IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 4, 2019, accepted July 18, 2019, date of publication July 24, 2019, date of current version August 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930803

A New Smart Router-Throttling Method to Mitigate DDoS Attacks

SHI-MING XIA[1], SHI-ZE GUO[2], WEI BAI[1], JUN-YANG QIU[3], HAO WEI[1], AND ZHI-SONG PAN[1]
[1]Laboratory of Intelligent Information Processing, Army Engineering University of PLA, Nanjing 210007, China
[2]The Institute of North Electronic Equipment, Beijing 100083, China
[3]School of Information Technology, Deakin University, Geelong, VIC 3216, Australia

Corresponding author: Zhi-Song Pan (hotpzs@hotmail.com)

This work was supported by the National Key Research Development Program of China under Grant 2017YFB0802800.

ABSTRACT The distributed denial of service (DDoS) attack is one of the most server threats to the current Internet and brings huge losses to society. Furthermore, it is challenging to defend DDoS due to the case that the DDoS traffic can appear similar to the legitimate ones. Router throttling is an accessible approach to defend DDoS attacks. Some existing router throttling methods dynamically adjust a given threshold value to keep the server load safe. However, these methods are not ideal as they exploit the information of the current time, so the perception of time series variations is poor. The DDoS problem can be seen as a Markov decision process (MDP). Multi-agent router throttling (MART) method based on hierarchical communication mechanism has been proposed to address this problem. However, each agent is independent with each other and has no communication among them, therefore, it is hard for them to collaborate to learn an ideal policy to defend DDoS. To solve this multi-agent partially observable MDP problem, we propose a centralized reinforcement learning router throttling method based on a centralized communication mechanism. Each router sends its own traffic reading to a central router, the central router then makes a decision for each router to choose the throttling rate. We also simulate the environment of the DDoS problem more realistic while modify the reward function of the MART to make the reward function of more coherent. To decrease the communication costs, we add a deep deterministic policy gradient network for each router to decide whether or not to send information to the central agent. The experiments validate that our proposed new smart router throttling method outperforms existing methods to the DDoS instruction response.

INDEX TERMS Distributed denial of service, router throttling, Markov decision process, multi-agent router throttling, hierarchical communication, centralized communication, communication costs.

I. INTRODUCTION

Denial of Service attacks constitute one of the major cyber threats and among the most complicated security problems in today's Internet [1]–[3]. Of particular concerns are Distributed Denial of Service (DDoS) attacks, whose impact can be proportionally severe. With little or no advance warning, a DDoS attack can easily exhaust the computing and communication resources of its victim server within a short period of time [4], [5]. What is worse, with the increase of Internet bandwidth and the continuous release of various DDoS hacking tools, the implementation of DDoS attacks is becoming easier [6], and the events of DDoS attacks are on the rise [2], [7]. Due to a variety of factors such as commercial competition, retaliation, and network extortion, many commercial sites, game servers, chat networks, and other network service providers have long been plagued by DDoS attacks.

The DDoS threat is challenging to defend for many reasons [4], [8]. First, the traffic is generated from terminals spreading all over the Internet, and then all traffic is aggregated at the victim server. Second, the large volume of the aggregated traffic is unlikely to be stopped by a single defense point near the victim. Third, the DDoS traffic can appear to be similar to the legitimate one since the damage may cause by the total volume of traffic and not the traffic content.

If we could know the legitimate traffic percentage, we can easily solve the problem by Linear Programming (LP). As we can know the total traffic reading of each throttling router and suppose that we could not distinguish the legitimate traffic from attack traffic. Thus, how to reduce the total traffic to

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

the victim server to allow the server work properly while ensuring traffic of legitimate user to access the server as much as possible is an urgent problem to be solved.

Router throttling [9] is a popular approach to defend against DDoS attacks, where the victim server signals a set of upstream routers to throttle traffic. The key element in the defense system is to allocate appropriate throttling rates at the distributed routing points. The appropriate throttles should depend on the current demand distributions, so dynamically negotiation is indispensable between the server and network routers. There are two baseline methods, Server-initiated Router Throttle (SIRT) algorithm and Fair Router Throttle (FRT) algorithm [9]. Both of SIRT and FRT adjust a given threshold value to keep the server load within $[L_s, U_s]$, L_s is the low-water mark of the server load and U_s is the load limits of the victim server. However, as these two methods only based on the traffic reading itself, and could not reuse the experience of past time. Thus, it is hard to make an ideal decision to set the throttling rate.

The DDoS problem can be formalized as a Markov Decision Process (MDP) [10], [11]. Thus, the agents can do actions based on current traffic information as well as its previous experience with the network environment. Malialis and Kudenko [12], [13] introduce Multi-agent Reinforcement Learning (MARL) method based on hierarchical communication mechanism, where multiple independent reinforcement learning agents are installed on a set of upstream routers and learn to throttle traffic towards the victim server. There is no communication between different router agents and each agent can only get information from its parent nodes in the network. As each agent is independent with each other, the only link is the global reward of all agents. Each agent treats the other agents as part of its environment and makes a decision of throttling rate. However, the strategies of other agents are uncertain and changing as training progresses [14], so the environment becomes unstable from the perspective of any individual agent and thus it is hard for agents to collaborate to learn an ideal policy.

Motivated by existing research about distributed router throttling methods, we propose a Centralized Reinforcement Learning Router Throttling with Less Communication (CRLRT-LC) method to learn a better policy while decreasing the communication costs.

The organization of the paper is as follows. Section II introduces the DDoS attack problem and some existing router throttling methods. Details of our proposed CRLRT-LC method and its design details are introduced in Section III. Data preparation, experimental settings, performance evaluation, and the follow-up discussion are presented in Section IV. Conclusions are given in Section V.

II. DISCUSSION DDOS PROBLEM AND ROUTER THROTTLING METHODS

In this section, the DDoS attack problem and some existing router throttling methods (SIRT, FRT, and MART) will be introduced.

A. DDOS PROBLEM DEFINITION

The strategy behind the DDoS attack is described by the DDoS attack model [15]. The terminals are compromised by the attacker, which constitute the botnet. Specifically, the attacker installs malicious software on vulnerable terminals to compromise them, thus being able to communicate with and control them. The attacker communicates with the handlers, which in turn control the terminals in order to launch a DDoS attack.

Router throttling is a framework for DDoS instruction response as shown in Fig.1. The **Environment** is the network topology, a server responds the request from users (or terminals). When the attack is happening, some users may be controlled by the attacker. This may lead the traffic accumulated in throttling router contains much attack traffic, which may lead the server crashed. The **Agents** are the throttling agents, they get **State** information from the environment, then respond **Action** to the environment and get feedback **Reward** from the environment. The purpose of router throttling is to keep the server work and let as much legitimate user request the server. So before the accumulated large amount of traffic reaching to the server, the throttling router should drop some traffic.

The DDoS attacks can be viewed as a resource management problem. Assume x_t^i is the legitimate traffic reading of router i at moment t, y_t^i is the attack traffic, a_t^i is the throttling rate, the goal is to maximize the passing rate of legitimate traffic of all nodes during a time T as shown in Equation (1), N is the number of router agents. And subject to the conditions that the passed total traffic is less than the load limits U_s as shown in Equation (2).

$$max \sum_{t=0}^{T} \frac{\sum_{i=0}^{N} x_t^i(1-a_t^i)}{\sum_{i=0}^{N} x_t^i} \tag{1}$$

$$s.t. \sum_{i=0}^{N}(x_t^i + y_t^i)(1-a_t^i) \leq U_s \tag{2}$$

As in reality, we could not know the exact value of the legitimate traffic x_t^i. And we can only get the sum value of $x_t^i + y_t^i$. The DDoS problem cannot be solved by Linear Programming (LP) method [16], [17] to get the optimal router throttling rate a_t^i for each router. Thus, if we want to make decisions based on incomplete, vague information, we need to get some feedback information from the network environment to tell us whether the decision we just made is efficiency.

B. SERVER-INITIATED ROUTER THROTTLE ALGORITHM

For Server-initiated Router Throttle (SIRT) algorithm, all routers share the same throttling rate $a_t \in [0, 1]$. Each router throttles a fraction a_t of traffic, to make sure the traffic to victim server is within $[L_s, U_s]$ during the DDoS attack. The fraction a_t is adjusted according to current server congestion. The fraction a_t is zero when no throttle is in effect. If Z_t (the total traffic transfered to victim server at moment t, as shown in Equation (3)) is more than U_s, this method will

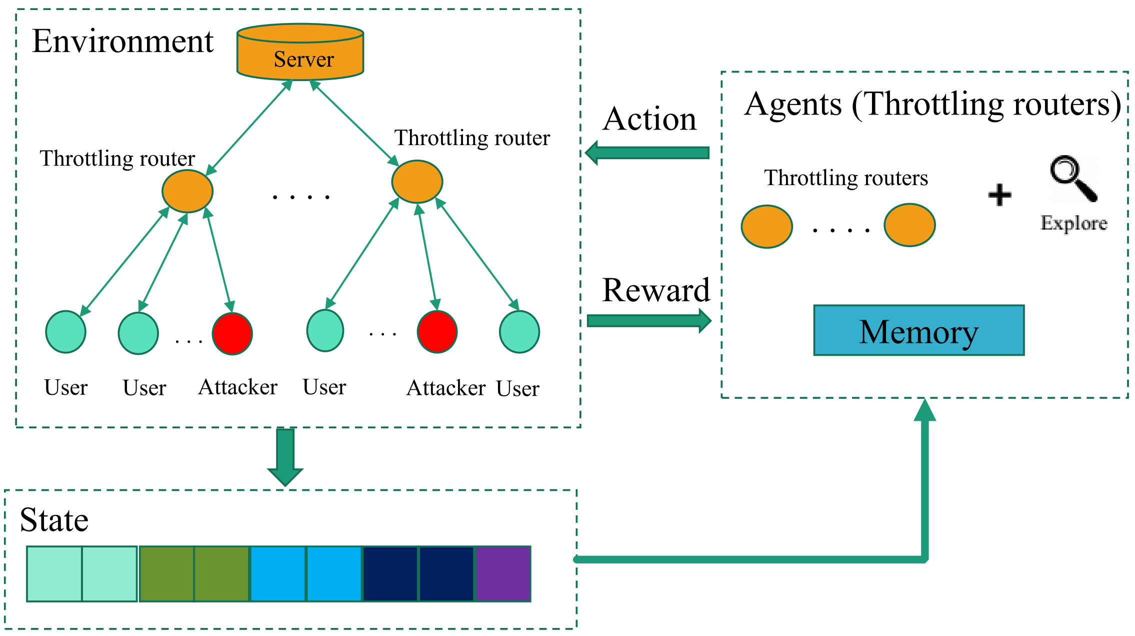


FIGURE 1. **The framework of router throttling method. The throttling router (agent) will get some information (state) from the environment and decide the throttling rate of the router (action) by its own policy. Then get some feedback (reward) from the environment. The interaction data (state, action, reward, new state) will be stored in a memory buffer. The data can help the agent improve its policy.**

increase the throttling rate a_t. If Z_t is less than L_s, it will decrease the throttling rate a_t, as shown in Equation (4). Here, $\alpha, \beta \in [0, 1]$.

$$Z_t = \sum_{i=0}^{N}(x_t^i + y_t^i)(1 - a_t^i) \tag{3}$$

$$\begin{cases} a_{t+1} = 1 - (1 - a_t)(1 - \alpha) & Z_t > U_s \\ a_{t+1} = 1 - (1 - a_t)(1 + \beta) & Z_t < L_s \end{cases} \tag{4}$$

As all routers share the same throttling rate a_t, this SIRT algorithm is very simple to conduct. But it is not fair because it penalizes all routers equally, irrespective of whether they are greedy or well behaving.

C. FAIR ROUTER THROTTLE ALGORITHM

For the Fair Router Throttle (FRT) algorithm, all routers do not share the same throttling rate. Each router throttles a fraction a_t^i of traffic ($a_t^i \in [0, 1]$), to make sure the traffic to victim server is within $[L_s, U_s]$. Here, given a threshold value R_s^t for each router, then the throttling rate a_t^i for each router is updated as Equation (5), the threshold value $R_s(t)$ is updated as Equation (6). Here, $\alpha, \beta \in [0, 1]$.

$$\begin{cases} a_{t+1}^i = \dfrac{x_t^i + y_t^i - R_s(t)}{x_t^i + y_t^i} & x_t^i + y_t^i > R_s(t) \\ a_{t+1}^i = 0 & x_t^i + y_t^i \leq R_s(t) \end{cases} \tag{5}$$

$$\begin{cases} R_s(t + 1) = R_s(t)(1 - \alpha) & Z_t > U_s \\ R_s(t + 1) = R_s(t)(1 + \beta) & Z_t < L_s \end{cases} \tag{6}$$

For attack traffic is much more than legitimate traffic, the FRT algorithm is better than the SIRT algorithm. But if the attack traffic is as much as legitimate traffic. This algorithm may get a bad result. In reality, some legitimate traffic is much more than others, for such problem, this FRT method may fail.

D. MULTI-AGENT ROUTER THROTTLING METHOD

Malialis and Kudenko [12], [13] introduce Multi-agent Router Throttling (MART) method based on hierarchical communication mechanism, where multiple independent reinforcement learning agents are installed on a set of upstream routers and learn to throttle traffic towards the victim server.

1) REINFORCEMENT LEARNING

Consider a standard Markov Decision Process (MDP) [18], [19], which is represented by a tuple:

$$M = \langle S, A, P, R, \gamma \rangle \tag{7}$$

S is the state space and A is the action space. The dynamic transition is specified by $P(\cdot) = \Pr(s_{t+1}|s_t, a_t)$ and $R(\cdot) = r(s_t, a_t)$ assigns reward for state-action pairs. By interacting with the environment, agents generate trajectories: $(s_1, a_1, r_1, s_2, \cdots, s_T)$. The goal of reinforcement learning is to maximize cumulative reward, as Equation (8) shows, where discounted factor γ is designed to make sure cumulative reward is bound.

$$\max \sum_{t=1}^{T} \gamma^{t-1} r_t \tag{8}$$

Similarily, Multi-agent Reinforcement Learning (MARL) is specified by a tuple:

$$\langle N, \{O_i\}_{i=1}^N, S, \{A_i\}_{i=1}^N, P, R, \gamma \rangle \tag{9}$$

Here, N is the number of agents. For specific agent i, $o_i^t \in O_i$ is its observation at the time t, $a_i \in A_i$ is its action at the time t. Different from the standard MDP, single agent cannot get access to global information $s_t \in S$. What's more, dynamic process and joint reward is determined by all agent's behaviors, which means that $P(\cdot) = \Pr(s_{t+1}|s_t, \{a_t^i\}_{i=1}^N)$ and $R(\cdot) = r(s_t, \{a_t^i\}_{i=1}^N)$. Although the team goal of MARL remains to maximize cumulative joint reward, but the target of one agent moves when other agents' polices change, which makes the optimization of MARL intractable.

2) HIERARCHICAL COMMUNICATION MECHANISM

Assume the communication condition is ideal. The victim's router signals its local traffic reading to the team leaders. The team leaders signal both their local traffic reading and the received reading from the server to their intermediate routers. Similarly, the intermediate routers signal their local traffic reading and the two received readings to their throttling routers. This is depicted in Fig.2(a) and Fig.2(b) by the un-directional arrows.

State Space (S): the state of each throttling router agent is the traffic reading from the server to router agent along the un-directional arrows in Fig.2. For example, router agent R_3 in Fig.2(b)) has four-dimensional, they are the traffic reading of server R_s, team leader R_1, intermedia router R_2 and the traffic reading of itself R_3. Thus, for throttling router agent R_3, its state is $[R_s, R_1, R_2, R_3]$.

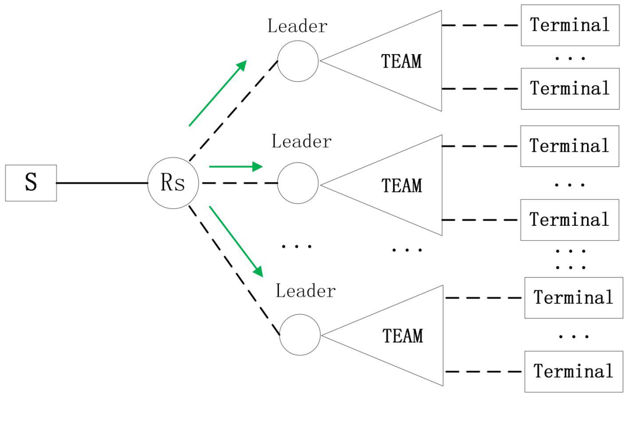
Action Space (A): each router applies to throttle via probabilistic traffic dropping. For example, action 0.2 means that the router will drop 20% of its aggregate traffic towards the victim server. The available action space is [0, 1].

Reward Function (R): for the basic Multi-agent Router Throttling (MART) [9], [16] method, each agent receives the same reward or punishment. The system has two important goals, which are directly encoded in the reward function as shown in Equation (10). The first goal is to keep the victim server operational, that is, to keep its load below the upper boundary U_s. When this is not the case, the system receives a punishment of -1. The second goal of the system is to allow as much legitimate traffic as possible to reach the victim server during the DDoS attack.

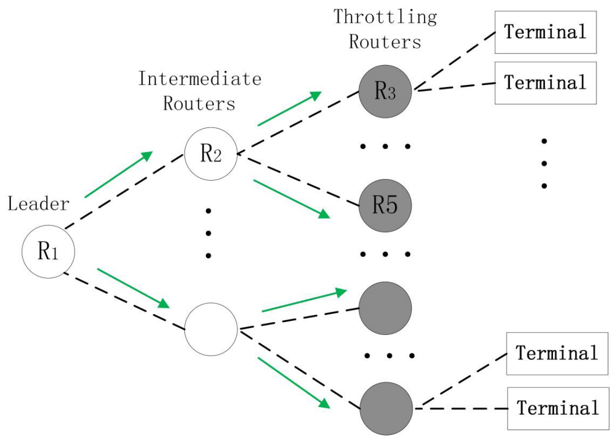
$$\begin{cases} R = \dfrac{\sum_{i=0}^N x_t^i(1-a_t^i)}{\sum_{i=0}^N x_t^i} & Z_t \le U_s \\ R = -1 & Z_t > U_s \end{cases} \tag{10}$$

3) DEEP DETERMINISTIC POLICY GRADIENT

The Deep Deterministic Policy Gradient (DDPG) is a model-free [20]–[22] off-policy actor-critic algorithm using deep function approximators that can learn policies in high-dimensional, continuous action spaces. It combines the actor-critic approach with insights from the recent success of Deep Q Network (DQN) [23]–[25].



(a) Team formation



(b) Team structure

FIGURE 2. Team structure based Hierarchical communication structure.

The Reinforcement Learning algorithms can be divided into three groups [26]. (i) Actor-only methods typically work with a parameterized family of policies over which optimization procedures can be used directly, such as policy gradient methods [27]. (ii) Critic-only methods that use temporal difference learning have lower variance in the estimates of expected returns, such as Q-learning. (iii) Actor-critic methods aim to combine the advantages of actor-only and critic-only methods. Actor-critic [28], [29] algorithm was generally believed that learning value functions using large, non-linear function approximators was difficult and unstable.

DQN algorithm is capable of human-level performance on many Atari video games using unprocessed pixels for input [24]. However, it can only handle discrete and low-dimensional action spaces. And cannot be straightforwardly applied to continuous domains since it relies on finding the action that maximizes the action-value function, which in the continuous-valued case requires an iterative optimization process at every step.

DDPG combines the previously successful DQN structure to improve the stability and convergence of actor-critic. DDPG can learn competitive policies using low-dimensional observations using the same hyper-parameters and network structure. A key feature of the approach is its simplicity: it requires only a straightforward actor-critic architecture and learning algorithm with very few "moving parts", making it easy to implement and scale to more difficult problems and larger networks. Therefore, in this paper, we adopt DDPG as a base reinforcement learning method.

4) MULTI-AGENT DEEP DETERMINISTIC POLICY GRADIENT METHOD

For the DDoS problem, each throttling agent gets its own partially observation and chooses an action from action space. Then each agent gets a reward from the environment to improve the Reinforcement Learning (RL) policy. This is a cooperative task, all agents cooperate to make the largest reward, as shown in Equation (8).

Here, we adopt the Multi-agent Deep Deterministic Policy Gradient (MADDPG) method [30], [31] as learning agents to solve this problem. MADDPG is one of the state-of-art techniques in Multi-agent Reinforcement Learning (MARL). The code refers (https://github.com/openai/maddpg). MADDPG is an extension of the actor-critic [29], [32] model. However, MADDPG has to train an independent policy network for each agent, where each agent would learn a policy specializing specific tasks [33] based on its own observation, and the policy network easily overfits to the number of agents. Therefore, MADDPG can hardly solve large-scale MARL problems.

III. PROPOSED NEW SMART ROUTER-THROTTLING METHOD

In this section, we first give an introduction to our proposed Centralized Reinforcement Learning Router Throttling (CRLRT) method, then introduce our less communication method.

A. CRLRT WITH CENTRALIZED COMMUNICATION

1) CENTRALIZED COMMUNICATION

Effective communication is a key ability for collaborative multi-agent systems. For the above MART method, each router agent picks action solely based on local observations. It is a cooperative multi-agent extension of a Partially Observable Markov Decision Process (POMDP) [34], [35]. As it is hard for those independent agents to conduct consistent to pass as much legitimate traffic as possible and at the same time make sure the total traffic is less than U_s.

We adopt a centralized communication mechanism. Centralized communication tackles the partial observability problem, where distributed agents cannot sense all of the relevant information necessary to complete a cooperative task. And a centralized decision is probably better than a decentralized decision to some extent. The **state space** of central agent is consisted of the traffic reading of N router agents, the server available bandwidth U_s^t (refer (11), will be introduced below) and the server upper boundary U_s. Here, the **state space** is $N+2$ dimensional and the **action space** is $A \in [0, 1]$ ($a^i = 0$ means all traffic of router i will be sent to servers; $a^i = 1$ means all traffic will be throttled).

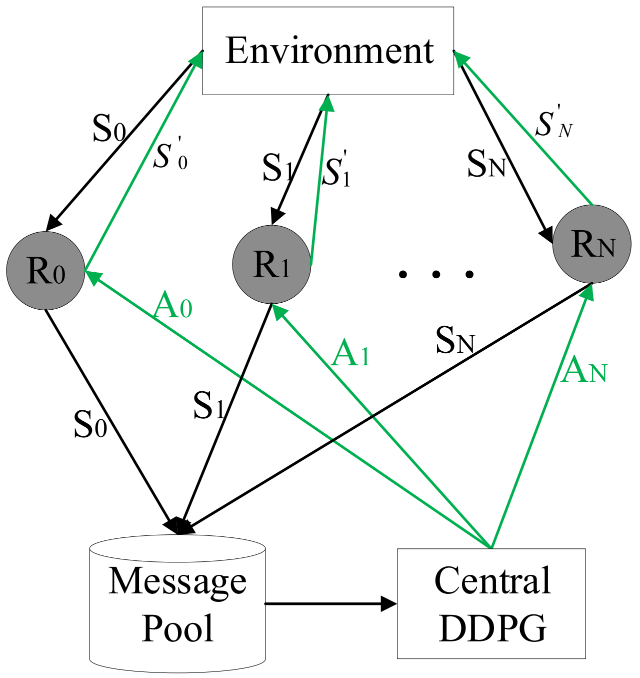


FIGURE 3. The framework of our Centralized Reinforcement Learning Router Throttling method. Each throttling router sends its state to the centralized agent, then the centralized agent makes decisions for each throttling router.

The overall framework of our CRLRT is illustrated in Fig.3. First of all, each agent (throttling router) gets state information from the **Environment** (network topology as shown in Fig.1) and sends its own traffic reading S_i to the central agent (**Central DDPG**). Then the central agent stores the received message into a **Message Pool** and makes the decision of the throttling actions for each router agent based on the message pool. After that, the central agent sends the action message A_i to each agent i, each throttling router (agent) throttles part of traffic based on the action message A_i, and sends the left traffic S_i' to the server, here $S_i' = S_i(1 - A_i)$. In the end, the environment will send back a reward R for the Central DDPG. We will store the $[S_0, \ldots, S_N, A_0, \ldots, A_N, R, S_0', \ldots, S_N']$ into a memory buffer. The Central DDPG can improve its policy with the stored information in the memory buffer.

2) MODIFIED REWARD FUNCTION

There is some problem with the reward function of Multi-agent Router Throttling (MART) method. For the DDoS problem (refer (1) and (2)). In a given state S, if agents can get excellent actions A, it can get a high reward, but if agents get actions near the optimal action and lead the total traffic more than U_s, it will get a punishment-reward of -1. Thus, the reward function $R(S, A)$ is an incoherent function for state action pair. What's more, the actions have no effect on the state of next time, thus, it is not a classical reinforcement learning problem.

To solve the above problems, we introduce the concept of the server available bandwidth U_s^t ($U_s^0 = U_s$) at moment t, it is updated as shown in Equation (11). If the total traffic Z_t (after routers throttling) is more than U_s^t, the bandwidth of next time U_s^{t+1} will be reduced. Otherwise, it will be increased (should be below the upper boundary U_s). When the next time U_s^{t+1} is less than 0, the server will be crashed, and we should restart the server. Therefore, if we add the server available bandwidth U_s^t into the state information, the actions can have an effect on the state of next time.

$$U_s^{t+1} = \min\{2 \times U_s^t - Z_t, U_s\} \tag{11}$$

Our modified reward function is shown as Equation (12), this is a coherent function for state action pair. If the total traffic Z_t (after routers throttling) is less than U_s^t, the reward function is the same as Equation (10). If the total traffic Z_t is more than U_s^t, some traffic cannot be responded by the server, so the reward function should be reduced by multiplied by U_s^t/Z_t (smaller than 1).

$$\begin{cases} R = \dfrac{\sum_{i=0}^{N} x_t^i(1-a_t^i)}{\sum_{i=0}^{N} x_t^i} & Z_t \le U_s^t \\[2ex] R = \dfrac{\sum_{i=0}^{N} x_t^i(1-a_t^i)}{\sum_{i=0}^{N} x_t^i} \times \dfrac{U_s^t}{Z_t} & Z_t > U_s^t \end{cases} \tag{12}$$

3) CURRICULUM LEARNING

As the reading of attack traffic can be as similar to legitimate traffic. It is difficult for agents to learn an ideal policy. We adopt curriculum learning [36], [37] to solve this problem. The main idea of curriculum learning is to decompose a hard learning task (target task) into several simple ones (subgoal tasks) [38]. Then, the learning agent can master subgoal tasks and reuse the gathered information to solve the target task. Which is faster than directly learning in the hard task from scratch. Here, we gradually increase the variance of normal traffic, so that at the beginning we can learn a policy much more easily, then gradually increase the difficulty.

B. CRLRT WITH LESS COMMUNICATION

As each throttling router (agent) needs to send information to the central agent, each time the central agent needs to receive N (the number of router agents) times communication information. Thus, we add a DDPG network (the detail introduction refers Section II-D.3) for each agent to decide whether or not to send information M_i to the central agent, as shown in Fig.4, which is based on Fig.3.

The input of the DDPG network for each throttling router is the pool (get from the prior time) and S_i, the output space of each agent is [−1,1], if the information of this time is similar to the prior information in the information pool, the router agent will not send information, otherwise it will send state information to the central agent. Then the central agent makes the decision of the throttling actions for each router agent based on the message pool and sends the action message A_i to each agent i, each throttling router (agent) throttles part of traffic based on the action message A_i, and send the left traffic S_i' to the server. In the end, the environment will send back a reward R for the Central DDPG. The Central DDPG can improve its policy with the stored information in the memory buffer.

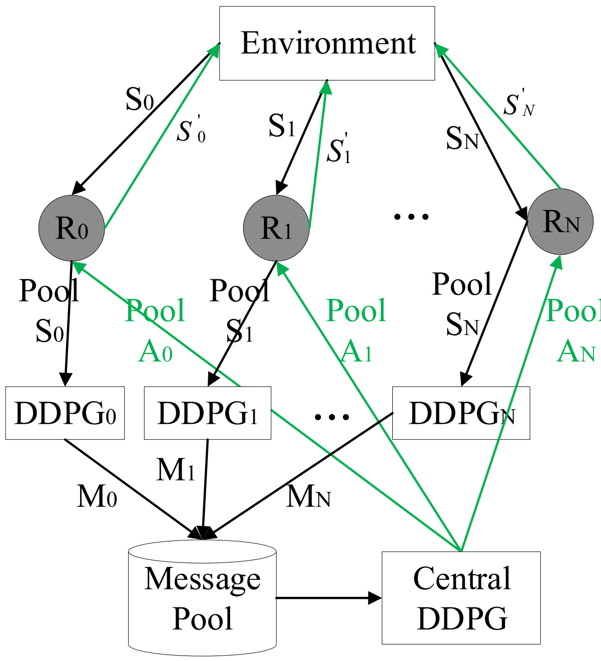


FIGURE 4. CRLRT with less communication structure. To decrease the communication cost, each throttling agent makes a decision of whether to send state to the centralized agent by a DDPG network.

IV. EXPERIMENTS AND RESULT ANALYSIS

In this section, we illustrate the performance of our proposed CRLRT architecture compared to SIRT, FRT, MART, and LP in both fixed attack location setting and dynamic attack location setting. All our experiments are conducted in an Ubuntu server with 64GB of RAM.

A. DATA COLLECTION AND EVALUATION METRICS

1) DATA COLLECTION

Prior experimental setup is based on Yau's work [9]. Each terminal is independently chosen to be a legitimate user and to be an attacker with probability to be 0.6 and 0.4. Legitimate users and attackers are to send traffic randomly and to uniformly choose from the given range [0, 1] and [2.5, 6] Mbit/s, respectively. However, this environment setting is too simple and ideal, in a real environment, the attack traffic can be as much as legitimate traffic to avoid been detected.

To validate the effectiveness of our proposed CRLRT method architecture, we adopt OPNET [39], [40] as our

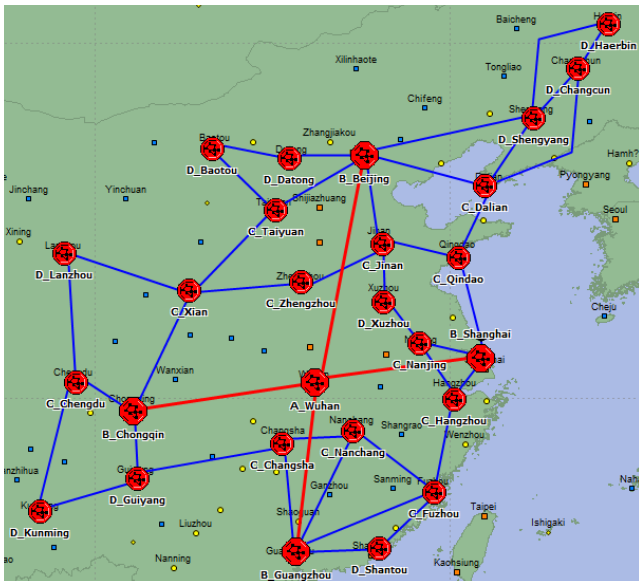


FIGURE 5. Network simulator based on OPNET.

network simulator to generate traffic data for our experiment. We simulate a national backbone network with 27 cities and each city has four throttling routers (total 108 router agents). As shown in Fig.5. There is a server located in the central city. In a real network, the legitimate traffic from different scale cities is different from each other. Thus, we choose 20 throttling routers as large-scale city routers and also choose 20 throttling routers as small-scale city routers, the left routers are media-scale city routers.

We simulate server episodes of data. Each episode lasts for 500 time steps, the attack traffic lasts for 300 time steps, at each time step each router generates some new traffic to the server. At the beginning of each episode, randomly choose N_{att} (54) throttling routers from the total N (108) throttling routers as the attack traffic. At each time, throttling routers (agents) can only sense the sum of legitimate traffic x_t^i and attack traffic y_t^i, but cannot distinguish between the two, and can select router throttling action a_t^i from the action range $[0, 1]$. The throttling action $a_t^i = 1$ means that all traffic is dropped, and $a_t^i = 0$ means all traffic can pass to the server. As heavy attack traffic can be detected easily, thus the attack traffic can be as much as legitimate traffic to avoid being detected. The legitimate traffic x_t^i and attack traffic y_t^i is considered to obey normal distribution. The load limits U_s of the victim server is shown as Equation (13).

$$U_s = \delta(\bar{x} \times N + \bar{y} \times N_{att}) \tag{13}$$

Here, $\bar{x}$ is the average of legitimate traffic, $\bar{y}$ is the average of attack traffic. $\delta \in [0, 1]$, N is the number of all the router agents, N_{att} is the number of router agents been attacked.

2) EVALUATION METRICS

During our experiment, there are three metrics, the reward curve during training and test period, the passing rate of legitimate traffic and the efficiency of action E_{action}.

For a Reinforcement Learning (RL) method, the reward function (10) is important to the training progress. In this paper, we will compare the reward curve (the change of reward in different time steps) of different methods to show the efficiency of methods.

For the DDoS problem, our goal is to keep the server not crashed and has less effect on legitimate users. Thus, the average passing rate of legitimate traffic $P_{legitimate}$ is the main evaluation metrics as Equation (14). Here, x_t^i is the legitimate traffic reading of router i at moment t, a_t^i is the throttling rate, T is a period of time steps, N is the number of router agents.

$$P_{legitimate} = \frac{1}{T}\sum_{t=0}^{T} \frac{\sum_{i=0}^{N} x_t^i(1 - a_t^i)}{\sum_{i=0}^{N} x_t^i} \tag{14}$$

We also analyze the efficiency of actions. The definition of the average attack passing rate P_{attack} is similar to (14), as shown in Equation (15), here y_t^i is the attack traffic of router i at moment t. As the upper boundary of the server is given (U_s), which means that the total (legitimate and attack) passing traffic should be controlled under U_s, we hope more and more legitimate traffic is passing to the server and at the same time the attack traffic passing rate will be reduced. Therefore, if the average legitimate passing rate $P_{legitimate}$ is much more than the average total passing rate P_{total} (as shown in Equation (16)), and the average attack passing rate P_{attack} is much less than total passing rate P_{total}, then the actions of the method are efficiency.

$$P_{attack} = \frac{1}{T}\sum_{t=0}^{T} \frac{\sum_{i=0}^{N} y_t^i(1 - a_t^i)}{\sum_{i=0}^{N} y_t^i} \tag{15}$$

$$P_{total} = \frac{1}{T}\sum_{t=0}^{T} \frac{\sum_{i=0}^{N} (x_t^i + y_t^i)(1 - a_t^i)}{\sum_{i=0}^{N} (x_t^i + y_t^i)} \tag{16}$$

The efficiency of action is defined as Equation (17). We hope the legitimate passing rate $P_{legitimate}$ is much more than the attack passing rate P_{attack}, at the same time, we should also make sure the legitimate passing rate $P_{legitimate}$ is as much as possible.

$$E_{action} = \frac{P_{legitimate} - P_{attack}}{1 - P_{legitimate}} \tag{17}$$

B. BASELINES AND PARAMETER SETTING

We compare our CRLRT method with baseline methods (SIRT, FRT, MART, and LP) in both fixed attack location setting and dynamic attack location setting.

1) BASELINES

- **Server-initiated Router Throttling (SIRT)**. As introduced in Section II-B. Each router agent shares the same throttling rate [9]. The server adjusts the throttling rate to make sure the accumulated traffic is within $[L_s, U_s]$ and sends back throttling rate information to each router agent.
- **Fair Router Throttle (FRT)**. As introduced in II-C. Each throttling router has a different throttling rate [9].

Each throttling agent adjusts its own throttling rate to make sure the accumulated traffic is within $[L_s, U_s]$.

- **Multi-agent Router Throttling (MART)**. This method relies on only local information and there is no communication between each router.
- **Linear Programming (LP)**. For the simulated environment, we can get the legitimate traffic x_t^i and attack traffic y_t^i. Thus, we can get the optimal result by LP as Equation (1) and (2). Note that LP is hardly impossible to implement in reality, we only use it as an upper bound for all kinds of router throttling algorithms.

2) PARAMETER SETTING

During our experiment, the parameter of U_s is set $\delta = 0.85$ (refer (13)), the number of total router $N = 108$, the number of attack router $N_{att} = 54$, the legitimate traffic x_t^i and attack traffic y_t^i is sampled from a different Gaussian distribution, the legitimate traffic of large-scale $x_t^i \sim N(1000{,}100)$, media-scale $x_t^i \sim N(800{,}100)$, and small-scale $x_t^i \sim N(600{,}100)$, $y_t^i \sim N(800{,}100)$.

To make fair comparisons, we adopt a grid search [41], [42] to find the optimal parameter for each algorithms. The optimal parameter of SIRT is $a_0 = 0$, $\beta = 0.05$, $\alpha = 0.2$, $L_s = 0.9 \times U_s$, we can refer (4) for detail definition. The optimal parameter of FRT is $R_s(0) = U_s/50$, $\beta = 0.05$, $\alpha = 0.5$, $L_s = 0.8 \times U_s$, we can refer (6) for detail definition. Our reinforcement learning algorithm is DDPG, the DDPG framework and parameter during training period are shown in Table.1. Each layer is connected with an ReLU [43], [44] function, the optimizer of the network is Adam [45]. The explore noise is begin with 0.2, and each step decrease with 0.2/Menory-size, end with 0.001.

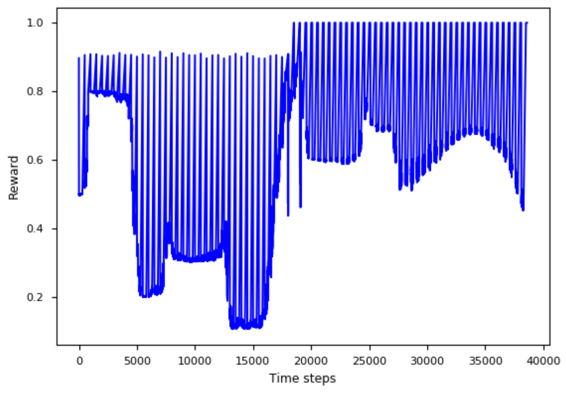
TABLE 1. DDPG framework and paremeters.

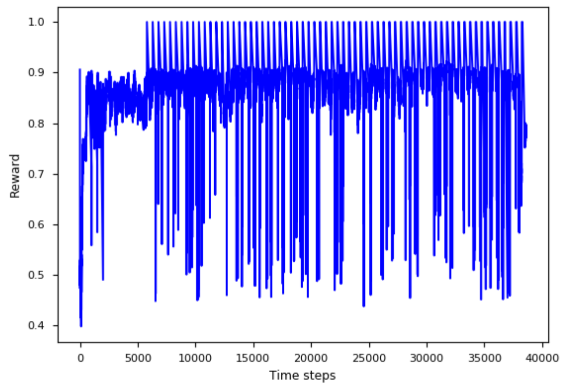
Name	Reference
Actor network	Three Layer MLP(64 units),ReLU
Critic network	Three Layer MLP(64 units),ReLU
Batch Size	64
Discount factor	0.99
Memory Size	6000
Actor Learning Rate	0.001
Critic Learning Rate	0.0001
Optimizer	Adam
Regularization	L2

The training period lasts for 900 episodes, and the test period lasts for 100 episodes, each episode lasts for 500 time steps, the attack traffic occurs in the first 300 time steps. We compare the metrics of different methods in both fixed attack location setting and dynamic attack location setting.

C. FIXED ATTACK LOCATION

For the fixed attack location setting, the location of attack agents is fixed during training and test period and with the



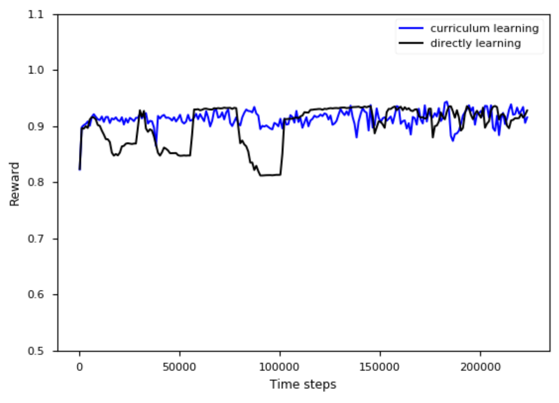


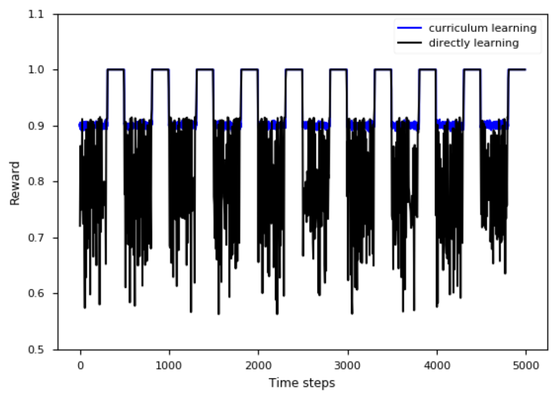
(b) CRLRT

FIGURE 6. The comparison of MART and CRLRT during training period.

same reward function. To validate the centralized decision is better than the decentralized decision in this environment setting, we plot the reward curve of MART and CRLRT during the test period, as illustrated in Fig.6, here, $y_t^i \sim N(800{,}100)$, $\delta = 0.85$. The training period of reinforcement learning is a little different from supervised learning, as reinforcement learning need to explore new action during training, the learning curve will rise volatility. From the darker area of Fig.6, we can see that the training period of our CRLRT method tends to be more stable than the MART method. And our method can learn an ideal policy faster than MART, as our CRLRT can get the reward of 0.9 in the first 10000 steps.

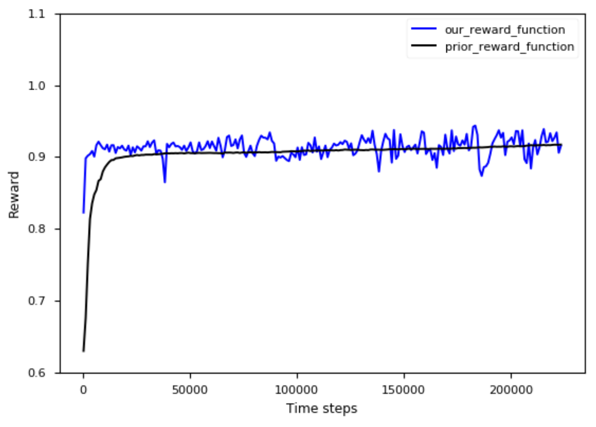
We then compare curriculum learning with directly learning. Here, $y_t^i \sim N(800{,}100)$, $\delta = 0.85$. We plot the training period of curriculum learning and directly learning in Fig.7(a). Here, we plot the average reward of each 1000 time steps for display and comparison more clearly. We also plot the test period in Fig.7(b), here we just show the first 10 episode result for display and comparison more clearly. From Fig.7(a), we can see that the training period of our CRLRT with curriculum learning tends to be more stable than directly learning. From Fig.7(b), we can see that the policy learned by CRLRT with curriculum learning is much better than directly learning, the test reward result of our CRLRT with curriculum learning is around 0.9, while the



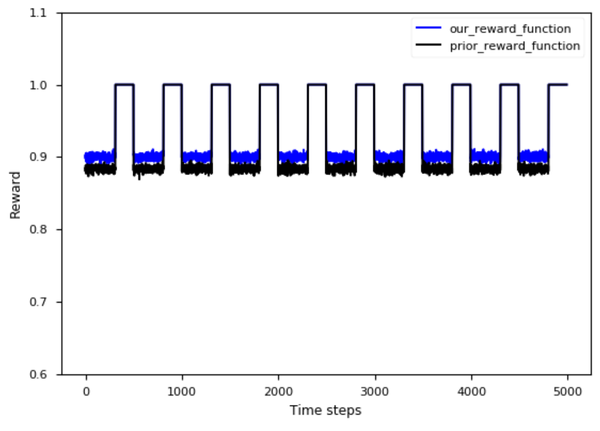


(b) Test

FIGURE 7. The comparison of curriculum learning and directly learning.



(a) Training



(b) Test

FIGURE 8. The reward function comparison of our method and prior method.

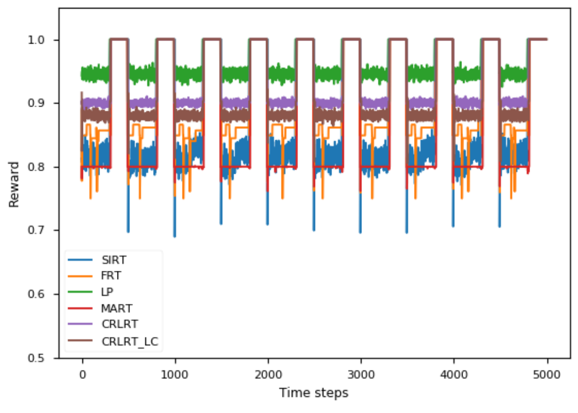


FIGURE 9. The reward curve comparison in fixed attack location.

directly learning reward is very unstable, and the reward is around 0.8, much less than our reward result of 0.9.

We also compared our reward function with prior reward function of MART. Here, $y_t^i \sim N(800,100)$, $\delta = 0.85$. From Fig.8(a), we can see that our training stage is not stable as the stage of prior reward, this is due to our environment setting is more complicated and more reality than prior environment setting, but the average reward of our reward function is a little higher than the prior reward function. From Fig.8(b), we can see that the policy learned by CRLRT with our reward function is a little better than the prior reward function of MART.

The test reward of different methods is shown in Fig.9. Here, in order to show more clearly, we just plot the first 10 episodes of reward. From the result, we can see that the performance of CRLRT is best among these methods besides Linear Programming (LP). The LP method can achieve a very high reward because it supposes that we could know the legitimate traffic x_t^i and attack traffic y_t^i in advance, but in reality, we can not know this information in advance. The result of the LP method can just be an optimal result for comparison. The reward curve of CRLRT-LC is a little worse than CRLRT, as the central agent of CRLRT-LC method used less information than CRLRT. There is still a big gap between our result with the optimal result got by LP.

In order to analyze the efficiency of actions learned by each method. We list the total passing rate P_{total}, the legitimate passing rate $P_{legitimate}$, and the attack passing rate P_{attack} (refer (14)). They are used to calculate the action efficiency E_{action} (refer (17)) during test period. We hope P_{total}, $P_{legitimate}$ and E_{action} to be as large as possible, the P_{attack} to be as small as possible. The metrics result of different methods

TABLE 2. Metrics of different router throttling methods in fixed attack location, the Total is the average passing rate of total traffic during test period, Legitimate and Attak are the average passing rate of legitimate and attack traffic respectively, E_{action} is the efficiency of actions taken by each method.

Methods	Total	Legitimate	Attack	E_{action}
SIRT	0.7675	0.8199	**0.6351**	1.0261
FRT	0.8951	0.9062	0.8515	0.5832
MART	0.7969	0.8078	0.7923	0.0806
LP	0.9052	**0.9453**	0.8063	**2.5411**
CRLRT	**0.9085**	0.9367	0.7955	2.2306
CRLRT-LC	0.8876	0.9158	0.7964	1.4181

is shown in Table.2. We can see that the action efficiency of SIRT, FRT and, MART is quite poor. We save the action result of MART and can find that all agents take similar actions, which indicates that the actions are useless, thus the action efficiency of MART is the worst of all. The action efficiency E_{action} of LP, our CRLRT, and CRLRT-LC is consistent with the result showed in Fig.9. Our CRLRT is better than other methods besides the LP method, and our CRLRT-LC is a little worse than CRLRT as CRLRT-LC uses less information.

D. DYNAMIC ATTACK LOCATION

As in reality, the attacker can dynamically change the attack location to avoid been detected. For dynamic attack location setting, the location of attack agents is dynamically changed for each episode and test period. The parameter setting is the same as the fixed attack location environment. We first generate 200 different attack locations. During the 900 episodes training period, use the first 100 attack location repeatedly. During test 100 episodes, use the left 100 attack location for the test of the learned policy.

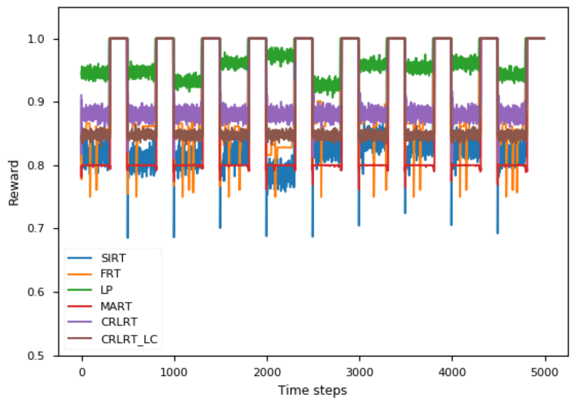


FIGURE 10. The reward curve comparison in dynamic attack location.

The test reward of different methods in the dynamic attack location setting is shown in Fig.10. Here, we also just plot the first 10 episodes of reward. From the result, we can see that the reward of each method is similar to Fig.9. In the dynamic attack location, the reward of Linear Programming (LP) has an obvious fluctuation. CRLRT is still best among these methods besides LP. The LP method can still achieve a very high reward. The reward of CRLRT with less communication (CRLRT-LC) is a little worse than CRLRT, as the central agent of this method used less information than CRLRT.

TABLE 3. Metrics of different router throttling methods in dynamic attack location.

Methods	Total	Legitimate	Attack	E_{action}
SIRT	0.7696	0.8220	**0.6354**	1.0483
FRT	0.8957	0.9068	0.8516	0.5923
MART	0.7999	0.8002	0.7994	0.0040
LP	**0.8998**	**0.9431**	0.7958	**2.5888**
CRLRT	0.8838	0.9177	0.7878	1.5783
CRLRT-LC	0.8759	0.8987	0.7955	1.0188

In order to analyze the efficiency of actions learned by each method in the dynamic attack location setting. We also list the total passing rate P_{total}, the legitimate passing rate $P_{legitimate}$, and the attack passing rate P_{attack}, and the action efficiency E_{action} during test period as shown in Table.3. The table setting is similar to Table.2. We can see that the result of SIRT, FRT, MART, and LP in dynamic attack location setting is similar to the fixed attack location setting, but the result of our CRLRT and CRLRT-LC in dynamic attack location setting is much less than fixed attack location setting. This is mainly because that the SIRT, FRT, and LP only use the traffic reading itself, do not care about the location of attack routers. But our CRLRT and CRLRT-LC are influenced by the attack location, as different attack location will lead to a quite different state for agents. The result of LP, our CRLRT, and CRLRT-LC is consistent with the result showed in Fig.10. The LP result is still the best, our CRLRT is better than other methods besides the LP method, and CRLRT-LC is a little worse than CRLRT.

V. CONCLUSION

In this paper, to solve the Partial Observability Markov Decision Problem of the DDoS attack, we make three contributions to mitigate the effect of the DDoS attack. First, we propose a Centralized Reinforcement Learning Router Throttling with Less Communication (CRLRT-LC) method to defend DDoS attacks. The central router makes a decision for each router to choose the throttling rate. Second, we simulate the environment of the DDoS problem more realistic and modify the reward function of MART to make the reward function more coherent. Third, we add a DDPG network for each router to decrease the communication time. For this DDoS problem, experimental results validate that our methods outperform existing router throttling methods. But there is still a big gap between our result with the optimal result produced by Linear Programming (LP), especially in dynamic attack location setting.

In future work, first, we will try to adopt imitate learning to imitate the policy of LP. Second, we will extend our work to a larger scale network using Hierarchical

Reinforcement Learning. Third, we will add communication delay to the communication to make the environment setting more realistic.

REFERENCES

[1] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[2] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proc. 10th IEEE Int. Conf. Netw. Protocols*, Nov. 2002, pp. 312–321.

[3] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017. doi: 10.1109/ACCESS.2017.2762418.

[4] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004.

[5] W. J. Blackert, D. M. Gregg, A. K. Castner, E. M. Kyle, R. L. Hom, and R. M. Jokerst, "Analyzing interaction between distributed denial of service attacks and mitigation technologies," in *Proc. DARPA Inf. Survivability Conf. Expo.*, Washington, DC, USA, vol. 1, Apr. 2003, p. 26–39. doi: 10.1109/DISCEX.2003.1194870.

[6] E. Bertino and N. Islam, "Botnets and Internet of Things security," *Computer*, vol. 50, no. 2, pp. 76–79, Feb. 2017. doi: 10.1109/MC.2017.62.

[7] F. Restuccia, S. D'oro, and T. Melodia, "Securing the Internet of Things in the age of machine learning and software-defined networking," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4829–4842, Dec. 2018. doi: 10.1109/JIOT.2018.2846040.

[8] J. Mirkovic, M. Robinson, and P. Reiher, "Alliance formation for DDoS defense," in *Proc. New Secur. Paradigms Workshop*, Ascona, Switzerland, Aug. 2003, pp. 11–18. doi: 10.1145/986655.986658.

[9] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 29–42, Feb. 2005.

[10] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, S. Liu, J. P. How, and J. Vian, "Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 231–258, Feb. 2017. doi: 10.1177/0278364917692864.

[11] F. de Nijs, "Resource-constrained multi-agent Markov decision processes," Ph.D. dissertation, Delft Univ. Technol., Delft, The Netherlands, 2019. [Online]. Available: http://resolver.tudelft.nl/uuid:89c0f1a2-d19f-4466-9cc5-52aeb3950e53

[12] K. Malialis and D. Kudenko, "Multiagent router throttling: Decentralized coordinated response against DDoS attacks," in *Proc. 25th AAAI Conf. Artif. Intell.*, Jun. 2013, pp. 1–6.

[13] K. Malialis and D. Kudenko, "Distributed response to network intrusions using multiagent reinforcement learning," *Eng. Appl. Artif. Intell.*, vol. 41, pp. 270–284, May 2015.

[14] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," 2018, *arXiv:1805.07733*. [Online]. Available: https://arxiv.org/abs/1805.07733

[15] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Comput. Netw.*, vol. 44, no. 5, pp. 643–666, Apr. 2004.

[16] A. Kumar, S. S. Appadoo, and C. R. Bector, "A note on 'Generalized fuzzy linear programming for decision making under uncertainty: Feasibility of fuzzy solutions and solving approach,'" *Inf. Sci.*, vol. 266, pp. 226–227, May 2014. doi: 10.1016/j.ins.2014.01.020.

[17] D. Tsimpoukis, T. Baarslag, M. Kaisers, and N. G. Paterakis, "Automated negotiations under user preference uncertainty: A linear programming approach," in *Proc. Int. Conf. Agreement Technol.*, Bergen, Norway, Dec. 2018, pp. 115–129. doi: 10.1007/978-3-030-17294-7_9.

[18] M. Prodel, V. Augusto, and X. Xie, "Hospitalization admission control of emergency patients using markovian decision processes and discrete event simulation," in *Proc. Winter Simul. Conf.*, Savannah, GA, USA, Dec. 2014, pp. 1433–1444. doi: 10.1109/WSC.2014.7019997.

[19] R. Heidari, M. Afsharchi, and R. Khanmohammadi, "A markovian decision process analysis of experienced agents joining Ad-Hoc teams," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Prague, Czech Republic, Aug. 2018, pp. 691–698. doi: 10.1109/DSD.2018.00007.

[20] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, Mar. 2014, pp. 387–395.

[21] N. Casas, "Deep deterministic policy gradient for urban traffic light control," 2017, *arXiv:1703.09035*. [Online]. Available: https://arxiv.org/abs/1703.09035

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations, (ICLR)*, San Juan, Puerto Rico, 2016. [Online]. Available: http://arxiv.org/abs/1509.02971

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: https://arxiv.org/abs/1312.5602

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[25] X. Li, J. Fang, W. Cheng, H. Duan, Z. Chen, and H. Li, "Intelligent power control for spectrum sharing in cognitive radios: A deep reinforcement learning approach," *IEEE Access*, vol. 6, pp. 25463–25473, 2018. doi: 10.1109/ACCESS.2018.2831240.

[26] V. R. Konda and J. N. Tsitsiklis, "Onactor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[27] V. Gullapalli, "A stochastic reinforcement learning algorithm for learning real-valued functions," *Neural Netw.*, vol. 3, no. 6, pp. 671–692, 1990.

[28] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[29] X. Bu, "Actor-critic reinforcement learning control of non-strict feedback nonaffine dynamic systems," *IEEE Access*, vol. 7, pp. 65569–65578, 2019. doi: 10.1109/ACCESS.2019.2917141.

[30] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 6382–6393. [Online]. Available: http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments

[31] G. Wen, C. L. P. Chen, J. Feng, and N. Zhou, "Optimized multi-agent formation control based on an identifier–actor–critic reinforcement learning algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 5, pp. 2719–2731, Oct. 2018. doi: 10.1109/TFUZZ.2017.2787561.

[32] V. S. Borkar, "An actor-critic algorithm for constrained Markov decision processes," *Syst. Control Lett.*, vol. 54, no. 3, pp. 207–213, Mar. 2005. doi: 10.1016/j.sysconle.2004.08.007.

[33] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, Sydney, NSW, Australia, Aug. 2017, pp. 1995–2003. [Online]. Available: http://proceedings.mlr.press/v70/le17a.html

[34] F. Wu, S. Zilberstein, and N. R. Jennings, "Monte-Carlo expectation maximization for decentralized POMDPs," in *Proc. 23rd Int. Joint Conf. Artif. Intell. IJCAI*, Beijing, China, Jun. 2013, pp. 397–403. [Online]. Available: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6789

[35] A. Saffidine, F. Schwarzentruber, and B. Zanuttini, "Knowledge-based policies for qualitative decentralized POMDPs," in *Proc. 22nd Conf. Artif. Intell., (AAAI)*, New Orleans, Louisiana, USA, Feb. 2018, pp. 6270–6277. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17029

[36] N. Zheng and P. Mazumder, "Hardware-friendly actor-critic reinforcement learning through modulation of spike-timing-dependent plasticity," *IEEE Trans. Comput.*, vol. 66, no. 2, pp. 299–311, Feb. 2017.

[37] A. Koenecke and A. Gajewar, "Curriculum learning in deep neural networks for financial forecasting," 2019, *arXiv:1904.12887*. [Online]. Available: https://arxiv.org/abs/1904.12887

[38] S. Narvekar, J. Sinapov, M. Leonetti, and P. Stone, "Source task creation for curriculum learning," in *Proc. 2016 Int. Conf. Auto. Agents Multiagent Syst.*, Singapore, May 2016, pp. 566–574. [Online]. Available: http://dl.acm.org/citation.cfm?id=2937007

[39] X. Chang, "Network simulations with OPNET," in *Proc. Winter Simulation Conf. Simulation Bridge Future*, Phoenix, AZ, USA, vol. 1, Dec. 1999, pp. 307–314. doi: 10.1109/WSC.1999.823089.

[40] L. Nie and S. Hu, "Simulation and analysis of campus network based on OPNET," *J. Comput. Methods Sci. Eng.*, vol. 19, no. 1, pp. 3–12, Jan. 2019. doi: 10.3233/JCM-180847.

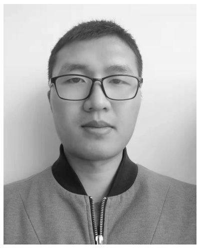
[41] P. Zhao, S. C. H. Hoi, R. Jin, and T. Yang, "Online AUC maximization," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, Bellevue, WA, USA, Jun. 2011, pp. 233–240. [Online]. Available: https://icml.cc/2011/papers/198_icmlpaper.pdf

[42] X. Wang, G. Gong, and N. Li, "Automated recognition of epileptic EEG states using a combination of Symlet wavelet processing, gradient boosting machine, and grid search optimizer," *Sensors*, vol. 19, no. 2, p. 219, Jan. 2019. doi: 10.3390/s19020219.

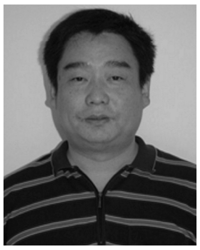
[43] B. Neyshabur, Y. Wu, R. Salakhutdinov, and N. Srebro, "Path-normalized optimization of recurrent neural networks with relu activations," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 3477–3485. [Online]. Available: http://papers.nips.cc/paper/6214-path-normalized-optimization-of-recurrent-neural-networks-with-relu-activations

[44] K. Eckle and J. Schmidt-Hieber, "A comparison of deep networks with ReLU activation function and linear spline-type methods," *Neural Netw.*, vol. 110, pp. 232–242, Feb. 2019. doi: 10.1016/j.neunet.2018.11.005.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

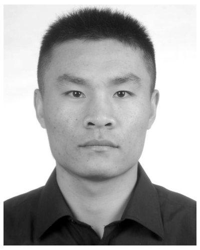


SHI-MING XIA received the M.S. degree in radar engineering and the M.S. degree in information and communication engineering from the PLA University of Science and Technology, Nanjing, China, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree in computer science with the College of Command and Control Engineering, Army Engineering University of PLA, Nanjing, China. His research interest includes network security and reinforcement learning.

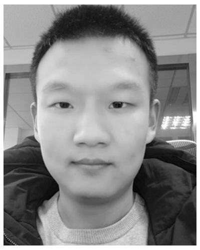


SHI-ZE GUO was born in 1964. He received the B.S. and M.S. degrees from the Ordnance Engineering College, China, in 1991 and 1988, respectively, and the Ph.D. degree from the Harbin Institute of Technology, in 1989. He is currently a Searcher with The Institute of North Electronic Equipment and also a Professor with the Beijing University of Posts and Telecommunications. His current research interests include information security and cryptography.

WEI BAI was born in 1983. He received the M.S. degree from the PLA University of Science and Technology, Nanjing, China, in 2008, where he is currently pursuing the Ph.D. degree. Since 2012, he has been a University Lecturer in network security, especially in security policy and security management.



JUN-YANG QIU received the B.S. degree from the PLA University of Science and Technology, in 2012, and the M.S. degree from the Nanjing University of Posts and Telecommunications, in 2015. He is currently pursuing the Ph.D. degree with the School of Information Technology, Deakin University. His research interests include cyber security and machine learning.



HAO WEI received the M.S. degree in information and communication engineering from the PLA University of Science and Technology, in 2016. He is currently pursuing the Ph.D. degree with the Army Engineering University of PLA. His research interests include complex network in machine learning, network embedding, and online time series prediction.

ZHI-SONG PAN received the B.S. degree in computer science and the M.S. degree in computer science and application from PLA Information Engineering University, Zhengzhou, China, in 1991 and 1994, respectively, and the Ph.D. degree from the Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2003. Since July 2006, he has been leading several key projects of intelligent data processing for the network management. Since July 2011, he has been a Full Professor with the PLA University of Science and Technology, China. His current research interests include pattern recognition, machine learning, and neural networks.