technologies

MDPI

Article

Multi-Agent Reinforcement Learning Framework in SDN-IoT for Transient Load Detection and Prevention

Delali Kwasi Dake *, James Dzisi Gadze, Griffith Selorm Klogo and Henry Nunoo-Mensah

Faculty of Electrical and Computer Engineering, Kwame Nkrumah University of Science and Technology (KNUST), PMB, Kumasi AK-039-5028, Ghana; jdgadze@gmail.com (J.D.G.); klogoselorm@yahoo.com (G.S.K.); hnunoo-mensah@knust.edu.gh (H.N.-M.)

* Correspondence: dakedelali@yahoo.com

Abstract: The fast emergence of IoT devices and its accompanying big and complex data has necessitated a shift from the traditional networking architecture to software-defined networks (SDNs) in recent times. Routing optimization and DDoS protection in the network has become a necessity for mobile network operators in maintaining a good QoS and QoE for customers. Inspired by the recent advancement in Machine Learning and Deep Reinforcement Learning (DRL), we propose a novel MADDPG integrated Multiagent framework in SDN for efficient multipath routing optimization and malicious DDoS traffic detection and prevention in the network. The two MARL agents cooperate within the same environment to accomplish network optimization task within a shorter time. The state, action, and reward of the proposed framework were further modelled mathematically using the Markov Decision Process (MDP) and later integrated into the MADDPG algorithm. We compared the proposed MADDPG-based framework to DDPG for network metrics: delay, jitter, packet loss rate, bandwidth usage, and intrusion detection. The results show a significant improvement in network metrics with the two agents.

Keywords: MADDPG; SDN; IoT; routing; reinforcement learning; DDoS

Citation: Dake, D.K.; Gadze, J.D.; Klogo, G.S.; Nunoo-Mensah, H. Multi-Agent Reinforcement Learning Framework in SDN-IoT for Transient Load Detection and Prevention. *Technologies* **2021**, *9*, 44. https://doi.org/10.3390/technologies9030044

Academic Editors: Konstantinos Oikonomou and Vasileios Komianos

Received: 19 May 2021
Accepted: 23 June 2021
Published: 29 June 2021

1. Introduction

The next generation telecommunication network has successfully laid the backbone for the proliferation of massive IoT (Internet of Things) and its accompanying big, varying, and complex data [1]. 5G networks will alleviate the performance tradeoffs of IoT devices under earlier wireless technologies [2] and completely revolutionized a connected world. These IoT devices, according to Ericsson mobility report, will generate close to 163 zettabytes (ZB) of data before 2025 with 22.3 billion different applications [3,4]. This vast interconnection of things will challenge mobile network operator's capacity to maintain an agile network devoid of congestion, network intrusions, and aggregated flows.

There has been a rampant surge in zombie IoT devices where attackers connect remotely to the IoT devices and install malicious bots to disrupt the normal functionality of the compromised devices. Such attacks including DDoS, spoofing, eavesdropping, and jamming [5] have security implications on the privacy [6] of the IoT network, benign flash event, and bandwidth consumption in the network. The vicious DDoS attack targeted at servers with false requests to consume network bandwidth is similar to genuine traffic burst, and this flooding attack speedily depletes network resources and leads to packet loss [7].

Currently, network operators are turning to Software-Defined Networking (SDN) [8–10] to keep pace with the emergence of IoT and enable programmability at a central and distributed level. Network robustness, adaptation, and abstraction with open APIs (Application Programming Interfaces) are core to the separation of the control plane from the forwarding devices in the SDN paradigm. With such separation, the network can

be intelligently programmed to react positively to possible transient network loads and intrusions that could destabilize its functionality and operations.

With the advent of Machine Learning [11,12], systems automation through data analytics has seen immense applications by researchers. Machine learning algorithms have diverse features with pros and cons in producing accurate models for classification, regression, clustering, and behavioral learning. The most popular categorization of machine learning algorithms is supervised learning models [13,14]. Supervised learning models train agents with a labelled dataset. After enough training, the agent is fed with new data for identification and classification. Supervised learning presents an obvious disadvantage of cost in labelling the dataset properly. With variations in data from diverse applications, IoT also presents a difficulty in accurately classifying data. Unsupervised learning [15,16] is more about identifying relationships and patterns and less of automation. This type of learning involves a training with unlabeled data and no output. With unsupervised learning, there is no guidance on efforts to get the right groupings. Reinforcement Learning (RL) [17], as shown in Figure 1, is the real basis for automation in machines where agents take actions on the environment based on intelligent policies guided by reward systems. Agents accumulate knowledge on the environment with continuous trials and error of actions until they master the relevant actions in an exploitation–exploration tradeoff [18].

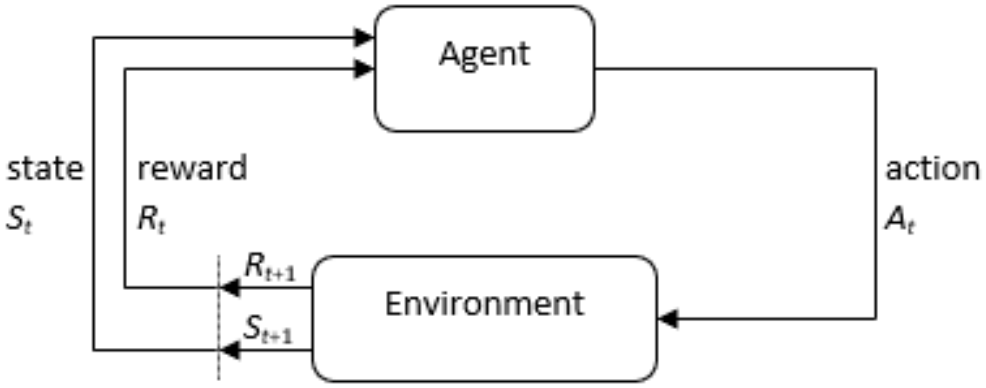


Figure 1. Reinforcement learning.

In maintaining efficiency in networks, RL agents have seen dynamic integrations in SDN-IoT domains due to the expected network flow complexities envisaged by network operators and the possible malicious intrusion from compromised IoT devices by attackers. The network must be protected from different attack loopholes [19] in the SDN architecture while engineering dynamic multi-path routing [20,21] strategies for packets from benign users. Deep Reinforcement Learning (DRL), which combines Deep Learning (DL) and Reinforcement Learning, has seen breakthroughs in complex game development, robotics, and network automation, among others, by using hidden layers to capture features and intricate details relevant to the RL agents [2,22]. The integration of DL in RL has also reduced the curse of dimensionality and improved the ability to solve high-dimensional problems [22]. The security of DL models and data privacy protection has been studied in [23,24] and remains a concern to service providers, especially with the massive connectivity of objects. These security threats can lead to inaccurate models for training the RL agents and adversely affect performance especially for time-bound applications. Compromises to data through network attacks [23,24] lead to the availability of sensitive information to hackers and the occurrence of this phenomenon will compromise the confidentiality of the system.

In this research, we propose a multi-agent RL framework in SDN for three use cases, as shown in Figure 2: traffic burst detection, elephant flow, and DDoS. We extended Deep Deterministic Policy Gradient (DDPG) in multi-agent environment, Multiagent Deep Deterministic Policy Gradient (MADDPG) algorithm, and used Markov Decision Process

(MDP) to mathematically model the state-action-reward representation. The rest of the article is organized as follows. Section 2 lays the foundation for single-agent RL (SARL) and multi-agent RL (MARL) in the SDN architecture. Section 3 reviews similar literature involving MARL. Sections 4–6 describe the system into details. Section 7 discusses the experimental results, and we conclude in Section 8.

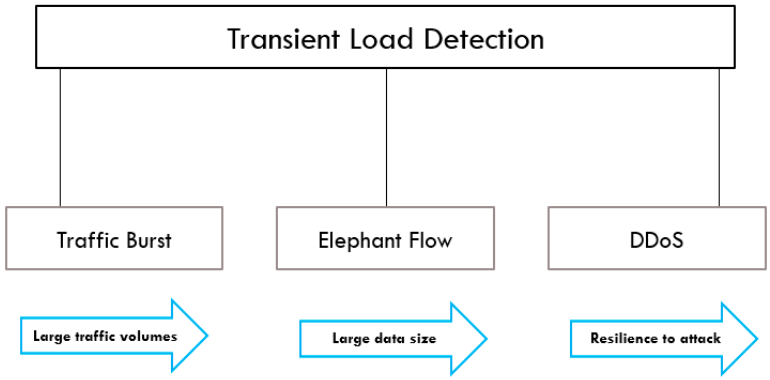


Figure 2. Transient load detection.

2. SARL and MARL

SARL and MARL are defined by the number of agents a reinforcement architecture deploys on the environment. Most research works for traffic engineering (TE) and intrusion detections (IDs) in SDN have adopted the SARL architecture. The common SARL implemented algorithms includes Q-learning [25], SARSA algorithm [26], the Deep Q-Network (DQN) [25], and the Deep Deterministic Policy Gradient Algorithm (DDPG) [27]. DQN and DDPG algorithms have seen the integration of deep neural networks (DNNs) in optimizing the policy network of agents for higher efficiency and scalability. As shown in Figure 3, for the SARL architecture, the SDN controller collects network metrices as state for the RL agent. The agent based on a policy enforces an action on the forwarding devices through the controller and subsequently receives a reward. The SDN controller then installs new flow rules for the OpenFlow [28] switches in the forwarding plane.

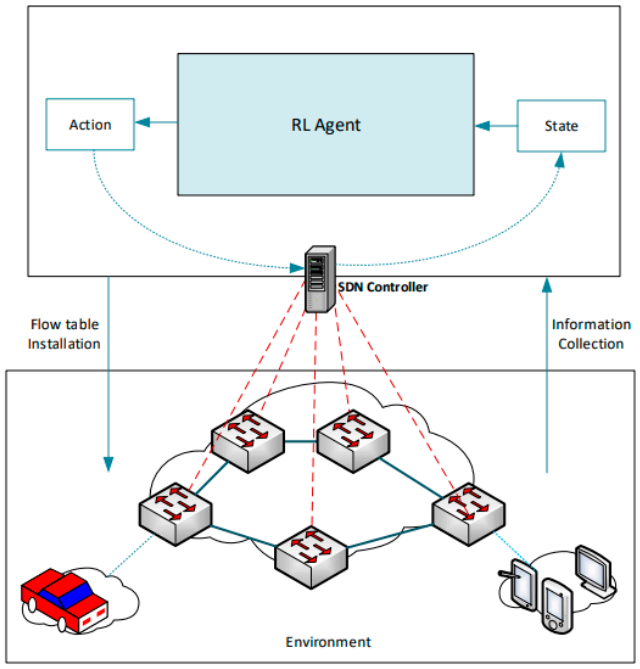


Figure 3. SARL.

With collaboration comes faster convergence in a cooperative and competitive environment among agents. Agents' cooperation increases the state space representation with exhaustive dimensionality, resulting in faster executions through parallel computation in the environment. Experience sharing and cooperation in IoT-related dynamic environment have necessitated a research discussion surge from SARL to MARL, especially with the advent of 5G and newer technologies.

MARL involves interactive, autonomous systems acting on a common environment to accomplish a task within a shorter time. These lead to agents executing decentralized actions with centralized training. MARL systems have seen applications in diverse domains including game development, network management, distributed control, and decision support systems [29–31].

As shown in Figure 4, multi-agents execute decentralized action on the stochastic environment using respective actor networks. A historic tuple using the replay buffer is used to store the state-action-reward of each agent after every episode. The centralized critic network is then used to evaluate the actions of the actor network of each agent through a policy gradient.

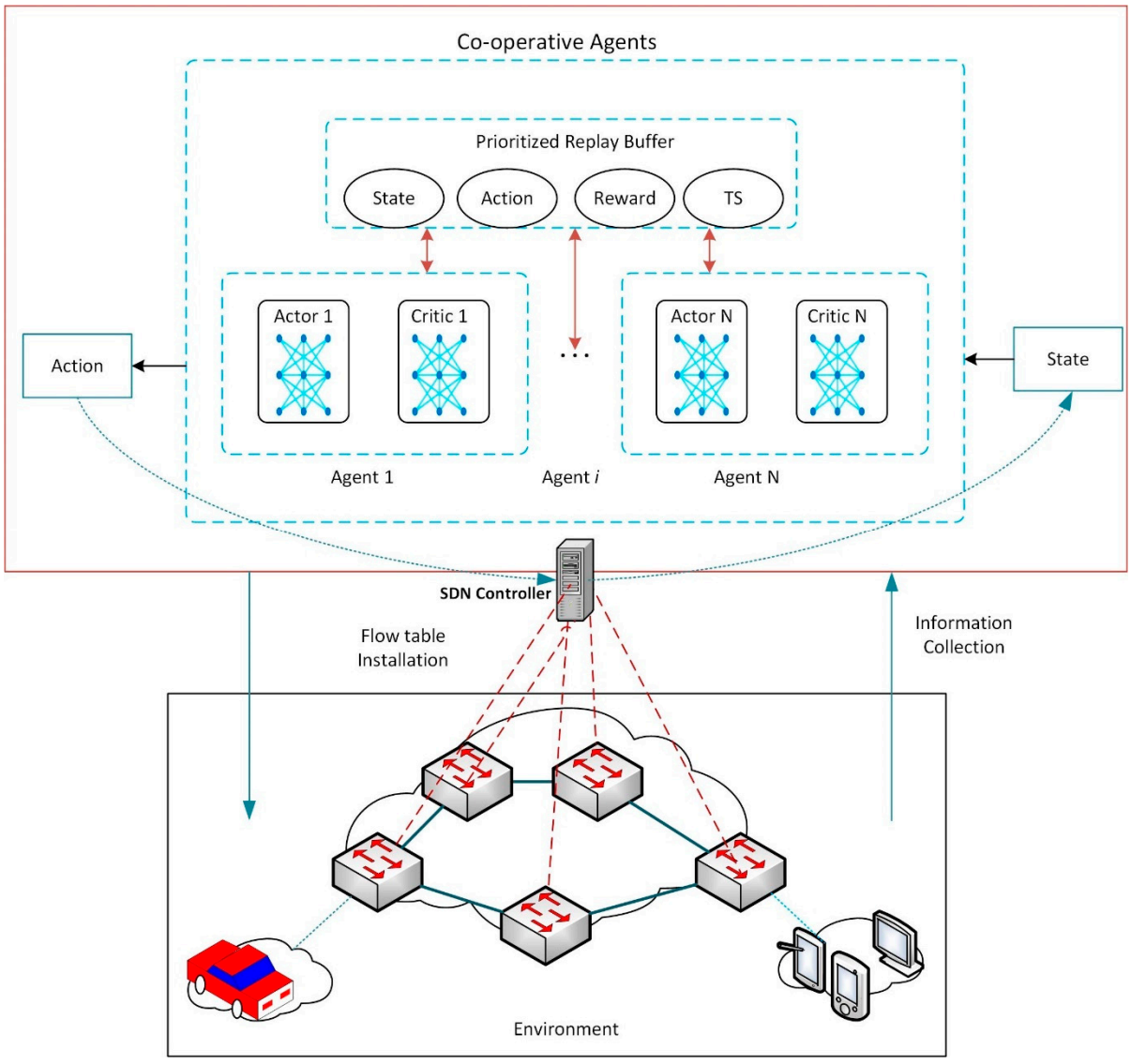


Figure 4. MARL.

3. Review of Similar Work

For general threat mitigation in SDN environment, [32] proposed a SARL ATMoS agent using Q-learning algorithm. The malicious host (MH) implemented in this framework is an Advanced Persistent Threats (APTs) vector, which uses dynamic attack vectors, tactics, and tools to evade detection. The RL agent operates in the controller and relies on a network observer module installed in forwarding devices for autonomous management and a reward feedback from the environment using a host behaviour profiling. Results show that the model using the SYN flood experiment converges faster after 150 iterations with 1 MH and 1 BH (Benign Host), while the APT technique converges after 1400 iterations. Adopting a proactive defence strategy in SDN, [20] proposed a security-awareness Q-learning Route Mutation (SQ-RM) scheme with adaptive learning rate adjustment for faster convergence. The controller is implemented as the RL agent and acts as the defender against threats. Four different attack methods, node degree, critical edge, multi-target coordinated, and prior experience, were investigated and compared. Results show that SQ-RM is not effective against edge attack strategies since delay increases in the network when implemented.

In dealing with DoS attacks, [33] proposed an RL-based Q-MIND algorithm that utilizes Q-learning in an SDN environment to mitigate stealthy attacks. The Q-MIND DoS detection scheme is implemented in the SDN application plane and communicates with the controller via the Northbound API. With 4 benign and 4 malicious hosts, the Q-MIND was compared to support vector machine (SVM) and Random Forest (RF) classifiers after 4000 normal and 4000 attack samples. Results show that Q-MIND performs better in detecting DoS after 100 iterations when compared with SVM and RF fixed classification algorithms. However, prior to the 100 iterations, SVM and RF perform better. To improve the research findings, [33,34] proposed a Deep Deterministic Policy Gradient (DDPG) algorithm implemented as a policy network in a RL mitigation agent to learn network flow patterns and throttle malicious TCP, SYN, UDP, and ICMP flooding attacks. The proposed framework results were compared to AIMD router throttling and CTL approaches for DDoS detection after 20,000 episodes with 10 s episode interval. Agent reward increased throughout the episode indicated by the throttling of most malicious traffic from attack nodes. The framework outperformed the AIMD router throttling approach by 3–9% and CTL approach by 18–28%.

As SDN-IoT size increases, network attack becomes prevalent. To improve network security while optimizing routing, [35] proposed a deep reinforcement learning-based quality-of-service (QoS)-aware secure routing protocol (DQSP). Five attack nodes were used to generate gray hole and DDoS attack methods with convolutional neural network (CNN) as the training network for the RL agent. The results were compared with OSPF protocol under three performance metrics: end-to-end delay, packet delivery ratio, and routing optimization probability in attack nodes. DQSP has a 10% increase in packet delivery ratio over OSPF under gray-hole attack and a slightly lower percentage under DDoS attack. The DQSP also outperforms OSPF for end-to-end delay and guarantees network QoS with earlier identification of attack nodes compared with OSPF.

To improve flow matching in SDN-based networks, [36] proposed a Q-learning integrated Q-DATA framework that uses support vector machine (SVM) algorithm to detect the performance status of forwarding devices. The Hping3 tool is used in 5 hosts to generate random traffic between destination servers and hosts. To enhance the use cases of the Q-DATA, Self-Organizing Map algorithm is implemented on top of the ONOS SDN controller to detect malicious traffic. Results from the simulation show that, with a high load to the OpenFlow switches, the Q-DATA framework outperforms FMS, DATA, and MMOS schemes and delivers efficient and appropriate flow policies in the network.

For energy efficient routing in SDN, [37] proposed a deep Q-network (DQN)-based Energy Efficient routing (DQN-EER) algorithm to find energy-aware data paths between OpenFlow switches. The RL agent is implemented with deep convolutional neural network and modelled with MDP to interact with the environment. Three controllers with 16 hosts and 20 switches were used to set up the network topology. After 2300 training episodes, the

DQN-EER was compared with CPLEX and CERA algorithms. Results show that DQN-EER outperforms CERA and CPLEX for energy-efficient routing when the state becomes larger.

In optimizing network routing, [38] proposed a DRL agent targeted at optimizing network delay in SDN environment. The proposed set up involves a network topology of 14 nodes with 10 traffic intensity that ranges incrementally from 12.5% to 125% relative to the aggregated network capacity. With 1000 distinct traffic configurations, the DRL agent was trained with 100,000 step traffic matrixes. Results show that, as the training time increases, the DRL agent's performance also increases and improves the overall network delay.

Yuan et al. [39] proposed a MADDPG integrated algorithm in multi-agents with cooperation to solve the internet of vehicles (IoV) dynamic controller assignment problem while minimizing network delays and packet loss. The architecture is deployed in the data and the control plane of the SD Networks. The data plane is split into two sub-layers: mobile and stationary layer. The mobile layer has on-board units integrated in moving vehicles for communication with roadside units (RSUs) and base stations (BSs). The control plane has a hierarchical distribution with multiple edge controllers at the lower tier with the top tier core controllers in remote data centers.

Wu et al. [40], in minimizing packet loss rate and increasing packet throughput in a network, proposed a MADDPG based traffic control and multi-channel reassignment (TCCA-MADDPG) algorithm in SDN-IoT. The heterogeneous IoT structure consisting of LTE network, WIFI network, and 3GPP network forms the edge layer in the architecture as layer three. The fourth layer, which consists of IoT devices, is the fog layer. The second layer is the core backbone layer that bridges the third layer to the first layer, the data center layer.

Yu et al. [41], in optimizing routing in SDN, proposed a DDPG Routing Optimization Mechanism (DROM) framework using DRL agent integrated within the control layer. The DROM agent uses a Traffic Matrix to receive the current network load state [10] s from the environment and changes the weight links of the network through an action a. The DROM agent receives a reward r, and the state changes from s to s'.

4. Main Concept

We present a novel MARL system implemented with MADDPG algorithm for traffic burst, elephant flow, and DDoS detection and prevention in an SDN-IoT environment. With compromised IoT devices, it has become extremely difficult to differentiate genuine transient or crowd event packets from the compromised malicious packets emanating from zombie IoT devices. The various attack methods such as SYN, TCP, ICMP, and UDP flooding [42] mimic genuine sudden spike traffic, and, if not blocked intelligently, will result in genuine packet loss to destination nodes. The primary concept of the framework, as shown in Figure 5, involves the implementation of two agents in the controller of the SDN architecture, agent 1 and agent 2. Agent 1 is responsible for genuine traffic burst and elephant flow multipath routing in the network while agent 2 is responsible for DDoS detection and blocking. During a packet miss from the OpenFlow switches, a packet-in message request is sent to the controller for new flow addresses to be installed in the flow table. The two agents instruct the SDN controller in installing new flow rules during packet-out messages to the OpenFlow switches.

The SDN controller executes the instructions of the multi-agents through actions in the flow table of the OpenFlow switches, as depicted in Figure 6.

With the proposed concept, each agent has a policy in updating the controller on the actions to execute on the forwarding devices based on the reward received from the environment. The reward gives an indication of the next action each agent should take. As the episode progresses, agents learn to take the best action that will result in higher rewards in an exploitation–exploration tradeoff.

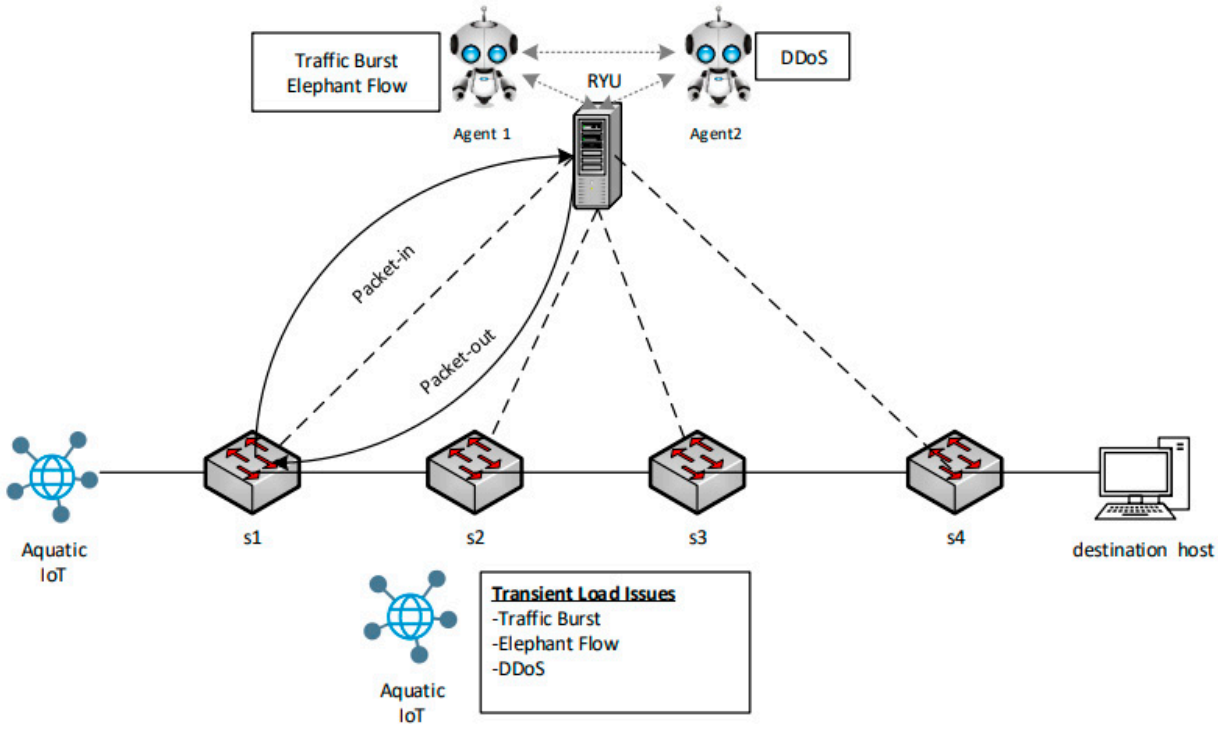


Figure 5. Proposed MARL MADDPG concept.

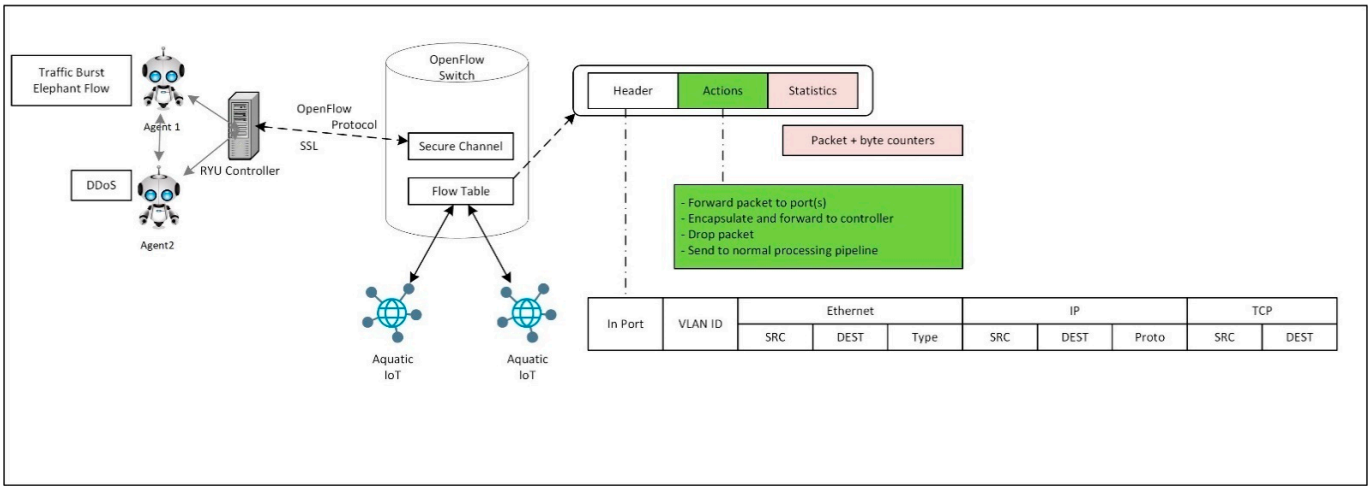


Figure 6. Proposed concept—OpenFlow switches.

5. Proposed MADDPG Framework

In this section, we proposed our novel MARL SDN-IoT framework with the objective to optimize the network through multipath routing while blocking DDoS traffic. The proposed architecture is the first of its kind in an SDN that utilizes two agents in a cooperative environment integrated with the MADDPG algorithm.

The MADDPG integrated MARL architecture, as shown in Figure 7, has an RL set up consisting of two main parts: two intelligent agents in the controller and an observed environment comprising data forwarding devices and IoT devices. The design is modelled as an MDP with a state space, an action space, and a reward function. In Algorithm 1, state S in MDP is defined as essential features in the agent's environment that are of interest. The state consists of a finite set $\{s^1, \ldots, s^N\}$ where $|S| = N$. N is the size of the state space. The action, A defined as a finite set $\{a^1, \ldots, a^K\}$ where $|A| = K$ is applied in the environment

to control the system's state to a new state. K represents the size of the action space [43]. With possible transitions, an action $a \in A$ in a state $s \in S$ changes the state from s to a new state $s\prime \in S$. This results in a transition state of $P(s, a, s\prime)$. The reward function is based on the action of the agent in a state. $R\ (s,\ a,\ s\prime)$ represents the reward after transition from s to a new state $s\prime$. A discount factor γ determines the agent's action that leads to distant future rewards relative to immediate rewards. A discount factor should range between $0 \leq \gamma < 1$.

Algorithm 1 Markov Decision Process (MDP) [44]

An MDP is a 5-tuple $(S,\ A,\ P,\ R,\ \gamma)$, where;
S is a set of states
A is a set of actions
P $(s,\ a,\ s\prime)$ is the probability that action a in state s at time t
will lead to state $s\prime$ at time $t+1$
R $(s,\ a,\ s\prime)$ is the immediate reward received after a transition
from state s to $s\prime$, due to action a
γ is the discounted factor, which is used to generate a discounted
reward

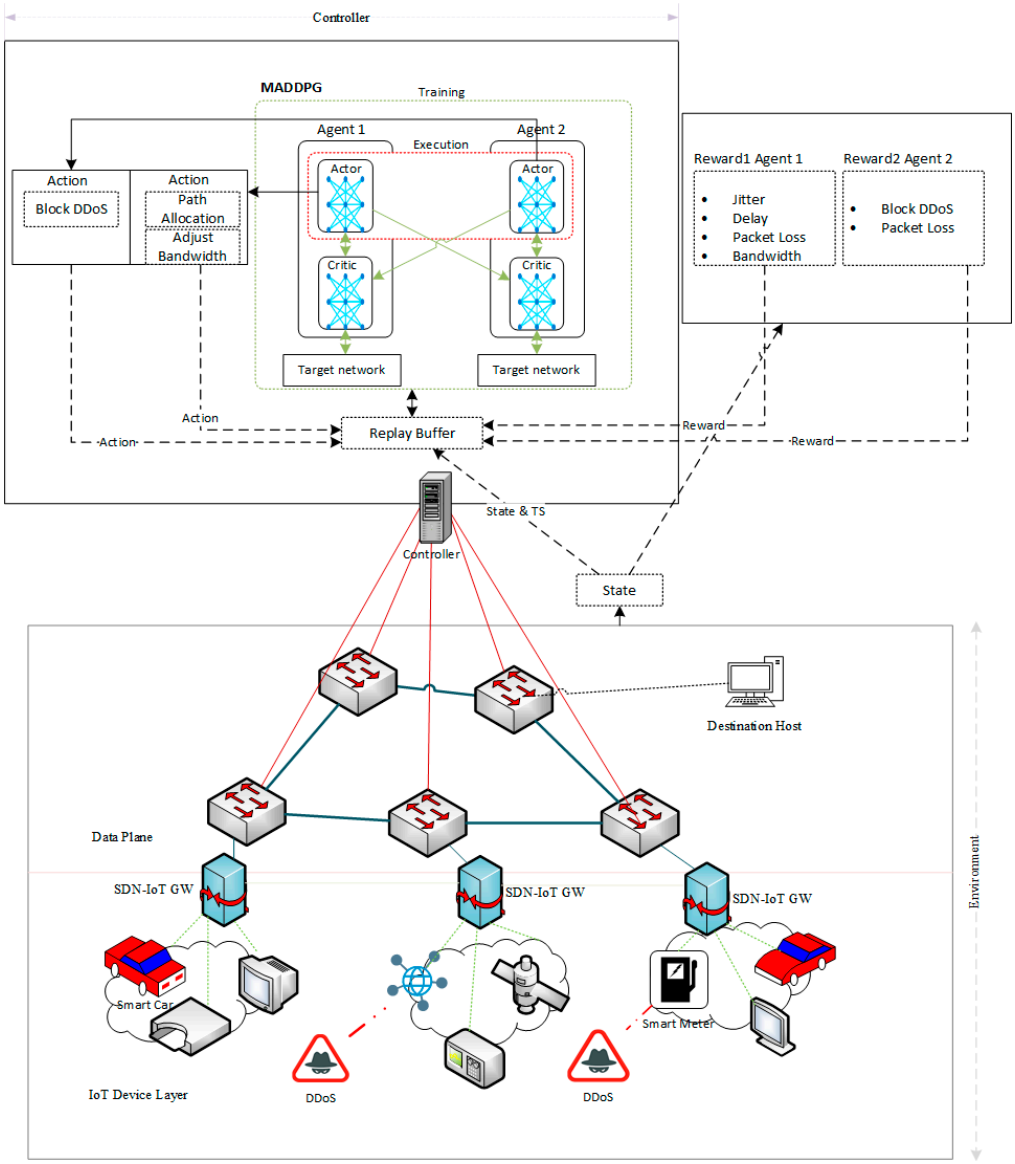


Figure 7. Proposed MARL MADDPG framework.

The environment, as shown in Figure 7, has an SDN-enabled network consisting of OpenFlow switches that relate statistics and forward data flows of the network under the instructions of the SDN controller. The IoT devices, as part of the environment, generate data packets and elephant flows to destination nodes via the SDN-IoT gateway and forwarding devices. The environment also consists of zombie or compromised IoT devices that generate malicious packets to random destinations in the network.

The two agents interact with the environment using actions that alter the state of the environment and receive rewards for their respective actions. The MARL agents execute the actions via the SDN controller, which has the responsibility to make decisions, collect network statistics, and enforce agent's actions by updating the flow table of the OpenFlow switches. Agent 1 is responsible for multipath routing of genuine transient packets and elephant flows in the network, while agent 2 is responsible for identifying and blocking DDoS flows. As depicted in Figure 8, the MARL MADDPG flow diagram illustrates the monitoring of network metrics and the action enforcement in the network through the RYU controller, an open source SDN controller [45].

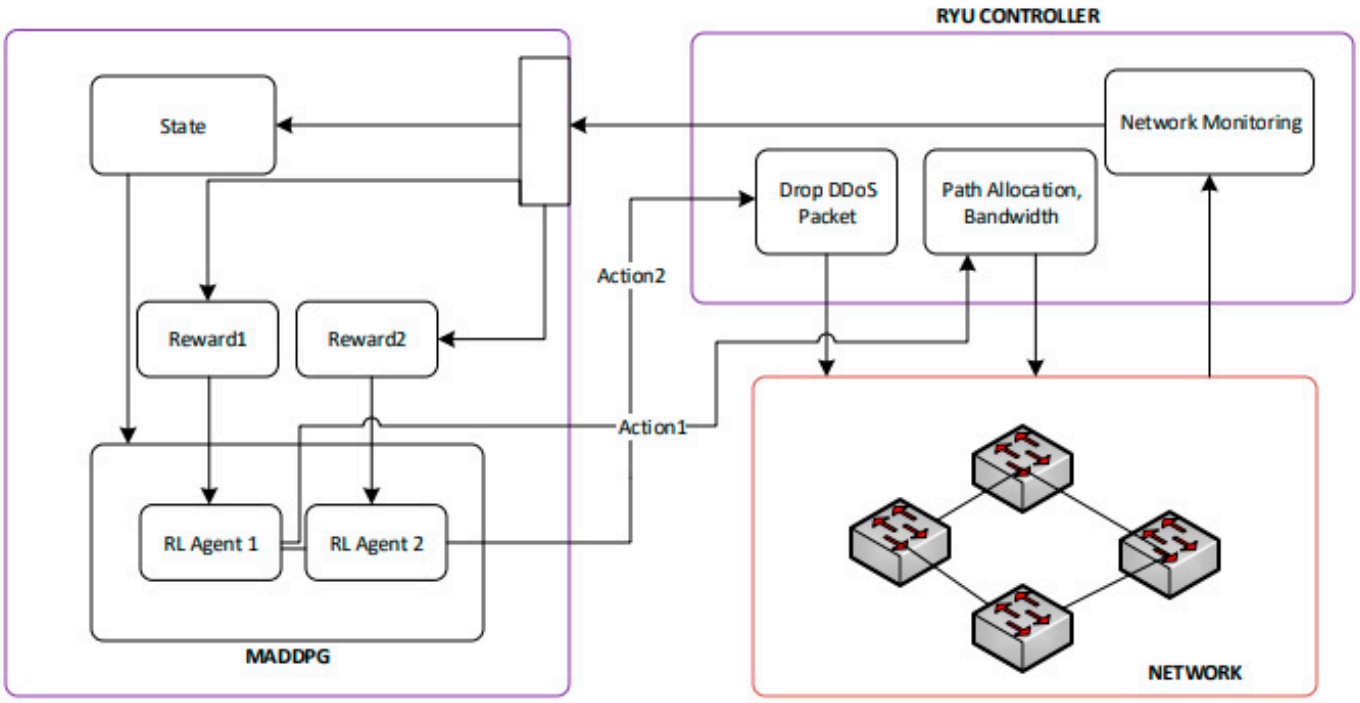


Figure 8. MARL MADDPG flow diagram.

The state space defines the environment and refers to the state of flows in the network. The state space s(t) is defined as a traffic matrix at time step t TM_t. An OpenFlow switch k_i $i \in (1, 2, 3, \ldots\ldots.. N)$ in time slot t, $t \in (1, 2, 3, \ldots\ldots\ldots\ldots T)$ consists of:

A. Data link occupancy rate between the flow and the OpenFlow switches $Bk_i(t)$

B. Channel link occupancy between packet-in messages from the OpenFlow switches $Gk_i(t)$

C. Channel link occupancy between packet-out messages to the OpenFlow switches $Ck_i(t)$ Equation (1):

$$State\ Space \qquad TM_t \cong \begin{bmatrix} Bk_1(t) & Bk_2(t) & \ldots & Bk_N(t) \\ Gk_1(t) & Gk_2(t) & \ldots & Gk_N(t) \\ Ck_1(t) & Ck_2(t) & \ldots & Ck_N(t) \end{bmatrix} \quad (1)$$

Each agent takes a decentralized action from the state using a policy. The respective action of the agents accumulates to global rewards. Agent 1 takes action based on genuine traffic burst and elephant flow and assigns the next hop to available OpenFlow switches during packet-out messages. Agent 1 also increases the bandwidth of the flow within the network demand range. Equation (2):

$$\textit{Action Space} \qquad a_{agent1}(t) = \left[a^{avail}_{k_{1\ldots n}}(t),\ a^{Bincrease}_{k_{1\ldots n}}(t) < TH\right] \tag{2}$$

$k_{1\ldots n}$ = switch 1 to n that are available as next hop for packet-out messages
TH = Bandwidth Threshold

Agent 2 takes an action involving dropping of packets transmitted beyond a rate to the controller. These lead to packet drop rule for forwarding switches Equation (3):

$$\textit{Action Space} \qquad a_{agent2}(t) = \left[a^{drop^{flow\_freq_i}}_{k_{1\ldots n}}(t) > RR\right] \tag{3}$$

$k_{1\ldots n}$ = switch 1 to n that send the packet-in messages
$a^{drop^{flow\_freq_i}}_{k_{1\ldots n}}(t)$; Drop flow from node i if the frequency is beyond threshold
RR = *Packet Transmission Threshold*

Based on the current state and action, each agent receives a local reward from the environment. The reward is an indication of the policy taken by the agents and subsequent refinement of policies. Each local reward aggregates to a cumulative global reward by the MARL system. Agent 1 is measured by a link utilization reward. The link utilization is a measure of delay, packet loss rate, bandwidth, and jitter metrices in the network Equation (4):

$$\textit{Reward Function} \qquad R_{agent1}(t) = \frac{1}{U} \tag{4}$$

$R_{agent1}(t) = \frac{1}{U}$, a stepwise decrease in link utilization, U gives +1 reward, and a stepwise increase gives -1 reward.

A successful packet dropping due to DDoS attracts +1 reward. Packet loss in the network attracts -1 reward Equation (5):

$$\textit{Reward Function} \qquad R_{agent2}(t) = \begin{cases} 1, & flow\_freq_i > RR\ !\ = k_{1\ldots n} \\ -1, & flow\_freq_i \cong loss \end{cases} \tag{5}$$

$flow\_freq_i > RR\ !\ = k_{1\ldots n}$, when a detected DDoS packet based on the rate of flow is not installed in the packet-out switches $k_{1\ldots n}$

$flow\_freq_i \cong loss$, when flow frequency from node i results in a packet loss.

After defining the state, action and reward function of the MARL system, we modify Algorithm 2 [46] using MDP representation and implement the new algorithm.

Algorithm 2 Proposed MADDPG [46]

1: **for** episode = 1 to M **do**
2: Initialize a random process $\mathcal{N}$ for action exploration
3: Receive =
$[\,Bk_1(t), Bk_2\,(t), \ldots\ldots\ldots Bk_N(t);\ Gk_1(t),\ Gk_2(t), \ldots\ldots\ldots\ldots Gk_N(t);\ Ck_1(t),\ Ck_2(t), \ldots\ldots\ldots\ldots Ck_N(t)\,]$
4: **for** $t = 1$ to max-episode-length **do**
5: for agent 1, select action $a_{agent1}(t)$
6: Executive $a_{agent1}(t) = [a^{avail}_{k_{1\ldots n}}(t),\ a^{Bincrease}_{k_{1\ldots n}}(t) < TH]$and observe $R_{agent1}(t) = \frac{1}{U}$
for agent 2, select action $a_{agent2}(t)$
Executive $a_{agent2}(t) = [a^{flow_i}_{k_{1\ldots n}}(t) > RR]$ and observe
$R_{agent2}(t) = \begin{cases} 1, & flow_i\,(t) > RR\,!= k_{1\ldots n} \\ -1, & z_i \cong loss \end{cases}$ and new state x'
7: Store $x, a_1, r_1 x' \ldots\ldots x, a_2, r_2 x'$ in replay buffer $\mathcal{D}$
8: $x \leftarrow x'$
9: **for** agent $i = 1$ to 2 **do**
10: Sample a random minibatch of S samples ($x^j, a^j_1, r^j_1, x'^j \ldots x^j, a^j_2, r^j_2, x'^j$) from $\mathcal{D}$
11: Set $y^j = r_i^j + \gamma Q_i^{\mu'}(x'^j, a'_1, a'_2)|a'_k = \mu'_k(o_k^j)$
12: Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j (y^j - Q_i^{\mu}(x^j, a^j_1, a^j_2))^2$
13: Update actor using the sampled policy gradient:
$$\nabla_{\theta i} J \approx \frac{1}{S}\sum_j \nabla_{\theta i}\mu_i\,(o_i^j)\nabla a_i Q_i^{\mu}(x^j, a^j_1, a^j_2)|a_i = \mu_i(o_i^s)$$
14: **end for**
15: Update target network parameters for each agent i
$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$
16: **end for**
17: **end for**

6. MADDPG and Deep Neural Network (DNN)

MADDPG is an extension of DDPG in multi-agents, which is an actor-critic algorithm used for continuous variables. With MADDPG, each agent has a DDPG algorithm with a centralized training network, the critic network. Each tuple transition of state-action-reward is stored in the Replay Buffer and used as the experienced buffer for the learning experiences in each batch for exploration. Each agent has two main networks, the primary and the target network, as shown in Figure 7.

Each agent has a primary network. It comprises of the actor primary network (APN) and the critic primary network (CPN). The APN is a DNN used to explore the policy using state-action pairs. The input to the DNN is the current state and the output is an action. The CPN estimates the performance of APN and provides a critic value used to train the APN to learn the gradient of the policy. The input of the CPN is the state and the current action of the APN. Its output is a critic value. With MADDPG, the critic network of each agent has full visibility of the actions and observations of the other agent for centralized training.

Each agent has a target network used to train the CPN. In training the CPN, the loss function comprising the critic target network (CTN) value, the reward function, and the CPN value is used. Unlike the primary network, the target network comprising actor target network (ATN) and CTN uses the transformed state (TS) of the state transition rule. The input to the ATN is therefore TS and not the current state. The target network updates itself using soft update values, which slowly updates the primary network.

7. Experimental Results and Performance Evaluation

In this section, we evaluate the performance of our model with the experimental set up and discuss the results. The setup consists of the tools, parameters, and hyper-parameters used for the RL agents.

7.1. Experimental Set up

The 20 and 60 nodes network topologies were implemented using TensorFlow and Keras running on Ubuntu-18.04 LTS (64 bit) operating system with an Intel® Pentium® CPU G2030 @ 3.00 GHz and 8 GB RAM capacity. The software requirement also includes Mininet-2.0 and Python-2.7.

From Table 1, we used RYU controller with 5 and 8 OpenFlow switches. The bandwidths for data link and channel link occupancy rates are 50 Mbps and 100 Mbps, respectively. The Transmission Control Protocol (TCP) is used to send benign traffic across the network with a sending rate of 2 Mbps. The tail drop queue management is used for dropping packets.

Table 1. Network parameters.

Scenario 1	No. of devices	20
	No. of controller	1
	No. of switches	5
Scenario 2	No. of devices	60
	No. of controller	1
	No. of switches	8
Bandwidth	50–100 Mbps	
Traffic Type	TCP	
Queue Type	Tail Drop	
Transmission Rate	2 Mbps	
SDN Controller	Ryu	

A fixed packet size of 56 bytes is randomly generated using Python library scapy, at an interval time of 0.1 s. The IPv6 protocol with a processing delay of 10 µs is used as the communication protocol to identify devices on the network.

DDoS packets are randomly generated from source nodes to random destinations at an interval of 0.02 s and restricted to only UDP flooding. The elephant flow is generated with a bandwidth range of 10 Mbps–1000 Mbps, at an interval of 0.1 s.

After multiple experiments, the actor learning rate α for agent1 and agent 2 is set to 0.0001. The actor discount factor γ for agents 1 and 2 is set to 0.99. The actor target soft update τ is set to 0.001, while the batch size is set to 64.

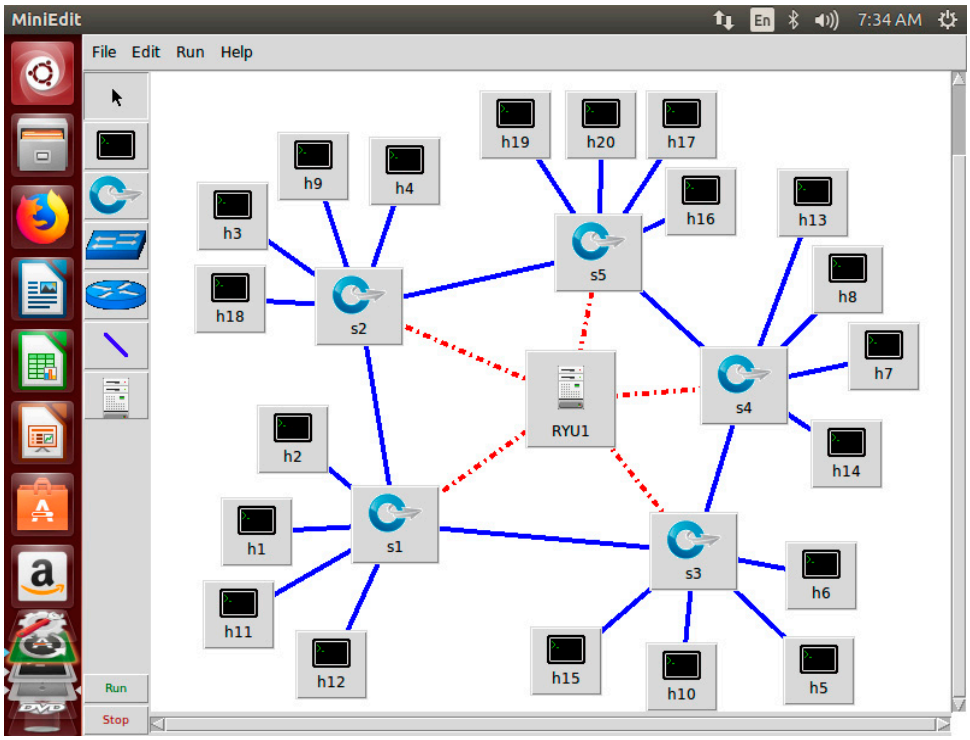
The critic agent parameter has the same set-up as the actor agent parameter with an $\varepsilon - greedy$ exploration of 0.1. Agent 1 and Agent 2 are set up with (300, 600) hidden units and neurons. For the actor network, the Deep Neural Network (DNN) hidden layer activation function for agent 1 and 2 is set to relu while the output layer is set to sigmoid. The critic DNN for agent 1 and 2 has a state hidden layer 1 activation function set to relu, action hidden layer 1 set to linear, state hidden layer 2 set to linear, action hidden layer 2 set to relu, and output layer activation function set to linear.

7.2. Experimental Results

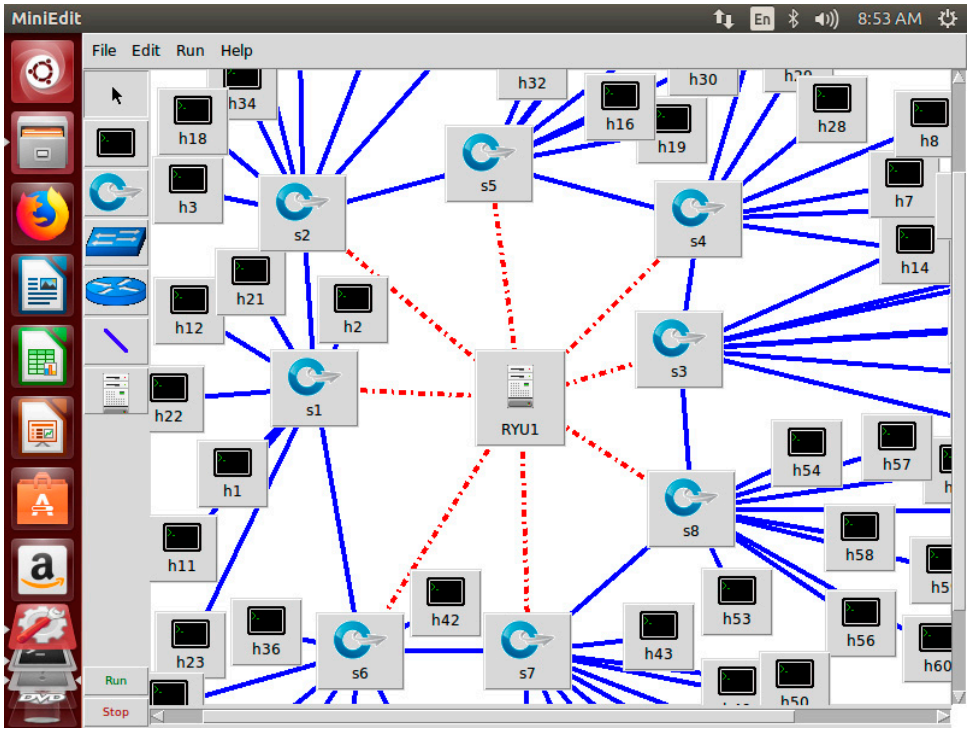
In This subsection, the performance metrics considered to evaluate the MADDPG algorithm and DDPG are discussed and the results analyzed. The Mininet simulations for 20-nodes and 60-nodes network topology are shown in Figure 9.

7.2.1. Reward

The reward is used to measure the results of the obtained action to a respective corresponding state that is perceived from the environment. In DRL, agents perform an action and obtain a reward based on the network policy. Average global reward is computed for 3000 episodes.



(**a**) 20 nodes-5 switches



(**b**) 60 nodes-8 switches

Figure 9. MADDPG set up in Mininet.

7.2.2. Jitter

Jitter in SDN represents the small intermittent delay in data transmission. It is caused by a number of factors including network congestion, interference, and collisions. Jitter represents the variation in the delay of packets transmitted from the source to destination nodes.

7.2.3. End to End Delay

Transmission delay refers to the time taken for data packets to move from source to destination nodes in the SDN network.

7.2.4. Packet Loss

Packet loss ratio is calculated as the difference between the number of packets received over the number of packets sent in the network. If the destination does not receive the packets, then the packet loss ratio is high; otherwise it is reduced.

7.2.5. Bandwidth Usage

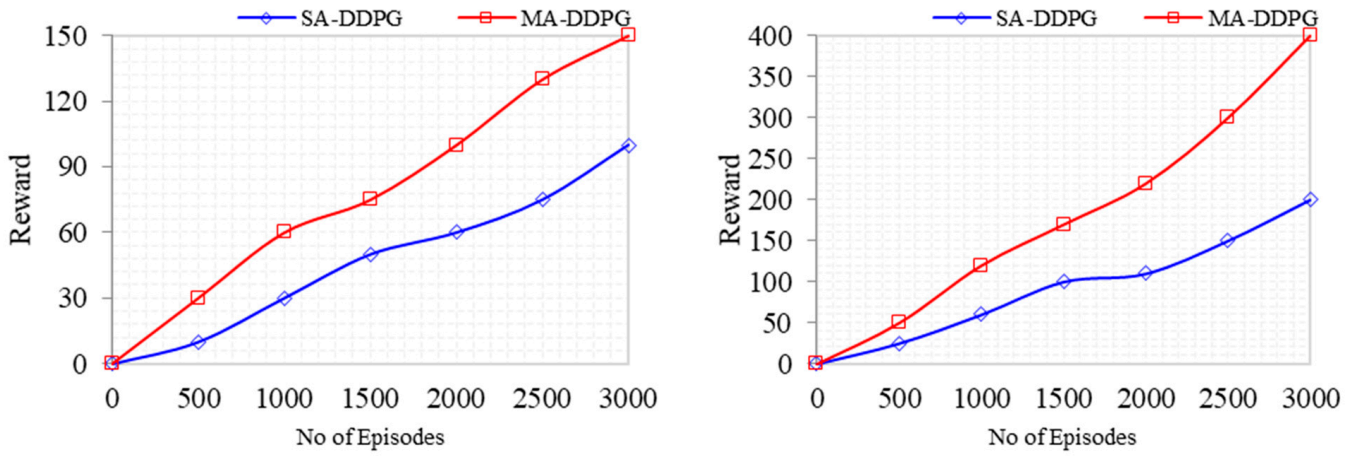
It is estimated in bits per second, and a higher bandwidth helps more data to pass through it. This metric also evaluates how the network allocates the bandwidth for data transmission. When the optimum bandwidth is allocated, then the data transmission is efficient and reliable. To provide scalable and reliable services for current network traffic beyond improving network bandwidth, the system must provide a flexible mode for traffic management.

7.2.6. DDoS Detection Rate

The detection rate of DDoS detection is defined as the number of packets successfully predicted as intruder packets.

7.2.7. Global Reward

As shown in Figure 10, after 3000 episodes, MADDPG gained a 25% cumulative global average reward over DDPG for the 20-nodes topology. The least reward gap for MADDPG was obtained from episode 1 to 1500 at 9%, while the later episodes gave a 16% reward. Due to exploration, the reward gap decreased along the episodes. For MADDPG, the average reward gap decreased from episode 1000 to 1500, while DDPG had a minimum decrease from episode 2000 to 2500. For the 60-nodes topology, the average global reward of MADDPG increased to 33% for the 3000-episode state-action transitions. MADDPG obtained the least average reward of 8% from episodes 1 to 1500, with a 24% reward for the later episodes. DDPG obtained an average least incremental reward gap from episodes 1500 to 2000, while MADDPG's least incremental gap was obtained from episode 1000 to 1500 and episodes 1500 to 2000.

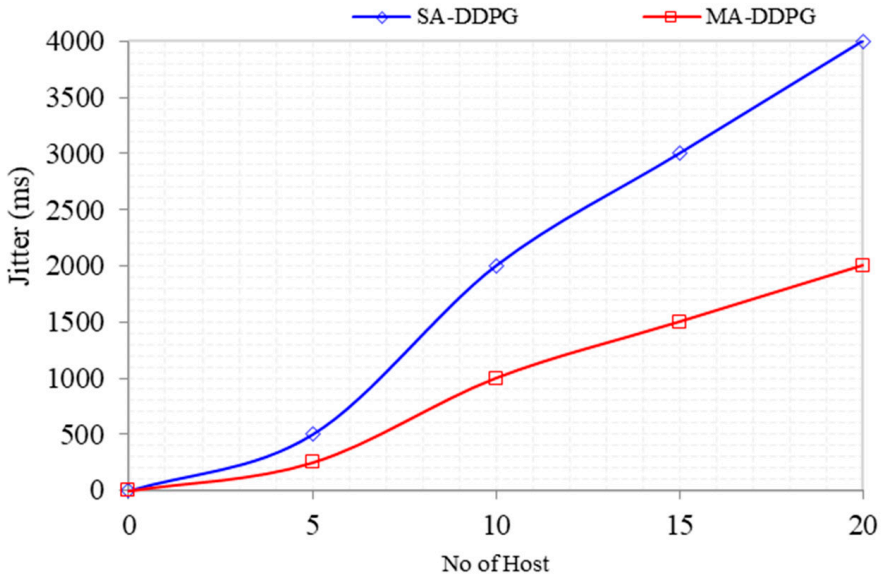


(**a**) 20 nodes (**b**) 60 nodes

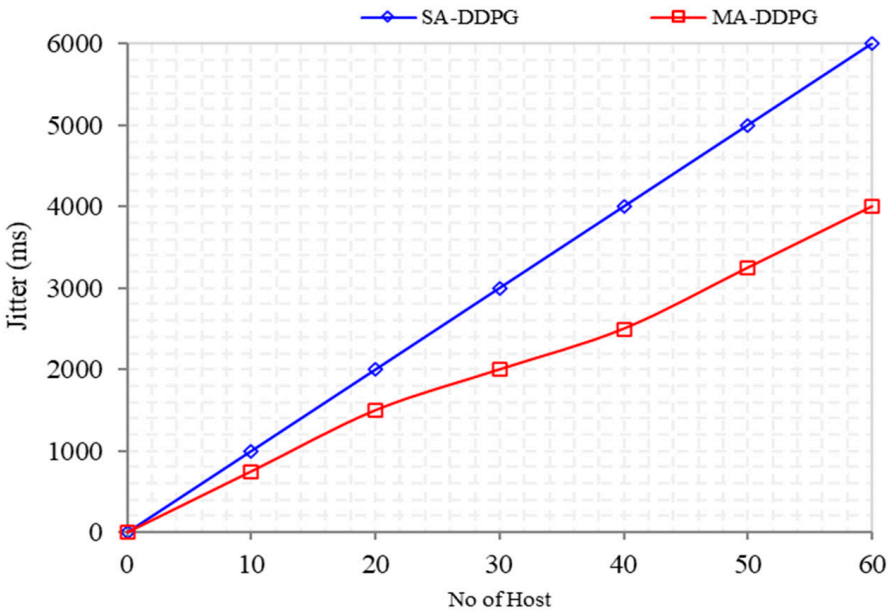
Figure 10. Reward vs. no. of episodes for SA-DDPG and MA-DDPG.

7.2.8. Jitter

As shown in Figure 11, as the number of nodes involved in packet transmittal increases, the average jitter increases due to link utilization and congestion. MADDPG for 20-nodes topology decreased jitter by 33% to 2000 ms compared with DDPG of 4000 ms. For the 60-nodes topology, the jitter performance for MADDPG decreased by 20% to 4000 ms compared with 6000 ms for DDPG. The initial jitter between MADDPG and DDPG until 5 nodes was close to a percentage of 2% for both 20 and 60 topologies.



(**a**) 20 nodes

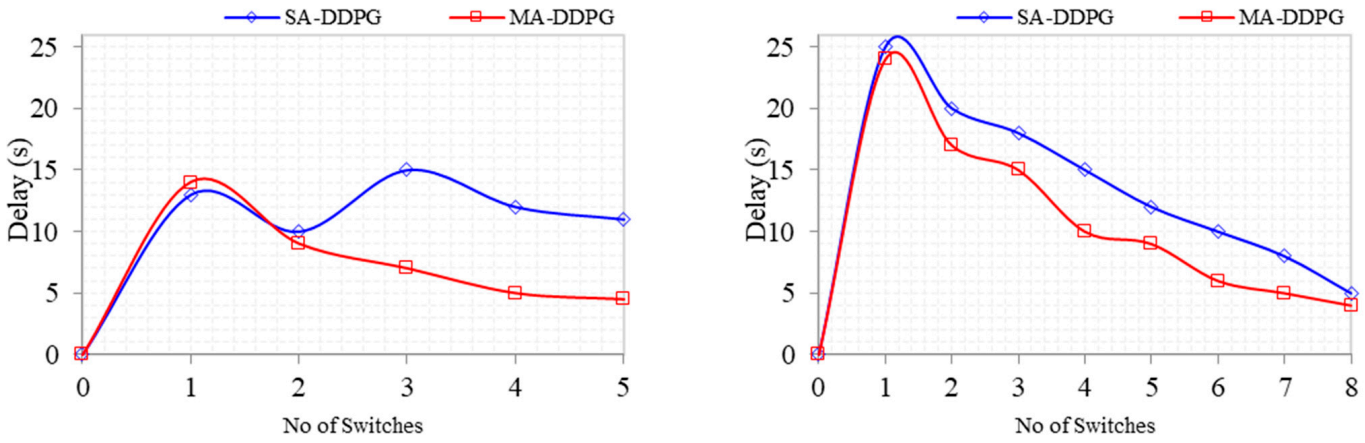


(**b**) 60 nodes

Figure 11. Jitter vs. no. of hosts for SA-DDPG and MA-DDPG.

7.2.9. Delay vs. No. of Switches

Delay is the time it takes for packets to move from source to destination nodes in the SDN network. Packets sent from source to destination nodes have a transmission delay monitored by the SDN switches using the delay in the current arrival packets from the nodes and the RYU controller. As shown in Figure 12, as the episode progresses and the number of switches involved in packet transmission increases, the overall delay in the network decreases for both MADDPG and DDPG due to multipath routing. From the graph, MADDPG with 5 switches and 20 nodes has a 24% better delay performance from switch 1 to switch 5 compared with DDPG. This spans from a delay of 14 s from switch 1 to 4.5 s at switch 5. As the number of nodes increased to 60 with 8 switches, MADDPG's delay performance decreased to 11% compared with DDPG. The delay performance was initially close until switch 2, giving no percentage change for the 5-switches topology and a 2% change for the 8-switches topology.

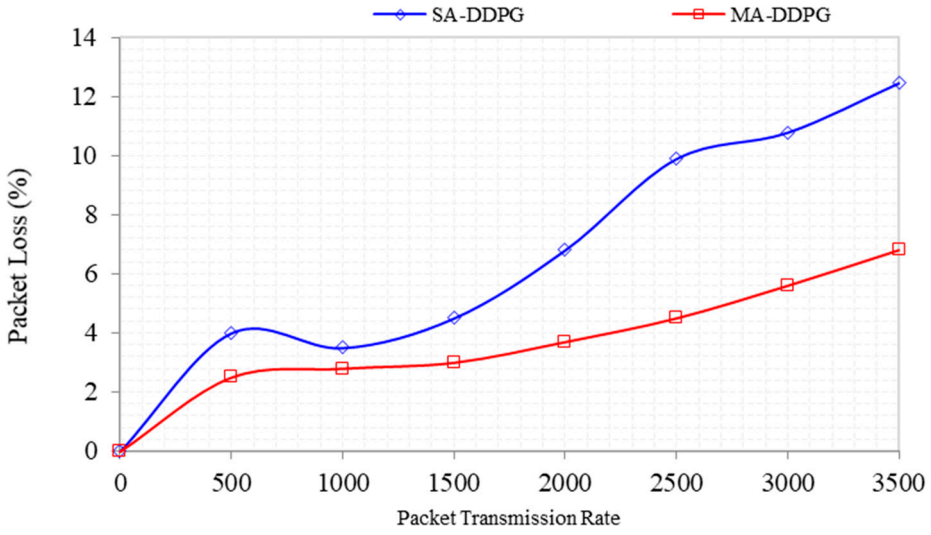


(**a**) 20 nodes (**b**) 60 nodes

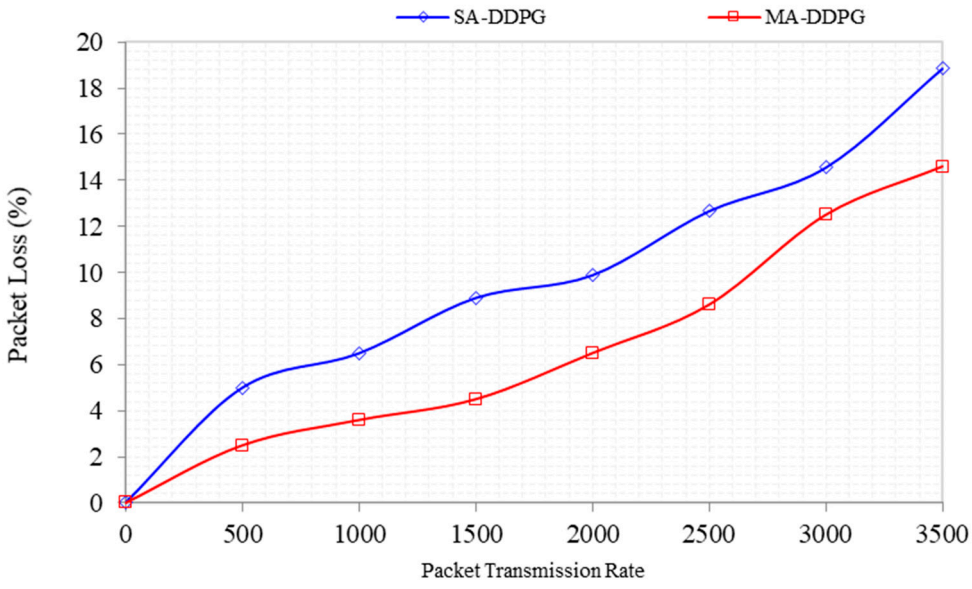
Figure 12. Delay vs. no. of switches for SA-DDPG and MA-DDPG.

7.2.10. Packet Loss

Packet loss is computed to ascertain the performance of the agents as the episode progresses. As the packet transmission rate is increased per second for flash events and DDoS, the packet loss is computed in percentage, relative to the total traffic in the network. From Figure 13, MADDPG agent for 20 nodes had a packet loss of 6.8% when 3500 packets were transmitted per second with an initial loss of 2.5% at a transmission rate of 500 packets per second. DDPG agents for 20 nodes lost an initial packet of 4% at 500 packet rate, which increased to 12.4% at 3500 packets per second. MADDPG reduced packet loss to almost half at each stage of the packet increment at the nodes. For the 60 nodes, since more nodes in proportion to the number of switches increased the packet transmission rate, MADDP lost 14.6% packets at 3500 packets per second with an initial loss of 2.5% that increased rapidly. DDPG lost 18.86% of its packets at 3500 packets per second against an initial loss of 5% at 500 packets per second. MADDPG for 60 nodes reduced the packet loss to a quarter when compared with DDPG.



(**a**) 20 nodes

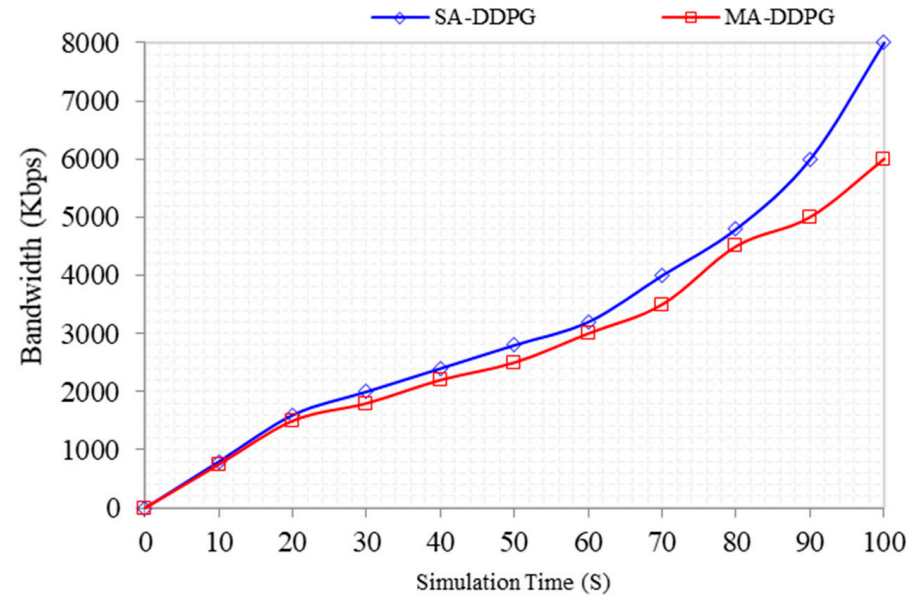


(**b**) 60 nodes

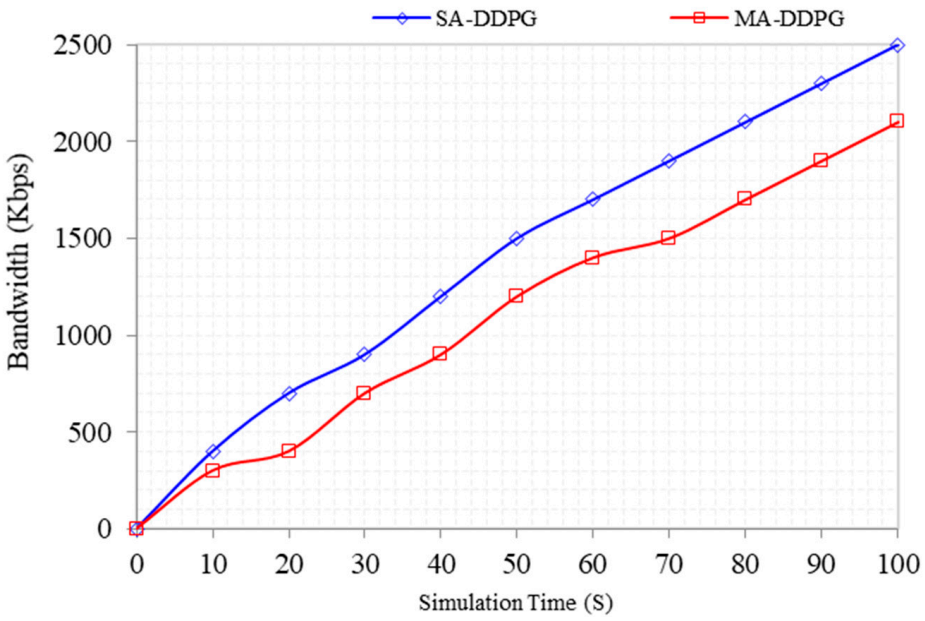
Figure 13. Packet loss vs. packet transmission rate for SA-DDPG and MA-DDPG.

7.2.11. Bandwidth Usage

The bandwidth was captured within a 100 s simulation time frame. As shown in Figure 14, MADDPG uses an 11%, slightly lower, bandwidth for the 20-nodes architecture than DDPG. At 100 s, MADDPG utilizes 2100 Kbps bandwidth while DDPG utilizes 2500 Kpbs bandwidth. For 60-nodes topology and the same time window, MADDPG has a 7% better bandwidth utilization compared with DDPG. The jump in bandwidth utilization for 60-nodes DDPG increased rapidly by 4.5% from the 90 s time window compared with the 3.5% before that time. The initial bandwidth utilization for the 60-nodes MADDPG at 10 s was 750 Kbps compared with 800 Kbps within the same time.



(**a**) 20 nodes



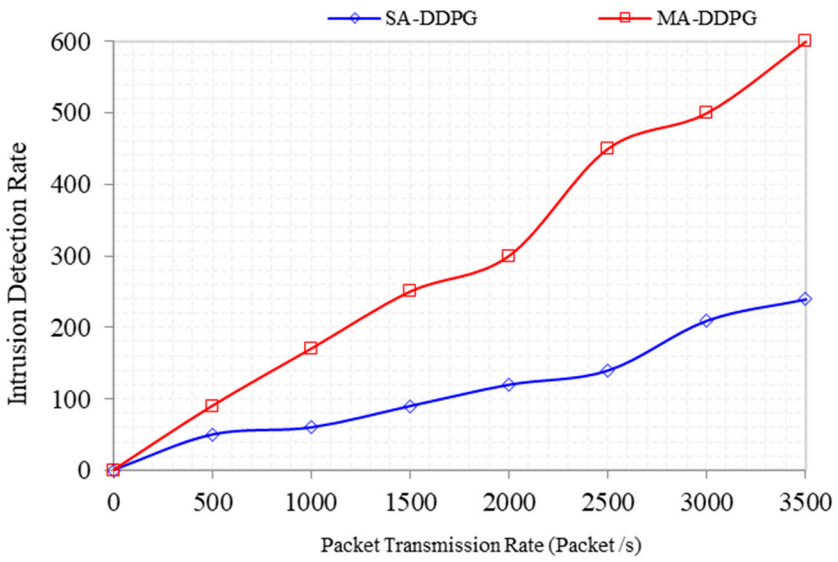
(**b**) 60 nodes

Figure 14. Bandwidth vs. simulation time for SA-DDPG and MA-DDPG.

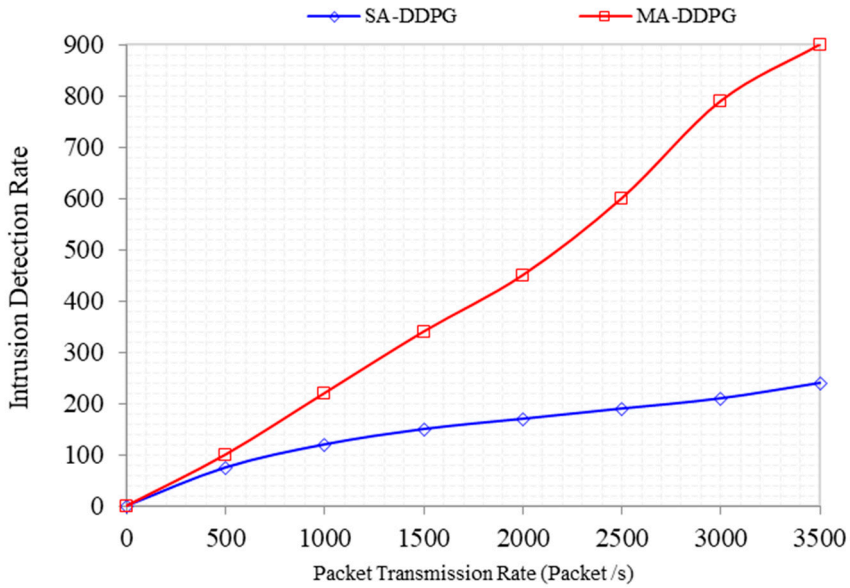
7.2.12. Intrusion Detection Rate

Intrusion Detection Rate captures the False Positive Rate in the network. It represents the number of compromised packets caused by DDoS as the transmission rate increases. As shown in Figure 15, when compared with DDPG, MADDPG has a 44% improvement in DDoS detection and detected 600 packets as DDoS packets at a transmission rate of 3500 packets per second against an initial detection of 90 packets at a packet transmission rate of 500 packets per second. DDPG detected an initial DDoS packet of 50 at a transmission rate of 500 packets per second against 240 packets when the transmission rate increased to 3500 packets per second. For the 60 nodes, MADDPG has a 50% improvement in DDoS

detection over DDPG. The MADDPG detected 900 DDoS packets at a transmission of 3500 packets per second with an initial detection value of 100 at a rate of 500 packets per second. DDPG detected 75 DDoS packets at an initial packet transmission rate of 500 packets per second against a high value of 240 DDoS packets against a transmission rate of 3500 packets per second.



(**a**) 20 nodes



(**b**) 60 nodes

Figure 15. Intrusion Detection Rates vs. Packet Transmission Rate for SA-DDPG and MA-DDPG.

8. Conclusions and Future Work

In this research, we proposed an MADDPG framework using two agents in an SDN-enabled IoT environment for multipath routing and DDoS detection and prevention. We utilized MDP to mathematically model the state-action-reward of the agents and provided an integrated MADDPG algorithm using our model. The proposed architecture as a novel framework that seeks to alleviate malicious traffic from an SD-network while optimizing

flow paths and transitions by updating the flow table of forwarding devices. Extensive simulation using network metrics demonstrates the significant intervention of the framework for traffic engineering and intrusion detection in SDN.

A possible future research is to extend the threat determination to cover other types of network intrusions. Deploying MADDPG algorithm in distributed controllers is another interesting research that will positively affect the efficient handling of network metrics in an SDN-IoT heterogenous environment. Combining MADDPG with other meta-heuristics algorithms in SDN-IoT is another promising research area.

Author Contributions: Conceptualization, D.K.D.; formal Analysis, D.K.D., H.N.-M. and J.D.G.; writing—review and editing, D.K.D. and G.S.K.; resources, D.K.D. and G.S.K.; concept design, D.K.D. and J.D.G.; original draft preparation, D.K.D. and J.D.G.; supervision, J.D.G., G.S.K. and H.N.-M.; project administration, J.D.G., H.N.-M. and G.S.K.; software, D.K.D. and H.N.-M.; validation, J.D.G., G.S.K. and H.N.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research did not receive any external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not Applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Santos, A.F.C.; Teles, Í.P.; Siqueira, O.M.P.; de Oliveira, A.A. Big data: A systematic review. *Adv. Intell. Syst. Comput.* **2018**, *558*, 501–506.
2. Ocean, B. Optical Studies on Sol-Gel Derived Lead Chloride Crystals. *J. Appl. Eng. Comput. Sci.* **2013**, *2*, 5.
3. Mehmood, Y.; Haider, N.; Imran, M.; Timm-Giel, A.; Guizani, M. M2M Communications in 5G: State-of-the-Art Architecture, Recent Advances, and Research Challenges. *IEEE Commun. Mag.* **2017**, *55*, 194–201. [CrossRef]
4. Mattisson, S. Overview of 5G requirements and future wireless networks. In Proceedings of the ESSCIRC 2017-43rd IEEE European Solid State Circuits Conference, Leuven, Belgium, 11–14 September 2017; pp. 1–6. [CrossRef]
5. Xiao, L.; Wan, X.; Lu, X.; Zhang, Y.; Wu, D. IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Process. Mag.* **2018**, *35*, 41–49. [CrossRef]
6. Ni, J.; Zhang, K.; Vasilakos, A.V. Security and Privacy for Mobile Edge Caching: Challenges and Solutions. *IEEE Wirel. Commun.* **2020**, 1–7. [CrossRef]
7. Vishwakarma, R.; Jain, A.K. A survey of DDoS attacking techniques and defence mechanisms in the IoT network. *Telecommun. Syst.* **2020**, *73*, 3–25. [CrossRef]
8. Xia, W.; Wen, Y.; Foh, C.H.; Niyato, D.; Xie, H. A Survey on Software-Defined Networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 27–51. [CrossRef]
9. Wickboldt, J.; De Jesus, W.; Isolani, P.; Both, C.; Rochol, J.; Granville, L. Software-defined networking: Management requirements and challenges. *IEEE Commun. Mag.* **2015**, *53*, 278–285. [CrossRef]
10. Hamdan, M.; Hassan, E.; Abdelaziz, A.; Elhigazi, A.; Mohammed, B.; Khan, S.; Vasilakos, A.V.; Marsono, M.N. A comprehensive survey of load balancing techniques in software-defined network. *J. Netw. Comput. Appl.* **2021**, *174*. [CrossRef]
11. Ray, S. A Quick Review of Machine Learning Algorithms. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; pp. 35–39. [CrossRef]
12. Almseidin, M.; Alzubi, M.; Kovacs, S.; Alkasassbeh, M. Evaluation of machine learning algorithms for intrusion detection system. In Proceedings of the 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 14–16 September 2017; pp. 277–282. [CrossRef]
13. Kuzhippallil, M.A.; Joseph, C.; Kannan, A. Comparative Analysis of Machine Learning Techniques for Indian Liver Disease Patients. In Proceedings of the 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Subotica, Serbia, 14–16 September 2017; pp. 778–782. [CrossRef]
14. Merkert, J.; Mueller, M.; Hubl, M. A survey of the application of machine learning in decision support systems. In Proceedings of the European Conference on Information Systems 2015, Münster, Germany, 26–29 May 2015; pp. 1–15.
15. Mishra, N.K.; Celebi, M.E. An Overview of Melanoma Detection in Dermoscopy Images Using Image Processing and Machine Learning. *arXiv* **2016**, arXiv:1601.07843.
16. Amruthnath, N.; Gupta, T. A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In Proceedings of the 2018 5th International Conference on Industrial Engineering and Applications (ICIEA), Singapore, 26–28 April 2018; pp. 355–361. [CrossRef]

17. Recht, B. A Tour of Reinforcement Learning: The View from Continuous Control. *Annu. Rev. Control Robot. Auton. Syst.* **2019**, *2*, 253–279. [CrossRef]
18. Asiain, E.; Clempner, J.B.; Poznyak, A.S. Controller exploitation-exploration reinforcement learning architecture for computing near-optimal policies. *Soft Comput.* **2019**, *23*, 3591–3604. [CrossRef]
19. Bhunia, S.S.; Gurusamy, M. Dynamic attack detection and mitigation in IoT using SDN. In Proceedings of the 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, VIC, Australia, 22–24 November 2017; pp. 1–6.
20. Zhang, J.; Ye, M.; Guo, Z.; Yen, C.Y.; Chao, H.J. CFR-RL: Traffic Engineering with Reinforcement Learning in SDN. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 2249–2259. [CrossRef]
21. Rischke, J.; Sossalla, P.; Salah, H.; Fitzek, F.H.P.; Reisslein, M. QR-SDN: Towards reinforcement learning states, actions, and rewards for direct flow routing in software-defined networks. *IEEE Access* **2020**, *8*, 174773–174791. [CrossRef]
22. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. [CrossRef]
23. Liu, X.; Xie, L.; Wang, Y.; Zou, J.; Xiong, J.; Ying, Z.; Vasilakos, A.V. Privacy and Security Issues in Deep Learning: A Survey. *IEEE Access* **2020**, *9*. [CrossRef]
24. Xu, G.; Li, H.; Ren, H.; Yang, K.; Deng, R.H. Data Security Issues in Deep Learning: Attacks, Countermeasures, and Opportunities. *IEEE Commun. Mag.* **2019**, *57*, 116–122. [CrossRef]
25. Fan, J.; Wang, Z.; Xie, Y.; Yang, Z. A Theoretical Analysis of Deep Q-Learning. *arXiv* **2019**, arXiv:1901.00137.
26. Tafazzol, S.; Fathi, E.; Rezaei, M.; Asali, E. Curious Exploration and Return-based Memory Restoration for Deep Reinforcement Learning. *arXiv* **2021**, arXiv:2105.00499.
27. Hou, Y.; Liu, L.; Wei, Q.; Xu, X.; Chen, C. A novel DDPG method with prioritized experience replay. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 316–321. [CrossRef]
28. Isyaku, B.; Mohd Zahid, M.S.; Bte Kamat, M.; Abu Bakar, K.; Ghaleb, F.A. Software Defined Networking Flow Table Management of OpenFlow Switches Performance and Security Challenges: A Survey. *Future Internet* **2020**, *12*, 147. [CrossRef]
29. Naderializadeh, N.; Sydir, J.; Simsek, M.; Nikopour, H. Resource Management in Wireless Networks via Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 3507–3523. [CrossRef]
30. Bedawy, A.; Yorino, N.; Mahmoud, K.; Zoka, Y.; Sasaki, Y. Optimal Voltage Control Strategy for Voltage Regulators in Active Unbalanced Distribution Systems Using Multi-Agents. *IEEE Trans. Power Syst.* **2020**, *35*, 1023–1035. [CrossRef]
31. Dharmadhikari, C.; Kulkarni, S.; Temkar, S.; Bendale, S.; Student, B.E. A Study of DDoS Attacks in Software Defined Networks. *Int. Res. J. Eng. Technol.* **2019**, 448–453. Available online: www.irjet.net (accessed on 20 May 2020).
32. Akbari, I.; Tahoun, E.; Salahuddin, M.A.; Limam, N.; Boutaba, R. ATMoS: Autonomous Threat Mitigation in SDN using Reinforcement Learning. In Proceedings of the NOMS 2020–2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 20–24 April 2020. [CrossRef]
33. Phan, T.V.; Gias, T.M.R.; Islam, S.T.; Huong, T.T.; Thanh, N.H.; Bauschert, T. Q-MIND: Defeating stealthy dos attacks in SDN with a machine-learning based defense framework. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019. [CrossRef]
34. Liu, Y.; Dong, M.; Ota, K.; Li, J.; Wu, J. Deep Reinforcement Learning based Smart Mitigation of DDoS Flooding in Software-Defined Networks. In Proceedings of the 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona, Spain, 17–19 September 2018. [CrossRef]
35. Guo, X.; Lin, H.; Li, Z.; Peng, M. Deep-Reinforcement-Learning-Based QoS-Aware Secure Routing for SDN-IoT. *IEEE Internet Things J.* **2020**, *7*, 6242–6251. [CrossRef]
36. Phan, T.V.; Islam, S.T.; Nguyen, T.G.; Bauschert, T. Q-DATA: Enhanced Traffic Flow Monitoring in Software-Defined Networks applying Q-learning. In Proceedings of the 2019 15th International Conference on Network and Service Management (CNSM), Halifax, NS, Canada, 21–25 October 2019. [CrossRef]
37. Yao, Z.; Wang, Y.; Qiu, X. DQN-based energy-efficient routing algorithm in software-defined data centers. *Int. J. Distrib. Sens. Netw.* **2020**, *16*. [CrossRef]
38. Stampa, G.; Arias, M.; Sanchez-Charles, D.; Muntes-Mulero, V.; Cabellos, A. A Deep-Reinforcement Learning Approach for Software-Defined Networking Routing Optimization. *arXiv* **2017**, arXiv:1709.07080.
39. Yuan, T.; da Rocha Neto, W.; Rothenberg, C.E.; Obraczka, K.; Barakat, C.; Turletti, T. Dynamic Controller Assignment in Software Defined Internet of Vehicles through Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 585–596. [CrossRef]
40. Wu, T.; Zhou, P.; Wang, B.; Li, A.; Tang, X.; Xu, Z.; Chen, K.; Ding, X. Joint Traffic Control and Multi-Channel Reassignment for Core Backbone Network in SDN-IoT: A Multi-Agent Deep Reinforcement Learning Approach. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 231–245. [CrossRef]
41. Yu, C.; Lan, J.; Guo, Z.; Hu, Y. DROM: Optimizing the Routing in Software-Defined Networks with Deep Reinforcement Learning. *IEEE Access* **2018**, *6*, 64533–64539. [CrossRef]
42. Gordon, H.; Batula, C.; Tushir, B.; Dezfouli, B.; Liu, Y. Securing Smart Homes via Software-Defined Networking and Low-Cost Traffic Classification. *arXiv* **2021**, arXiv:2104.00296.

43. Van Otterlo, M.; Wiering, M. Reinforcement learning and markov decision processes. *Adapt. Learn. Optim.* **2012**, *12*, 3–42. [CrossRef]
44. Imani, M.; Ghoreishi, S.F.; Braga-Neto, U.M. Bayesian control of large MDPs with unknown dynamics in data-poor environments. *Adv. Neural Inf. Process. Syst.* **2018**, *2018-Decem*, 8146–8156.
45. Asadollahi, S.; Sameer, M. Ryu Controller's Scalability Experiment on Software Defined Networks. In Proceedings of the 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), Bangalore, India, 1–2 February 2018; pp. 3–7.
46. Li, S. Multi-Agent Deep Deterministic Policy Gradient for Traffic Signal Control on Urban Road Network. In Proceedings of the 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), Dalian, China, 25–27 August 2020; pp. 896–900. [CrossRef]