

**Improving AODV using Acknowledgment based Punishment Algorithm
for MANETs in the Presence of Selfish Nodes**



Dagmawit Tadesse Hirpa

A Thesis Submitted to

The Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment for the Degree of Masters of Science in

Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February 2022
Adama, Ethiopia

**Improving AODV using Acknowledgment based Punishment Algorithm
for MANETs in the Presence of Selfish Nodes**

Dagmawit Tadesse Hirpa

Advisor: Ketema Adere (Ph.D)

A Thesis Submitted to

The Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment for the Degree of Masters of Science in

Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February 2022

Adama, Ethiopia

Declaration

I hereby declare that this Master Thesis entitled “**Improving AODV using Acknowledgment based Punishment Algorithm for MANETs in the Presence of Selfish Nodes**” is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Dagmawit Tadesse Hirpa

Name of the student

Signature

Date

Recommendation

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled “**Improving AODV using Acknowledgment based Punishment Algorithm for MANETs in the Presence of Selfish Nodes**” prepared under my guidance by **Dagmawit Tadesse Hirpa** submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science and Engineering.

Therefore, I recommend the submission of the revised version of the thesis to the department following the applicable procedures.

Dr. Ketema Adere

Major Advisor

Signature

Date

Approval Sheet

I, the advisors of the thesis entitled “**Improving AODV using Acknowledgment based Punishment Algorithm for MANETs in the Presence of Selfish Nodes**” and developed by **Dagmawit Tadesse Hirpa**, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. Ketema Adere

Major Advisor

Signature

Date

We, the undersigned, members of the Board of Examiners of the thesis by **Dagmawit Tadesse Hirpa** have read and evaluated the thesis entitled “**Improving AODV using Acknowledgment based Punishment Algorithm for MANETs in the Presence of Selfish Nodes**” and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

Chairperson

Signature

Date

Internal Examiner

Signature

Date

External Examiner

Signature

Date

Finally, approval and acceptance of the thesis are contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head

Signature

Date

School Dean

Signature

Date

Office of Postgraduate Studies, Dean

Signature

Date

ACKNOWLEDGEMENT

Most of all, I would like to give special thanks to the Almighty God for helping me to stand today. Next, I would like to give the deepest gratitude to the Virgin Mary, who always led me through her intercession.

I would like to extend my special thanks and respect to my advisor Dr. Ketema Adere for the support, positive feedback, his kindly approach, encouragement, and appreciation through the process of research thesis work until completion. This research cannot be accomplished without his support and mentor. He is a great, understanding person and one of the best mentors.

Finally, I would like to give my sincere thanks to my family especially to my Mother Tiruwork Mekuriya for the sacrifices she pays to make me stand here. May God give her long life and make her healthy. I also give my heartfelt thanks to my friends especially Network SIG mates for their support and advice which help me to finish my work.

Table of Contents

ACKNOWLEDGEMENT	i
LIST OF TABLES.....	v
LIST OF FIGURES	vi
ACRONYMS AND ABBREVIATIONS	vii
ABSTRACT	viii
CHAPTER ONE.....	1
1. INTRODUCTION	1
1.1. Background of the Study.....	1
1.2. Motivation of the Study.....	4
1.3. Statements of the Problem.....	4
1.4. Research Questions	5
1.5. Objectives of the Study	5
1.5.1. General Objective	5
1.5.2. Specific Objectives	5
1.6. Methodology of the Study.....	6
1.7. Scope and Limitations of the study.....	7
1.7.1. Scope of the Study	7
1.8. Beneficiaries of the study.....	8
1.9. Organization of the Thesis	8
CHAPTER TWO.....	9
2. LITERATURE REVIEW AND RELATED WORKS	9
2.1. Overview of MANET.....	9
2.1.1. MANET Characteristics.....	10
2.1.2. Architectural Model of MANET	11
2.2. MANET Routing Protocols.....	12
2.2.1. Proactive Routing Protocols (Table-Driven)	13
2.2.2. Reactive Routing Protocol (Demand-based)	13
2.2.3. Hybrid Routing Protocol.....	17
2.3. Mobility Model of Nodes	18

2.4.	Misbehavior of Nodes in MANET	20
2.5.	Selfish Nodes.....	21
2.5.1.	Characteristics of Selfish Nodes	22
2.6.	Algorithms to Solve Selfish Node Issues.....	23
2.6.1.	Reputation-Based Scheme	23
2.6.2.	Credit-Based Scheme (“Serve and Earn”).....	23
2.6.3.	Acknowledgment-Based Scheme	23
2.6.4.	Game-Theoretic Model.....	23
2.7.	Related Works	25
2.8.	Related Work Summary	30
CHAPTER THREE		31
3.	PROPOSED SELFISH PUNISHMENT ALGORITHM	31
3.1.	Overview of Proposed Approach	31
3.2.	General Assumption.....	31
3.3.	General Architecture of Proposed Selfish Punishment Algorithm	31
3.4.	Selfish Identification with Proposed Algorithm.....	33
3.5.	Selfish Punishment with Proposed Algorithm	34
3.6.	Flow Chart Diagram for Proposed Ack-AODV Algorithm.....	36
3.7.	Summary	38
CHAPTER FOUR		39
4.	SIMULATION AND PERFORMANCE EVALUATION	39
4.1.	MANET Simulation Tools	39
4.1.1.	Network Simulator Version 3 (NS-3) Simulation Tool.....	39
4.1.2.	Network Simulator Version 2 (NS-2) Simulation Tool.....	40
4.1.3.	Comparison of Simulation Tools	41
4.1.4.	Objective Modular Network Testbed in C++ (OMNET ++).....	42
4.1.5.	QualNet Simulation Tool.....	43
4.2.	Simulation Setup	44
4.3.	Performance Evaluation Metrics	45
4.4.	Simulation Result	46
4.5.	Analysis of Simulation Result.....	48
4.5.1.	Comparison Result in terms of Throughput.....	48
4.5.2.	Comparison Result in terms of Packet Drop rate.....	50

4.6. Evaluation Result	51
4.7. Discussion	53
4.8. Summary	55
CONCLUSION, FUTURE WORK, AND CONTRIBUTION	56
Conclusion	56
Future Work	56
Contribution	57
REFERENCES	59
APPENDIXES	65
Appendix A: Terminal commands to run a file for the Ns-2.35 simulator	65
Appendix B: Ns-2.35 Simulation Tool Directory	65
Appendix C: Sample tcl file for network configuration and node setup.....	66
Appendix D: Sample tcl file (selfish.tcl) Selfish Node Configuration	69
Appendix E: Sample Code for Selfish Node Assignment and Drop Packet.....	69
Appendix F: Sample Trace file (aadv.tr) Nodes Communication	70

LIST OF TABLES

Table 2-1 RREQ and RREP packet format.....	15
Table 2-2 Comparison summary of DSR and AODV	17
Table 2-3 MANET routing protocol summary	18
Table 2-4: Selfish node identification algorithms.....	24
Table 2-5 Related work summary	28
Table 3-1 Punished selfish node FLAG table	35
Table 4-2 Comparison of NS-3 and NS-2.....	41
Table 4-3 Comparison of OMNET++ and QualNet	43
Table 4-4 Simulation environment setup	44
Table 4-5 Factors affect performance evaluation metrics.....	46
Table 4-6 AODV vs Ack-AODV.....	51
Table 4-7 AODV Vs Ack-AODV.....	52

LIST OF FIGURES

Figure 1.1 Mobile Ad-hoc network.....	2
Figure 1.2 Methodology of proposed research study.....	7
Figure 2.1 MANET characteristics	10
Figure 2.2 Architecture of MANET.....	12
Figure 2.3 MANET routing protocols classification	13
Figure 2.4 AODV route discovery (RREQ flooding and RREP unicast).....	15
Figure 2.5 Route maintenance (RERR)	16
Figure 2.6 Mobility models of nodes	20
Figure 2.7 Classification of misbehavior nodes in MANET	21
Figure 2.8 Effect of selfish node	22
Figure 3.1 General architecture of proposed solution.....	32
Figure 3.2 Proposed system model	36
Figure 3.3 General transmission flow of Ack-AODV proposed algorithm.....	37
Figure 4.1 Overall architecture of NS-3.....	40
Figure 4.2 Overall architecture of NS-2.....	41
Figure 4.3 OMNET++ module structure.....	42
Figure 4.4 QualNet architecture.....	43
Figure 4.5 Node representation on AODV in MANET	47
Figure 4.6 Selfish nodes on AODV in MANET	48
Figure 4.7 AODV vs Ack-AODV in terms of throughput.....	49
Figure 4.8 AODV vs Ack-AODV in terms of E2E delay.....	50
Figure 4.9 AODV vs Ack-AODV in terms of packet drop rate	51
Figure 4.10 Comparison summary of AODV vs Ack-AODV.....	53

ACRONYMS AND ABBREVIATIONS

AWK	Alfred Aho Peter Weinberger and Brian Kernighan
Ack-AODV	Acknowledgement based Ad-hoc On-demand Distance Vector
AODV	Ad-hoc On-demand Distance Vector
CLI	Command Line Interface
CPU	Central Processing Unit
DDR	Distributed Dynamic Routing
DN	Destination Node
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
E-TwoACK	Enhanced Two Acknowledgement
GM	Gauss-Markov
GUI	Graphical User Interface
INs	Intermediate Nodes
iNCV	Improved Neighbor Credit Value
MANET	Mobile Ad-hoc Network
NS-2	Network Simulator 2
NS-3	Network Simulator 3
OLSR	Optimized Link State Routing
OMNET++	Objective Modular Network Testbed in C++
RREP	Route Reply
RREQ	Route Request
RW	Random Walk
RWP	Random Waypoint
SLFN	Selfish Node
SN	Source Node
SNRRM	Selfish Node Removal using Reputation Model
TBUT	Token-Based Umpiring Technique
TORA	Temporally Ordered Routing
TWO-ACK	Two Acknowledgement
WRP	Wireless Routing Protocol
ZHLS	Zone-based Hierarchical Link State
ZRP	Zone Routing Protocol

ABSTRACT

Link breakage is a routing challenge in Mobile Ad-hoc Network (MANET) which affects the performance of Ad-hoc On-demand Distance Vector (AODV) routing protocol. Several reasons are stated for link breakage, but one of the major challenges is the presence of Selfish Nodes (SLFNs). In AODV nodes are assumed as a Normal Node (NN), but some nodes may behave selfishly to conserve their resources such as power and memory. This selfish behavior of nodes leads to refuse participation and drop packet which issued from another neighbor node. This selfishness misbehavior leads to several impacts as such; network partitioning, refusal to accept requests, and packet drop. To resolve the above issues, in this research Acknowledgement based Punishment on AODV (Ack-AODV) algorithm is designed. The proposed Ack-AODV algorithm identifies SLFN and punishes them to be triggered and become a NN in the network. Based on node residual energy and their request replay credit the nodes which drop a packet and have low credit value are identified as SLFN and will be blocked from any service. By applying a punishment mechanism those SLFN that are blocked will be demanded to cooperate. If SLFN reply for the acknowledgment request sent from an authorized NN and earns a credit will be revived and free from the blocked state. To analyze the performance of the proposed Ack-AODV algorithm ns2 network simulator tool is used. Based on the generating result Ack-AODV performance is measured by comparing with existing AODV. As the simulation, the result indicates Ack-AODV has achieved a good performance compare to the existing AODV. By deploying a total of 50 nodes the network throughput is increased from 80.7% to 86.3%, however end-to-end delay and packet drop rate is reduced from 35% to 23.3% and 34.39% to 25.78% respectively. To conclude, based on the comparison result Ack-AODV has a good performance in the above-mentioned metrics compared with existing AODV in the presence of SLFN. `

Keywords: *Acknowledgement based Punishment, AODV, Link breakage, MANET, Selfish nodes.*

CHAPTER ONE

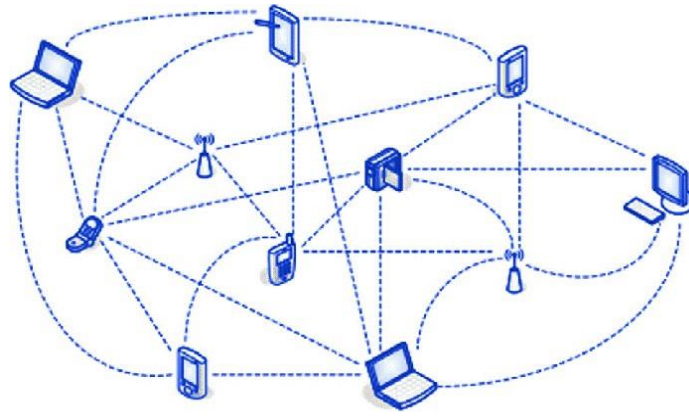
1. INTRODUCTION

1.1. Background of the Study

Wireless networks are preferred for multiple solutions and improvements implemented on top of wireless technologies such as computers or smartphones. Technology and network integration gained attention from researchers and brought an expanding investigation intrigued by mobile wireless networks. Mobile Ad-hoc Network (MANET) is one area of continuous research issue since it gives promising solutions to real problems in wireless and mobile networks (Alvarez Aldana et al., 2018).

An ad-hoc network is a sub-category of a wireless network designed without infrastructure. The scheme nature of the ad-hoc network is preferable and widely used for fields such as military sectors in reconnaissance, rescue missions, and emergency or disaster relief actions. However, its characteristics existed without infrastructure with high interference and the dynamic connection is the main problem for data transmission (Ragab, 2020).

MANET is a well-known short-lived (Ul Islam Khan et al., 2018), self-governing and fully self-structured network having the capability to work anywhere without infrastructure (Sharma et al., 2019). MANET does not need a central authority in which nodes behave like a transmitter or receiver can be declared as a router. MANET networks are dynamic by their nature with high node movement. Spontaneous devices connection challenges to develop an accurate, efficient, effective, and reliable routing protocol to establish communication. As shown in the next figure (Nagendranath & Babu, 2020), communication goes through the node themselves within the transmission range. Source Node (SN) connect by means of Intermediate Nodes (INs) to reach Destination Node (DN). All nodes are located in the same level and has the same privilege, one-time node will be a router and can be a transmitter other time.



1

Figure 1.1 Mobile Ad-hoc network

In MANET considering self-configured nodes, it is crucial to look at routing protocols used for finding a path and transferring the data packets within or between the networks from source to destination. According to their functionality routing protocols are categorized into three different types.

Proactive (also called table-driven) routing protocols in which all routing information is usually reserved in tables. Whenever the network topology deviated these tables are consistently updated, according to the change, such protocols are Optimized Link State Routing (OLSR), Destination Sequenced Distance Vector (DSDV), and Wireless Routing Protocol (WRP). Reactive (demand-driven) routing protocols are designed to set up routes when required and discover the destination. A flooding technique is used to broadcast requests by the source until the destination is found. Dynamic Source Routing (DSR), Ad-hoc on-demand Distance Vector (AODV), Temporally Ordered Routing Algorithm (TORA) are mentioned as reactive protocol. Hybrid routing protocols are the third protocol that reduces rebroadcasting and create route discovery within a selected group of nodes. Zone Routing Protocol (ZRP), Zone-based Hierarchical Link State (ZHLS), and Distributed Dynamic Routing algorithm (DDR) are a type of hybrid protocol (Kaur & Thakur, 2019).

Above all regarding MANET routing protocol, reactive protocol applies a demand-based method that establishes a communication based on request which has less routing overhead compared to other routing protocol categories. Specifically, AODV is adaptable to implement

¹ https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FMobile-Ad-Hoc-Network-architecture-MANET_fig1_316291648&psig=AOvVaw2fMCRw2tCKxKe3HamI9QLP&ust=1645565246310000&source=images&cd=vfe&ved=0CAgQjRxqFwoTCLDc17PekfYCFQAAAAAdAAAAABAE

in highly dynamic networks. In AODV INs participate on behalf of the SN by keeping the route information of the target node address only. Which makes it uniquely chosen from the rest of the protocols.

Due to having the properties of infrastructure-less, unstable topology, energy and bandwidth constrained, or frequent routing update for data transmission impact routing performance of MANET lead to unreliable communication. One of the challenges in routing protocols was facing the unpredictable change of the topology which varies rapidly over time by the mobility behavior of the nodes. When nodes move with high mobility, a power constraint happens. Power constraints in routing make nodes step away from the range of the network or change the behavior of nodes. Nodes need to gain maximum profit to persist their resource such as energy, CPU, Memory by refusing to the participation of routing process. These nodes titled as Selfish Nodes (SLFN) that save their power and are unready to resend packets that come from Normal Nodes (NN) and drop other node packets (Aifa & Thomas, 2018) (Devi & Gill, 2019).

In AODV, SLFN is categorized into three types based on their refusal participation. SLFN type 1, nodes are reluctant to forward the data packet but can join in the process of route discovery and maintenance phase. SLFN type 2 does not participate in the discovery, maintenance process as well as data forwarding too. In SLFN type 3 acts like NN when their energy level is maximum but they stop participating when the energy level is getting low (H. Yadav & Pati, 2018). When SN appears in between it restricts an exchange with legitimate nodes and is unwilling to retransmit or reply to hello messages. Not participating in the route discovery or maintenance phase is a reason for link breakage and network partitioning. This behavior of node is a factor for packet drop, transmission error, communication loss, and overall degradation of routing performance.

Considering the performance of a network most recent researches were proposed SLFN identification and isolation algorithms that brought visible change and improvement on network performance. Some of the detection approaches such as the reputation method (nodes need feedback to differentiate selfishly nodes and NNs, a node with low reputation value considered as SLFN). Credit-based method (during forwarding packets if nodes earn a poor credit it detects as SLFN). Acknowledgment method (nodes expect reply packets to ensure the acceptance of the original node packet). Game theory (based on action and decision player payoff considered as an outcome for SLFN detection). Moreover, SLFN punishment and motivation algorithms are also proposed to empower SLFN rather than isolating from the

network. This study majorly contributes to minimizing the link breakage due to the self-centered behavior of nodes in AODV by preparing a technique that identifies and punishes during route discovery.

1.2. Motivation of the Study

MANET is a promising technology that offers temporary connections without pre-existing setup needed during unusual circumstances like emergencies, disastrous recuperation ranges, and battlefields (Khudayer et al., 2020). The uprising flexible use of mobile devices and a large number of mobile data traffic is a reason for MANET to design with no pre-existing infrastructure. The structure and type of protocol in MANET is more vulnerable to different challenges. AODV standard assumes nodes are cooperative but due to energy/power drained those nodes refuse to participate with other nodes. The presence of selfish behaved nodes affects the network by degrading the routing performance (increases end-to-end delay, high packet drops rate, and low throughput). Refusing to forward packets results in link breakage (Shan et al., 2020). The above issue of MANET motivates us to propose a solution to this concept of AODV related limitation of SLFN. The main motivation of this work is studying and analyzing the effect of those SLFN and proposing the Ack-AODV algorithm to punish them to work together. This study simulates a SLFN impact with MANET of AODV. The simulation of the thesis evaluates in terms of those performance metrics like the throughput, end-to-end delay, and packet drop rate.

1.3. Statements of the Problem

AODV is a self-setup network with inclusive separate wireless nodes which create links with each other. Collaboration of nodes should be mandatory to provide the basic functionality of the network by assuming nodes are legitimate. In AODV nodes are spontaneous and can move freely. But because of mobility in this type of network resources are limited. To conserve a resource like battery power or bandwidth nodes change their behavior and act selfishly. Such nodes are called Selfish Nodes (SLFN). SLFN saves power and maximizes their benefit by refuse participate in a route discovery process and dropping RREQ (route request) and RREP (route reply) packets. When sender node broadcasts RREQ packet to find an optimal route to DN, an IN broadcast on behalf until it reaches to the targeted destination but SLFN in between drop request and reply packet. This selfish misbehavior of nodes causes link breakage which severely degrades network performance and reduces network reliability. Link breakage in

AODV causes retransmission (delay until packets reach to destination), dropping data, and decreasing throughput (Yadav & Pati, 2018).

Since performance is an issue SLFN affects functionalities and performance of AODV by saving its resource to maximize its benefit. Existence of SLFN makes normal cooperating nodes become unfairly loaded during routing which degrades the performance. For this reason, SLFN become a reason for link failure (Sharah et al., 2020) (Susan et al., 2020) (Education et al., 2021). The above issues are solved by the proposed Ack-AODV algorithm. The simulation is performed under standard AODV using a simulation tool called Ns-2. From this study, the SLFN impact under AODV improved using the acknowledgement based punishment approach.

1.4. Research Questions

Based on the problems stated in section 1.3, this study stated three research questions. These research questions were used to guide the research until its completion. The following questions should be answered at the end of this research work. These research questions are:-

RQ1: How does Ack-AODV way of identifying improve the performance of AODV?

RQ2: How to simulate the proposed algorithm using the NS-2 network simulator?

RQ3: How to evaluate and compare the newly proposed algorithm with a well-known existing AODV algorithm in respect of throughput, end-to-end delay, and packet drop rate?

1.5. Objectives of the Study

1.5.1. General Objective

The general objective of this research was to improve AODV using an Acknowledgment based punishment algorithm for MANET in the presence of a selfish node.

1.5.2. Specific Objectives

To succeed in the general objective, the following specific objectives need to be accomplished.

- Propose Ack-AODV algorithm way of identifying and punishing to improve the performance of AODV.
- Simulate the proposed algorithm using the NS-2 network simulator.
- Evaluate and compare the newly proposed algorithm with a well-known existing AODV algorithm in terms of throughput, end-to-end delay, and packet drop rate.

1.6. Methodology of the Study

To reach the above objectives of the study stated in section 1.5, the following research methods are specified. The employed approach is used to respond the research questions stated in section 1.4. This methodology phases are:

- **Literature Review**

To achieve the above objectives different research papers, conference papers, journals, articles were studied to understand the research work and to collect relevant information based on SLFN identification and punishment techniques. In this phase literature were reviewed and conducted to understand the research area regarding the effect of SLFNs in MANET. The problem was identified based on the gap to propose a solution.

- **Propose a Solution**

After problem formulation through different literature reviews, the proposed solution was modeled to increase the performance by identifying and punishing SLFNs.

- **Design Ack-AODV Algorithm**

The newly proposed algorithm was developed based on the existed AODV protocol which considers SNs. The solution mainly proposed to promote SLFN to be cooperative rather than isolating them. Nodes residual energy (which is derived from nodes initial and current energy) and credit of nodes based on reply packet are used to detect a selfish behaved node. The punishment strategy proposed a FLAG table that tags to the SLFNs cache table to prevent the network services participation at all.

- **Simulate the Proposed Algorithm in NS2**

Hardware and Software Tools will be used to simulate the proposed algorithm. As a software tool, Ns-2 network simulation tools were used and preferred to work on MANET using AODV protocol. In designing diagrams and flowcharts Lucid chart and E-draw max architecture and diagrams drawing chart were used.

- **Evaluate the New Algorithm and Compare it with the Existing Algorithm**

The proposed algorithm is compared with the existing one by taking a different parameter (throughput, end-to-end delay, and packet drop rate) to evaluate performance. Generally, the phases are pictured in the following diagram.

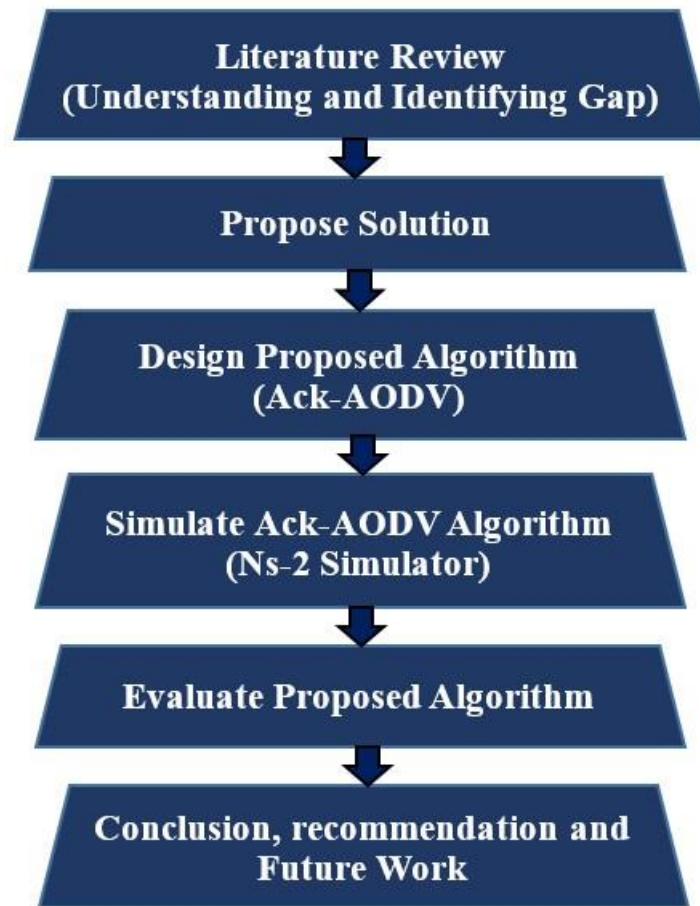


Figure 1.2 Methodology of proposed research study

1.7. Scope and Limitations of the study

1.7.1. Scope of the Study

This research work mainly focuses on minimizing the effect of link breakage caused by SLFN under AODV routing protocol by identifying and punishing them to change to a cooperative normal node to participate in the AODV routing process especially in route discovery to eliminate the refusal, delay, ignorance, not participating on route request process. The proposed acknowledge-based scheme used in AODV protocol in a route discovery phase uses a special message type other than hello message as ACK technique for selfish node detection which helps to punish them.

1.7.2. Limitations of the Study

Since ad-hoc network specifically MANET is a vast research area many issues were raised regarding the performance improvement on MANET distinctively on AODV protocol. This thesis is limited with only nodes which behaved as SLFN who drop route discovery control packets which be a reason for link breakage. There is a list of reasons for link breakage like mobility, power exhaustion, security issues, but this work is only limited and works on node behaviors particularly on SLFNs which lead to link breakage. Additionally, the study was limited with SLFN thus does not incorporate smart SLFN and malicious nodes misbehavior.

1.8. Beneficiaries of the study

Wireless technologies particularly the uprising use of wireless devices in MANET brought a new way of experiencing in different application areas. MANET is hard to set up as a complete communication network by its instability. Mostly applied in rough, disastrous or, rescue mission areas. Other than these areas from this research some of the beneficiaries are commercial, educational sectors, and used for sensor-based networks. Application in MANET varied from small-scale to large-scale networks which are controlled by power sources.

1.9. Organization of the Thesis

The content in this research study is categorized into four chapters generally. Each chapter describes different contents based on the chapter topic. The first chapter explains all about introduction along with a background of the study, statements of the problem, objective of the study, research questions, methodology, and significance of the study. The second chapter describes the whole literature review and related work regarding selfish node behaviors under AODV in MANET. The third chapter briefly describes the proposed selfish identification and penalized solution and designs the architecture and transmission flow in the proposed algorithm. In the fourth chapter simulation study and results are discussed and precisely described. Finally, the last describes the conclusion, recommendation, and further future studies in this research area.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORKS

2.1. Overview of MANET

The development of MANET was characterized by different generations. The early generation of MANET was coined around the 1970s particularly in 1972 introduced as a Packet Radio Network (PRINT). The PRINT program was made to test conflicted areas by providing different networking properties like Aerial Location of Hazardous Atmosphere (ALOHA) and CSMA (Carrier Sense Medium Access). PRINT was considered as distance vector routing protocol type for medium access control. In the late 1980s, the early generation evolved which was considered as the second generation of ad-hoc network introduced SURAN (Survivable Adaptive Radio Network) (Ul Islam Khan et al., 2018)(Shrivastava, 2020). SURAN was provided with an improved radio device performance in an infrastructure-less network withstand in war and competitive environments (Ul Islam Khan et al., 2018). The rise of the ad-hoc network was developed around the 1990s not only for dangerous areas also for commercial or non-militarized purposes too. The development of various mobile devices was a reason for the recent concept of the ad-hoc network were born by Internet Engineering Task Force (IETF) worked on standardizing regular routing protocol for MANET (Puri & Arora, 2014).

MANETs are many associations of mobile nodes in which they communicate between them with the help of radiofrequency and/or infrared with no pre-build/existing infrastructure. MANET is a remote network has dynamic topology because of high mobile or moveable nodes sometimes leave and break the connection of the network range (N. Yadav & Chug, 2019). MANET can be used in any situation where temporary communication is required. MANET is considered an autonomous network in which communication of mobile nodes is restricted within its facilities (Kushwah et al., 2019). The primary purpose of MANET is to expand mobility to independent mobile and wireless domains where several nodes can be merged to form routers and hosts as well. Due to the above primary purpose mobility are a base that refers to actions and their roaming in various domains on the Internet (Alo et al., 2018).

2.1.1. MANET Characteristics

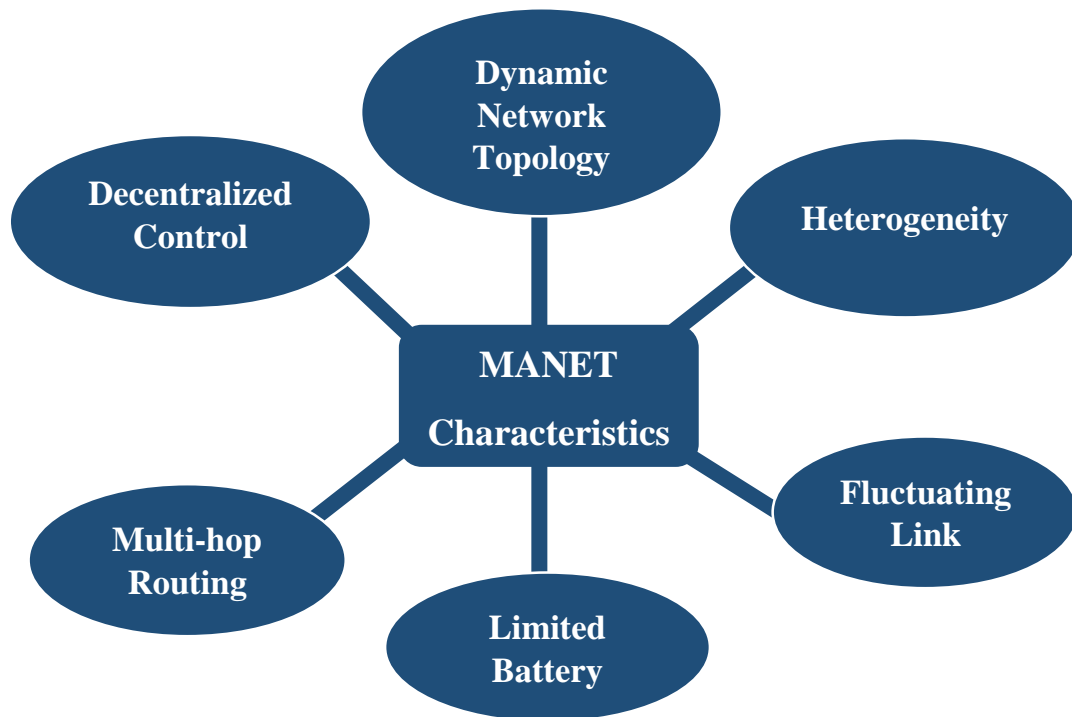


Figure 2.1 MANET characteristics ²

According to N. Yadav and U. Chug (N. Yadav & Chug, 2019), MANETs have the following characteristics:

Dynamic Network Topology: nodes can flow freely in every path and create routing between themselves. Nodes are active at one time and passive at another time because of the high movement of devices.

Decentralized Control: MANETs rely on the collaboration of partaking hubs. Nodes participate and communicate with each other for packet transmission.

Restricted Battery Power: hosts are limited with little battery power which makes them leave and reappear again in the MANET network creating destruction and loss of packet during transmission.

Multi-hop Routing: communication of information from source to destination goes through once or combination of intermediate nodes in their transmission range (Journal, n.d.-a).

² https://www.researchgate.net/figure/Characteristics-of-MANET_fig2_324753716

Heterogeneity: unlike wired networks, some nodes are heterogeneous. It is conceivable to establish an ad-hoc network between laptops, smartphones, sensor devices, and palmtops. Although connection can be formed between similar devices such as a couple of laptops.

Fluctuating Link Capacity: the unstable fashion of wireless link capacity never be constant because of fading, environmental factors, or device limitations. To compensate for fluctuation OFDM (Orthogonal Frequency Division Multiplexing- transmitting significant amount of digital data over radio waves) is used in MANET physical layer (John Justin Thangaraj et al., 2019).

2.1.2. Architectural Model of MANET

MANET is made up of different technologies implemented in remote areas. As figure 2.2 explained below, MANET architecture is categorized into three main sub-divisions which are enabling technologies, network layer, and middleware and application.

Enabling technologies: basic enabling technologies which are used parallel with the birth of MANET. Those enabling technologies were used and implemented in MANET application fields. This technology consists of PAN (Personal Area Network), LAN (Local Area Network), WAN (Wide Area Network) including Bluetooth and 802.11 standards.

Network layer: ensures how the information can be transferred and communication between source and destination devices in a reliable way. In this layer exact location of a device need to map along with the device's logical address.

Middleware and Applications: more integrated with user-side applications which are managed by middleware services like service location, shared memory in the network, and group communication (Raj & Neeraja, 2019).

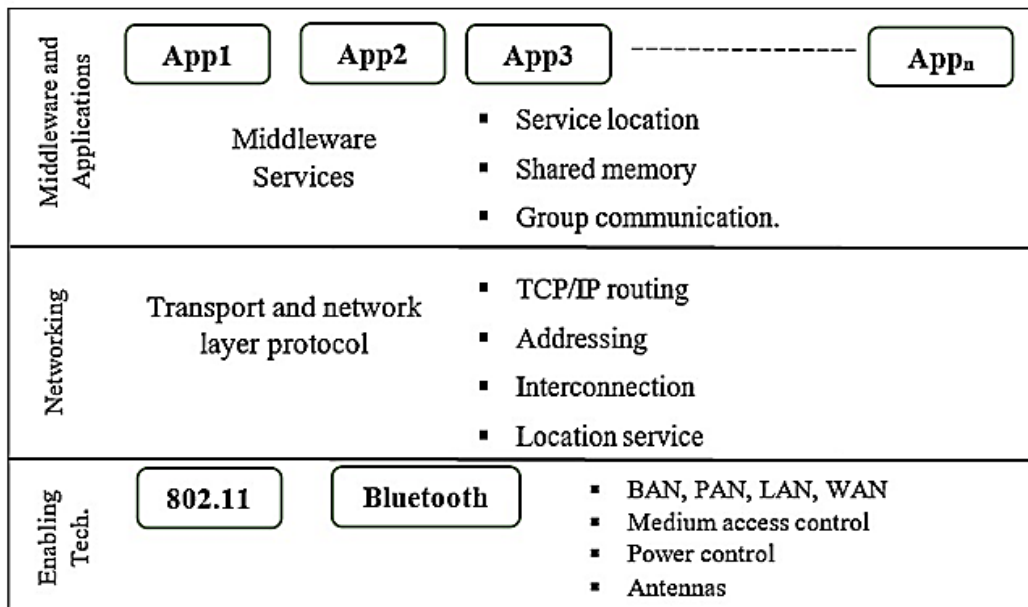


Figure 2.2 Architecture of MANET ³

2.2. MANET Routing Protocols

Communication of nodes implemented through single-hop or multi-hop method for data exchange in MANET's network undertakes by INs. Due to the shortage of nodes, the communication is implemented through multi-hop. Nodes exchange data using routing protocols. Routing protocols are designed to routes and transport packets from the SN to the DN. Various routing protocols are designed in MANET's network which improves the performance of the network (Al-Dhief et al., 2018). Broadly two routing protocol approaches are stated, topology-based routing protocol (desired to know link information of nodes for packet forwarding) further classified as proactive, reactive, and hybrid. Unlike topology based routing protocol position/geographic-based routing protocol (store information on one hop neighbor accessible thru radio than having global view) GPSR (Greedy Perimeter Stateless Routing) most known position-based protocol (Arnous et al., 2019). Prominent MANET protocols are presented in figure 2.3.

³ https://www.researchgate.net/figure/MANET-Architecture-5_fig3_322112847

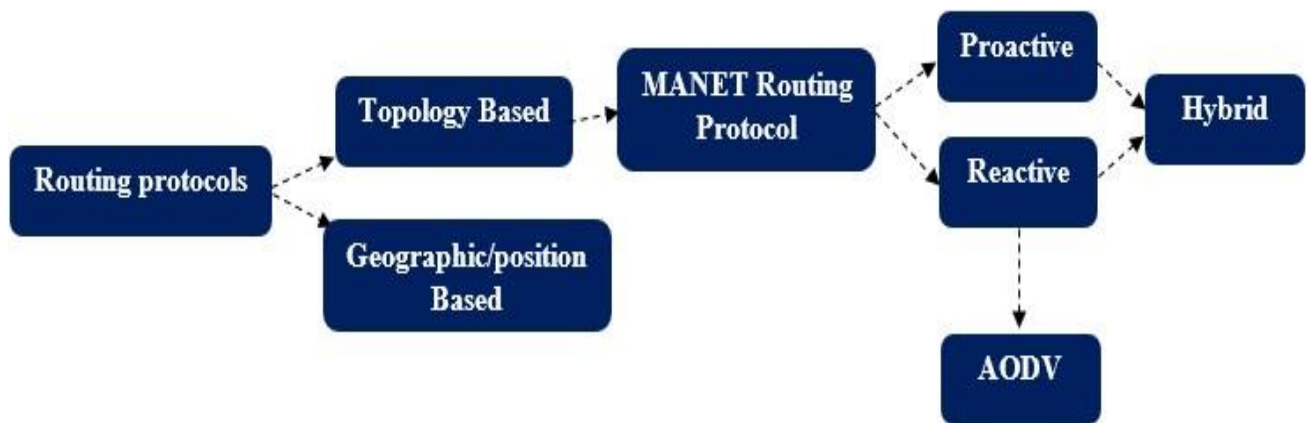


Figure 2.3 MANET routing protocols classification ⁴

2.2.1. Proactive Routing Protocols (Table-Driven)

Proactively maintains routing information to the entire nodes at any time. Nodes when using proactive routing protocols exchange routing information periodically and are stored in the routing table (Alkahtani & Alturki, 2021). Adopting this protocol is helpful since it isn't necessarily a route discovery procedure all the time. Due to the pre-established routing table in which nodes use during forwarding packets nevertheless maintaining routes with entire nodes all the time creates routing overhead which impacts the performance of the network (Arnous et al., 2019). Proactive is inefficient for big networks because all nodes' information in the routing table requires updates when there is a periodical topology change (Devi & Gill, 2019). DSDV, OLSR. Considered as an example of the proactive protocol.

2.2.2. Reactive Routing Protocol (Demand-based)

Reactive is also named an on-demand routing protocol or source-initiated protocol (Elleithy & Loud, 2019). In this protocol, the networks do not keep information for all nodes all the time, but to get information about the DN if it is in the same network to connect and send a message, the sender sends a request on-demand to the neighbor nodes until it arrives at the targeted node. AODV, DSR are some of the examples of reactive routing protocols.

DSR: dynamic source protocol particularly designed for multi-hop routing which maintains a route when there is a demand which minimizes routing overhead. DSR works based on route discovery and maintenance approach. If SN couldn't get a direct link, route discovery is initiated to create a connection with the DN. Similarly, route maintenance is used in the time of active route error. In the process of the route discovery approach, each node maintains route

⁴ https://www.researchgate.net/figure/VANET-Routing-Protocols-35_fig2_276893197

information that has been learned through the route cache. When the SN tries to send data to a DN foremost source reviews route-cache if the route exists or not. If the source found the route it transmits the packet with it, if more paths are found route selection is applied to select a single path. However, if no route is available a route discovery process is introduced. Route discovery is performed by broadcasting RREQ. If the IN receives RREQ which is unfamiliar or not seen before they attach their id and rebroadcast, it again. RREP packet is then generated by the target destination. Nodes create a RERR packet when link breakage is noticed and sent to the original node for new route discovery (Arnous et al., 2019).

AODV: A unique reactive routing protocol designed to route packets from source to destination when there is a demand that is requested by active SNs. It uses bidirectional links (route discovery and route maintenance). Creates information about network routes only when the network demands it. There are two basic operations performed in the AODV routing protocol. A control message (to find a route to the DN) and data messages (contain the actual data and transfer for a specified destination). In AODV control messages have two categories. Route discovery phase and route maintenance phase.

Route Discovery Phase: When SN demand to forward data to the destination, first it checks the routes if they are available to the destination. If the route is available, the packets forward to the next node which is called the neighbor node. But if the route is inapproachable and the source wants to connect, the SN initiates a request if the destination is willing to connect. Route discovery has three phases of control messages: RREQ (Route Request) and RREP (Route Reply).

- **RREQ (Route Request):** created by the SN that desires to initiate a communication with nodes in the network by RREQ message. RREQ hold address of the source, sequence number of the source, address of the destination, sequence number of the destination, and broadcast id. RREQ messages are broadcast with the flooding technique. Then the nodes which have the control packet forward to other neighbors. To prevent sending the same RREQ packet repeatedly AODV uses a sequence number. The sequence numbers that are used will ensure the route freshness and also determine the exact path and updated path to the destination. As shown in figure 2.2 SN “A” broadcasts RREQ packet to INs to establish a path with DN “J” (Deepak & Anandakumar, 2019).

- **RREP (Route Reply):** used as a confirmation or acknowledgment from the DN to the SN to connect. After the RREQ packet reach at the destination the forwarding process ends and relays the RREP packet which is the reverse route if the RREQ IP address matches. The DN responds to RREP messages to the source in a unicast form. Figure 2.2 describes the RREP process which is taken by a single path from DN “J” to SN “A”.

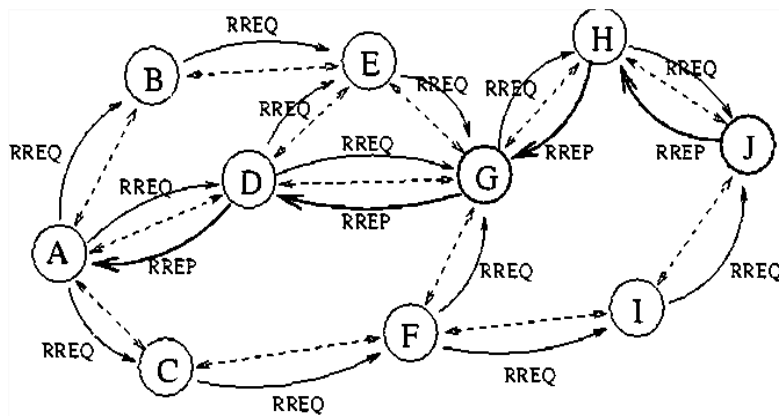


Figure 2.4 AODV route discovery (RREQ flooding and RREP unicast) ⁵

Table 2-1 RREQ and RREP packet format ⁶

<i>AODV RREQ packet</i>	<i>AODV RREP packet</i>
Source IP address	Destination IP address
Source sequence number	Destination sequence number
Destination IP address	Source IP address
Destination sequence number	Source sequence number
Broadcast ID	RREQ ID

Route Maintenance Phase:

- **RERR (Route Error):** During the process of sending packets all nodes are active and monitor their neighbor node. This process is performed to check whether the nodes are active in the participation. But some nodes sometimes failed to be active because of reasons such as if the node energy is inefficient which enforces to leave the node, the rapid motion of the nodes. In case a node gets failed and the neighbor node notices the

⁵ <http://www.gyanvihar.org/journals/wp-content/uploads/2018/12/fig-1.png>

⁶ https://www.researchgate.net/figure/Modified-AODV-RREQ-packet-and-modified-AODV-RREP-packet_fig5_257877888

failed node it sends RERR messages. It identifies nodes that are not contributing to the routing process by informing the neighbor node about the link breakage (Al-Dhief et al., 2018).

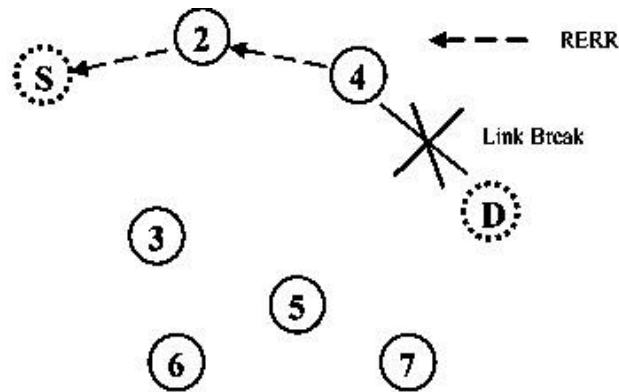


Figure 2.5 Route maintenance (RERR) ⁷

Advantage of AODV

- Routes are established between nodes based on request/ demand.
- AODV uses a sequence number in their packet format (RREQ & RREP) to figure out latest or freshness of a path during the route to the destination.
- When a high topological change occurs and affects active nodes AODV has a high response to the change.
- AODV allows unicasting and multicasting transmission.

Disadvantage of AODV

- A solitary request has multiple route reply responses result in massive control overhead.
- Route learning (route information) is limited only to route packets forwarded by the source.
- AODV assumes that all nodes must be cooperative to set up a route, but if the nodes are uncooperative route setup isn't done. Because of this property, AODV is poor, vulnerable, and defenseless to various kinds of attacks.
- Unnecessary bandwidth consumption occurs because of frequent hello messages in AODV (Richhriya, 2020) (Talawar & Ashoka, 2020).

⁷ https://www.researchgate.net/figure/AODV-Route-Error-message-generation_fig1_303311755

Table 2-2 Comparison summary of DSR and AODV

<i>DSR</i>	<i>AODV</i>
▪ Contain full routing information	▪ Contain targeted address only
▪ Increase routing length, grow packet size header and increase traffic/link breakage due to SN	▪ Reduce data header size as route information is distributed to INs
▪ More routing overhead due to flooding RREQ	▪ Less routing overhead
▪ Good for low mobility network	▪ Adaptable to highly changeable network
▪ Doesn't use the periodic hello message	▪ Use the periodic hello message
▪ Uses route-cache	▪ Uses routing table
▪ Does not use sequence number	▪ Use sequence number to find the latest route
▪ Route information keeps in the source	▪ Route information keeps in the INs

2.2.3. Hybrid Routing Protocol

Inherit the features of proactive and reactive routing protocols. It is most preferable for hierarchal routing in which nodes can be organized in a cluster or zone. Hybrid protocol contributes by decreasing rebroadcast and initiates route discovery for selected zone only. Hybrid routing protocol merges the advantage of both (proactive and reactive) to overcome their defect. Hybrid nodes are located in a predefined zone were divided into inside zone (nodes communicate proactively with other zones) and outside zone (nodes act reactively within its neighborhood). To preserve topological information one of the limitations is consuming memory and power (Tabbana, 2020) (Devi & Gill, 2019).

Table 2-3 MANET routing protocol summary

<i>Characteristics</i>	<i>Proactive</i>	<i>Reactive</i>	<i>Hybrid</i>
Routing Overhead	High (Dynamic Topology)	Low	Medium
latency	Less (routing table)	More (flooding)	Inside zone (Low) Outside zone (High)
Routing	Source routing	Hop by Hop routing	Zone-based
Routing Updates	Periodic	Event-based	Both
Traffic control	High	Less	Medium
Design Nature	Small, slowly moving and frequent data transmission in MANET network	Large, fast-moving, and infrequent data transmission in MAET network	Large network
Energy requirement	More	Less	Medium

2.3. Mobility Model of Nodes

MANET's network is designed as a wireless ad-hoc network in which nodes are portable with a dynamic topology that moves freely in any direction. But nodes need some kind of way to manage their movement. Mobility model analyze and determine the behavior of mobility form of mobile nodes along with routing protocols. Nodes can move towards any path and speed but in MANET simulation mobility models are used to discover the network performance (Appiah & Cudjoe, 2018).

Mobility models represent and manage MANET mobile devices' movement and change of speed and direction through time. Their movement pattern is distinguishing by mobility model type and routing protocol-specific characteristics which apply to the models. In MANET simulation study mobility model is classified into two ways: trace, real mobility pattern obtained through genuine experiments which exist in real life. Synthetic the second simulation method try to represent mobility movement of nodes during unavailability of trace through data generated statistical method (Shrivastava, 2020). Some of the mobility model categories are explained below:

Random Waypoint (RWP) Mobility Model

RWP is a common method used in MANET which is one of the entity mobility model classes. In the RWP model movement of nodes are free but, during their movement, they exist sometimes in a position which is a pause time. Those moving nodes have a randomly specified starting coordinate with random velocity and speed to select their random destination position when their pause time is up. When the targeted destination is reached it stays for some time interval, then it changes to other determined positions randomly. This operation is continual until the processing time is complete (Israr et al., 2017).

Random Walk (RW) Mobility Model

Another type of well-known entity mobility model in which the portable nodes change position spontaneously from their current place through random selection of speed and direction. During their position change, new speed and direction are selected based on a predefined minimum and maximum speed and direction range. In the RW model simulation area when nodes reach the border, a new direction is calculated by determining the incoming direction to bounce to another new position. In this model, movement occurs over some time interval/constant time interval or in a fixed distance at the end new speed and direction are computed. In RW the entities do not choose a destination and do not have a pause time unlike RWP (Hamedani, 2018) (Naik et al., 2019).

Gauss-Markov (GM) mobility model

A type of model which is built with memory. Node current direction and speed depend on past stored direction and speed values. In this model each time when nodes' direction and speed are updated nodes have to set time then they change their direction and speed based on updated values (Bugarcic et al., 2018).

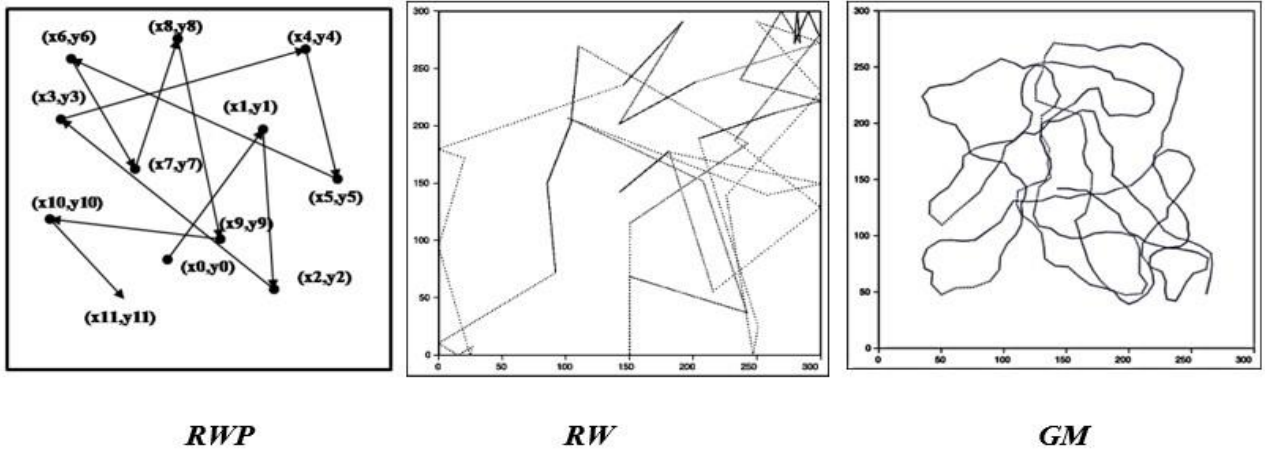


Figure 2.6 Mobility models of nodes ⁸

2.4. Misbehavior of Nodes in MANET

Security issues in MANET are a big concern. Mostly exposed to issues as a result of being wireless infrastructure-less network, has a dynamic changing topology, no central administrator at all, and cooperativeness of nodes. From those security issues, the node's cooperation is the main concern. Resource and energy usage is high during route detection and forwarding packets. When node energy gets lower most probably leave the network, so to be an active node and to stay in the network they try to preserve their resource by being a SLFN which shows selfish behavior (Aifa & Thomas, 2018). Based on their participation behavior there are three types of nodes in MANET. A normal node or regular node participates in all routing processes which is known as a cooperative node, SLFN needs to salvage their resources and refuse to participate, Malicious nodes have misled a direction to the wrong way during forwarding data packets to collapse and crash the network in which power is not a primary concern. Malicious nodes are hard to detect because of their hidden behaviors from the network. (Mohamed Musthafa et al., 2020).

⁸ https://www.researchgate.net/figure/Traveling-pattern-of-an-MN-using-the-Random-Waypoint-Mobility-Model_fig1_236161928
https://www.researchgate.net/figure/Traveling-pattern-of-an-MN-using-the-2-D-Random-Walk-Mobility-Model_fig3_236161928
https://www.researchgate.net/figure/Traveling-pattern-of-an-MN-using-the-Gauss-Markov-Mobility-Model_fig11_236161928

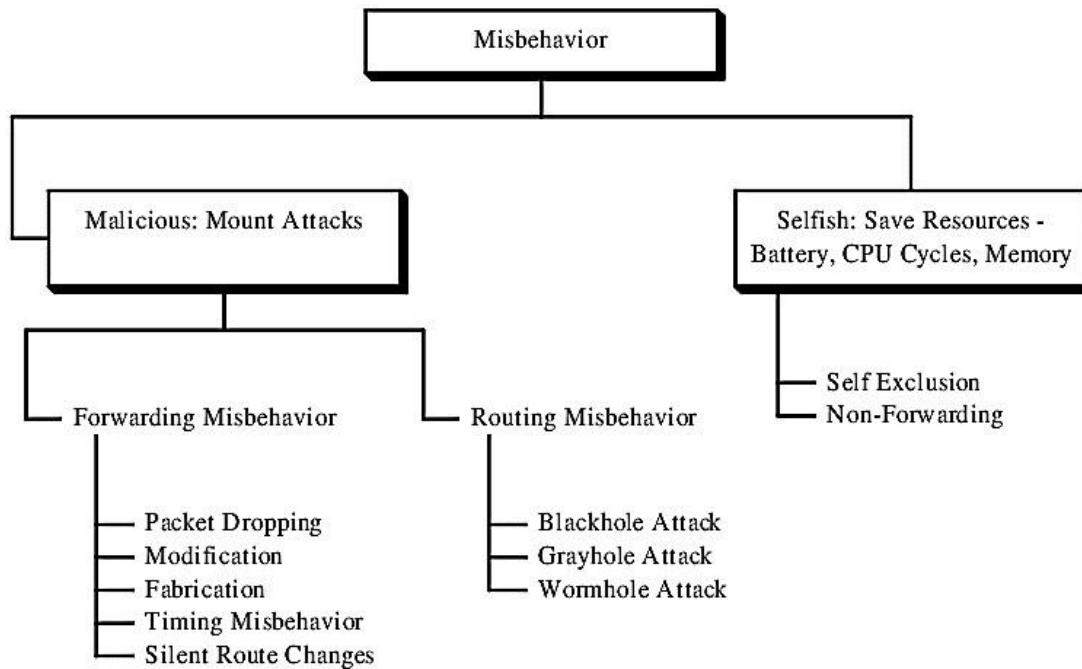


Figure 2.7 Classification of misbehavior nodes in MANET ⁹

In AODV route is performed through a high changeable network. The change could affect the nodes and marks to be selfish or malicious misbehaved nodes. This study area majorly focuses on selfish misbehavior nodes which is a reason for link breakup due to the power limitation in a spontaneous network. When misbehaving nodes manifest as malicious node it is difficult for the system to deal with them, which maximize the damage to the system for its benefit. The only method to prevent this misconducted node is detecting and isolating from the network but isolating decreases the probability of nodes reappearing again. Nevertheless, misbehaving nodes manifest as SN the system can still handle and are always predictable. They always behave to maximize their benefits but making them a cooperative node is the best option here. SLFN can be self-exclusive which the nodes exclude the routing process. The SLFN does not participate when route discovery is performed which saves its power. The other selfish behavior is selfish non-forwarding misbehavior in which SLFN participates in the route discovery phase but refuses forwarding packets (Sen, 2010).

2.5. Selfish Nodes

Selfishness is a behavior that is used during own something in the process of protecting oneself. This selfishness enjoys every resource but never gives its own (Das et al., 2014). Selfishness in nodes is caused because of a shortage of resources like battery life, memory, CPU power,

⁹ https://www.researchgate.net/figure/Nodes-Misbehavior-in-MANETs-and-WSNs_fig1_48166324

network bandwidth, energy. This results in the nodes saving their resources by refuse on the participation of routing service which consumes their energy. SLFN uses other node resources to forward their packet for communicating but refuses to accept and forward the packet to other nodes (Susan et al., 2020). Selfishness can happen in static or dynamic ways. Nodes in static selfishness are unchanged in the whole process but the dynamic selfishness of nodes is changed in some way to be a normal node and sometimes to preserve its energy nodes will be selfish (Shan et al., 2021).

2.5.1. Characteristics of Selfish Nodes

SLFN behave in different ways to save their resources. SLFN does the following misbehaving acts in various ways which degrade the performance of the network.

- Intentionally delay RREQ packets.
- Refuse in routing processes (unable to detect the presence of node for packet forwarding).
- Do not send or reply Hello the message.
- Do not participate in the route discovery and maintenance process.
- Drop data packets.
- Depending on the level of energy they have sometimes participated in the forward packet process and sometimes refuse to participate in it (Susan et al., 2020) (Mohamed Musthafa et al., 2020).

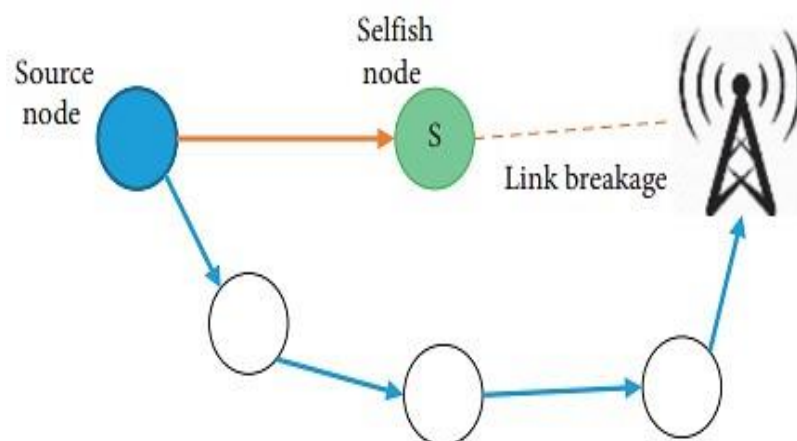


Figure 2.8 Effect of selfish node ¹⁰

¹⁰ https://www.researchgate.net/figure/Effect-of-a-selfish-node-in-routing_fig1_352001823

2.6. Algorithms to Solve Selfish Node Issues

The impact of SLFN brought a lot of challenges and problems in MANET particularly for routing protocol to not function properly including degradation on the performance. For this reason, SLFN should be identified, detected, and isolated for the sake of network performance. There are existing approaches that were developed to get rid of SLFN. Some of the techniques are explained below.

2.6.1. Reputation-Based Scheme

Reputation Based Scheme approaches to detect misbehaving nodes and applied to all nodes in the network. In these approaches, nodes exchange feedback during their communication to reward a Reputation when they participate and forward but, if the node acts selfishly and drops another node packet it will earn a bad reputation which helps to isolate the bad nodes. The nodes are evaluated by the Reputation value (on the feedback of nodes) based on the amount of forwarded and drop packets of the nodes. Nodes with a good Reputation consider cooperative nodes. This scheme uses two well-known models which are the watchdog approach and the Path rater (Mohamed Musthafa et al., 2020) (H. Yadav & Pati, 2018).

2.6.2. Credit-Based Scheme (“Serve and Earn”)

The approach is used to empower uncooperative nodes to be genuine nodes and participate in the routing process. When nodes forwarded a packet that comes from the neighbor node and when nodes request another neighbor node to forward a packet the SN rewards credit to the nodes for their service. But if their participation is poor they will receive low credit value (Samal et al., 2018) (H. Yadav & Pati, 2018).

2.6.3. Acknowledgment-Based Scheme

An acknowledgment is expected of nodes to identify if the node is regular or selfish. The receiver node (DN or can be an IN) should send an ACK message back to the originator node to confirm the received forwarded packet. If no ACK message is received by the SN from the DN it is assumed that the node is selfish and is detected (Samal et al., 2018).

2.6.4. Game-Theoretic Model

Game theory is a methodology that applies a mathematical model in a situation where players make a decision based on a different action to maximize players' payoff which affects the action of others. Game theory in other expressions is used for decision-making to select routes for transmission of the data from source to DN (Ilavendhan & Saruladha, 2018) (Premkumar et al., 2019). Game theory is a combination of players (any entity who can make a decision),

strategy (actions taken by players), outcome (result gained by players), pay-off (reward given for players based on the outcome (Vij et al., 2018).

Table 2-4: Selfish node identification algorithms

<i>Algorithm</i>	<i>Method</i>	<i>Advantage</i>	<i>Disadvantage</i>
Reputation-based (Watchdog and pathrater)	Monitors nearby nodes and identify selfish nodes. Each node obtain a specific reputation based on nodes “feedback”.	High throughput. Less end-to-end delay. High detection rate. Less channel traffic.	Can’t identify misbehaved nodes in the existence of collision. Low throughput. Low detection accuracy. High energy consumption.
Credit-based	Credit is given for packet forwarding for empowering the participating non-cooperative nodes.	Less channel traffics. High throughput. Less end-to-end delay.	High latency. Communication overhead.
Acknowledgment-based TWO-ACK (DSR), E-TwoACK (AODV)	Guarantee to forward a packet of a node through an acknowledgment message.	Less false-positive rate. High detection rate.	Increase delay and overhead due to rediscovering a route again. High traffic load.
Game theory	Finding the best strategy chosen for a better payoff.	Less end-to-end delay. High detection rate. Less channel traffic.	High false-positive rate. High end-to-end delay.

2.7. Related Works

A lot of research was done on MANET due to the high demand of experiencing this ad-hoc network and based on the advantage it provides for various application areas. One of the research areas in which researchers proposed solutions is link breakage on AODV in the presence of SLFN in MANET over the last years. Researchers mostly focus on detecting and punishing SLFN to prevent link failure related to our research area. Our related work is also describing some of the research related to SLFN identification, isolation and punishment.

In (Rama Abirami & Sumithra, 2018) research stated to overcome the problem of SLFN detection technique on the existed credit-based detection mechanism which represented those genuine nodes as malicious nodes were tried to eliminate and proposed a new Neighbor Credit Value-based AODV (NCV-AODV) routing algorithm to avoid false selfishness detection. In the proposed work nodes prepare a neighbor credit table (which helps all the nodes to track their awarded credit value), when nodes forward data packets automatically their neighbor credit table is updated and credit value will be increased, in which other neighbor nodes also assume those nodes as genuine nodes because of their increased credit value. To forward and to find the route nodes look next hop based on their credit value if a node needs to route to the next-hop first it checks the minimum credit value of next-hop. In the existing credit-based false detection technique (normal nodes detected as malicious nodes) which was a basic problem but the authors proposed a solution to differentiate the normal nodes (in case the normal nodes do not participate in the routing process by different reasons but if they remain normal nodes) each node sends a dummy data which helps to initiate the normal nodes to participate in the process to get credit value.

The authors, M. Education, M. Ponnusamy, and W. Bengal (Education et al., 2021) focus on Selfish Node Removal using Reputation Model (SNRRM). The proposed SNRRM is designed based on two basic concepts in which every node reputation calculation is done through the current energy levels of node and their communication ratio. The SN and DN fall under a communication range (calculated based on node request and reply messages). When communication begins from SN to DN and if they fall under the communication range, nodes check the reputation value of the SN, but if they do not fall under the communication range, the SN sends a control message to its neighbor node and wait for a reply. However, SLFN by their selfish behavior they are not sending a reply message easily, so the communication ratio of the nodes computed relay on request and received a reply the message.

In (Sharah et al., 2020) research on misbehaved node detection and punishment method is designed using a game-based approach. The research proposed a slave mode selfish dynamic punishment scheme using cooperative repeated game approaches. The scheme is proposed to avoid SLFN and stimulate the nodes to make them cooperative with other nodes. The overall view of the proposed work is designed by a game approach that considers their model as a coalition game. In the coalition, every node can join at first as a normal node which a coalition permits to become a member. As new nodes join, you have different neighboring nodes that would start observing and monitoring their willingness to share their resources.

When nodes refuse to share a resource the other neighboring nodes record in the misbehaved record table. This process is done with several iterations. If the nodes continue not to cooperate, the node mark as a SLFN, and punishment will be decided. For the selfish behaved nodes, the proposed punishment model uses nodes (neighbor node) who have a direct connection with a specific node can examine the degree of cooperation of that node which the neighbor node updates the misbehaving record table. In their punishment strategy when nodes stop cooperating, an alarm will be triggered all over the coalition, the triggered alarm enters the node to the punishment state. The strategy proposes four main steps. The first stage uses a slave mode that helps all nodes to forward and exchange data, the second is resource exhausting used to reuse the power of the punished node to bring to a cooperative state. The third punishment iteration stays under slave mode to ensure the rehabilitation of SLFN. The last one is the monitoring stage after the punishment of nodes coalition keeps watching SLFN before consider as a genuine node.

An Improved Token-based umpiring technique (TBUT) on incentive-based tolerant technique method (Susan et al., 2020) was proposed in this work for the isolation of selfish behavior nodes. TBUT mainly focuses on tokens for nodes that are given from their starting point that contain node ID, status, and reputation fields. Node ID differentiates the node uniquely on the network whereas status bit and reputation value gives either freedom of participation on the network activities or deny the node on the participation. At first status bit and reputation value are initialized to „0“ but when nodes need participation should announce the values (status and reputation). However, if the value of status bit and reputation goes to (1, -1) respectively it represents the routing protocol that prevents the node from participation in the network.

Regarding (Mubeen, 2018) research selfish isolation is performed using non-cooperative game theory approach. For uncooperative nodes perfect information game theory approach uses a

game tree. Game tree has payoff value for both normal and selfish nodes. The researcher analyze identification before forwarding information by two cases. First case, if both nodes are normal and from their payoff value there Nash equilibrium (2, 2) nodes are allowed to forward. In the second case, if the first is normal and the second is selfish with payoff value (1, 2) respectively which indicate Nash equilibrium is not met in which sources prevent itself to forward to the selfish node.

Table 2-5 Related work summary

<i>No</i>	<i>Title</i>	<i>Identified problem</i>	<i>Method used</i>	<i>Improvement</i>	<i>Gap</i>
1.	Detection of Selfish Node Through Reputation Model In Mobile Ad-hoc Network MANET (Education et al., 2021)	Link failure	Selfish Node Removal using Reputation Model (SNRRM). Current energy levels and communication ratio of nodes considered.	Better PDR. high average reputation ratio. Consume less energy.	No measurement is taken and decision is specified to isolate or punish after punishment.
2.	An Improved Token-Based Umpiring Technique for Detecting and Eliminating Selfish Node in Mobile Ad-hoc Networks (Susan et al., 2020)	Cooperative node unfairly loaded.	Incentive mechanism by the use of the token for each node.	Increase throughput. Decrease number of quarantine nodes.	Punishment mechanism is not instigated. No decision is made on quarantine nodes to participate in forwarding process.
3.	Selfish Dynamic Punishment Scheme: Misbehavior Detection in MANETs Using Cooperative Repeated Game (Sharah et al., 2020)	Affect performance and network functions	Slave mode punishment scheme using cooperative repeated game approaches.	System delay reduced. Number of SLFN reduced.	The result of this work evaluates in terms of delay but doesn't incorporate with other parameters.

4.	Evaluation of neighbor credit value based AODV routing algorithms for selfish node behavior detection (Rama Abirami & Sumithra, 2019)	False detection of genuine nodes	Improved Neighbor Credit Value-based AODV (iNCV-AODV)	Low maliciously dropped packet. Improve throughput and PDR. Less end-to-end delay and routing load.	No penalizing technique is applied.
5.	Isolating Selfish Nodes and Analyzing Performance of Ad-Hoc Network Using Perfect Information Game Theory (Mubeen, 2018)	Nodes individual decision without coalition	Non-cooperative game theory (Perfect information game theory)	Increase throughput Decrease end-to-end delay	Selfish nodes are identified by only payoff values. Selfish punishment technique is not applied.

2.8. Related Work Summary

Its presence, popularity, and being interesting area MANET catch researchers' intention to study and solve problems related to performance issues. However, there are a lot of factors that impact MANET to suffer regarding its performance. MANET nodes are mobile and able to move freely, but nodes change their behavior because of the dynamic nature of MANET. The behavior of nodes impacts the AODV routing protocol which results in link breakage. AODV assumes nodes are cooperative but energy, power constraints change them to be uncooperatively SLFN. SLFN drop routing control packets during the route discovery process. Drop control packet causes link breakage between neighboring nodes which affects the communication process. Most MANET SLFN related research consider selfish behaved nodes and proposed solutions through incentive, reputation, trust-based, or acknowledged based method which detect and isolate to reduce its impact on overall performance. The above-reviewed research papers also consider SLFN detection and removal but the main gap was most of the papers do not propose punishment/stimulation mechanisms for SLFN to give a chance to make them cooperative nodes. Therefore, based on the above gap of related work we have proposed an acknowledgment-based punishment algorithm integrated with the existing AODV protocol which penalizes the selfish node from the network services first and gives a chance to be a genuine node and earn a credit. Penalizing SLFN decreases the effect of link breakage and increases network performance at all.

CHAPTER THREE

3. PROPOSED SELFISH PUNISHMENT ALGORITHM

3.1. Overview of Proposed Approach

In the AODV routing protocol nodes are expected to be cooperative, but sometimes they refuse to be willing in the transmission process of (control messages and data packets) by ignoring partially or completely. To save their resources behavior makes them selfish nodes. The selfish behavior leads to a link breakage between nodes as well as between different networks which brings a huge impact on network performance. This research was studied to track and identify those SN and punish them to be part of the network with their cooperative nature by reusing their resources.

3.2. General Assumption

The proposed solution specifies the general assumptions in the process of detection and punishment of SLFNs:

Assumption 1: SLFNs are assumed as dynamic.

Assumption 2: SN and DN are considered normal nodes; they are free from Selfish behavior.

Assumption 3: Nodes residual energy is computed after the first RREP packet which sends to SN.

Assumption 4: Selfish Nodes do not drop Ack-packet.

3.3. General Architecture of Proposed Selfish Punishment Algorithm

In the AODV routing protocol, link breakage and network partitioning happen in the presence of SLFN. SLFN refuses in the participation of transmission of packets, drop data and ignore messages assigned directly to it. This behavior of nodes causes a lot of damage to the network. In AODV identifying SLFN reduces the security risk in the network performance. But isolating a SLFN is not always the right choice for vulnerability reduction in the MANET network. Mitigating and eliminating a SLFN from the network reduces the availability in the network and participation with other nodes. This problem motivated us to propose a selfish punishment algorithm called Acknowledgement based AODV (Ack-AODV) algorithm. The

comprehensive structure of the proposed Ack-AODV is implemented to punish those selfish misbehave nodes. The proposed algorithm is distributed mainly into two phases.

Phase 1: focus on SLFN identification and Phase 2: deal with SLFN punishment. In the first phase, SLFN is identified based on residual energy (derived from the difference between initial energy and current energy of the node) and earned credit value with their defined threshold value. In the proposed Ack-AODV at first, nodes are deployed as NN. But, from the nature of the network (mobility); residual energy is expected to vary. Collaboration with nodes and exchanging when nodes reply for every request, nodes earn a credit.

Phase 2: selfish punishment algorithm is activated after the identification phase is completed. In the second phase those identified SLFN fences into a block state by two characteristics, one is when their credit value snatch by normal IN, and second when their residual energy is getting decrease from the expected threshold. During the block, authorized cooperative IN attached a FLAG punishment table to the SLFN. Unless SLFN is willing to go out from the block state they are restricted from any service in the network, but after a while, authorized IN sends Ack-packet and forces the SLFN to participate in the Ack-reply process, once they start to participate their credit value will be incremented again. The following figure describes the comprehensive structure of the proposed solution.

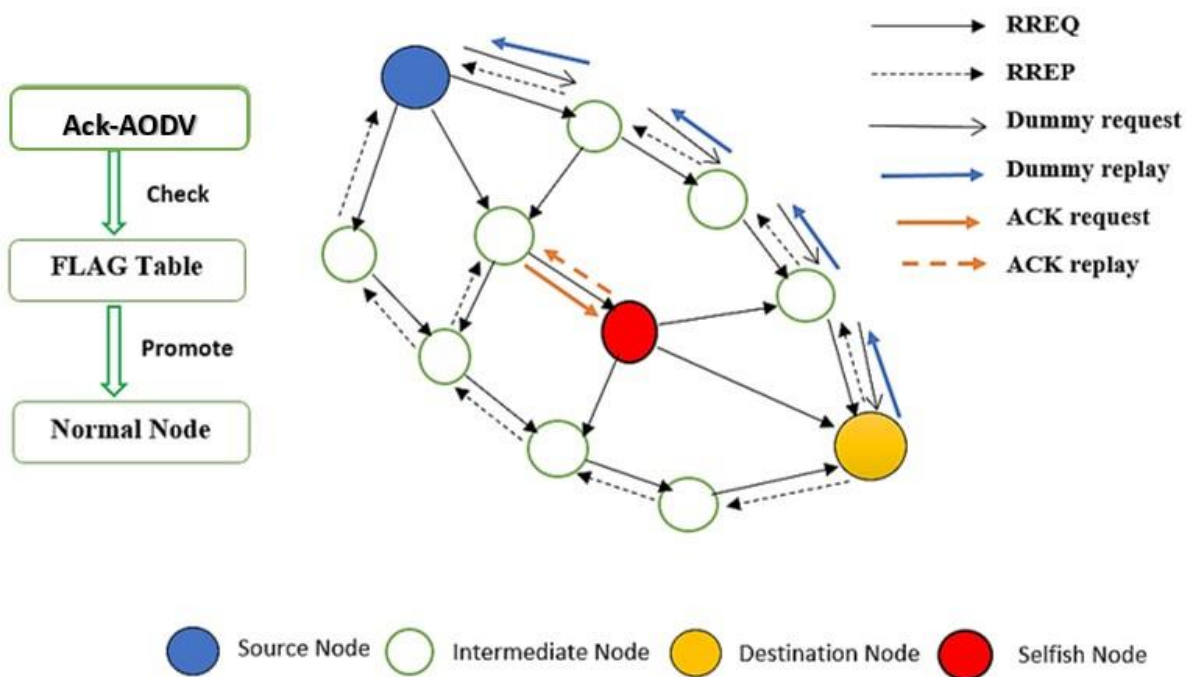


Figure 3.1 General architecture of proposed solution

3.4. Selfish Identification with Proposed Algorithm

This stage establishes a path through discovering a route. Route discovery in the proposed solution is the same as the original AODV protocol. At first, SN broadcasts the RREQ packet to all active nodes in the network to discover a route to the DN. The first RREQ packet is sent to INs in which they rebroadcast to other nodes on behalf of the SN until it reaches the targeted DN. When a route is available to the DN, the RREP packet is expected from those relay nodes and DNs to the source. If nodes acknowledge the request of the SN and send a reply packet based on a request it will earn a credit as a reward by the legitimate neighbor node. At the moment of route reply, the node's residual energy is measured and checked by the legitimate neighbor node. Measuring the residual energy helps to know if the nodes are participating in the transmission process. When nodes residual energy changes or varies it indicates that the nodes are replied to requests. But if the residual energy remains the same it indicates that the node tries to save its energy by refusing on request participation to stay in the network. After measuring node residual energy to make sure about the degree of being a NN or SLFN, a checkup or dummy packet is sent to make sure who participate in the request reply process. Selfish identification starts by checking the following characteristics:

- When SNs reply to requests their credit is incremented by 1. Their credit value is stored in their cache table. But when nodes refuse to reply and drop cooperative nodes' packets their credit value decrement by 1. In the outcome, if nodes credit is between 0 and negative value it is considered as SLFN and assigned -1 to indicate its selfishness.
- Whenever SNs move, their residual energy will fluctuate. In the proposed solution if node residual energy remains above 80% its identified as SLFN.

The generalized pseudo-code of the proposed selfish identification of Ack-AODV was described in algorithm 3.1.

Algorithm 1: Pseudo code of the proposed selfish identification

Step 1: Set up the network with a finite number of interconnected nodes

Step 2: Source S checks if routes exist from source S to destination D

Step 3: If (S finds routes to D) {
 S start sends a data packet to D
Else
 S broadcast RREQ to all neighboring nodes }

Step 4: D checks if the S request is correct

Step 5: D sends RREP to source S through intermediate node IN

Step 6: Credit ++

Step 7: Source S sends dummy packet to nodes who participate in RREP

Step 8: Reply to dummy request

Step 9: Credit ++

Step 10: Nodes check their neighbor node residual energy and credit value

Step 11: If Node.Residual Energy $\geq 80\%$ {
 Else (If Node.Credit ≤ 0 and Node.Credit \leq negative value {
 Node is Selfish [assigned (-1) selfish node];
 } }
}

Step 12: If Node.Residual Energy \leq threshold {
 Else If Node.Credit \geq threshold {
 Node is Normal;
 } }
}

Step 13: End

3.5. Selfish Punishment with Proposed Algorithm

Once the node admits as a selfish behaved node's a punishment scheme was proposed to stimulate and make the node supportive and cooperative with other normal nodes to work in the exchange data process. After identification of SLFN, the normal neighbor node enters the SLFN into a block state by preparing and tagging the punishment table called the FLAG table, which stays until the node changes its state to a normal node. FLAG table is described as follows.

Table 3-1 Punished selfish node FLAG table

SLFN_Address	Selfishness_level	Revoked_Service	Earned_Credit
--------------	-------------------	-----------------	---------------

SLFN_Address: contain the address of a SLFN.

Selfishness_Level: contain a negative value that represents the SLFN (value -1).

Revoked_Service: describe the service in which the SLFN can't participate. (Permitted on request acceptance and reply process but can't forward data packets or can't forward its data to another neighbor node).

Earned_Credit: describe the earned credit value of a SLFN. (SLFN has low credit value).

Algorithm 2: Pseudo-code of the proposed selfish punishment

Step 1:	Intermediate node IN prepare FLAG table
Step 2:	Intermediate Node IN send ACK-request to selfish node SN) {
Step 3:	if (selfish node SN reply to ACK-request) {
Step 4:	Then
Step 5:	Credit ++
Step 6:	Else
Step 7:	Credit --
Step 8:	If (selfish node SN.Credit >=1){
Step 9:	then
Step 10:	Node promoted normal node
Step 11:	End

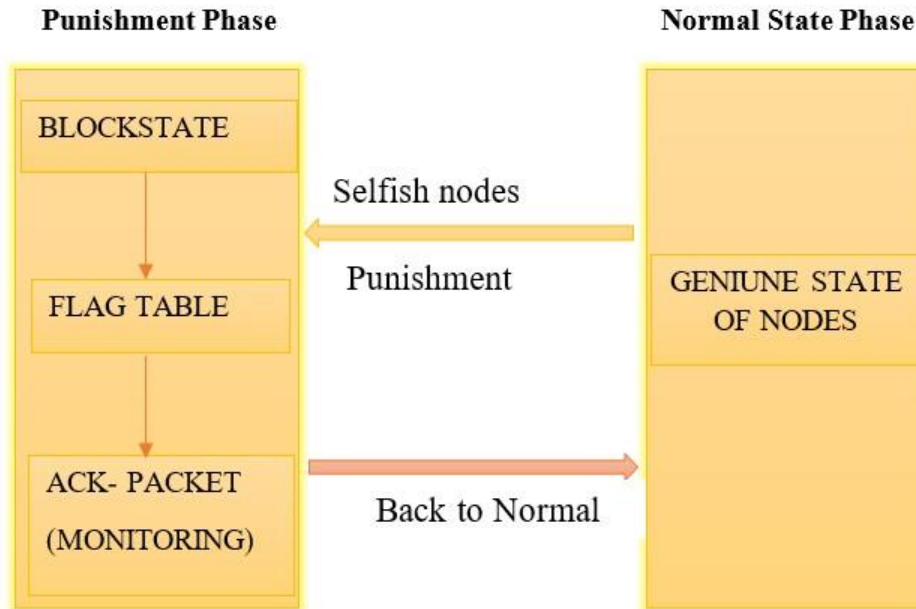


Figure 3.2 Proposed system model

3.6. Flow Chart Diagram for Proposed Ack-AODV Algorithm

This study identified SLFN before they impact the performance of the network and being a reason for link failure which punishes them to be genuine nodes under the AODV protocol. In this study during route establishment, if a route exists from source to destination, the SN can directly connect and send a data packet to the destination but if a source cannot find a destination path a route discovery is initiated by the SN through route request control packet to relay and DNs. A request-reply message will be sent to the source which tells about active nodes which helps to set a path between SN and DN.

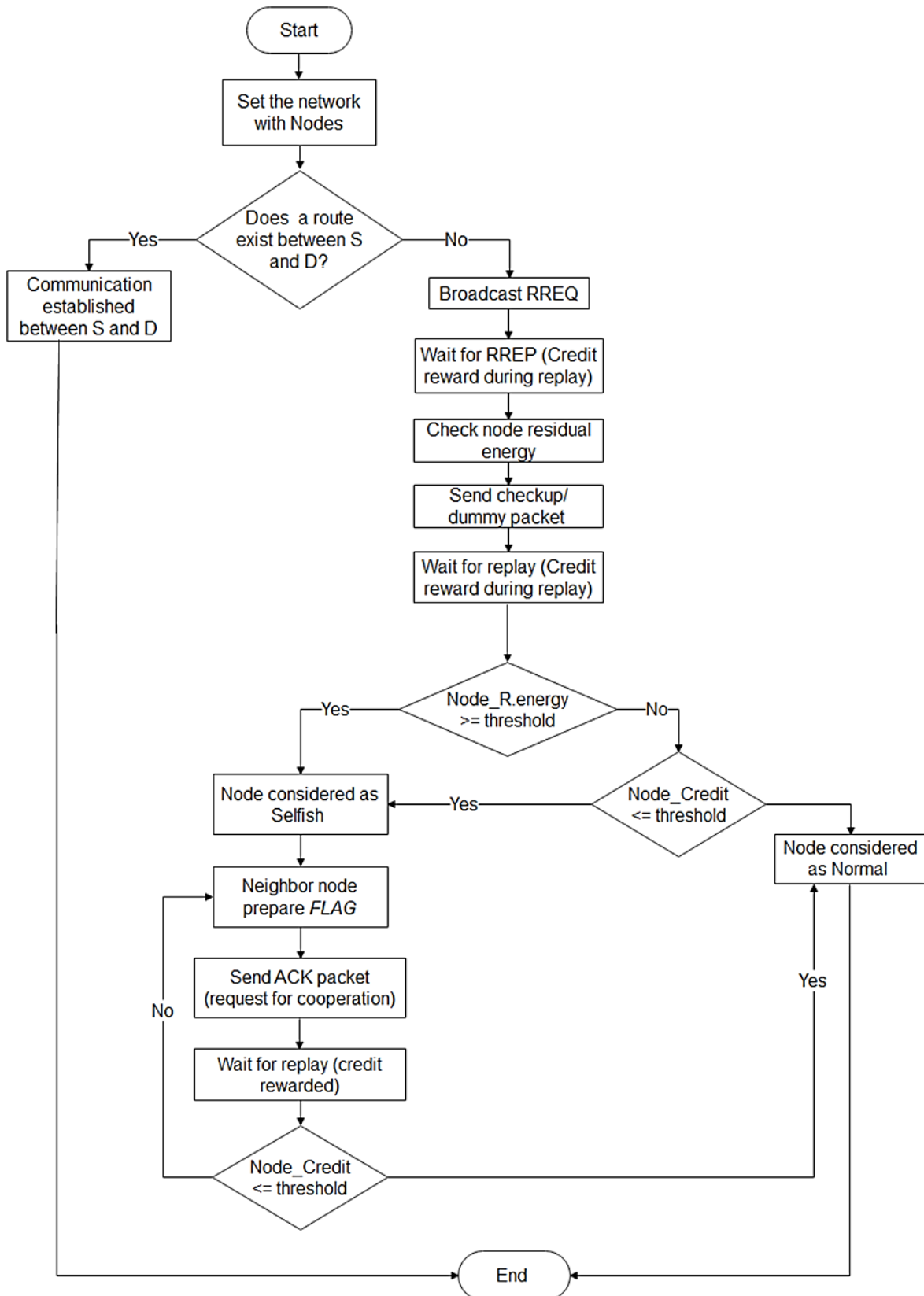


Figure 3.3 General transmission flow of Ack-AODV proposed algorithm

3.7. Summary

The new proposed Ack-AODV punishes the selfish behaved nodes to cooperate in order to minimize the probability of link breakage in AODV. The Ack-AODV algorithm applies during route discovery which detects the SLFN by residual energy and credit value, and the punishment strategy to promote the selfish node a FLAG table is tagged which block the node from the network services. The selfish behaved node is added to the AODV protocol to detect the selfish node. The code modification is made on the Ns2 network simulator on the existing AODV integrated with the new proposed algorithm. The proposed algorithm was added to the original code of AODV.

CHAPTER FOUR

4. SIMULATION AND PERFORMANCE EVALUATION

The simulation section describes the working environment for the proposed solution and performance comparison using the simulation tool based on AODV in MANET used to conduct the outcome of the simulation integrated to evaluation performance metrics.

4.1. MANET Simulation Tools

The simulation tool supported by a computer generated framework or environment to work on and to check various performance parameters. In the network environment, various simulation tools are presented based on the type of ad hoc network. In this section, MANET related general simulation tools were tried to identify and discussed for choosing a proper simulating tool for the proposed solution.

4.1.1. Network Simulator Version 3 (NS-3) Simulation Tool

NS3 is an open-source simulating tool, a new simulator that is not an extension of NS2 that is built using C++ and python to develop models (how the network, packet work and perform) and provide GUI for the user to perform simulation experiments to evaluate and test the network. NS3 is a discrete-event network simulator mainly designed for Internet systems targeted mostly for educational and research purposes. NS3 supports and focuses on Wi-Fi, WiMAX, and LTE physical layer simulations (Venkataramanan & Lakshmi, 2018).

NS3 provides different model types that permit users to define all of the various network components. Users must first create the network components such as nodes, devices that represent network actual part, channel, protocols, header, and packets to run the simulation.

Some advantage of NS-3 simulators is:

- NS-3 is flexible
- Provide emulation mode to integrate with real networks
- Provide realistic environment
- Faster simulation (L. et al., 2018).

NS-3 simulator is made up of four basic classifications stated in the figure below which is the application (user working environment like a generator), protocol stack (type of protocol used), the network device (hardware devices like interface cards), and channel (interconnection links).

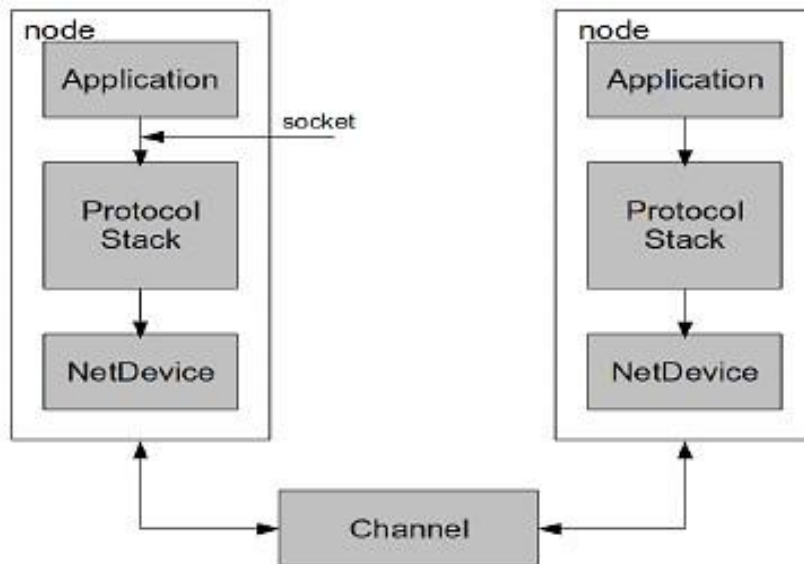


Figure 4.1 Overall architecture of NS-3

4.1.2. Network Simulator Version 2 (NS-2) Simulation Tool

NS-2 is a primary UNIX-based and simulated on wire and wireless layered network. It is a discrete event simulator same as NS3 which targeted a packet-level network for network research and teaching. NS2 supports and simulates a group of protocols such like TCP, UDP, FTP, and HTTP. NS2 is written with C++ and Otcl (Object-oriented Tool Command Language) script (scripting language defines the network (nodes, links, source destination, traffic type, and protocol type) (Journal, n.d.-b).

Some advantages of NS-2 are:

- Different available models
- Support TCP, multicast, and routing (wired and wireless)
- Provide easy traffic and pattern movement in an efficient energy model (Journal, n.d.-b)

The NS-2 architecture in the figure below describes components majorly used to run a simulation. Tcl simulation (a script that sets network topology, generates code, and shows the result), C++ handles the development of various protocols and simulation libraries and uses a NAM and X-Graph as a graphical visualizer for representing the result.

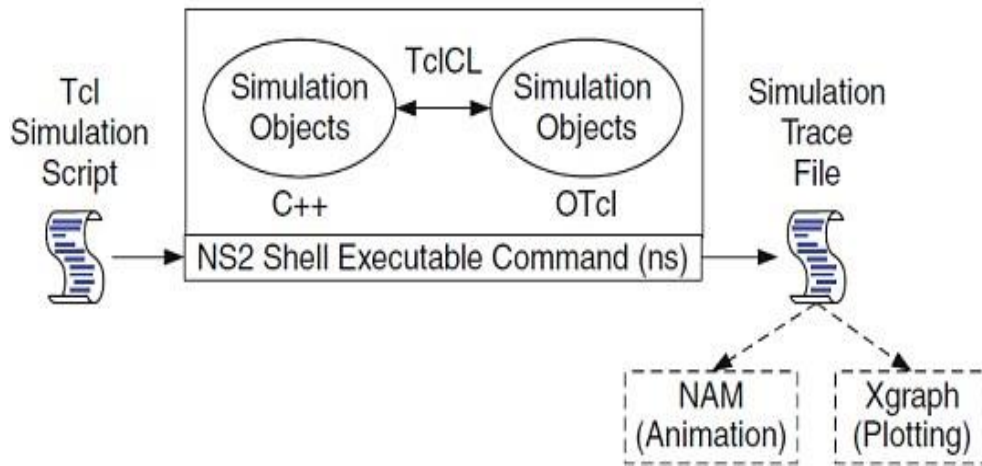


Figure 4.2 Overall architecture of NS-2

4.1.3. Comparison of Simulation Tools

Simulation tools are available in different types and versions based on the type of network, mobility models, routing, and language used. Being a popular simulation for the MANET network the following table shows a summary comparison between NS2 and NS-3 (L. et al., 2018).

Table 4-1 Comparison of NS-3 and NS-2

<i>Features</i>	<i>NS3</i>	<i>NS2</i>
Graphical visualizer	NetAnim	NAM
Language	C++, Python	C++, TCL
Type	Simulator + emulator	Simulator
Model	Must be ported	Set manually
Mobility models	Configuring	Configuring
License	Open source	Open source
Interface	Command-line	Command-line
OS	Window, Linux	Window, Linux
Scalability	Large-scale	Small-scale
Network layer protocol	DSR, AODV, DSDV	IPv4, IPv6, OSLR, AODV, DSR, DSDV

4.1.4. Objective Modular Network Testbed in C++ (OMNET ++)

OMNET ++ is created to build a wireless ad-hoc network simulation platform. It is a modular and component-based discrete event simulator framework that uses the C++ language to create the simulation component and provides the architecture for models to make bigger models with a high-level language called NED (Network Description). It is similar to NS2 and NS3 working on wired and wireless network simulation, but its widely used for a wireless network of sensors. OMNET ++ is based on components such as a simulation kernel which manages the class library of simulation, GUI for demonstration and debugging, and CLI for execution purposes. Advantage of OMNET ++:

- OMNET ++ is not limited to network protocol simulation also used for protocol and queuing modeling.
- Uses a GUI for executing and performing
- It is a modular model and uses IPv6

In OMNET ++ structure figure 4.3 describes the modeling concepts which is simple and compound module. An active simple module in OMNET ++ creates a compound module which is created by C++ language. Arrows in every simple module indicate gates or connections which used as an entry of module input and output interface (L. et al., 2018) (Journal, n.d.-b).

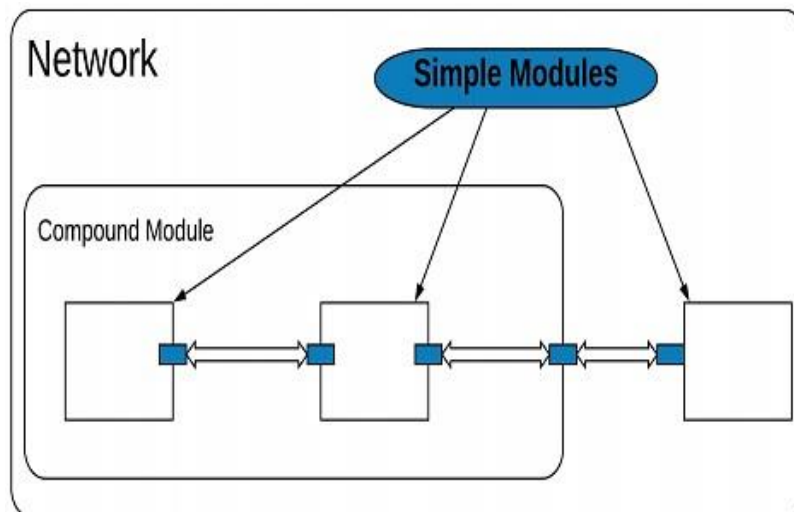


Figure 4.3 OMNET++ module structure

4.1.5. QualNet Simulation Tool

A commercial and ultra-high fidelity network simulation software that predicts wired and wireless network platforms and network device performance. QualNet can execute heterogeneous and large networks, for instance, can model thousands of nodes that support up to 500-20,000 nodes. C/C++ language is implemented in QualNet for implementing new protocols and to simulate different network behaviors under various user operating scenarios which run in a distributed machine (Journal, n.d.-b).

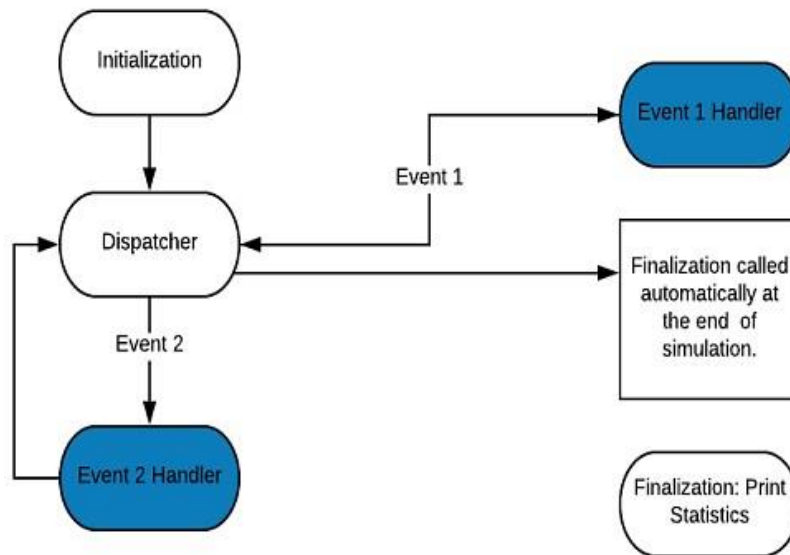


Figure 4.4 QualNet architecture

Table 4-2 Comparison of OMNET++ and QualNet

<i>Features</i>	<i>OMNET++</i>	<i>QualNet</i>
Graphical visualizer	GUI and CLI	GUI and CLI
Language	C++ and IDE	C++
Simulation	Wireless network of sensors	Wired and wireless
License	Open source	Commercial
Purpose	Academic and non-profit	Defense
Scalability	Large-scale	Large-scale

4.2. Simulation Setup

To demonstrate the proposed algorithm on the existing AODV routing protocol different simulation setup was used. To simulate the solution, the ns-2.35 simulator tool is used with a simulation area of 1000m x 1000m coverage was applied with a total number of 50 nodes. Ns-2.35 network simulator was chosen in because it is best simulation tool for MANET. Ns-2.35 is best for small scale network which is best to show clearly the effect of SLFN effect. In a large and very dense network SLFN effect doesn't highly impact the network at all. Nodes in the simulation configuration were designed to use a Random Waypoint (RWP) mobility model which is limited with 1000byte of packet size generated with 0.5s interval and paused with 50ms. In the simulation, nodes are assigned with 100.00w initial energy with CBR traffic source which generates constant packet through UDP connection rather than TCP connection in which SN stop connection if the acknowledgment packet is not received. The proposed algorithm was implemented on the existing module of "AODV.cc" and "AODV.h" file. Creating and analyzing mobile nodes in the simulation environment was done using a TCL script. The simulation was run in a network animator "/.NAM" file which shows how to request, reply packets are addressed and show dropped packets by nodes. Performance parameter and routing result is examined by a trace file ".tr" which shows the overall communication. The general simulation setup and parameter summary listed (Richhriya, 2020) are described in table 4-4 below.

Table 4-3 Simulation environment setup

<i>Simulation Setup</i>	<i>Parameter Values</i>
Simulator Software	Ns-2.35
Routing Protocol	AODV
Simulation Area	1000m * 1000m
Mobility Model	RWP
Number of Nodes	50
Simulation Time	50ms
Packet Size	1000bytes
Traffic Type	CBR
Connection Type	UDP

4.3. Performance Evaluation Metrics

To implement the proposed Ack-AODV algorithm different simulating setup was used. NS2 simulator contains an AODV routing protocol module which is the best fit for MANET than other network simulator tools. Determining the network if it is best or not based on performance metrics. The performance metrics were used to compare the simulation results.

The performance evaluation metrics were used to compare throughput, end-to-end delay, and packet drop rate of the existing AODV with the proposed Ack-AODV algorithm.

Throughput: describes the ratio of total reached packet to the destination. Throughput in MANET is one of the main performance evaluation parameters which measure the strength of the network that is expressed in *bits/bytes per sec.* mathematically expressed as follows:

$$Throughput = \frac{No. Received Byte}{Simulation time} \quad (4.1)$$

End-to-End Delay: defines the estimated time to transfer the packet from the source to the overall time a packet arrived at the D. Delay is defined in terms of time which is expressed in *sec.* that a packet takes to go across the network.

$$E2E Delay = \frac{1}{n} \sum_{i=1}^j (Received Pack. t - Sent Pack. t) \quad (4.2)$$

Packet Drop rate: explain the difference of packet drop before reach or received by the destination (Ahmed & Khalifa, 2020). Packet drop occurs when a packet failed to reach the destination when a network fails. Packet drop rate measured in the percent of packet lost to packet sent.

$$Packet Drop Rate = (No. Sent Pack. - No. of Received Pack.) * 100 \quad (4.3)$$

Table 4-5 describes the challenges which limit the performance evaluation metrics to perform better in a network. The factors affect the value gained from the metrics.

Table 4-4 Factors affect performance evaluation metrics

<i>Performance Metrics</i>	<i>Factors affect Performance metrics</i>
Throughput	<ul style="list-style-type: none"> ▪ Topology change ▪ Unreliable communication ▪ Limited bandwidth ▪ Energy ▪ Traffic
End-to-End Delay	<ul style="list-style-type: none"> ▪ Medium and large network (high dense of mobile nodes)
Packet Drop Rate	<ul style="list-style-type: none"> ▪ Network congestion

4.4. Simulation Result

The simulation result experiments through ns-2. It is most preferable for MANET network with C++ and oTcl based simulator which runs on both Linux and Windows platform. In this thesis work, 50 nodes are chosen to simulate in general with a simulation time of 50ms. Through 50 nodes there are 5 SLFN that drop packets to save resources. Based on table 4-4 simulation setup figure 4.5 simulation result was generated using NAM Window. From the simulation result observation, all nodes are normal nodes without SLFN added. In the scenario source node, intermediate node, and destination nodes are deployed. Regarding the simulation, all nodes appear and a random waypoint mobility model was applied. The process of node creation is explained in Appendix C.

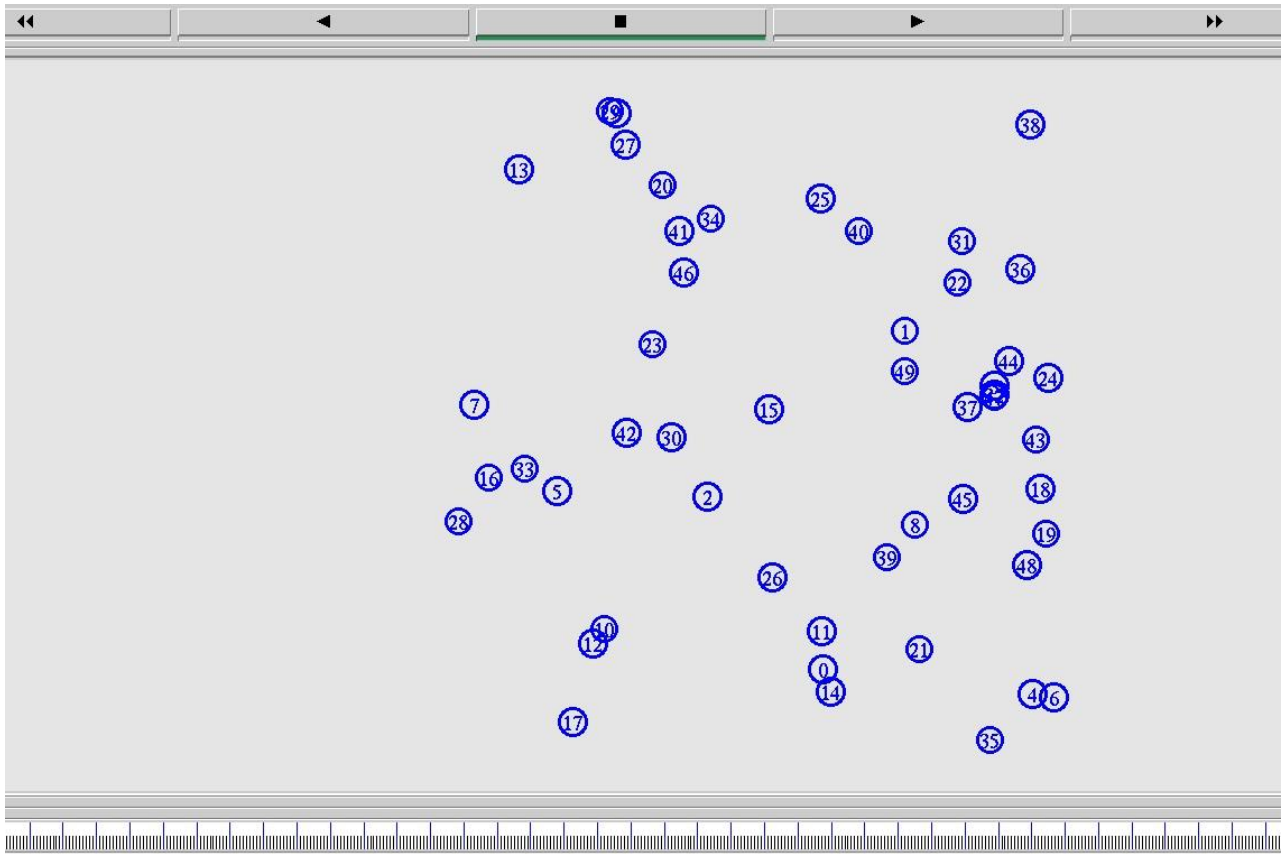


Figure 4.5 Node representation on AODV in MANET

In the figure 4.6 simulation scenario, normal nodes are configured as SN randomly. Out of 50 normal nodes, 5 nodes are SLFN that drop RREQ and RREP control packets. RREQ packets are dropped and are not reachable because of the existence of SLFN. The result in this scenario affects and decreases the rate of throughput and increases packet drop rate as well as end-to-end delay. The SLFN was added in the “selfish.tcl” script which is described in Appendix D.

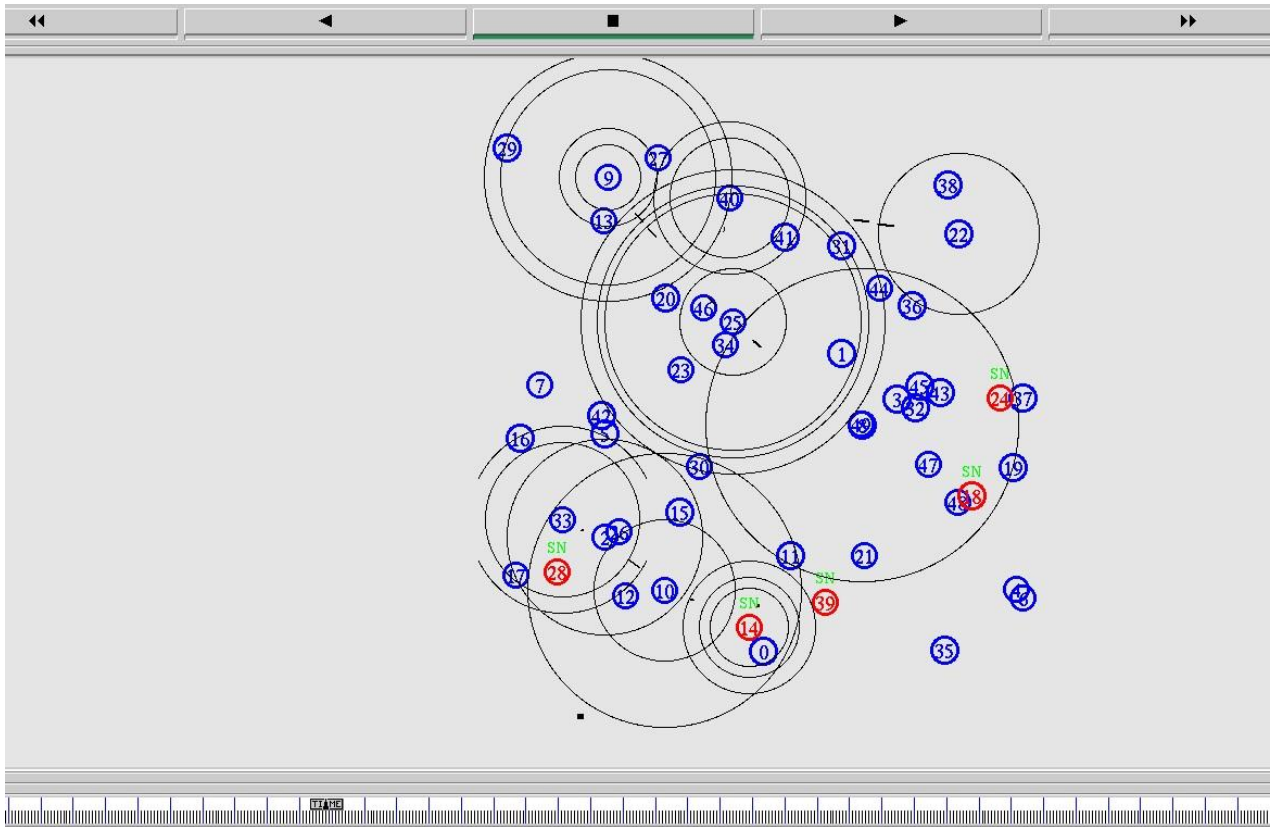


Figure 4.6 Selfish nodes on AODV in MANET

4.5. Analysis of Simulation Result

In this portion, a result was generated based on the trace file. Graphs were mapped by comparing the original AODV in the presence of SN with the proposed Ack-AODV algorithm. The effect of SLFN was examined and presented in the form of pictorial representation in terms of metrics such as throughput, end-to-end delay, and packet drop. The result in the existence of SN was aligned with the proposed solution result which shows clearly how the network throughput improved with decreasing packet drop rate, end-to-end delay.

4.5.1. Comparison Result in terms of Throughput

As figure 4.7 demonstrates how the presence of SLFN affects the network performance on the existing AODV protocol. In the simulation scenario route discovery between nodes starts at around 10ms and randomly SLFN start showing their behavior to affect the network after 15ms. When the number of SLFN increases parallel with the increase number of node in the network, it will affect throughput of the network. As the graph indicates when a selfish node occurs, the number of packets delivered to the destination is lower. Identifying and punishing

SLFN through the proposed Ack-AODV improved throughput of the network compared to the existing AODV.

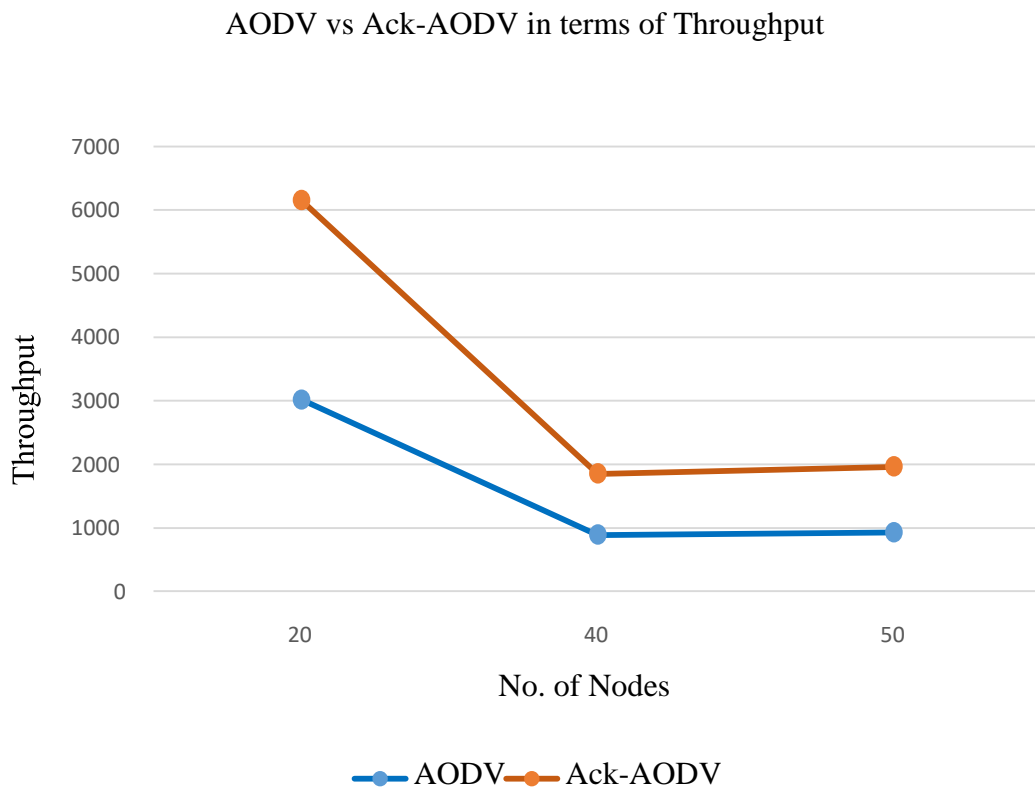


Figure 4.7 AODV vs Ack-AODV in terms of throughput

Figure 4.8 designates an end-to-end delay comparison between existing AODV in the presence of SLFN with the proposed Ack-AODV algorithm. SLFN impact the end-to-end delay of the network by delaying packets that were sent by the sender node. In the simulation design, SLFN is located in between of source and destination node which drops control packets. INs play a major role in packet forwarding but, in chapter 3 section 3.6 declared as an assumption the source and DN are free from selfish behavior. Receiving nodes were not acknowledged about the packet that was sent by the source in which intermediate nodes' refusal increase the delay ratio. In figure 4.8 effect of SLFN is a reason for an end-to-end delay to increase. Punishing SN in the proposed algorithm decreases the time that took the packet to arrive from source to destination according to generated result. An End-to-end delay was decreased by 23.3 % in the proposed solution.

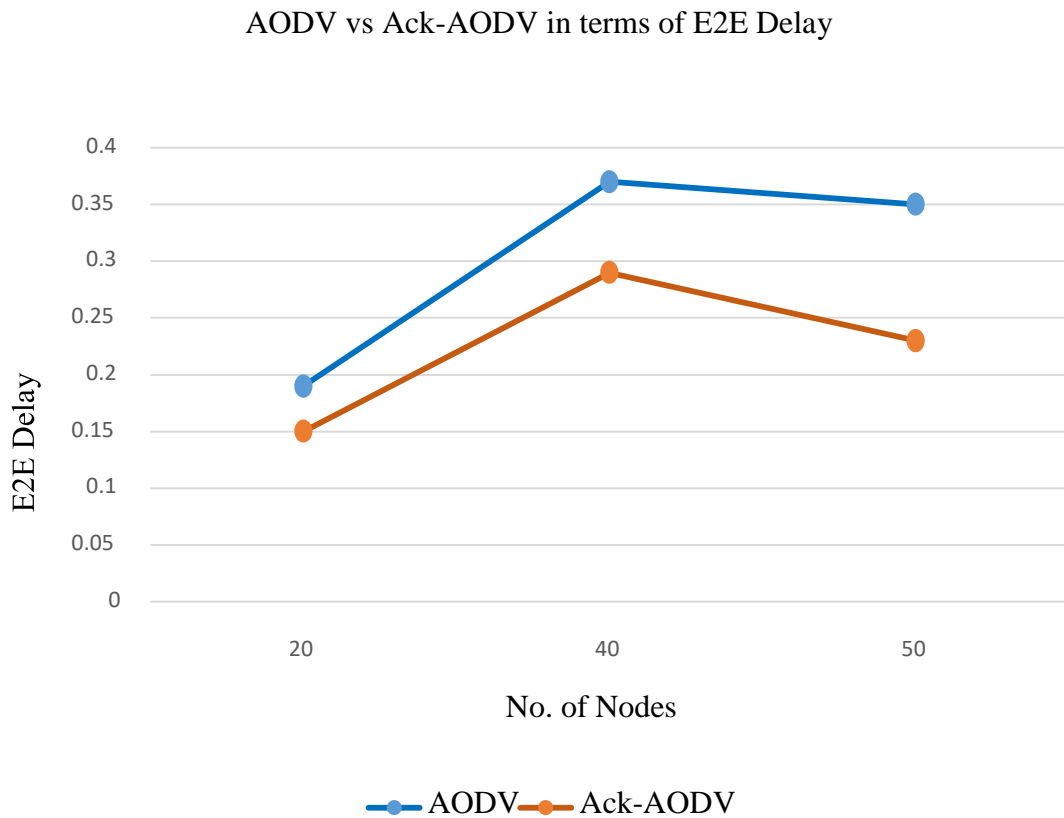


Figure 4.8 AODV vs Ack-AODV in terms of E2E delay

4.5.2. Comparison Result in terms of Packet Drop rate

As the Figure 4.9 shows the comparison of the dropped packet between AODV and Ack-AODV. The result is calculated and produced using AWK script from the trace file. As the graph shows when the number of SLFN increases more packets were dropped in the original AODV than the proposed Ack-AODV. This was due to Ack-AODV nodes being punished and takeoff their credit rather when they had individual benefits. The Ack-AODV proposed solution was more efficient in the case of changing SLFN to normal nodes. The dropping rate in the Ack-AODV decreased by 25.78%.

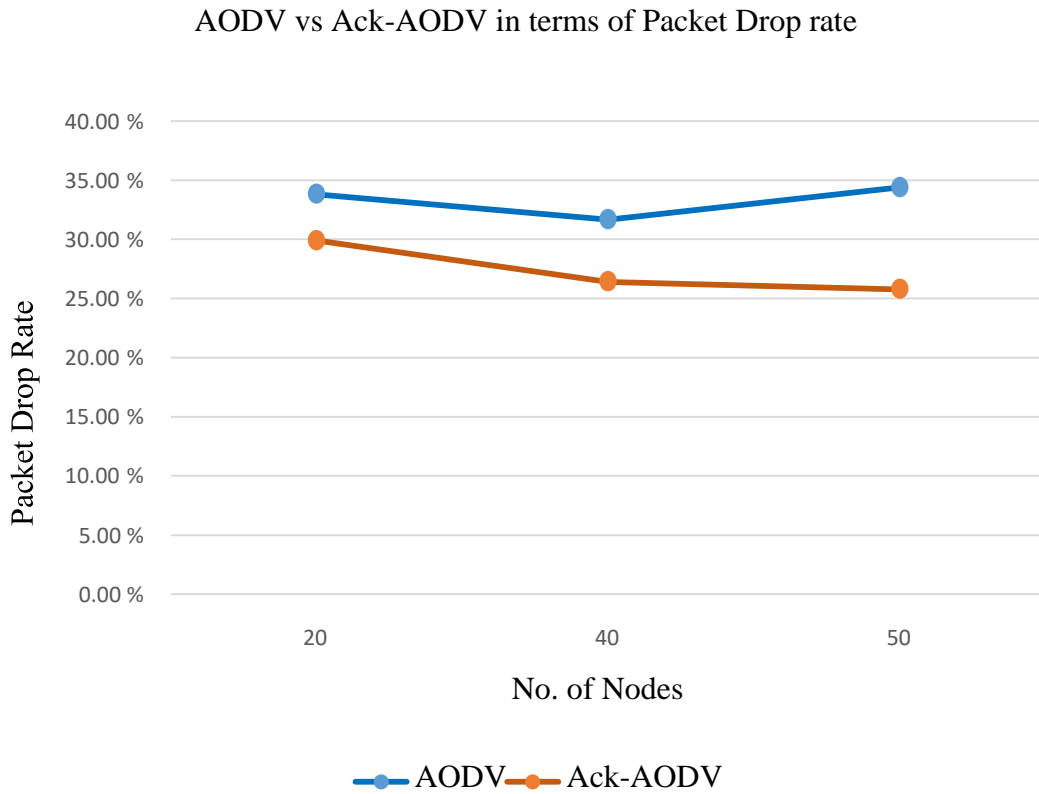


Figure 4.9 AODV vs Ack-AODV in terms of packet drop rate

4.6. Evaluation Result

Performance metrics evaluation of existing AODV in the presence of SLFN and the proposed Ack-AODV is presented individually in the above graphs. The existing AODV with Ack-AODV was aligned and compared detailed. The result was taken and represented from the trace file which is recorded and saved during the running simulation. Based on the result it concludes that the proposed Ack-AODV algorithm improves performance by increasing packet throughput, reducing end-to-end delay, and reducing packet drop rate compared to existing AODV. From table 4-6 the proposed Ack-AODV has improved the result in all listed performance metrics.

Table 4-5 AODV vs Ack-AODV

<i>Performance Metrics</i>	<i>AODV</i>	<i>Ack-AODV</i>
Throughput (Kbps)	927.06	1035.51
End-to-End Delay (sec)	0.35	0.23
Packet Drop rate	34.39 %	25.78 %

The percentage ratio between existing AODV and Ack-AODV was explained in the following table 4-7. As the table shows the proposed Ack-AODV shows a better improvement compared with existing selfish added AODV.

Table 4-6 AODV Vs Ack-AODV

<i>Performance Metrics</i>	<i>AODV (%)</i>	<i>Ack-AODV (%)</i>
Throughput	80.7%	86.3% (Improved)
End-to-end Delay	35.0%	23.3% (Reduced)
Packet Drop rate	34.39%	25.78% (Reduced)

Generally, figure 4.10 describes the summary of the above tables. The figure shows by what percent the existing AODV and the proposed ACK-AODV vary in the presence of selfish node and after the punishment of selfish node. From the figure representation, there is a big difference between existing AODV and proposed Ack-AODV in all performance metrics. As the figure explain when a selfish node appears in the communication the existing AODV routing performance was decreased. This problem was addressed and improved by our proposed Ack-AODV algorithm. The algorithm punishes the selfish node and triggering them to be a normal node through acknowledging by sending Ack-packet.

Packet throughput: in the existence of SN the existing AODV was 80.7 % and by 86.3 % packet throughput increased in Ack-AODV.

End-to-end Delay: the existing AODV of 35 % is reduced by 23.3% on Ack-AODV.

Packet Drop rate: reduced by 25.78% on Ack-AODV than existing AODV of 34.39 %.

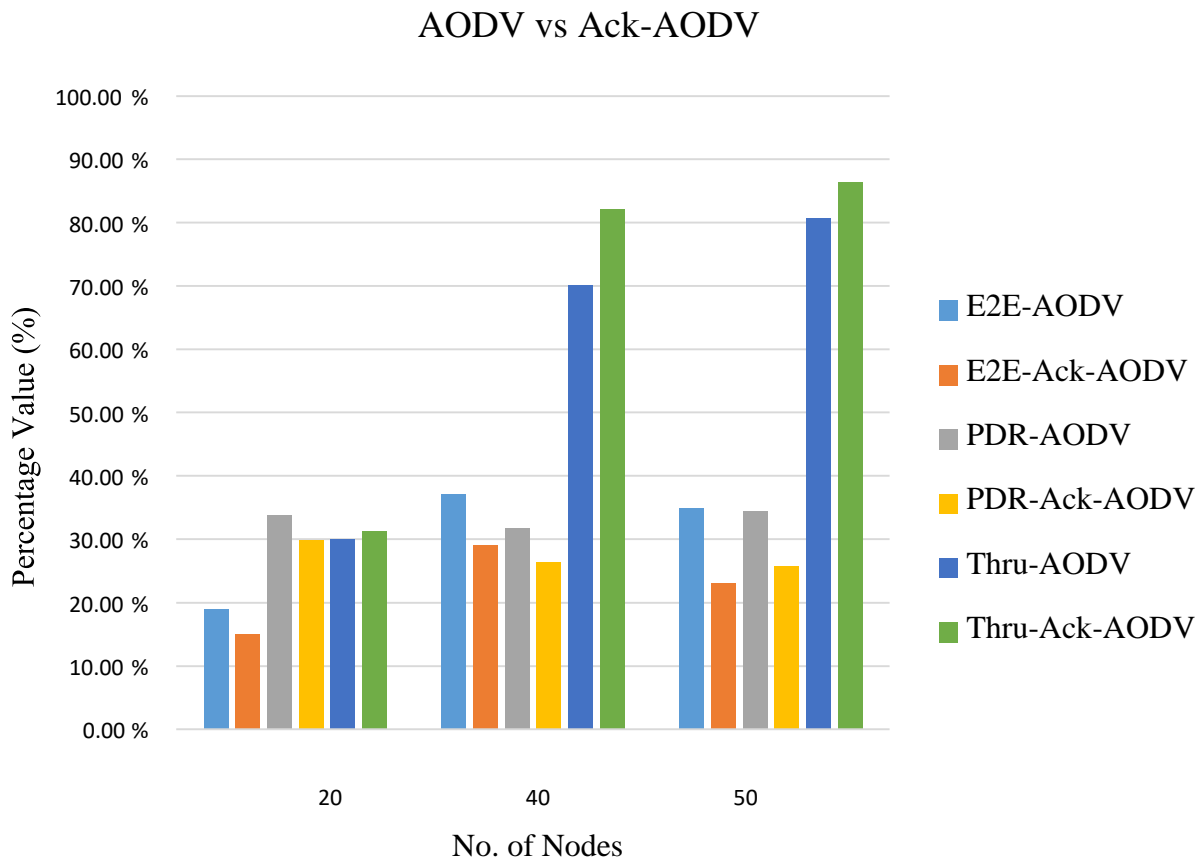


Figure 4.10 Comparison summary of AODV vs Ack-AODV

4.7. Discussion

The preliminary objective of this thesis was to create a mechanism to make nodes genuine by punishing them from using the service for individual benefit when they show selfish behavior. The punishment Ack-AODV method confirms it is the best method for reducing the SN effect and triggering them for categorizing under the genuine node. To get the evaluation result and for comparison purposes, the ns-2.35 network simulator especially for MANET is used to apply the Ack-AODV proposed algorithm. Initially, the performance metrics explained in table 4.4 were used to assess the performance of the network which gives a good improvement when Ack-AODV is implemented in the original AODV in the existence of selfish nodes.

Ultimately, considering the overall scenario, three stated questions in chapter 1 section 1.4 were answered based on the generated result from performance metrics inclusively. The questions were discussed and answered as follows.

RQ-1- How does Ack-AODV way of identifying improve the performance of AODV?

Performance of AODV can be affected by SLFN. SLFN degrade the network in all aspects. In this research to rid of SLFN an identification and punishment algorithm was proposed. Ack-AODV algorithm identified through node residual energy and credit value with checkup dummy packet. Nodes are label as a selfish if the residual energy and credit value decrease from the specified threshold. To ensure node selfishness a dummy packet sent which initiate to participate on the reply process. Once the SLFN identified a punishment method is proposed using FLAG table in which it blocks the nodes until it stop refuse. This propose solution improve the performance of existing AODV.

RQ-2- How to simulate the proposed algorithm using the NS-2 network simulator?

Chapter 4 section 4.4 description answers the second question. In the simulation scenario, around 50 nodes are selected in general. From the total number of nodes, 5 nodes are selected randomly. In chapter 3 section 3.3 the SLFN are identified by their credit value and residual energy threshold. When nodes are identified as SLFN if it has high residual energy and low rewarded credit. When a SLFN was identified by genuine neighbor node, the neighbor node builds a flag table and attaches to the selfish which blocks any route discovery and reply participation. To unblock the SLFN Ack-AODV algorithm designed to send an Ack packet for connection initiation. If the selfish reply, and to unbox from the blocked state until credit value increment reaches to the threshold.

RQ-3- How to evaluate and compare the newly proposed algorithm with a well-known existing AODV algorithm in terms of throughput, end-to-end delay, and packet drop rate?

The performance of the existing AODV with the proposed Ack-AODV has been evaluated in terms of throughput, E2E delay, and packet drop rate parameter values. Through the above parameter values, the comparison of the proposed Ack-AODV algorithm with the existing AODV was compared.

Even though the proposed Ack-AODV achieve a good result in terms of throughput, end-to-end delay and packet drop rate performance metrics, however using acknowledgment approach in SLFN punishment phase and applying dummy/checkup packet in SLFN identification phase brought a challenge for unexpected outcome in terms of packet overhead ratio. From the simulation result average packet overhead ratio had low improvement compare to the other performance evaluation metrics.

4.8. Summary

As discussed in the related work, the existing AODV performance is affected by different factors. SLFN behavior was considered as one factor. When nodes become selfish due to different reasons such as experiencing reduced battery power, their performance is reduced by dropping packets than forwarding. Due to SLFN effect, performance improvement method is proposed through Acknowledgement based (Ack-AODV) punishment algorithm. The algorithm punishes the SLFN by creating a flag table that stores their information to permit or block them from participating in the service. Once the SLFN is stated in the blocked state genuine neighbor node send an acknowledgment packet as a request for the nodes to be cooperative (genuine) by giving credit as a reward when they participate to reply. Generally, Ack-AODV benefit to punish the SLFN to decreases the negativity effect regarding performance and creates cooperation among nodes. Ack-AODV increases delivered packet than existing AODV. Packet drop rate and end-to-end delay were decreased in Ack-AODV. Overall, from the generated results, it is concluded that the proposed Ack-AODV algorithm accomplishes good results and performance compared with the existing AODV.

CONCLUSION, FUTURE WORK, AND CONTRIBUTION

Conclusion

In this research one major problem was addressed, punishing selfish behaved nodes. SLFN reduces the routing performance in AODV thus leading to having an increased packet drop rate, delay, and throughput reduction. To punish SN various literature was reviewed to find the solution. Since the goal of this study was to punish the selfish node with our proposed Ack-AODV algorithm. The proposed Ack-AODV punishes and helps SN to be a normal node. To indicate the improvement of Ack-AODV, simulations were tested in terms of performance metrics of throughput, end-to-end delay, and packet drop rate respectively with the ns-2.35 simulator.

The comparison result of our proposed Ack-AODV algorithm perform better and generate improved outcome compared to the existing AODV. To conclude, our proposed algorithm increases the major network performance parameter that of throughput even though the number of nodes increases. Ack-AODV also reduces the delay to deliver the packet to the destination and decreases the number of packets dropped by the selfish node.

Future Work

One of the fundamental findings of this research was punishing SLFN and making them normal nodes. In the future, we recommended another researcher who is interested to work on MANET specifically on selfish node punishment:

- We are recommended to work on studying the SLFN effect who drop route discovery and data packets at all and the punishment mechanism.
- In the proposed Ack-AODV intermediate nodes are considered SLFN. For further studies assume the DS as SN and analyze the impact in terms of performance metrics.
- We recommended studying to decreasing the average overhead ratio in a dense network.
- We recommended studying the effect of a selfish node in a dense AODV based MANET network and the result gained after punishment.

Contribution

The significance of this study was proposed to reduce node selfishness that obstructs the regular flow of communication and influence the network performance by blocking communication and breaking node links under the AODV protocol. Most papers included in chapter 2 section 2.7 in the related work described the selfish node identification and elimination method. There is a lot of mechanisms such as umpiring technique, reputation technique, or credit mechanism to identify and isolate from the network. When nodes change their status to selfish nodes, the normal neighbor node is initiated to block the selfish node for participation. To change the status of the node an acknowledge-based packet is sent to the selfish node and wait for the reply. The major contribution in this study was Punishing the selfish node and revoking it from any service. As a second contribution is changing the node status and triggering the selfish node to be a normal node.

SPECIAL ACKNOWLEDGMENT

This research work was funded by Adama Science and Technology University under grant number:

ASTU/SM-R/272/21

Adama, Ethiopia

REFERENCES

- Ahmed, D. E. M., & Khalifa, O. O. (2020). Performance Evaluation of AODV, OLSR, and GRP for Transmitting Video Conferencing over MANETs. *International Journal of Computer and Information Security (IJCSIS)*, 18(4), 45–50.
- Aifa, S., & Thomas, T. (2018). Review on Different Techniques used in Selfish Node Detection. *2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET 2018*, 1–4. <https://doi.org/10.1109/ICCSDET.2018.8821063>
- Al-Dhief, F. T., Sabri, N., Salim, M. S., Fouad, S., & Aljunid, S. A. (2018). MANET Routing Protocols Evaluation: AODV, DSR and DSDV Perspective. *MATEC Web of Conferences*, 150, 1–6. <https://doi.org/10.1051/mateconf/201815006024>
- Alkahtani, S. M., & Alturki, F. (2021). Performance Evaluation of Different Mobile Ad-hoc Network Routing Protocols in Difficult Situations. *International Journal of Advanced Computer Science and Applications*, 12(1), 158–167. <https://doi.org/10.14569/IJACSA.2021.0120119>
- Alo, R. U., Stanly, N. I., & Onwe, N. F. (2018). *Mobile Ad Hoc Network (MANET): Applications, Benefits and Performance Issues in a Global Positioning System*. 983–987.
- Alvarez Aldana, J. A., Maag, S., & Zaïdi, F. (2018). MANETs interoperability: Current trends and open research. *Proceedings - 32nd IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2018, 2018Janua*, 481–487. <https://doi.org/10.1109/WAINA.2018.00132>
- Appiah, M., & Cudjoe, R. (2018). A Comparative Study of Reactive and Proactive Routing Protocols on a Mobility Model in Mobile Ad Hoc Network (MANET). *2018 International Conference on Smart Computing and Electronic Enterprise, ICSC EE 2018*, 1–7. <https://doi.org/10.1109/ICSC EE.2018.8538398>
- Arnous, R., M., E.-S., & Saber, M. (2019). A Proposed Routing Protocol for Mobile Ad Hoc Networks. *International Journal of Computer Applications*, 178(41), 26–30. <https://doi.org/10.5120/ijca2019919305>
- Bugarcic, P. D., Malnar, M. Z., & Jevtic, N. J. (2018). Performance Analysis of MANET Networks Based on AODV Protocol in NS-3 Simulator. *2018 26th Telecommunications*

Forum, TELFOR 2018 - Proceedings, 1–4.
<https://doi.org/10.1109/TELFOR.2018.8612100>

Das, S. K., Saha, B. J., & Chatterjee, P. S. (2014). Selfish node detection and its behavior in WSN. *Journal of Engineering and Applied Sciences, 9*(6), 231–236.
<https://doi.org/10.3923/jeasci.2014.231.236>

Devi, M., & Gill, N. S. (2019). *Comparison analysis of MANET routing protocols to identify their suitability in smart environment. March.*

Education, M., Ponnusamy, M., & Bengal, W. (2021). *Detection of Selfish Nodes Through Reputation Model In Mobile Adhoc Network - - E Current* □ . *12*(9), 2404–2410.

Elleithy, A., & Loud, V. (2019). Selfish Nodes Mitigation in Mobile Ad-Hoc Networks. *2019 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2019, 1–5.*
<https://doi.org/10.1109/LISAT.2019.8817336>

Hamedani, M. M. (2018). *Mobility Models and Middleware Support for Mobile AdHoc Networks. 4*(3).

Ilavendhan, A., & Saruladha, K. (2018). Comparative study of game theoretic approaches to mitigate network layer attacks in VANETs. *ICT Express, 4*(1), 46–50.
<https://doi.org/10.1016/j.icte.2017.12.002>

Israr, A., Alvi, B., Aamir, M., & Rakocevic, V. (2017). Link Duration Analysis of Entity Mobility Models in the Network of Moving Objects. *Indian Journal of Science and Technology, 11*(47), 1–15. <https://doi.org/10.17485/ijst/2018/v11i47/137407>

John Justin Thangaraj, S., Rengarajan, A., & Selvanayagi, S. (2019). Comprehensive learning on characteristics, applications, issues and limitations of manets. *International Journal of Innovative Technology and Exploring Engineering, 8*(9 Special Issue 2), 311–314.
<https://doi.org/10.35940/ijitee.I1064.0789S219>

Journal, I. (n.d.-a). *Introduction to manet.*

Journal, I. (n.d.-b). *IRJET- STUDY OF VARIOUS NETWORK SIMULATORS.*

Kaur, G., & Thakur, P. (2019). Routing Protocols in MANET: An Overview. *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control*

Technologies, *ICICICT* 2019, 935–941.
<https://doi.org/10.1109/ICICICT46008.2019.8993294>

Khudayer, B. H., Anbar, M., Hanshi, S. M., & Wan, T. C. (2020). Efficient route discovery and link failure detection mechanisms for source routing protocol in mobile ad-hoc networks. *IEEE Access*, 8, 24019–24032.
<https://doi.org/10.1109/ACCESS.2020.2970279>

Kushwah, R., Tapaswi, S., & Kumar, A. (2019). A Detailed Study on Internet Connectivity Schemes for Mobile ad Hoc Network. In *Wireless Personal Communications* (Vol. 104, Issue 4). Springer US. <https://doi.org/10.1007/s11277-018-6093-7>

L., R., J., M., & J., A. (2018). Survey on Network Simulators. *International Journal of Computer Applications*, 182(21), 23–30. <https://doi.org/10.5120/ijca2018917974>

Mohamed Musthafa, M., Vanitha, K., Zubair Rahman, A. M. J. M. D., & Anitha, K. (2020). An efficient approach to identify selfish node in MANET. *2020 International Conference on Computer Communication and Informatics, ICCCI 2020*, 30–32.
<https://doi.org/10.1109/ICCCI48352.2020.9104076>

Mubeen, S. (2018). *Isolating Selfish Nodes and Analyzing Performance of Ad-Hoc Network Using Perfect Information Game Theory*. 6(December), 31–37.

Nagendranath, M. V. S. S., & Babu, A. R. (2020). An efficient mobility aware stable and secure clustering protocol for mobile ADHOC networks. *Peer-to-Peer Networking and Applications*, 13(4), 1185–1192. <https://doi.org/10.1007/s12083-019-00868-3>

Naik, K. C. K., Balaswamy, C., & Reddy, P. R. (2019). Performance Analysis of OLSR Protocol for MANETs under Realistic Mobility Model. *Proceedings of 2019 3rd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2019*, 1–5. <https://doi.org/10.1109/ICECCT.2019.8869406>

Premkumar, M., Chitra, M. P., Alisha Celestin, X., Kausalya, T., & Nandhini Priya, M. N. (2019). Game theory based ad-hoc on demand distance vector routing protocol to extend the wireless sensor networks life time. *Indonesian Journal of Electrical Engineering and Informatics*, 7(3), 463–471. <https://doi.org/10.11591/ijeei.v7i3.872>

Puri, S., & Arora, V. (2014). Routing Protocols in MANET: A Survey. *International Journal of Computer Applications*, 96(13), 7–12. <https://doi.org/10.5120/16852-6718>

- Ragab, A. R. (2020). A new classification for ad-hoc network. *International Journal of Interactive Mobile Technologies*, 14(14), 214–223. <https://doi.org/10.3991/IJIM.V14I14.14871>
- Raj, S. N. M., & Neeraja, S. (2019). Comparison of Routing Convention in Manet: a Survey. *International Journal of Hybrid Information Technology*, 12(1), 9–16. <https://doi.org/10.21742/ijhit.2019.12.1.02>
- Rama Abirami, K., & Sumithra, M. G. (2019). Evaluation of neighbor credit value based AODV routing algorithms for selfish node behavior detection. *Cluster Computing*, 22, 13307–13316. <https://doi.org/10.1007/s10586-018-1851-6>
- Rama Abirami, K., & Sumithra, M. G. (2018). Preventing the impact of selfish behavior under MANET using Neighbor Credit Value based AODV routing algorithm. *Sadhana - Academy Proceedings in Engineering Sciences*, 43(4). <https://doi.org/10.1007/s12046018-0803-4>
- Richhriya, V. (2020). *Performance Evaluation of Malicious node in Mobile Ad Hoc Network using AODV Protocol*. 3, 102–105.
- Samal, S., Ranjan Mishra, M., Pati, B., Rani Panigrahi, C., & Lal Sarkar, J. (2018). A Novel Approach to Avoid Selfish Nodes During Allocation of Data Items in MANETs. *Advances in Intelligent Systems and Computing*, 563, 571–580. https://doi.org/10.1007/978-981-10-6872-0_55
- Sen, J. (2010). *A Survey on Reputation and Trust-Based Systems for Wireless Communication Networks*.
- Shan, A., Fan, X., Wu, C., Zhang, X., & Fan, S. (2021). Quantitative study on the impact of energy consumption based dynamic selfishness in Manets. *Sensors (Switzerland)*, 21(3), 1–19. <https://doi.org/10.3390/s21030716>
- Sharah, A. Al, Alhaj, M., & Hassan, M. (2020). *Selfish Dynamic Punishment Scheme : Misbehavior Detection in MANETs Using Selfish Dynamic Punishment Scheme : Misbehavior Detection in MANETs Using Cooperative Repeated Game*. March.
- Sharma, M., Singh, M., Walia, K., & Kaur, K. (2019). A Comprehensive Study of Performance Parameters for MANET, VANET and FANET. *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON*

2019, 643–646. <https://doi.org/10.1109/IEMCON.2019.8936159>

Shrivastava, S. (2020). *Survey of Routing Protocols , Simulation , Testing Tools and Mobility Models in MANET*. 22(9), 70–77.

Susan, K., C., K., M., J.-O. A., & S., M. E. (2020). An Improved Token-Based Umpiring Technique for Detecting and Eliminating Selfish Nodes in Mobile Ad-hoc Networks. *Egyptian Computer Science Journal (ECS)*, 44(2), 74--85. <https://www.researchgate.net/publication/341597901%0A>

Tabbana, F. (2020). Performance Comparison and Analysis of Proactive, Reactive and Hybrid Routing Protocols for Wireless Sensor Networks. *International Journal of Wireless & Mobile Networks*, 12(4), 1–20. <https://doi.org/10.5121/ijwmn.2020.12401>

Talawar, M. B., & Ashoka, D. V. (2020). Link failure detection in manet: A survey. *Studies in Computational Intelligence*, 885, 169–182. https://doi.org/10.1007/978-3-030-384456_13

Ul Islam Khan, B., Olanrewaju, R. F., Anwar, F., Najeeb, A. R., & Yaacob, M. (2018). A survey on MANETs: Architecture, evolution, applications, security issues and solutions. *Indonesian Journal of Electrical Engineering and Computer Science*, 12(2), 832–842. <https://doi.org/10.11591/ijeecs.v12.i2.pp832-842>

Venkataramanan, V., & Lakshmi, S. (2018). A case study of various wireless network simulation tools. *International Journal of Communication Networks and Information Security*, 10(2), 389–396. <https://doi.org/10.13140/RG.2.2.32223.30885>

Vij, A., Sharma, V., & Nand, P. (2018). Selfish Node Detection using Game Theory in MANET. *Proceedings - IEEE 2018 International Conference on Advances in Computing, Communication Control and Networking, ICACCCN 2018*, 104–109. <https://doi.org/10.1109/ICACCCN.2018.8748632>

Yadav, H., & Pati, H. K. (2018). A Survey on Selfish Node Detection in MANET. *Proceedings - IEEE 2018 International Conference on Advances in Computing, Communication Control and Networking, ICACCCN 2018*, 217–221. <https://doi.org/10.1109/ICACCCN.2018.8748420>

Yadav, N., & Chug, U. (2019). Secure Routing in MANET: A Review. *Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel*

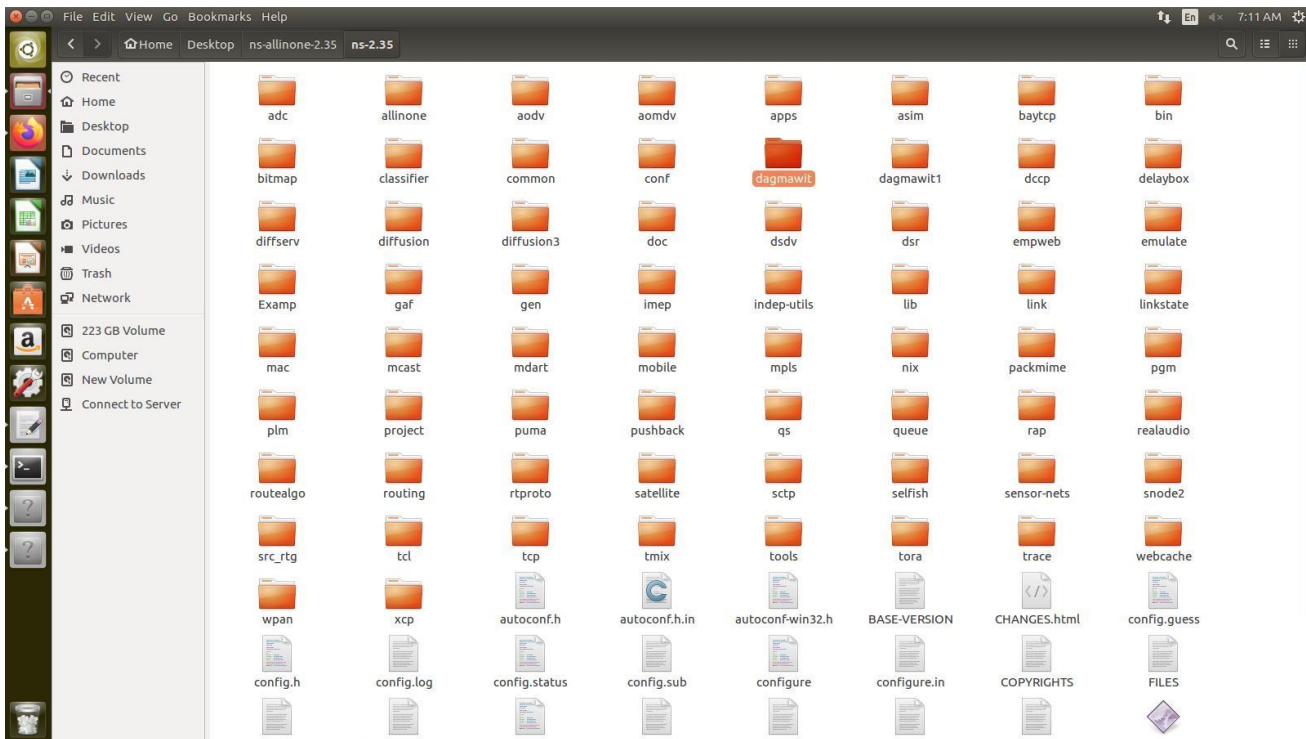
- Computing: Trends, Perspectives and Prospects, COMIT Con 2019*, 375–379.
<https://doi.org/10.1109/COMITCon.2019.8862238>
- Aifa, S., & Thomas, T. (2018). Review on Different Techniques used in Selfish Node Detection. *2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET 2018*, 1–4. <https://doi.org/10.1109/ICCSDET.2018.8821063>
- Deepak, S., & Anandakumar, H. (2019). AODV Route Discovery and Route Maintenance in MANETs. *2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019*, 1187–1191.
<https://doi.org/10.1109/ICACCS.2019.8728456>
- Devi, M., & Gill, N. S. (2019). *Comparison analysis of MANET routing protocols to identify their suitability in smart environment. March.*
- Richhriya, V. (2020). *Performance Evaluation of Malicious node in Mobile Ad Hoc Network using AODV Protocol. 3*, 102–105.
- Shan, A., Fan, X., & Zhang, X. (2020). Quantitative Study on Impact of Node Selfishness on Performance of MANETs. *Proceedings - 2020 IEEE International Conference on Smart Internet of Things, SmartIoT 2020*, 9–14.
<https://doi.org/10.1109/SmartIoT49966.2020.00011>
- Shrivastava, S. (2020). *Survey of Routing Protocols , Simulation , Testing Tools and Mobility Models in MANET. 22(9)*, 70–77.
- Ul Islam Khan, B., Olanrewaju, R. F., Anwar, F., Najeeb, A. R., & Yaacob, M. (2018). A survey on MANETs: Architecture, evolution, applications, security issues and solutions. *Indonesian Journal of Electrical Engineering and Computer Science, 12(2)*, 832–842.
<https://doi.org/10.11591/ijeecs.v12.i2.pp832-842>
- Yadav, H., & Pati, H. K. (2018). A Survey on Selfish Node Detection in MANET. *Proceedings - IEEE 2018 International Conference on Advances in Computing, Communication Control and Networking, ICACCCN 2018*, 217–221.
<https://doi.org/10.1109/ICACCCN.2018.8748420>

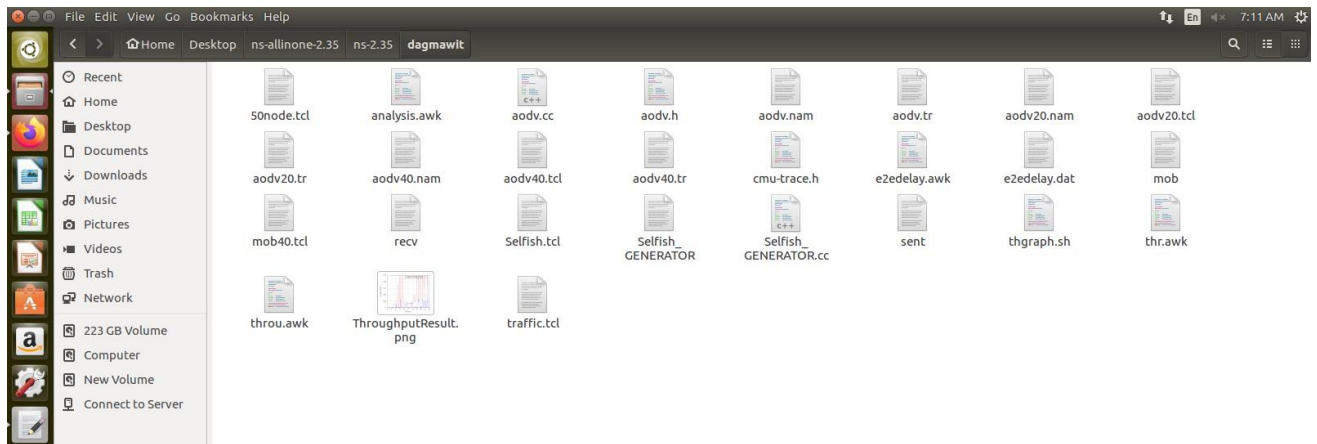
APPENDIXES

Appendix A: Terminal commands to run a file for the Ns-2.35 simulator

```
astu@astu-OptiPlex-3040: ~/Desktop/ns-allinone-2.35/ns-2.35/dagmawit
astu@astu-OptiPlex-3040:~$ cd Desktop/
astu@astu-OptiPlex-3040:~/Desktop$ ls
analysis.awk aodv.tcl automate.sh gnuplot_script ns-allinone-2.35 Output samples setdest
astu@astu-OptiPlex-3040:~/Desktop$ cd ns-allinone-2.35/
astu@astu-OptiPlex-3040:~/Desktop/ns-allinone-2.35$ ls
bin det80211nr-1.1.4 include INSTALL.WIN32 nam-1.15 otcl README share tclcl tk8.5.10 zlib-1.2.3
cweb gt-itm install lib ns-2.35 otcl-1.14 sgb tcl8.5.10 tclcl-1.20 xgraph-12.2
astu@astu-OptiPlex-3040:~/Desktop/ns-allinone-2.35$ cd ns-2.35/
astu@astu-OptiPlex-3040:~/Desktop/ns-allinone-2.35/ns-2.35$ ls
adc BASE-VERSION config.h dccp emulate INSTALL.WIN32 ncast packmime README sensor-nets tora
allinone baytcp config.log delaybox Examp lib mdart realaudio trace
aodv bin config.status diffserv FILES LICENSES mobile pin release_steps.txt src_rtg validate
aodmv bitmap config.sub diffusion gaf link mpl project routealgo tcl validate.out
apps CHANGES.html configure diffusion3 gen linkstate nix puma routing tcp VERSION
asin classifier configure.in doc HOWTO-CONTRIBUTE mac ns pushback rtproto test-all webcache
autoconf.h common COPYRIGHTS dsdv inep Makefile ns.1 qs pushback satellite tmx wpan
autoconf.h.in conf dagmawit dsr indep-utils Makefile.in ns_tclsh.cc sctp TODD.html xcp
autoconf-win32.h config_guess dagmawit empweb install-sh makefile.vc nstkl rap selfsh tools
astu@astu-OptiPlex-3040:~/Desktop/ns-allinone-2.35/ns-2.35$ cd dagmawit
astu@astu-OptiPlex-3040:~/Desktop/ns-allinone-2.35/ns-2.35/dagmawit$ ls
5node.tcl aodv2.tcl aodv4.tcl aodv.h cmu-trace.h mob Selfish_GENERATOR sent thru.awk
analysis.awk aodv2.tr aodv4.tr aodv.nam e2edelay.awk mob40.tcl Selfish_GENERATOR.cc thgraph.sh ThroughputResult.png
aodv2.nam aodv4.nam aodv.cc aodv.tr e2edelay.dat rcv Selfish.tcl thr.awk traffic.tcl
astu@astu-OptiPlex-3040:~/Desktop/ns-allinone-2.35/ns-2.35/dagmawit$ ns 5node.tcl
num_nodes is set 50
INITIALIZE THE LIST xlisthead
Starting Random WayPoint of nodes...
Starting Traffic...
Starting Selfish nodes...
Starting Simulation...
SORTING LISTS ...DONE!
channel.cc:sendUp - Calc highestAntennaZ_and dstCST_
highestAntennaZ = 1.5, dstCST = 406.3
END SIMULATION...
```

Appendix B: Ns-2.35 Simulation Tool Directory





Appendix C: Sample tcl file for network configuration and node setup

```

set val(canal) Channel/WirelessChannel; # Channel(1-11)
set val(propacacao) Propagation/TwoRayGround; # Radio Propagation
set val(antena) Antenna/OmniAntenna; # Aerial (omni/straight)
set val(layer2) LL; # Link Layer
set val(drop) Queue/DropTail/PriQueue; # Queue type
set val(fileSize) 50; # Queue size
set val(wlan0) Phy/WirelessPhy; # DSSS
set val(mac) Mac/802_11; # MAC Type
set val(routP) AODV; # Routing Protocol
set val(node) 50; # Node Number
set val(x) 1000; # Axis X
set val(y) 1000; # Axis Y
set val(trafego) 10; # Traffic Source Quantity
set val(TX) 1.2W; # Default NS2 - 0.400 -> 0.000509W/PKT
set val(RX) 0.6W; # Default NS2 - 0.300 -> 0.000156W/PKT
set val(IniEner) 100.00; # Initial Energy
set val(ModEner) EnergyModel; # Energy Model
set val(terminal) 50; # Simulation Time

$val(mac) set SlotTime_ 0.000020; # 20us
$val(mac) set SIFS_ 0.000010; # 10us
$val(mac) set DIFS_ 0.000050; # 50us
$val(mac) set CWMin_ 31; # Min Backoff [0, CW]
$val(mac) set CWMax_ 1023; # Max Backoff [0, CW]
$val(mac) set PreambleLength_ 144; # 144 bit
$val(mac) set PLCPHeaderLength_ 48; # 48 bits MAC_Address
$val(mac) set PLCPDataRate_ 1.0e6; # 1Mbps
$val(mac) set dataRate_ 11.0e6; # 11Mbps
$val(wlan0) set bandwidth_ 11.0e6; # Bandwidth
$val(mac) set basicRate_ 1.0e6; # 1Mbps
$val(wlan0) set freq_ 2.4e9; # 2.4 GHz 802.11b.
$val(wlan0) set Pt_ 3.3962527e-2; # Power TX.
$val(wlan0) set RXThresh_ 6.309573e-12; # RX Threshold.
$val(wlan0) set CSThresh_ 6.309573e-12; # Carrie Sense Threshold.
$val(wlan0) set RTSThreshold_ 3000; # Use RTS/CTS for packets larger 3000 bytes

# Begin Simulation
set ns_ [new Simulator]

# Trace File Writing
set selfishTrace [open aodv.tr w]
$ns_ trace-all $selfishTrace

# NAM File Writing
set selfishNam [open aodv.nam w]
$ns_ namtrace-all $selfishNam
$ns_ namtrace-all-wireless $selfishNam $

# Topology
set topology [new Topography]
$topology load_flatgrid $val(x) $val(y)

# "GOD (General Operations Director)"
set god_ [create-god $val(node)]

# Starting Channel 1
set chan_11_ [new $val(canal)]

```

```

# Node Setup
$ns_ node-config -adhocRouting $val(routP) \
    -llType $val(layer2) \
    -macType $val(mac) \
    -ifqType $val(drop) \
    -ifqLen $val(fileSize) \
    -antType $val(antena) \
    -propType $val(propacacao) \
    -phyType $val(wlan0) \
    -topoInstance $topology \
    -channel $chan_11_ \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace ON \
    -movementTrace ON \
    -energyModel $val(ModEner) \
    -initialEnergy $val(IniEner) \
    -txPower $val(TX) \
    -rxPower $val(RX) \
    -idlePower 0.45 \
    -sleepPower 0.05 \
    -wiredRouting OFF

# Wireless nodes
for {set i 0} {$i < $val(node)} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) color "blue"
    $ns_ at 0.0 "$node_($i) color blue"
    $node_($i) random-motion 0 ;#disable
}

puts "Starting Random WayPoint of nodes..."
source mob
puts "Starting Traffic..."

```

```

# NAM Position
for {set i 0} {$i < $val(node) } {incr i} {
  $ns_ initial_node_pos $node_($i) 40
}

# Stop nodes simulation
for {set i 0} {$i < $val(node) } {incr i} {
  $ns_ at $val(terminal).000 "$node_($i) reset";
}

proc stop {} {
  global ns_ selfishTrace selfishNam val geral
  $ns_ flush-trace
  close $selfishTrace
  close $selfishNam
  exec nam aadv.nam &
  exit 0
}

puts "Starting Simulation..."

$ns_ at $val(terminal).001 "$ns_ nam-end-wireless $val(terminal)"
$ns_ at $val(terminal).002 "puts \"END SIMULATION...\"; stop"
$ns_ at $val(terminal).003 "$ns_ halt"
$ns_ run

# nodes: 50, max conn: 12, send rate: 0.0078125, seed: 0.75 # 8 connecting to 9 at time 10.022502297546017
# 2 connecting to 3 at time 10.091066715303374
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 1000
$cbr_(0) set rate_ 128.0kb
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 1000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 05.091066715303374 "$cbr_(0) start"

# 4 connecting to 5 at time 10.057555911018307
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 1000
$cbr_(1) set rate_ 128.0kb
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 1000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 10.057555911018307 "$cbr_(1) start"

# 6 connecting to 7 at time 10.041585770501563
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 1000
$cbr_(2) set rate_ 128.0kb
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 1000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 10.041585770501563 "$cbr_(2) start"

set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 1000
$cbr_(3) set rate_ 128.0kb
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 1000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 05.022502297546017 "$cbr_(3) start"

# 10 connecting to 11 at time 10.036486962035525
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(10) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(11) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 1000
$cbr_(4) set rate_ 128.0kb
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 1000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 10.036486962035525 "$cbr_(4) start"

# 12 connecting to 13 at time 10.083444275932127
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(12) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(13) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 1000
$cbr_(5) set rate_ 128.0kb
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 1000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 10.083444275932127 "$cbr_(5) start"

```

Appendix D: Sample tcl file (selfish.tcl) Selfish Node Configuration

```
puts "Starting Selfish nodes..."
#####
$ns_ at 0.5 "$node_(28) color red"
$node_(28) color "red"
$ns_ at 0.5["$node_(28) set ragent_]selfishnode"
$ns_ at 0.5 "$node_(28) label \"SN\""

$ns_ at 0.5 "$node_(24) color red"
$node_(24) color "red"
$ns_ at 0.5["$node_(24) set ragent_]selfishnode"
$ns_ at 0.5 "$node_(24) label \"SN\""

$ns_ at 0.5 "$node_(39) color red"
$node_(39) color "red"
$ns_ at 0.5["$node_(39) set ragent_]selfishnode"
$ns_ at 0.5 "$node_(39) label \"SN\""

$ns_ at 0.5 "$node_(14) color red"
$node_(14) color "red"
$ns_ at 0.5["$node_(14) set ragent_]selfishnode"
$ns_ at 0.5 "$node_(14) label \"SN\""

$ns_ at 0.5 "$node_(18) color red"
$node_(18) color "red"
$ns_ at 0.5["$node_(18) set ragent_]selfishnode"
$ns_ at 0.5 "$node_(18) label \"SN\""
```

Appendix E: Sample Code for Selfish Node Assignment and Drop Packet

```
//start selfish node
if(strcmp(argv[1], "selfishnode") == 0)
{
    selfish = true;
    return TCL_OK;
}

selfish = false;
```

```
void
AODV::rt_resolve(Packet *p) {
    struct hdr_cmn *ch = HDR_CMN(p);
    struct hdr_ip *ih = HDR_IP(p);
    aadv_rt_entry *rt;

    if(selfish==true)
    {
        drop(p, DROP_SEL);
    }
}
```

Appendix F: Sample Trace file (aodv.tr) Nodes Communication

```
N -t 5.022618 -n 26 -e 97.739313
N -t 5.022618 -n 35 -e 97.739313
N -t 5.022618 -n 4 -e 97.739313
N -t 5.022618 -n 6 -e 97.739313
N -t 5.022618 -n 30 -e 97.739313
N -t 5.022618 -n 2 -e 97.739313
N -t 5.022619 -n 36 -e 97.739313
N -t 5.022619 -n 22 -e 97.739313
r 5.023465678 _45_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465701 _39_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465865 _49_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465866 _21_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465872 _47_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465893 _48_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465934 _18_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465954 _32_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465964 _11_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023465974 _3_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023466033 _19_ MAC 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
r 5.023466033 _37_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es
0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)
```

r 5.023466112 _1_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466166 _15_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466242 _24_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466253 _44_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466256 _0_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466277 _14_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466280 _26_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466301 _35_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466327 _4_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466421 _6_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466425 _30_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466499 _2_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023466529 _36_ MAC --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490678 _45_ RTR 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490701 _39_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490865 _49_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490866 _21_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490872 _47_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490893 _48_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490934 _18_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490954 _32_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490964 _11_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490974 _3_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023490999 _43_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491033 _19_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491033 _37_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491112 _1_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491166 _15_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491253 _44_ RTR 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491256 _0_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491277 _14_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491280 _26_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491301 _35_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491327 _4_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491421 _6_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491425 _30_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491499 _2_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491529 _36_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

r 5.023491594 _22_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [8:255 -1:255 30 0] [0x2 1 1 [9 0] [8 4]] (REQUEST)

s 5.023918944 _43_ RTR --- 0 AODV 48 [0 ffffffff 8 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [43:255 -1:255 29 0] [0x2 2 1 [9 0] [8 4]] (REQUEST)

s 5.024353944 _43_ MAC --- 0 AODV 106 [0 ffffffff 2b 800] [energy 97.739313 ei 2.260 es 0.000 et 0.000 er 0.001] ----- [43:255 -1:255 29 0] [0x2 2 1 [9 0] [8 4]] (REQUEST)

r 49.999350692 _38_ RTR --- 7782 cbr 1020 [13a 26 18 800] [energy 73.346526 ei 12.205 es 0.000 et 1.444 er 13.005] ----- [24:0 25:0 30 38] [636] 1 3

f 49.999350692 _38_ RTR --- 7782 cbr 1020 [13a 26 18 800] [energy 73.346526 ei 12.205 es 0.000 et 1.444 er 13.005] ----- [24:0 25:0 29 18] [636] 1 3

r 49.999640830 _24_ MAC --- 0 ACK 38 [0 18 0 0] [energy 73.037037 ei 12.216 es 0.000 et 1.148 er 14.622] ----- [1:255 2:255 30 32] [0x4 2 [3 86] 4294967260.000000] (REPLY)

D 50.000000000 _1_ IFQ END 0 AODV 44 [0 0 1 800] [energy 73.126955 ei 11.103 es 0.000 et 1.722 er 12.424] ----- [2:255 40:255 30 32] [0x4 4 [7 72] 9.000000] (REPLY)

D 50.000000000 _2_ IFQ END 0 AODV 44 [0 0 2 800] [energy 72.964621 ei 10.130 es 0.000 et 0.824 er 16.081] ----- [3:255 40:255 30 17] [0x4 1 [3 116] 10.000000] (REPLY)

D 50.000000000 _3_ IFQ END 0 AODV 44 [0 0 3 800] [energy 72.964621 ei 10.130 es 0.000 et 0.824 er 16.081] ----- [3:255 2:255 30 26] [0x4 1 [3 114] 10.000000] (REPLY)

D 50.000000000 _3_ IFQ END 0 AODV 44 [0 0 3 800] [energy 72.964621 ei 10.130 es 0.000 et 0.824 er 16.081] ----- [3:255 2:255 30 42] [0x4 1 [3 68] 10.000000] (REPLY)

D 50.000000000 _3_ IFQ END 0 AODV 44 [0 0 3 800] [energy 72.964621 ei 10.130 es 0.000 et 0.824 er 16.081] ----- [3:255 13:255 30 12] [0x4 3 [25 50] 9.000000] (REPLY)

D 50.000000000 _3_ IFQ END 0 AODV 44 [0 0 3 800] [energy 72.964621 ei 10.130 es 0.000 et 0.824 er 16.081] ----- [3:255 2:255 30 39] [0x4 1 [3 4] 10.000000] (REPLY)

D 50.000000000 _4_ IFQ END 0 AODV 44 [0 0 4 800] [energy 73.853051 ei 14.101 es 0.000 et 1.695 er 10.351] ----- [4:255 41:255 30 41] [0x4 6 [5 70] 9.000000] (REPLY)

D 50.000000000 _5_ IFQ END 0 AODV 44 [13a 5 5 800] [energy 72.909763 ei 10.046 es 0.000 et 0.878 er 16.167] ----- [26:255 24:255 29 24] [0x4 3 [25 126] 9.000000] (REPLY)

D 50.000000000 _5_ IFQ END 0 AODV 44 [0 0 5 800] [energy 72.909763 ei 10.046 es 0.000 et 0.878 er 16.167] ----- [5:255 4:255 30 47] [0x4 1 [5 66] 10.000000] (REPLY)

D 50.000000000 _5_ IFQ END 0 AODV 44 [0 0 5 800] [energy 72.909763 ei 10.046 es 0.000 et 0.878 er 16.167] ----- [5:255 4:255 30 44] [0x4 1 [5 44] 10.000000] (REPLY)