Engineering Applications of Artificial Intelligence 41 (2015) 270–284

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Distributed response to network intrusions using multiagent reinforcement learning

Kleanthis Malialis *, Daniel Kudenko

Department of Computer Science, University of York, York YO10 5GH, UK

ARTICLE INFO

Article history:
Received 3 July 2014
Received in revised form
19 January 2015
Accepted 21 January 2015
Available online 21 February 2015

Keywords:
Reinforcement learning
Coordination and cooperation
Network security
DDoS attacks

ABSTRACT

Distributed denial of service (DDoS) attacks constitute a rapidly evolving threat in the current Internet. Multiagent Router Throttling is a novel approach to defend against DDoS attacks where multiple reinforcement learning agents are installed on a set of routers and learn to throttle or rate-limit traffic towards a victim server. It has been demonstrated to perform well against DDoS attacks in small-scale network topologies. The focus of this paper is to tackle the scalability challenge. Scalability is one of the most important aspects of a defence system since a non-scalable defence mechanism will never be considered, let alone adopted, for wide deployment by a company or organisation. In this paper we introduce Coordinated Team Learning (CTL) which is a novel design to the original Multiagent Router Throttling approach. One of the novel characteristics of our approach is that it provides a decentralised coordinated response to the DDoS problem. It incorporates several mechanisms, namely, hierarchical team-based communication, task decomposition and team rewards and its scalability is successfully demonstrated in experiments involving up to 100 reinforcement learning agents. We compare our proposed approach against a baseline and a popular state-of-the-art router throttling technique from the network security literature and we show that our approach significantly outperforms both of them in a series of scenarios with increasingly sophisticated attack dynamics. Furthermore, we show that our approach is more resilient and adaptable than the existing throttling approaches.

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 (Douligeris and Mitrokotsa, 2004). A DDoS attack is a highly coordinated attack where the attacker 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 or customers.

The Arbor Network's worldwide security survey (Anstee et al., 2013) conducted among more than 290 companies and organisations reveals that 50% of the participants see 1–10 DDoS attacks per month, while 12% experience more than 100. The average size of a DDoS attack is 1.77 Gbps while larger incidents are observed in the range 2–10 Gbps (Kerner, 2013). 62% of the attacks are currently less than 1 Gbps. The average cost to defend against a DDoS attack was $2.5 million. Beyond the financial loss caused by DDoS attacks, victims also suffer from loss to their reputation which results in customer or user dissatisfaction and loss of trust.

To tackle the distributed nature of these attacks, a distributed and coordinated defence mechanism is required, where many defensive nodes, across different locations cooperate in order to stop or reduce the flood. Yau et al. (2005) is a popular approach to defend against DDoS attacks, where the victim server sends throttle signals to a set of upstream routers to rate-limit traffic towards it. Similar techniques to throttling are implemented by network operators (Douligeris and Mitrokotsa, 2004).

Malialis and Kudenko (2013) is a novel throttling approach where multiple reinforcement learning agents are installed on the set of upstream routers and learn to throttle or rate-limit 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" (Sutton and Barto, 1998) when scaling-up. The main focus of this paper is scalability and we propose a novel design that resolves this problem. Our contributions in this paper are the following.

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

* Corresponding author.
E-mail addresses: malialis@cs.york.ac.uk (K. Malialis), kudenko@cs.york.ac.uk (D. Kudenko).

http://dx.doi.org/10.1016/j.engappai.2015.01.013
0952-1976/

investigates the applicability of multiagent systems and machine learning to intrusion *response*. In this paper we are interested in distributed rate-limiting approaches to defend against DDoS attacks.

We introduce the Coordinated Team Learning (CTL) approach, which is a novel approach to the original Multiagent Router Throttling. The proposed approach combines several mechanisms, namely, hierarchical team-based communication, task decomposition and team rewards and its scalability is demonstrated in experiments involving up to 100 reinforcement learning agents. It is also empirically demonstrated that the performance of our approach remains unaffected by the addition of new teams of learning agents in the system. The CTL approach can be useful in other related multiagent domains for example congestion problems such as air traffic management and traffic light control.

One of the novel characteristics of our approach is its decentralised architecture and response to the DDoS threat. We compare our approach against a baseline and a popular state-of-the-art throttling technique from the network security literature (Yau et al., 2005). These approaches provide a distributed response but are victim-initiated, that is, the throttle signals are centrally generated from the victim server. Our proposed approach is more resilient since it does not have a single point of control.

Lastly, our proposed approach provides an automated and effective response against the highly complex and multi-dimensional DDoS threat. We evaluate our approach in a series of scenarios with increasingly sophisticated attack dynamics and show that the CTL approach outperforms both the baseline and the popular throttling techniques. Furthermore, the network environment is highly dynamic and our approach is highly responsive to the attackers' dynamics thus providing flexible behaviours over frequent environmental changes.

The organisation of the paper is as follows. Section 2 presents the necessary background material on reinforcement learning and DDoS attacks. We discuss the related work in Section 3 focussing on intrusion response and distributed rate-limiting mechanisms. Our proposed Multiagent Router Throttling approach and its design details are described in Section 4. The experimental setup is provided in Section 5 and experiments involving offline and online learning are presented in Sections 6 and 7 respectively. We conclude in Section 8 where we discuss the advantages and deployments issues of our proposed approach, and present directions for future work.

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 (Sutton and Barto, 1998). The feedback received is used to improve the agent's actions. The problem of solving a reinforcement learning task is to find a policy (i.e. a mapping from states to actions) which maximises the accumulated reward.

When the environment dynamics (such as the reward function) are available, this task can be solved using dynamic programming (Sutton and Barto, 1998). 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 the 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 (Sutton and Barto, 1998). After each real transition, $(s,a)\rightarrow(s',r)$, in the environment, it updates state-action values by the formula:

$$Q(s,a)\leftarrow Q(s,a)+\alpha[r+\gamma Q(s',a')-Q(s,a)] \quad (1)$$

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 (Sutton and Barto, 1998).

Applications of reinforcement learning to multiagent systems typically take one of two approaches; multiple individual learners or joint action learners (Claus and Boutilier, 1998). The former is the deployment of multiple agents each using a single-agent reinforcement learning algorithm. The latter is a group of multiagent specific algorithms designed to consider the existence of other agents; in this setting an agent observes the actions of the other agents or each agent communicates its action to the others.

Multiple individual learners 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 added to the system. Therefore, as we are interested in scalability and minimal communication between agents, this work focusses on multiple individual learners and not joint action learners.

2.2. Distributed denial of service (DDoS) attacks

To reduce the impact of scalability and deal with large and continuous state and action spaces, function approximation is used to reduce an agent's exploration (Sutton and Barto, 1998). Tile coding is one of the most common techniques which partitions the state space into a number of tilings and tiles where state feature values are grouped into.

A DDoS attack is a highly coordinated attack; the strategy behind it is described by the agent–handler model (Douligeris and Mitrokotsa, 2004) as shown in Fig. 1. The model consists of four elements, the attacker, handlers, agents and victim. The handler (or master) and the agent (or slave or zombie or daemons) are hosts compromised by the attacker, which constitute the botnet. Specifically, the attacker installs a malicious software called Trojan on vulnerable hosts to compromise them, thus being able to communicate with and control them. The attacker communicates with the handlers, which in turn control the agents in order to launch a DDoS attack.

The basic agent–handler model can be extended by removing the handlers layer and allowing communication between the attacker and agents via Internet Relay Chat (IRC) channels. A more recent extension occurs at the architectural level, where the centralised control is replaced by a peer-to-peer architecture.

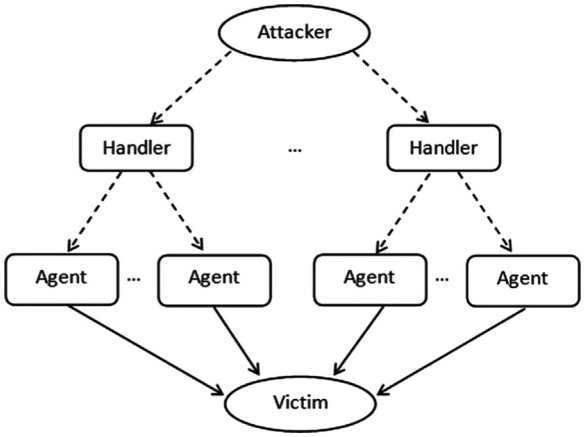

Fig. 1. The agent–handler model.

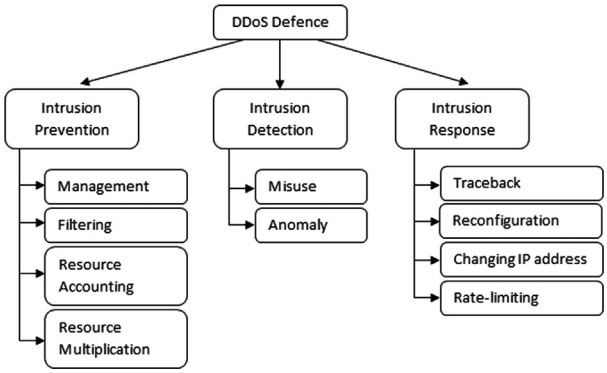

Fig. 2. DDoS defence taxonomy.

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. These extensions make the DDoS problem orders of magnitude harder to tackle because they offer limited botnet exposure, a high degree of anonymity and they provide robust connectivity. The DDoS threat is challenging for many reasons (Mirkovic and Reiher, 2004), including the following:

- *Distributed traffic*: The traffic flows originate from agent machines spread all over the Internet, in which they all aggregate at the victim.
- *Large volume*: The large volume of the aggregated traffic is unlikely to be stopped by a single defence point near the victim.
- *Large number of agents*: The number of compromised agent machines is large, thus making an automated response a necessary requirement.
- *Similarity to legitimate packets*: DDoS packets appear to be similar to legitimate ones, since the victim damage is caused by the total volume and not packet contents. A defence system cannot make an accurate decision based on a packet-by-packet basis. It requires to keep some statistical data in order to correlate packets and detect anomalies, for example, "all traffic directed towards a specific destination address".
- *Difficult to traceback*: It is difficult to discover even the agent machines, let alone the actual attackers, firstly because of the agent–handler model's architecture, and secondly because of IP spoofing.

It is evident that to combat the distributed nature of these attacks, a distributed and coordinated defence mechanism is necessary where many defensive nodes, across different locations cooperate in order to stop or reduce the flood.

2.3. DDoS defence taxonomy

In Fig. 2 we present a taxonomy of DDoS defence approaches based on Mirkovic's and Reiher's (Mirkovic and Reiher, 2004) taxonomy. Defence mechanisms are classified into three high-level categories, namely, intrusion prevention, intrusion detection and intrusion response.

Preventive mechanisms attempt to eliminate the possibility of an attack happening, or help the victim tolerate the attack without affecting its legitimate users. Management mechanisms are about keeping your system's state in such a way that the possibility of being compromised (and thus taking part in an attack) or becoming a victim is minimised. Filtering mechanisms drop network packets according to specific rules or criteria. Resource accounting mechanisms regulate a user's access to resources according to his privileges or behaviour. Resource multiplication mechanisms provide a very large amount of resources to enable the victim to tolerate the attack.

Preventive mechanisms are important and essential, but they are not perfect. Furthermore, "it's often cheaper to prevent some of the attacks and detect the rest than it is to try to prevent everything" (Anderson, 2008). Intrusion detection monitors the log files and network traffic and triggers an intrusion alarm if a suspicious event is detected. Misuse (or signature-based) detection aims at identifying already known attacks by monitoring the traffic for signatures i.e. known characteristics of attacks. The disadvantage of this approach is that it cannot uncover novel or mutated intrusions. Anomaly detection aims at uncovering novel or mutated attacks by attempting to define the normal network behaviour; if an activity deviates from the normal profile, it is marked as intrusive. The disadvantage of this approach is that it usually suffers from a high rate of false positives and negatives. Anomaly detection typically makes use of statistical or machine learning techniques.

Response mechanisms aim at mitigating the DDoS impact on the victim, while keeping collateral damage levels to a minimum. Collateral damage occurs when legitimate traffic is punished along with the attack traffic. Traceback mechanisms aim at identifying the agent machines responsible for the attack. Reconfiguration mechanisms alter the topology of the network in order to add more resources or isolate the attack traffic. Changing the IP address of the victim server constitutes another type of DDoS response, which is related to the concept of moving target defence. Rate limiting mechanisms drop some fraction of the suspicious network traffic. These mechanisms are typically used when the detection mechanism cannot precisely characterise the attack traffic i.e. when attack signatures cannot be derived.

The next section describes representative defence mechanisms from each category and discusses related work.

3. Defence mechanisms and related work

3.1. Intrusion prevention

Management mechanisms include keeping your system up-to-date by applying security patches, disabling unused services and disabling amplifiers[1] (Douligeris and Mitrokotsa, 2004).

Another popular mechanism is the ingress filtering (Ferguson, 2000) which monitors and blocks harmful inbound traffic to enter

[1] In an amplification attack, the attacker exploits the broadcasting feature of an amplification (or reflector) device such as a router or a server. Specifically, the attacker sends to a number of amplifiers spoofed IP packets with the source address set to the victim's. These amplifiers will reply to the victim, thus causing a DDoS flood.

a network, for example dropping a packet whose IP address does not belong to a known domain prefix. Similarly egress filtering (Brenton, 2007) monitors and blocks harmful outbound traffic, for example blocking a spoofed IP packet from leaving the network. A limitation of an ingress/egress firewall is its susceptibility to flooding or brute-force DDoS attacks.

Route-based distributed packet filtering (Park and Lee, 2001) is a technique aiming to stop spoofed IP packets based on route information. Interestingly, the approach is very effective in partial deployment; about 18% of Internet's autonomous systems. The drawback of this approach is that filters require to know the network topology, and assuming they do, they also need to update it since the topology changes over time.

A popular resource accounting mechanism is the Secure Overlay Services (SOS) architecture (Keromytis et al., 2002); an overlay network is a set of nodes which communicate with one another atop of the underlying network. SOS allows communication with a server only with confirmed users. The user's packets need to be authenticated and authorised by SOS, before they are allowed to flow through the overlay to the server. Effectively, the overlay hides the server's location and drops illegitimate traffic. A limitation is that packets are routed through a series of overlay nodes which introduces a latency which is far from minimal.

Another limitation of SOS is that only pre-authorised users can access the server. In general, providing access only to pre-authorised users has the following limitations (Iyengar et al., 2010). Firstly, it may deter clients from using the service. Secondly, it may not be feasible to create a list of all users authorised to use a service. For example, this is the case for open e-commerce websites like eBay or Amazon. Lastly, it is difficult to ensure that authorised users will indeed behave benignly.

Some resource accounting mechanisms include challenge-based or "proof-of-work" approaches such as cryptographic puzzles. In Chang Feng and Kaiser (2012), a website's URL (HTML) tag is updated to include a cryptographic puzzle. More malicious clients are presented with more difficult puzzles based on historical data and the current server load. When a client's browser finds such a protected link it runs a server provided script (using JavaScript), to provide a solution to the puzzle. Depending on the solution, the client is assigned a priority level and service is provided accordingly. A limitation of this approach is that it cannot handle brute-force DDoS attacks.

Resource accounting mechanisms also include quality of service (QoS) regulation. In Iyengar et al. (2010), a client first contacts a challenge server to obtain a puzzle. Upon successful solution it receives an initial trust token. Trust tokens encode the QoS level the client is eligible to receive from the protected server. A client's token is included in all the future requests to the server. A client that presents a valid token will be served at the priority level encoded in the token. The server updates the client's priority level based on its recent requests and the amount of server resources they have consumed. A limitation of this approach is that it cannot handle brute-force DDoS attacks.

The typical resource multiplication approach is load balancing (Mirkovic and Reiher, 2004). These mechanisms raise the bar on how many machines need to be compromised for an effective DDoS attack, but they are very expensive. So far these mechanisms have been proved sufficient for those who can afford the cost (Mirkovic and Reiher, 2004). However, there are continuing worries that gigantic DDoS attacks originating from monster botnets will occur in the future (Anderson, 2008).

3.2. Intrusion detection

Snort (Roesch et al., 1999) is a well-known system which uses misuse detection. Anomaly detection typically makes use of statistical or machine learning techniques.

Servin and Kudenko (2008) propose a distributed RL approach in order to detect flooding DDoS attacks. Agents are organised in a hierarchy and a sender agent (lower hierarchical level) learns semantic-less communication signals which represent the "summary" of its local state observations. The recipient agent (higher hierarchical level) also needs to learn how to interpret these semantic-less signals. Finally, the root of the hierarchy learns whether or not an intrusion alarm should be triggered.

Xu et al. (2007) state that information sharing such as combining local information or decisions among agents can improve detection accuracy but it can be costly. The authors use distributed reinforcement learning in order to optimise communication costs of information exchange among agents, by learning when to broadcast information to other agents. Other examples include Jung et al. (2002), Hussain et al (2003), Mirkovic et al. (2002), and Marquardt et al. (2013).

3.3. Intrusion response

Probabilistic Packet Marking (PPM) (Savage et al., 2000) is a popular traceback technique, where upstream routers probabilistically mark IP packets so the attack route can be reconstructed. Traceback can be very expensive though and it is virtually impossible to trace due to the large number of attack paths (Papadopoulos et al., 2003). It is also questionable whether it is useful to spend large amounts of resources to traceback and identify individual zombies when the actual attackers continue to operate unnoticed and uninhibited (Papadopoulos et al., 2003).

Replication techniques constitute a sub-category of reconfiguration mechanisms. XenoService (Yan et al., 2000) is such a mechanism where a number of ISPs install Xenoservers that offer web hosting services. A website is monitored and if reduction in the quality of service is experienced due to a surge in demand, the website is replicated to other Xenoservers. The attacker needs to attack all replication points in order to deny access to the legitimate users.

Replication techniques have the following limitations (Keromytis et al., 2002). They are not suitable in cases where information needs to be frequently updated (especially during an attack) or it is dynamic by nature (e.g. live audio or video stream). In case of sensitive information security being a major concern, engineering a solution that replicates sensitive information without any "leaks" is challenging.

The typical moving target approach is that the name of a service is stable but its IP address is not (Shue et al., 2012). Therefore, any traffic that knows the currently valid IP address is granted access to the server while the rest is discarded. A client is required to periodically query the DNS server to obtain the server's IP address. One of the criticisms of this approach is that advanced DDoS attack tools can include a DNS tracing function to track the IP address changes (Geng and Whinston, 2000).

Mittal et al. (2012) introduce Mirage which tackles this issue. Mirage requires all clients that wish to access a server to first solve a cryptographic puzzle using JavaScript. Specifically, when a client makes a DNS lookup request, it is directed to a puzzle server to retrieve a puzzle, for which the solution is the server's current IP address. Upstream routers filter traffic that is destined to all inactive IP destinations.

Mirage however does not come without its limitations. Mirage needs to ensure that the distribution of puzzles is not subject to DDoS attacks. To address this issue, replication techniques (described earlier) have been suggested (Mittal et al., 2012). Furthermore, the acquisition of valid IP addresses depends on the computational power of the DDoS machines (botnet). Therefore, distributed rate-limiting on upstream routers for the active IP destinations is required

to further reduce the power of the attackers and to ensure a fair treatment to the legitimate users.

Moreover, the new IP address is assigned for all client sessions on a relatively long time period. These approaches in general require the system administrators to make a series of changes in the DNS entries and routing table entries so that traffic is directed to the new IP address (Geng and Whinston, 2000). The suggested time period for Mirage is 5 min (Mittal et al., 2012).

IP Fast Hopping (Krylov and Kravtsov, 2015; Krylov and Ponomarev, 2012) improves over Mirage and allows the server's IP address to change much faster. To access a protected server, the client must first be tested for legitimacy on an authorisation server. If successful, the client is re-directed to a special server called IP Hopper Manager, which is responsible for enhanced secured sessions. An enhanced secure session is a communication session between a client and a server which is protected by IP Fast Hopping. This allows the IP address of the server to change every millisecond for a client–server communication session.

However, IP Fast Hopping has its risks as well. On contrary to Mirage, it requires the clients to install a special software (IP Hopper Core). Similar to Mirage, the system needs to ensure that the authorisation server and the IP Hopper Manager are not subject to DDoS attacks. Furthermore, like Mirage, the security of the approach depends on the space of IP addresses. If the IP pool is small and/or the botnet is large, the DDoS machines can start bombarding each IP address. In such a case, the victim's ISP (or several ISPs) can rate-limit or filter traffic to mitigate the effectiveness of the DDoS attack.

Our proposed multiagent reinforcement learning-based solution, which learns distributed rate-limits, can work in cooperation with IP Fast Hopping and related approaches to complement each other.

3.3.1. Distributed rate-limiting

This section describes work closely related to ours, focussing on distributed, cooperative rate-limiting mechanisms. These mechanisms are typically used when the detection mechanism cannot precisely characterise the attack traffic i.e. when attack signatures cannot be derived.

One of the first and most influential work in the field is the Aggregate-based Congestion Control (ACC) and Pushback mechanisms by Mahajan et al. (2002). The authors view the DDoS attacks as a router congestion problem. 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 because hosts can spoof traffic, disobey congestion signals, etc.). A local ACC agent is installed on the victim's router which monitors the drop history. If the drop history deviates from the normal,[2] the local ACC reduces the throughput of the aggregate by calculating and setting a rate-limit.

Pushback is a cooperative mechanism. The local ACC can optionally request from adjacent upstream ACC routers to rate limit the aggregate according to a max–min fashion, a form of equal-share fairness, where bandwidth allocation is equally divided among all adjacent upstream routers. An example of a max–min fairness allocation is the following. Consider three links with arrival rates of 2, 5 and 12 Mbit/s respectively, and the desired rate is 10 Mbit/s. The max–min fair limits sent to each link would be 2, 4 and 4 Mbit/s respectively. Rate limiting of the aggregate recursively propagates upstream towards its sources in a hierarchical fashion.

[2] The authors distinguish between typical congestion levels (e.g. observed during peak times) and unusual or serious congestion levels caused by DDoS attacks (although serious congestion can occur due to other reasons as well e.g. a fiber cut).

The major limitation of Pushback is that it causes collateral damage, that is, when legitimate traffic is rate limited along with the attack traffic. This is because the resource sharing starts at the congested point, where the traffic is highly aggregated and contains a lot of legitimate traffic within it.

Another popular work is the Router Throttling mechanism by Yau et al. (2005). The authors view the DDoS attacks as a resource management problem and they adopt a server-initiated approach. According to Douligeris and Mitrokotsa (2004), similar techniques to throttling are used by network operators. Router Throttling is described as follows. When a server operates below an upper boundary U_s, it needs no protection (this includes cases of weak or ineffective DDoS attacks). When the server experiences heavy load, it requests from upstream routers to install a throttle on the aggregate. In case the server load is still over the upper boundary U_s, the server asks from upstream routers to increase the throttle. If the server load drops below a lower boundary L_s, the server asks the upstream routers to relax the throttle. The goal is to keep the server load within the boundaries $[L_s, U_s]$ during a DDoS attack. Router Throttling, unlike Pushback, is more of an end-to-end approach initiated by the server and therefore collateral damage is significantly reduced.

Yau et al. (2005) present the Baseline throttling approach in which all upstream routers throttle traffic towards the server, by forwarding only a fraction of it. This approach penalises all upstream routers equally, irrespective of whether they are well behaving or not. The authors then propose the AIMD (additive-increase/multiplicative-decrease) throttling algorithm, which installs a uniform leaky bucket rate at each upstream router that achieves the level-k max–min fairness among the routers $R(k)$.

4. Multiagent router throttling

4.1. Network model and assumptions

The network model is similar to the one used by Yau et al. (2005). 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 they can spoof traffic, disobey congestion signals, etc. An internal node represents a router, which forwards or drops traffic received from its connected hosts or peer routers. The set of routers is denoted by R, and they are assumed to be trusted, i.e. not to be compromised. This assumption is realistic 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 (Keromytis et al., 2002). 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 Fig. 3. 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, $r_a \gg r_l$ (dropping traffic based on source addresses can be harmful because, as mentioned, hosts cannot be trusted). 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 (Yau et al., 2005). 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

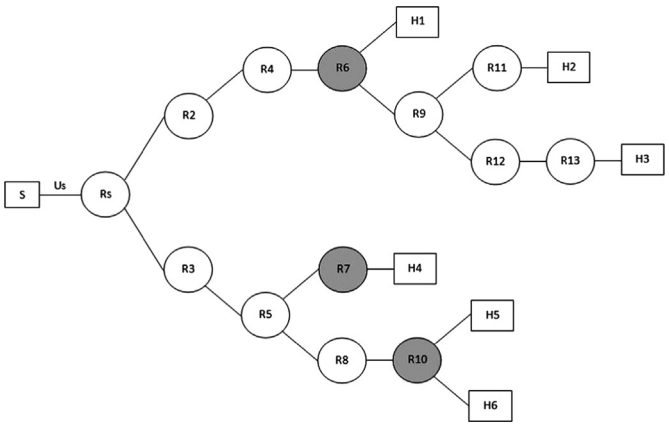

Fig. 3. Network topology showing defensive routers.

the upper boundary i.e. $r_l \ll U_s$, where U_s can be determined by observing how users normally access the server.

4.2. Basic design (MARL)

The basic MARL design is based on Malialis and Kudenko (2013). The selection method is similar to the one used by Yau et al. (2005). 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.[3] Consider for example the network topology shown in Fig. 3. Learning agents are installed on the set $R(5)$, which consists of routers *R6, R7* and *R10*. Router *R6* is included in the set $R(5)$, although it is only 4 hops away from the server, because it is directly attached to the host *H1*.

Recall that the aggregate is defined as the traffic that is directed towards a specific destination address i.e. the victim (Mahajan et al., 2002). In the basic design, each agent's state space consists of a single state feature, which is its aggregate *load*. The aggregate load is defined as the aggregate traffic arrived at the router over the last T seconds, which is called the monitoring window. The monitoring window should be set to be about the maximum round trip time between the server S and the router in $R(k)$ Yau et al. (2005). The time step of the learning algorithm is set to be the same as the monitoring window size.

Each router applies throttling via probabilistic packet 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 not included in the action space of any of the routers. The reason is 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 server access to its legitimate users (Mahajan et al., 2002).

Global reward: With a global or system reward function each agent 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. We consider that legitimate users are all of equal importance, therefore there is no prioritisation between them. The global reward function is shown in Algorithm 1.

Algorithm 1. Global (G) reward function.

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

At this point we discuss the availability of L in the different cases of offline and online learning. In the case of offline learning, we can keep track of and identify the legitimate traffic. This is because the defensive system can be trained in simulation, or in any other controlled environment (e.g. wide-area testbed, small-scale lab) where legitimate traffic is known a priori, and then deployed in a realistic network where such information is not available.

Let us now consider the case of online learning i.e. when the system is trained directly in a realistic network. If the detection mechanism is accurate enough to derive attack signatures (i.e. known characteristics) then the problem can be simply solved by filtering the attack traffic. However, we are interested in cases where the detection mechanism cannot precisely characterise the attack traffic i.e. when attack signatures cannot be derived. Inevitably, in such cases L can only be estimated.

There are different ways to measure legitimate traffic, for example by observing the behaviour or habits of customers and regular users of the victim's services and detect deviations. Another way is by observing the IP addresses that have been seen before; work conducted by Jung et al. (2002) reveals that during a DDoS attack to a website, most of the sent requests were generated by IP addresses that did not appear before. For example for the CodeRed worm (Moore et al., 2002) only 0.6–14% of the IP addresses appeared before.[4] Another way is by observing whether IP packets have appeared before or after the DDoS impact, as the latter suggests that they are likely illegitimate (Hussain et al, 2003).

4.3. Hierarchical communication (Comm)

As it is later demonstrated the basic MARL approach suffers from the "curse of dimensionality" and fails to scale-up in large scenarios. To scale-up we propose a number of mechanisms based on the divide-and-conquer paradigm. Generally, the divide-and-conquer paradigm breaks down a large problem into a number of sub-problems which are easier to be solved. The individual solutions to the sub-problems are then combined to provide a solution to the original problem.

[3] Collaboration between different administrative domains is desirable but the motivation for deployment is low.

[4] The CodeRed worm appeared in June 2001. The first phase of its operation was to infect machines and turn them into zombies. More than 359,000 machines were infected in just 14 h (Moore et al., 2002). The second phase was to launch a DDoS attack against the White House's website.

The first step towards scalability is to form teams of agents as shown in Fig. 4. Dashed lines do not necessarily represent nodes with a direct connection. Teams can either be homogeneous or heterogeneous. The structure of the team is shown in Fig. 5. 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 e.g. an ISP. In case of collaborative domains, each team can belong to a different administrative domain. Note that only the throttling routers are reinforcement learning agents. The rest of the routers are non-learning agents and we will explain their role shortly. The number of teams and their type (homogeneous or heterogeneous) depends on the underlying topology and network model.

The second step towards scalability involves communication. Direct communication is defined as a purely communicative act in order to transmit information (i.e. a speech act) (Matarić, 1998). Indirect communication is concerned with the observation of other agents' behaviour and its effects on the environment. Specifically, communication tackles the partial observability problem, where distributed agents cannot sense all of the relevant information necessary to complete a cooperative task.

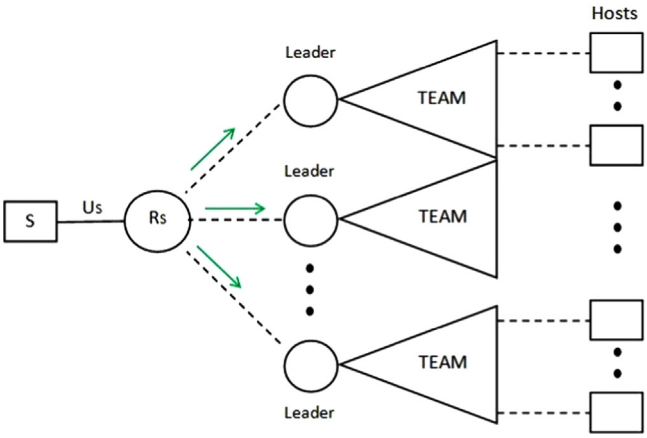

Fig. 4. Team formation.

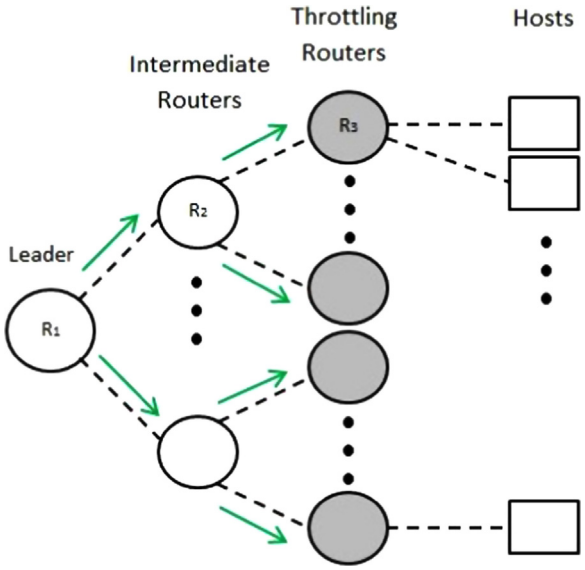

Fig. 5. Team structure.

The domain does not permit reliance on a complex communication scheme. We propose a hierarchical uni-directional communication scheme. The victim's router signals its local load reading to the team leaders. 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 Figs. 4 and 5 by the uni-directional arrows. We should note that this constitutes an indirect communication scheme because the local load readings of the signallers, are essentially the effects of the throttling agents' actions. The state space of each reinforcement learning agent now consists of four features. Consider for example the router R_s and the routers R_1, R_2, R_3 in Figs. 4 and 5 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 $\langle r_s, r_1, r_2, r_3 \rangle$. The hierarchical communication method uses the same global reward function as the basic approach which is described in Section 4.2.

4.4. Independent and coordinated team learning (ITL and CTL)

The final step towards scalability is the use of task decomposition and team rewards. For task decomposition, it is now assumed 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.[5] The hypothetical upper boundary of each leader depends on its traffic sources (i.e. the amount of host machines). Assuming a defence system of homogeneous teams, with respect to their sources, the hypothetical upper boundary for each team leader is given by U_s/teams.

Moreover, agents are now provided with rewards at the team level rather than the global level, that is, agents belonging to the same team receive the same reward or punishment. In this section we propose the independent team learning and coordinated team learning approaches. Their reward functions are as follows:

Independent team reward: An agent within a team receives a punishment of -1 if the team's load exceeds its hypothetical upper boundary. It receives a reward of $L \in [0, 1]$, where L denotes the proportion of the legitimate traffic that reached the team leader (with respect to the total legitimate of the team). The independent team reward function of each reinforcement learning agent is shown in Algorithm 2.

Coordinated team reward: This approach involves coordination between the teams of agents by allowing a team's load to exceed its hypothetical upper boundary as long as the victim's router load remains below the global upper boundary. The coordinated team reward function of each reinforcement learning agent is shown in Algorithm 3.

Algorithm 2. Independent team (IT) reward function.

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

Algorithm 3. Coordinated team (CT) reward function.

if $(loadRouter_{leader} > (U_s/teams))$ AND

[5] 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.

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

Lastly, as it is later demonstrated that the hierarchical communication functionality is beneficial to the system, therefore both of the approaches include it.

5. Experimental setup

To conduct experiments we have developed a network emulator which serves as a testbed for demonstrating the effectiveness of our proposed approach. The emulator treats the internal model of a network as a black box and mimics the observable behaviour of the network by only considering inputs and outputs of the model. It is important to note that this is adequate to demonstrate the functionality of throttling approaches.

Our network model and experimental setup are based on Yau's et al. (2005) work. 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, which constitutes the upper boundary for the server load. The rest of the links has 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 $q = 1-p$. Parameters p and q are set 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] Mbit/s respectively. We refer to an *episode*, as an instance of the network model just described.

Reinforcement learning agents are installed on throttling routers. Our approach uses a linear decreasing ϵ-greedy exploration strategy (initial ϵ values are given later for each individual experiment) and the learning rate is set to $\alpha=0.05$. We use function approximation, specifically Tile Coding (Sutton and Barto, 1998), for the representation of the continuous state space and discretise the continuous action space into ten actions: 0.0, 0.1, …, 0.9 which correspond to 0%, 10% …, 90% traffic drop probabilities (recall that action 1.0 is prohibited).

Due to the nature of the domain, the current network state has not necessarily been entirely affected by the actions taken by the agents at the previous time step. This is because the domain is highly probabilistic and exhibits unpredictable behaviour for example, at any time a DDoS attack can be initiated or stopped, more attackers can join or withdraw during an attack, attackers can alter their strategy which may be known or unknown to the network operator, legitimate users can start or quit using the victim's service, legitimate users can also alter their behaviour, routers can fail, network paths can change etc. For this reason we are only interested in immediate rewards, that is, the agents learn a reactive mapping based on the current sensations. Therefore we set the discount factor to $\gamma=0$. Moreover, we use the popular SARSA (Sutton and Barto, 1998) reinforcement learning algorithm; Q-values are initialised to zero and each agent uses the following update formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r - Q(s,a)] \quad (2)$$

For the purposes of our experiments we use tree network topologies consisting of homogeneous teams of agents. Notice that our proposed approach is not restricted to tree network topologies; the proposed defensive architecture constitutes an overlay hierarchy i.e. atop of the underlying network topology. Therefore, parent–child relationships can still be obtained even if the underlying topology is not tree-structured.

Each team of agents contains two intermediate routers and six throttling routers (i.e. six reinforcement learning agents, three for each intermediate router). There are also 12 host machines corresponding to each team. The upper boundary depends on the topology size and is set to be equal to $U_s = \text{Hosts}+2$. For example, for the network topology consisting of 2 teams the upper boundary is given by $U_s = 24+2 = 26$.

Finally, the control parameters for the Baseline and AIMD throttling techniques are configured based on values or range of values recommended by their authors.

6. Offline learning experiments

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

6.1. Performance

The first experiment of this section aims at learning a universal policy, that is, the "best" response for all possible instances of the network model using the topology consisting of 30 learning agents (5 teams). The system is trained for 100,000 episodes. 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 (described in Section 5).

For this particular experiment we use an initial $\epsilon=0.3$ and exploration is stopped after the 80,000th episode. Each episode runs for 1000 time steps. The system training attempts to learn a universal policy for all network instances.

MARL, Comm, ITL+Comm and CTL+Comm refer to the basic approach (Malialis and Kudenko, 2013) (described in Section 4.2), hierarchical communication (described in Section 4.3), independent and coordinated team learning (described in Section 4.4) approaches respectively.

We plot the global reward at the last time step of each episode, averaged over ten repetitions (i.e. over ten universal policies). Training results for the four reinforcement learning-based approaches are presented in Fig. 6. It is clear that the system learns and improves over time until it finally converges. Because of the probabilistic nature of the environment, different rewards will be yielded in each episode and hence the shape of the graph. Consider the network topology shown in Fig. 3. In the first episode for example, there may be two attackers, let us say *H1* and *H3*. In the second episode there may also be two attackers, but different ones, let us say *H2* and *H5*. The third episode may have the same attackers as the first one, but their sending rates are different.

We evaluate our approach against the Baseline and the popular AIMD router throttling (Yau et al., 2005) approaches. Each reinforcement learning agent uses its policy learnt during the system training. For evaluation we randomly sample 100 episodes 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 $t=55$. All approaches use an upper

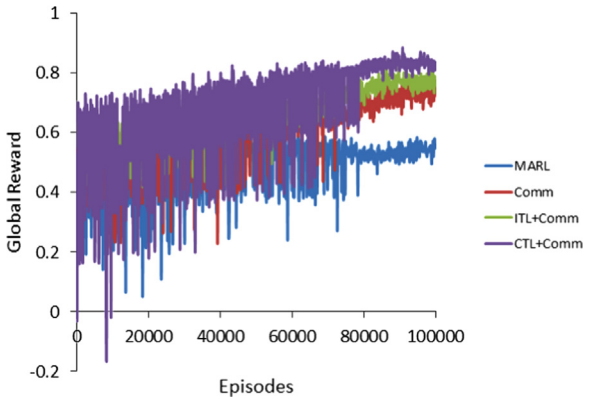

Fig. 6. Offline learning for 30 RLs.

boundary for the victim of $U_s = 62$ and the Baseline and AIMD also use a lower boundary of $L_s = 56$.

We evaluate our approach using different patterns of attack dynamics. The majority of DDoS attacks use a constant-rate mechanism where agent machines generate traffic at a steady rate (Mirkovic and Reiher, 2004). The DDoS impact is rapid but the large and continuous flood can be easily detected. It is also the most cost-effective method for the attacker. The attacker can also deploy a variable rate mechanism to delay or avoid detection and response.

Evaluation is performed using the following five different patterns of attack dynamics of different sophistication levels. We emphasise that these patterns have not been previously seen by our system during the training period. The attack dynamics are described below (Mirkovic and Reiher, 2004):

- *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 two different attacks namely the high and low pulse attacks which have a period of $D=5$ and $D=2$ time steps respectively.
- *Group attack*: Attackers are split into two groups and each group performs simultaneously a different attack pattern. We choose the first group to perform a constant-rate and the second to perform a low pulse attack.

Figs. 7–11 show the average performance over the 100 episodes for the 10 policies learnt during the system training, for the five types of attack dynamics respectively; error bars show the standard error around the mean. Performance is measured as the percentage of the legitimate traffic that reached the victim throughout an episode; the higher the value on the graph the better. For completeness, the figures also show the percentage of the DDoS traffic (note that the contents of the DDoS traffic do not cause damage but the problem is caused by the aggregated volume).

As expected, the AIMD approach outperforms the Baseline approach in all five scenarios. Furthermore, the basic MARL approach fails to perform well in this large-scale domain. The Comm approach offers great benefit to the system's performance over the basic MARL approach. Recall that the goal of an agent in offline learning is to obtain a universal policy, that is, the "best" response for all possible situations the network might be found in. Hierarchical communication helps each agent to distinguish between the different situations and therefore to learn a better policy for similar ones.

ITL+Comm and CTL+Comm further improve the system performance as they use task decomposition and team rewards. As

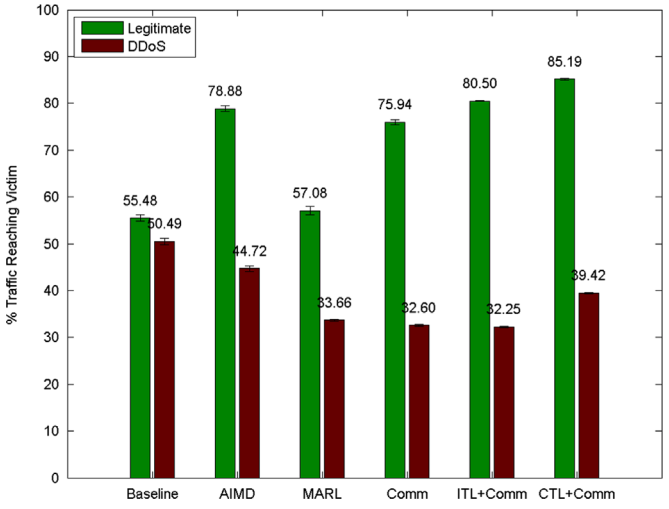

Fig. 7. Performance for constant-rate attack (30 RLs).

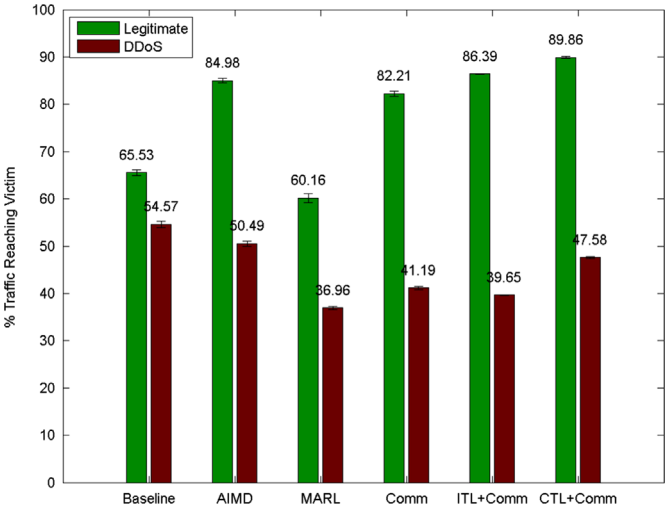

Fig. 8. Performance for increasing-rate attack (30 RLs).

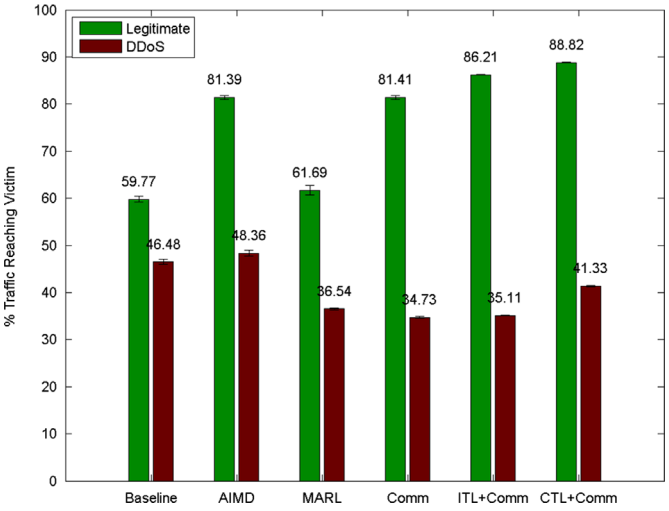

Fig. 9. Performance for high pulse attack (30 RLs).

expected, CTL+Comm performs better since it allows a team leader's load to exceed its hypothetical boundary as long as the victim router's load is below its limit.

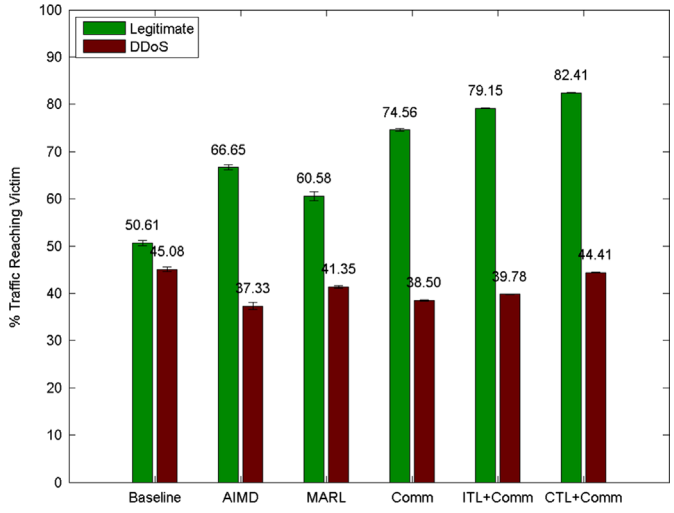

Fig. 10. Performance for low pulse attack (30 RLs).

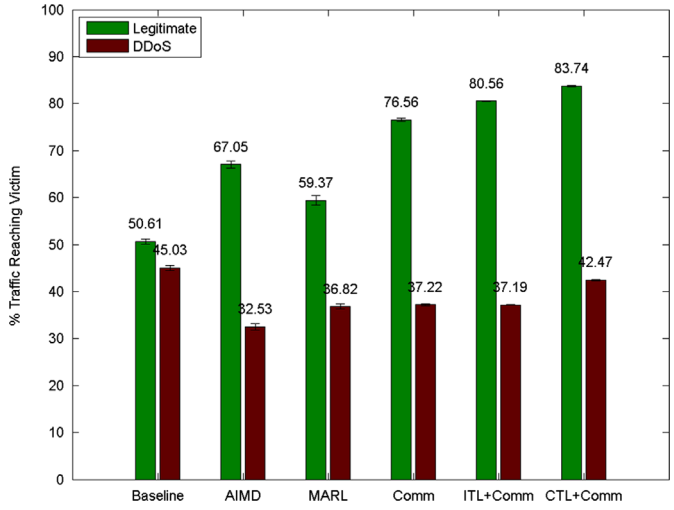

Fig. 11. Performance for group attack (30 RLs).

Most importantly, CTL+Comm outperforms the Baseline, AIMD and all other learning-based approaches in all scenarios. Specifically, it outperforms the Baseline and AIMD approaches about 24–33% and 5–17% respectively. The AIMD approach suffers the most in the highly dynamic scenarios of high pulse, low pulse and group attacks.

CTL+Comm outperforms AIMD for the following two reasons. Firstly because CTL+Comm learns a better behaviour during the training phase, and secondly because our proposed approach is more adaptable and responsive to the attackers' dynamics. These are examined in Section 6.3.

6.2. Scalability

This series of experiments aims at examining the scalability of each individual learning-based approach. Experiments are conducted for the constant-rate attack scenario for up to 30 reinforcement learning agents. As previously explained, each agent is first trained offline to obtain a universal policy that it will later use for evaluation.

Figs. 12–15 show the scalability results for the MARL, Comm, ITL+Comm and CTL+Comm approaches respectively. It is shown that the performance of MARL remains unaffected for up to 18 agents but severely declines in the case of 30 learning agents. The performance of Comm improves from 6 to 12 agents but then it starts declining when moving to the cases of 18 and 30 agents, although the performance drop is not as severe as in the case of MARL.

Lastly, it is demonstrated that the performance of both ITL+Comm and CTL+Comm remains unaffected by the addition of new teams of learning agents. This is a strong result suggesting that the two approaches are capable of scaling to large network topologies.

6.3. Aggregate load convergence

This series of experiments aims at shedding light on the previous experimental results from Section 6.1 by investigating how the aggregate load behaves during a DDoS attack. Recall from Section 3.3.1 that the original Router Throttling approach (Baseline, AIMD Yau et al., 2005) requires the aggregate load to converge within the lower and upper boundaries i.e. $r_s \in [L_s, U_s]$. Also, recall from Section 4.1 that Multiagent Router Throttling requires to bring the aggregate load below the upper boundary i.e.

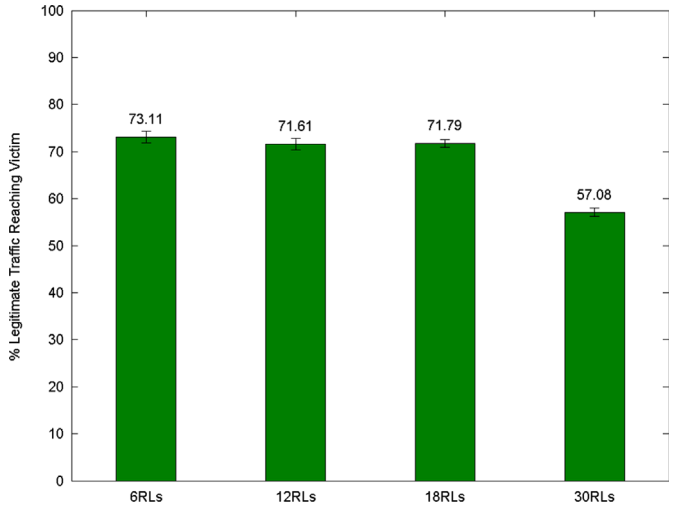

Fig. 12. Scalability of MARL.

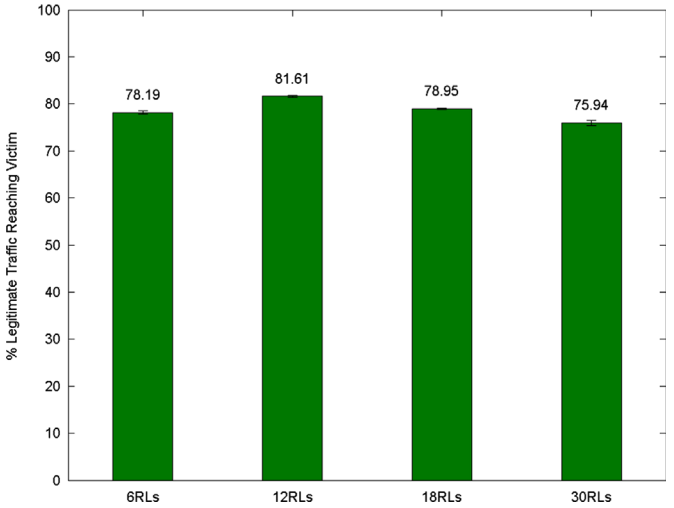

Fig. 13. Scalability of Comm.

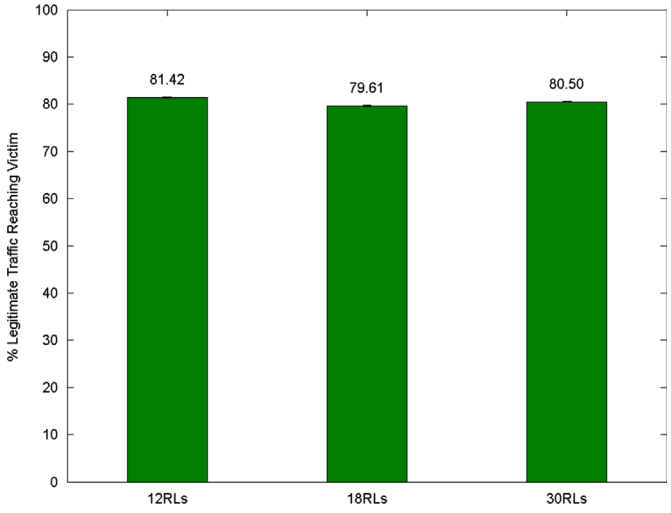

Fig. 14. Scalability of ITL+Comm.

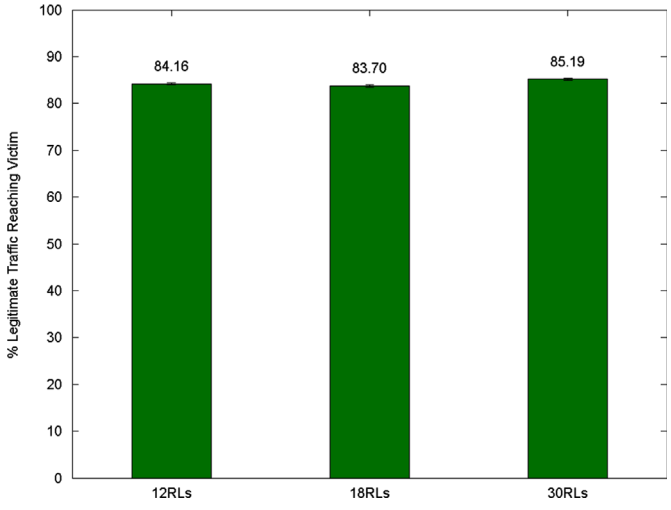

Fig. 15. Scalability of CTL+Comm.

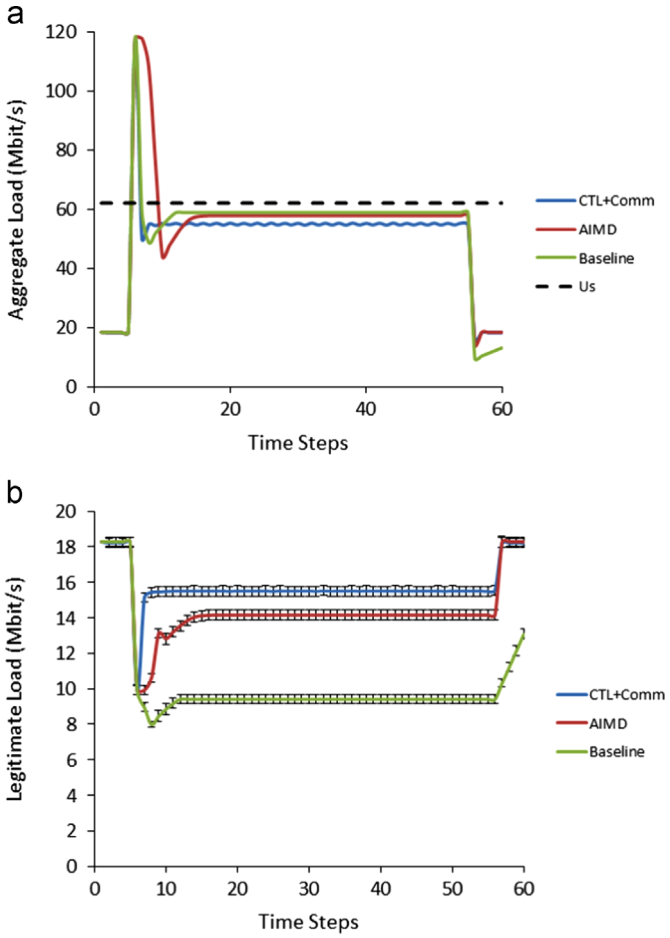

Fig. 16. Load for constant-rate attack (30 RLs). (a) Aggregate load, (b) legitimate load.

$r_s \leq U_s$. To better examine this we consider the scenarios of constant-rate, increasing-rate and high pulse attacks. At this point we are only interested in the CTL+Comm approach since it can better scale-up (Section 6.2) and significantly outperforms the other reinforcement learning-based approaches (Section 6.1).

Fig. 16a shows how the aggregate load varies for the constant-rate attack scenario when the Baseline, AIMD and CTL+Comm approaches are used. There exist 30 reinforcement learning agents that use their universal policies learnt during offline training in Section 6.1, and values are averaged over the same 100 episodes used earlier for evaluation purposes in Section 6.1. The upper boundary for the victim is $U_s = 62$; the Baseline and AIMD approaches also use a lower boundary of $L_s = 56$. All three approaches do what they are intended to do, that is, to bring down the aggregate load to acceptable levels i.e. $r_s \leq 62$ and $r_s \in [56, 62]$ for the CTL+Comm and the non-learning approaches respectively.

Fig. 16b shows how the legitimate load varies for the constant-rate attack scenario. This is averaged over the 100 episodes and error bars are plotted which show the standard error around the mean; the higher the value on the graph the better. The plots verify the previous results from Section 6.1 i.e. that CTL+Comm outperforms AIMD.

These results shed light on why this occurs. Our proposed approach outperforms the existing throttling approach for two reasons. Firstly, the system behaviour learnt during offline training is better than the "hard-wired" AIMD algorithm. Secondly, the existing non-learning approaches require a number of oscillations to bring the victim load to desirable levels; this will become more apparent as we continue to more dynamic attack scenarios. In contrast, our proposed CTL+Comm is highly responsive to the attackers' dynamics since the system learns the router throttles during offline training and as a result it does not require any system oscillations.

We repeat the same experiments with the increasing-rate DDoS scenario where similar results are obtained. Fig. 17a and b shows how the aggregate and legitimate load varies respectively for the increasing-rate attack scenario when the Baseline, AIMD and CTL+Comm approaches are used, averaged over the 100 episodes. Results show that all three approaches bring down the victim router's load to acceptable levels but our proposed approach outperforms the existing throttling approaches. As previously, this occurs because CTL+Comm has learnt a better behaviour during offline learning and also because it is highly responsive to environmental changes.

Finally, we repeat the experiments with the highly dynamic scenario of high pulse DDoS. Fig. 18a and b shows how the aggregate and legitimate load respectively varies for the high pulse attack scenario when the Baseline, AIMD and CTL+Comm approaches are used, averaged over the 100 episodes. It is evident that the AIMD approach requires considerably more time to bring the aggregate load within the desired boundaries, while the other two approaches do so much quicker. Also, our reinforcement

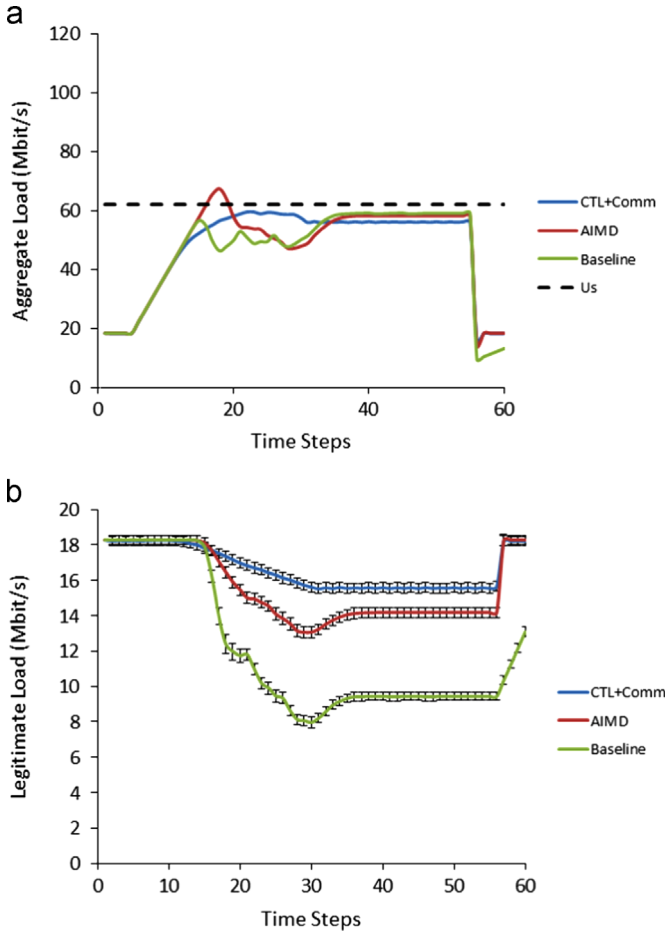

Fig. 17. Load for increasing-rate attack (30 RLs). (a) Aggregate load, (b) legitimate load.

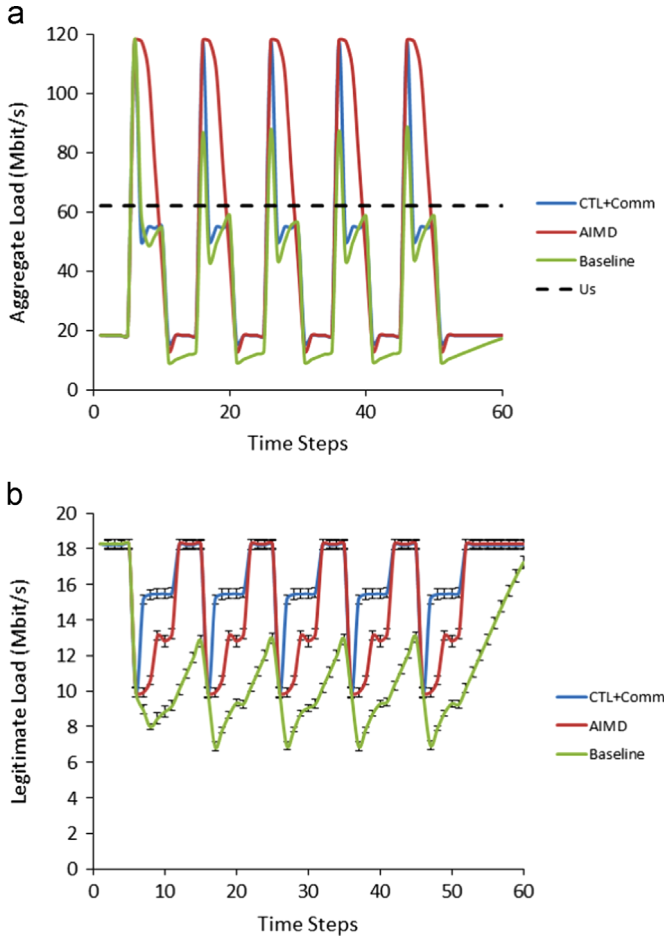

Fig. 18. Load for high pulse attack (30 RLs). (a) Aggregate load, (b) legitimate load.

learning-based approach allows more legitimate traffic than the existing approaches.

7. Online learning experiments

Finally, we present some early results on online learning experiments. On contrary to offline learning where the goal is to learn a universal policy (the "best" response for all possible situations the network might be found in), the goal of online learning is to learn the best response for the particular situation the network is currently found in.

Fig. 19a shows how the Baseline, AIMD and the reinforcement learning-based approaches compare to each other in the topology involving 30 throttling agents (5 teams) for the constant-rate attack scenario. Reinforcement learning agents use an initial $\epsilon = 0.2$. Each episode runs for 10,000 time steps. 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 (described in Section 5). The values are averaged over 500 episodes and error bars showing the standard error around the mean are plotted (hence the thickness of the plots). A higher value indicates a better performance.

Hierarchical communication is again shown to be beneficial to the system. Noteworthy is the fact that Comm requires a considerably larger amount of time (about 5000 time steps) to overcome MARL. Again, this is expected as in Comm each learning agent has four state features, as opposed to an agent using MARL that has one state feature.

It is also worth mentioning that although it performs better than MARL, this improvement is not as great as it used be in the offline learning setting. Recall that the goal of offline learning is for an agent to learn a universal policy, that is, the "best" response for all possible situations the network might be found in. Therefore, the observation makes sense since in the offline learning scenario hierarchical communication helps each agent to distinguish between the different situations and therefore to learn a better policy for similar ones. On the contrary the goal of online learning is to find the best response for the particular situation the network is currently found in.

The AIMD approach outperforms the Baseline approach as expected. Similar to the offline learning experimental results, our proposed CTL+Comm is shown to perform very well in the online learning scenario. Specifically, CTL+Comm outperforms the Baseline, AIMD and all other learning-based approaches. However, it requires about 6000 time steps to overcome AIMD.

For the same experiment, we show in Fig. 19b how the aggregate load at the victim behaves for the Baseline, AIMD and CTL+Comm approaches. All three approaches do what they are intended to do, that is, to bring down the aggregate load to acceptable levels i.e. $r_s \leq 62$ and $r_s \in [56, 62]$ for the CTL+Comm and the non-learning approaches respectively. Specifically, the AIMD approach requires 10 time steps to bring down the aggregate load to acceptable levels, while the CTL+Comm approach requires about 25 time steps.

The question now is what all these figures mean in practice about the learning speed. Since we use a network emulator and also due to the lack of an actual ISP topology we can only provide an estimate of the learning speed. Recall from Section 4.2 that the

monitoring window or time step of the throttling algorithms should be about the maximum round trip time between the victim server S and a router in $R(k)$. Assuming a round trip time of 100–200 ms means that the AIMD approach would require a few seconds to bring the aggregate load to acceptable levels (Fig. 19b) and the same time to reach its maximum value (Fig. 19a); this is consistent with experiments in Yau et al. (2005). The CTL+Comm approach would require a few more seconds to bring the aggregate load below the upper boundary (Fig. 19b) but it would require about 10–20 min to overcome the performance of AIMD (Fig. 19a). We state that the figures just mentioned constitute estimates in order to provide an indication of the learning speed.

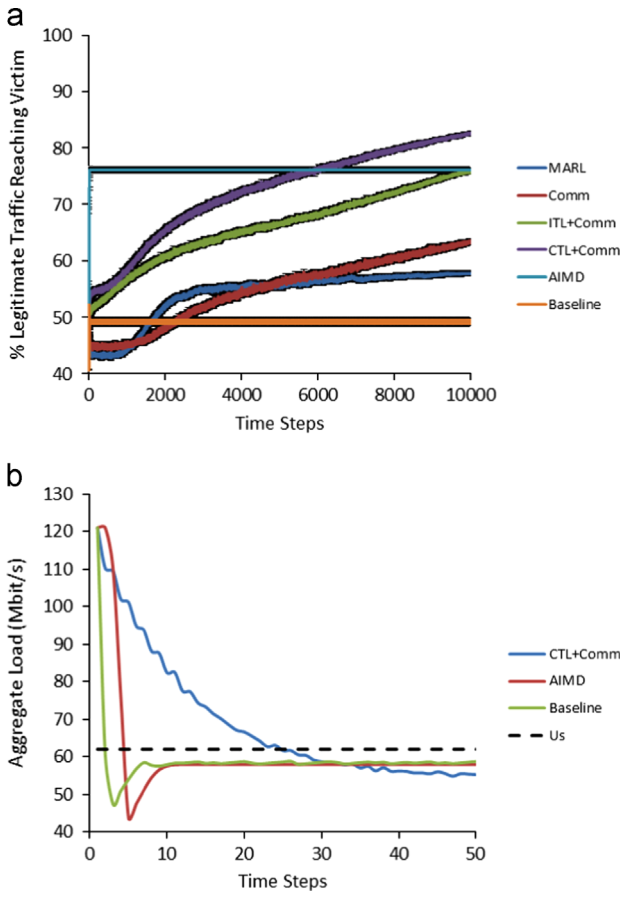

Fig. 19. Online learning for 30 RLs. (a) % Legitimate traffic reaching victim, (b) aggregate load.

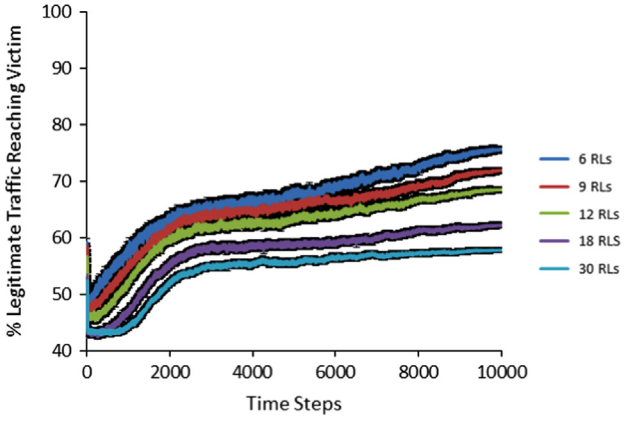

Fig. 20. Scalability of MARL.

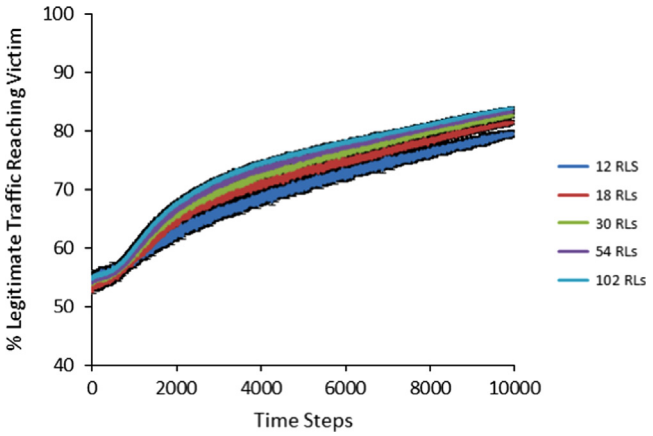

Fig. 21. Scalability of CTL+Comm.

Recall from Section 4.1 that the server is assumed to be operating normally if its load is below the upper boundary i.e. $r_s \leq U_s$. Therefore, although our proposed approach requires about 10–20 min to overcome AIMD, the victim server will be operational during a DDoS attack. Taking into consideration that 88% of DDoS attacks last for less than an hour (Anstee et al., 2013), this constitutes a promising result. However, during this period of time the victim can suffer from financial loss and customer or user dissatisfaction, and for this reason, future work will focus on improving the learning speed. The potential of online learning is further discussed in the next section.

To examine the scalability of the individual approaches in the online learning setting we conduct the following experiments. Fig. 20 shows the performance of the MARL approach when applied to topologies including up to 30 reinforcement learning agents. The values are averaged over 500 episodes and error bars showing the standard error around the mean are plotted. As expected, the basic design MARL fails to scale-up; the performance declines as the number of learning agents increases.

Fig. 21 shows the behaviour of CTL+Comm when scaling up to 102 reinforcement learning agents. The values are averaged over 500 episodes and error bars showing the standard error around the mean are plotted. We obtain a very strong result from this experiment. The approach is shown to be scalable since its performance remains unaffected by the addition of new teams of learning agents. Strictly speaking, it is observed that its performance slightly improves as the number of agents increase. We believe this occurs for two reasons. Firstly, because of the generalisation of Tile Coding (Sutton and Barto, 1998), since we keep the same number of tiles and tilings for all topologies. Secondly, it is expected that as the number of agents in the system increases, the noise created by their actions cancels out; this is known as the *the wisdom of crowds*[6] (Surowiecki, 2005).

8. Discussion and conclusion

8.1. Advantages

Resiliency: The original throttling approach (Baseline, AIMD Yau et al., 2005) is victim-initiated, that is, the victim controls and sends the throttle signals to the upstream routers. However, it is based on the assumption that either the victim remains operational during a DDoS attack or that a helper machine is introduced

[6] The term "wisdom of crowds" supports that the aggregation of information in a group often leads to better decisions, than a decision made by any individual (even an expert) in the group.

to deal with the throttle signalling (Yau et al., 2001). The first assumption can be violated 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 may arise because the existing throttling approach is victim-initiated, that is, it has a single point of control. In other words, although it offers a distributed response, it is still a centralised approach. Our proposed approach has a decentralised architecture and provides a decentralised coordinated response to the DDoS threat, thus eliminating the single point of control.

Performance: The CTL+Comm approach significantly outperforms both the Baseline and AIMD throttling approaches in a series of increasingly sophisticated attack dynamics involving constant-rate, increasing-rate, pulse attacks and a combination of the aforementioned as demonstrated in Section 6.1. The learnt decentralised behaviour performs better than the AIMD's centralised "hard-wired" behaviour allowing more legitimate traffic to reach the victim server during a DDoS attack.

Scalability: We have shown that our proposed CTL+Comm approach can significantly scale-up to large network topologies. Specifically, it has been demonstrated that its performance remains unaffected by the addition of new teams of learning agents. This is demonstrated for offline learning in topologies involving up to 30 reinforcement agents (Section 6.2) and for online learning in topologies involving up to 102 learning agents (Section 7).

The question now is how scalable does the defensive system need to be. In a study conducted by Spring et al. (2002), the authors consider 10 ISPs and estimate that the number of core (non-customer) routers in an ISP is between 11 and 1018; notice that these numbers refer to the total number of core routers and not the defensive routers. The former refers to a small ISP in India while the latter refers to a large corporate ISP in the US. It is observed that the difference is 100 times larger than the smallest networks.

To provide an estimate of the number of defensive routers consider a tree-structured network topology with a branching factor of two i.e. a binary tree. The root of the tree is the customer or victim router. A binary tree of depth four has a total number of 14 core (i.e. excluding the root) routers. Assuming the defensive routers are found in depth four (i.e. three hops away from the root) this gives a total number of 8 defensive routers. Similarly for a binary tree of depth 10 this gives a total number of 1022 core routers and 512 defensive routers. Therefore, it is expected that the number of defensive routers will be between 8 and 512. Notice that all the figures mentioned constitute estimates.

In fact, in practice it is expected that this number would be less than 512. This is because it is unlikely that an ISP would universally support the proposed functionality on all routers. Instead, it is expected that the proposed functionality will be supported on routers which see a substantial amount of network traffic. This holds true for the AIMD approach as well. As long as the majority of the DDoS traffic passes through these points our proposed approach would still be effective.

The empirical results in the paper suggest that our proposed approach is scalable enough to be potentially deployed in a medium-sized ISP network. Note that the proposed approach can be useful in other related multiagent domains for example congestion problems such as air traffic management and traffic light control.

Convergence: The original throttling approach (Baseline, AIMD Yau et al., 2005) can suffer from stability problems because of system oscillations in order to settle the aggregate 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.

Even if the system does not suffer from stability problems, it still requires a number of oscillations which can cause an increase in the time required for the aggregate load to settle within the desired range. In contrast, our proposed CTL+Comm is highly responsive to the attackers' dynamics since the system learns the router throttles during offline training and as a result it does not require any system oscillations as demonstrated in Section 6.3.

Adaptability: Unlike non-learning approaches, one of the core advantages of reinforcement learning is its capability for online learning. Let us first discuss the importance of online learning.

Firstly, training our system in an offline manner requires to have a reasonable knowledge of the network model and topology. The same applies to the Baseline and AIMD approaches which both require parameter tuning based on this knowledge. However, if these are inaccurate (i.e. they do not reflect the actual ones) or change in due course our approach would require re-training, and the existing non-learning approaches would require parameter re-tuning.

Secondly, learning a universal policy i.e. a policy that does well in every possible situation that the network might be found in, is time consuming. For example, in the experiments conducted in Section 6.1 which involved 30 reinforcement learning agents, the system was trained offline for 100,000 episodes in order for each agent to be able to learn a universal policy.

Lastly, online learning could create a robust throttling mechanism. The online learning capability will enable our system to adapt and recover from unexpected situations such as router failures.

Although the main focus of this paper is on offline learning, we have presented in Section 7 some early promising results on online learning. The proposed CTL+Comm approach is shown to be scalable, as its performance remains unaffected by the addition of new teams of learning agents. Furthermore, it is shown to outperform the AIMD approach; however, it requires a considerable amount of time to achieve that. Online learning will be the main focus of our future work and will investigate ways to improve the learning speed of our approach.

8.2. Deployment issues

Team formation: For the purposes of our experiments, we have considered a potential victim and statically formed the teams of learning agents. Future work should investigate the potential of a dynamic team formation mechanism.

"Meek" attackers: Similar to the original throttling AIMD (Yau et al., 2005) approach, our system does not take into consideration the case of "meek" attackers i.e. where an attacker's sending rate is similar to the rate of a legitimate user. As already stated, this requires that an attacker compromises and recruits a considerably higher number of host machines (zombies) in order to launch an attack of the same effect. Effectively tackling this problem would require the enrichment of the state space of an agent, that is, to introduce more statistical features other than the local router load. This is necessary because in the case of "meek" attackers, the system cannot differentiate between legitimate and attack traffic by just taking into account router loads.

Acknowledgements

The authors would like to thank Sam Devlin for all his help and the anonymous reviewers for their invaluable feedback.

References

Anderson, R., 2008. Security engineering.
Anstee, D., Cockburn, A., Sockrider, G., 2013. Worldwide infrastructure security report. Technical Report, Arbor Networks.
Brenton, C., 2007. Egress filtering faq. Technical Report.

Chang Feng, W., Kaiser, E., 2012. Systems and methods for protecting against denial of service attacks. US patent 8321955 B2.
Claus, C., Boutilier, C., 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In: AAAI/IAAI.
Douligeris, C., Mitrokotsa, A., 2004. Ddos attacks and defense mechanisms: classification and state-of-the-art. Comput. Netw. 44 (5), 643–666.
Ferguson, P., 2000. Network ingress filtering: defeating denial of service attacks which employ ip source address spoofing. Technical Report.
Geng, X., Whinston, A.B., 2000. Defeating distributed denial of service attacks. IT Prof. 2 (4), 36–42.
Hussain, A., Heidemann, J., Papadopoulos, C., 2003. A framework for classifying denial of service attacks. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.
Iyengar, A.K., Srivatsa, M., Yin, J., 2010. Protecting against denial of service attacks using trust, quality of service, personalization, and hide port messages. US patent 8250631 B2.
Jung, J., Krishnamurthy, B., Rabinovich, M., 2002. Flash crowds and denial of service attacks: characterization and implications for cdns and web sites. In: Proceedings of the 11th International Conference on World Wide Web.
Kerner, S.M., 2013. Ddos attacks: growing, but how much? URL ⟨http://www.esecurityplanet.com/network-security/ddos-attacks-growing-but-how-much.html⟩.
Keromytis, A.D., Misra, V., Rubenstein, D., 2002. Sos: secure overlay services. ACM SIGCOMM Comput. Commun. Rev. 32 (4), 61–72.
Krylov, V., Kravtsov, K., 2015. Ddos attack and interception resistance ip fast hopping based protocol. arXiv preprint arXiv:1403.7371.
Krylov, V.V., Ponomarev, D.M., 2012. Method of interaction of terminal client device with server over internet with high level of security from ddos attack and system for realising said method. RU patent 2496136 C1.
Mahajan, R., Bellovin, S.M., Floyd, S., Ioannidis, J., Paxson, V., Shenker, S., 2002. Controlling high bandwidth aggregates in the network. Comput. Commun. Rev. 32 (3), 62–73.
Malialis, K., Kudenko, D., 2013. Multiagent router throttling: Decentralized coordinated response against ddos attacks. In: Proceedings of the 25th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI).
Marquardt, D.R., Paranjape, P.A., Patil, P.S., 2013. Securing a communication protocol against attacks. US patent 8424106 B2.
Matarić, M.J., 1998. Using communication to reduce locality in distributed multiagent learning. J. Exp. Theor. Artif. Intell. (Special issue on Learning in DAI Systems, Gerhard Weiss, ed.) 10 (3), 357–369.
Mirkovic, J., Reiher, P.L., 2004. A taxonomy of ddos attack and ddos defense mechanisms. Comput. Commun. Rev. 34 (2), 39–53.
Mirkovic, J., Prier, G., Reiher, P., 2002. Attacking ddos at the source. In: Proceedings of 10th IEEE International Conference on Network Protocols, 2002.
Mittal, P., Kim, D., Hu, Y.-C., Caesar, M., 2012. Mirage: towards deployable ddos defense for web applications. arXiv preprint arXiv:1110.1060.
Moore, D., Shannon, C., et al., 2002. Code-red: a case study on the spread and victims of an internet worm. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment.
Papadopoulos, C., Hussain, A., Govindan, R., Lindell, R., Mehringer, J., 2003. Cossack: coordinated suppression of simultaneous attacks. In: DARPA Information Survivability Conference and Exposition.
Park, Kihong, Lee, Heejo, 2001. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In: Proceedings of the ACM SIGCOMM Computer Communication Review. Vol. 31(4), pp. 15 -26.
Roesch, M., et al., 1999. Snort: lightweight intrusion detection for networks. In: LISA, vol. 99, pp. 229–238.
Savage, S., Wetherall, D., Karlin, A., Anderson, T., 2000. Practical network support for ip traceback. ACM SIGCOMM Comput. Commun. Rev. 30 (4), 295–306.
Servin, A., Kudenko, D., 2008. Multi-agent reinforcement learning for intrusion detection: a case study and evaluation. In: Bergmann, R., Lindemann, G., Kirn, S., Pechoucek, M. (Eds.), Multiagent System Technologies, Lecture Notes in Computer Science, vol. 5244, Springer, Berlin, Heidelberg, pp. 159–170.
Shue, C.A., Kalafut, A.J., Allman, M., Taylor, C.R., 2012. On building inexpensive network capabilities. ACM SIGCOMM Comput. Commun. Rev. 42 (2), 72–79.
Spring, Neil, Mahajan, Ratul, Wetherall, David, 2002. Measuring ISP topologies with Rocketfuel. In: Proceedings of the ACM SIGCOMM Computer Communication Review. Vol. 32 (4) pp. 133 -145.
Surowiecki, J., 2005. The wisdom of crowds. Anchor, New York, USA.
Sutton, R.S., Barto, A.G., 1998. Introduction to Reinforcement Learning. MIT Press Cambridge, MA, USA.
Xu, X., Sun, Y., Huang, Z., 2007. Defending ddos attacks using hidden Markov models and cooperative reinforcement learning. In: Intelligence and Security Informatics, Lecture Notes in Compute Science, vol. 4430, Springer, Berlin, Heidelberg, pp. 196–207.
Yan, J., Early, S., Anderson, R., 2000. The xenoservice—a distributed defeat for distributed denial of service. In: Proceedings of ISW.
Yau, D.K.Y., Liang, F., Lui, J.C.S., 2001. On defending against distributed denial-of-service attacks with server-centric router throttles. Technical Report CSD TR 01-008, Department of Computer Science, Purdue University.
Yau, D.K.Y., Lui, J.C.S., Liang, F., Yam, Y., 2005. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. In: IEEE/ACM Transactions on Networking.