Large-Scale DDoS Response Using Cooperative Reinforcement Learning

Kleanthis Malialis and Daniel Kudenko

Department of Computer Science
University of York, UK
{malialis,kudenko}@cs.york.ac.uk

Abstract. Distributed denial of service (DDoS) attacks constitute a rapidly evolving threat in the current Internet. Multiagent Router Throttling has been demonstrated to perform well against DDoS attacks in small-scale network topologies. In this paper we tackle the scalability challenge. Scalability is arguably one of the most important aspects of a defence system since a non-scalable defence mechanism will never be adopted for wide deployment by a company or organisation. We introduce a novel design by incorporating several mechanisms; these are (i) team formation and role specialisation (ii) hierarchical communication (iii) task decomposition and (iv) reward shaping, and we show that our approach can significantly scale-up. We compare our proposed approach against a baseline and a popular state of the art throttling technique from the literature and we show that it outperforms both of them.

1 Introduction

One of the most serious threats in the current Internet is posed by distributed denial of service (DDoS) attacks, which target the availability of a system [1]. A DDoS attack is a highly coordinated attack where the attacker (or attackers) takes under his control a large number of hosts, called the botnet (network of bots), which start bombarding the target when they are instructed to do so. Such an attack is designed to exhaust a server's resources or congest a network's infrastructure, and therefore renders the victim incapable of providing services to its legitimate users.

Router Throttling [2] is a popular approach to defend against DDoS attacks, where the victim server signals a set of upstream routers to throttle traffic towards it. Similar techniques to throttling are implemented by network operators [1]. Multiagent Router Throttling [3] is a novel throttling approach where multiple reinforcement learning agents are installed on the set of upstream routers and learn to throttle traffic towards the victim server. The approach has been demonstrated to perform well against DDoS attacks in small-scale network topologies, but suffers from the "curse of dimensionality" [4] when scaling-up. In this paper we focus on scalability and we propose a novel design that resolves this problem. Our contributions in this paper are the following.

We show that our proposed approach can significantly scale-up by combining several mechanisms; these are, team formation and role specialisation, hierarchical communication, task decomposition and reward shaping. We demonstrate this in a probabilistic domain with experiments involving 30 reinforcement learning agents. We note that our approach can be useful in other related multiagent domains. Due to the high complexity and multidimensionality of the DDoS threat, our approach creates an autonomous and effective response. The network environment is highly dynamic and our approach provides adaptable behaviours over frequent environmental changes. We compare our approach against a baseline and a popular throttling technique from the network security literature [2]. We evaluate our approach in a series of attack scenarios with increasing sophistication and we show that our proposed approach outperforms the two techniques.

One of the novel characteristics of our approach is its decentralised architecture and response to the DDoS threat. We show that our approach is more secure than the two existing throttling techniques [2], since it does not have a single point of failure. Furthermore, there is an extensive literature regarding the application of machine learning to intrusion detection, specifically anomaly detection where no action is performed beyond triggering an intrusion alarm when an anomalous event is detected. Our work investigates the applicability of multiagent systems and machine learning to intrusion *response*. To the best of our knowledge, the introduction of Multiagent Router Throttling in [3] constitutes the first time multiagent learning is used for DDoS response.

2 Background

2.1 Reinforcement Learning

Reinforcement learning is a paradigm in which an active decision-making agent interacts with its environment and learns from reinforcement, that is, a numeric feedback in the form of reward or punishment [4]. The feedback received is used to improve the agent's actions. Typically, reinforcement learning uses a Markov Decision Process (MDP) as a mathematical model.

An MDP is a tuple $\langle S, A, T, R \rangle$, where S represents the state space, A represents the action space, $T(s, a, s') = Pr(s'|s, a)$ is the transition probability function which returns the probability of reaching state s' when action a is executed in state s, and $R(s, a, s')$ is the reward function which returns the immediate reward r when action a executed in state s results in a transition to state s'. The problem of solving an MDP is to find a policy (i.e. a mapping from states to actions) which maximises the accumulated reward. When the environment dynamics (transition probabilities and reward function) are available, this task can be solved using dynamic programming [5].

In most real-world domains, the environment dynamics are not available and therefore the assumption of perfect problem domain knowledge makes dynamic programming to be of limited practicality. The concept of an iterative approach constitutes the backbone of the majority of reinforcement learning algorithms.

These algorithms apply so called temporal-difference updates to propagate information about values of states, $V(s)$, or state-action, $Q(s, a)$, pairs. These updates are based on the difference of the two temporally different estimates of a particular state or state-action value. The SARSA algorithm is such a method [4]. After each real transition, $(s, a) \to (s', r)$, in the environment, it updates state-action values by the formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

where α is the rate of learning and γ is the discount factor. It modifies the value of taking action a in state s, when after executing this action the environment returned reward r, moved to a new state s', and action a' was chosen in state s'.

The exploration-exploitation trade-off constitutes a critical issue in the design of a reinforcement learning agent. It aims to offer a balance between the exploitation of the agent's knowledge and the exploration through which the agent's knowledge is enriched. A common method of doing so is ϵ-greedy, where the agent behaves greedily most of the time, but with a probability ϵ it selects an action randomly. To get the best of both exploration and exploitation, it is advised to reduce ϵ over time [4].

Applications of reinforcement learning to multiagent systems typically take one of two approaches; multiple individual learners or joint action learners [6]. Multiple learners use a single-agent reinforcement learning algorithm and assume any other agents to be a part of the environment and so, as the others simultaneously learn, the environment appears to be dynamic as the probability of transition when taking action a in state s changes over time. To overcome the appearance of a dynamic environment, joint action learners were developed that extend their value function to consider for each state the value of each possible combination of actions by all agents. The consideration of the joint action causes an exponential increase in the number of values that must be calculated with each additional agent.

2.2 Distributed Denial of Service

A DDoS attack is a highly coordinated attack; the strategy behind it is described by the agent-handler model [1]. The model consists of four elements, the attackers, handlers, agents and victim. A handler (or master) and the agent (or slave or deamon or zombie) are hosts compromised by the attackers, which constitute the botnet. Specifically, the attackers install a malicious software called Trojan on vulnerable hosts to compromise them, thus being able to communicate with and control them. The attackers communicate with the handlers, which in turn control the agents in order to launch a DDoS attack. Moreover, an attacker can use IP spoofing, that is, hiding his true identity by placing a fake source address in the IP packet's source address.

The DDoS threat is challenging for many reasons [7]. Firstly, the traffic flows originate from agent machines spread all over the Internet, which they all aggregate at the victim. Then there is the large volume of the aggregated traffic, which is unlikely to be stopped by a single defence point near the victim. The number of compromised agent machines is also large, thus making an automated response a necessary requirement. Another challenge is that DDoS packets appear to be similar to legitimate ones, since the victim damage is caused by the aggregated volume and not packet contents. A defence system is thus required to keep aggregated statistical data in order to correlate packets and detect anomalies. Finally, it is difficult to discover even the agent machines, let alone the actual attackers, firstly because of the botnet architecture and secondly because of IP spoofing. It is evident that to combat the distributed nature of these attacks, a distributed coordinated defence mechanism is necessary, where many defensive nodes, across different locations cooperate in order to stop or reduce the flood.

3 Network Model and Assumptions

The network model is similar to the one used by Yau et al [2]. A network is a connected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. All leaf nodes are hosts and denoted by H. Hosts can be traffic sources and are not trusted because of IP spoofing. An internal node represents a router, which forwards or drops traffic received from its connected hosts or peer routers. The set of routers are denoted by R, and they are assumed to be trusted, i.e. not to be compromised. This assumption is rational since it is much more difficult to compromise a router than an end host or server, because routers have a limited number of potentially exploitable services [8]. The set of hosts $H = V - R$ is partitioned into the set of legitimate users and the set of attackers. A leaf node denoted by S represents the victim server. Consider for example the network topology shown in Figure 1. It consists of 20 nodes, these are, the victim server denoted by S, 13 routers denoted by R_s and $R2 - R13$ and six end hosts denoted by $H1 - H6$, which are traffic sources towards the server.

A legitimate user sends packets towards the server S at a rate r_l, and an attacker at a rate r_a. We assume that the attacker's rate is significantly higher than that of a legitimate user, that is, $r_a >> r_l$ [1]. This assumption is based on the rationale that if an attacker sends at a similar rate to a legitimate user, then the attacker must recruit a considerably larger number of agent hosts in order to launch an attack with a similar effect [2]. A server S is assumed to be working normally if its load r_s is below a specified upper boundary U_s, that is, $r_s \leq U_s$ (this includes cases of weak or ineffective DDoS attacks). The rate r_l of a legitimate user is significantly lower than the upper boundary i.e. $r_l << U_s$, where U_s can be determined by observing how users normally access the server.

[1] Dropping traffic based on host addresses can be harmful because, as mentioned, hosts cannot be trusted.

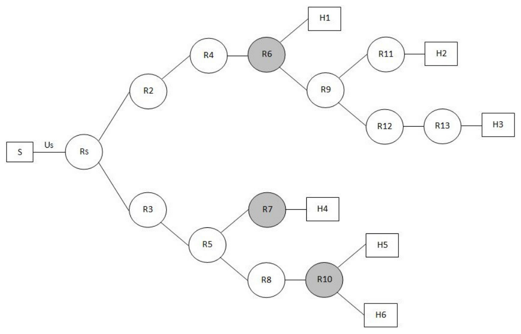

Fig. 1: Network topology showing defensive routers

4 Multiagent Router Throttling

4.1 Basic Design

Agent Selection: The selection method is similar to the one used by Yau et al [2]. Reinforcement learning agents are installed on locations that are determined by a positive integer k, and are given by $R(k) \subseteq R$. $R(k)$ is defined as the set of routers that are either k hops away from the server, or less than k hops away but are directly attached to a host. The effectiveness of throttling increases with an increasing value of k, provided that routers in $R(k)$ must belong to the same administrative domain e.g. an Internet Service Provider (ISP) or collaborative domains[2]. We emphasise that no topology modelling is required. Consider for example the network topology shown in Figure 1. Learning agents are installed on the set $R(5)$, which consists of routers $R6, R7$ and $R10$.

State Space: Firstly, the *aggregate* is defined as the traffic that is directed towards a specific destination address i.e. the victim (note that source addresses cannot be trusted due to IP spoofing) [9]. Each agent's state space consists of a single state feature, which is its average *load*. The average load is defined as the aggregate traffic arrived at the router over the last T seconds, which is called the monitoring window. The time step of the learning algorithm is set to be the same as the monitoring window size.

Action Space: Each router applies throttling via probabilistic traffic dropping. For example action 0.4 means that the router will drop (approximately) 40% of its aggregate traffic towards the server, thus setting a throttle or allowing only 60% of it to reach the server. The action is applied throughout the time step. Completely shutting off the aggregate traffic destined to the server is prohibited, that is, the action 1.0 (which corresponds to 100% drop probability) is

[2] Collaboration between different administrative domains is desirable but the motivation for deployment is low. This limitation may be overcome in the future by legislation enforcement.

not included in the action space of any of the routers. The reason being that the incoming traffic likely contains some legitimate traffic as well, and therefore dropping all the incoming traffic facilitates the task of the attacker, which is to deny legitimate users access to the server [9].

Reward Function: Each agent has the same reward function and therefore receives the same reward or punishment. The system has two important goals, which are directly encoded in the reward function. The first goal is to keep the 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 server during a period of congestion. In this case, the system receives a reward of $L \in [0, 1]$, where L denotes the proportion of the legitimate traffic that reached the server during a time step. The global reward function is shown in Algorithm 1.

4.2 Team Formation and Role Specialisation

The first step towards scalability, is to form teams of agents as shown in Figure 2. Dashed lines do not necessarily represent nodes with a direct connection. In other words, our proposed system does not require a contiguous deployment. Teams can either be homogeneous or heterogeneous. The structure of the team is shown in Figure 3. Each team consists of its leader, an inner layer of intermediate routers, and the throttling routers which are k hops away from the server. Recall that the throttling routers belong to the same administrative domain. In case of collaborative domains, each team can belong to a different administrative domain. Note that only the throttling routers are learning agents. The rest of the routers are non-learning agents and we will explain their role in the next section. The number of teams and their type (homogeneous or heterogeneous) depend on the size and structure of the administrative domain and the network model.

4.3 Hierarchical Communication

Our first proposed approach involves communication. Communication tackles the hidden state problem, where distributed agents cannot sense all of the rel-

```
if $loadRouter_{server} > U_s$
then
    // Punishment
    $r = -1$
else
    // Reward in [0,1]
    $r = legitimateLoad_{server}/legitimateLoad_{total}$
end
```

Algorithm 1: Global Reward Function

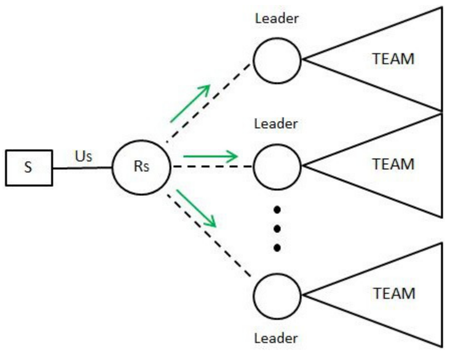

Fig. 2: Team formation

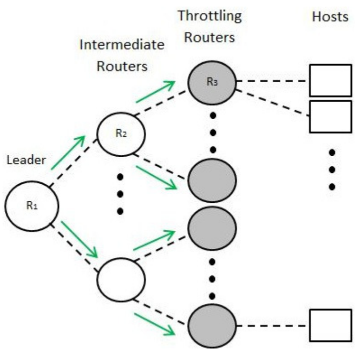

Fig. 3: Team structure

evant information necessary to complete a cooperative task [10]. The domain does not permit reliance on a complex communication scheme. We propose a hierarchical uni-directional communication scheme. It is based on the assumption that agents are synchronous and works as follows. The victim's router signals its local load reading to the team leaders. This is depicted in Figure 2 by the uni-directional arrows. The team leaders signal both their local load reading and the received reading from the victim's router to their intermediate routers. Similarly, the intermediate routers signal their local load reading and the two received readings to their throttling routers. This is depicted in Figure 3 by the uni-directional arrows. The state space of each agent consists of four features. Consider for example the router R_s and the routers R_1, R_2, R_3 in Figure 2 and Figure 3 respectively. Assuming their local instantaneous traffic rates are r_s, r_1, r_2 and r_3 respectively, the state features of router R_3 are $< r_s, r_1, r_2, r_3 >$.

4.4 Independent Team Learning via Task Decomposition

The second approach towards scalability applies task decomposition by considering that each team is independent of each other. This is achieved by assuming that instead of having a big DDoS problem at the victim, there are several smaller DDoS problems where the hypothetical victims are the team leaders[3]. The hypothetical upper boundary of each leader depends on its team size. Assuming a defence system of homogeneous teams, the hypothetical upper boundary for each team leader is given by $U_s/\#teams$ and the reward function of each reinforcement learning agent is shown in Algorithm 2. Task decomposition has also been demonstrated to be beneficial in other multiagent domains [11].

Hierarchical communication is shown to be beneficial (as we demonstrate in section 5), therefore we decided to keep this functionality. Since teams are independent and do not coordinate, the victim's router does not need to signal the team leaders. Therefore, the state space of each learning agent consists of

[3] An ISP backbone or core router, like a team leader or an intermediate router, is able to handle large amounts of traffic therefore it is unlikely to become a victim itself.

```
if $loadRouter_{leader} > (U_s/\#teams)$
then
    // Punishment
    $r = -1$
else
    // Reward in [0,1]
    $r = legitimateLoad_{leader}/legitimateLoad_{team}$
end
```

Algorithm 2: Team Reward Function

three features. Consider for example Figure 3. The state features of router R_3 are $< r_1, r_2, r_3 >$, which represent the instantaneous local loads of routers R_1, R_2 and R_3 respectively.

4.5 Coordinated Team Learning via Reward Shaping

Our final approach involves coordination between the teams of agents. The goal of this approach is to combine the advantages of the hierarchical communication and the independent team learning approaches described earlier. This is achieved by shaping the reward function. Coordinated team learning is very similar to the hierarchical communication approach described in section 4.3, with the only difference being the reward function of the reinforcement learning agents. The shaped reward function allows a team's load to exceed its hypothetical upper boundary as long as the victim's router load remains below the global upper boundary. This is shown in Algorithm 3.

```
if $(loadRouter_{leader} > (U_s/\#teams))\ AND\ (loadRouter_{server} > U_s)$
then
    // Punishment
    $r = -1$
else
    // Reward in [0,1]
    $r = legitimateLoad_{leader}/legitimateLoad_{team}$
end
```

Algorithm 3: Shaped Reward Function

5 Experiments

Experiments were conducted using an abstract emulator. The small-scale experiments in [3] were carried out using the popular ns-2 network simulator [12]. However, for the scalability experiments it was essential to use the abstract emulator. We emphasise though that the small-scale experiments were repeated and re-run using the abstract emulator with very similar results.

Notice the two different phases, namely, training and evaluation. During the training of our system, we keep track of, and distinguish between legitimate and attack traffic (requirement for the reward function). However, we particularly emphasise that this is not the case during the evaluation of our system. The rationale behind this is that the defensive system can be trained in simulation, or in any other controlled environment (e.g. small-scale lab, wide-area testbed), where legitimate and DDoS packets are known a priori, and then deployed in a realistic network, where such knowledge is not available.

5.1 System Training

The network topology used for training our system consists of five homogeneous teams of agents; specifically, each team consists of two intermediate routers and six throttling routers (three for each intermediate router). Therefore the whole defence system has a total of five teams leaders, ten intermediate routers and 30 throttling agents. There are also 60 host machines, that is, 12 hosts corresponding to each team.

Our network model is based on the model used by Yau et al [2] as described in section 3. As a convention, bandwidth and traffic rates are measured in $Mbit/s$. The bottleneck link $S-R_s$ has a limited bandwidth of $U_s = 62$, which constitutes the upper boundary for the server load. The rest of the links have an infinite bandwidth. Legitimate users and attackers are evenly distributed, specifically each host is independently chosen to be a legitimate user with probability p and an attacker with probability $q = 1-p$. We have chosen p and q to be 0.6 and 0.4 respectively. Legitimate users and attackers are chosen to send UDP traffic at constant rates, randomly and uniformly drawn from the range $[0, 1]$ and $[2.5, 6]$ respectively.

Multiagent Router Throttling consists of 30 reinforcement learning agents installed on throttling routers. Our approach uses a linear decreasing ϵ-greedy exploration strategy with an initial $\epsilon = 0.3$ and the learning rate is set to $\alpha = 0.05$. We have used function approximation, specifically Tile Coding [4], for the representation of the continuous state space and have discretised the continuous action space into ten actions: $0.0, 0.1, ..., 0.9$ (recall that action 1.0 is prohibited) which correspond to $0\%, 10\%..., 90\%$ packet drop probabilities. We have also decided to use the popular SARSA [4] reinforcement learning algorithm, which was described earlier. We are only interested in immediate rewards, that is, the agents learn a reactive mapping based on the current sensations. Therefore we have set the discount factor to $\gamma = 0$, and the resulting SARSA update rule is given by the formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r - Q(s, a) \right]$$

The system is trained for 100000 episodes, with no exploration after the 80000th episode. The term episode refers to an instance of the network model. All traffic starts at time $t = 0$ and each episode lasts for 1000 time steps. We particularly emphasise that the environment is probabilistic, that is, each episode

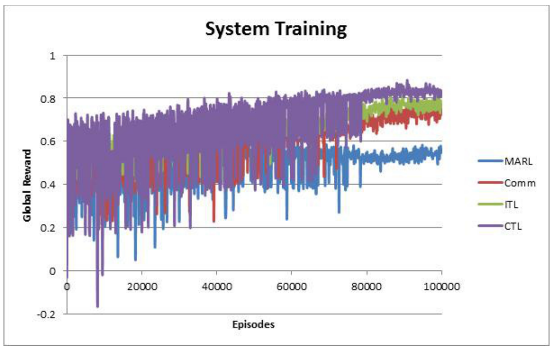

Fig. 4: System training

is most likely to be different from each other. At the start of each episode a new network instance is generated i.e. we re-choose the legitimate users, attackers and their rates according to the model. The system training attempts to cover all possible episodes i.e. instances of the network model and each agent attempts to learn a "universal" policy for all network instances. It is important to note that a "universal" policy can be different from the optimal policy of a single network instance.

We plot the last global reward of each episode, averaged over 10 repetitions; this is presented in Figure 4. MARL, Comm, ITL and CTL represent the basic approach [3] (described in 4.1), the hierarchical communication (described in section 4.3), independent team learning (described in section 4.4) and coordinated team learning (described in section 4.5) approaches respectively. It is clear that in all the approaches, the system learns and improves over time until it finally converges. It is important to note that because of the probabilistic nature of the environment, different rewards will be yielded in each episode and hence the shape of the graphs.

5.2 Evaluation Results

We evaluated our approach against the Baseline and the popular AIMD (additive increase/multiplicative decrease) Router Throttling [2] approaches [4]. Their control parameters are configured based on values and ranges recommended by their authors. As far as our approach is concerned, each reinforcement learning agent uses its policy learnt during the system training.

For evaluation we randomly sample 100 episodes (includes only effective DDoS attacks) each of a duration of 60 time steps. Legitimate traffic is started at $t = 0$ and stopped at $t = 60$. Attack traffic lasts for 50 time steps; it is started at $t = 5$ and stopped at $t = 55$. Evaluation is performed in six scenarios with

[4] The Baseline technique penalises all upstream routers equally irrespective of whether they are well behaving or not while the AIMD installs a uniform leaky bucket rate at each upstream router.

different sophistication levels. The six different patterns of attack dynamics are described below [13]. We emphasise that these patterns have not been previously seen by our system during the training period.

- **Constant-rate attack:** The maximum rate is achieved immediately when the attack is started.
- **Increasing-rate attack:** The maximum rate is achieved gradually over 25 time steps.
- **Pulse attack:** The attack rate oscillates between the maximum rate and zero. The duration of the active and inactive period is the same and represented by D. We create three different attacks namely the long, medium and short pulse attacks which use a period of $D = 5$, $D = 2$ and $D = 1$ time steps respectively. Note that a short pulse has the same duration as the time step of the reinforcement learning algorithm.
- **Group attack:** Attackers are split into two groups and each group performs simultaneously a different attack pattern. We have chosen the first group to perform a constant-rate and the second to perform a medium pulse attack.

The goal is to reduce collateral damage, that is, when legitimate traffic is punished, in this case rate limited, along with the attack traffic. Therefore, performance is measured as the percentage of legitimate traffic that reached the server. Figures 5 - 10 show the average performance over the 10 policies learnt during the system training, for the six types of attacks; error bars show the standard error around the mean. For completeness, the figures also show the percentage of the DDoS traffic[5]. Experimental results reveal the following.

The AIMD outperforms (differences are statistically significant) the Baseline approach in five out of the six scenarios. Noteworthy is the fact that in the case of short pulse attacks, the AIMD has a similar (slightly worse) performance to the Baseline approach. The basic MARL approach fails to perform well in this large-scale probabilistic domain; it performs worse than the AIMD approach in all scenarios except in the case of short pulse attacks. The proposed Comm, ITL and CTL approaches outperform the Baseline approach in all scenarios. Most importantly, the CTL outperforms the AIMD approach in all scenarios. The ITL approach also performs really well.

6 Discussion

6.1 Advantages

The Baseline and AIMD throttling approaches [2] are victim-initiated, that is, the victim determines and controls the throttle signals, and send them to the upstream routers. However, they are based on the assumption that victim remains operational during the DDoS attack, or that a helper machine is introduced to deal with the throttle signalling [14]. The first assumption is unlikely to be valid

[5] Recall that the DDoS traffic is not malicious; the problem is caused by the aggregate volume and not packet contents.

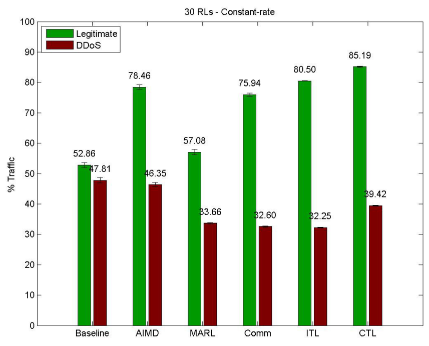

Fig. 5: Constant-rate

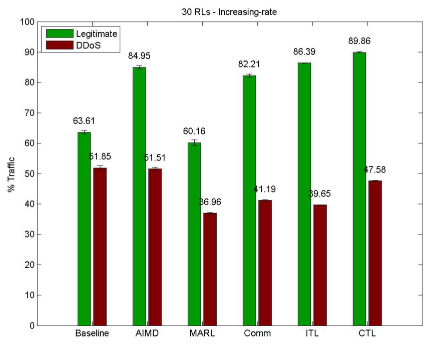

Fig. 6: Increasing-rate

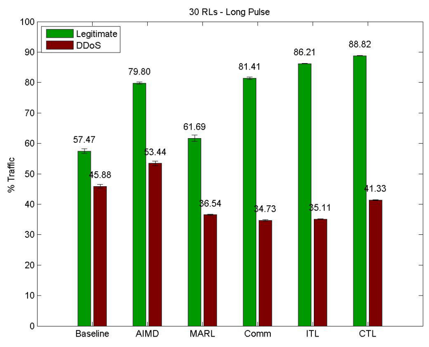

Fig. 7: Long pulse

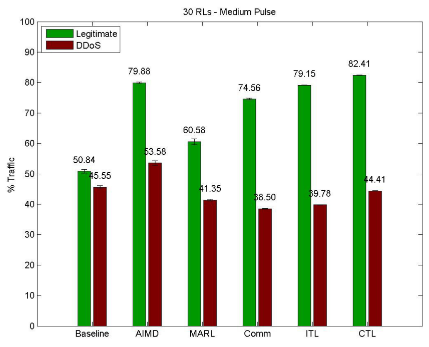

Fig. 8: Medium pulse

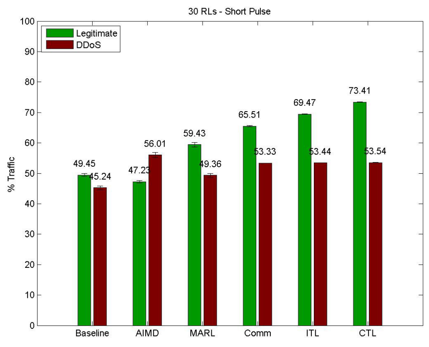

Fig. 9: Short pulse

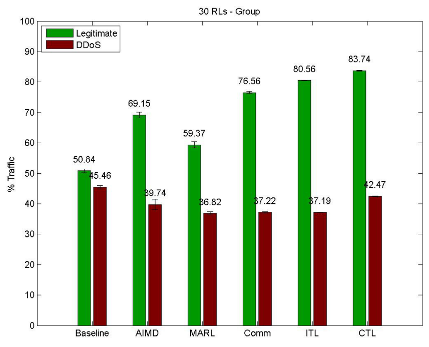

Fig. 10: Group

in a real-world scenario. As far as the second assumption is concerned, the helper machine can also become a target of the attack. In essence, the problem arises because the existing approaches are victim-initiated, that is, they have a single point of failure. One of the novel characteristics of Multiagent Router Throttling is its decentralised architecture and response to the DDoS threat. Admittedly, our approach requires communication signals from downstream (including the victim) routers, but the throttle decisions are taken by the upstream routers themselves. Our approach is therefore more secure since it eliminates the single point of failure.

The Baseline and AIMD throttling approaches [2] can suffer from stability problems because of system oscillations in order to settle the load to a desired level within the lower L_s and upper U_s boundaries. Performing throttling becomes challenging as the range $[L_s, U_s]$ becomes smaller. In contrast, the Multiagent Router Throttling approach is highly adaptable; it learns the router throttles during training phase and therefore it does not require to oscillate. That is exactly the reason the CTL approach performs very well even in the cases of highly dynamic scenarios such as the group attack.

We have demonstrated in a probabilistic domain that our approach can significantly scale-up by combining several mechanisms; these are, team formation and role specialisation, hierarchical communication, task decomposition and reward shaping. We note that our approach can be useful in other related multiagent domains.

6.2 Limitations

Our approach does not consider the case of "meek" attackers, i.e. where an attacker's sending rate is similar to the rate of a legitimate user. As already discussed, this requires that an attacker compromises and recruits a very high number of host machines. In the case of "meek" attackers, our system cannot differentiate between legitimate and attack traffic by just taking into account local router loads. Effectively tackling this problem would require traffic differentiation and to apply selective throttling on a per flow (or packet) basis. Therefore, it would require the enrichment of the state space of an agent, that is, to introduce more statistical features other than local router loads. This is a natural point to apply the experience gained and lessons learnt from the existing literature on the application of machine learning to anomaly detection.

The performance of Multiagent Router Throttling depends on the accuracy of the network model. As already discussed, during the training phase, the reward function requires to distinguish between legitimate and attack traffic. However, the network's behaviour, especially the attacker's, is unpredictable; capturing this into a model is very difficult. This would require an active data collection because the system might be found in states that have not been observed (frequently or even at all) during the training phase. Future research should investigate the potential of online learning. Online learning would impose extra communication since reward signals need to be communicated to the throttling agents.

As a solution to this problem we introduce the following "safety" mechanism. We install a hard-wired throttling agent on the server's router (that would be on router R_s in Figure 2). The "safety" mechanism is only initiated when the aggregate arrival rate r_s is greater than the upper boundary U_s and ensures that the server load is brought down to U_s. The dropping probability of the aggregate is given by $p_{drop} = 1 - U_s/r_s$ [9]. The experiments described in this paper include this "safety" mechanism. We should point out that this "safety" mechanism does not intend to replace the online learning functionality. Even with the addition of online learning, the theoretical convergence guarantees obtained in single-agent reinforcement learning does not hold for cases of multiagent reinforcement learning. In simpler words, the addition of online learning would initiate the "safety" mechanism less often.

References

1. C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2004.
2. 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," in *IEEE/ACM Transactions on Networking*, 2005, pp. 29–42.
3. K. Malialis and D. Kudenko, "Multiagent router throttling: Decentralized coordinated response against ddos attacks," in *IAAI*, 2013.
4. R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning.* MIT Press Cambridge, MA, USA, 1998.
5. D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming.* Athena Scientific, 1996.
6. C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *AAAI/IAAI*, 1998, pp. 746–752.
7. J. Mirkovic, M. Robinson, and P. L. Reiher, "Alliance formation for ddos defense," in *NSPW*, 2003, pp. 11–18.
8. A. D. Keromytis, V. Misra, and D. Rubenstein, "Sos: secure overlay services," in *Proceedings of ACM SIGCOMM*, 2002, pp. 61–72.
9. R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *Computer Communication Review*, vol. 32, no. 3, pp. 62–73, 2002.
10. M. J. Matarić, "Using communication to reduce locality in distributed multi-agent learning," *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Learning in DAI Systems, Gerhard Weiss, ed.*, vol. 10, no. 3, pp. 357–369, Jul 1998. [Online]. Available: `http://robotics.usc.edu/publications/351/`
11. C. HolmesParker and K. Tumer, "Combining hierarchies and shaped rewards to improve coordination and scalability in multiagent systems," in *AAMAS*, 2012.
12. L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. Mccanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *Computer*, vol. 33, no. 5, pp. 59–67, 2000.
13. J. Mirkovic and P. L. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.
14. D. K. Y. Yau, F. Liang, and J. C. S. Lui, "On defending against distributed denial-of-service attacks with server-centric router throttles," Department of Computer Science, Purdue University, Tech. Rep. CSD TR #01-008, 2001.