An Improved Token-Based Umpiring Technique for Detecting and Eliminating Selfish Nodes in Mobile Ad-hoc Networks

Konyeha Susan[1], Konyeha C.[2], John-Otumu A. M.[3] and Mughele E. S.[4]

[1].Department of Computer Science, University of Benin, Benin City, Nigeria

[2].Department of Electrical/Electronic Engineering, Benson Idahosa University, Benin City, Nigeria

[3].Department of Information Technology, Federal University of Technology, Owerri, Nigeria

[4].Department of Computer Science, Delta State School of Marine Technology, Burutu, Nigeria

Email: susan.konyeha@uniben.edu, ckonyeha@biu.edu.ng, adetokunbo.johnotumu@futo.edu.ng, prettysophy99@gmail.com

Abstract

Mobile Ad hoc network (MANET) performance, is dependent on the participation of network nodes to forward packets efficiently due to its decentralized control architecture. A node may refuse to forward packets from neighbouring nodes to conserve its resources while still using the network to transmit its traffic. This selfish behaviour, if exhibited by many nodes, degrades network performance and cooperating nodes become unfairly loaded. Existing detection and quarantining mechanisms have ensured that selfish nodes do not receive network services, thus penalizing selfish nodes. However, if many nodes become selfish, network communication itself becomes impossible. In this work, we introduce a hybrid incentive with tolerance mechanism to the token-based umpiring technique (TBUT), an existing selfish node detection and quarantining technique, to discourage selfishness amongst nodes, and reduce the impact of quarantining selfish nodes on network performance. By introducing an S_count field to the token field of each node, for tolerance, we reduced the number of quarantined nodes during the simulation time. This helped to mitigate the impact of selfish node quarantining which is implemented by placing a selfish node in a block list. The simulation results show that our method enhances TBUT framework performance. The developed model is robust and reliable, also considering its benefits, and the problem it solves, we consider it to be a more efficient and secure strategy for MANET development.

Keywords: *MANET, packet forwarding, routing, collaborate, selfish node, misbehaviour, token-based, umpire, network performance.*

1. Introduction

A mobile ad hoc network (MANET) is an infrastructure-less, dynamic topology network, consisting of a collection of mobile nodes that communicate with each other without the use of centralized control. These wireless nodes move about arbitrarily and can act as a source, destination, or intermediate routers to do both data transmission and reception in a network. The mobility of nodes causes the network topology to be unpredictable and to change frequently. The absence of a centralized authority introduces the need for nodes to cooperate and self-organize to form a working communication network.

Each mobile node in a MANET consists of a transceiver which uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to achieve efficient communication among nodes. Communication among nodes may be established directly or indirectly, a source node will have direct communication with the destination node that is within radio range. Nodes which are not within radio range will need the cooperation of other nodes in the network, to forward their packets using multi-hop communication. This results in indirect communication amongst nodes [1].

The flexibility of MANETs makes it attractive for emergencies such as military deployment in a hostile environment and search-and-rescue operations. It is also used in areas where a communication network is required, but there is none available, or the existing infrastructure has been destroyed by a disaster or a war.

Packet routing is a very important function in MANETs and requires that for any pair of non-adjacent nodes, the intermediate nodes forward data packets. It is therefore assumed by opportunistic routing algorithms that every node will forward each packet it receives promptly, but this is not always the case as some nodes may make use of other node's resources for communicating and refuse to forward packets from other nodes within their radio range. These nodes are termed as selfish or misbehaving [2].

The most common cause of selfishness in nodes is the shortage of resources including battery life, memory, network bandwidth and processor power. This results in the node saving its resources while disregarding the requests to forward packets to other nodes in the network. Furthermore, a node could become selfish to stop malicious data packets from untrusted nodes, hence a node may be selfish for node A and unselfish for node B [2]. However, the misbehaviour of nodes for whatever reason can significantly diminish the performance of MANETs [3].

The performance of a MANET is dependent on the collaboration of all nodes in the network for route discovery, route maintenance and to forward packets for each other. The selfish behaviour of nodes, if exhibited by many nodes, degrades network performance and cooperating nodes become unfairly loaded. The detection and quarantining mechanisms proposed in literature depend on promiscuous monitoring methods [3] and have succeeded in ensuring that on detecting selfish nodes, they are isolated from using any network services, thus penalizing the selfish node. However, simply detecting and quarantining selfish nodes may not be enough to improve the performance of a mobile ad-hoc network. Incentives can be introduced to reward cooperating nodes [4].

According to Tamilarasi and Selvam [5], some of the characteristics of selfish nodes, are as follows; selfish nodes may engage in route detection, but not take part in the packet routing process, selfish nodes may not be detected by the other nodes when required, selfish nodes do not reply or send hello messages, selfish nodes exempt themselves from participating in routing, or they may drop the data packets while relaying it to the next hop such that degradation in the performance of the routing path is observed by the source and destination, but they are unable to detect the problematic link. Selfish nodes tend to drop data packets of others to save their energy so that they might continue to transmit their packets, and hence reduce the latency of their packets [6].

Selvan and Selvakumar [7] consider the issue of security to be a critical problem when implementing mobile ad hoc networks (MANETs). They introduced an intrusion detection system to the traditional method of firewall and encryption, to protect the network since the intrusion detection system can be configured to monitor nodes and identify misbehaviour among them.

Section 1 provides an insight to mobile ad hoc networks (Manets), its benefits, and the problems caused by selfish nodes. Section 2, describes different strategies to detect the selfish nodes, as well as the strengths and weaknesses of the algorithms to address packet dropping, by the selfish nodes. In Section 3, the architecture of proposed system, proposed algorithm to discover the selfish node and address packet dropping, and incentive mechanism to prevent selfishness while reducing the effect of quarantining, is described. In Section 4, the performance testing of the proposed algorithm is done using the NS2simulator software, and evaluated with the TBU algoriths. We also discuss the results

2. Related Work

Several techniques for selfish node detection have been proposed to address the selfish behaviour in nodes from the network perspective, these include the techniques that address packet dropping or refusal to forward packets. The techniques are classified into three categories as stated in [6]. These are:

Reputation-based systems: Each node monitors the other nodes, observes their behaviour and uses the acquired information for routing.

Credit-based systems: Each node gives credit to other nodes as a reward for forwarding data.

Acknowledgement-based systems: Each destination node sends an acknowledgement to prove that it received the data packet forwarded by the source.

A summary of some of the related works reviewed is presented in Table 1

Table 1: Summary of related works

Author	Method	Strength	Weakness
Enrique et al. [8]	Proposed a collaborative watchdog approach (CoCoWa), which is based on the fast diffusion of selfish nodes awareness	This approach reduces the time and increases the precision of selfish node detection	This technique is used for just detecting selfish nodes and there is no penalty on such nodes after they are detected.
Sengathir and Manoharan [9]	Proposed an Erlang distribution based Conditional Reliability Mechanism (ECRCM) that aids in detecting and isolating the selfish nodes present in an ad hoc environment	Estimates the impact of selfish nodes on the network and performs well as compared with other selfishness mitigating techniques	Selfish nodes do not participate further in any network activity and this increases the workload on the cooperating nodes.
Gupta et al. [10]	Proposed a strategy for dealing with selfish node behaviour which is dependent on selfish node concentration to mitigate the impact of concentration on the quality of service in MANETs.	Estimates the concentration of selfish nodes in the network and the impact on the network activity at different level of selfish node concentration.	Selfish nodes do not participate further in network activity and this increases the workload on the cooperating nodes. This scheme requires beaconing which may incur quite a bit of overhead
Yuanjie and Xiaojun [11]	Proposed a scheme to encourage packet forwarding and discipline selfish behaviour in a non-cooperative ad hoc network. A punishment scheme is designed to punish selfish nodes.	It does not require tamper-resistant hardware in contrast to the tamper-resistant hardware, which is not applicable for a pure ad hoc network	Selfish nodes do not participate further in network activity and this increases the workload on the cooperating nodes. This scheme requires beaconing which may incur quite a bit of overhead
Kumar et al. [12]	Proposed a token-based umpiring technique (TBUT) in which every node needs a token to participate in the network activities and the neighboring nodes acts as an umpire. Umpire nodes will monitor the behavior of the nodes and detect if any node is misbehaving.	It is very efficient with reduced detection time and less overhead. It is predominant in enhancing the network performance and improving the security of most of the real application	Selfish nodes do not participate further in any network activity and this increases the workload on the cooperating nodes.
Manoharan and Sengathir [13]	Proposed a mathematical model called Erlang coefficient based conditional probabilistic model (ECCPM) which aids decision on isolating selfish nodes through the manipulation of Conditional	The scheme estimates the level of negative impact produced by selfish nodes toward the resilience of the network.	The scheme increases the packet delivery ratio and throughput to a maximum, and reduces control overhead influence of minimum and maximum threshold-based detection.

	Probabilistic Coefficient (CPC) factor.		

3. Methodology

The benefit of an efficient MANET is best obtained by nodes if every node participates in packet forwarding, however, if a node refuses to forward other nodes' packets, this results in the uncooperative node packets' being forwarded to destination node while refusing to forward packets from other nodes to their destination nodes if all nodes refuse to forward packets then no network is formed.

Sumiti and Sumit [14] provided three different algorithms to recognize three different forms of selfish nodes. The token based, agent based and watch dog methods were simulated in a random mobile network with selfish nodes present. Their performance was compared in terms of packet delivery, bytes communication, packet loss and delay parameters. The result showed that the token-based method provided a better result for packet delivery, byte communication and packet loss parameters. Whereas, the results of packet delay were mixed and the delay in case of watchdog and agent based was better than token - based method.

The simulation results for TBUT in [12] show that the technique is efficient for packet routing, however it could also give room for improving performance by minimizing the penalty to an innocent node.

As a result of the aforementioned, this research, therefore, faults the TBUT based on its unfair quarantining nature of selfish nodes immediately they are detected. Also, if all the nodes in a MANET are quarantined, then there might at some point no be network communication as all nodes tend to be selfish.

This research work is based on the technique proposed by Kumar et al [12]; the system employed the use of tokens in its analysis to accelerate the detection and elimination of selfish nodes. Each node is issued with a token at its inception. The token consists of three fields: NodeID, status, and reputation. NodeID is assumed to be unique and deemed to be beyond manipulation; status is a single-bit flag. The status bit and the reputation value are initialized to zero. The token with a green flag and positive reputation is a permit issued to each node, which confers on it the freedom to participate in all network activities. Each node to participate in any network activity has to announce its token status bit and reputation value. If token status bit and reputation value are "1" and "-1" respectively, the protocol does not allow the node to participate in any network activity.

Therefore, this research work is aimed at improving the performance of TBUT by employing an incentive mechanism, to encourage cooperation and discourage selfishness among nodes in a MANET, and modifying TBUT by introducing a tolerance technique to reduce the impact of selfish node quarantining on the network, thus leading to more robust and reliable network performance.

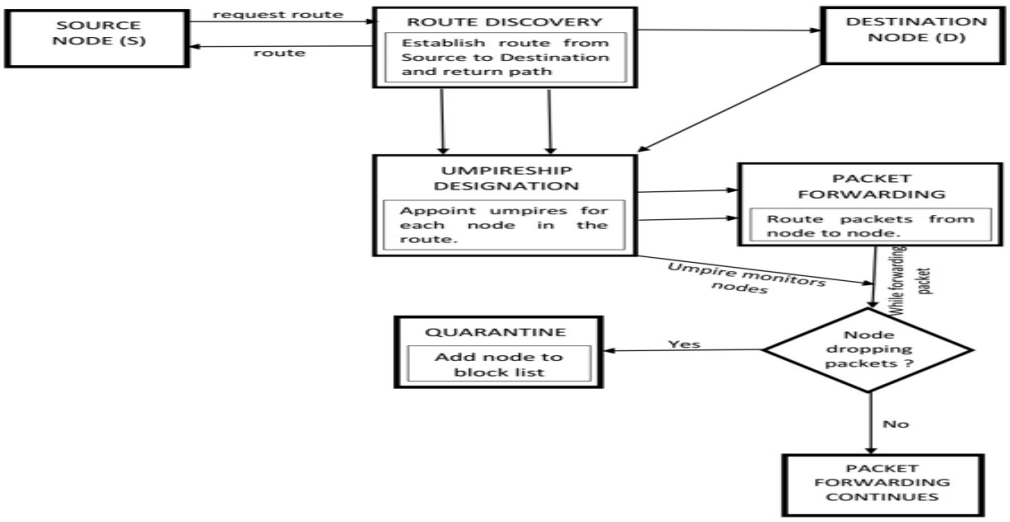

Figure 1: A representation of the TBUT model [12]

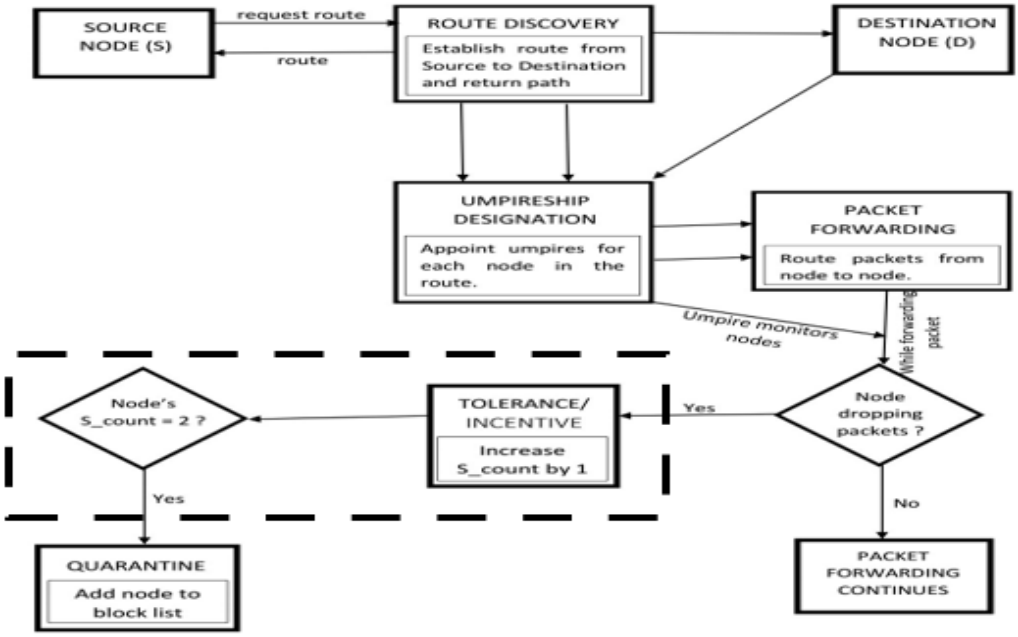

Figure 2: Propose improved TBUT architecture

3.1 Algorithm for Selfish Node Detection and Quarantining

Figure 3 shows the selfish node detection and quarantine algorithm used in this research work.

```
Begin
1.  //Route Discovery
2.  Step 1: Source (S) sends a RREQ to Destination (D)
3.  Step 2: Destination (D) receives the RREQ from source (S) and send RREP
    to the source (S)
4.  Step 3: Umpiring node are appointed to the corresponding neighboring
    nodes in the network.
5.  //Reputation Calculation
6.  Step 4: Neighbor information is gathered and sensed.
7.  Step 5: Cross checking with the neighbour reputation information and their
    own reputation calculation.
8.  If (own reputation calculation value == neighbour reputation information)
9.  {
10. If (selfish node is detected)
11. Set S_Count = S_Count + 1
12. If (S_Count ==2)
13. Set flag bit "1" and reputation value bit "-1"
14. Add selfish node to the selfish block list (SBL)
15. Else
16. Reputation value is incremented by 1
17. Transfer the data to its corresponding destination node;
18. }
19. Step 6: Finally, the performance is measured.
```

Figure 3: Selfish Node Detection and Quarantine Algorithm

3.2 Incentive Mechanism

A selfish node will need an incentive to forward other node's packets since doing so will incur a cost (of energy and other resources). The incentive mechanism ensures that messages from defaulting nodes are not accepted, thus forcing them to participate if they want their messages delivered. Therefore, whenever a node B has its S_Count as 1, it is put on probation and is considered selfish by node A and the other two umpires, it is notified, and node C won't accept any messages from it (but node A will keep on forwarding its messages to node B). Therefore, a selfish node might end up not being able to send its messages, unless it becomes unselfish.

4. Performance Evaluation using NS-2

This research made use of a popular network simulator called Network Simulator 2 (NS2) for performance testing due to its large number of available realistic mobility models, powerful scripting, simulation setup, and most importantly, its large number of the user community for technical support. Qualnet 5.0, an event-driven network simulator was used to implement the TBUT framework, but due to its unavailability in the commercial-off-the-shelf market, although it is a commercial software product as stated by its vendors, we were left with the choice of simulation using the next best tool that can support the simulation of our ideas.

NS2 provides support for simulation of Transmission Control Protocol (TCP), routing, and multicast protocols over wired and wireless (local and satellite) networks [11]. It is based on C++ which is used to define the behaviour of protocols, and OTcl (Object Oriented Tool Command Language), which is used for scenario configuration, manipulation of existing C++ objects, periodic or triggered actions. NS-2 uses OTcl programming language to interpret user simulation scripts. The Tcl language is fully compatible with the C++ programming language. At the top layer, NS-2 is an interpreter of Tcl scripts of the users [12]. Fig. 4 shows the use of NS-2 in interpreting OTcl script.

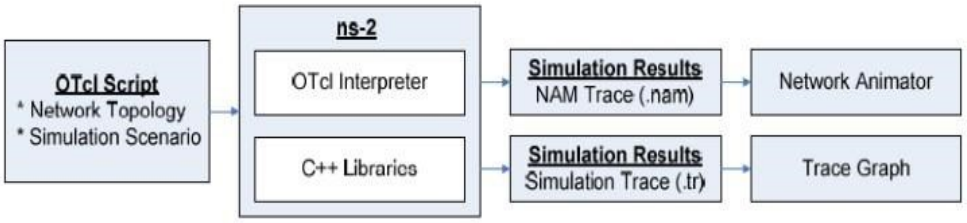

Figure 4. NS-2 Schema [12]

NS-2 has the NAM (Network Animator) object which is a Tcl based animation tool for viewing network simulation traces, and it shows the visual animation of the simulation. Another component is the trace object that contains a history of the behaviour of all objects in the simulation. They are both created as files by NS-2. The former is a .nam file used by NAM software that comes along with NS-2.

Fig. 5 shows the Nam environment consisting of 100 mobile nodes in NS-2. The latter is a ".tr" file that includes all the simulation traces in the text format. NS project is normally distributed along with various packages (ns, nam, tcl, otcl etc) named as "all-in-one package", but they can also be found and downloaded separately.

In this study, we have used version 2.35 of NS all-in-one package and installed the package in the Linux environment. The Tcl script was written in a text editor (gedit) and the Linux command, "ns filename.tcl" was used to initiate the simulation.

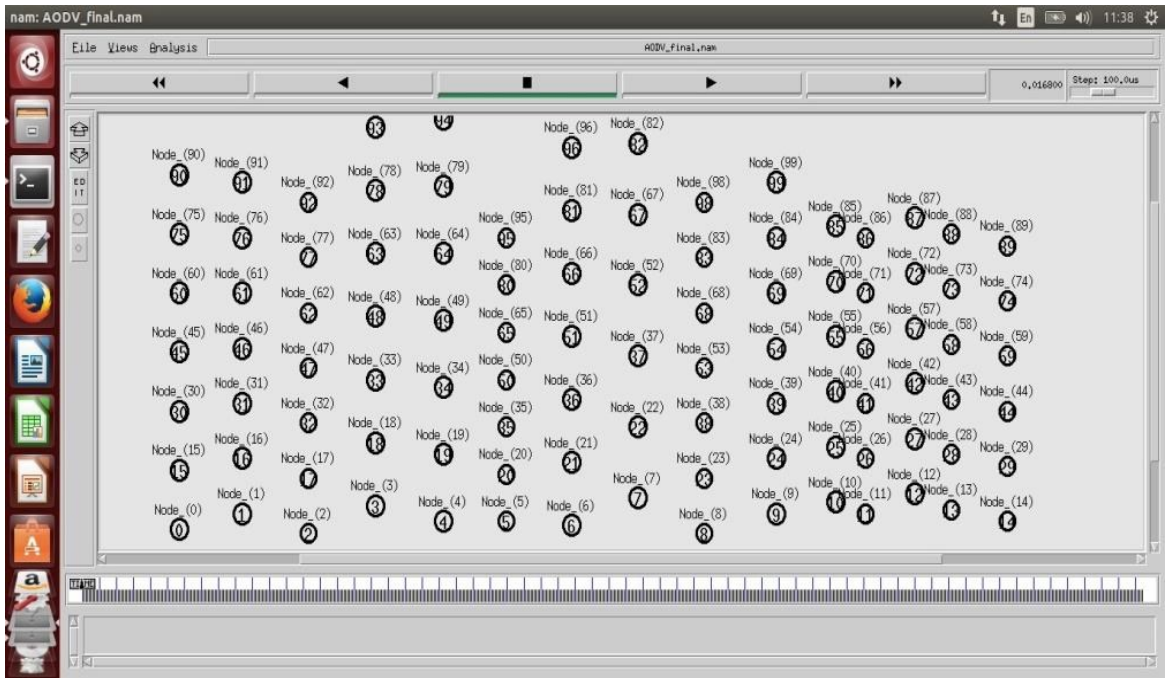

Figure 5: NAM environment of NS – 2

4.1 Performance Test and Simulation Parameters

Our simulation parameters are set as shown in Table 2.

Table 2: Simulation parameters

Simulation Parameters	Values
Simulation time	1000s
Transmission range	250m

Bandwidth	2Mbps
Mobility model	Random waypoint
Propagation model	Two-ray ground reflection
Maximum speed	0 – 20 m/s
Pause time	0s
Traffic type	Constant Bit Rate (CBR)

4.2 System Implementation

4.2.1 Implementation of the Selfish Node Behaviour

Though NS 2.35 contains the routing protocol (AODV) we require for this research, it does not have any module to simulate the selfish behaviour in nodes. To this effect, we modified the AODV routing protocol file using C++ in the NS 2.35. We implemented the selfish node behaviour by editing the aodv.cc and aodv.h file whereby the selfish node will simply drop the packet, then recompiled NS-2 files to create an object. Haven finished compilation; we had a new testbed to simulate selfish nodes using the AODV protocol.

4.2.2 Implementation of Selfish Node Detection and Quarantine

In this research work, we achieved selfish node detection by setting the nodes in promiscuous mode, whereby each node in the network listens for packets transmitted by its neighbour nodes and takes action in the event of a packet drop. To accomplish this, we modified three files; aodv.cc, aodv.h and ns-mobile node.tcl.

4.2.3 Implementation of the Tolerance and Incentive Mechanism

The hybrid tolerance with incentive mechanism was implemented by modifying the token. We added to the token of each node, a S_count field which serves the dual purpose of reducing node quarantine and discouraging selfishness among nodes by punishing of non-cooperating nodes.

4.3 Performance Metrics

The presence of selfish nodes in the network decreases the throughput and increases the total and control overhead [15]. Hence the developed technique is analyzed based on its packet delivery ratio.

4.3.1 Throughput of nodes at different Speeds

We generated the throughput for nodes moving at speeds between 0 – 20 m/s mobility as shown in Fig. 6, using an awk file with the command;

awk –f genthroughput.awk filename.tr

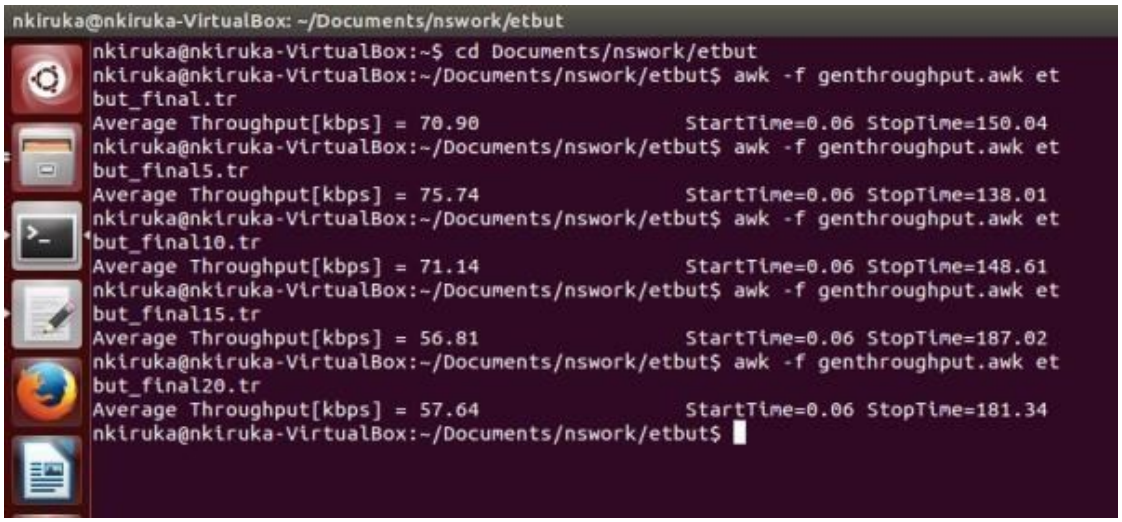

Figure 6: Computing the throughput for nodes between 0 – 20 m/s mobility

4.3.2 Packet Delivery Ratio (PDR)

This is the ratio of the number of packets received by destination nodes to the number of packets sent to it by source nodes. It is accepted as a standard measure of throughput. Fig. 7 shows the packet delivery ratio in the presence of 30% selfish nodes. Five different files were created to obtain the packet delivery ratio for mobility from 0 to 20 m/s. PDR decrease as speed of nodes increase, we notice that TBUT performs slightly better than ETBUT for speeds between 0 – 20 m/s for now.

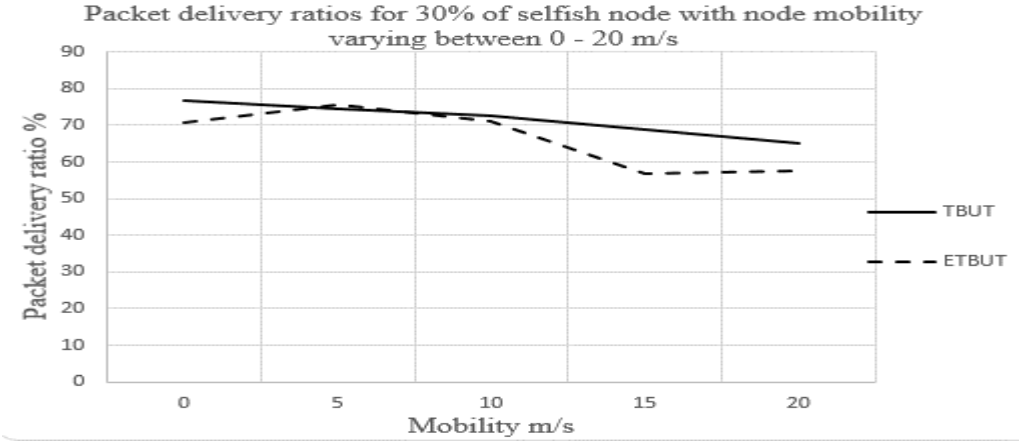

Figure 7: Packet delivery ratio, for 30% selfish node with node mobility varying between 0 - 20 m/s.

4.4 Throughput of Generating Packets

The graph plotted in Fig. 8 shows the throughput of forwarded packets on the y-axis versus the simulation time on the x-axis. Fig. 8 reveals an increase in throughput at a simulation time of 60s. We find that the plots agree with literature and provide the basis for the result obtained.

Let number of packets generated by nodes in the network be represented by P, number of packets dropped by nodes be represented by Q, and number of packets reaching destination nodes be represented by R.

Then,

$$packet\ delivery\ ratio = \frac{no.of\ packets\ reaching\ destination}{no.of\ packets\ generated\ in\ the\ network} = \frac{R}{P} \quad (1)$$

$$packet\ drop\ ratio = \frac{no.of\ packets\ dropped}{no.of\ packets\ generated\ in\ the\ network} = \frac{Q}{P} \quad (2)$$

But, $P = R + Q$, (3)

Hence, $1 = \frac{R}{P} + \frac{Q}{P}$ (4)

And, $\frac{R}{P} = 1 - \frac{Q}{P}$ (5)

Using Equation (5), we see that if an increase in per cent of selfish nodes in the network should arise, and consequently, the packet drop ratio increase, then the packet delivery ratio will fall. In Fig. 8, we capture the packet delivery ratio with 30% selfish nodes, while Fig. 9 captures the packet drop ratio with 30 % selfish nodes, and note the complimentary character of packet delivery ratios and packet drop ratios.

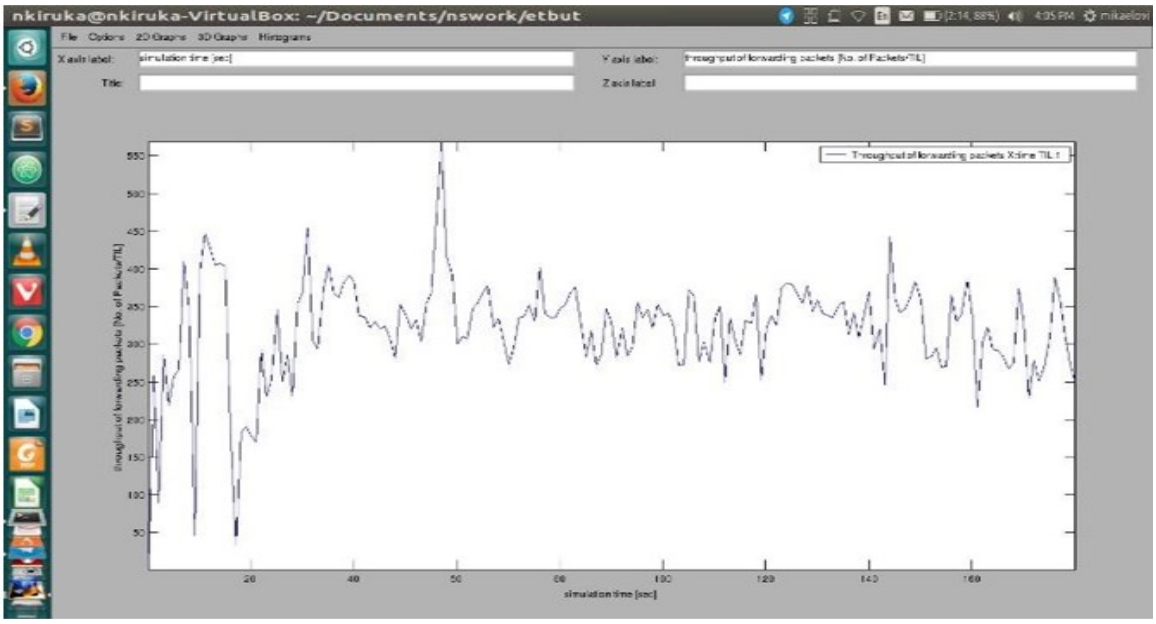

Figure 8: Throughput of generating packets Vs simulation time

4.4.1 Throughput of Dropped Packets

Fig. 9 shows the graph of the throughput of dropping packets on the y-axis and the simulation time on the x-axis.

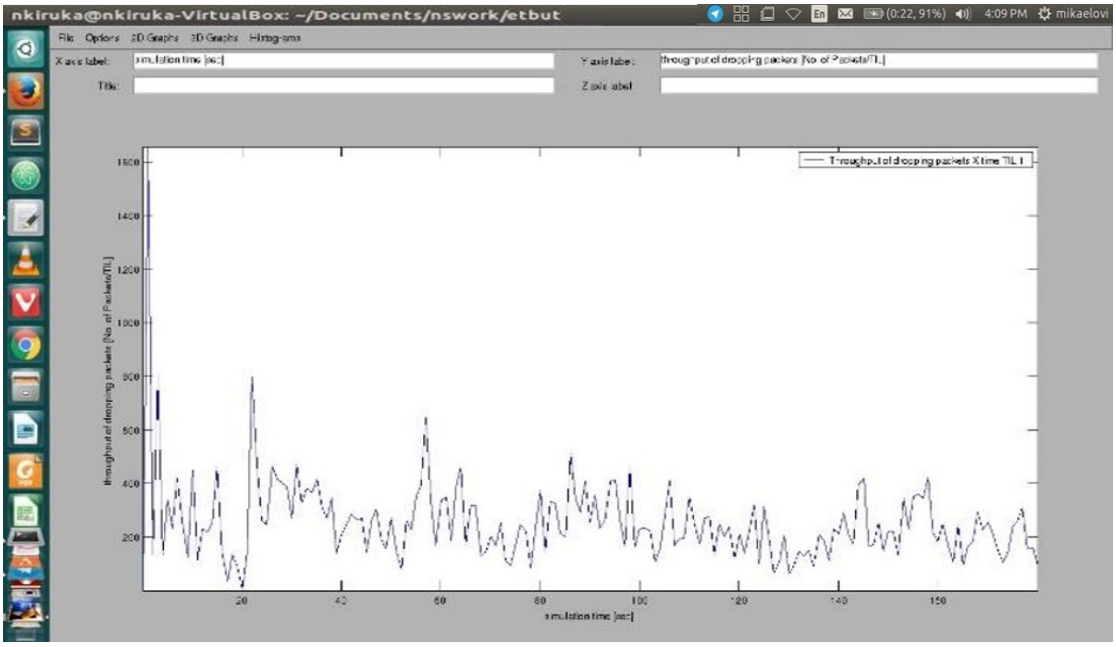

Figure 9: Throughput of dropping packets Vs simulation time

4.5 Result and Discussion

In our simulation, we observed that introducing the S_count field reduced the number of quarantined nodes in the 1000s simulation time. This helped to mitigate the impact of selfish node quarantining which is implemented by placing a selfish node in a block list. Though it is not possible to implement a block list in

NS2, however, we were able to achieve our idea by moving the selfish node out of radio range so that they are unable to communicate.

The S_count field which was added to the token of each node in the mobile ad-hoc network served its purpose of reducing node quarantine and also as a discouragement for selfishness among nodes by way of punishing non-cooperative nodes.

During our investigation, using the packet delivery ratio parameter, we observed that, with 30% selfish nodes and node mobility varying between 0 - 20 m/s, the developed system revealed an increase in throughput at a simulation time of 60s. This is comparable with the case of TBUT [12], however, beyond this time the throughput deviates from the performance of TBUT slightly.

5. Conclusions and Future Work

Presently, most literature available on selfish node detection and prevention focus more on detecting and eliminating selfish nodes immediately they drop packets. This research faults this elimination approach, for the following reasons:

- Many legitimate reasons exist for packet dropping behaviour in nodes.
- Every node in MANET tends towards being selfish and quarantining or eliminating them on the spot without giving them a second chance might at some point result in degradation in network performance or all together, no network communication.

However, based on the benefits of the developed model, to the problems it stands to mitigates; we consider it to be a more efficient and secure security scheme for MANET.

Further work is required to investigate the possibility of ensuring that extra overheads arising from resending lost packets are avoided. We believe that packets dropped by selfish nodes can be forwarded by engaging the umpires in the routing process. We also aim to develop an efficient method of forcing a quarantined node to resume forwarding packets to other participating nodes.

References

[1] Aarti, D. S. (2013). Study of MANET: Characteristics, Challenges, Application and Security Attacks. International Journal of Advanced Research in Computer Science and Software Engineering.

[2] Ciobanu, R., Dobre, C., Dascălu, M., Trăuşan-Matu, S., and Cristea, V. (2014). SENSE: A collaborative selfish node detection and incentive mechanism for opportunistic networks. Journal of Network and Computer Applications, 41, 240-249

[3] Hussain, M. A., Nadeem, A., Khan, O., Iqbal, S., and Salam, A. (2012). Evaluating network layer selfish behaviour and a method to detect and mitigate its effect in MANETs

[4] Sanaz, N., and Shahram, B. (2018). A Credit-based Method to Selfish Node Detection in Mobile Ad-hoc Network, Applied Computer Systems, 23(2), 118–127.

[5] Tamilarasi, G., and Selvam, D. (2013). Allocation of Replicas by Solving Selfishness in MANET, International Journal of Engineering Research & Technology (IJERT), 2(1), 1 – 6.

[6] Kashif, M., and Kadam, P. V. (2016). Improved collaborative watchdog system for detection of selfish node in MANET. International Journal of Science, Engineering and Technology Research (IJSETR).

[7] Selvan, M. and Selvakumar, S. (2019). Malicious node identification using quantitative intrusion detection techniques in MANET. Cluster Computing. 22. 1-9. 10.1007/s10586-018-2418-2.

[8] Enrique H., Olmos, M. D. S., Cano, J., Calafate, C. T., and Manzoni, P. (2015). CoCoWa: A Collaborative Contact-Based Watchdog for Detecting Selfish Nodes. IEEE Transactions on Mobile Computing (Volume: 14, Issue 6, June, 2015, pp. 37 – 42

[9] Sengathir, J., and Manoharan, R. (2016). An Erlang Factor-Based Conditional Reliability Mechanism for Enforcing Co-operation in MANETs. Serbian Journal of Electrical Engineering, Vol. 13, No. 2, June 2016, 265-284

[10] Gupta, S., Nagpal, C., and Singla, C. (2011). Impact of Selfish Node Concentration in MANETs. International Journal of Wireless & Mobile Networks. 3. 10.5121/ijwmn.2011.3203.

[11] Yuanjie L., and Xiaojun, W. (2019). Cooperative packet-forwarding strategies in mobile ad hoc networks with unreliable channels: An evolutionary game approach International Journal of Distributed Sensor Networks 2019, Vol. 15(9), pages 1–14.

[12] Kumar, A. K., Jebakumar, M., and Singh, P. J. (2015). A unified approach for detecting and eliminating selfish nodes in MANETs using TBUT. EURASIP Journal on Wireless Communications and Networking, Volume 10, No. 30, June 2016, pages 32–37.

[13] Manoharan, R., and Sengathir, J. (2016). Erlang coefficient based conditional probabilistic model for reliable data dissemination in MANETs. Journal of King Saud University - Computer and Information Sciences (2016) 28, 289-302.

[14] Sumiti, and Sumit M. (2019). An Experimental Analysis on Selfish Node Detection Measures and Methods. International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-8 Issue-8S3, June 2019.

[15] Madhurikkha, S., and Sabitha, R. (2016). Detection of malicious packet droppers in MANET based on legitimate routing information. Journal of Chemical and Pharmaceutical Sciences (JCPS), ISSN: 0974-2115, Volume 9 Issue 3, 1621 – 1626, July - September 2016.