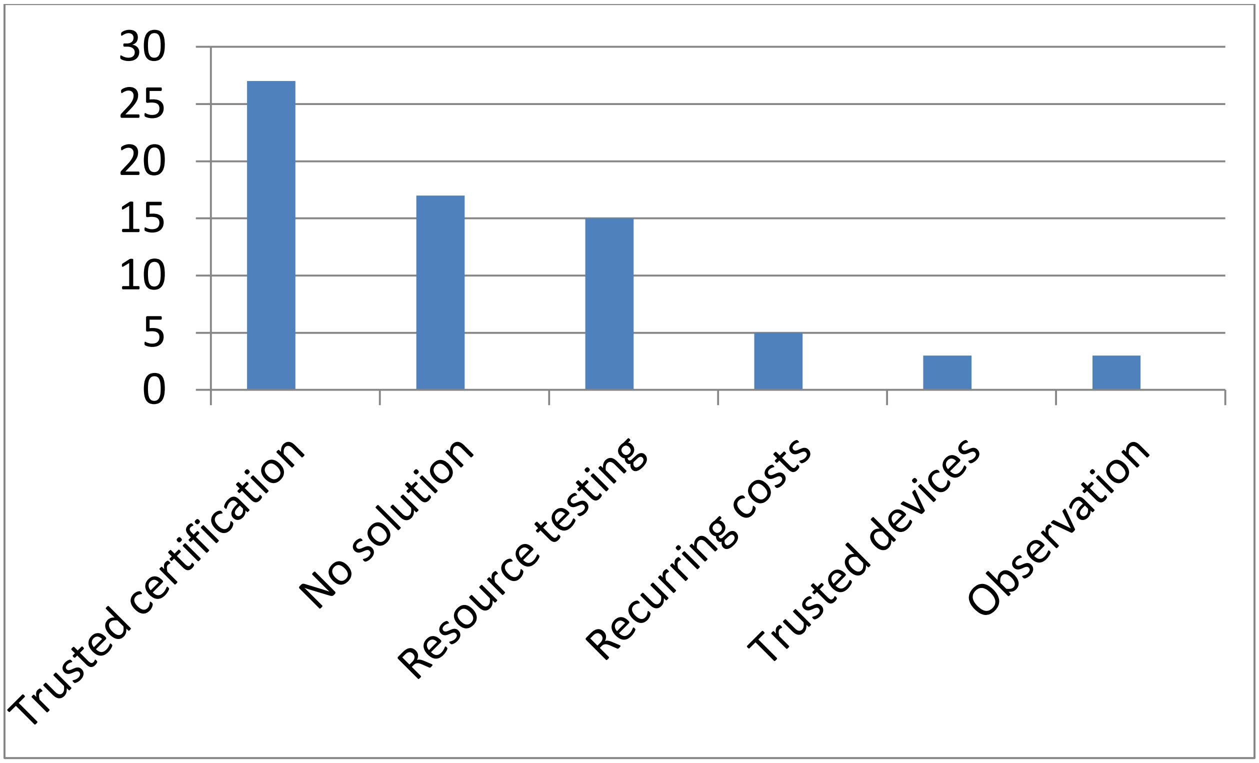


Fig 2.1 General solution approach vs. no. of citations

2.1.1 Trusted Certification-

It has proven that trusted certification is the only approach which is potent to completely eradicate Sybil attacks. That is why; it is cited as the most common solution. However, trusted certification is dependent on a centralized authority whose job is to ensure that each entity is assigned exactly one identity which is indicated by possession of a certificate. Factually, the author does not offer any method for ensuring this uniqueness, and in practical scenarios it must be done manually or by in-person process. This can be costly to create a performance bottleneck in large-scale systems. Moreover, to become effective, the certifying authority must ensure that lost or stolen identities are discovered and revoked [10]. If the performance and security implications can be solved, then this approach can eliminate the Sybil attack [10].

2.1.2 No Solution

Though many researchers know Sybil attack as a potential problem, they present no solution to it in their work, and there are many papers published to cite this.

2.1.3 Resource Testing

The aim of resource testing is to try to find out if a number of identities possess fewer resources than they would possess if they were independent. These tests consist of checks for computational ability, storage capacity, and network bandwidth, also limited IP addresses. **Cornelli et al**. and **Freedman** and **Morris** mainly propose testing for IP addresses in various domains or autonomous systems. Needing heterogeneous IP addresses although provide prevention for some attacks but fail to discourage others (such as zombie networks) and limits the use domain of an application. Douceur has proven the ineffectiveness of resource tests, but many suggest them as a minimal Sybil attack defence. For a variety of applications this is not sufficient if an attacker can get enough identities for causing a successful attack, though expensive. In the Tor communication system, for example, only two identities are required for an attack on anonymity [10]. In a type of resource test, **Yu et al.**'s Sybil-Guard technique relies on limited availability of real-world friendship edges between nodes. However, the p2p application in use may have little intersection with the real-world friends represented in the graph. These friendship relationships can also be expensive to build since the proposal needs out-of-band key sharing and a stronger trust relationship than is typical in social networks.

2.1.4 Recurring Costs and Fees

A variation of resource testing, in many papers identities are re-validated periodically using resource tests. The approach puts a limit on the number of Sybil attackers with limited resources can introduce in a time period. However, in many applications very few Sybil identities are required for an effective attack. Also in these papers, computational power is tested. Computational power mostly involves a one-time cost (for example, the purchase of computing hardware), so an attacker could recover over time even a high initial cost of claiming a large number of identities. **Awerbuch** and **Scheidler** suggest the use of Turing tests, for example CAPTCHAs, to impose recurring fees. **Dragovic et al.** require certification of identities, but this certification is not trusted; rather, it is viewed as a way of imposing identity creation costs. **Gatti et al**.'s "**Sufficiently Secure Peer-to-Peer Networks"** uses an economic, game-theoretical approach to examine when attacks on censorship resistant networks are cost-effective. For many applications, recurring fees can incur a cost to the Sybil attack that increases linearly with the total number of identities participating.

2.1.5 Trusted Devices

Similar to certification authorities, entities in an application can be linked in to a particular hardware device securely. Analogous to any central authority handing out cryptographic certificates, there are no special methods of preventing an attacker from obtaining multiple devices other than manual intervention. However, cost of acquiring multiple devices may be high.

2.1.5 Application Domains :---

2.1.5.1 Mobile Networks

Observing location can distinguish between different devices and limits of realistic mobility can constrain attacker movement. For an attacker having a single device, all Sybil identities will always appear to move together. The defence is not applicable beyond mobile networks, and it does not protect against a single entity controlling multiple devices, each having a non-recurring cost.

2.1.5.2 Auditing

In some cases, audit can be used to determine the correctness of identity behaviour. If audit is cheap, the Sybil attack has little benefit: for instance, a large number of apparently independent identities cannot successfully convince another entity that they have factored a large number unless they have actually done so.

2.1.5.3 Cash Economies

In these, identities explicitly exchange currency for desired goods or services. In most cases, such applications are not susceptible to the Sybil attack, since they do not rely on redundancy.

2.1.5.4 Reputation Systems

For many p2p systems, including ad hoc networks and online markets, reputation systems have received a significant amount of attention as a solution for mitigating the effects of malicious peers. In an important work, **Cheng** and **Friedman** evaluated the vulnerability of reputation systems to the Sybil attack, classifying them as **symmetric** or **asymmetric** approaches.

2.1.5.5 Symmetric Reputation

A symmetric reputation system is one in which an identity’s reputation depends solely on the topology of the trust graph, and not the naming or identity of nodes. An attacker that wishes to increase its reputation simply uses Sybil identities to create a copy of the existing graph representing trust relationships. A symmetric reputation system cannot distinguish original nodes from the copies, and thus some Sybil node has reputations equal or better to any original node.

2.1.5.6 Asymmetric Reputation Systems

In asymmetric reputation systems, there are specifically trusted nodes from which all reputation values propagate. Alternatively, each entity separately computes a trust value along their unique paths to every other identity in the system. Since the trusted nodes cannot be impersonated, no Sybil attacker can create a duplicate graph as explained in the symmetric case. This trust value can change over time as the entity interacts with and observes the behaviour of different identities. Asymmetric reputation systems can be effective at raising the cost of Sybil attacks because attackers are forced to build up trust before effectively launching attacks. Unfortunately, these systems inevitably penalize newcomers who must prove themselves by offering benefits before getting anything in return.

2.2 Role of Reputation System-

Sonja Buchegger, University of California at Berkeley Jean-Yves Le Boudec, EPFL-IC-LCA in their paper published in IEEE Communications Magazine • July 2005 suggestif Self-polishing of mobile ad hoc network by réputation system. Reputation systems can be used to cope with any kind of misbehavior as long as it is observable.

The goal of a detection and reputation system is to enable nodes to adapt to changes in the network environment caused by misbehaving nodes. This is achieved by the following functions.

- **Monitoring**
- **Reputation**
- **Response**

2.2.1 Monitoring

Monitoring systems detect misbehavior that can be distinguished from regular behavior by observation. Nodes can automatically learn about new misbehavior in analogy to the human immune system.

2.2.2 Reputation

The terms reputation and trust have been used for various concepts, also synonymously. Reputation here is to mean the performance of a node in participating in the base protocol as seen by other nodes. For mobile ad hoc networking this means participation in routing and forwarding. By trust we mean the performance of a node in the policing protocol that protects the base protocol, here reliability as a witness to provide honest reports.

2.2.3 Response

Detection and reputation systems aim at isolating nodes that are deemed misbehaving by not using them for routing and forwarding, and most also isolate them additionally by denying them service.

2.3 Features of a Reputation System

2.3.1 Representation of Information and classification:-

These determine how monitored events are stored and translated into reputation ratings, and how ratings are classified for response. Use of second-hand information, Reputation systems can either rely exclusively on their own observations or also consider information obtained by others.

2.3.2 Trust:-

The use of trust influences the decision of using second-hand information. The design choices are about how to build trust, out-of-band trust vs. building trust on experience, how to represent trust, and how to manage the influence of trust on responses.

2.3.3 Redemption and secondary response:-

When a node has been isolated, it can no longer be observed. The question of how those nodes should be rated over time is addressed by these two features. If the misbehavior of a node is temporary, a redemption mechanism ensures that it can come back to the network. That is, however, desirable to prevent recidivists from exploiting a redemption mechanism. This can be achieved by secondary response, meaning a quicker response to a recurring threat, in analogy to the human immune system.

2.3.4 Liar Detection

In this scenario nodes not only misbehave in forwarding (and routing), but also in the reputation system itself, by spreading spurious ratings. Untrustworthy nodes can have different strategies to publish their falsified first-hand information when attempting to influence reputation ratings (e.g., when they want to discredit regular nodes).If the lies are big, they will not pass the deviation test of CONFIDANT. A more sophisticated alternative is stealthy lies. Although nodes do not know the content of the reputation ratings held by others, they could try to infer from published first-hand information and then lie only enough to just pass the deviation test. CORE does not consider negative ratings, so only flattering has an impact. SORI are vulnerable to liars that are cooperative when forwarding. Context-aware detection copes with single liars or very small collusions by majority voting. Path-rater has no defense against liars.

CHAPTER 3

RSNAM (RANDOM SELECTION OF NEIGHBORS FOR MONITORING WITH ALERT MECHANISM)

There are many algorithms are existing in different literatures for implementing reputation system in mobile ad hoc network. These have been implemented as an add-on to the DSR [Dynamic Source Routing] routing protocol. In MANET [Mobile Ad-hoc network] the nodes have to cooperate to find path between nodes [route discovery, route maintenance etc.].The successful design of a reputation system is decided by how the system is free from misbehaving nodes where misbehaviors are packet dropping, identity spoofing and packet modification.

3.1 Existing algorithms:

3.1.1 WATCHDOG & PATHRATER-Marti proposed this in 2000[4, 18]:

In this each node contains two components watchdog and path-rater. Each node operates in promiscuous mode. It maintains a buffer of recently sent packets, compares it with overheard packet, if same removes from buffer else decrements the reputation value[of next hop node] after a timeout. If the reputation falls below the threshold the node is considered selfish and the path-rater relieves the node from burden of forwarding packet.

3.1.2 CONFIDANT (Cooperation of Nodes: Fairness in Dynamic Ad hoc NeTwork) - Buchegger proposed this in 2002[1, 4, 12]:

The protocol adds a trust manager and a reputation system to the watchdog and path rater scheme. The trust manager evaluates the events reported by the watchdog and issues alarms to warn the nodes in the friends list regarding malicious nodes

3.1.3 CORE (Collaborative Reputation mechanism)-Michiardi proposed this in 2002[1, 4, 19]:

The reputation metric is computed based on data monitored by the local entity and some information provided by the other nodes involved in each operation. The reputation value is proportional to amount of resources the node can utilize.

3.1.4 OCEAN (Observation based Cooperation Enforcement in Ad hoc Networks)-Bansal proposed this in 2003[1, 20]:

The routing decisions are based on direct observation of neighboring nodes behavior and it completely disallows the exchange of second hand reputation. It also employs a punishment scheme by completely rejecting the traffic from the misbehaving nodes.

3.1.5 Locally Aware Reputation System (LARS)-Hu proposes this in 2006[14]:

Reputation of nodes is derived by using direct observation. When a node detects a packet drop behavior of its neighbor then the reputation value of the neighbor gets decremented. When a selfish node is identified then its k-hop neighbors become aware of the selfishness, where k is a parameter which is adaptive to the security requirement of the network. To avoid false accusation and the associated trust issues, conviction of the selfish node is valid only if m different neighbors accuse, where m – 1 is an upper bound on the number of malicious nodes in the neighborhood. The success of this relies on proper selection of m.

3.1.6 *PLRSA-Promiscuous Listening Routing Security Algorithm-Li proposed this in 2006[21]:*

It Enable the promiscuous mode of every mobile host to intercept all the packets passing through the mobile host regardless of the destination address of the packet. Once when a node performs malicious behaviors, such as maliciously dropping of data packets or fabricating the spurious packets, the other nearby nodes may detect the spiteful behaviors. If the value of trust level is lower than a threshold defined by PLRSA then the node is considered as a malicious and further the malicious nodes are not considered for routing.

3.1.7 *E-Hermes-Zouridaki proposed this in 2009[1]:*

Each node determines the trustworthiness of the other nodes with respect to reliable packet forwarding by combining first-hand trust information obtained independently of other nodes and second-hand trust information obtained via recommendations from other nodes.

3.1.8 *LMRSA- Local Monitoring based Reputation System with Alert-Gopalakrishnan Proposed this in 2010[3]:*

This scheme derives the trustworthiness based on the direct observation experienced by a node from its next hop neighbors and also it does not exchange the trust values with the rest of the nodes in the network. This scheme generates an explicit alert and sends it to source node of the monitored transmission, whenever it declares its next hop node as a misbehaving node. This enables the packet originating node to select an alternate route for its current transmission, which in turn increases the overall network throughput.

3.1.9 *CARS (Collaborative Alert in a Reputation System): Gopalakrishnan proposed this in 2011[2]:*

Based on neighborhood monitoring approach to detect and isolate the colluding packet droppers with explicit alert mechanism.

3.1.10 *Neighborhood Monitoring Based Collaborative Alert Mechanism [1] proposed by the same author in 2011 which acts as a base paper for our research**:***

It differs from others by means of introducing a timeout approach for detecting the active neighbors before monitoring the transmissions which involves it. This approach does the timely generation of an explicit route error packet to inform about the misbehaving link, reintroduction of misbehaving nodes and dissemination of misbehaving node information through route request packet in a unique manner.

3.2 Summary of existing algorithm:

3.2.1 Next-hop monitoring:-

The algorithms 1,2,3,4,7,8 are based on next hop monitoring, in which the nodes except the destination and its previous hop in the source route of the packet has to monitor the behavior of its next hop in order to identify the node misbehavior.

3.2.1 Neighborhood Monitoring:-

The algorithms 5, 6,9,10 are based on neighborhood monitoring adds flexibility in monitoring by allowing a node to monitor the neighboring transmissions even if those transmissions does not involve it.

3.3 Drawbacks of existing algorithms:-

Following drawbacks exists more in 1st category algorithms and less in 2nd category algorithms-

3.3.1 Ambiguous Collision:

One node cannot overhear the transmission of the next node due to the concurrent transmission in its neighborhood and hence thinks it to be selfish.

3.3.2 Receiver Collision-

The transmission of the next node is overheard by the node but due to collision it did not receive then malicious node will not resend it.

3.3.3 False Misbehavior:-

The claim by a node is that a node has behaved selfishly although it is not the case.

3.3.4 Limited Transmission Power:-

A node makes its power so low that it can be overheard by the neighbor but cannot reach the receiver.

3.3.5 Multiple Colliding Nodes-

Two nodes generally collide to mischief.

Both category 1 and category 2 algorithms are having their advantages and disadvantages. We can exploit the benefits of both the kinds by developing a kind of mixed approach. We call it as **Random Selection of Neighbors for monitoring with Alert Mechanism against node misbehavior [RSNAM] in MANET. The aim of this algorithm is to achieve a drawbacks**

resistant reputation system in power effective manner. It differs from algorithm in our base paper in terms of number of neighbors of a node engaged in overhearing a node's transmission.

3.4 RSNAM: Detailed of our proposed algorithm:-

As in our base paper each node will have a NCL [Neighborhood Connectivity List] which will be having data structure as follows-

MODE	IP ADD	MAC ADD	TIME STAMP	REPUTATION	FLAG	**CTR**

The NCL should have the list of all the neighbors of a node. When a node 1st time overhears another node it adds it to the NCL with timestamp value. *It will also contain an entry for it-self with flag, mode, & ctr initialized to 0.*We add 3 fields more in this list to achieve our objective.

It also consists of three main components namely a Monitor, Reputation System and a Path Manager as an add-on to the existing Dynamic Source Routing (DSR) protocol functionality as in NMCAM.

Whenever a node overhears a packet from the neighboring node for the first time then the neighboring node information is stored in the Neighbor Connectivity List (NCL) along with the timestamp at which the packet is overheard. Its reputation value is initialized to 0.

The timestamp and the trust value are updated for the subsequent packet overhearing from the neighboring node.

If the node is overhearing then its flag is set. The mode is set 1 if it is 0 else vice versa after each overhearing by a node. After a time slice the flag is reset.

For subsequent overhearing by a node the ctr is incremented by 2 if mode is 0 or by 1 if mode is 1.After a regular time slice the ctr is decremented by 1 if the mode is 1 or incremented by1 if mode is 0.This is in order to achieve randomness in ctr value.

For subsequent overhearing by a node the ctr is incremented by 2 if mode is 0 or by 1 if mode is 1.After a regular time slice the ctr is decremented by 1 if the mode is 1 or incremented by1 if mode is 0.This is in order to achieve randomness in ctr value.

A node will be allowed to overhear if it is the sender or its counter value is even.

If the Trust Manager receives a positive event from the Monitor then the trust value of the corresponding node is incremented by 1.

In the case of a negative event reported from the monitor then the trust manager decrements the trust value by 2 for packet dropping misbehavior and by 4 for packet modification and identity spoofing misbehavior.

Once a trust value of a node reaches a *Negative Threshold limit then the faulty flag is set for it* and any packet to and from the misbehaving node will be rejected.

The faulty list [contains nodes with faulty flag on] is disseminated using a *RREQ packet so that the malicious node information is widely spread over the* network as well as it does not incur extra control overhead for disseminating the faulty list.

When a node receives a *RREQ packet, it extracts the faulty list from it and sets the suspicious flag on for the nodes in the faulty list in its NCL for the 1st time or sets the faulty flag on for the suspected nodes.*

If the received node is not a destination or an intermediate node that has a route to the destination then it will merge its own faulty list into the faulty list in the *RREQ packet and then rebroadcast it.*

A variable length list is added into *RREQ packet in order to accommodate the faulty list.*

A node rejects a route discovery and maintenance packet if its faulty table contains a node present in the source route of the received packet

After a node is added into the faulty list an explicit route error packet will be generated, which serves as an alert and sends it to the source of the monitored packet to inform about the misbehaving link. The source accordingly decides the routing path.

It also gives a chance to the selfish node after a certain timestamp but its reputation is made half.

3.5 Correctness and Completeness of algorithm:-

3.5.1 Packet Dropping:-

If the node can’t overhear the transmission then it regards it as packet dropping and accordingly step is taken. It can perform at least as good as NMCAM but with less power.

3.5.2 Identity spoofing:-

Identity spoofing is checked by comparing node info in received packet with the NCL entry.

3.5.3 Packet modification:-

By checksum calculation and comparing it with stored one.

3.5.4 Low power transmission:-

This is possible if the distance between the sender and receiver is more than that between observed node and overhearing node which is very rare as many nodes are overhearing simultaneously.

3.5.5 Colluding Packets:-

Due to the randomness of the selection of overhearing node the nodes can’t cooperate to mischief together.

3.5.6 *False Misbehavior:-*

The honest nodes will not suffer in our system because the node reported 1st time is taken as suspicious unlike the NMCAM.

Also we give 2nd chance to the selfish node after a certain time stamp.

CHAPTER 5

SIMULATION STUDY

AND

RESULTS

5.1 Simulation Study

The proposed algorithm RSNAM was implemented in OMNET++ as an addition on to the DSR routing algorithm. OMNeT++ is a modular discrete event network simulation framework. This simulator is based on object oriented approach. We utilize the Random Way Point model for mobility of the nodes as it well depicts a real world situation. This mobility model is based on entity mobility model where the nodes move independent of each other. We have taken following parameters for implementation.

Parameter	Value
Simulation Area	1600m*600m
Simulation Time	3000s
Number of nodes	100
Node Mobility	Random Way Point
Pay load size	512B
Positive Threshold	40
Negative Threshold	-40
Initial Trust Value of a Node	0
Carrier Frequency	2.4 GHz
Mobility Speed	10 mps
Transmitter Power	2.0Mw
Snirthreshold	4dB
Bitrate	54Mbps
Thermal noise	-110 dBm
Sensitivity	-90 dBm
Send buffer timeout	300s

5.1.1 Node Misbehaviour-

Many kinds of node misbehaviours were implemented. Some nodes participated in route discovery and route maintenance but did not do so in forwarding packet that too selectively. Some other totally did not participate in the process. Some others were designed to do packet modification. So that the implementation can be done properly

5.1.2 Selection of Neighbours-

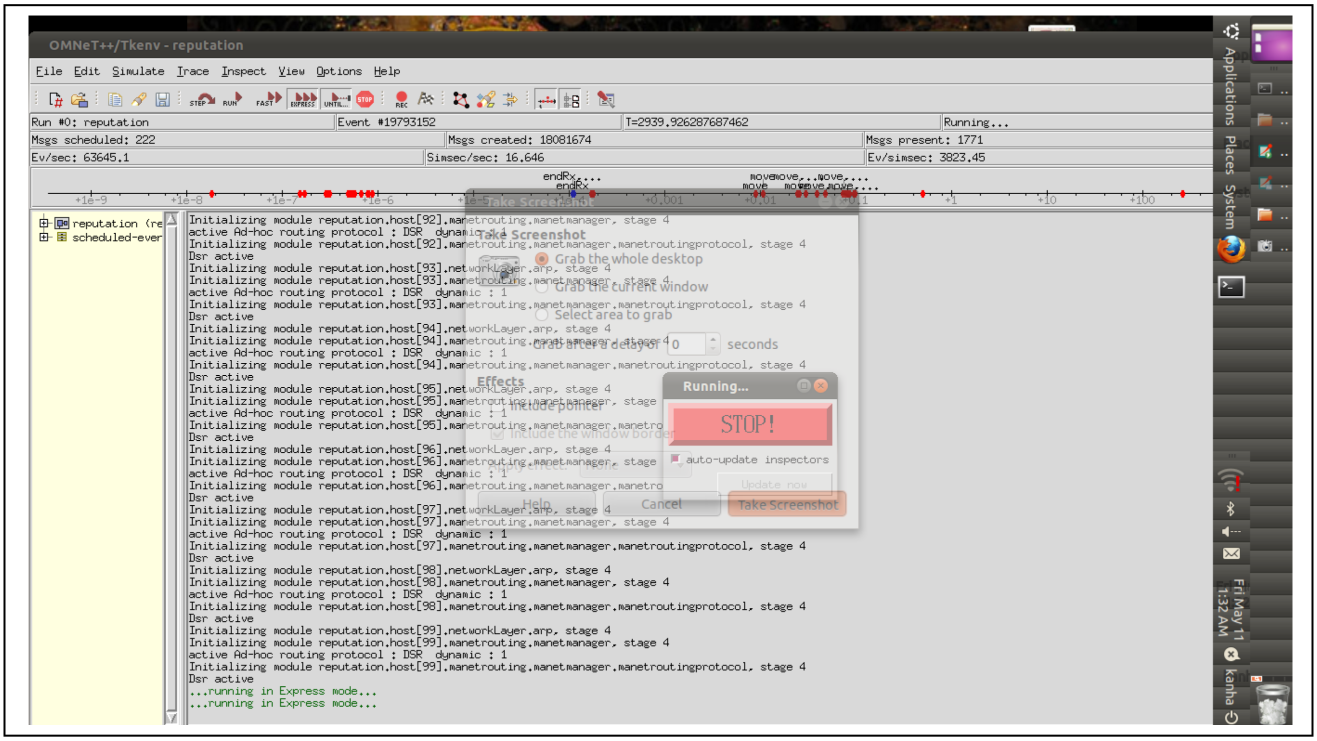
The selection of neighbours in detecting node misbehaviour was done in a random manner. The nodes were having a random number generator inside them so that every time they need to see its value before overhearing the channel. If the random number is evaluated as 0 then they were allowed to turn on their promiscuous mode to overhear the channel else they had to remain idle. This resulted in a lot of power saving of the nodes without affecting the fault detection.

5.1.3 *Dynamic Source Routing [DSR]-*

The Dynamic Source Routing protocol (DSR) is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. The protocol is composed of the two main mechanisms of "Route Discovery" and "Route Maintenance", which work together to allow nodes to discover and maintain routes to arbitrary destinations in the ad hoc network. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that needed to react to changes in the routes currently in use. The protocol allows multiple routes to any destination and allows each sender to select and control the routes used in routing its packets, for example for use in load balancing or for increased robustness.

Other advantages of the DSR protocol include easily guaranteed loop-free routing, operation in networks containing unidirectional links, use of only "soft state" in routing, and very rapid recovery when routes in the network change. The DSR protocol is designed mainly for mobile ad hoc networks of up to about two hundred nodes, and is designed to work well with even very high rates of mobility.

5.1.4 *Some Snapshots of Implementation*

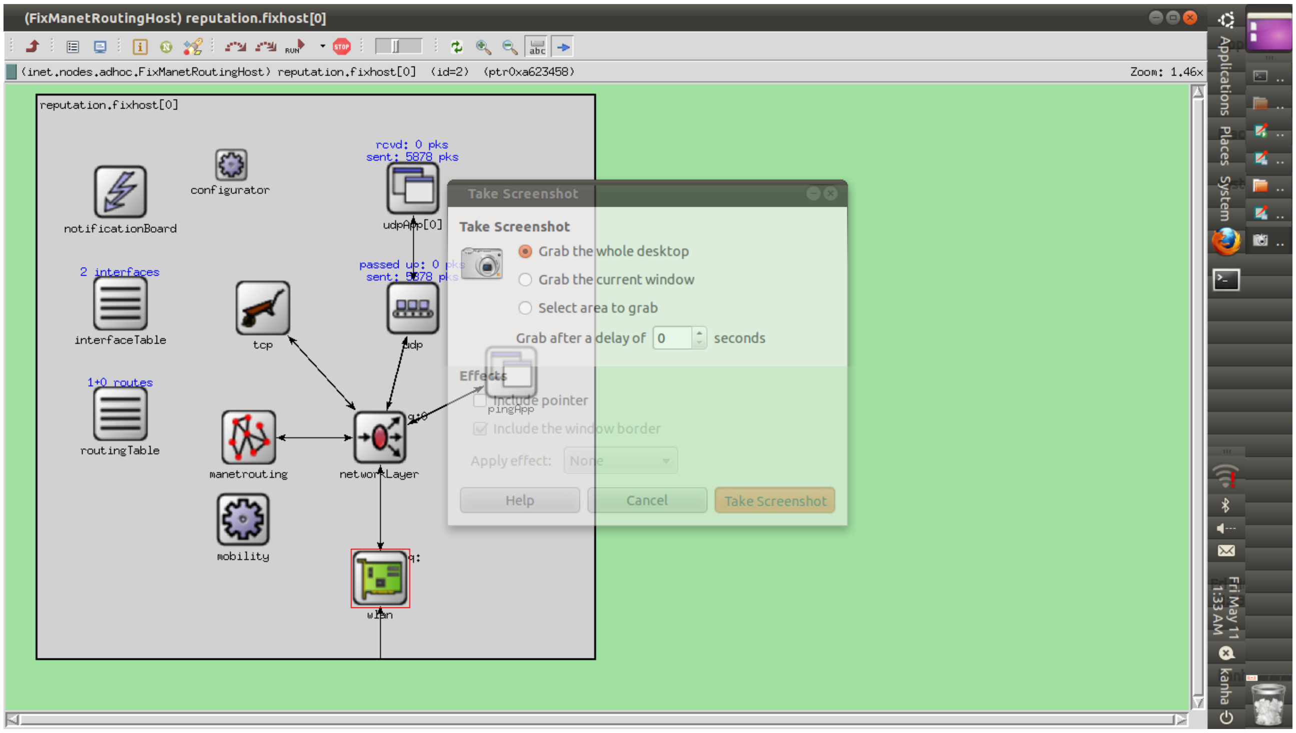


```
Run #0: reputation          Event #19793152          T=2939.926287687462          Running...
Msgs scheduled: 222                 Msgs created: 18081674                 Msgs present: 1771
Ev/sec: 63645.1                 Simsec/sec: 16.646                 Ev/simsec: 3823.45
```

```
Initializing module reputation.host[92].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[92].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
Initializing module reputation.host[93].networkLayer.arp, stage 4
Initializing module reputation.host[93].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[93].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
Initializing module reputation.host[94].networkLayer.arp, stage 4
Initializing module reputation.host[94].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[94].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
Initializing module reputation.host[95].networkLayer.arp, stage 4
Initializing module reputation.host[95].manetrouting.manetmanager, stage
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[95].manetrouting.manetmanager.manetro
Dsr active
Initializing module reputation.host[96].networkLayer.arp, stage 4
Initializing module reputation.host[96].manetrouting.manetmanager, stage
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[96].manetrouting.manetmanager.manetro
Dsr active
Initializing module reputation.host[97].networkLayer.arp, stage 4
Initializing module reputation.host[97].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[97].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
Initializing module reputation.host[98].networkLayer.arp, stage 4
Initializing module reputation.host[98].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[98].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
Initializing module reputation.host[99].networkLayer.arp, stage 4
Initializing module reputation.host[99].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[99].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
...running in Express mode...
...running in Express mode...
```

Fig 4.1 Omnet++/Tkenv running in Express mode

Fig 4.2 Layered Structure of a Node



(inet.nodes.adhoc.FixManetRoutingHost) reputation.fixhost[0] (id=2) (ptr0xa623458)

Take Screenshot

● Grab the whole desktop

○ Grab the current window

○ Select area to grab

Grab after a delay of 0 seconds

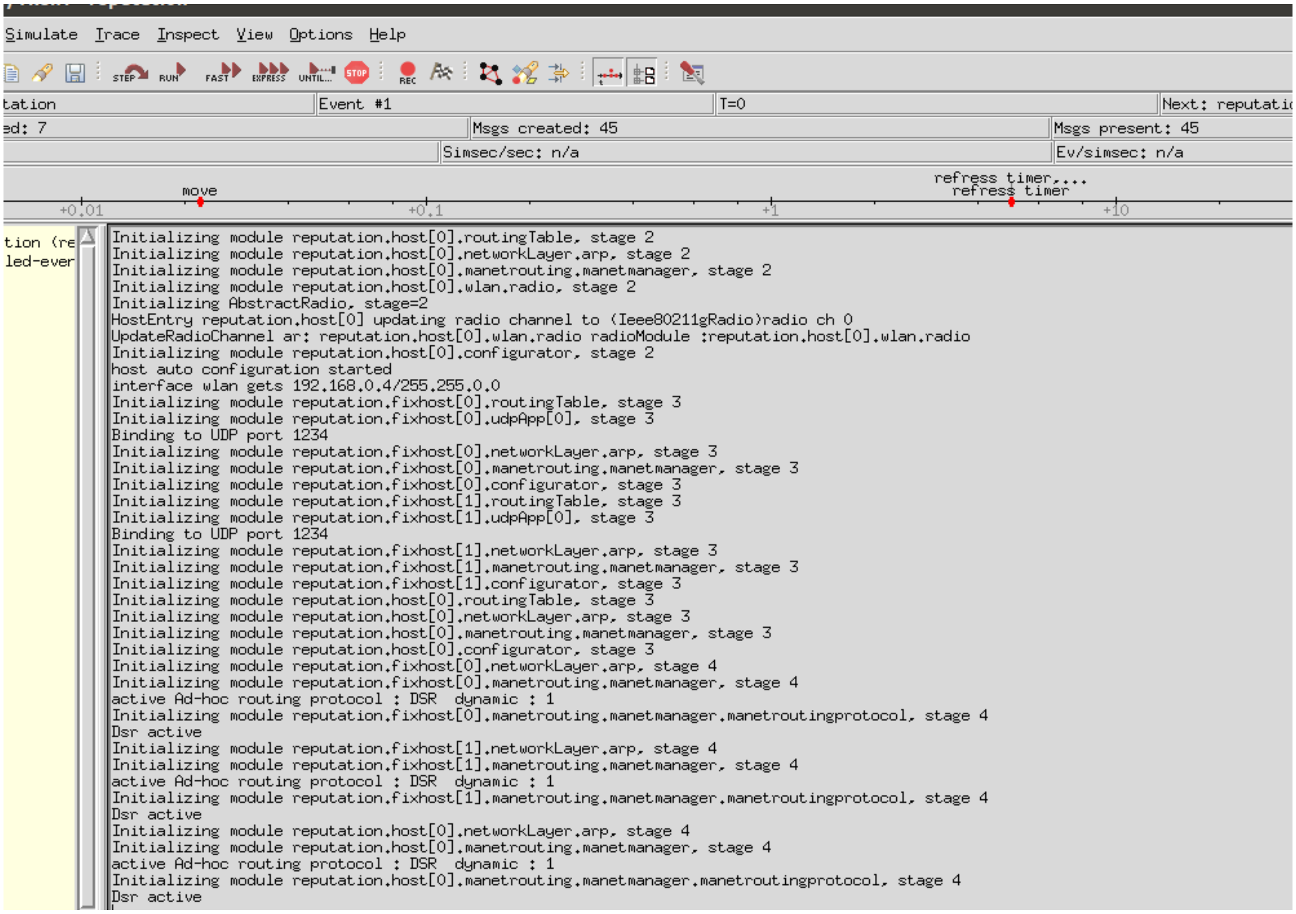
Effects

□ Include pointer

☑ Include the window border

Apply effect: None

Fig 4.3 Pre-run Initialization

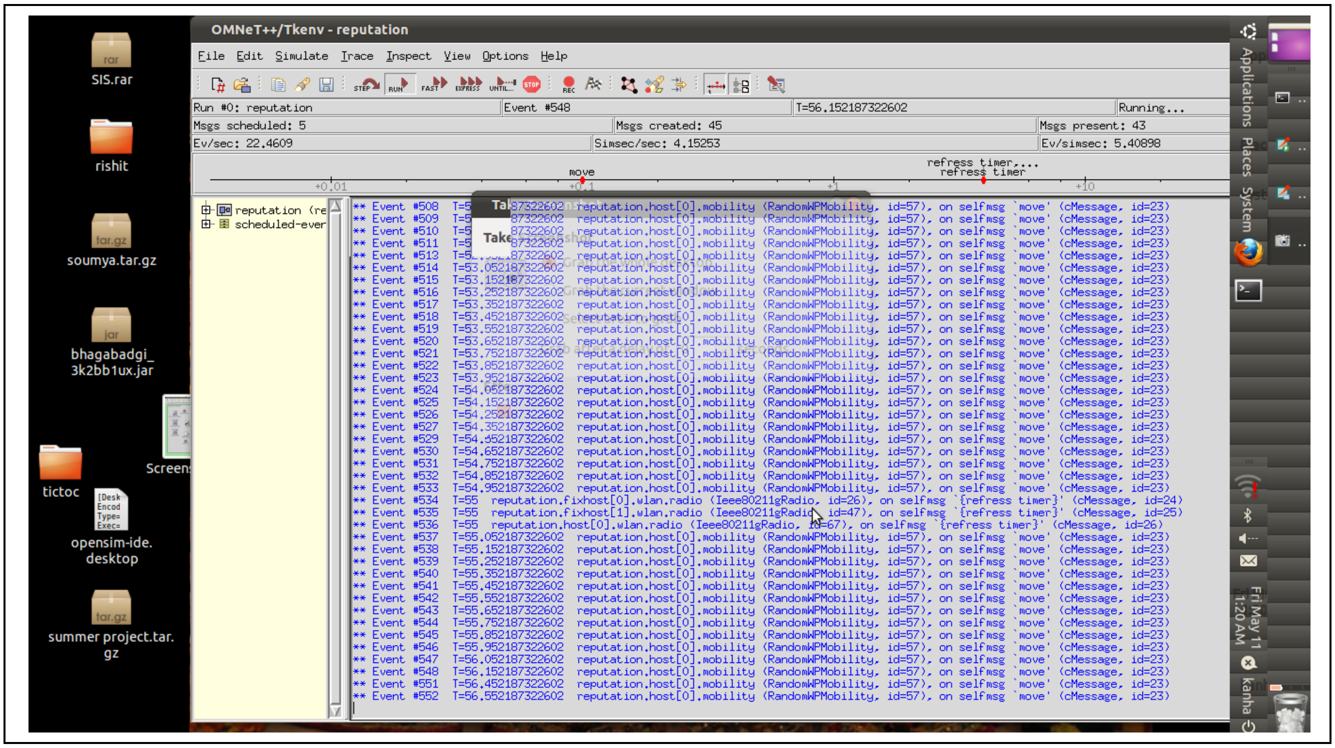


tation | Event #1 | T=0 | Next: reputati

ed: 7 | Msgs created: 45 | Msgs present: 45

Simsec/sec: n/a | Ev/simsec: n/a

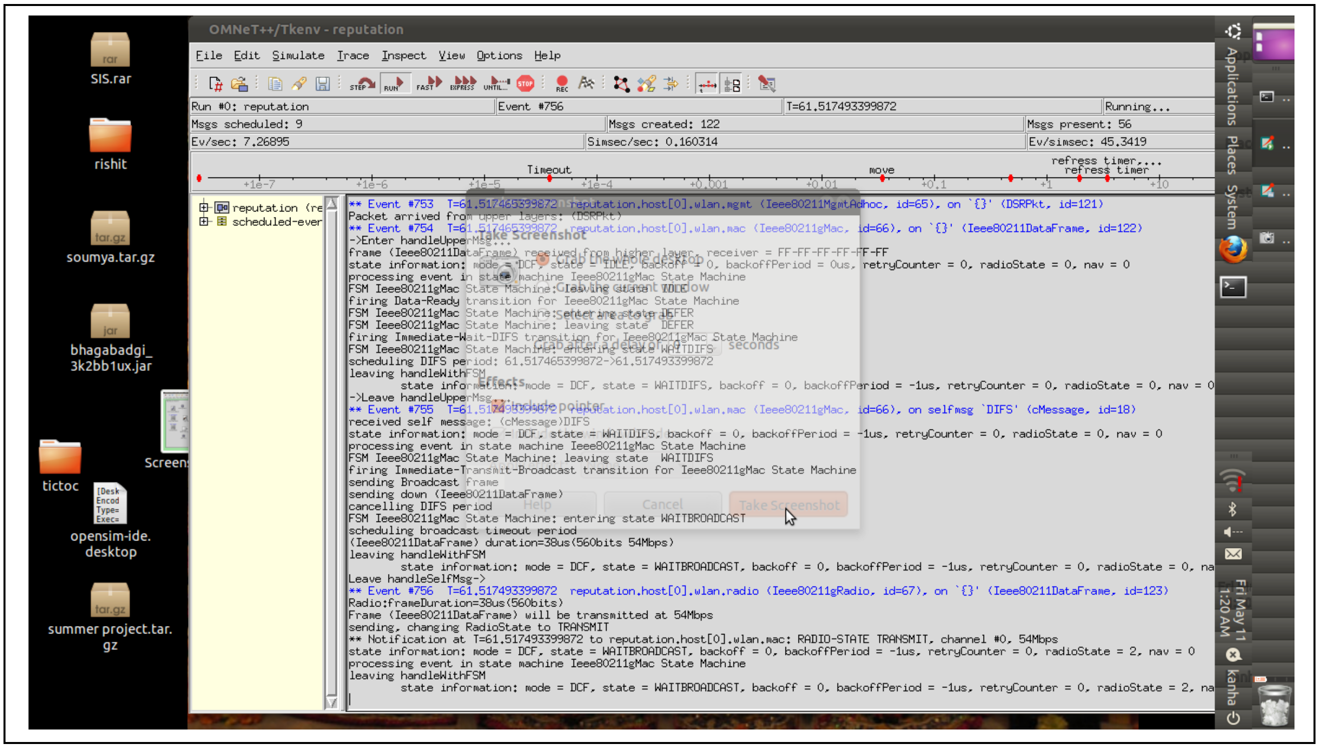
```
Initializing module reputation.host[0].routingTable, stage 2
Initializing module reputation.host[0].networkLayer.arp, stage 2
Initializing module reputation.host[0].manetrouting.manetmanager, stage 2
Initializing module reputation.host[0].wlan.radio, stage 2
Initializing AbstractRadio, stage=2
HostEntry reputation.host[0] updating radio channel to (Ieee80211gRadio)radio ch 0
UpdateRadioChannel ar: reputation.host[0].wlan.radio radioModule :reputation.host[0].wlan.radio
Initializing module reputation.host[0].configurator, stage 2
host auto configuration started
interface wlan gets 192.168.0.4/255.255.0.0
Initializing module reputation.fixhost[0].routingTable, stage 3
Initializing module reputation.fixhost[0].udpApp[0], stage 3
Binding to UDP port 1234
Initializing module reputation.fixhost[0].networkLayer.arp, stage 3
Initializing module reputation.fixhost[0].manetrouting.manetmanager, stage 3
Initializing module reputation.fixhost[0].configurator, stage 3
Initializing module reputation.fixhost[1].routingTable, stage 3
Initializing module reputation.fixhost[1].udpApp[0], stage 3
Binding to UDP port 1234
Initializing module reputation.fixhost[1].networkLayer.arp, stage 3
Initializing module reputation.fixhost[1].manetrouting.manetmanager, stage 3
Initializing module reputation.fixhost[1].configurator, stage 3
Initializing module reputation.host[0].routingTable, stage 3
Initializing module reputation.host[0].networkLayer.arp, stage 3
Initializing module reputation.host[0].manetrouting.manetmanager, stage 3
Initializing module reputation.host[0].configurator, stage 3
Initializing module reputation.fixhost[0].networkLayer.arp, stage 4
Initializing module reputation.fixhost[0].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.fixhost[0].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
Initializing module reputation.fixhost[1].networkLayer.arp, stage 4
Initializing module reputation.fixhost[1].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.fixhost[1].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
Initializing module reputation.host[0].networkLayer.arp, stage 4
Initializing module reputation.host[0].manetrouting.manetmanager, stage 4
active Ad-hoc routing protocol : DSR  dynamic : 1
Initializing module reputation.host[0].manetrouting.manetmanager.manetroutingprotocol, stage 4
Dsr active
```



```
Run #0: reputation          Event #548          T=56.152187322602          Running...
Msgs scheduled: 5           Msgs created: 45                    Msgs present: 43
Ev/sec: 22.4609             Simsec/sec: 4.15253                 Ev/simsec: 5.40898

** Event #508  T=5[illegible]87322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #509  T=5[illegible]87322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #510  T=5[illegible]87322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #511  T=5[illegible]87322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #513  T=5[illegible]87322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #514  T=53.052187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #515  T=53.152187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #516  T=53.252187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #517  T=53.352187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #518  T=53.452187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #519  T=53.552187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #520  T=53.652187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #521  T=53.752187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #522  T=53.852187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #523  T=53.952187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #524  T=54.052187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #525  T=54.152187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #526  T=54.252187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #527  T=54.352187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #529  T=54.552187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #530  T=54.652187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #531  T=54.752187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #532  T=54.852187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #533  T=54.952187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #534  T=55  reputation.fixhost[0].wlan.radio (Ieee80211gRadio, id=26), on selfmsg `{refress timer}' (cMessage, id=24)
** Event #535  T=55  reputation.fixhost[1].wlan.radio (Ieee80211gRadio, id=47), on selfmsg `{refress timer}' (cMessage, id=25)
** Event #536  T=55  reputation.host[0].wlan.radio (Ieee80211gRadio, id=67), on selfmsg `{refress timer}' (cMessage, id=26)
** Event #537  T=55.052187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #538  T=55.152187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #539  T=55.252187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #540  T=55.352187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #541  T=55.452187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #542  T=55.552187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #543  T=55.652187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #544  T=55.752187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #545  T=55.852187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #546  T=55.952187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #547  T=56.052187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #548  T=56.152187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #551  T=56.452187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #552  T=56.552187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
```

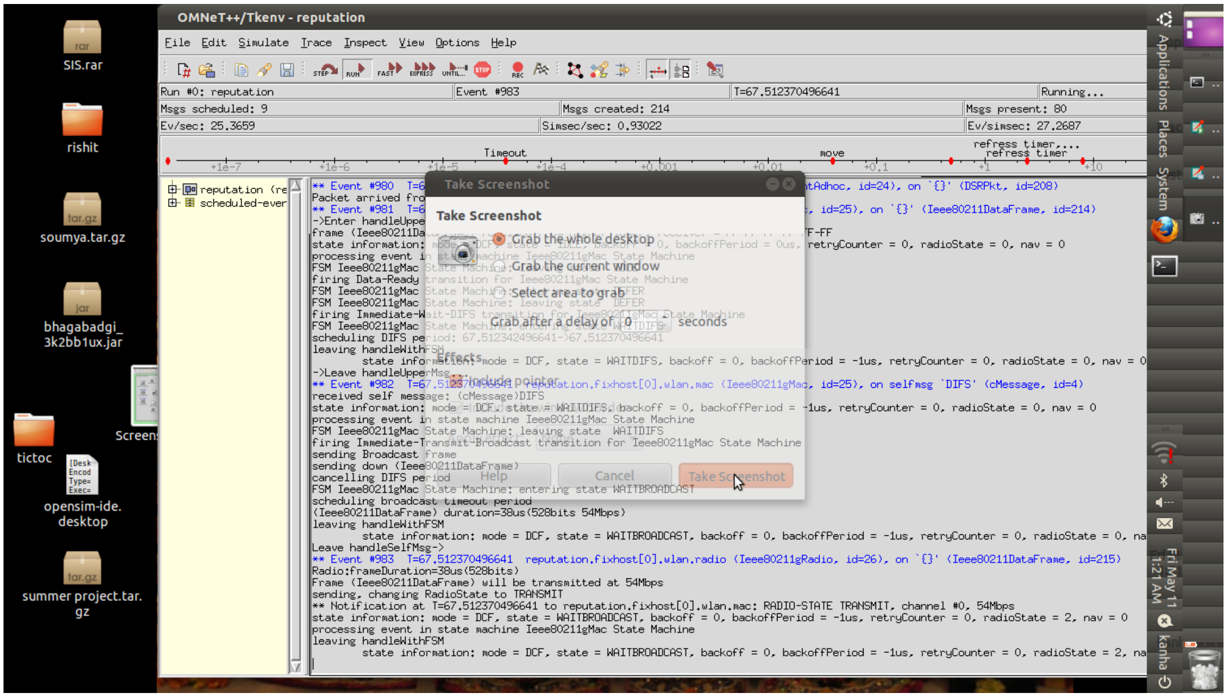
Fig 4.4 Running in Normal Mode (Mobility)



```
Run #0: reputation          Event #756          T=61.517493399872          Running...
Msgs scheduled: 9           Msgs created: 122                   Msgs present: 56
Ev/sec: 7.26895             Simsec/sec: 0.160314                Ev/simsec: 45.3419

** Event #753  T=61.517465399872  reputation.host[0].wlan.mgmt (Ieee80211MgmtAdhoc, id=65), on `{}' (DSRPkt, id=121)
Packet arrived from upper layers: (DSRPkt)
** Event #754  T=61.517465399872  reputation.host[0].wlan.mac (Ieee80211gMac, id=66), on `{}' (Ieee80211DataFrame, id=122)
->Enter handleUpperMsg...
frame (Ieee80211DataFrame) received from higher layer, receiver = FF-FF-FF-FF-FF-FF
state information: mode = DCF, state = IDLE, backoff = 0, backoffPeriod = 0us, retryCounter = 0, radioState = 0, nav = 0
processing event in state machine Ieee80211gMac State Machine
FSM Ieee80211gMac State Machine: leaving state  IDLE
firing Data-Ready transition for Ieee80211gMac State Machine
FSM Ieee80211gMac State Machine: entering state DEFER
FSM Ieee80211gMac State Machine: leaving state  DEFER
firing Immediate-Wait-DIFS transition for Ieee80211gMac State Machine
FSM Ieee80211gMac State Machine: entering state WAITDIFS
scheduling DIFS period: 61.517465399872->61.517493399872
leaving handleWithFSM
        state information: mode = DCF, state = WAITDIFS, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 0, nav = 0
->Leave handleUpperMsg...
** Event #755  T=61.517493399872  reputation.host[0].wlan.mac (Ieee80211gMac, id=66), on selfmsg `DIFS' (cMessage, id=18)
received self message: (cMessage)DIFS
state information: mode = DCF, state = WAITDIFS, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 0, nav = 0
processing event in state machine Ieee80211gMac State Machine
FSM Ieee80211gMac State Machine: leaving state  WAITDIFS
firing Immediate-Transmit-Broadcast transition for Ieee80211gMac State Machine
sending Broadcast frame
sending down (Ieee80211DataFrame)
cancelling DIFS period
FSM Ieee80211gMac State Machine: entering state WAITBROADCAST
scheduling broadcast timeout period
(Ieee80211DataFrame) duration=38us(560bits 54Mbps)
leaving handleWithFSM
        state information: mode = DCF, state = WAITBROADCAST, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 0, na
Leave handleSelfMsg->
** Event #756  T=61.517493399872  reputation.host[0].wlan.radio (Ieee80211gRadio, id=67), on `{}' (Ieee80211DataFrame, id=123)
Radio:frameDuration=38us(560bits)
Frame (Ieee80211DataFrame) will be transmitted at 54Mbps
sending, changing RadioState to TRANSMIT
** Notification at T=61.517493399872 to reputation.host[0].wlan.mac: RADIO-STATE TRANSMIT, channel #0, 54Mbps
state information: mode = DCF, state = WAITBROADCAST, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 2, nav = 0
processing event in state machine Ieee80211gMac State Machine
leaving handleWithFSM
        state information: mode = DCF, state = WAITBROADCAST, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 2, na
```

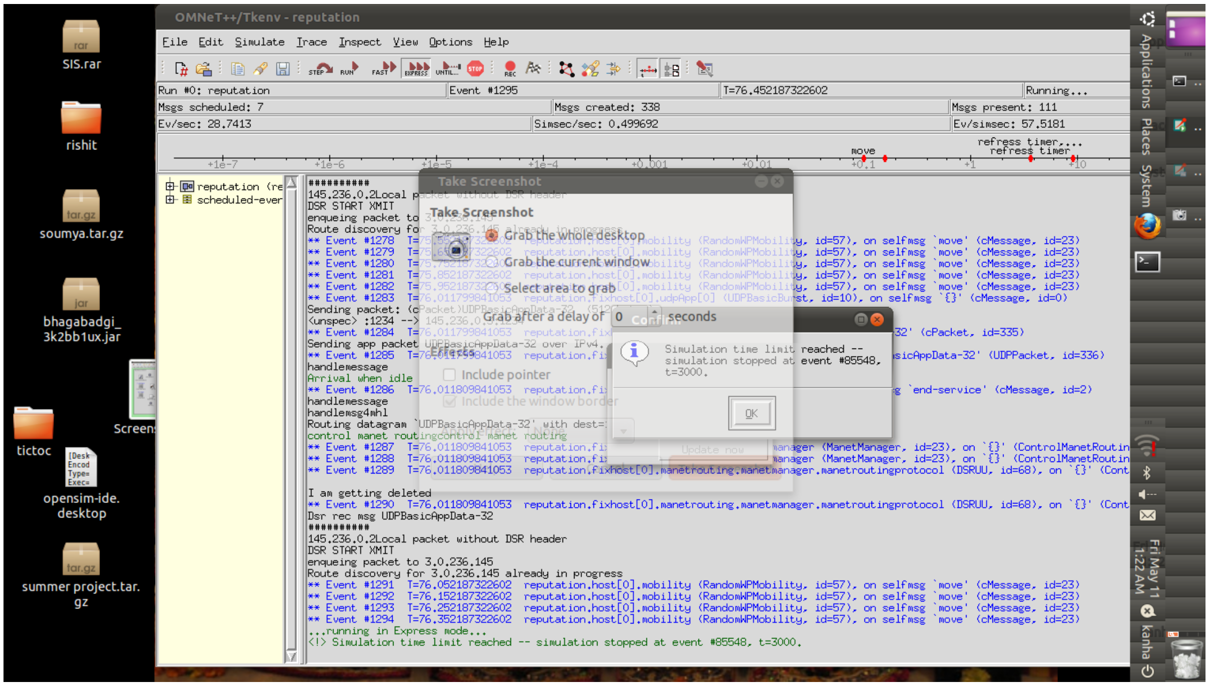
Fig 4.5 Running in Normal Mode (DSR)



```
Run #0: reputation    Event #983    T=67.512370496641    Running...
Msgs scheduled: 9    Msgs created: 214    Msgs present: 80
Ev/sec: 25.3659    Simsec/sec: 0.93022    Ev/simsec: 27.2687

** Event #980  T=6                                                tAdhoc, id=24), on `{}' (DSRPkt, id=208)
Packet arrived fro
** Event #981  T=6                                                , id=25), on `{}' (Ieee80211DataFrame, id=214)
->Enter handleUppe
frame (Ieee80211Da                                                F-FF
state information: mode = DCF, state = IDLE, backoff = 0, backoffPeriod = 0us, retryCounter = 0, radioState = 0, nav = 0
processing event in state machine Ieee80211gMac State Machine
FSM Ieee80211gMac State Machine: leaving state
firing Data-Ready transition for Ieee80211gMac State Machine
FSM Ieee80211gMac State Machine: entering state DEFER
FSM Ieee80211gMac State Machine: leaving state  DEFER
firing Immediate-Wait-DIFS transition for Ieee80211gMac State Machine
FSM Ieee80211gMac State Machine: entering state WAITDIFS
scheduling DIFS period: 67.512342496641->67.512370496641
leaving handleWithFSM
        state information: mode = DCF, state = WAITDIFS, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 0, nav = 0
->Leave handleUpperMsg
** Event #982  T=67.512370496641  reputation.fixhost[0].wlan.mac (Ieee80211gMac, id=25), on selfmsg `DIFS' (cMessage, id=4)
received self message: (cMessage)DIFS
state information: mode = DCF, state = WAITDIFS, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 0, nav = 0
processing event in state machine Ieee80211gMac State Machine
FSM Ieee80211gMac State Machine: leaving state  WAITDIFS
firing Immediate-Transmit-Broadcast transition for Ieee80211gMac State Machine
sending Broadcast frame
sending down (Ieee80211DataFrame)
cancelling DIFS period
FSM Ieee80211gMac State Machine: entering state WAITBROADCAST
scheduling broadcast timeout period
(Ieee80211DataFrame) duration=38us(528bits 54Mbps)
leaving handleWithFSM
        state information: mode = DCF, state = WAITBROADCAST, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 0, na
Leave handleSelfMsg->
** Event #983  T=67.512370496641  reputation.fixhost[0].wlan.radio (Ieee80211gRadio, id=26), on `{}' (Ieee80211DataFrame, id=215)
Radio:frameDuration=38us(528bits)
Frame (Ieee80211DataFrame) will be transmitted at 54Mbps
sending, changing RadioState to TRANSMIT
** Notification at T=67.512370496641 to reputation.fixhost[0].wlan.mac: RADIO-STATE TRANSMIT, channel #0, 54Mbps
state information: mode = DCF, state = WAITBROADCAST, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 2, nav = 0
processing event in state machine Ieee80211gMac State Machine
leaving handleWithFSM
        state information: mode = DCF, state = WAITBROADCAST, backoff = 0, backoffPeriod = -1us, retryCounter = 0, radioState = 2, na
```

Fi 4.6 Running in Normal Mode (Data link layer)



Simulation time limit reached -- simulation stopped at event #85548, t=3000.

```
Run #0: reputation    Event #1295    T=76.452187322602    Running...
Msgs scheduled: 7    Msgs created: 338    Msgs present: 111
Ev/sec: 28.7413    Simsec/sec: 0.499692    Ev/simsec: 57.5181

##########
145.236.0.2Local packet without DSR header
DSR START XMIT
enqueing packet to 3.0.236.145
Route discovery for 3.0.236.145 already in progress
** Event #1278  T=75.552187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #1279  T=75.652187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #1280  T=75.752187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #1281  T=75.852187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #1282  T=75.952187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #1283  T=76.011799841053  reputation.fixhost[0].udpApp[0] (UDPBasicBurst, id=10), on selfmsg `{}' (cMessage, id=0)
Sending packet: (cPacket)UDPBasicAppData-32  (512
<unspec> :1234 --> 145.236.0.3:1234
** Event #1284  T=76.011799841053  reputation.fix                         32' (cPacket, id=335)
Sending app packet UDPBasicAppData-32 over IPv4.
** Event #1285  T=76.011799841053  reputation.fi                          sicAppData-32' (UDPPacket, id=336)
handlemessage
Arrival when idle
** Event #1286  T=76.011809841053  reputation.fi                          g `end-service' (cMessage, id=2)
handlemessage
handlemsg4mhl
Routing datagram `UDPBasicAppData-32' with dest=
control manet routingcontrol manet routing
** Event #1287  T=76.011809841053  reputation.fi                          anager (ManetManager, id=23), on `{}' (ControlManetRoutin
** Event #1288  T=76.011809841053  reputation.fi                          anager (ManetManager, id=23), on `{}' (ControlManetRoutin
** Event #1289  T=76.011809841053  reputation.fixhost[0].manetrouting.manetmanager.manetroutingprotocol (DSRUU, id=68), on `{}' (Cont

I am getting deleted
** Event #1290  T=76.011809841053  reputation.fixhost[0].manetrouting.manetmanager.manetroutingprotocol (DSRUU, id=68), on `{}' (Cont
Dsr rec msg UDPBasicAppData-32
##########
145.236.0.2Local packet without DSR header
DSR START XMIT
enqueing packet to 3.0.236.145
Route discovery for 3.0.236.145 already in progress
** Event #1291  T=76.052187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #1292  T=76.152187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #1293  T=76.252187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
** Event #1294  T=76.352187322602  reputation.host[0].mobility (RandomWPMobility, id=57), on selfmsg `move' (cMessage, id=23)
...running in Express mode...
<!> Simulation time limit reached -- simulation stopped at event #85548, t=3000.
```

Fig 4.7 Running in Express Mode (Simulation time reached)

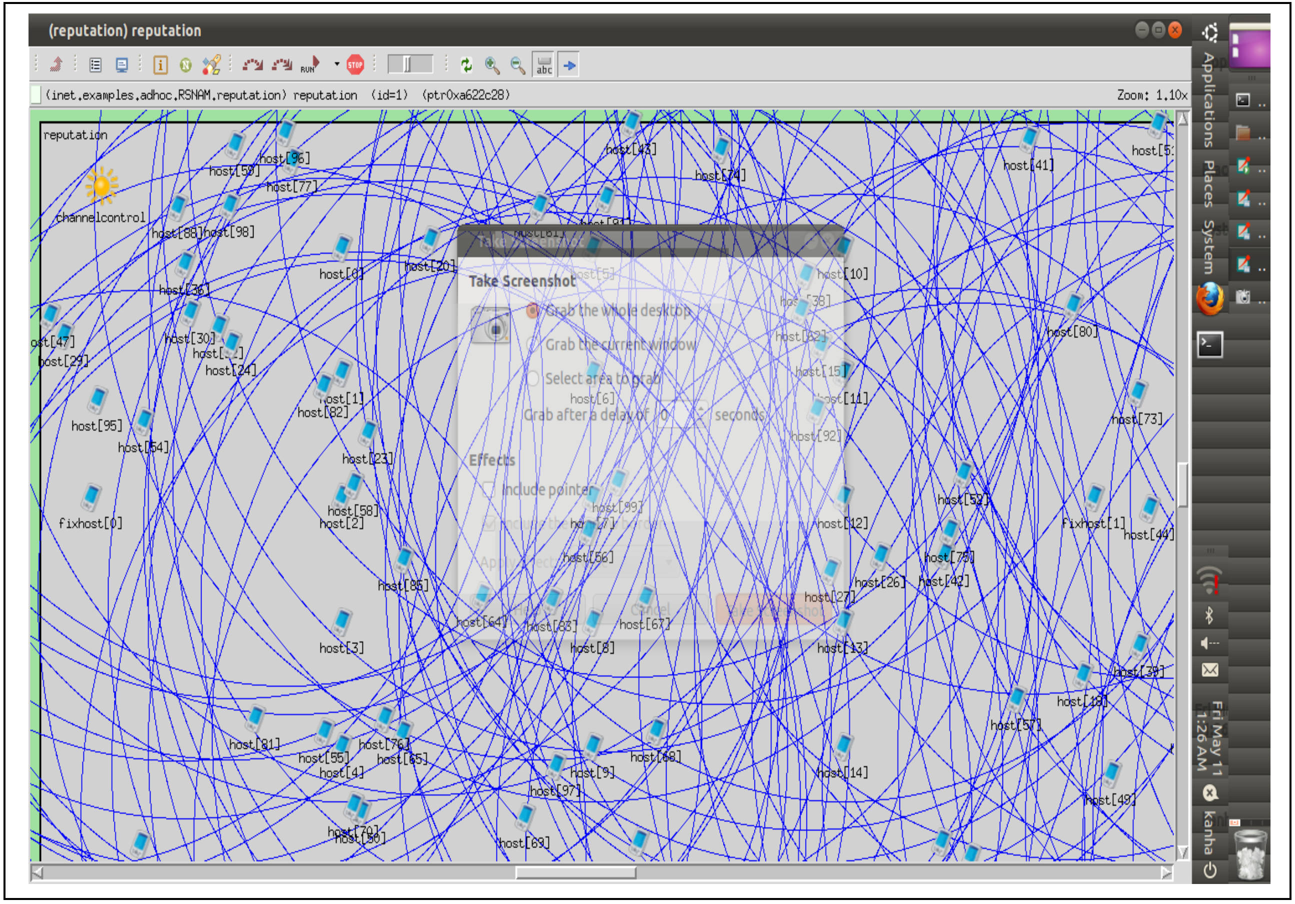


Fig 4.8 Graphical view of Nodes

5.2 Results

Ideally, power consumed for a MANET in non-promiscuous mode is 73 watt/hr/node when we consider the node to be a simple laptop. In promiscuous mode average power consumption is much higher. On an average let us say 105watt/hr/node. In case of previous algorithms based on neighborhood monitoring, all neighbors have to overhear the channel. So 100% nodes are to be in promiscuous mode consuming enormous power, but our proposed algorithm selects randomly some neighbors to overhear the channel. Hence nearly 50% (average case) of nodes are to be in promiscuous mode in this case. Rest of the nodes can be in non-promiscuous mode.

We give the results in terms of following graphs.

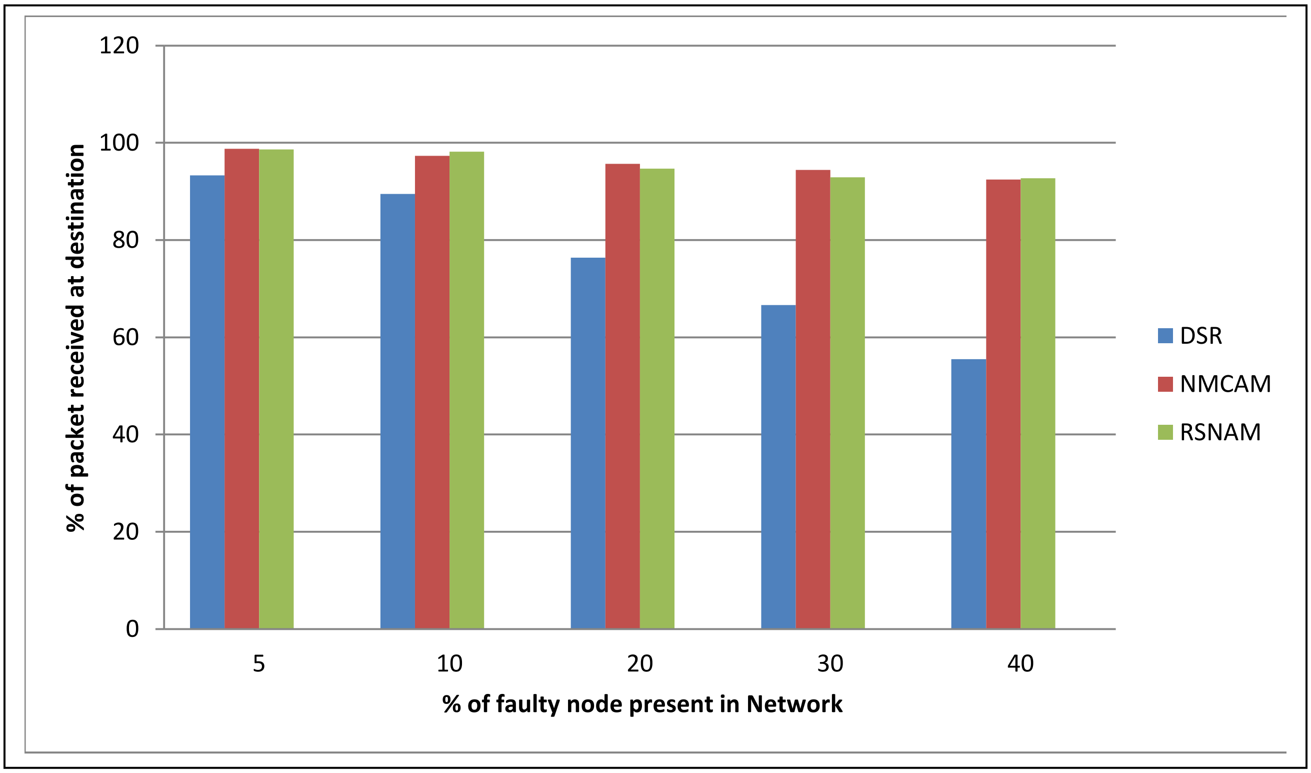


Fig 5.9 comparison of packet received % in DSR, NMCAM and RSNAM

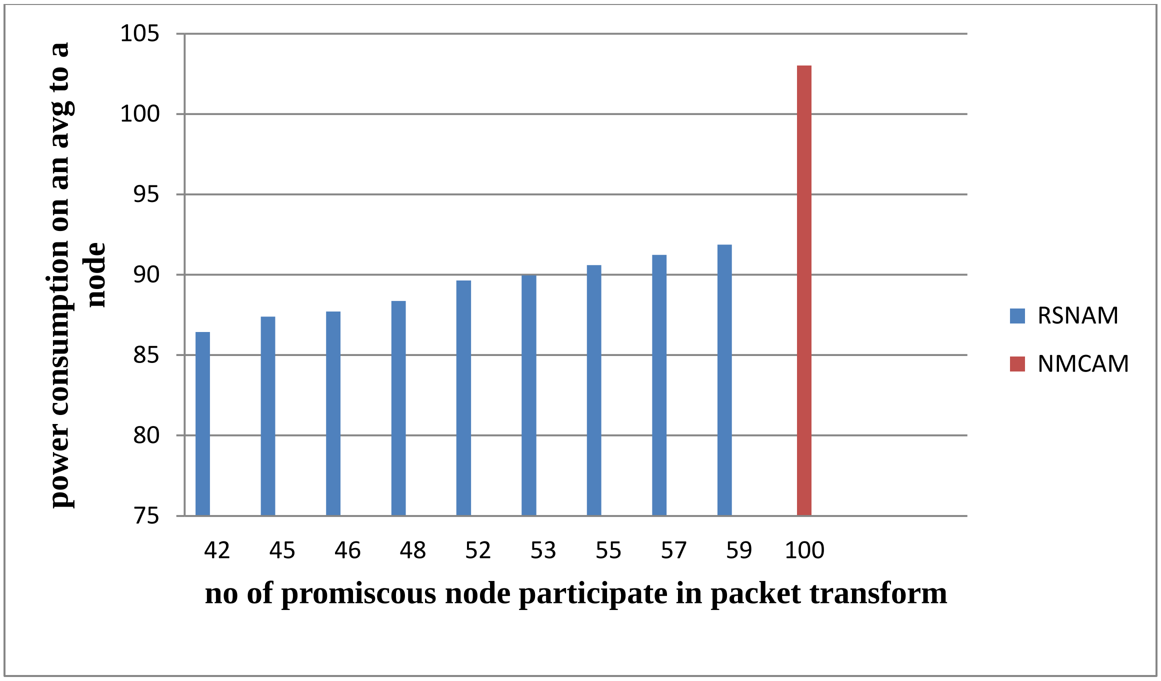


Fig 5.10 Average power consumption comparison

CHAPTER 6
CONCLUSION
AND
FUTURE WORK

CONCLUSION

Previously, all research works show that how effectively a packet can be sent form source to the destination. Our aim here focuses on doing the same thing in a power effective manner. That is the packet loss ratio is minimized but in less power utilization by nodes. Hence we select some of the nodes in the neighborhood but not all to overhear the channel. These nodes are selected randomly. This makes the power consumption very less as compared to previous works done.

FUTURE WORK

Power efficiency may be reduced more than what we have done in this research and in future selfishness of a node can also be checked which yet to be done. To remove ambiguous collision and false misbehavior will be our research.

BIBILIOGRAPHY

[1] K. Gopalakrishnan & Rhymend Uthariaraj, in V.. 2011, “Neighborhood Monitoring Based Collaborative Alert Mechanism to Thwart the Misbehaving Nodes in Mobile Ad-Hoc Network ”, European Journal of Scientific Research ISSN 1450-216X Vol.57 No.3 pp.411-425.

[2] K. Gopalakrishnan, Rhymend Uthariaraj , V..2011. “Collaborative Alert in a Reputation System to Alleviate Colluding Packet Droppers in Mobile Ad Hoc Networks”, *In: Meghanathan, N., Kaushik, B.K., Nagamalai, D. (Eds.) CCSIT 2011. Part I, CCIS, vol. 131, pp.*135–146. Springer, Heidelberg.

[3] K. Gopalakrishnan, V. Rhymend Uthariaraj V..2010. “Local Monitoring based Reputation System with Alert to Mitigate the Misbehaving Nodes in Mobile Ad Hoc Networks”, *In: Das, V. Vijaykumar, R. (Eds.) ICT 2010.Part II, CCIS, vol. 101, pp. 344–349. Springer,* Heidelberg

[4] Sonja Buchegger, Jean-Yves Le Boudec, V.. 2005*. “*Self-Policing Mobile Ad Hoc Networks by Reputation Systems*”, In:* IEEE Communications Magazine.

[5] Tiranuch Anantvalee, Jie Wu, V.. 2007. “Reputation-Based System for Encouraging the Cooperation of Nodes in Mobile Ad Hoc Networks”, *In*: IEEE Communications Magazine.

[6] Malamati Louta, Stylianos Kraounakis, Angelos Michalas, V.. 2010. “A Survey On Reputation-Based Cooperation Enforcement Schemes In Wireless Ad Hoc Networks”,

[7] “Mobile adhoc networks and application”[available online] http://en.wikipedia.org/wiki/Mobile Ad Hoc

[8] “Omnet and How to write codes in it” [available online] **http://www.omnetpp.org/doc/omnetpp/manual/usman.html**

[9] “How to install Omnet in Ubuntu Operating system” [available online] **http://omnetpp.org/doc/omnetpp/InstallGuide.pdf**

[10] Brian Neil Levine, Clay Shields, N Boris Margolin, V.. 2005. “A Survey of Solutions to the Sybil Attack. ”

[11] “Mobile ad hoc networks and features present on it” [available online] **http://www.bluetronix.net/mobile_ad_hoc_networks.htm**

[12] Buchegger, S., Boudec, JY Le., 2002. “Performance Analysis of the CONFIDANT Protocol (Cooperation of Nodes: Fairness In Dynamic Ad-hoc NeTworks)”, *In: IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing*, pp. 226-236. ACM, Lausanne.

[13] Marti, S., Giuli, TJ., Lai, K., Baker, M., 2000. “Mitigating Routing Misbehavior in Mobile Ad-Hoc Networks”, *In: 6th International Conference on Mobile Computing and Networking*, pp.255-265. ACM, Boston.

[14] Hu, J., Burmester, M., 2006. “LARS - A Locally Aware Reputation System for Mobile Ad Hoc Networks”, *In: 44th Annual Southeast Regional Conference*, pp. 119-123. ACM, Melbourne.

[15] Yanbin Liu. Yang Richard Yang, V..2003. “Reputation Propagation and Agreement in Mobile Ad-Hoc Networks”.

[16] Tiranuch Anantvalee and Jie Wu, V.. 2007. “Reputation-Based System for Encouraging the Cooperation of Nodes in Mobile Ad Hoc Networks” Department of Computer Science and Engineering Florida Atlantic University,

[17] Zouridaki, C., Mark, B., Hejmo, M., Thomas, R., V.. 2009. “E-Hermes: A robust cooperative trust establishment scheme for mobile ad hoc networks”, *Journal of ELSEVIER Ad Hoc Networks*, pp. 1156–1168.

[18] S. Marti *et al.*, “Mitigating Routing Misbehavior in Mobile Ad Hoc Networks,” *Proc. MOBICOM 2000*, pp. 255–65.

[19] Michiardi, P. Molva, R., 2002. “CORE: A COllaborative REputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks”, *In: 6th Joint Working Conference on Communications and Multimedia Security*, vol. 228, pp. 107-121. Kluwer, Portoroz.

[20] Bansal, S., Baker, M., 2003. “Observation-based Cooperation Enforcement in Ad-hoc Networks”, *Technical Report*, Stanford University.

[21] Li, J-S., Lee, C-T., 2006. “Improve Routing Trust with Promiscuous Listening Routing Security Algorithm in Mobile Ad Hoc Networks”, *Journal of ELSEVIER Computer Communications*, pp. 1121-1132.