

Machine Learning Based Trust Management framework for Sustainable Vehicular Networks



By
Unaiza Alvi
00000319231

Supervisor
Dr. Asad W. Malik
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree
of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(December 2021)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Machine Learning Based Trust Management framework for Sustainable Vehicular Networks" written by UNAIZA ALVI, (Registration No 00000319231), of SECS has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: _____  _____

Name of Advisor: Dr. Asad Waqar Malik _____

Date: _____ **29-Nov-2021** _____

Signature (HOD): _____

Date: _____

Signature (Dean/Principal): _____

Date: _____

Approval

It is certified that the contents and form of the thesis entitled "Machine Learning Based Trust Management framework for Sustainable Vehicular Networks" submitted by UNAIZA ALVI have been found satisfactory for the requirement of the degree

Advisor : Dr. Asad Waqar Malik

Signature: Asad

Date: 29-Nov-2021

Committee Member 1: Dr. Anis Ur Rahman

Signature: Anis Ur Rahman

30-Nov-2021

Committee Member 2: Dr Muazzam A Khan
Khattak

Signature: Muazzam A Khan Khattak

Date: 30-Nov-2021

Signature: _____

Date: _____


Dedication

Dedicated to my family
for their love, kindness, and support

Certificate of Originality

I hereby declare that this submission titled "Machine Learning Based Trust Management framework for Sustainable Vehicular Networks" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEecs or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEecs or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name:UNAIZA ALVI

Student Signature:  _____

Acknowledgment

First and foremost, I would like to express my gratitude to Allah Almighty for all of his blessings. Following that, I owe a debt of gratitude to my mentor, Dr. Asad Waqar Malik, for his invaluable guidance, humility, encouragement, dedication and patience. I am grateful to my advising and evaluating committee, Dr. Anis ur Rahman and Dr. Muazzam A. Khan Khattak, for their time, useful suggestions, and constructive criticism. I would also like to express my gratitude to all of my hostel companions for their unwavering support throughout my highs and lows, especially during the tough times of COVID-19 pandemic. Finally, I want to thank my family for their constant love, generosity, and support.

Table of Contents

List of Abbreviations	viii
List of Tables	x
List of Figures	xi
1 Introduction	1
1.1 Background	1
1.2 Security Threats in Inter Vehicle Communication	2
1.2.1 Types of Attackers	3
1.2.2 Types of Attacks	3
1.3 Trust Management in VANETs	4
1.3.1 Elements of Trust	4
1.3.2 Direct Trust	4
1.3.3 Indirect Trust	4
1.3.4 Types of Trust Models in VANETs	4
1.4 From VANETs to VEC	5
1.5 The notion of Trusted Task Offloading in VEC	6
1.6 Motivation	6
1.7 Objectives of our Research	6
1.8 Organization of Thesis	7
2 Literature Review	8
2.1 Conventional Techniques	8
2.2 Machine learning based Techniques	10
2.3 Summary	11
3 Problem Statement and Mobility Simulation	12
3.1 Problem Statement	12
3.2 Network Model	13
3.3 Communication Model	13

3.4	Task Computational Model	14
3.5	Adversary Model/Attack Model	15
3.6	Vehicle Mobility Simulation Tools	16
3.6.1	Converting Manhattan City Map into SUMO Network	17
3.6.2	Vehicle Trips and Route Generation	18
3.6.3	Setting Configuration File and Running the network .	20
3.7	Summary	20
4	Proposed Intelligent Multi-criteria Trusted Offloading Decision Framework	21
4.1	Data set Generation	21
4.2	Machine Learning Model Training, Selection and Deployment .	23
4.3	Multi Criteria Offloading Framework	25
5	Experimental Evaluation, Results and Discussion	31
6	Conclusion and Future Work	39
6.1	Conclusion	39
6.2	Future Work	39

List of Abbreviations

Abbreviations

ITS	Intelligent Transport System
VANETs	Vehicular ad hoc networks
GPS	Global Positioning System
MANETs	Mobile ad hoc networks
OBUs	On Board Units
RSUs	Road Side Units
VEC	Vehicular Edge Computing
WHO	World Health Organization
LA	Local Authority
DTC	Digital Trustworthiness Card
TS	Trust Server
IoT	Internet of Things
SVM	Support Vector Machines
KNN	K Nearest Neighbour
ANN	Artificial Neural Network
DDoS	Distributed Denial of Service
TraCI	Traffic Control Interface
OSM	Open Street Map
LSTM	Long Short Term Memory
ROC	Receiver Operating Curve
AUC	Area Under the Curve

Nomenclature

V	Set of vehicles
R	set of RSUs
T	computational task
S_T	Input size of task
C_T	CPU cycles required for task processing
D_T	Task completion deadline
T_{local}	Local execution time
f_v	Vehicle compute power
Q	Task queue
W_x	Waiting time at local queue
O_T	Offloading decision
T_{dest}	Task completion time at destination vehicle
T_{src}	Task completion time at source vehicle
NBR	List of vehicles neighbors
l	Size of neighbors list
$T_{offload}$	Execution time for offloaded task
P	Set containing vehicle predictions
α	Cumulative count for honest predictions
β	Cumulative count for malicious predictions
λ	Vehicle final selected label

List of Tables

4.1	Description of data-set features.	22
5.1	Simulation Environment and System Specifications	31

List of Figures

1.1	A typical inter vehicle communication scenario in VANETs [1].	2
1.2	Classification of trust models in VANETs	5
3.1	A snapshot demonstrating the Manhattan city area used from Open Street Map.	17
3.2	Input command for the conversion of OSM map into SUMO network.	17
3.3	Output file after the successful conversion of OSM map into SUMO network file.	18
3.4	Command to generate vehicle routes and network edges in SUMO simulator.	18
3.5	Vehicles trips output file to generate random trips in the simulation.	19
3.6	Output file for the generated vehicle routes.	19
3.7	The simulated vehicular network with vehicles on the roads. .	20
4.1	The work flow of Machine Learning Framework.	24
4.2	Architecture diagram demonstrating the working of proposed framework	29
5.1	The reported precision with varying percentage of attacking vehicles for various models.	32
5.2	The reported recall of various machine learning models as the percentage of attacking vehicle varies.	32
5.3	F1 score of various machine learning models against different attacking vehicles ratio.	33
5.4	The reported F2 score with varying density of attacker nodes for various machine learning models.	33
5.5	A graph representing log loss with varying attacker density for various models.	34
5.6	The reported ROC for various models at 50% attacker density	34
5.7	Task Efficiency achieved at varying attacker density.	35

5.8	Effectiveness at varying attacker density.	36
5.9	Black hole failures at varying attacker density.	36
5.10	Average wait time incurred at varying attacker density.	37

Abstract

Vehicular Edge Computing (VEC) augments resource-constrained devices in vehicular networks by bringing services and processing power closer to the end-user at the network edge. However, over time, these devices get overburdened as a result of incoming requests, and network performance degrades. This can be addressed by utilizing nearby idle nodes and assigning tasks to nearby vehicles; a process known as computational task offloading. However, the presence of malicious nodes might put the entire network at risk; tasks cannot be offloaded if the node is untrustworthy; therefore service trustworthiness is critical. The majority of the work on task offloading has focused on resource optimization, and the trustworthiness of services has received less attention. The traditional trust models focus on the aggregation of both direct and indirect observations. The choice of optimal weights for different factors in traditional approaches is still a key challenge given the dynamic nature of VANETs (vehicular ad hoc networks). Thus, we employ the establishment of trust and identification of malicious vehicles as a classification problem for task offloading in vehicular networks. We have simulated multiple attacking patterns and generated a novel data set to identify misbehaving nodes. We trained multiple machine learning models on the generated data set. LSTM (Long Short Term Memory) reported the highest performance gain. Moreover, we deployed the trained model on edge nodes and proposed a multi-criteria task offloading framework in vehicular networks. In the presence of adversary nodes in the network, the proposed task offloading framework with integrated intelligent layer outperformed baseline techniques in terms of task efficiency, effectiveness, and black hole failures.

Chapter 1

Introduction

In recent years, rapid urbanization and population growth in Pakistan has emerged the concept of Smart cities. Intelligent Transport System (ITS) is an essential element in Smart Cities to enhance road safety and driving experience. VANETs (Vehicular ad hoc networks) are the building blocks for ITS, thus it's security and privacy issues have to be handled smartly for its commercial deployment. Vehicular networks offer tremendous benefits in terms of road safety and value-added services; however, its vulnerability to various risks pose great hurdles for its commercial deployment. One of the threat, is the injection of forged messages within the network. The vehicles highly rely on the exchanged messages for making decisions. If that information is compromised, it can have dire consequences. The presence of adversary nodes can jeopardize the network by spreading bogus messages, causing road accidents and traffic congestion. So a trust model is crucial, that can assist the nodes for evaluating the trustworthiness of the nodes and evicting the malicious nodes to ensure the reliability of messages and to make informed decisions.

1.1 Background

The idea of VANETs emerged in the 90's, with the easy availability of GPS (Global Positioning System) and low cost wireless devices. Various projects such as ASV (Advanced Safety Vehicle) and promote chauffeur established a foundation in the various areas of vehicular networks such as communication, design, routing protocols and architecture etc. Vehicular networks are extension of MANETs (mobile ad hoc networks). In a typical vehicular network, each vehicle equipped with smart sensors and OBUs (On Board Units) is considered as a node, capable of exchanging communication messages with

other nodes or stationary access points present among different junctions of the roads called RSUs (Road Side Units).

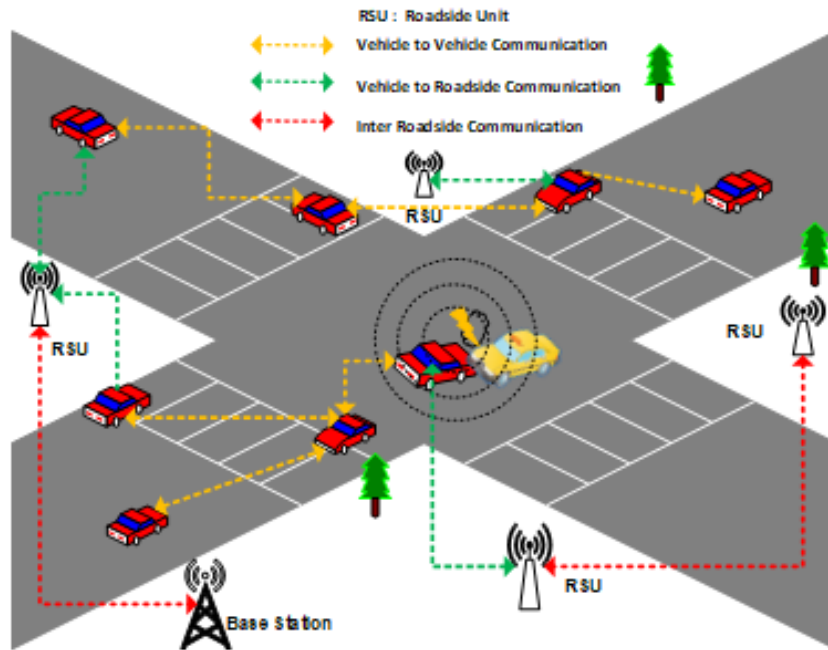


Figure 1.1: A typical inter vehicle communication scenario in VANETs [1].

They offer a plethora of applications ranging from basic entertainment applications to safety critical application such as accident alert systems. Its applications can be broadly categorized into three classes : road safety, industrial and infotainment. It is one of the key component of the ITS with the goal to improve driver and road safety in general [2]. The World Health Organization (WHO) report states that road accident fatalities are the eight leading cause of death, with approximately 1.35 million reported deaths. Thus VANETs can play a vital role in traffic safety by generating early accident alerts or warning the driver etc. [1]

1.2 Security Threats in Inter Vehicle Communication

The VANETs pose greater future prospects, however their highly mobile and ad-hoc nature makes it vulnerable to various security threats. The attackers can vary according to the kind of malicious activities they perform and the

access they have to the network. In this section we will discuss different kind of attackers and different attacks in context of vehicular network.

1.2.1 Types of Attackers

The attackers in a vehicular network can be divided into following categories:

Insider vs Outsider

The insider attacker is someone who is already in the network i.e a legitimate node whereas external attacker refers to the node who do not have access to the network.

Malicious vs. Rational

The malicious attacker's intention is to deteriorate the entire network without any personal gains, however a rational attacker harms the network for its own personal gains.

Active vs. Passive

The active attacker tries to forge the message, however a passive attacker tries to sense the network messages.

Local vs Extended

The local attacker has a limited reach for attack, whereas an extended attacker cooperates with other entities to expand its reach and capacity to generate a more powerful attack [3].

1.2.2 Types of Attacks

We discussed about various kind of attackers according to their roles and attacking capacity above. The most common types of attacks in a vehicular network are discussed below :

Sybil Attack

In this attack, the attacking node pretends to have multiple identities.

Denial-of-Service (DoS) Attack

In this attack, the intruder overwhelms the entire network with multiple requests such that the services are no longer available to the legitimate nodes.

Black-hole Attack

The attacker simply discards the received packets when operating in this mode.

Wormhole Attack

Two or more nodes are involved in such kind of attack, they harm the network by modifying logical topology of the network by advertising they know the shortest path, they aim to modify a larger portion of traffic.

Replay Attack

It is the kind of attack where an intruder stores a piece of information and then communicates it to other nodes, when the shared information is no longer valid.

Passive Eavesdropping Attack

In this attack the goal of the intruder is to monitor the network activities of various nodes, without forging any kind of information [4].

1.3 Trust Management in VANETs

1.3.1 Elements of Trust

Trust can be divided into two elements based upon the interactions among the participating nodes.

1.3.2 Direct Trust

It is based upon the direct observation between the trustor i.e the vehicle that is computing the trust and the target vehicle i.e the vehicle about which trust is calculated. Direct trust is often given more preference as opposed to indirect trust.

1.3.3 Indirect Trust

It considers the rating of the vehicles in neighborhood to establish trust for the target vehicles. The neighbors give opinion for the target vehicle based on their past experiences [5].

1.3.4 Types of Trust Models in VANETs

The trust models in vehicular networks can be divided into following categories:

Entity Oriented–

These models are concerned with whether to trust a given node or not, they are based on the nodes interaction data upon which they make any decision about its trust-worthiness.

Data Oriented–

These models establish trust based on the evidence collected to verify whether

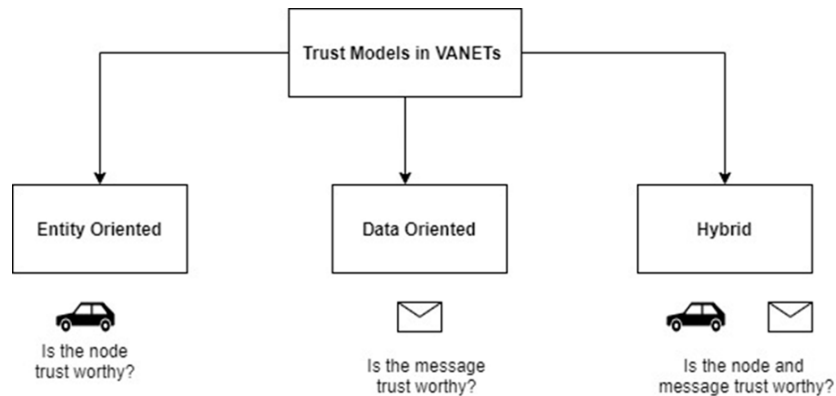


Figure 1.2: Classification of trust models in VANETs

the information that is communicated is correct or not. Each piece of evidence is given a specific weight-age.

Hybrid/Combined–

In this model the trust-worthiness of received information and the trust-worthiness of the sending node both are evaluated [6].

1.4 From VANETs to VEC

The number of automobiles on the roads are increasing sharply, consequently the need for communication and computing requirements for the timely provision of services is rapidly growing. Many compute-intensive systems need a high level of quality-of-service assurance. Edge computing in vehicular networks is proposed to alleviate the challenges by extending computing capability to the network edge. The services are brought closer to the end user, resulting in a new computing domain know as VEC (Vehicular Edge Computing). With the introduction of VEC, the user experience is enhanced since the services are provided directly at the network edge. Its lightweight and rich resources provide a better experience to the end user. Moreover, it helps to enhance the overall user experience because of the additional knowledge of applications, such as its location and context, so the needs can be catered accordingly [7].

1.5 The notion of Trusted Task Offloading in VEC

Although with the advent of VEC, overall user experience was enhanced by making accessibility to the services easier and readily available, particularly for the time sensitive applications. However, resource-constrained sites are frequently swamped by a large number of incoming requests, making it difficult to maintain lower end-to-end delays for delay-sensitive applications. To cater these, nearby idle nodes with enough computational resources can be utilised, to offload the tasks and achieve performance gain. However, one can not comment on the participating nodes if they will meet the tasks requirements, since some participating nodes could be malicious, which necessitates the implementation of a trust management system within the network to fully utilise task offloading. One can not judge a node if it's a malicious node or honest node, thus this introduces the concept of "**trusted task offloading**" i.e the task is offloaded to the node only if it's trustworthy [8].

1.6 Motivation

The rapid urbanization and growing population has necessitated the incorporation of technologies for better lifestyles, one such thing is the concept of the transformation of cities into Smart Cities. Smart Cities can offer tremendous applications in terms of entertainment and road safety in general. VANETs are the key technologies used in smart transportation. VANETs have been a topic of interest for researchers, since long time and it has seen many technological changes such as the introduction of edge computing, volunteer computing, task offloading, to name a few. The main focus of this research is to introduce the notion of trusted task offloading in a vehicular network. Since majority of the works focus on general intrusion detection and task offloading with an added element of trust has been least explored. Task offloading is critical in edge networks where resources are overwhelmed, thus they can utilise the idle resources in their vicinity, however the reliability of the offered services is critical, since the presence of adversary nodes can deteriorate the network performance.

1.7 Objectives of our Research

Our research study focuses on simulating a mobility model to represent a task offloading scenario. The most commonly occurring attacks are also modeled

in the network. Once the mobility and attacker model is built, the next step is the development of a novel data set for intrusion detection in a task offloading environment in vehicular networks. The main objective behind the generation of a data-set is due to the fact that, most of the studies focus on intrusion detection in general in a VANET environment and no data set is available for intrusion detection in the particular case of task offloading. Some studies that did simulate the different attacking patterns have not made those data sets publicly available, moreover the data sets lack heterogeneity and diversity i.e vehicles may exhibit more than one malicious pattern. Additionally we trained various machine learning models that can detect the adversary nodes within the vehicular network. Moreover, we also utilise these trained models and incorporate them in the simulation, which are used to filter the malicious nodes. With this, we proposed an offloading framework that uses intelligent layer to select the appropriate node for the task execution, since the notion of trusted task offloading by employing machine learning techniques is still in its infancy.

1.8 Organization of Thesis

This research study has been organised into 6 chapters. Chapter 1 introduces the concept of VANETs, why we need them and their issues in general. In Chapter 2, we cover an extensive literature on the proposed problem. Chapter 3 discusses, the problem formulation, system entities involved and their communication methods. Moreover, we explain in detail the simulation of vehicle mobility in the simulator, the development of the machine learning framework, which is later on deployed in the simulation. We also demonstrate the proposed offloading framework with intelligent layer incorporated. Experiments performed to validate our proposed model are discussed in Chapter 5. Finally the conclusion and future work is presented in Chapter 6.

Chapter 2

Literature Review

In this section, we will discuss various research studies proposed for trust management and identification of malicious nodes in a vehicular network. We have divided the studies into two classes namely conventional and machine learning based techniques, which are discussed below.

2.1 Conventional Techniques

In this section, we will discuss conventional or statistical techniques for reputation management in vehicular networks. The conventional techniques use different parameters and assign different weights to these parameters to compute a trust value, based on which the decisions regarding the trust-worthy or malicious nature of the node are made.

In [7] using multi weighted subject logic in VEC, the authors suggested a reputation system for vehicles distributed in nature. The LA (Local Authority) is in charge of aggregating vehicle reputations based on three criteria: familiarity, punctuality, and similarity. The reputation values for vehicles are updated after each engagement. They also discussed a framework for resource selection based on the principles of trust, i.e. a reliable resource is selected for the purpose of offloading.

Similarly, this research study [8] established the notion of trusted task offloading, which employs a distributed block chain to keep track of the vehicle reputation scores. The participating vehicles in the network are assigned some tasks that were offloaded from other resource constrained devices that could not execute the tasks locally. If a vehicle completes the assigned task, it is rewarded and if it does not complete the task it is penalised based on a criteria. This reward and penalty mechanism is used to update the trust values for the participating vehicles.

To remove malicious nodes authors in [9] proposed a hybrid scheme for trust establishment. It's a centralised reputation management technique that screens out malicious nodes that operates in one of three adversarial modes: bogus, secret, or collude. Their scheme converges quicker than previous centralised methods and outperforms them. Their method converges faster and with a 10% increase in accuracy. It detects untrustworthy vehicular messages by relying on global reputation and only taking into account those messages that are sent by reputable vehicles. It verifies data in order to keep the vehicle's reputation up to date. When a vehicle reaches a local server's zone, the local server checks its reputation with the central server and changes it based on the validations obtained for that specific vehicle during that interaction. The vehicle's reputation is globally synchronised after it leaves that location. Based on the number of fraudulent messages received, the vehicle's reputation is decreased exponentially.

Accordingly, to identify malicious nodes propagating fraudulent messages, authors in [10] proposed a reputation management system. The reputation score is derived by assigning weights to various criteria. Once the trust scores are calculated, the decision regarding the acceptance or rejection of the incoming messages is made. Accordingly, the vehicles are awarded or penalised further. The messages are categorised into three classes namely high, medium, or low severity. The technique ensures that messages having high severity level are only accepted from those vehicle which posses a highly trust able status and vice versa.

In [11], a fog computing-based trust system is presented. If the interaction among the nodes exist, the reputation is calculated based on direct interactions and if no interaction is available for that particular node then the neighbors (who have interacted with that node) recommendation is considered. In their study, for the computation of trust score, direct experience is always given higher weight-age than the indirect experience, implying that neighbors recommendation will only be considered and given higher weight age only if direct experience among those nodes does not exist.

This research [12] focuses on the placement of fog nodes in VANETs for the purpose of assessing trust. The Vehicle Digital Trustworthiness Card (DTC) is employed, and it contains the historical trust record for vehicles issued by trusted authorities. The study is based on vehicle trust data from the past interactions. The Trust Server (TS) compiles the information gathered from various sources. The maintained scores are further classified into two classes Frequent Visitors and Occasional visitors. The score can be retrieved directly if the node belongs to class A or class B and if the queried vehicle is new in the network, the score has to be accessed from the central server, which adds up to the total cost in terms of communication delay.

2.2 Machine learning based Techniques

Various studies employ machine learning models to identify and evict malicious vehicles in the vehicular network. In this section, we will discuss about how researchers have used machine learning techniques for the malicious node identification particularly in the context of vehicular networks.

The authors in [13] used a real IoT (Internet of Things) data set and converted into a VANET format for the purpose of this study. They performed feature extraction and used features like similarity, number of packets delivered and familiarity features for the computation of trust score. The trust score was computed using two methods namely individual scores and mean parametric scores. After the computation of trust scores, a threshold value was defined and vehicles were classified as malicious or honest based on that value. They employed multiple models such as SVM (Support Vector Machines), KNN (K Nearest Neighbour) etc. The KNN model with trust value computation using parametric scores reported better results.

J.Kamel et al. [14] presented a framework for misbehavior detection in vehicular networks based on the messages exchanged between the smart vehicles. They performed different plausibility checks to distinguish between the adversary nodes and honest nodes. They simulated six VANET attacks in their work and then employed machine learning techniques to detect these. Their framework allows other researchers to incorporate attacking patterns of their own and then evaluate the framework performance.

Li et al. [15] proposed a technique to identify malicious vehicles that could have multiple attacking patterns. They used SVM to create a boundary between the honest and malicious nodes. They used different behaviors observed by neighbors and various contextual information to train the models and identify adversary nodes.

In [16] authors considered trust in two aspects namely identify and behavior. In their work they associate task sensitivity for each task, making highly sensitive tasks being assigned to highly trust-able nodes only and vice versa. Identity trust is cor-related with social layer. SVM is used for the classification of nodes as trustworthy and un-trustworthy. The filtering module guarantees that only nodes that meet the task sensitivity requirements are selected.

F.A. Ghaleb et al. [17] proposed framework to identify malicious nodes using ANN (Artificial Neural Network). The data is gathered by the vehicle's sensors which is later broadcasted by the vehicles and fed to the feature extraction module. The model is then trained on the historical data. The results indicated that the model can generalize better as opposed to the baseline model.

Similarly in [18] a DDoS (Distributed Denial of Service) attack in vehicular networks was simulated. They executed the simulation with added attacking behaviors along with normal traffic and generated a data-set for the training of intelligent models. Multiple models were trained on data-set generated through simulation and all showed great results in identification of DDoS attack traffic. However, their study did not focus on the heterogeneity and attacker diversity in the simulated data-set.

2.3 Summary

We discussed various techniques for trust evaluation and intrusion detection in vehicular networks. We categorised the recent work into two broader classes namely conventional and machine learning based techniques. The literature covers many techniques that identify malicious nodes or compute a trust value, but all those studies have been more focused in general intrusion or misbehavior detection in VANETs, however the special case for trust establishment in an offloading environment in a vehicular network has been less explored. Most of the techniques that discuss task offloading discuss about reducing latency or resource allocations, and the notion of the trustworthiness and reliability of those resources is less considered. We will employ machine learning techniques to treat the problem of trust in task offloading, approaching it as a classification problem, since machine learning based models are still in infancy and conventional techniques face the challenge of how different weights should be quantified to various parameters.

Chapter 3

Problem Statement and Mobility Simulation

In this chapter we will illustrate the problem statement and problem formulation in detail. Equations and theories used for the network entities, their communication modes, computational models, offloading policies and the simulation of different network attacks will be discussed. Moreover, we will discuss in detail the tools and technologies used for the simulation of VANET scenarios. The simulation represents a vehicular mobility model where different vehicles run on different roads and junctions in a Manhattan city network. Each vehicle has a unique identity and a trip (i.e the route that will be travelled by the vehicle during the simulation time). The normal and adversary both traffics with their different behaving patterns are represented in the network simulation.

3.1 Problem Statement

Task offloading is used to enhance user experience in vehicular networks by offloading tasks from overburdened nodes to the idle nodes. However, a trust management system is required to achieve this, since the absence of such system would make it difficult to identify the malicious nodes from the honest nodes within the internal network. The presence of even one adversary node can jeopardize the entire network. Different statistical techniques are used to compute a trust factor, however how much weight-age should be given to different metrics is still a challenging task. Thus, machine learning techniques could be utilised, and this problem can be approached as a classification task to classify between a malicious node or honest node.

3.2 Network Model

Let $V = \{v_1, v_2, \dots, v_n\}$ and $R = \{r_1, r_2, \dots, r_m\}$ in a vehicular network, be the sets that represent vehicles and RSUs, respectively. The RSUs are stationed at various points along the vehicular network where the vehicles in range can reach them. A smart vehicle with OBUs and smart sensors generates the tasks for a variety of applications, including object detection, event notification, and so on. The generated task T is defined as a (S_T, C_T, D_T) representing input size of the task, CPU cycles required for the processing of task and deadline to complete the task respectively. The participating nodes in the network can operate in one of two types of behaviour, as described below:

Honest Behaviour – A node’s honest behaviour is described as behaviour that follows a definite trend throughout the simulation time, such as task generation and local and offloaded task execution. However, realistic circumstances show that a trustworthy vehicle may miss a deadline due to resource restrictions or drop a network packet due to network limitations. For realistic scenario modelling, these behaviours are reflected with a small degree of likelihood.

Malicious Behaviour – Malicious vehicle behavior is characterized as when a vehicle purposefully disturbs the network by missing task deadlines or rejecting tasks received from other vehicles. The goal of malicious nodes is to degrade network performance.

3.3 Communication Model

The overall delay incurred for transmission between connected nodes is referred to as the communication cost. The offloading transmission among the nodes for a vehicle $v \in V$ with a computational task T can be wired or wireless and is defined as follows:

$$B = \begin{cases} X_{v \leftrightarrow r}, & \text{if } v \in V, r \in R \text{ (wireless)} \\ Y_{r_1 \leftrightarrow r_2}, & \text{if } (r_1, r_2) \in R \text{ (wired)} \end{cases} \quad (3.1)$$

where X denotes a wireless offloading transfer to an RSU $r \in R$ and Y denotes a wired offloading transmission between two RSUs. The Shannon Hartley theorem is used to calculate the wireless data offloading rate X , defined below :

$$X = W \log_2 \left(1 + \frac{P}{N_0 W} \right) \quad (3.2)$$

where W denotes the available bandwidth, P is the channel transmission power, and N_0 represents the noise power spectral density. In this study we consider symmetric data transmission rates between vehicle-to-RSU and RSU-to-vehicle.

3.4 Task Computational Model

The computational tasks are generated by all the entities $v \in V$, these tasks can be computed by the vehicle itself using local resources or may be off-loaded to edge nodes based on a certain offloading policy.

Local Execution – The total time spent in completing the computational task by utilising the vehicle’s own local resources is defined as T_{local} i.e. the local execution of the task T , for a vehicle $v \in V$ is defined as :

$$T_{local} = \frac{1}{f_v} (C_T + \sum_{x \in Q} W_x) \quad (3.3)$$

f_v denotes the compute power (i.e. CPU cycles per second) of a vehicle $v \in V$, C_T represents the total number of CPU cycles required by the task for execution, and W_x denotes the total waiting time at local queue for the tasks. Due to the heterogeneous nature of generated tasks and computational capacities of the vehicles, the time it takes to complete a task i.e T_{local} varies among different vehicles.

Task Offloading Policy – The local computational power of a vehicle $v \in V$ may become overburdened with time, hence task offloading can be used to compensate for computational resource scarcity on end devices. When a vehicle is unable to meet the task deadline locally, it decides to offload a task T . The offloading decision denoted by (O_T) for a computational task T is made on the following criteria.

$$O_T = \begin{cases} 1, & \text{if } T_{dest} < T_{src}, dest \in NBR \\ 0, & \text{if } T_{src} \leq T_{dest}, dest \in NBR \end{cases} \quad (3.4)$$

where 1 denotes the offloading of task and 0 indicates the task will be executed locally, source vehicle (src) is defined as the vehicle that needs to offload and destination vehicle (dest) is defined as the vehicle to whom task will be

offloaded and is one of the neighbor of the source vehicle. Each vehicle maintains a list of neighboring vehicles defined as $NBR = \{dest_1, dest_2, \dots, dest_l\}$ where l is the number of vehicles in the neighborhood of source vehicle, the list is updated with time.

Offloaded Execution – The total execution time for an offloaded task $T_{offload}$ is defined as :

$$T_{offload} = \frac{S_T}{B_{mn}} + \frac{1}{f_v} (C_T + \sum_{x \in Q} W_x) \quad (3.5)$$

The first term is the transmission delay for offloading a task from node m to node n , and the second term is the cumulative delay at node n .

3.5 Adversary Model/Attack Model

Throughout the simulation, an honest vehicle maintains a steady pattern, whereas a malicious vehicle’s pattern evolves over time, making it more difficult to distinguish between both. We investigated malicious patterns that a vehicle could represent as a destination vehicle in a task offloading environment, i.e. the vehicle that receives the offloaded tasks. Any misbehaviour on the transmitting side (task generator) during task offloading falls outside the scope of this paper and is not reflected in the simulation. Following is a list of malicious vehicle modes:

Intelligent Cheater (Mode 0) – A malicious vehicle uses this mode to hide its malicious activities, build credibility among other vehicles, and deceive the intelligent layer. When a malicious vehicle enters this mode, it acts like a trustworthy vehicle, completing offloaded tasks and meeting the tasks deadlines intentionally to establish trust.

Black hole attack (Mode 1) – A malicious vehicle in this mode mirrors the black hole attacking pattern, which means it immediately discards the tasks it receives from other vehicles. It processes its own locally generated tasks and discards those received from other vehicles, therefore tasks received from other vehicles have no impact on its computational resources. As a result of its relatively high availability of computational resources, the malicious vehicle tends to attract other vehicles.

Delayed Response (Mode 2) – In this mode, a malicious vehicle intentionally delays responses to offloaded tasks received from other vehicles,

causing deadlines to be missed and the entire network to deteriorate, especially for time-sensitive applications such as accident alerts. The problem is that malicious vehicles advertise their idle resources because they seldom process the assigned offloaded tasks, attracting multiple nodes to offload their tasks in order to receive timely service.

A malicious vehicle enters the simulation in default mode 0 and adapts an arbitrary pattern in which it selects at random from a list of vehicle modes (0,1,2), each with a different probability distribution in the simulation, to determine whether to switch to mode 1 or mode 2, or to remain in default mode 0. Before determining which mode to utilise, a vehicle runs through each mode for 30 seconds. After one cycle, a vehicle returns to its default mode 0 for a 60-second cycle. Multiple malicious behaviours, each represented by an unique vehicle, may exist in a same time frame. A single vehicle, on the other hand, may only have one malicious pattern operational at any given moment and cannot have multiple malicious patterns activated at the same time.

We elaborated in detail the network entities involved in the simulation, how are network is build and how the entities communicate and share information. We discussed about a computational task, what are its execution requirements, how is it executed, when it is executed locally and when it will be offloaded to the edge nodes. Additionally, the communication cost among the participating nodes is also discussed. Moreover, attacking model used in the simulation is also explained in detail with various modes in which the malicious vehicles operate, now we will discuss about the simulation tools used in this work.

3.6 Vehicle Mobility Simulation Tools

We have used **SUMO** simulator [19] for simulating a VANET environment. It is an open source tool that can be used to simulate vehicles and pedestrians along with a network of roads, junctions and edges, reflecting a real life traffic system. **TraCI** (Traffic Control Interface) is connected to the SUMO simulator by using a python [20] script by which we can access and modify the network and its entities.

3.6.1 Converting Manhattan City Map into SUMO Network

To simulate a real life network for this study, we used an Open Street Map (OSM) [21] of Manhattan city, representing different street views of the city. We downloaded the OSM map for Manhattan city through the website as shown in figure 3.1.

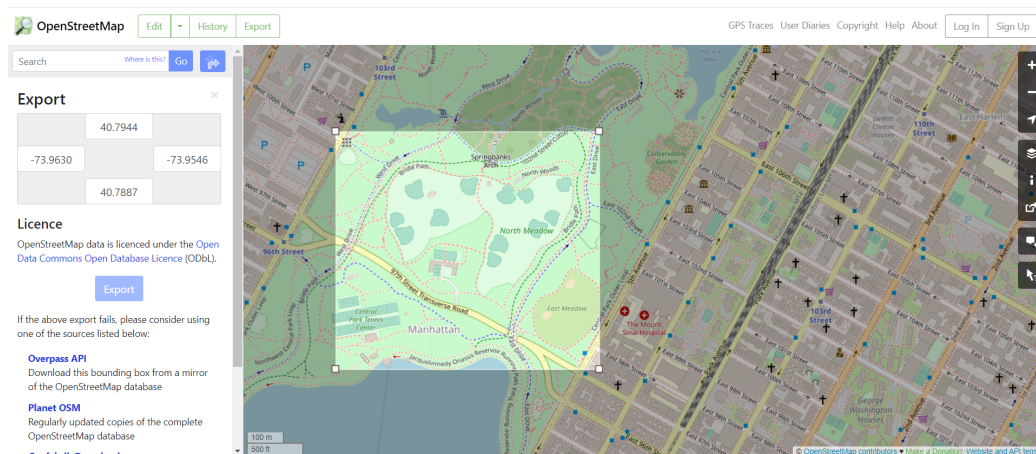


Figure 3.1: A snapshot demonstrating the Manhattan city area used from Open Street Map.

However, to use the Manhattan city map in the simulation, it must be converted to the format supported by SUMO. Thus, the OSM file containing the Manhattan map was converted into a SUMO supported network by using command prompt with the command as described in figure 3.2 that uses a netconverter that comes along with SUMO tools to convert an OSM file into a network file.

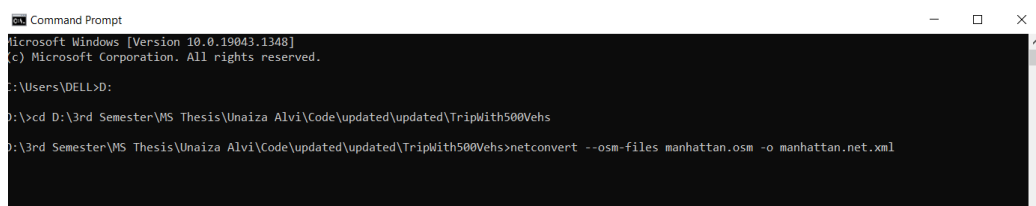


Figure 3.2: Input command for the conversion of OSM map into SUMO network.

The successful execution of the command generates an output file in an XML format representing the Manhattan city network that contains descrip-

3.6.3 Setting Configuration File and Running the network

The successful generation of road network and vehicle routes leads us to the final step of setting up the entire network with a network configuration file. A configuration file is used to setup the entire network by linking the routes and the network files. We have also used an additional re-router file, the purpose of which is to make the vehicle's trips longer in the simulation. Once all things are finalized, the configuration file is loaded in the SUMO simulator and is executed, the output of which is shown in figure 3.7

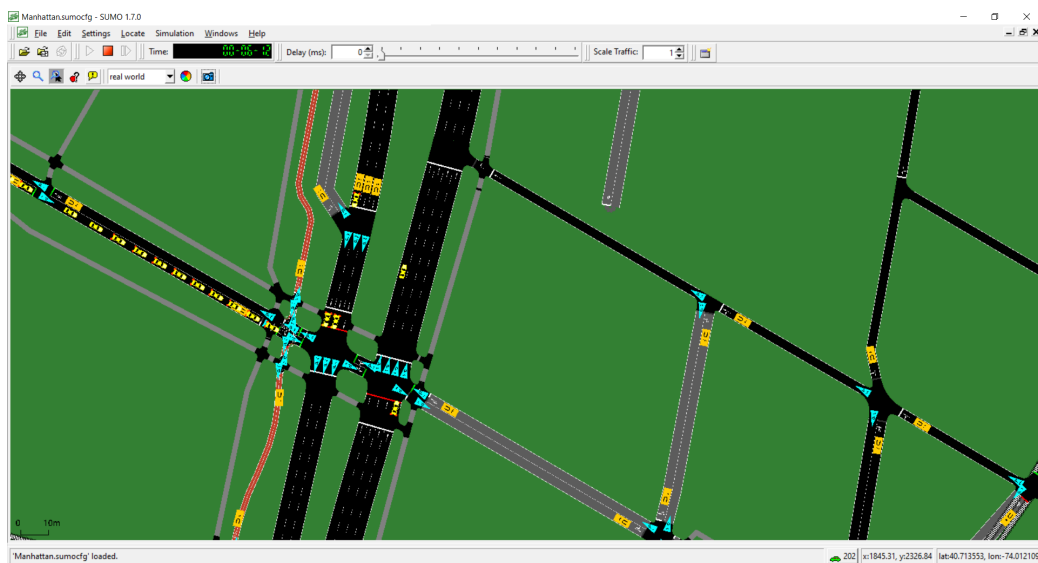


Figure 3.7: The simulated vehicular network with vehicles on the roads.

3.7 Summary

In this chapter, we elaborated in detail the network entities involved in the simulation. The vehicle mobility model is simulated as explained above, once the mobility model is created a python script and TraCI server are used to simulate different network behaviors for different network entities, such as malicious and honest vehicles, different attacking patterns for malicious vehicles, task generation modules, task computation modules, task offloading among the vehicles etc. all such behaviors are simulated using a python script and later on machine learning models are also integrated within the simulation as explained in later chapters

Chapter 4

Proposed Intelligent Multi-criteria Trusted Offloading Decision Framework

In this chapter, we will demonstrate the development of our framework. The vehicle behaviors are simulated in the network with malicious vehicles having multiple malicious patterns as explained in chapter 3. We run the simulation multiple times for the generation of data set which is then used to train multiple machine learning models and finally we will discuss our offloading framework with added intelligent layer.

4.1 Data set Generation

We have used SUMO [19] simulator and a python [20] script to simulate a task offloading scenario in VANETs. The tasks are generated by each vehicle present in the simulation. A computational task T is defined as a three-tuple (S_T, C_T, D_T) consisting input size of the task, CPU cycles required for execution of the task and deadline to complete the task respectively. The offloading decision is made when the vehicle can not meet the task deadline locally. As a result, it decides to offload the task to the vehicle with the least wait time among its neighbours. If no vehicle with a wait time smaller than the source vehicle is found, the task is added to the local queue. We have added different percentages of malicious vehicles with different probability distribution of malicious vehicle modes. The tasks are offloaded based on the criteria mentioned in section 3.4. We have maintained the interaction data for each offloaded task. The data-set features are described below in table 4.1.

Table 4.1: Description of data-set features.

Features	Description
Task_Size	The number of CPU cycles required for the execution of task.
dest_veh_WaitTime	Total wait time at the local queue of the destination vehicle.
Task_Receiving_Time	Time (seconds) at which the result of offloaded computational task is received from the destination vehicle at the source vehicle.
task_deadline:	The task deadline for a computational task.
task_completion_time:	Total time (seconds) in which the computational task was executed.
Task_deadline_achieved	A boolean representing if the deadline for an offloaded computational task was received or not.
Time_Deviated_Deadline	Total time (seconds) deviated from the task deadline.
dest_veh_Tsk_Rcv	Total number of offloaded tasks received by the destination vehicle at a particular time frame.
dest_veh_Tsk_Rcv_Exec	Total number of offloaded tasks executed by the destination vehicle at a particular time frame.
dest_veh_local_computation	Total number of tasks locally executed by the destination vehicle at a particular time frame.
dest_veh_Tsk_Generated	Total number of computational tasks generated by the destination vehicle at a particular time frame.
dest_veh_Tsk_offloaded	Total number of computational tasks offloaded by the destination vehicle at a particular time frame.

Features	Description
Dest_veh_behavior	A label representing the true behavior of destination vehicle at a particular time frame, with value of 1 representing a malicious behavior and value of 0 representing honest behavior.

Training Data-set — In each simulation run, we used varied parameters such as initial seed values, proportion of malicious vehicles, and distribution of malicious vehicle modes (0,1,2) to conduct several simulations for 1000 seconds each. Each simulation’s data is kept in its own csv file (comma separated values). After that, we merged all of the csv files into one file.

Testing Data-set — To ensure unbiased findings, we constructed new data sets with different seed values than those used in the training set, rather than splitting the previously generated data set into train and test splits. Since, changing the seed values entirely affects the distribution of data. We constructed five test sets with malicious vehicle percentages of 10, 20, 30, 40, and 50 %, based on different distributions of malicious vehicle modes.

Pre-Processing of Data — The "*Task_Receiving_Time*" data-set feature describes the time range in which the vehicle receives the result for an offloaded task, based on which we calculate if the task deadline was accomplished or not. The tasks that are offloaded in the last few seconds of the simulation are still in processing, and the value of -1 for "*Task_Receiving_Time*" shows that the task is still in processing, i.e. unprocessed, because the simulation terminates after an assigned period. We excluded the records of unprocessed tasks to simplify our case because no conclusion could be drawn from them.

4.2 Machine Learning Model Training, Selection and Deployment

Training Phase — Different machine learning models such as Logistic Regression, Decision trees, Random Forests , Deep Neural Network and LSTM (Long Short Term Memory), were trained on the training data-set generated through the simulation with the following structure.

Input Features — Task_Size, dest_veh_WaitTime, Task_Offloading_Time,

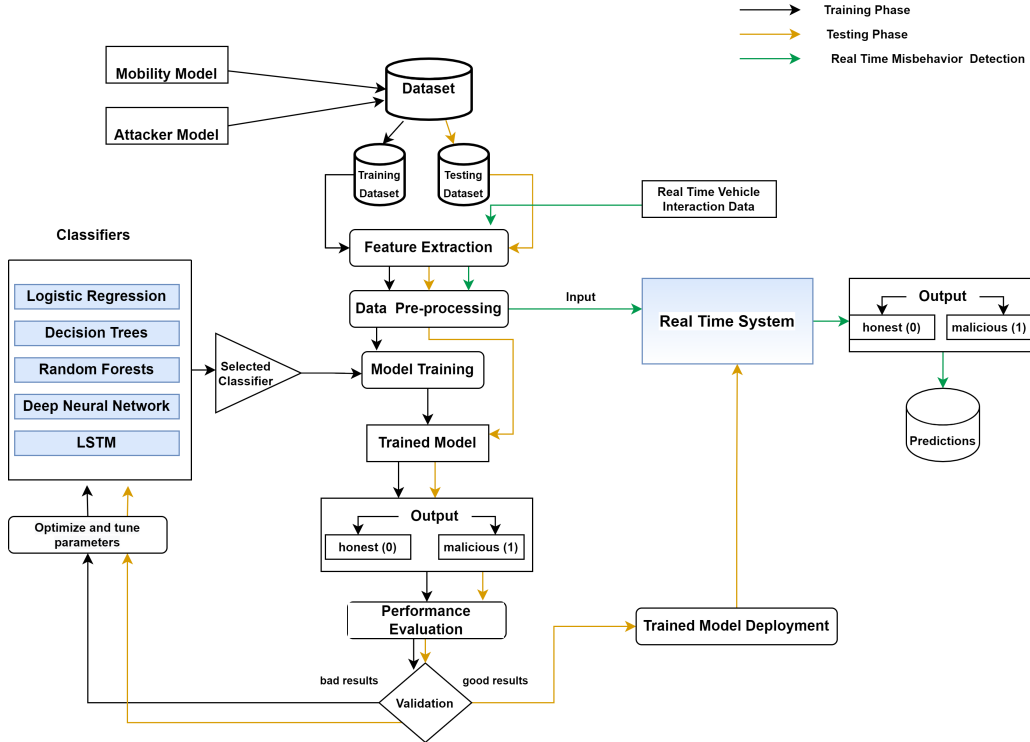


Figure 4.1: The work flow of Machine Learning Framework.

Task_Receiving_Time, task_deadline, task_completion_time, Task_deadline_achieved, Time_Deviated_Deadline, dest_veh_Tsk_Rcv, dest_veh_Tsk_Rcv_Exec, dest_veh_local_computation, dest_veh_Tsk_Generated, dest_veh_Tsk_offloaded.

Target Variable — Dest_veh_behavior

Testing and Selection Phase — The trained models were evaluated on previously unseen testing data set generated through simulation. We measured each model’s performance using a variety of metrics, including recall, precision, F1-score, and F2-score. Because our application is recall critical, the impact of having more false negatives is significantly higher than false positives, i.e., if a malicious vehicle remains undetected, the network suffers more penalties than if an honest vehicle is classified as malicious. We chose LSTM for edge node deployment since it had the highest recall, F2-score, and least False Negatives.

Deployment — The trained machine learning model is deployed on edge nodes (RSUs) to make predictions on the vehicle’s behavior and classify it

as malicious or honest, in real time. The trained machine learning model is fed on the vehicle real time interaction data collected through data collection module to classify the given vehicle as malicious or honest.

The proposed machine learning framework is demonstrated in figure 4.1. The procedure starts with simulating the mobility model and attacking model in the simulator with defined attacking patterns. Once those behaviors are simulated, the simulation is executed, with each simulation run generating a data set, this process is repeated several times. After the generation of two different data-sets i.e training and testing , the next step is to train the models. In testing phase, we perform feature extraction for the data, followed by data pre-processing. The next step is to select the machine learning model, one of the given model is selected. Once the model is trained, the next phase is model testing. The model is tested and evaluated on various metrics, if the results reported are not good, the hyper parameters are fine tuned until satisfactory performance is achieved. After achieving the performance goal, the model is sent to be deployed and integrated in real time system. This process is repeated for all the five models, and the model with the best results is finally deployed in real time system, in our case it is LSTM. The trained model is incorporated in real time system, the data collection module collects the vehicle interaction data, the data is fed to the real time system where it is pre-processed and finally the system gives the prediction on whether the fed vehicle is malicious or honest. As a final step the predictions given by the model are then saved in the central predictions data base.

4.3 Multi Criteria Offloading Framework

In this section, we will explain in detail our offloading framework i.e the criteria for the selection of nodes, while making offloading decisions. We will explain each module of framework in detail.

Cold Start

By feeding the model with real-time vehicle interaction data obtained during simulation, the trained machine learning model is deployed on edge nodes (RSUs) to make intelligent decisions in real time. When making offloading decisions, the vehicles are initially chosen based on the least waiting time. The main goal for integrating the intelligent layer for offloading decisions after the cold start duration is that, at first, the vehicles may be able to complete the generated tasks locally, necessitating less offloading, but as the computational resources become overburdened, offloading becomes necessary, as a result, the vehicles must offload; however, because our architecture is built

on offloading interaction data, we may not need to use the intelligent layer at first because the data is not available. The intelligent layer is initiated after the cold start period has passed. When a vehicle wants to offload a computational task after the cold start period, it queries its nearest edge node to provide trusted candidates list in its vicinity to offload the tasks.

Offloading Framework

The proposed offloading framework is illustrated in Algorithm 4.1, where a vehicle accesses its nearest edge node when it decides to offload a task. It requests a list of trusted neighbours for task offloading from the selected edge node, where a machine learning model is installed. As soon as the interaction data for those nodes is available, the selected edge node scans through the predictions provided by the intelligent layer for each adjacent vehicle through the central predictions database. If there are several trusted candidates, a list of them is returned, from which the one with the shortest wait time is chosen. Hence ensuring that the most reliable and competent vehicle is selected for task offloading.

Algorithm 4.1 Multi Criteria Trusted Offloading Framework

Input:

E: nearest edge node to the source vehicle

T : computational task

Output:

status message

```

1: while true do
2:   if offTime > cold_start then           ▷ Cold start phase is passed
3:     trusCand ← E.getTrustedCandidates(S, S_nbr)
4:     if len(trusCand) > 1 then             ▷ multiple candidates
5:       dest ← leastWaitTimeSelection(trusCand)
6:     else if len(trusCand) > 0 then
7:       dest ← trusCand
8:     end if
9:   else if offTime < cold_start then
10:    dest ← VehicleSelectionWaitTime()
11:  end if
12:  OffloadTask (T, dest)
13: end while

```

Selection of Trusted Vehicle

The RSU takes into account previous vehicle events and searches for intelli-

gent model predictions for each vehicle interaction. Because the algorithms are designed to make predictions for every event, a single vehicle may have multiple predictions. The following selection conditions are used by the edge node to make any decision.

Let $P = \{p_1, p_2, \dots, p_n\}$ be the collection of predictions generated by the intelligent model for each unique interaction for the selected vehicle $v \in V$, with n predictions made by the intelligent model in various time frames. Let α denote the cumulative count for honest predictions and β denote the count of malicious predictions, for the vehicle v , and let λ denote the label selected at the end, with label 1 and label 0 representing malicious and honest vehicles respectively.

$$\lambda = \begin{cases} 0, & \text{if } \alpha > \beta \\ 1, & \text{if } \beta > \alpha \\ 0, & \text{if } \beta = \alpha \end{cases} \quad (4.1)$$

A vehicle is labelled as malicious if the number of malicious predictions for that selected vehicle is higher than the number of honest predictions, and vice versa. In the event of a tie, the vehicle would be regarded as honest if the cumulative count of honest and malicious predictions were equal as demonstrated in equation 4.1.

Rather than taking into account the previously predicted label by the model, our technique ensures that the proper vehicle is selected for each time frame based on its prior interaction data. Because malicious vehicles might occasionally act as honest vehicles, misleading the intelligent layer, we evaluate all of the interactions the vehicle has had and apply machine learning prediction to each of those encounters. Edge node checks the predictions given by the deployed model for each nearby vehicle of the source vehicle (vehicle that chooses to offload tasks). A list of trusted candidate vehicles is maintained by adding the trusted neighbors predicted by the model.

In Algorithm 4.2, the process of trusted node selection is shown. If available, the predicted label of each neighbour of the source vehicle is checked. If there are multiple predictions for a vehicle, the majority voting module is used to choose the final label. With the passage of time, the list of trustworthy and malicious neighbours is updated. This means that if a vehicle was previously predicted to be malicious, but the majority now believes it is honest for a specific period of time, the vehicle's classification will be updated accordingly. This guarantees that the most recent viewpoint is taken into account.

Vehicle selection from Candidates List

The edge node keeps track of trusted neighbours and then chooses the vehicle with the shortest wait time, i.e. the one with the maximum computational power among them.

Algorithm 4.2 Trusted candidates selection at edge node

Input:

S: source vehicle id

S_nbr: a list of source vehicle's neighbors

Output:

trustedCandidates: a list of trusted neighbors

```
1: while true do                                ▷ check predictions for every neighbor
2:   if v in S_nbr then                            ▷ iterate through the complete list
3:     predictions ← getPredictions(v)
4:     if len(predictions) > 1 then                ▷ multiple predictions
5:       label ← getMajorityVoting(v, predictions)
6:     else if len(predictions) > 0 then
7:       label ← predictions
8:     end if
9:     if label == 0 then                            ▷ v is honest
10:      trustedCandidates.append(v)                 ▷ add v in list
11:      if v in S_blacklist then
12:        S_blacklist.remove(v)                     ▷ update reputation
13:      end if
14:      else if label == 1 then                      ▷ v is malicious
15:        S_blacklist.append(v)                     ▷ update reputation
16:      end if
17:    end if
18: end while
19: return trustedCandidates
```

Data Gathering and Predictions Module

The behavioural data collection module is activated once the offloading contact among the connected vehicles begins. The tasks are offloaded from the source vehicle to the destination vehicle, and the vehicle behavioural data module captures the interaction data once the status of the offloaded task is known. The acquired data is run through the entire machine learning framework pipeline, which extracts features and pre-processes the data. After passing through these phases, the vehicle interaction data is supplied to

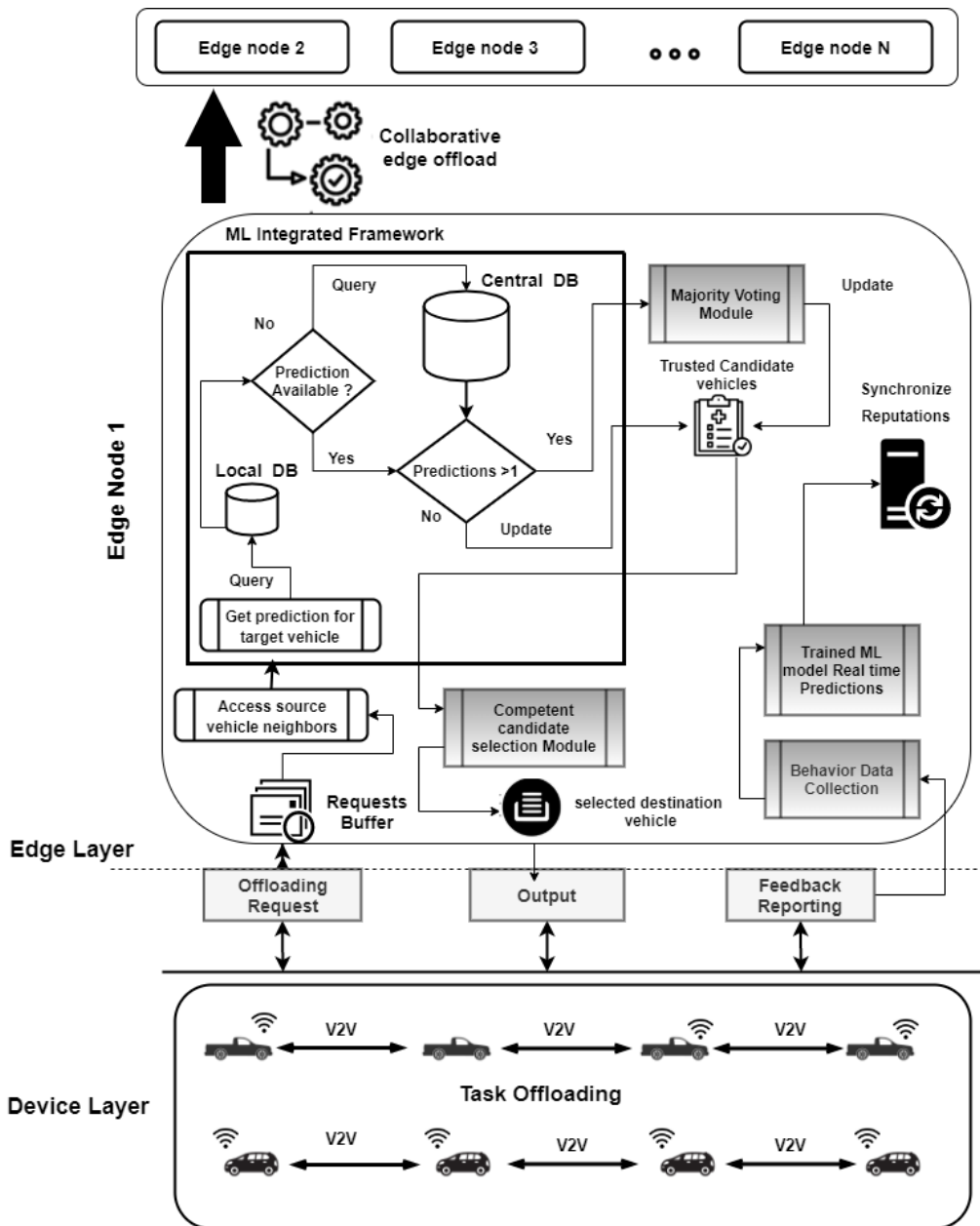


Figure 4.2: Architecture diagram demonstrating the working of proposed framework

the intelligence layer, which classifies the vehicle as malicious or honest. The central predictions database, which is synchronised with all the edge nodes, stores the predicted label for that specific vehicle, which can be accessed by all other edge nodes present in the network. The proposed framework archi-

tecture is illustrated in figure 4.2.

Synchronized Reputation Management

When the cold start timer expires, every vehicle that needs to offload (source vehicle) requests its nearest edge node for a list of trustworthy neighbours. When the request reaches the edge node, it checks if predictions are available in the local data base for each neighbour of the source vehicle, if not, it scans the centralised intelligent layer predictions data base and assigns the label for each neighbour based on majority voting decision criteria as explained in equation 4.1.

Every edge node in the network has the same intelligent layer, which means that every edge node has the same machine learning model. We pass the interaction data to the nearest edge node for each offloaded interaction, allowing the machine learning model to generate a prediction on that specific vehicle. Once the vehicle's label for that time period is determined, it is kept in a centralised database that is synced with all of the edge nodes.

Thus, it is especially important when the source vehicle has not interacted with one of its neighbours, but the distributed reputations available for vehicles can assist the vehicle in making an intelligent decision. As a result, the edge node checks its historical interaction data; the source vehicle may not have interacted with that particular neighbour, but other vehicles may have, and thus it can assist us in making smart offloading decisions. In this stage, the edge node keeps a list of trusted neighbours for any vehicle that asks a trusted neighbour to offload a task.

Chapter 5

Experimental Evaluation, Results and Discussion

This chapter discusses about the experiments performed in detail, demonstrating each experiment and their reported results. The simulation setup and parameters used are defined in table 5.1. We will cover the trained machine learning model results, followed by their deployment in the simulation and finally the network results after incorporation of the intelligent layer.

Table 5.1: Simulation Environment and System Specifications

Parameter	Value
Simulation Time	1000 seconds
Simulation Area	3X3 Km
Simulation runs	5 times
cold start duration	300 seconds
Total vehicles	300
Vehicle speed	1.20 m/s^2
Vehicle Compute Capacity	100-200 MIPS
Compute request size	200-400 MIPS
Task generation	random
CPU	1.3 GHz Intel Core i7
RAM	16GB
OS	Microsoft Windows 10
Simulator	SUMO

The trained machine learning models are tested on 5 different test datasets containing 10, 20, 30, 40, and 50% malicious vehicles respectively. In

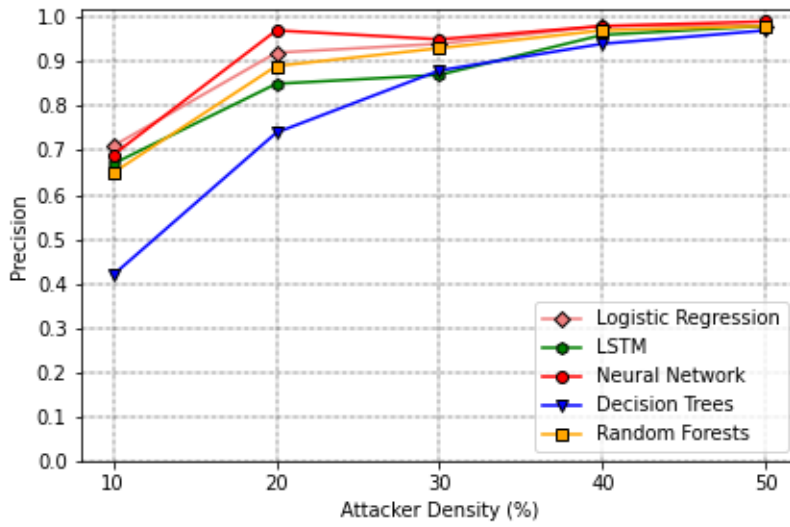


Figure 5.1: The reported precision with varying percentage of attacking vehicles for various models.

figure 5.1 the reported precision for different model at different ratios of attacking vehicles is demonstrated, Neural Network reported highest precision at highest attacker density i.e 50% malicious vehicles and Decision Trees reported lowest precision at lowest attacker density i.e 10% malicious vehicles.

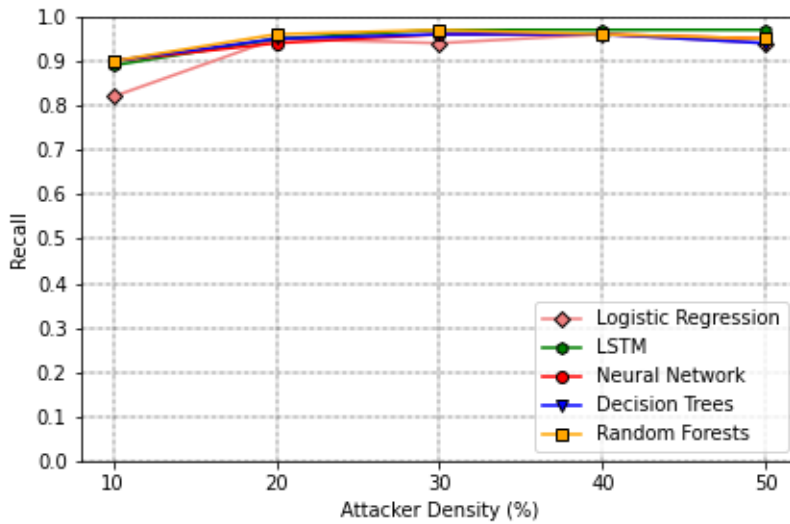


Figure 5.2: The reported recall of various machine learning models as the percentage of attacking vehicle varies.

Similarly, figure 5.2 shows reported recall at varying ratio of attackers in the network.

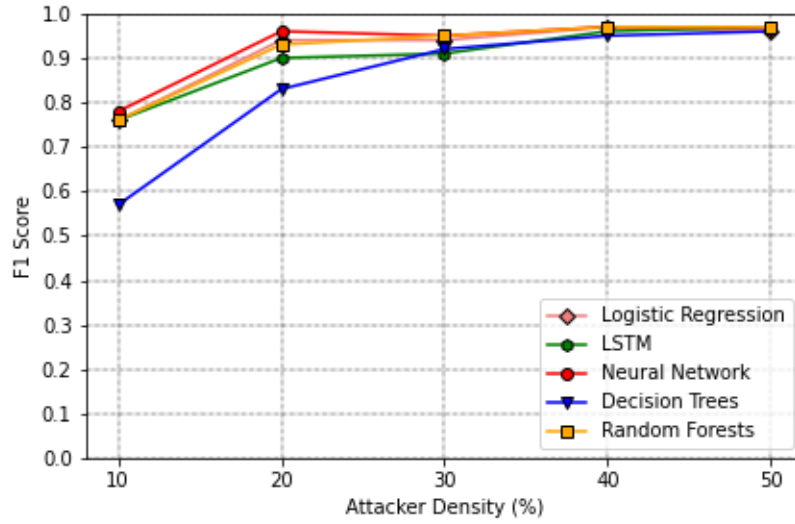


Figure 5.3: F1 score of various machine learning models against different attacking vehicles ratio.

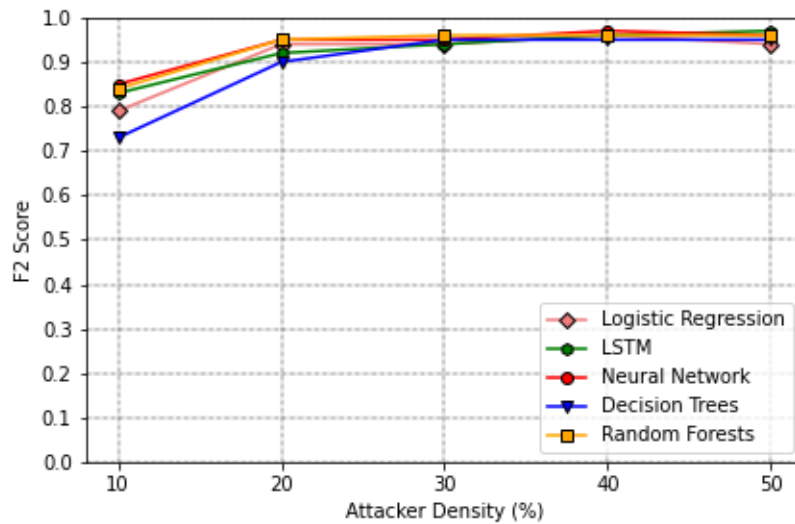


Figure 5.4: The reported F2 score with varying density of attacker nodes for various machine learning models.

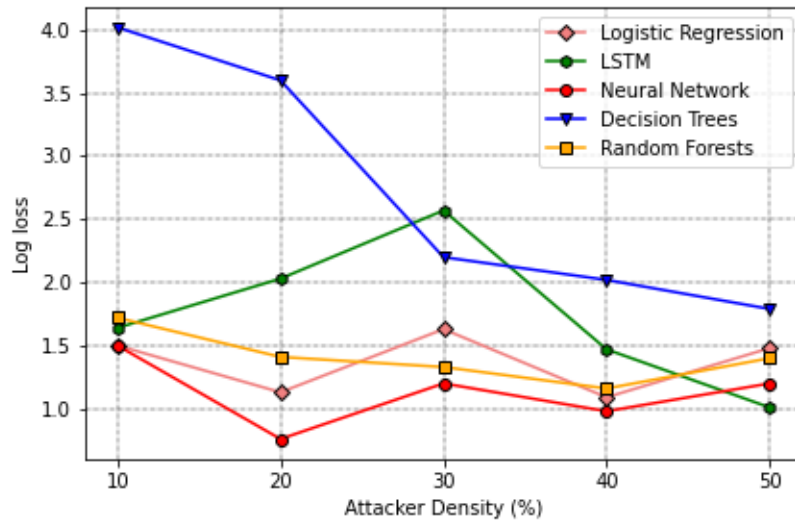


Figure 5.5: A graph representing log loss with varying attacker density for various models.

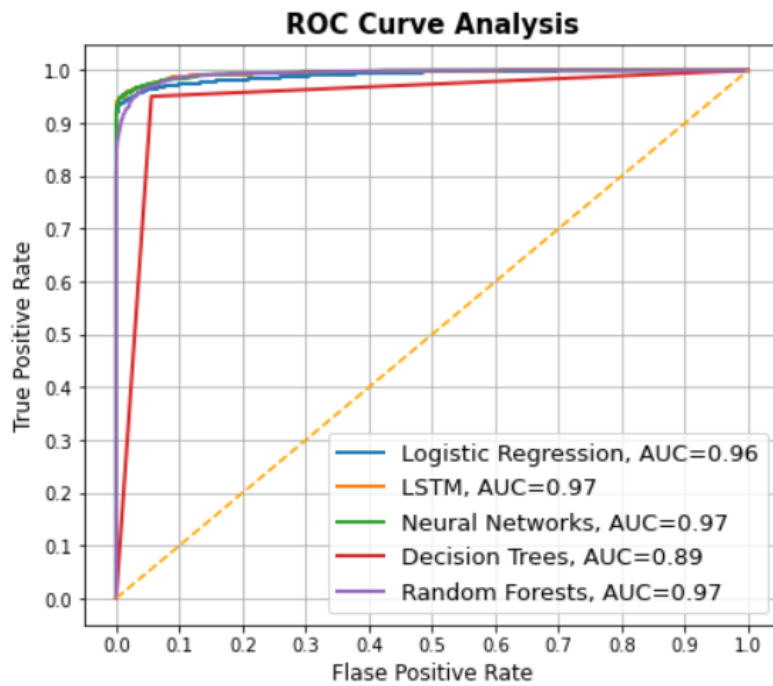


Figure 5.6: The reported ROC for various models at 50% attacker density

LSTM reported highest recall at a attacker density of 50% malicious

nodes, whereas Logistic Regression reported lowest recall at an attacker density of 10%. The figures 5.3 and 5.4 represent the F1-score and F2-score respectively, Neural Network reported highest F1-score on average for varying percentages of malicious vehicles and LSTM reported highest F2-score on average.

Log loss for various machine learning models at a varying ratio of attacker nodes is illustrated in 5.5. Neural Network reported lowest log loss on average taken for all the percentages of malicious vehicles in contrast to log loss and Decision Trees reported highest log loss on average. ROC (Receiver Operating Curve) for various models is presented in 5.6 with 50% malicious nodes in the network. The Neural Network and LSTM show comparable AUC (Area under the curve). The training and testing of various models lead us to the final step of model deployment within the simulation. We choose LSTM for the final deployment, since it yielded least false negatives, highest recall and F2-score. The particular case of identifying malicious vehicles in the network makes our application recall critical, thus employing LSTM in the real time system for the identification of adversary nodes makes it the appropriate choice.

We now demonstrate various network results after the incorporation of intelligent layer in task offloading framework in our simulated network.

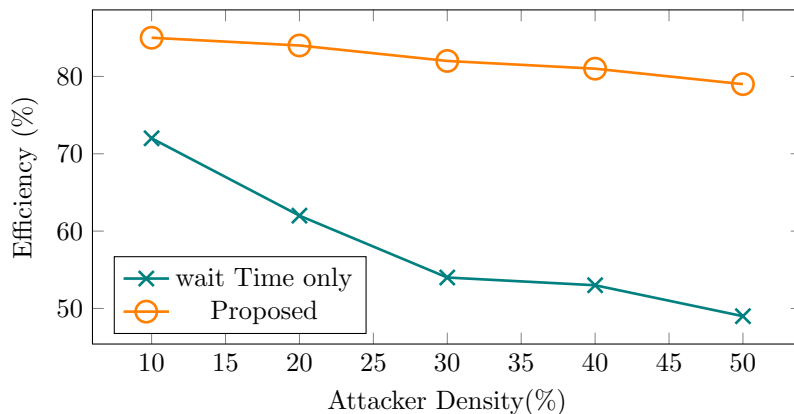


Figure 5.7: Task Efficiency achieved at varying attacker density.

In figure 5.7 task efficiency i.e number of offloaded tasks successfully executed is demonstrated at varying ratios of malicious vehicles ranging from 10% to 50%. The results demonstrate a significant difference between the baseline approach i.e the offloading decisions that are based on vehicle’s wait time only , the one with the least wait time is selected. However, in our proposed approach, once the cold start duration is passed, it is ensured that

an honest vehicle is selected based on the intelligent layer predictions, in case of multiple trusted candidate vehicles, the one with the least wait time is selected, thus ensuring that the most trustworthy and most competent vehicle is selected for task offloading.

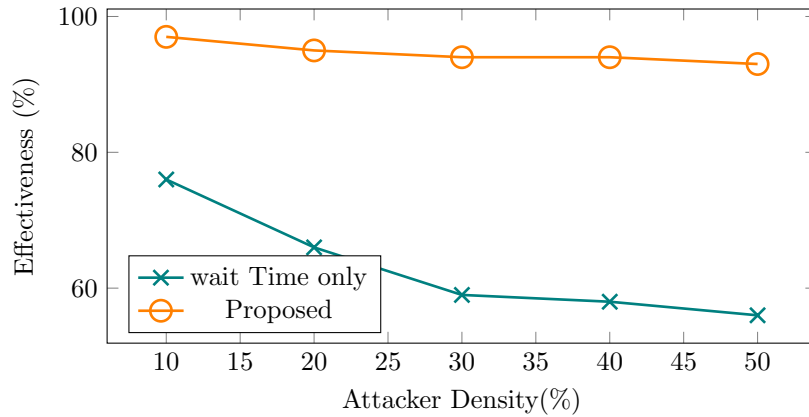


Figure 5.8: Effectiveness at varying attacker density.

Our technique outperformed, the traditional approach even in the worst case scenario i.e at 50% malicious vehicles, where the baseline approach achieved task efficiency rate of 49% only whereas our proposed technique achieved task efficiency rate of 79% i.e 79% of the tasks were executed, even with about half of the malicious resources being filtered, since our technique ensures that no task is offloaded to the malicious vehicle based on the predictions of intelligent layer.

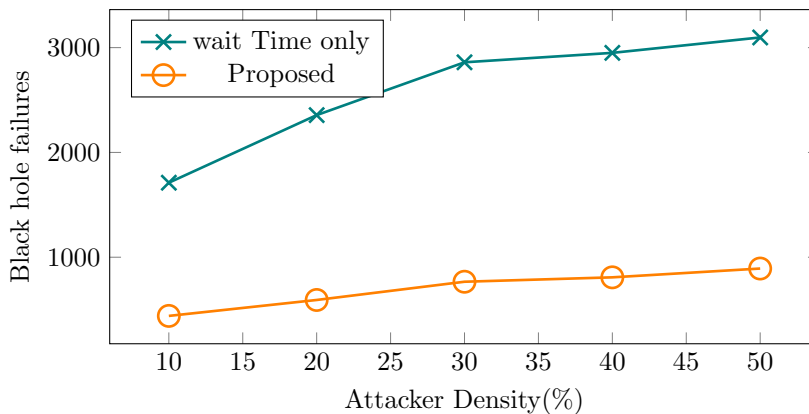


Figure 5.9: Black hole failures at varying attacker density.

Similarly, in figure 5.8, we compare the effectiveness of our approach as compared to the wait time only technique. We demonstrate the effectiveness

of our technique at varying percentages of malicious vehicles ranging from 10% to 50%. As shown, at lowest ratio of 10% for malicious vehicles our proposed technique achieved a high effectiveness of 97% whereas the wait time only approach achieved an effectiveness of 76%. The effectiveness of proposed technique dropped to only 93% in worst case scenario of 50% malicious vehicles whereas the baseline technique could only achieve effectiveness of 56% which is very low as compared to the proposed technique.

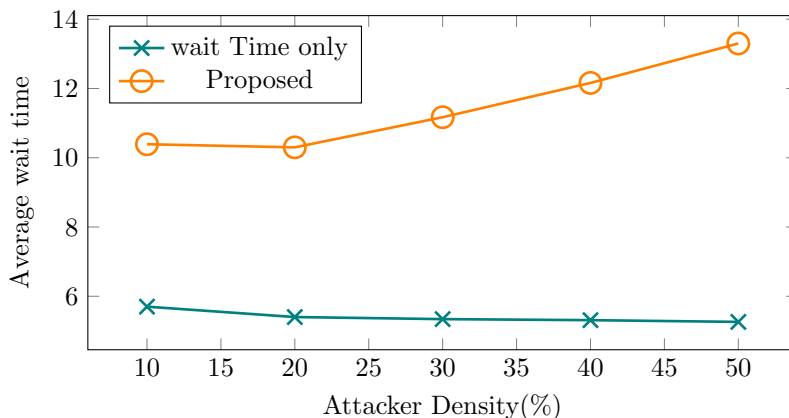


Figure 5.10: Average wait time incurred at varying attacker density.

In figure 5.9 the black hole failures i.e the number of tasks that were simply discarded by the malicious vehicles is compared for the proposed and baseline approach (wait time only) at varying ratios of malicious vehicles ranging from 10% to 50%. The results demonstrate a drastic difference between the baseline and proposed approach. Our technique outperformed the baseline approach and significant difference can be seen at worst case scenario of 50% malicious vehicles.

In figure 5.10 average wait time at varying attacking vehicles is presented. The proposed technique incurred higher wait time as compared to baseline approach. This is because in our proposed technique, we filter the malicious vehicles i.e we ensure that for every task that has to be offloaded only trustworthy vehicle is selected. It means, when 10% malicious nodes are present in the network, it filters out the malicious vehicles i.e tasks are not offloaded to those vehicles. It means the network load is same, however the resources have been filtered out. Let's say we have 100 vehicles and 10% of them are malicious it means, the workload of 100 vehicles will be executed by the remaining 90 vehicles after filtering out the 10% malicious, so on up to having 50% malicious nodes in the network would mean now the workload would

be carried out by 50 vehicles only, after filtering the malicious nodes. Thus, as the number of malicious vehicles increase in the network, the resources would be filtered out, however the network load remains the same, which adds up to the total wait time, thus more wait time as compared to the baseline approach.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

To fully utilise task offloading in a vehicular network an element of trust among the vehicles is essential, the absence of one can jeopardize the entire network. We simulated different attacking behaviors in VANET task offloading environment and generated train and test data-sets. The machine learning techniques were employed to classify the vehicles as honest or malicious. We demonstrated the benefits of having an intelligent layer incorporated in offloading framework which showed significant results in terms of overall network efficiency and task completion rate as opposed to offloading decisions based on wait time only.

6.2 Future Work

We intend to extend this work by adding more malicious patterns particularly in terms of a task generating vehicle (source vehicle). Moreover, we plan to employ game theory and reinforcement learning techniques, i.e the model can learn in an online fashion learning from the environment with time and then making the intelligent decisions.

Bibliography

- [1] U. Alvi, M. A. K. Khattak, B. Shabir, A. W. Malik, and S. R. Muhammad, “A comprehensive study on iot based accident detection systems for smart vehicles,” *IEEE Access*, vol. 8, pp. 122480–122497, 2020.
- [2] M. M. Hamdi, L. Audah, S. A. Rashid, A. H. Mohammed, S. Alani, and A. S. Mustafa, “A review of applications, characteristics and challenges in vehicular ad hoc networks (vanets),” in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–7, IEEE, 2020.
- [3] A. S. Mustafa, M. M. Hamdi, H. F. Mahdi, and M. S. Abood, “Vanet: Towards security issues review,” in *2020 IEEE 5th International Symposium on Telecommunication Technologies (ISTT)*, pp. 151–156, IEEE, 2020.
- [4] F. Sakiz and S. Sen, “A survey of attacks and detection mechanisms on intelligent transportation systems: Vanets and iov,” *Ad Hoc Networks*, vol. 61, pp. 33–50, 2017.
- [5] S. A. Siddiqui, A. Mahmood, Q. Z. Sheng, H. Suzuki, and W. Ni, “A survey of trust management in the internet of vehicles,” *Electronics*, vol. 10, no. 18, p. 2223, 2021.
- [6] H. Sateesh and P. Zavorsky, “State-of-the-art vanet trust models: Challenges and recommendations,” in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0757–0764, IEEE, 2020.
- [7] X. Huang, R. Yu, J. Kang, and Y. Zhang, “Distributed reputation management for secure and efficient vehicular edge computing and networks,” *IEEE Access*, vol. 5, pp. 25408–25420, 2017.

- [8] S. Iqbal, A. W. Malik, A. U. Rahman, and R. M. Noor, "Blockchain-based reputation management for task offloading in micro-level vehicular fog network," *IEEE Access*, vol. 8, pp. 52968–52980, 2020.
- [9] S. Su, Z. Tian, S. Liang, S. Li, S. Du, and N. Guizani, "A reputation management scheme for efficient malicious vehicle identification over 5g networks," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 46–52, 2020.
- [10] F. G. Mármol and G. M. Pérez, "Trip, a trust and reputation infrastructure-based proposal for vehicular ad hoc networks," *Journal of network and computer applications*, vol. 35, no. 3, pp. 934–941, 2012.
- [11] M. Al-Khafajiy, T. Baker, M. Asim, Z. Guo, R. Ranjan, A. Longo, D. Puthal, and M. Taylor, "Comitment: a fog computing trust management approach," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 1–16, 2020.
- [12] R. J. Atwah, P. Flocchini, and A. Nayak, "Towards smart trust management of vanets," in *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–5, IEEE, 2020.
- [13] S. A. Siddiqui, A. Mahmood, W. E. Zhang, and Q. Z. Sheng, "Machine learning based trust model for misbehaviour detection in internet-of-vehicles," in *International Conference on Neural Information Processing*, pp. 512–520, Springer, 2019.
- [14] J. Kamel, M. R. Ansari, J. Petit, A. Kaiser, I. B. Jemaa, and P. Urien, "Simulation framework for misbehavior detection in vehicular networks," *IEEE transactions on vehicular technology*, vol. 69, no. 6, pp. 6631–6643, 2020.
- [15] W. Li, A. Joshi, and T. Finin, "Svm-case: An svm-based context aware security framework for vehicular ad-hoc networks," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, pp. 1–5, IEEE, 2015.
- [16] D. Wu, G. Shen, Z. Huang, Y. Cao, and T. Du, "A trust-aware task offloading framework in mobile edge computing," *IEEE Access*, vol. 7, pp. 150105–150119, 2019.
- [17] F. A. Ghaleb, A. Zainal, M. A. Rassam, and F. Mohammed, "An effective misbehavior detection model using artificial neural network for vehicular ad hoc network applications," in *2017 IEEE Conference on Application, Information and Network Security (AINS)*, pp. 13–18, IEEE, 2017.

- [18] F. A. Alhaidari and A. M. Alrehan, “A simulation work for generating a novel dataset to detect distributed denial of service attacks on vehicular ad hoc network systems,” *International Journal of Distributed Sensor Networks*, vol. 17, no. 3, p. 15501477211000287, 2021.
- [19] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, “Microscopic traffic simulation using sumo,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2575–2582, IEEE, 2018.
- [20] “Python Programming Language.” Available :<https://www.python.org/> . Accessed on: December 15,2020 [Online].
- [21] “Open Street Map.” Available : <https://www.openstreetmap.org/map=16/40.7915/-73.9532>. Accessed on: December 15,2020 [Online].