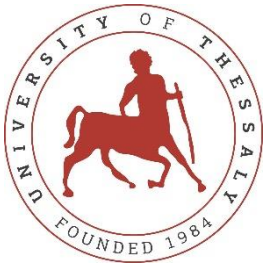


UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Dynamic Parameterization of the Operating Characteristics of Wireless SDN Networks Using RL

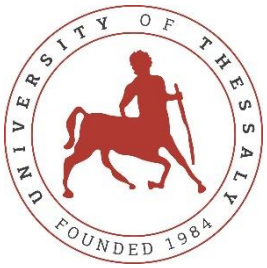
Master Thesis

Prassas Apostolis

Supervisor: Korakis Athanasios

Volos, 2023

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27



UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Dynamic Parameterization of the Operating Characteristics of Wireless SDN Networks Using RL

Master Thesis

Prassas Apostolis

Supervisor: Korakis Athanasios

Volos, 2023

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δυναμική Παραμετροποίηση των Χαρακτηριστικών Λειτουργίας των Ασύρματων SDN Δικτύων με Χρήση RL

Μεταπτυχιακή Διπλωματική Εργασία

Πρασσάς Αποστόλης

Επιβλέπων: Κοράκης Αθανάσιος

Βόλος, 2023

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Approved by the Examination Committee:

Supervisor **Athanasios Korakis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Dimitrios Bargiotas**

Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Argiriou Antonios**

Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελούν αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλουν οποιασδήποτε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχουν έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο/Η Δηλών/ούσα

Ονοματεπώνυμο Φοιτητή/ήτριας

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The Declarant

Name of Student

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Acknowledgements

First of all, I would like express my gratitude to my supervisor professor Thanasis Korakis, for not only imparting knowledge during my academic journey but also for providing me with the invaluable opportunity to collaborate with him in the successful culmination of this thesis.

Furthermore, I extend my appreciation to PhD candidate Ilias Syrigos for his invaluable mentorship, unwavering support, and valuable counsel during the successful completion of this thesis. I am deeply honored and fortunate to have had the opportunity to glean insights from his remarkable expertise.

Equally significant has been the steadfast camaraderie and boundless patience exhibited by my peers and friends over the years. I hold profound gratitude for their enduring support.

Lastly, I wish to convey my heartfelt thanks to my family, whose unique and unwavering backing has consistently propelled me toward achieving my aspirations.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Μεταπτυχιακή Εργασία

Δυναμική Παραμετροποίηση των Χαρακτηριστικών Λειτουργίας των Ασύρματων SDN Δικτύων με Χρήση RL

Πρασσάς Αποστόλης

Περίληψη

Τα Mobile Ad Hoc Networks (MANET) υπόσχονται πολλά για αποκεντρωμένη και αξιόπιστη επικοινωνία σε σενάρια όπως στρατιωτικές επιχειρήσεις ή επιχειρήσεις πυρόσβεσης. Η ενσωμάτωση των δυνατοτήτων δικτύωσης που καθορίζεται από λογισμικό (SDN) στα MANET αντιμετωπίζει την ανάγκη για μια ολοκληρωμένη προοπτική δικτύου για τη διευκόλυνση των δυναμικών προσαρμογών. Αν και η ενσωμάτωση του SDN στα MANET επιβάλλει αξιοσημείωτες προκλήσεις, όπως τα ζητήματα αξιοπιστίας και απρόβλεπτης κινητικότητας, σύμφωνα με τη βιβλιογραφία, αυτά φαίνεται να αντιμετωπίζονται με μηχανισμούς που προσφερουν ανοχή σφαλμάτων οταν ένας κόμβος του δικτύου τεθεί εκτός αυτού, αλλά και μηχανισμούς για την άμεση ανακάλυψη της τοπολογίας. Ωστόσο, τέτοιοι προτεινόμενοι μηχανισμοί οδηγούν σε νέους συμβιβασμούς σχετικά με τη ρύθμιση των παραμέτρων του συστήματος. Στη συνέχεια, η παρούσα διατριβή μελετά και υλοποιεί ένα σχήμα μοντέλων που βασίζεται σε δεδομένα για να παρέχει μια δυναμική και αυτόνομη διαμόρφωση των προαναφερώμενων παραμέτρων του συστήματος. Χρησιμοποιώντας πράκτορες Offline Ενισχυτικής Μάθησης, οι οποίοι εκπαιδεύονται με το σύνολο δεδομένων που προέρχεται από το περιβάλλον του συστήματος, καταφέρνουμε να βελτιώσουμε την επιλογή τιμών των παραμέτρων του συστήματος, δεδομένων των υποκείμενων συνθηκών του δικτύου και της δυναμικής κίνησης των κόμβων.

Λέξεις-κλειδιά:

SDN, MANET, Ανοχή σφαλμάτων, Ανακάλυψη τοπολογίας, Εκμάθηση ενίσχυσης εκτός σύνδεσης, Απώλεια πακέτων, Καθυστέρηση, CORE/EMANE

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Master Thesis

Dynamic Parameterization of the Operating Characteristics of Wireless SDN Networks Using RL

Prassas Apostolis

Abstract

Mobile Ad Hoc Networks (MANETs) hold great promise for decentralized and dependable communication in scenarios like military or firefighting operations. Integrating Software Defined Networking (SDN) capabilities into MANETs addresses the need for a comprehensive network perspective to facilitate dynamic adaptations. Although, the incorporation of SDN into MANETs impose notable challenges, like the reliability and unpredictability issues, according to the literature, seems they to be addressed with mechanisms like fault tolerance and topology discovery. However, such proposed mechanisms, drive to new trade-offs regarding the fine-tuning of the system's parameters. Then, this thesis studies and implements a data driven model scheme to provide a dynamic and autonomous configuration for these system's hyperparameters. By employing Offline RL (RL) agents, which are trained with the dataset coming from the system's environment, we manage to improve system parameters trade-off given the underlying network conditions and the dynamic movement of the nodes.

Keywords:

SDN, MANET, Fault tolerance, Topology discovery, Offline Reinforcement Learning, Packet Loss, Delay, CORE/EMANE,

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Contents

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Table of Figures

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Table of tables

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Abbreviations

API	Application Programming Interface
CORE	Common Open Research Emulator
CQL	Conservative Q Learning
DCNN	Deep Convolutional Neural Network
DDQN	Double Deep Q Network
DQN	Deep Q Network
EEL	Emulation Event Log
EMANE	Extendable Mobile Ad-hoc Network Emulator
FNN	Feed-forward neural networks
GUI	Graphical User Interface
LC	Local Controller
m/s	Meters per Seconds
MAC	Mac (Data Link) Layer
MANET	Mobile Ad-Hoc Network
MC	Master Controller
MDP	Markov Decision Process
ms	Millie seconds
NEM	Network Emulation Module
NRL	Naval Research Laboratory
NS2	Network Simulator 2
OTA	Over the Air
OvS	Open vSwitch
PHY	Physical Layer
PTPd	Precision Time Protocol daemon
ReLU	Rectified Linear Units
RL	Reinforcement Learning
RTT	Round Trip Time
SDN	Software Define Network

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Chapter 1: Introduction

1.1 Motivation

In [1] is presented an innovative type of network architecture which is built based on two conflicting concepts in networking, the Software Define Network (SDN) and Mobile Ad-Hoc Network (MANET) technologies. As described in [1], the cross layer SDN architecture for MANETs provides a centralized control, via the SDN controller, over the distributed communication protocol that MANETs operate. While this architecture encompasses fundamental principles such as decentralization and trustworthiness, it encounters limitations and trade-offs. Upon thorough examination of this research, it becomes apparent that both the topology detection and fault tolerance mechanism components impose costs on the network, whether in terms of losses or overhead. Thus, the objective of this study is to mitigate these shortcomings.

To address these weaknesses, a Reinforcement Learning (RL) approach is employed. A Deep Q Network (DQN) model is developed and trained to autonomously and adaptively adjust the crucial parameters associated with topology discovery and reliability, namely heartbeat interval, election timeout and HELLO INTERVAL. These parameters are essential for maintaining a robust and efficient network. By leveraging the capabilities of RL, the proposed model aims to optimize these parameters dynamically, thereby enhancing the network's performance.

1.2 Contribution

This thesis encompasses several significant contributions, which are summarized as follows:

1. Comprehensive Trade-Off Analysis: We conducted an extensive examination of trade-offs associated with various system parameters. Our objective was to assess the performance impact of these parameters and gain insights into their interdependencies. Through meticulous analysis, we investigated how different configurations influenced system behavior and performance.
2. RL-based Parameter Configuration: In order to automate the process of system parameter configuration, we developed and deployed RL agents. These intelligent agents leverage network monitoring data collected from nodes in the SDN-MANET network. By observing and learning from this data, the agents are able to autonomously adjust system parameters in real-time, optimizing performance based on dynamic network conditions.
3. Prototype Evaluation in Realistic Scenarios: To evaluate the practicality and effectiveness of our approach, we implemented a prototype system and conducted evaluations in realistic scenarios. By subjecting our prototype to these scenarios, we were able to assess its performance, robustness, and adaptability in dynamic and challenging network conditions.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

1.3 Structure of this document

The rest of this thesis is organized as follows:

Chapter 2: introduces an intuitive technical background regarding the SDN and MANET technology. In particular, describes the specific SDN architecture which is built above MANET [1]. Then, a basic reference on RL theory follows.

Chapter 3: presents the software tools, such as the simulator and Machine Learning libraries, that used for the completion of this Thesis.

Chapter 4: includes a comprehensive description of the system's design architecture, as well as the implementation details.

Chapter 5: the evaluation of the implemented system is delivered into this chapter.

Chapter 6: provides the conclusion to this Thesis and thoughts for future work.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Chapter 2: Technical Background

2.1 SDN & MANET

SDN and MANETs are two distinct networking paradigms that serve different purposes and exhibit contrasting characteristics.

In response to the growing complexity and demand for networks, Software-Defined Networking (SDN) emerged to address the need for network management that is both flexible and efficient. SDN is founded on the principle of separating network control from forwarding. This division promotes a network that is more agile and effective compared to traditional architectures, where these layers are tightly interconnected, leading to less adaptable systems where changes can have far-reaching effects. SDN not only streamlines network management and provisioning, but it also enhances them by integrating advanced functionalities like automation. SDN environments offer the ability to automate numerous network operations, resulting in significant reductions in time and resources required for managing the network

A Mobile Ad hoc Network (MANET) refers to a cluster of mobile (or temporarily static) nodes that aim to facilitate the transmission of voice, data, and video between arbitrary pairs of nodes. This is achieved by utilizing other nodes as relays, eliminating the need for infrastructure. Within an actual mesh network, radios have the ability to join or depart from the network at any given time, prompting continuous adaptation of the network's topology as nodes move around. This implies a decentralized architectural design where there is no necessity for central "master" radio nodes to govern network control, ensuring that communication remains intact even in the event of one or more node losses.

While SDN focuses on providing centralized control and network management the MANET's target is to enable reliable and efficient communication in a distributed way. However, there can be areas of overlap and potential integration between SDN and MANETs. Towards this end, a comprehensive SDN framework designed specifically for MANETs is introduced and assessed in [1]. The framework aims to combine the advantages of distributed MANET protocols and centralized SDN approaches, while addressing their limitations.

This thesis, which is based on the SDN framework that is described in [1], focuses on providing the ability for dynamic and autonomous configuration of key parameters for this framework. These key parameters are related to the Fault Tolerance and Topology Discovery aspects and are described in the following subsections along with the overall design description of this framework.

2.1.1 SDN Architecture for 802.11 MANETs Overview

An overview of the fault-tolerant SDN framework designed for MANETs is provided in Figure 1. In this framework, each node in the MANET network operates an SDN Local Controller (LC) application alongside a virtual switch based on Open vSwitch (OVS) [2]. The control plane, which connects the controller(s) and switches, can be implemented in two ways: the out-of-band control plane or the in-band control plane. The former scheme separates the control plane traffic from the data plane traffic to be

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

traversed in different network infrastructures. That is, out of band provides isolation between the two planes which can improve the performance, however it requests additional network resources. Alternatively, with the in-band control plane, data and control planes share the same network infrastructure keeping simple the network design and not using many resources. The in-band control plane was chosen as it allows complete network control in an SDN manner and works with devices having a single wireless interface.

To enhance the reliability and fault tolerance of the system, the Raft Consensus algorithm [3] is used as a distributed protocol. This algorithm facilitates the coordination of communication between SDN LCs and enables the election of a Master Controller (MC). The MC plays a crucial role in managing the SDN network within a MANET cluster. It is responsible for applying flow rules to virtual switches using the OpenFlow protocol [4], which is the standard communication protocol between controllers and switches.

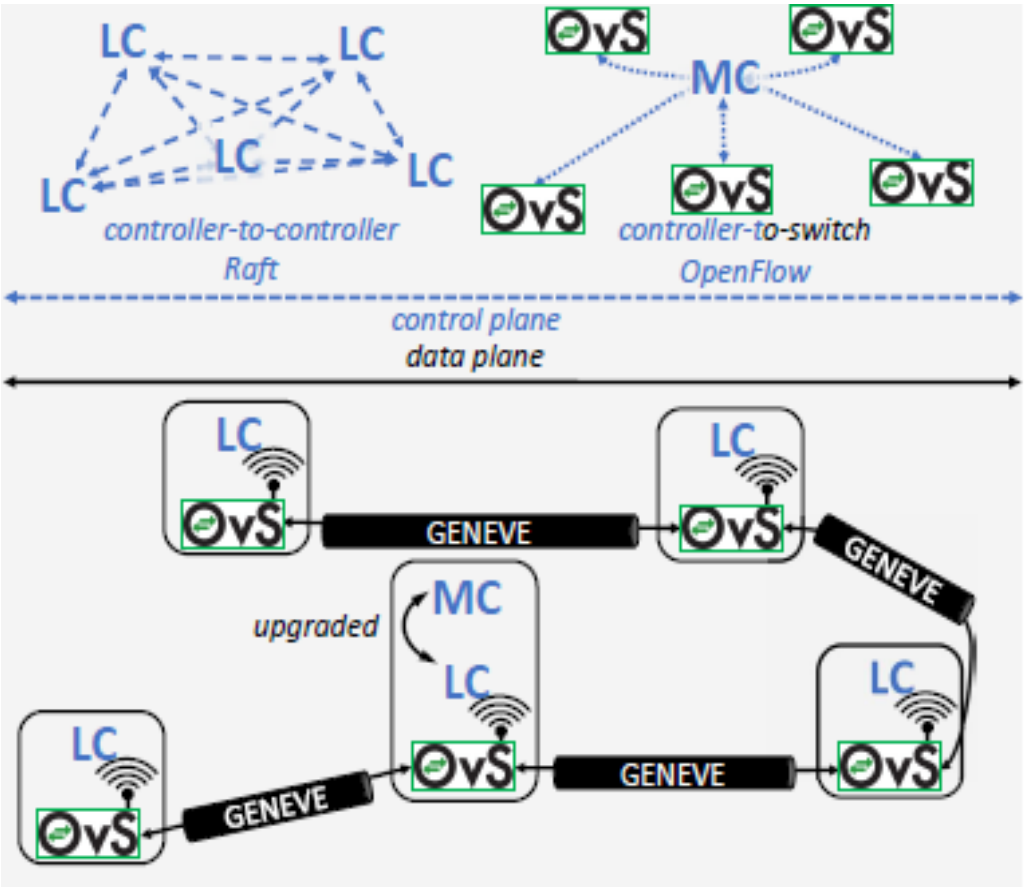


Figure 1: SDN-MANET architecture [1].

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

2.1.2 Fault Tolerance

The SDN controller is the central brain of a software-defined network. This makes it a critical component of the network, and also a potential weak point. If the controller fails or becomes unavailable, the entire network can go down. This means that the network will not be able to route traffic, adapt to changes in the environment, or maintain centralized control.

To overcome this challenge, the authors in [1] developed a fault-tolerant SDN network where each switch has the capability to act as a controller when needed. Each switch in the network runs a Ryu [5] controller process alongside OvS that connects to all other controllers/switches in the network. By leveraging OpenFlow's capability to assign different roles to controllers, the conflicts caused by concurrent flow entry installations in switches and effectively coordinate multiple controllers are minimized. According to the OpenFlow, a controller can perform in one of the following roles:

- *Equal*: This is the default role where the switch is fully under the control of the controller and can receive switch-generated asynchronous messages. It can send and modify controller-to-switch commands.
- *Slave*: A Slave controller has read-only access to the switch and cannot receive asynchronous switch messages. It is restricted from issuing controller-to-switch commands.
- *Master*: The Master controller can control the switch fully, both reading from it and writing to it. Only one controller can be the lead at any time. When a controller becomes the lead, the roles of all other controllers automatically change to Slave.

In this way, each switch operates with an LC process, but only one switch can assume the role of the MC at any given time. This MC is responsible for controlling and applying flow rules to the other switches, ensuring the centralized nature of SDN. The MC is designated as the Master in the OvS instances, while the remaining switches are configured as Slaves, except for collocated switches, which are configured as Equals.

This approach eliminates the vulnerability of a single point of failure, while enhances the robustness and resilience of the network. To determine the MC, the Raft election scheme is used to automatically declare the leader node in the Raft cluster as the MC of the SDN network.

The behavior of the SDN network's controller-to-controller communication is controlled by two parameters: the *heartbeat interval* and the *election timeout*. According to [2], the MC maintains its leadership by periodically sending heartbeat messages to all followers. If a follower does not receive any communication, including heartbeats, for a certain duration called the election timeout, it assumes that there is no MC and starts a new election as a candidate. During an election, the candidate increments its term count, votes for itself, and sends RequestVote messages to all other servers. The candidate repeats this process until it either wins the election or another server becomes the MC. In some cases, no candidate wins the election for a particular period of time due to multiple nodes having the same number of votes.

As it emerges from the above, these two parameters are crucial for the performance of the network. In scenarios with high node mobility, where topology changes are frequent, it is evident that the heartbeat interval as well as the election timeout should be short to promptly detect topology changes. This results in a large number of exchanged heartbeat messages among the nodes, thus increasing the network overhead. On the other hand, when there are no random patterns on the movement or very low node mobility, having long intervals for heartbeat messages exchange, the network benefits by the low overhead. Thus, a trade-

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

off arises where the implementation of this thesis is called to solve and, consequently, improve the performance of the architecture in [1].

2.1.3 Topology Discovery

In MANET, the nodes must know their one hop distance nodes, so they are able to communicate with every other node, even those that are more hops away. Such a case is the described network architecture in [1], where the communication protocol is the 802.11 in adhoc mode and thus the nodes can reach only their neighboring nodes. However, the SDN is integrated, which offers a centralized way of management and control of the network. The MC should have the whole picture for the topology of the nodes in the network and be able to select the routes for the communication between them. To bridge these two opposing approaches, the authors in [1] implemented the discovery topology based on the exchange of two special messages, ECHO_MESSAGES and TOPOLOGY_MESSAGES.

The ECHO_MESSAGES are propagated through each node across the network topology and their existence has a double role. Initially, these messages serve to discover links between neighboring (one hop distance) nodes. In parallel, each neighboring node that receives such an ECHO_MESSAGE forwards it to its own neighborhood (one hop distance neighbors) and in this way these messages reach all the nodes of the topology. Consequently, this procedure has the effect that each originator node of ECHO_MESSAGES discovers routes to nodes that are more hops away from it. Apparently, this scheme of message exchanging deals with the distributed nature of the MANET. The TOPOLOGY_MESSAGES are about to cover the requirements of the SDN logic. More specifically, each node which has connection with the MC, will send him a TOPOLOGY_MESSAGE to inform him which are its neighbors. Therefore, the MC is in position to aggregate all these messages it receives by each node, and then to create a complete graph that depicts the network topology. In consequence, the MC supports the nodes routing part for their communication.

The ECHO_MESSAGES include three fields, the originator's node id, the transmit quality of the route and some extra flags. Similarly, the TOPOLOGY_MESSAGES, include the originator's node id, the originator's discovered links, which are the links with their one hop neighbors and some extra flags. Both of them are encapsulated into the specific LLDP packets and are transmitted with the same frequency, which is the HELLO_INTERVAL. As follows, the immediate responsiveness of the MC regarding the topology status of each node is dependent of the HELLO_INTERVAL value. The shorter this interval will be the sooner the MC be aware. In the opposite, the longer this period is, the less overhead there is in the network. In the same vein with the aforementioned etcd parameters, heartbeat and election_timeout intervals, this thesis's proposition will try to medicate this situation and provide a solution which leads to the most efficient HELLO_INTERVAL tuning for the networks favor.

2.1.4 Tradeoff Analysis

Focusing on the aforementioned dilemma about the proper adjustment of the specific variables regarding the fault tolerance and the topology discovery mechanisms, we investigate how these variables affect the network's behavior. The below graphs illustrate the tradeoffs and metrics related to fault tolerance and topology discovery mechanisms in three different scenarios: no mobility topology, mobility with a velocity of up to 5m/s, and mobility with a velocity of up to 10m/s. These scenarios were designed to provide a comprehensive understanding of the network's behavior under various conditions.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

In more detail, a predefined set of values in seconds was set, which is {0.5, 0.9, 2, 4}. Considering these values as representative values for the HELLO_INTERVAL, for each one of them we conducted experiments for all the mobility scenarios. The etcd parameters values were static defined in all of these examination tests, heartbeat_interval = 1000 ms and election_timeout = 10000 ms, so to better understand how the HELLO_INTERVAL different values affect the network. Also, we conducted multiple times each experiment to ensure the reproducibility of the results. During each experiment the network's total packet loss as well as the total delay were measured. Therefore, as depicted in Figure 2, resulting that the sooner the HELLO_INTERVAL period is, the smaller the packet loss rate is. On the other hand, for the delay metric, the reverse applies, as can be seen also in Figure 2. That is, as more frequently packets are transmitted the delay is increasing. This is subsequent of the delay's correlation with the network's overhead. The more the transmissions are, the larger the network's overhead and thus the delay. Also, recall that the used MAC protocol in our experiments is 802.11, which includes the backoff mechanism.

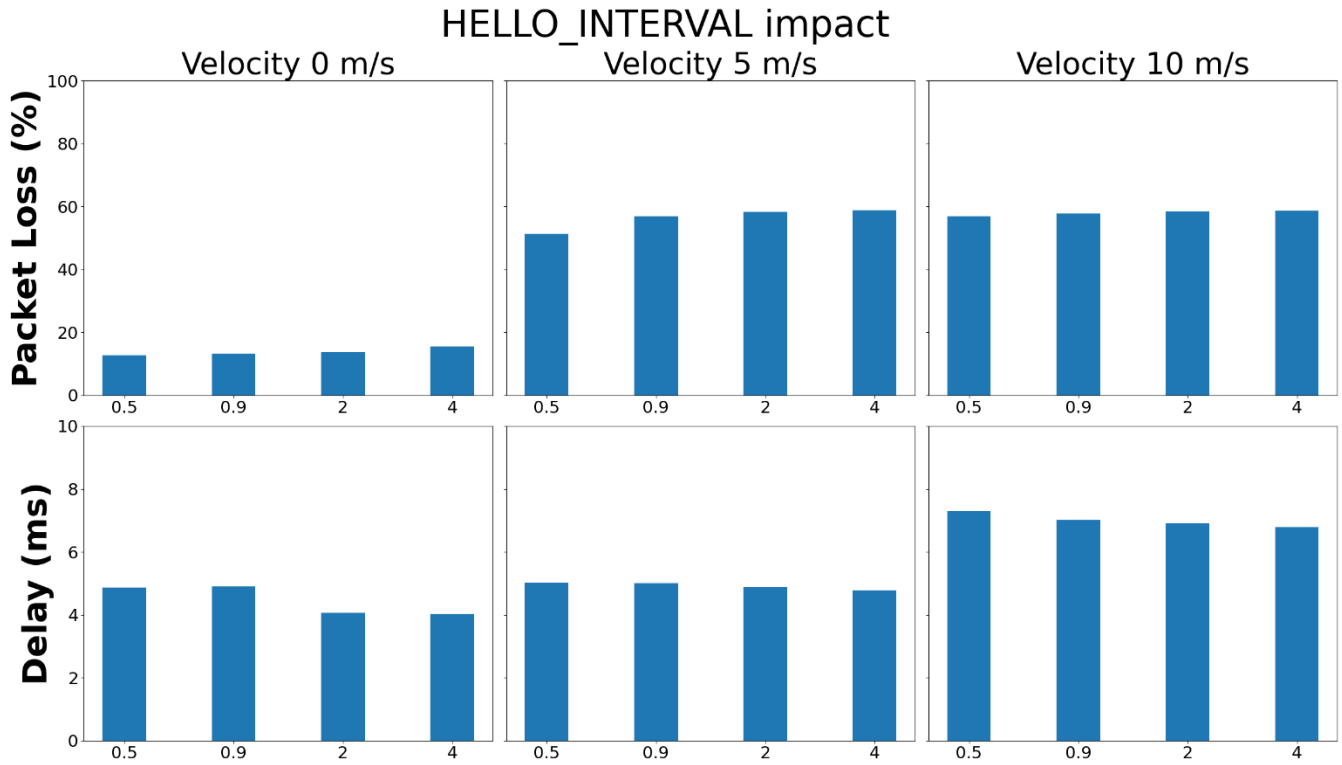


Figure 2: HELLO_INTERVAL impact on the network's behavior.

The examination of the etcd parameters, on how they affect the network's behavior, followed the same procedure. Once again, a set with symbolic values was defined following the order (heartbeat_interval - election_timeout), which is {(100 - 1000), (500 - 5000), (1000 - 10000), (4000 - 40000)} in ms. In smililar way, this time HELLO_INTERVAL was defined as static, with the value of 2 seconds. As it is demonstrated in Figure 3, the subplots follow the same patterns with that in Figure 2. That means, for the

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

heartbeat_interval and the election_timeout variables, the shorter the interval is, the smaller the total packet loss rate for the network results. On the contrary, the delay is decreasing when these parameters are set with larger timeouts.

Noteworthy, in both Figure 2 and Figure 3, is the subplot regarding the packet loss rate when the mobility scenario has velocity zero. Due to the lack of mobility, it is expected the packet loss rate to be zero. However, these measurements include also the bootstrap phase of the system. That is, the elections regarding the MC take place, and during this period the lost packet are taken into account. Further, it is evident in both figures, that as the velocity is increasing the ranges for the delay values and the packet loss rate values are increasing, too.

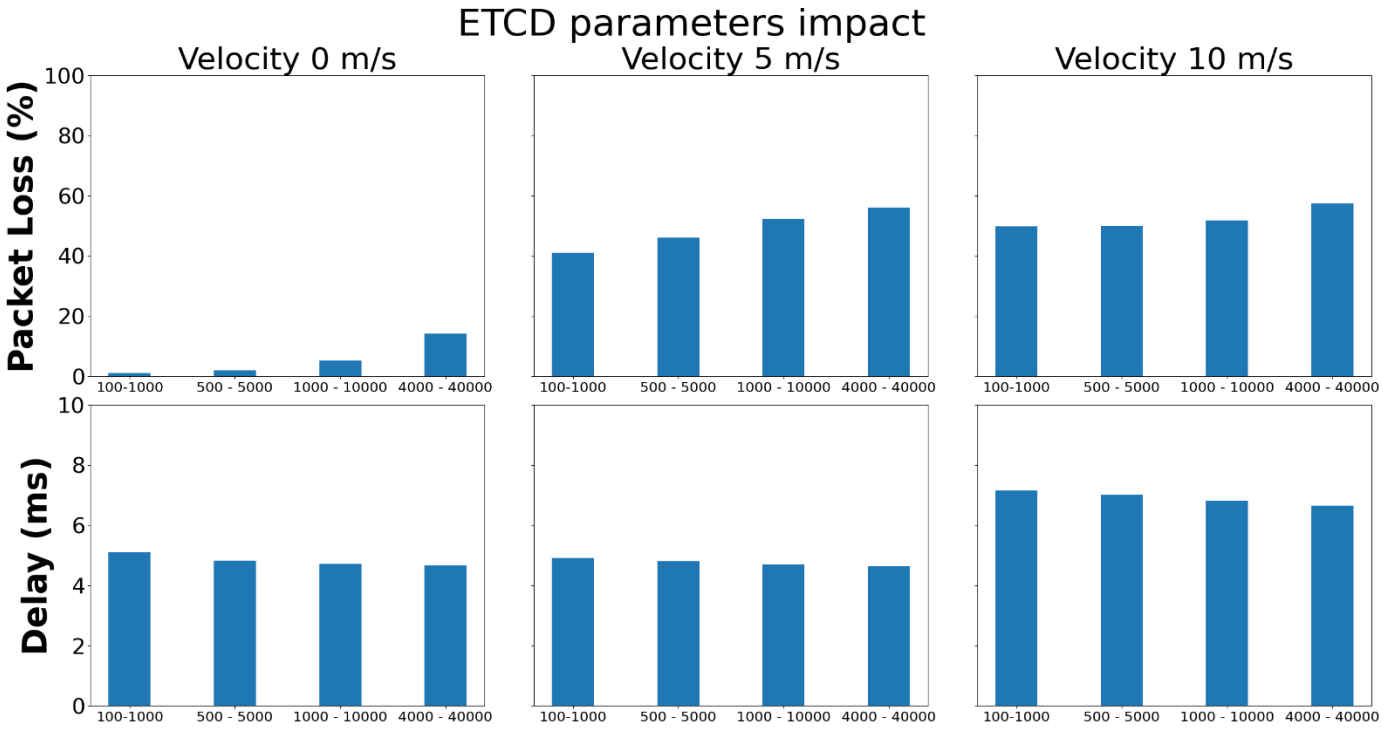


Figure 3: Etcd parameters (heartbeat_interval, election_timeout) impact on the network's behavior.

In a nutshell, the plots in Figure 2 and Figure 3 show in practice that these parameters HELLO_INTERVAL, heartbeat_interval and election_timeout, depending on their values, significantly affect the behavior of the network. Afterwards, we focused on the metrics of the packet loss rate and the total delay of the network, metrics that, as described later on this document, were used to define the reward function.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

2.2 Reinforcement Learning

RL is a machine learning approach that automates tasks by defining a goal and rules for achieving it. RL differs from other methods by having the agent learns from direct feedback of its actions in the environment [6]. In contrast to supervised learning, RL does not require labeled data, making it useful in unexplored or data-scarce environments. However, hybrid approaches and offline RL are gaining interest as they have shown potential in solving various problems.

2.2.1 Markov Decision Process

To understand RL, it is essential to comprehend the underlying mathematics of Markov Decision Process (MDP), especially when the MDP has finite state and action spaces (finite MDP). According to Sutton et al. [6], a vast part of modern RL can be grasped simply by understanding finite MDPs. MDP requires the Markov property, where the future state only relies on the current state and action. MDP is significant because it helps to predict the expected next reward and the following state. The state-transition probability can be calculated using the probability of transitioning to the next state given the present state and action.

$$p(s', r|s, a) = \Pr\{St+1 = s', Rt+1 = r \mid St = s, At = a\} \quad (1)$$

The fully expressed dynamics of finite MDPs enable computation of several interactions between the agent and the environment. The agent can take actions that affect the environment, and the environment can transition to different states based on the agent's actions. The state-transition probability is a measure of how likely the agent is to transition to a particular state after taking a specific action in a given state:

$$p(s'|s, a) = \Pr\{St+1 = s' \mid St = s, At = a\} \quad (2)$$

An MDP is represented as a 5-tuple (S, A, P, R, γ) where:

- S – a finite set of states
- A – a finite set of actions
- P – a set of state-transition probabilities
- $R_a(s, s')$ – the reward that the agent receives immediately after taking an action and transitioning to a new state s' through action a

To make it clear, a group of values known as states depict the agent's situation, and a group of values known as actions explain the agent's activities. The discount factor is a value between zero and one that determines the importance of future rewards in the present.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

2.2.2 Elements of RL

According to [6], the key constituents of RL include the agent, environment, policy, reward function, value function, and model. These fundamental components form the basis of RL and will be elaborated further in the below subsequent sections.

2.2.2.1 Agent and Environment

In RL, the primary components are the agent and the environment. The agent is the learner and the decision-maker, and it interacts with everything outside of it, which is referred to as the environment. The agent's current situation is described by two terms: state and action. A state provides information about the agent's location in the environment, while an action denotes the agent's intended action. The agent and the environment are linked through their interactions; the agent selects an action, and the environment responds by producing a new state. This interplay among the agent and the environment takes place at every point in time, which is a sequence of discrete time steps denoted by t = 0, 1, 2, 3, and so on, as noted in [6]. While time steps can be continuous, they necessitate more intricate how calculations are different when using discontinuous time steps, as stated by Sutton et al.

Sutton et al. created Figure 4 to demonstrate the relationship between the agent and environment in RL. Initially, at time step t, the agent selects an action At based on the initial state St and reward Rt. The environment responds by offering a new state St+1 and reward St+1, and the process continues. The agent aims to maximize rewards that the environment generates over time. Although the boundary between agent and environment is occasionally unclear, it is generally accepted that anything outside of the agent's control is considered part of its environment.

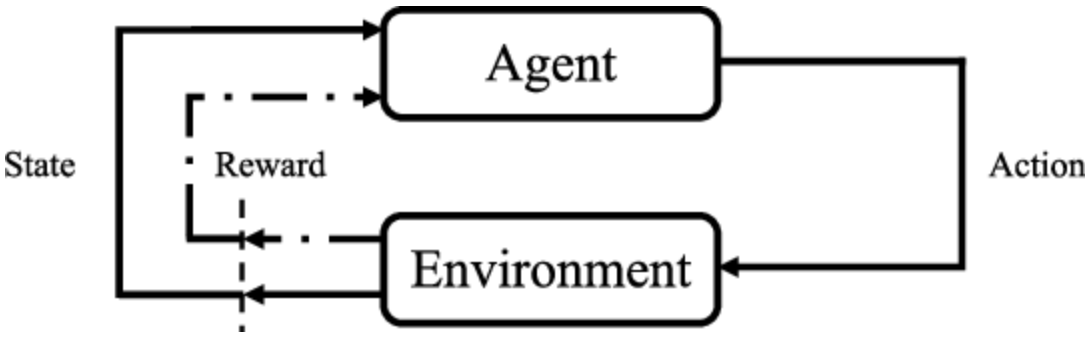


Figure 4: Agent and environment interaction visualized

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

2.2.2.2 Policy

The agent's plan, called policy, determines how it behaves in different situations. It creates a connection between each situation and the action the agent takes, along with the chance of selecting a particular action. The main aim in solving a RL task is to find the best policy that gives the agent the most rewards over time. We call this the optimal policy. Depending on how the agent achieves this optimal policy, RL methods can be grouped into two categories: on-policy and off-policy. On-policy methods involve the agent learning by watching the rewards it gets while following a specific plan. On the other hand, off-policy methods involve the agent learning from one plan (behavior policy) while observing the rewards it gets from following a different plan (target policy).

2.2.2.3 Reward function

In RL, rewards represent numerical values that agents receive during training, offering insight into their performance and guiding the learning process. The primary aim is to optimize cumulative rewards throughout the training, achieved through a well-designed reward function that encourages desired behaviors. Rewards also influence strategy adjustments, enabling agents to adapt future actions based on their understanding of optimal choices. The learning process may involve episodes, discrete segments within the agent-environment interaction, starting from a default state and ending at a terminal state or predefined stopping point [6].

The anticipated payoff refers to the cumulative rewards that the agent is projected to accumulate. In simpler scenarios, it can be calculated by summing up the individual rewards:

$$Gt = R_t+1 + R_{t+2} + R_{t+3} + \cdots + R_T \quad (3)$$

where Rt is the reward obtained at time step t and T is the final time step. To address infinite rewards, a discount rate γ is used, which is a value between 0 and 1 that determines the value of future rewards. A low γ prioritizes immediate rewards, while a higher γ considers future rewards. The discounted return is crucial in avoiding the agent from being stuck in the same actions repeatedly without reaching the goal.

$$Gt = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \quad \gamma^k R_{t+k+1} \quad (4)$$

The above equation states that a reward obtained k time steps ahead is valued at $\gamma k-1$ times its immediate worth [6].

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

2.2.2.4 Value functions

The value functions are closely connected to the reward function in RL. While the reward function defines the optimal action to take into the current state to maximize the immediate reward, the value functions determine the best actions to take to accumulate the maximum reward over a longer period of time. The value of a state is the total expected reward that an agent can obtain in the future, starting from a specific state s and following a policy π. The state-value function is used to calculate this value and can be expressed as [6]:

$$v_\pi(s) = E_\pi[G_t | S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid St = s\right] \quad (5)$$

$E\pi$ represents the anticipated value, Gt denotes the discounted return, and t corresponds to any given time step. Another function that measures value is the action-value function, providing the value associated with executing an action a in a state s according to a specific policy π [6]:

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid St = s, A_t = a\right] \quad (6)$$

The value functions, $v\pi$ and $q\pi$, are important as they account for the fact that States with initially low rewards may result in states with substantially higher rewards, and vice versa - states that initially yield high rewards may result in states with diminished rewards. The state-value function v_π can be used to extract the formula called the Bellman equation [6]:

$$v_\pi(s) = \sum_{k=a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad (7)$$

The Bellman equation relates the value of a state to the value of future states. It involves the probabilities of taking actions and ending up in different states with rewards, given a current state and policy. The value is calculated by summing over possible actions, future states, and rewards, with each weighted by the probability and a discount factor. The resulting expected value is the sum of all possible outcomes.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

2.2.3.5 Model

The model is the environment's representation, which can be employed in diverse manners to enhance an RL system. An instance of its application is in forecasting the subsequent state and reward based on the present state and action. Possessing a model facilitates strategic thinking, enabling us to make informed choices considering forthcoming circumstances. RL methods that incorporate a model are known as model-based, whereas those solely reliant on experimentation are labeled as model-free [6].

2.3 Offline Reinforcement Learning

RL algorithms which are trained using pre-existing datasets without the need for interaction with an environment are known as offline RL [7]. These models have proved that could take the best decisions based on huge datasets. Therefore, we aim to explain the central idea behind offline RL, as well as its advantages and disadvantages.

In RL, also called online RL, the agent learns by exploring the environment and making decisions through trial and error. However, a such approach is not feasible when dealing with large and complex tasks that lack a realistic simulated environment for the agent to train in. For instance, creating accurate simulated environments for tasks such as healthcare or MANET networks can be both expensive and dangerous. Moreover, it can be challenging to create accurately simulated environments that are representative of the real world, especially when dealing with complex tasks that differ significantly from each other.

Offline RL offers a solution to these challenges. In this approach, the agent does not need to interact with the environment. Instead, it exploits a pre-existing dataset to learn the optimal policy. The idea is that with a sufficiently large dataset, the agent can learn the optimal policy quickly and reliably. The algorithm is given a static dataset of transitions (i.e., sets of states, actions, rewards, and resulting states) and must learn the optimal policy based on it.

The primary benefit of successful implementation of offline RL is that there is no need for a simulated environment. Additionally, the agent can learn based on previously collected data, without needing to explore the environment to collect new data. This eliminates the need to address the exploration-exploitation tradeoff, which arises when deciding whether to exploit known actions or explore new ones.

One of the main drawbacks of offline reinforcement learning (RL) is that it relies entirely on the data that is provided to the agent. This means that the agent may not be able to learn about parts of the environment that are not included in the data set. Additionally, offline RL requires a large amount of data to train effectively. The more complex the problem and the algorithm, the more data is needed. However, even with a large data set, there is no guarantee that the agent will find the best possible policy[7].

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

2.4 Algorithms

Most RL algorithms follow a similar process of learning where an agent engages with the environment by adhering to a prescribed policy. During this process, the agent observes the state, selects an action, and observes the next state and associated reward. This procedure is repeated several times, and the policy is adjusted based on the observed transitions. While there numerous effective methods for addressing RL challenges, choosing one to use can be challenging. In this thesis, the Deep Q-Network model was chosen due to its widespread use and success in solving tasks in communications and networking as described in [8]. Additionally, the availability of several libraries with the algorithm implemented can speed up the development process.

2.4.1 Q Learning

Over the years, many new algorithms have emerged in RL, with most being improved versions of previous ideas and approaches. Q-learning is a RL algorithm that operates independently of any underlying model, using dynamic optimization techniques to facilitate the agent's learning process through the evaluation of action outcomes within the environment. By systematically exploring all feasible actions across various states, the agent acquires knowledge of which actions yield the highest anticipated returns based on a discounted evaluation. The primary objective of Q-learning is to estimate the Q-values associated with an optimal policy, enabling the agent to make informed decisions regarding the most favorable actions within specific states. These Q-values are determined by the Q-function, which shares similarities with the previously discussed action-value function. Each calculated Q-value is subsequently updated and stored within a lookup table for future reference and computation. The Q-learning algorithm is mathematically represented by the following equation:

$$Q^{\text{new}}(S_t, A_t) \longleftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (8)$$

The formula shows that the Q-value, which is updated at each time step, is determined by the sum of three factors:

- $Q(S_t, A_t) + \alpha$: The current Q-value multiplied by the learning rate
- αR_{t+1}: The reward in the following time step multiplied by the learning rate
- $\alpha \max Q(S_{t+1}, a)$: The action that the agent believes is most likely to lead to the best outcome, given all of the possible actions that it can take in the next state.

The learning rate, represented by $0 \leq \alpha \leq 1$, controls the rate at which the agent updates previously learned behavior. This means that it defines the rate the agent can learn new behaviors. If the learning rate is too low, the agent will learn very slowly. If the learning rate is too high, the agent will forget everything it has learned and start over [6].

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

2.4.2 Deep Q learning

Q-learning faces a significant challenge in tasks that entail a large number of states and actions since its look-up table quickly expands in size. Moreover, accurately evaluating an action's value requires the agent to explore all states in the environment, that is unfeasible in bigger environments. To overcome these obstacles, Minh et al. [9] proposed a solution in the form of the DQN, which combines Q-learning with deep neural networks. The DQN has a notable advantage over Q-learning as it can approximate the optimal action-value function by leveraging a deep convolutional neural network in lieu of a traditional lookup table. This feature makes DQN more efficient than Q-learning in tasks with extensive state- and action spaces.

A deep convolutional neural network (DCNN) falls under the classification of feed-forward neural networks (FNN) and is depicted in Figure 5. The functioning of the network involves propagating the input through intermediary layers that make determinations based on prior layers and assess the potential impact of a stochastic alteration on the output, either positive or negative. The term "deep" is attributed to a neural network when it encompasses multiple hidden layers in its architecture.

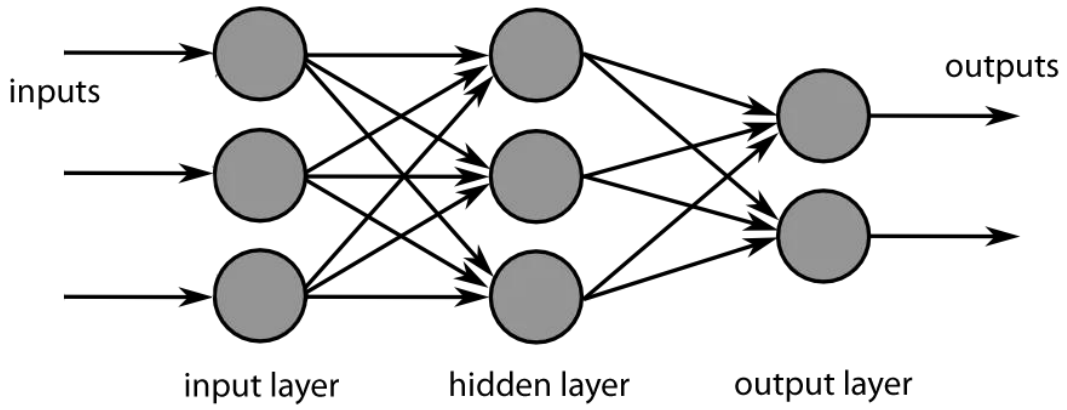


Figure 5: FNN retrieved from http://web.utk.edu/

Neural networks can cause instability in the action value approximation in RL methods [9]. According to Minh et al., on the grounds of two facts for this instability. Firstly, small updates to Q can lead to significant changes in the policy, and secondly, the action-values and target values have correlations. To overcome this instability, Minh et al. introduce two techniques: experience replay and a target network [9].

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Chapter 3: Software Tools

In this chapter the software emulator and the libraries that were used for the development of this thesis are described.

3.1 CORE-EMANE Emulator

Emulators are software programs or hardware devices that enable a computer or a different hardware system to imitate the behavior of another computer system or gaming console. In other words, they allow a computer to mimic the functionality of a different computer or console, including its operating system and hardware, without actually needing the original system. In this section, an overview of the network emulator CORE-EMANE that was used for the completion of this project is provided. The goal is to present the key elements of architecture and utilities with focus on their use in scenarios regarding this thesis.

3.1.1 Overview of CORE/EMANE

CORE [10][11], the Common Open Research Emulator, is a platform developed by the U.S. Naval Research Laboratory (NRL) that enables the emulation of networks on physical computer machines. Its functionality includes the ability to emulate routers, hosts, and simulate networks and network links in real time. CORE is compatible with Linux operating systems and utilizes Linux virtualization tools. In particular, it employs a form of virtualization known as para-virtualization, which involves virtualizing only a portion of the operating system. This approach allows for the isolation of multiple Linux server environments on the same hardware machine. CORE achieves this through the utilization of Linux namespace containers, which are interconnected using Linux bridging and virtual interfaces.

CORE offers both a graphical user interface (GUI) and a Python framework that facilitate the construction of virtual networks. Through the GUI, users are presented with an empty canvas where they can position routers, end hosts, and other networking devices like hubs and switches. These components can then be interconnected to form networks. In CORE, a session represents a collection of nodes and links that operate together to create a specific scenario.

On the other hand, the Extendable Mobile Ad-hoc Network Emulator (EMANE) [12] is a modular framework developed by Adjacent Link for emulating MANETs. EMANE focuses on Layers 1 and 2, which encompass the physical and data link layers, unlike CORE which primarily concentrates on emulating Layers 3 and above (network, transport, session, and application layers). In EMANE, a virtual device represents a single radio and is known as a Network Emulation Module (NEM). Each NEM is constructed by combining a MAC/link-layer module with a physical-layer module. An over-the-air (OTA) manager facilitates the exchange of packets between NEMs. Additionally, EMANE incorporates an event system that handles events such as location and pathloss, and allows for the insertion of optional shims between components.

By integrating EMANE into the CORE environment [13], it becomes possible to emulate high-fidelity wireless networks across the entire protocol stack instead of building Linux Ethernet bridging networks

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

within CORE. This integration combines the advantages of both tools, providing a user-friendly graphical interface for designing and configuring virtual networks that consist of lightweight virtual machines connected through adaptable Data Link Layer (MAC) and Physical Layer (PHY) models.

The integration of CORE and EMANE not only expands the options available to users but also offers a comprehensive view of the system. Additionally, it enables the reuse of existing mobility scripting formats without the need for re-implementation or conversion. This is made possible because EMANE incorporates event generators that can interpret the emulation event log (EEL) format. Therefore, when CORE is utilized in conjunction with EMANE radios, to make devices move around and connect to each other differently, based on instructions from these written files.

3.1.2 CORE architecture

Figure 6 below illustrates the key elements comprising the architecture of CORE. The core-daemon module serves as the central component responsible for managing the emulated sessions, encompassing nodes and links within a network. Nodes are generated using Linux namespaces, while links are established through Linux bridges and virtual Ethernet peers.

The core-gui module provides a user-friendly interface that supports the intuitive creation of nodes and links through drag-and-drop functionality. Additionally, it enables the launch of terminals for emulated nodes during active sessions. Communication between the core-gui and core-daemon modules occurs through an Application Programming Interface (API).

To interact with the nodes, users can utilize the vcmd command line utility, which enables the transmission of shell commands. Control commands directed at the core-daemon can be sent using the coresendmsg utility, granting users greater flexibility and control over the emulated environment.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

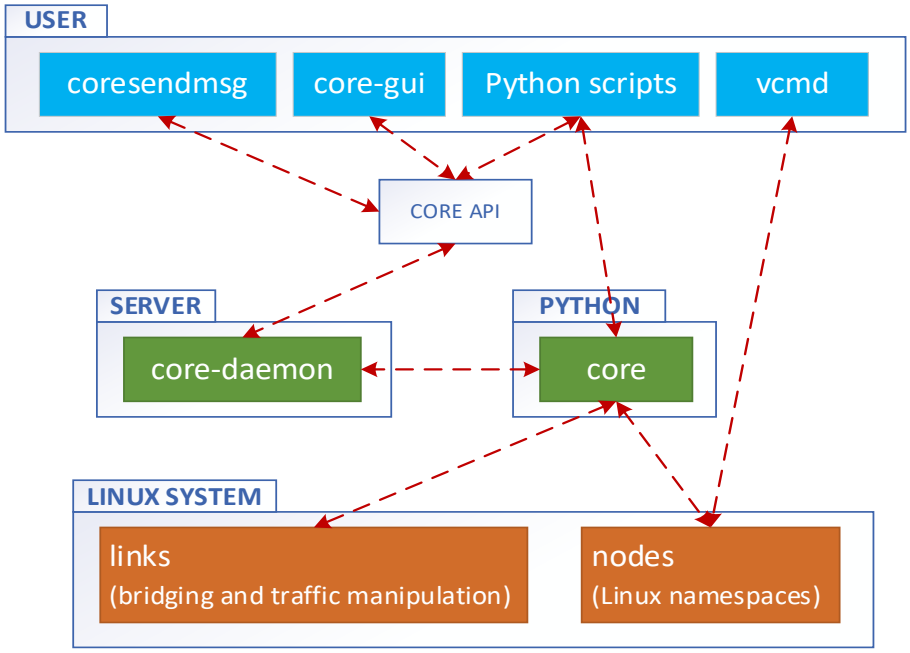


Figure 6: The main components of the Core architecture.

CORE provides support for emulating mobile networks by incorporating various mobility patterns and mobility scripts. Users have the ability to control the mobility of emulated nodes and the connectivity between EMANE network interfaces by utilizing integrated traffic and mobility generators that are capable of interpreting specific script formats. When CORE is utilized alongside EMANE radios, mobility and connectivity can be managed through the event generators mentioned earlier, which are capable of reading the EEL format. The traffic tool leverages MGEN traffic generator scripts, while the mobility generator utilizes NS2 mobility scripts, similar to those used in the NS2 network simulator. This integration allows for precise control and customization of the mobility and traffic patterns within the emulated mobile networks.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

3.1.3 EMANE architecture

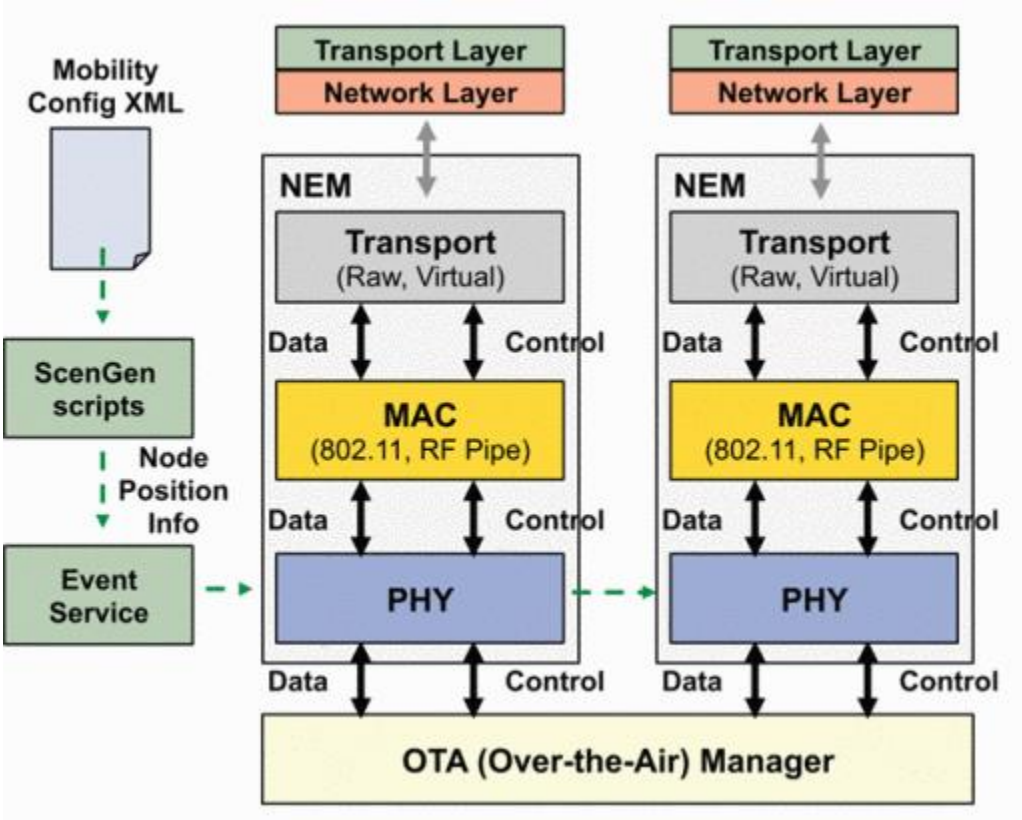


Figure 7: EMANE architecture [14]

In Figure 7 [14], the architecture of EMANE is depicted with all of its key components that work together to simulate the behavior of MANETs. These components are the NEM, the Event Service, the PHY Layer, the MAC Layer, Network Layer and the Over-the-Air Manager (OTA).

The NEM is the core component of EMANE. It represents a simulated network node or device in the MANET. Each NEM operates independently and can be configured with different parameters such as transmission power, antenna characteristics, and mobility patterns. Each NEM consists of two or three main components (PHY, MAC, Transport Layers) interconnected by bidirectional data and control paths. The PHY layer implementation handles the emulation of the physical layer, while the MAC layer implementation manages the link layer and medium access. The Transport component provides the physical interfaces for data transmission to a connected router. It offers both a raw interface and a virtual interface for NEM stack entry/exit points. The Event Service is responsible for managing and delivering events within EMANE. Events represent changes in the network, such as packet transmissions, link state updates, or mobility updates. The Event Service ensures that events are delivered to the appropriate NEMs based on their timing and context. The OTA is a component responsible for managing wireless propagation and communication within the simulated network. It handles the simulation of wireless transmissions, including the propagation of radio signals, signal strength calculations, and interference modeling.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

As also described in [14], in the case of conventional MAC protocols like 802.11, the information regarding the node's position is injected into the PHY layer periodically (per-second) using the event service. In more detail, the pipeline of packets transmission is that the PHY layer emulates the process of sending the data OTA by forwarding them to the OTA manager, which then it multicasts the packets to all the subscribed NEMs. Therefore, the receiver NEMs assess if the received data are physically be collected based on the position information of the transmitter NEM. If the transmitter is out of range for the receiver NEM, then the data packets are dropped. On the contrary, the PHY layer forwards the to the MAC layer. Once the MAC layer receives the packet, it examines the operating modulation rate, checks for collisions, and determines if there are any packet errors. It processes the packet accordingly, and if it is received without any errors, it is transmitted through the transport layer.

3.2 Libraries

Libraries are packages of pre-written pieces of code or routines that offer specific features for a variety of tasks. They contain reusable code that developers can incorporate into their own programs to simplify development, reduce duplication of effort, and leverage existing solutions. In the next subsections are described the libraries that were used for this thesis.

3.2.1 PyTorch

PyTorch has gained significant popularity as a machine learning framework the past few years. It was created by Facebook and released as an open-source platform in 2017, accompanied by the publication [15]. PyTorch is built upon Torch, a library for tensor computations, and the calculations performed in PyTorch. PyTorch is software that anyone can use for free, and its code can be changed by anyone. It is released under a license called the modified BSD license. PyTorch can be used with both Python and C++, but Python is easier to use and is what the developers are working on the most.

3.2.2 D3rlpy

D3rlpy [16] is a library specifically designed for building offline RL agents. It stands out from other libraries in the field by offering a unique combination of online and offline RL algorithms. Built on the PyTorch framework, D3rlpy provides a wide range of implemented algorithms and a plethora of features, continuously expanding with regular updates. Takuma Seno, the developer, highlights that this library stands out as a pioneering solution, catering both to the researchers and real-world practical applications in the real world. D3rlpy was selected as the ideal choice for implementing the offline RL agent due to its user-friendly API and comprehensive documentation. Apart from its wide range of algorithms, the library also offers convenient features for generating MDPDatasets, which serve as the essential input datasets for training the offline RL agents.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

3.2.3 Numpy

NumPy [17] serves as a fundamental library in the realm of scientific computing, offering indispensable functionalities. Its key contribution lies in the provision of multidimensional array objects, referred to as ND Arrays, alongside a range of routines that enable rapid operations on arrays, encompassing mathematical computations and shape manipulation. Given the significance of tensors, which essentially correspond to multidimensional arrays, NumPy assumes a central role within machine learning systems. Notably, prominent framework libraries like PyTorch rely on NumPy for crucial calculations at their core. Furthermore, NumPy plays a pivotal role in creating the dataset used for the offline RL agent, a topic that will be explored in more detail later.

3.2.4 Pandas

Pandas is a versatile library that is highly valued in machine learning endeavors. It is built on the Numpy library and offers a lightweight yet robust solution. Pandas excels in data handling tasks and is often employed in various machine learning projects. This library has the capability to effortlessly read diverse data formats, including CSV files, and convert them into convenient data frame objects. These data frames resemble a spreadsheet, possessing a tabular structure with rows and columns [18]. Leveraging data frames, analysts can efficiently analyze and manipulate data in numerous ways. Within the context of this particular project, Pandas plays a pivotal role in the preprocessing phase, where it effectively reads, manipulates, and stores the ship log data, which is stored in CSV format.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Chapter 4: System Architecture and Implementation

In this chapter the description of the system architecture that was designed in this thesis will be outlined. More specifically, an overview description of the system's components and their implementation details follows.

The main idea is that each follower controller in the topology informs the leader controller about its state, meaning its observations. Then, the MC after receiving this information from each controller, feeds the pre-trained RL model and in turn this model predicts the most appropriate actions for each follower controller. In turn, the MC communicates the actions to the corresponding nodes, so they execute these predicted actions. In this way the system in automatic manner is dynamically adjusted in any situations, either when the nodes are moving with high velocities or be totally inactive.

In the next sub-sections a detailed description for each aspect of this design follows.

4.1 Dataset

Although the CORE/EMANE emulator it is a complete and powerful platform for research development in the field of communications, it requests a lot of computing resources. The procedure of the Online RL, which requests to capture the states of the environment online, then to act and waiting for the new observations along with the reward it is an overkill for our testbed. Recall that there are already other processes of running such as etcd, Ryu, etc. Therefore, the Offline RL it was the one-way solution. As it is already mentioned in chapter two, the Offline RL is based entirely on the dataset because it cannot interact with the environment. Hence, a specific system was designed for retrieving the environments states. More specifically, Figure 8 shows that each controller contains five more entities, these are the four parsers, which fetch the environment observations as well as the coordinator process. The last has the task to synchronize the four parsers so they are coordinated when retrieve the state observations.

4.1.1. Observations

The creation of the dataset was one of the most challenging tasks during this thesis. The basic idea was the collection of as many possible data from different layers in the network stack. That is, the data that are included into the dataset are relative to the PHY and MAC layers as well as to the Network/IP layer. The gathering of this information derived from four different processes that each parsing the desired data.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

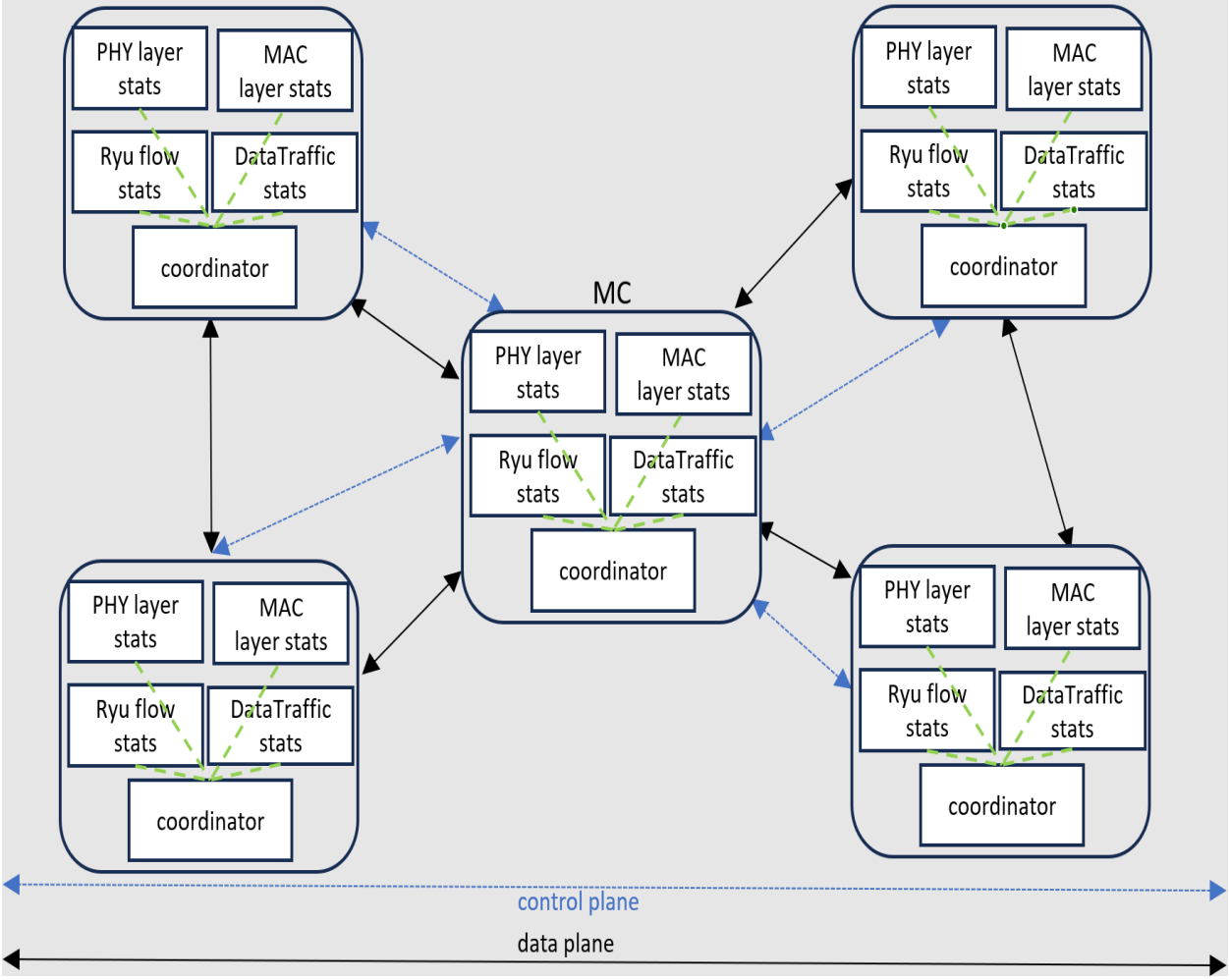


Figure 8: The coordinator and the 4 parsers entities in each node.

4.1.1.1 PHY and MAC layers

Data for both of these two layers retrieved from the emulator. More specifically, CORE/EMANE offers *emanesh*, a command line interface that allows users to interact with and control the EMANE framework. Retaining from chapter three, where the CORE/EMANE description was given, the EMANE is responsible for the emulation of the MAC and PHY layers. Therefore, building on the *emanesh* tool we developed two separate parsers to observe the network behavior regarding these two layers. The data that was collected with these parsers are shown in Table 1 and Table 2 along with a short description. In Table 1, there are stats from PHY layer, such as the *RX_power* which is a real-time value metric, meaning that in each time slot contains the real instant value of the node's receiving power. On the other hand, the *Drop_Unicast_RX_sensitivity* variable shows the number of packets that get lost due to the receiver's sensitivity. That is, this variable is a counter and increases as the session keeps going. Considering that we care for periodically collection of the environment's observation, to overcome this constraint the parsers except from fetching the information, performs calculation so to retrieve the actual data from variables that

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

are counters. In addition, both the counters and status variables are arranged in pairs of the node and its one hop distance neighbors. Then, to have an overview of this variable for the node, the parsers apply four statistics metrics <mean, max, min, std> on such variables, excluding Longitude and Latitude.

Finally, there are the synthetic variables that are produced by combining info from the emanesh tables, like average distance, velocity and packet loss. The average distance and velocity are computed based on coordinates (Longitude, Latitude). Regarding the former parameter, the parser measures the distances between the node and the rest of the nodes in the topology, and then calculates the average value of it as well as the standard deviation. The velocity of the node is calculated based on the covered distance of the node in the time – period of fetching the coordinates. The packet loss variable shows the packet loss on MAC layer level and it is calculated based on the Metric_Missed_Packet and the Metric_RX_Packets MAC stats.

Table 1:PHY Layer stats

Variable Name	Description
Status_RX_power	Instant Receive Power
Latitude	Nort-south position of node.
Longitude	East-west position of node.
Drop_Unicast_RX_sensitivity	Counter for dropped packets due to Receiver sensitivity
Average_Distance	The average distance of the node from the rest topology's nodes.
Std_Distance	The std distance of the node from the rest topology's nodes.
Velocity*	The velocity of each node. Synthetic statistic derived from Longitude, Latitude and time.

Table 2: MAC Layer stats

Variable Name	Description
Status_RX_Packets	Instant Received Packets.
Status_TX_Packets	Instant Transmitted Packets.
Status_BW_Util_Ratio	BandWidth (BW) utilization ratio.
Status_SINR_Avg	Average of Signal-to-Interference-plus-noise-ratio (SINR).
Status_NF_Avg	Average of Noise Frequency.
Metric_RX_Packets	Counter of received Packets.
Metric_TX_Packets	Counter of transmitted Packets.
Metric_Missed_Packets	Counter of Missed Packets.
Drop_Unicast_SINR	Counter of dropped Packets due to SINR.
Drop_Unicast_Duplicate	Counter of dropped Packets due to Duplicates.
Drop_Unicast_RX_during_TX	Counter of dropped Packets due to Receiving Packets during Transmitting one.
Drop_Unicast_Hidden_Busy	Counter of dropped Packets due to Hidden Terminal (node).
Num_Bytes_Rx	Counter of bytes received.
Num_Bytes_Tx	Counter of bytes transmitted
Packet_Loss*	The MAC layer Packet Loss. Synthetic statistic based on counters of missed and received Packets

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

4.1.1.2 SDN statistics

Another component of the dataset is the data that are related to SDN controller, and specifically the statistics of the flows into the switch's Flow Table. As earlier mentioned in this document, the SDN controller in [1] was built using the Ryu framework. Beyond the fact that it is an open-source software library, it provides useful interfaces for the developer. Particularly, we leveraged on the REST API that is offered to retrieve the switch's statistics. In detail, a third parser was designed to call from the Ryu controller application the statistics which are showed in Table 3. Once again, these metrics are counters that increase as the session goes on, so this parser also computes the difference between the current fetched value and the previous one.

Table 3: Switch's Flow Table stats

Variable Name	Description
`byte_count`	Counter of total bytes
`flow_count`	Counter of total flows
`packet_count`	Counter of total packets

4.1.1.3 Traffic statistics

The fourth and last parser was designed to take statistics regarding the data network traffic. As mentioned before, the SDN logic separates the communication traffic between the nodes to control and data planes. The control plane includes packets that are related to control functionalities, like OpenFlow packets, while the data plane regards the actual data exchange between the nodes. However, the network infrastructure that we built on, as described in [1], uses one network domain for both of these planes. Thereafter, this parser first generates data traffic (data plane traffic) and then collect the statistics. More specifically, the *fping* command tool used for this task. The fping sends ICMP Echo Request packets (ping) in parallel to multiple nodes in order to determine their reachability and measure their round-trip time. Noteworthy the option flags used by this command as shown below:

```
fping -q -p {PERIOD} -C {PACKET_COUNTER} -t {TIMEOUT} -A {destinations}
```

- -q: quite mode
- -p {PERIOD}: PERIOD defines the interval in milliseconds (ms) that packets are send to the destination nodes and is correlated with computing resources of the system.
- -C {PACKET_COUNTER}: PACKET_COUNTER defines the number of the packets that fping send to each node destination. More, this option flag has a specific output for the results of the communication. Response times for each target are shown in a way that makes it easy to collect them automatically., where the lost packets are symbolled with dashes (-). This is beneficial for the parser as explained below.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

- -t {TIMEOUT}: TIMEOUT defines the interval in ms, that fping waits between successive packets to an individual target. If this time pass the fping considers as loss the unreceived packets. In this way, we limit the cases where packets can be received in time interval bigger than TIMEOUT.
- -A: where the destinations nodes' IP addresses are defined.

When the fping is executed, the parser collects the outputs per node destination and compute the packet loss and the Round Trip Time (RTT). Based on the specific output that is provided due to *-C* option, the parser computes the packet loss per destination node, based on the ratio received packets to all packets. To summarize the packet loss metric for each node, the average value of per destination node packet loss is measured. Regarding the RTT average, the same pattern of calculation is followed except the last part, the overall RTT measurement. That is, the average statistic we consider that it is not suitable metric to have an overview of the RTT for a node. In contrast we used the median absolute deviation to have a better view of the total node's RTT. That was the main reason to use the -C flag and not get the ready provided statistics from fping.

Important to clarify is that the above fping metrics would be meaningless without a special modification of the SDN controller application. Specifically, when a flow rule is established to the switch's flow table by the controller, there are three options for its existence. One choice is the flow to remain into the flow table forever, the second option is to remain there as far as it used otherwise is deleted after a time interval, called IDLE_TIMEOUT. The third option is to predefine its existence by the flag HARD_TIMEOUT. Dealing with mobile nodes, the first two options will lead to false data for each node, because when the relative flow rule would be installed to the source node's switch, thereafter and while the topology has changed, the node will continue forwarding based on the same flow rule which is not updated. Then as our routing algorithm is reactive and updates flows on demand, the third option was chosen in order to constantly request for new flow rules and better evaluate the aforementioned metrics.

Table 4: Traffic stats

Variable Name	Description
`Averaged_packet_loss_rate`	Averaged Packet loss rate
`Averaged_rtt`	Averaged RTT

4.1.2 Actions

Recalling from chapter two, where the fault tolerance as well as the topology discovery mechanisms were described, two specific points were highlighted. Firstly, regarding the fault tolerance, there are two key parameters, the *heartbeat_interval* and the *election_timeout.* The former defines how often the nodes communicate with each other who the MC is, while the last sets the time interval that each node into the cluster waits until consider the MC as lost. In parallel, the mobility of the nodes makes their communication

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

more challenging because of the frequent link status changes. This issue is resolved, as mentioned in chapter two, by the HELLO_INTERVAL parameter.

Consequently, the actions set is consisted of these two sub-actions, the tune of etcd and Ryu parameters. According to etcd documentation, it is recommended that heartbeat interval is set to the maximum average RTT between members. The election timeout should be configured based on the heartbeat interval and the average RTT between members. The election timeout needs to be at least 10 times the round-trip time to account for network variations [19]. Based on this, we ended up to perform as one action the concurrently tuning of both the heartbeat interval and the election timeout. Thus, we have predefined a set of three pairs with values (*heartbeat_interva, election_timeout*). Namely these values are: { (100, 1000), (1000, 10000), (4000, 40000) }. The units of measurements for these time intervals are expressed in milliseconds (ms).

As regards the HELLO_INTERVAL, the values for this action are {(0.5), (2), (4)}. The units of measurements for these time intervals are expressed in seconds (s).

Noteworthy, that there was not a special defined policy that was followed during the collection of data from parsers to create the dataset. Literally, the aforementioned actions were chosen in each step in random. However, in Offline RL it is a common technique to collect the data in this way.

4.1.3 Reward

The reward function relies on the packet loss and the RTT resulting from the fping command that is executed by the relevant parser, and is defined in (7):

$$R = \frac{1}{(w_1 * packet\_loss) + (w_2 * RTT)} \quad (7)$$

Following (7), it comes that the higher the packet_loss is, the smaller the reward will be. Same situation for the RTT. Then, the agent during its training will take great reward if its actions lead to low packet loss and RTT values, while it will be penalized with small rewards if this is not the case. The parameters w_1 and w_2, are also of great importance, as they determine the weight the two metrics have regarding the reward. That is, the value setting for w_1 and w_2 can lead to different rewards values, and this is related to the objective the agent has to succeed. The value that each *w* can take is between zero to one and in our implementation, we ended up by equally defines these two weights. That means in our model, the packet loss metric is as crucial as the RTT metric is for the definition of the reward function.

4.1.4 Dataset Structure

At this point it is worth mentioning, the format of the final dataset that will be given to the algorithm for training. Using the d3rlpy library, with the provided *MDPDataset* class, the aforementioned data, observations, actions and rewards, are stored in a similar way as the structure of a Markov Decision Process.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

More specifically, the data are organized into episodes and transitions. An episode is a sequence of interactions between the agent and the environment. It starts with an initial state, and the agent takes actions based on the observed states, receives rewards, and transitions to new states. This process continues until a terminal state is reached, indicating the end of the episode.

A transition is a single step within an episode, representing the agent's observation, action, reward, next observation, and terminal flag. It captures the information of the agent's state, the action it took, the reward it received, the resulting state, and whether that state is terminal or not. In Figure 9, it is depicted this logic, which is inspired from the double linked-lists data structure.

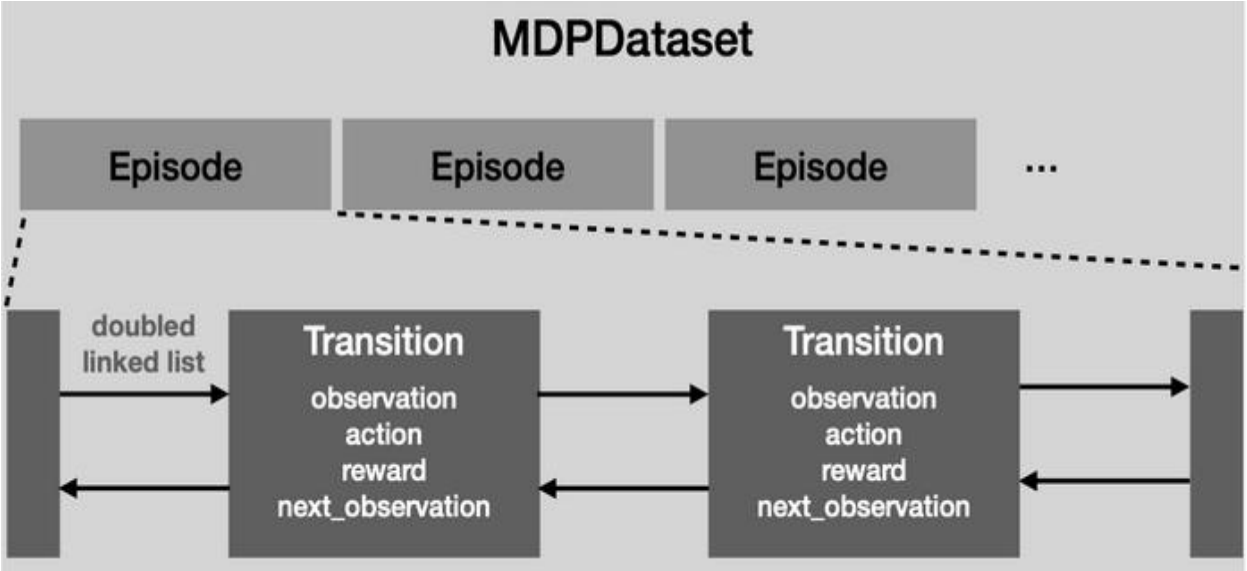


Figure 9: MDPDataset structure [16]

4.1.5 Mobility

Mobility is an integral part of the MANETs, thus we need to experiment with mobility scenarios to collect the data. CORE/EMANE emulator supports the experimentation with mobility scenarios with three different ways, Network Simulator 2 (NS2)-scripts, gRPC API,and EMANE events. In this project, the first one was chosen. ns2 is an open-source software simulation tool for analysis and investigation in communication networks [20]. Towards to this direction, to generate the mobility scripts, in ns2 format, we used the BonnMotion [21]. BonnMotion, a Java application, generates and examines mobility situations. It is crafted by the Communication Systems team at the Institute of Computer Science IV, located at the University of Bonn in Germany. This software functions as a valuable resource for exploring the attributes of MANETs. Additionally, the scenarios it produces can be exported for utilization in various network simulators such as ns-2.

BonnMotion offers several mobility models, like ManhattanGrid, RandomDirection, RandomWaypoint etc. The chosen model for the experiments was the RandomWaypoint. This mobility model involves mobile

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

entities moving within a defined area in a random manner. In this scenario, each entity selects a random destination within the area and moves towards it at a specified speed. Upon reaching the destination, a new random destination is chosen, and the process repeats. RandomWaypoint is commonly used to study and evaluate the performance of mobile networks, ad hoc networks, and routing protocols. It allows researchers and network designers to analyze how network behavior and communication patterns are influenced by random movement and varying speeds of mobile entities.

The main parameters we define on the creation of the mobility scenarios were the coordinates in x-axis and y-axis on CORE/EMANE canvas as well as the velocity. More specifically, we define x=500 and y=500 as the simulation area that nodes can move and regarding the velocity, it varies in a range from 5 km/h to 40 km/h.

4.2 Preprocessing

Data normalization is a widely adopted technique in the field of machine learning. It involves the conversion of numeric columns to a standardized scale. In machine learning, certain feature values may possess significantly larger magnitudes compared to others. These features with higher values can exert a disproportionate influence on the learning process. However, it is important to note that this does not imply their greater importance in predicting the model's outcome. Data normalization aims to harmonize the scales of variables with different ranges. By normalizing the data, all variables obtain comparable significance in the model, leading to enhanced stability and performance of the learning algorithm.

There are many ways to standardize data in statistics, however in this thesis used the Standard, or z-score, and the Min-Max scalers. The former used to rescale the observations set, and actually it is offered from the d3rlpy library. Then, by defining the standard scaler class giving as input the mean and the standard deviation of the observations. The fit and transformation of the data are performed internally by the algorithm before the train process starts. The Min-Max scaler used to rescale the reward values for each transition. In a similar way, the Min-Max scaler is also provided by the d3rlpy library and this time giving as inputs the min and max values of the rewards to the corresponding scaler’s class, the transformation of the data is internally achieved.

4.3 RL Agent Design

As derived from the set of actions, it becomes evident that these actions exhibit discrete values. Consequently, in accordance with the principles of RL theory, a particular category of algorithms emerges as apt for effectively addressing the training and resolution of the underlying problem. In our implementation, deliberately opted for the adoption of the Deep Q Learning approach, primarily due to its simplicity and the numerous analogous endeavors in the field of communications and networking. Furthermore, we derived substantial advantages by leveraging a bespoke algorithm known as Discrete Conservative Q Learning (CQL) [22], which incorporates the core principles of Deep Q Learning while consciously addressing the unique demands and challenges posed by offline training scenarios.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Once again, we take advantage from the utilization of the d3rlpy library, which offers a pre-existing implementation of the Discrete CQL algorithm. This enables us to employ this algorithm without the necessity of constructing it from scratch. Subsequently, we proceed with some of the key features of this algorithm and how they have been integrated into this thesis.

4.3.1 Double Deep Q Network (DDQN) architecture

The Discrete version of CQL algorithm represents an advancement in the field of data-driven deep RL, building upon the foundation of the DDQN approach. This neural network is trained using a combination of supervised learning and RL techniques, facilitating the learning of complex patterns and non-linear relationships within the data. In particular, this architecture contains two identical DNN, that each has a specific role. The first one is used to estimate each time the action's Q value for the current state, while the second one, called Target network, is used to update the current action's Q value. The target network is typically updated less frequently than the primary network. There is a dedicated hyperparameter, namely *target_update_interval*, that defines how often the Target network is updated. This helps to stabilize the learning process by making the target values more stable and reducing the variance in the training updates. The choice of the target update interval is a trade-off between stability and responsiveness to changes in the environment, and is typically set based on empirical results and experimentation.

By default, the neural network architecture in Discrete CQL implemented in d3rlpy consists of two hidden layers with the state of the art regarding the activation functions, Rectified Linear Units (ReLU)[23]. The number of neurons in each hidden layer can be specified using the n_neurons parameter. The output layer has separate neurons for each action in the action space, and it uses linear activation.

4.3.2 Gamma factor

Another significant hyperparameter of this algorithm is the gamma factor. The gamma facto in deep Q-learning is a number that controls how much the agent cares about future rewards. It is a scalar value that lies between 0 and 1, and determines the amount of weight given to future rewards versus immediate rewards. A high value of gamma (e.g. 0.9 or 0.99) means that the agent places a high value on future rewards, and is willing to delay its reward in order to achieve a more valuable long-term outcome. In other words, the agent is more focused on the long-term goals and takes a more strategic approach. A low value of gamma (e.g. 0.1 or 0.5), on the other hand, means that the agent places a higher emphasis on immediate rewards, and is more focused on the short-term goals. This can result in more aggressive or riskier behavior. The choice of the gamma factor can have a significant impact on the behavior of the agent, and it is usually a parameter that needs to be tuned based on the task and environment being considered.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

4.3.3 Batch Normalization

Group adjustment is a methodology used to train highly complex neural networks by standardizing the inputs to each layer for every subset of data. This results in stabilizing the learning procedure and significantly reducing the number of training iterations needed for training intricate neural networks. One aspect of this difficulty lies in the fact that the model is refreshed iteratively, moving from the output to the input, employing an estimation of error that acknowledges the fixed weights in the layers preceding the current layer. Group adjustment provides a comprehensive approach to parameterizing almost any intricate neural network. The reparameterization essentially mitigates the challenge of coordinating updates across multiple layers.

Table 5: RL Agent's hyperparameters values

Hyperparameter	Value
Gamma	0.7
Target_update_interval	100
Number of neurons	256 x 2
Epochs	100
Batch size	32

4.4 Integration of RL Agent into the system

After the completion of the training process for the RL Agent, the system architecture of this project, as shown in Figure 10, is illustrated. The RL agent, which is a RL model that has already been trained, receives environment observations as input and predicts the optimal actions for the MC and the follower controllers. The term "optimal" refers to the actions that align closely with the predefined reward, which aims to minimize packet loss and overhead in the network.

The incorporation of the RL Agent into the system adheres to the centralized concept of SDN. In this approach, the MC gathers observations from each follower controller within the cluster in order to make informed decisions. The subsequent sub-section provides a detailed explanation of this procedure.

4.4.1 Communication between Controllers

Referring to Figure 8, the controller-to-switch communication is represented by the dotted blue lines, which are established through the utilization of the OpenFlow protocol. In the context of SDN technology, OpenFlow has emerged as the dominant communication protocol between controllers and switches due to its ability to offer enhanced flexibility, control, and innovation in network management. Moreover, OpenFlow facilitates the development and experimentation of customized functionalities beyond the standard OpenFlow specification by incorporating an extension known as OpenFlow Experimenter.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

By leveraging this extension, we harness the capabilities of the OpenFlow protocol by introducing specialized OpenFlow Experimenter messages. These messages play a pivotal role in facilitating the transmission of state observations between the follower controllers and the MC. Upon receiving the observations from its followers, the MC feeds them into the pre-trained RL Agent and awaits the corresponding action predictions. Subsequently, the MC communicates the predicted actions back to the follower controllers via these experimenter messages. This entire process is illustrated by the dotted blue lines depicted in Figure 10.

To ensure the effectiveness and efficiency of the communication, these messages are transmitted periodically. This approach accounts for the delay required for an action to be executed while minimizing the overhead associated with the communication process. Taking into consideration the OpenFlow Experimenter header fields along with the payload, which encompasses the selected observations from the environment by each controller, the total size of the packet can be determined:

$$Total\ Packet\ Size = size(type(Experimenter\ Field)) + size(type(Exp\_type)) + size(Data\_filed) \quad (9)$$

$$Total\ Packet\ Size = size(int) + size(int) + size(Data\_field) \quad (10)$$

$$Total\ Packet\ Size = 4 + 4 + size(Data\_field) \quad (11)$$

As it comes from the Tables 1 to 4, the total observations each controller sends to MC are twenty-five environment variables which are all type of float64 meaning that are of 8 bytes size. Then, the (4) is:

$$Total\ Packet\ Size = 4\ bytes + 4\ bytes + 25 * 8\ bytes \quad (12)$$

$$Total\ Packet\ Size = 108\ bytes \quad (13)$$

Although, this packet is sent periodically and by each follower controller to the leader controller, it seems to not add great overhead. Noteworthy, that when the MC interacts with RL agent, as depicted with the red arrow in Figure 10, sends back to the controllers a relevant experimenter message containing only the predicted action which is type of integer (4 bytes), meaning the response size is 12 bytes.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

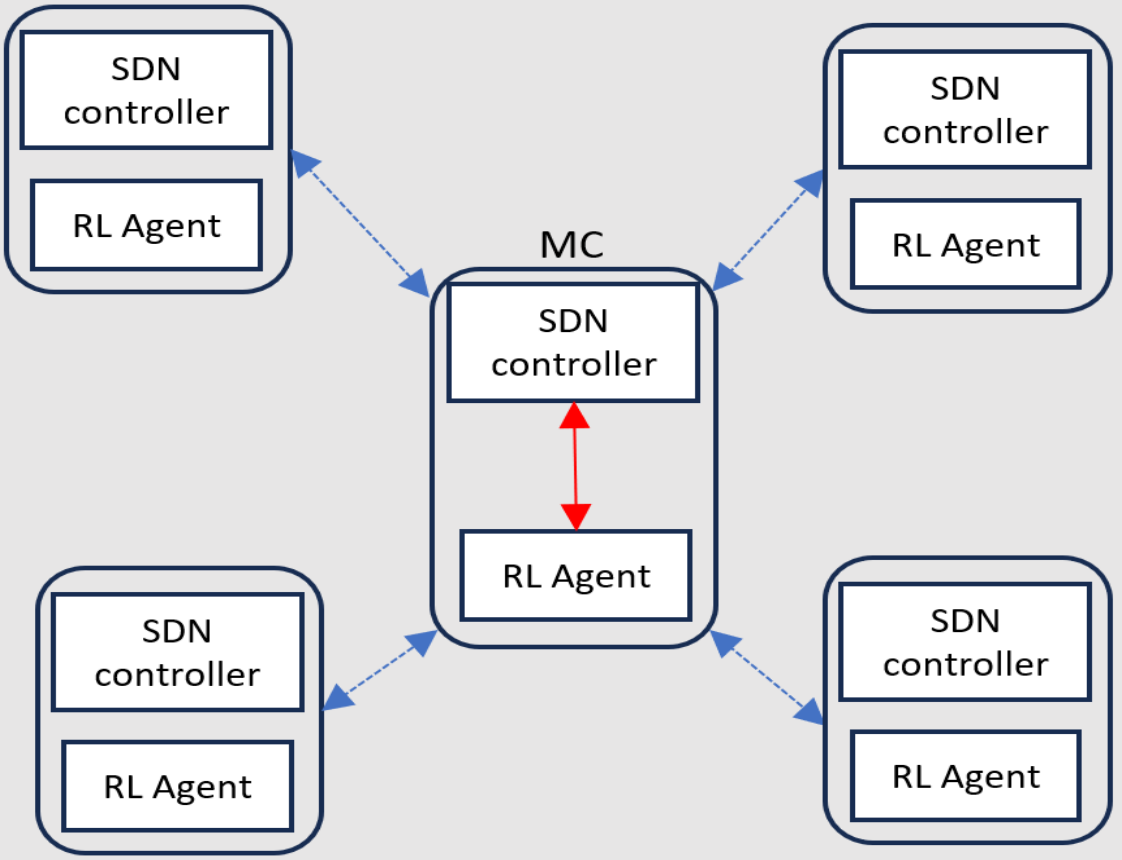


Figure 10: Integration of the RL Agent into the System's architecture

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Chapter 5: Results and Evaluation

After integrating the trained RL Agent into the SDN-MANET architecture, we conducted performance testing and evaluation. Using the BonnMotion tool, we created various topologies with different movement patterns and nodes' velocities. For the movement patterns, we utilized the RandomWaypoint algorithm, which generated diverse patterns while keeping the seed unchanged. Regarding node speeds, we designed three scenarios: topologies with stationary nodes, nodes moving at speeds up to 5 m/s, and nodes moving at speeds up to 10 m/s.

The evaluation process of the Agent involved an iterative approach with the training procedure. In other words, after training the algorithm, we applied the evaluation process described earlier. As a result, we observed that the gamma parameter significantly influences the agent's performance. In essence, the gamma parameter affects how the algorithm learns. For small gamma values, the algorithm tends to prioritize immediate rewards, while for larger gamma values, it strives to achieve higher rewards in the future. Therefore, we conducted a comparison between trained RL agents with different gamma values. The rest hyperparameters remained the same as they are showed into Table 5.

As it is already mentioned, the gamma parameter takes values from the range zero to one. Then, we trained the RL Agent by changing each time the gamma value, for values into this range with step of 0.1, and we ended up with ten different RL Agents.

The bar plots in Figure 11, Figure 12 and Figure 14 concern the metric of the total network delay of the topology, and each one of them refers to the respective scenario: static nodes, nodes with a maximum speed of 5 m/s, and nodes with a maximum speed of 10 m/s. Similarly, the Figure 13 and Figure 15 refer to the same comparison of different values of the parameter gamma with which the RL Agent was trained this time regarding the total packet loss of the topology. It's worth noting that for the scenario of static nodes, where there is no mobility, the total packet loss ratio remains zero, so such a figure is not applicable. By comparing the values of gamma among them for each scenario, we conclude that the RL Agent with a gamma value of 0.7 has the best overall performance in terms of both the Delay metric and the Packet loss rate metric. Thus, this was the selected value for the gamma parameter.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

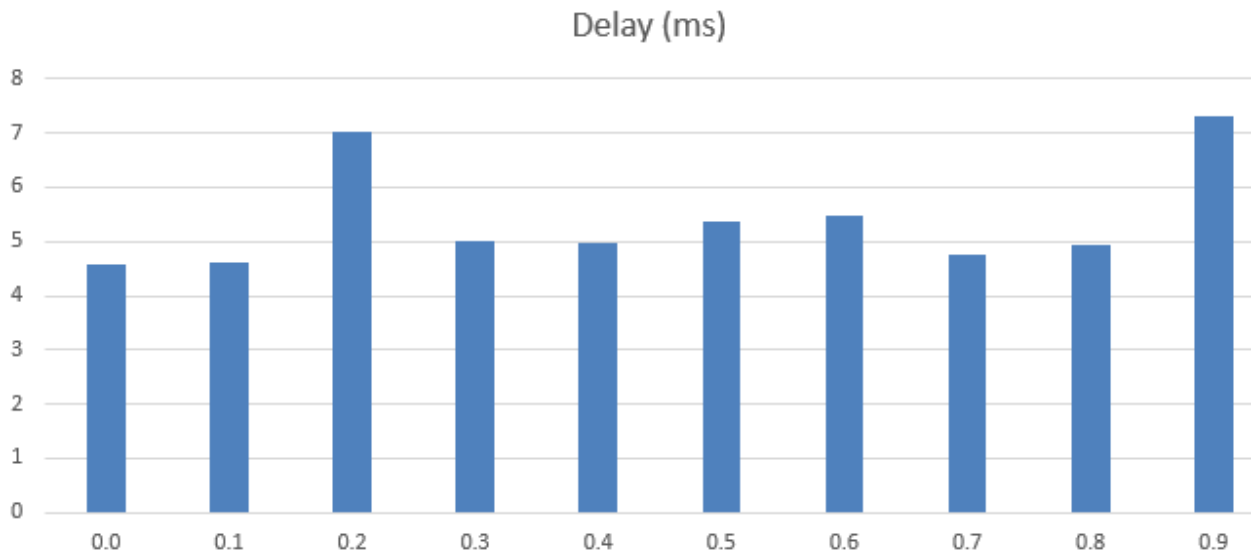


*Figure 11:*Comparing different gamma values for the RL Agent.in scenarios with static nodes and average the total network's delay.

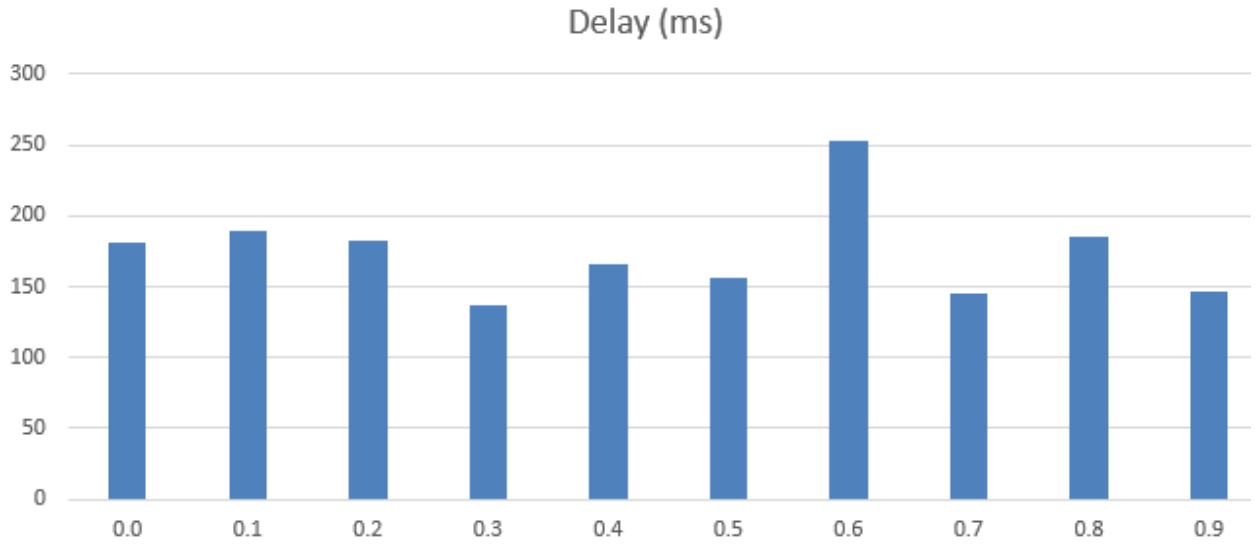


Figure 12: Comparing different gamma values for the RL Agent in scenarios with speed up to 5 m/s and average the total network's delay.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

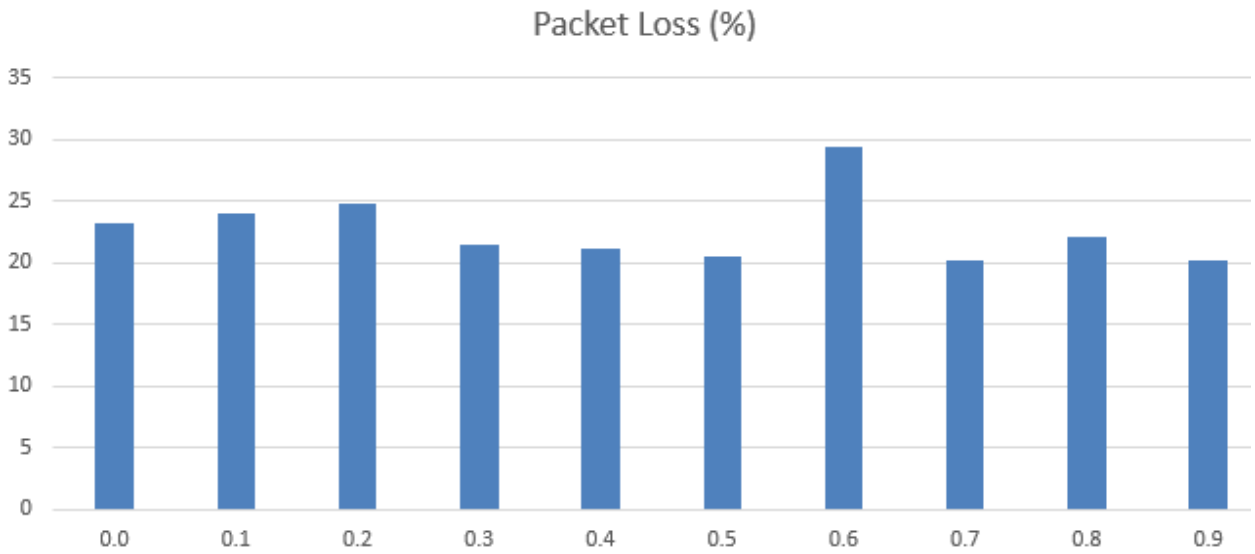


Figure 13: Delay - Comparing different gamma values for the RL Agent in scenarios with speed up to 5 m/s and average the total network's packet loss rate.

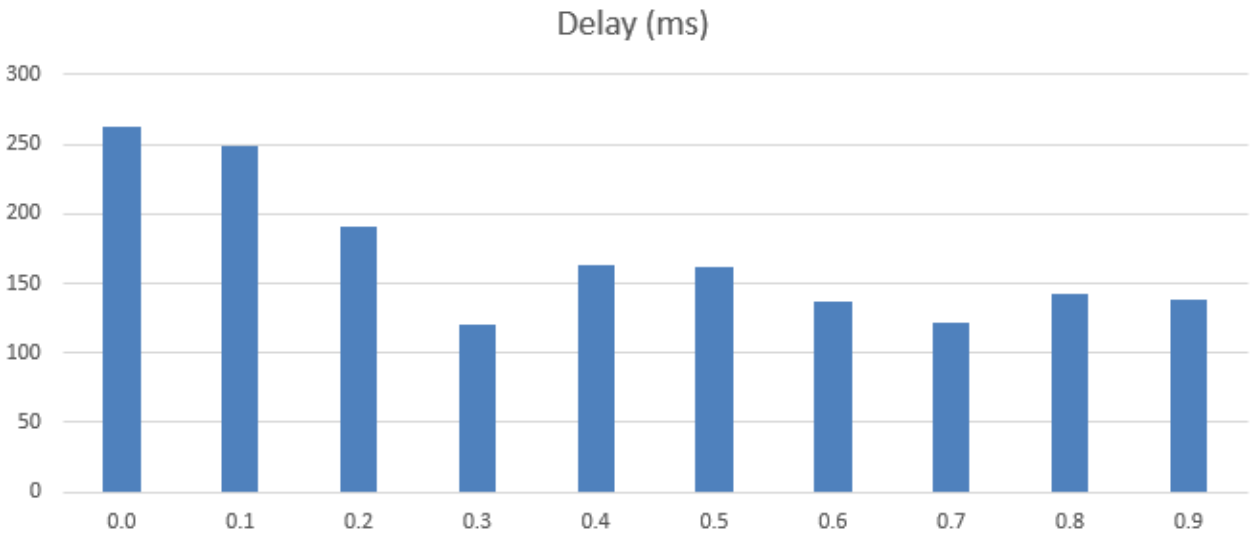


Figure 14: Comparing different gamma values for the RL Agent in scenarios with speed up to 10 m/s and average the total network's delay.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

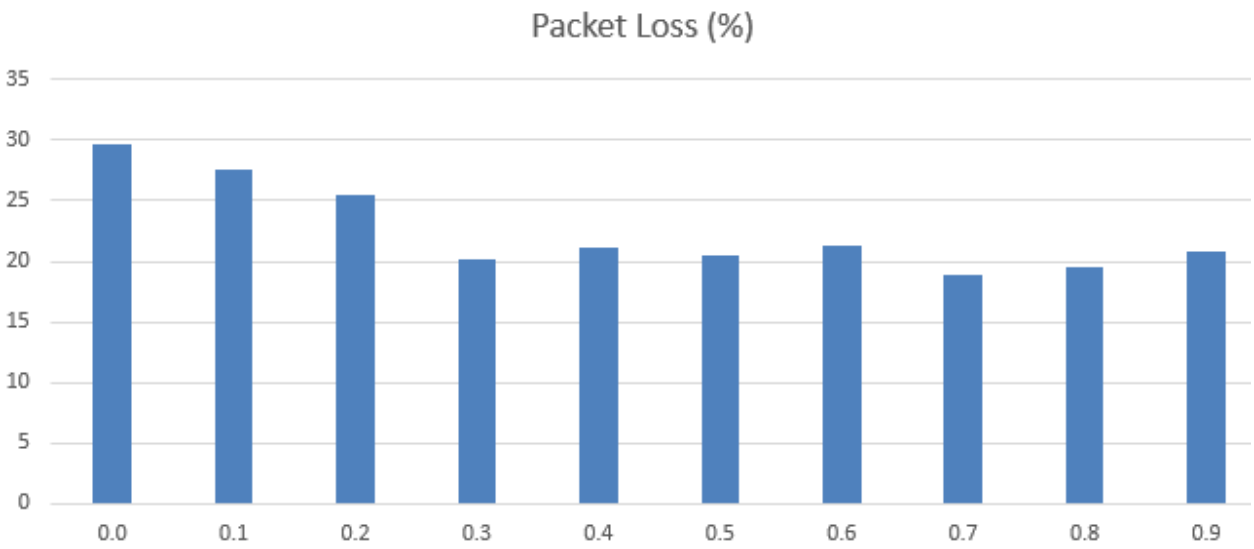


Figure 15: Comparing different gamma values for the RL Agent in scenarios with speed up to 10 m/s and average the total network's packet loss rate.

In the following bar plots, the comparison is made between the trained RL Agent with gamma value to be 0.7, the default architecture where no agent is used for dynamic parameterization of the topology discovery (HELLO_INTERVAL) and fault tolerance (heartbeat_interval, election_timeout) mechanisms (no_model), and an agent that randomly selects the dynamic parameterization of these variables (random_model). It's worth noting that the random model was the method of collecting data for creating the dataset. As seen in the following diagrams, the trained RL Agent we propose outperforms almost in all scenarios and metrics.

In detail, the Figure 16 shows the unique case where our model has exactly the same performance as the classical architecture (no_model) regarding the delay in a topology with static nodes. The random_model has worse performance in this scenario for the same metric. In scenarios related to node mobility, our model significantly outperforms the other two proposals/alternatives. Specifically, in scenarios with speeds up to 5 m/s, Figure 17, our model shows a decrease in the total delay to 145.6 ms from the 200 ms of the no model and 159 ms of the random model. Similarly, for the same metric, as shown in the Figure 19, the total delay value drops to 121.7 ms from the 168.7 ms of the no_model and 260 ms of the random_model. Similarly, in Figure 18 and Figure 20, the comparison of these three models regarding the total packet loss of the network is shown for scenarios with speeds up to 5 m/s and 10 m/s respectively. In the first case, the packet loss rate achieved by the RL Agent is 20%, while the no_model is 23.6% and the random_model is 21.7%. In the second case, the RL Agent achieves a total packet loss rate of 18.8%, compared to 22% for the no_model and 30% for the random_model. Once again, the bar plot for the static mobility scenario is omitted as all of the tree models have the same performance.

It's important to mention that during our experiments, we had to deal with a significant obstacle regarding randomness, the events generated by EMANE. For this reason, we repeated the experiments several times for each scenario and then calculated the average of the results to eliminate the variation in them.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

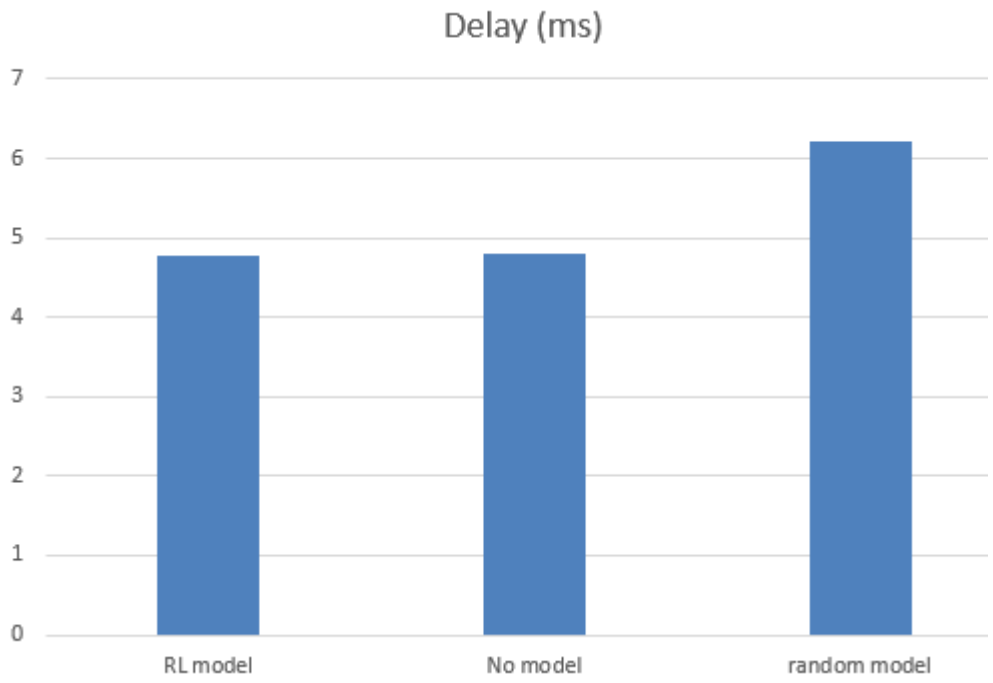


Figure 16: Comparison of the trained RL Agent with gamma=0.7 against using no model and using an agent chooses its actions in random (random model) on scenarios with static nodes regarding the total network's delay.

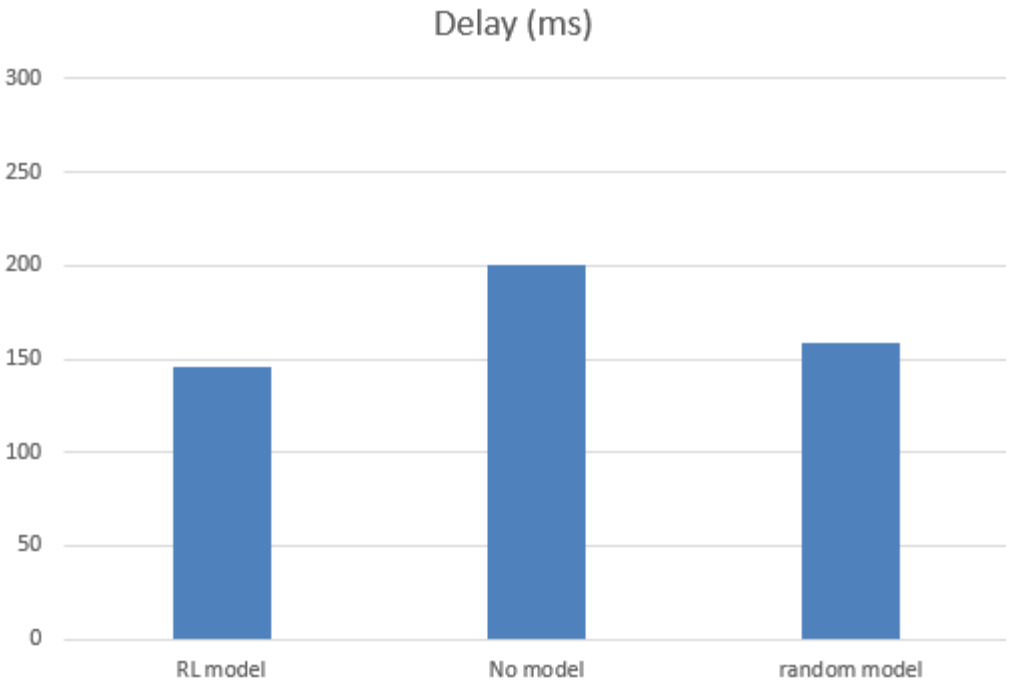


*Figure 17: C*omparison of the trained RL Agent with gamma=0.7 against using no model and using an agent chooses its actions in random (random model) on scenarios with speed up to 5 m/s, regarding the total network's delay.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

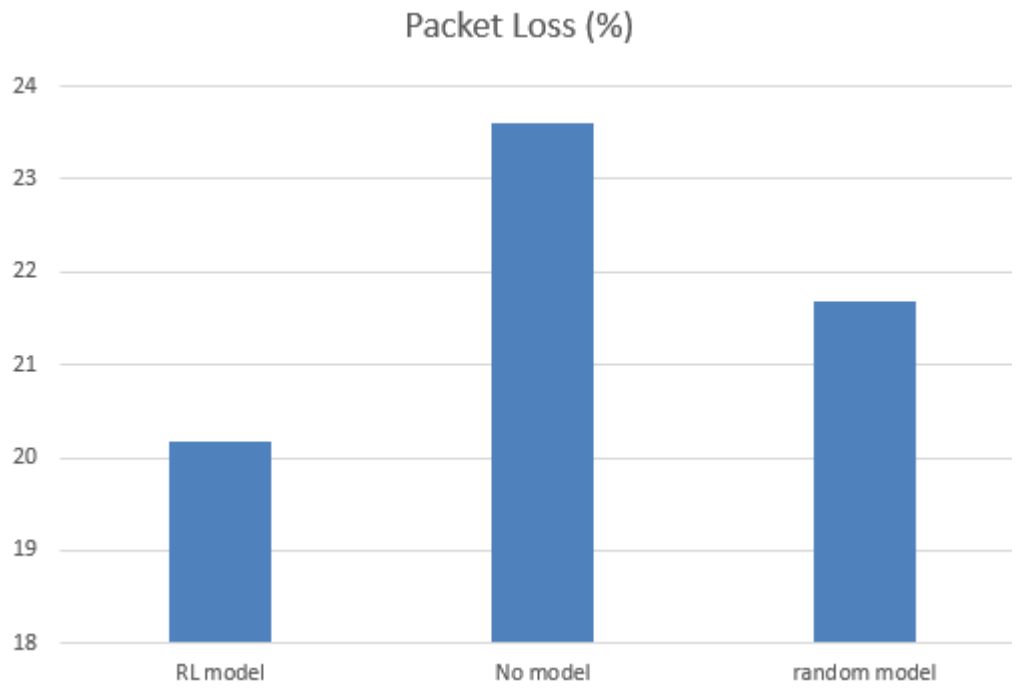


*Figure 18:*Comparison of the trained RL Agent with gamma=0.7 against using no model and using an agent chooses its actions in random (random model) on scenarios with speed up to 5 m/s, regarding the total network's packet loss rate.

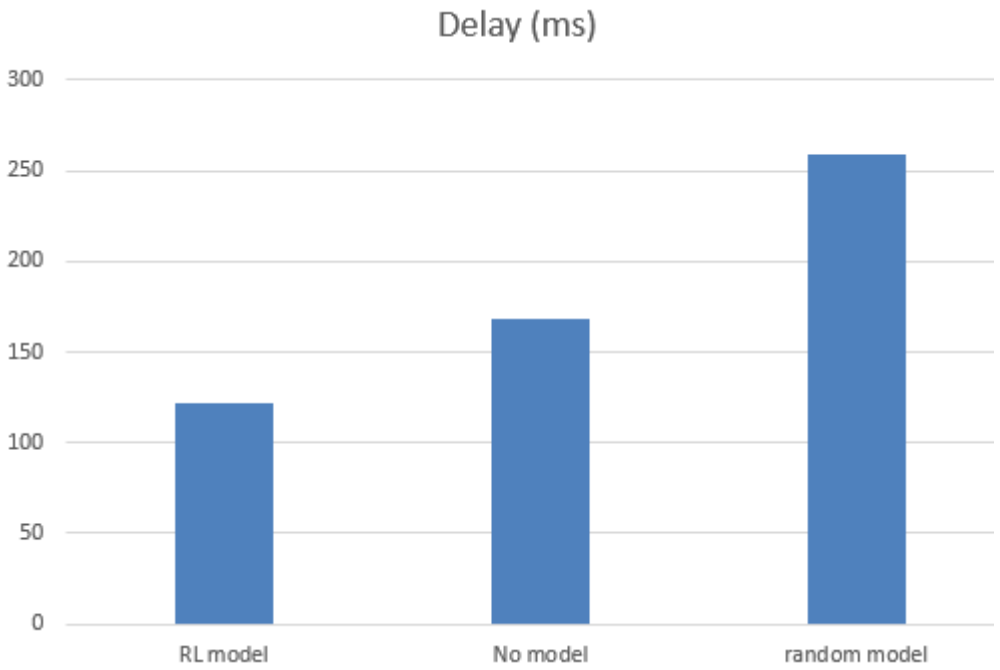


Figure 19: Comparison of the trained RL Agent with gamma=0.7 against using no model and using an agent chooses its actions in random (random model) on scenarios with speed up to 10 m/s, regarding the total network's delay

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

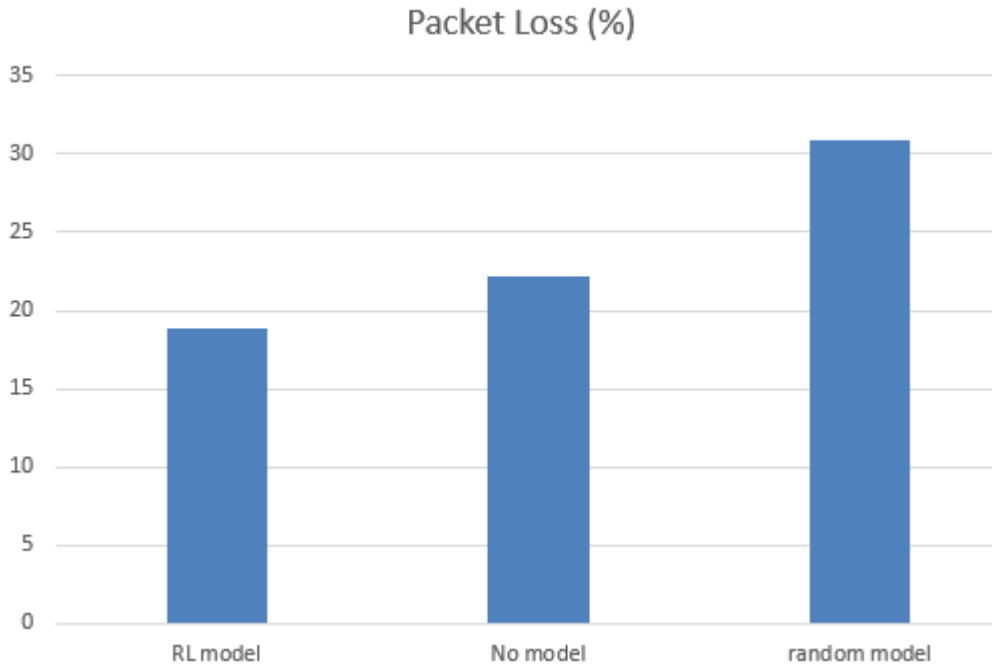


Figure 20: Comparison of the trained RL Agent with gamma=0.7 against using no model and using an agent chooses its actions in random (random model) on scenarios with speed up to 10 m/s, regarding the total network's delay.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

Chapter 6: Conclusion and Future work

In this thesis, the influence of parameters on the behavior/performance of an SDN-MANET system was extensively studied. Specifically, variables related to fault tolerance, namely heartbeat_interval and election timeout, and topology discovery mechanisms, HELLO_INTERVAL, were thoroughly examined. It was observed that depending on the node movement conditions in the topology, these variables negatively impact the overall network state. Based on these findings, an RL Agent was studied, designed, and implemented using Offline RL. Initially, the RL agent was trained on a dataset consisting of the nodes' observations, actions, and rewards. Subsequently, this trained RL agent was integrated into the SDN-MANET architecture and evaluated on multiple and diverse topologies based on speed and pattern mobility. The results demonstrate that the network performs better overall regarding the packet loss rate and the delay, when the RL agent is employed to predict the values of these variables for each individual node. However, there are new ideas that we believe will enhance this work and provide it with a new dynamic.

It is important to mention that data collection for constructing the dataset is time-consuming procedure. As it is known, Offline RL algorithms require a massive volume of data to be trained and achieve significant performance. This is because the agent cannot interact and learn from the environment in real-time (online training) but solely relies on the data provided by the dataset. As evident, the more data and diverse scenarios the dataset contains, the more suitable it is for the comprehensive training of the algorithm. Unfortunately, within the scope of this thesis, there was not the possibility to collect such a large amount of data. In addition, in the context of the proof-of-concept implementation, we have demonstrated that the use of an Offline RL agent can help and improve the SDN-MANET architecture. However, we consider the creation of a larger dataset will be a significant factor on contributing to the improvement of the RL Agent's efficiency.

Another objective is to conduct further research regarding the tuning of hyperparameters for the RL agent. The exploration and identification of the most suitable set of values for variables such as the number of neurons, learning rate, or target_update_interval is demanding and time-consuming. Therefore, it is something that we will be investigating to see if the model can be improved in its predictions.

Finally, one of the next steps in this work is the passive data collection method. Specifically, regarding the metrics of the packet loss ratio and the delay, the collection method, as described earlier in Chapter 4, is performed by executing the "fping" command and collecting the output. However, this is not a practical approach that could be applied in a real-world environment. On the contrary, our goal for the future is to implement the Precision Time Protocol (PTP) for synchronizing clocks in a network. In this way, we can record the timestamps of the control packets sent by the nodes in our observations. Control packets include LLDP packets, etcd packets, and OpenFlow packets. This approach allows us to avoid generating additional unnecessary traffic. Additionally, this approach highlights the need for the centralized approach that the system architecture has, as all the processing with the packet timestamps must be done by a central node that collects all the information. Otherwise, each node separately only has knowledge of the timestamps of the packets it has sent, which is not sufficient to calculate the delay and the packet loss rate. From the implementation perspective of such a protocol, the Precision Time Protocol daemon (PTPd) library [24] could be used, which achieves node synchronization within the topology through software. Alternatively, there is the option of using specialized hardware, such as installing GPS modules on the topology's nodes.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

References

[1]. Syrigos I., Koukoulis I., Prassas A., Choumas K., Korakis T., “On the Implementation of a Cross-Layer SDN Architecture for 802.11 MANETs”, ICC, 2023

[2]. Open vSwitch. "Open vSwitch." Open vSwitch, https://www.openvswitch.org/

[3]. Onaro D., Ousterhout, “In Search of an Understandable Consensus Algorithm”, USENIX Annual Technical Conference, 2014.

[4]. Open Networking Foundation. "OpenFlow Switch Specification Version 1.5.1." Open Networking Foundation, 2015. https://www.opennetworking.org/wp-content/uploads/2015/03/openflow-switch-v1.5.1.pdf

[5]. Ryu Project. "Ryu: SDN Framework for Python." Ryu Project, 2022. https://ryu-sdn.org/.

[6]. Sutton R. and Barto A., RL: An Introduction, 2nd ed. Cambridge: The MIT Press, 2015.

[7]. Levine S., Kumar A., Tucker G. and Fu J., "Offline RL: Tutorial, Review, and Perspectives on Open Problems", arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/2005.01643v3. [Accessed:22/02/2023].

[8]. Luong N. C.et. all, “Applications of Deep RL in Communications and Networking: A Survey”, IEEE Communications Survey & Tutorial, vol. 21, no. 4, 2019. [Accessed: 10/03/2022].

[9]. Mnih V. et al., "Human-level control through deep RL", Nature, vol. 518, no. 7540, pp. 529-533, 2015. Available: 10.1038/nature14236 [Accessed 22/08/2022].

[10]. Ahrenholtz J., “Comparison of CORE network emulation platforms”, Proceedings of the Military Communications Conference (MILCOM), 2010, DOI: 10.1109/MILCOM.2010.5680218

[11]. https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/CORE/ (Accessed: 30/05/2023).

[12]. EMANE tutorial: https://github.com/adjacentlink/emane-tutorial/wiki (Accessed: 30/05/2023).

[13]. Ahrenholz J., Goff T, Adamson B., “Integration of the CORE and EMANE Network Emulators”, Proceedings of the Military Communications Conference (MILCOM), 2011, DOI:10.1109/MILCOM.2011.6127585

[14]. Veyster L., Cheng B.N., Charland R., “Integrating radio to router protocols into EMANE”, Proceedings of the Military Communications Conference (MILCOM), 2012, DOI: 10.1109/MILCOM.2012.6415571

[15]. Paszke A. et al., "Automatic differentiation in PyTorch", OpenReview”, 2017. https://openreview.net/forum?id=BJJsrmfCZ. (Accessed: 30/05/2021).

[16]. Seno T., Imal M., "d3rlpy: An offline deep reinforcement library", The Journal of Machine Learning Research, Volume 23, pp 14205–14224, 2022.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27

[17]. "NumPy", https://numpy.org/ (Accessed: 30/05/2023).

[18]. Pandas - Python Data Analysis Library, https://pandas.pydata.org/. (Accessed: 30/05/2023).

[19]. etcd, etcd tuning parameters, https://etcd.io/docs/v3.4/tuning/

[20]. NS2 software simulator, https://www.tutorialsweb.com/ns2/NS2-1.htm accessed: (30/05/2023).

[21]. BonnMotion software tool, https://net.cs.uni-bonn.de/wg/cs/applications/bonnmotion/ accessed: (30/05/2023)

[22]. Kumar A., Zhou A., Tucker G., Levine S., “Conservative Q-Learning for Offline RL”, Advances in Neural Information Processing Systems 33, 2020

[23]. Agarap A.F., “Deep Learning using Rectified Linear Units (ReLU)”, 2018, accessed: (10/06/2023)

[24]. PTPd. "PTPd - Precision Time Protocol Daemon - Version 2.3.2." PTPd, 2022. http://ptpd.sourceforge.net/.

Institutional Repository - Library & Information Centre - University of Thessaly
27/07/2024 06:02:38 EEST - 3.143.25.27