分类号：　　学校代码：10389

密　级：　　学　　号：52311050005

福建农林大学

硕士学位论文

（专业学位）

SDN 环境下基于强化学习的洪泛攻击检测与缓解研究

学习形式：全日制

学位类别：电子信息硕士专业学位

专业领域：网络与信息安全

研究方向：软件定义网络及其安全机制

学生姓名：庄涛

指导教师：舒兆港 副教授

完成时间：二〇二六年四月

Reinforcement Learning-Based Flooding Attack Detection and Mitigation in Software-Defined Networking

By

Tao Zhuang

Supervised by Prof. Zhaogang Shu

A Thesis Submitted to

Fujian Agriculture and Forestry University

in Partial Fulfillment of the Requirements

for

Professional Master's Degree in

Electronic and Information Engineering

College of Computer and Information Sciences

Fujian Agriculture and Forestry University

Fujian, P.R. China

Completion Date: (May, 2026)

Commencement Date (December, 2024)

独创性声明

本人声明，所呈交的学位（毕业）论文，是本人在指导教师的指导下独立完成的研究成果，并且是自己撰写的。尽我所知，除了文中作了标注和致谢中已作了答谢的地方外，论文中不包含其他人发表或撰写过的研究成果。与我一同对本研究做出贡献的同志，都在论文中作了明确的说明并表示了谢意，如被查有侵犯他人知识产权的行为，由本人承担应有的责任。

学位（毕业）论文作者亲笔签名：　　日期：

论文使用授权的说明

本人完全了解福建农林大学有关保留、使用学位（毕业）论文的规定，即学校有权送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

延期公开，在　　年后可适用本授权书。　□

公开，本论文属于不延期公开论文。　□

学位（毕业）论文作者亲笔签名：　　日期：

指导教师亲笔签名：　　日期：

目 录

摘 要

随着网络规模的持续扩展以及各类业务对连续性与可靠性的依赖不断加深，分布式拒绝服务（Distributed Denial of Service，DDoS）攻击已逐渐成为影响网络服务可用性的关键威胁。在 IPv4 向 IPv6 过渡的长期阶段内，双栈网络形态将持续存在，但现有研究多围绕单一协议栈展开，对双栈环境下的统一防护问题关注不足。同时，一些工作在追求检测精度的过程中，对防御策略在实际运行中的资源消耗及对网络稳定性的影响考虑不够，限制了其工程应用价值。基于上述背景，本文围绕双栈软件定义网络（Software Defined Networking，SDN）网络中的洪泛攻击防御问题，从协议层与数据层两个维度展开系统研究。

在协议层面，针对 DHCP 饥饿攻击可能导致地址池资源耗尽的问题，设计了一种基于 SDN 控制器的 DHCP 攻击防御框架。该框架由恶意 IP 定位器、恶意请求过滤器以及智能 DHCP 请求动态处理算法组成，通过对 DHCP 行为的持续分析与自适应处理，实现对异常主机的快速识别与动态防御。实验结果表明，本文提出的 DHCP 饥饿攻击防御框架在不同网络场景下均优于现有防御机制，相比现有最佳方法 DHCPGuard 的检测效果仍提升约 20%。同时，在引入防御机制后，控制器 CPU 使用率由不足 5% 增长至 5.72%，额外开销约为 14%；RAM 使用率由不足 20% 增长至 27.4%，额外开销约为 30%，表明该方法能够在较低系统开销下有效缓解 DHCP 饥饿攻击。

在数据层面，针对双栈环境中 DDoS 攻击特征复杂以及传统防御策略适应性不足的问题，提出了一种融合信息熵与深度强化学习的 DDoS 攻击检测与缓解方法。首先，通过对 IPv4 与 IPv6 流量特征进行统一处理，实现多协议流量的一致性建模；随后，构建多维信息熵体系，对网络状态变化进行实时感知与分析，从而提高异常流量检测能力。在防御阶段，将攻击检测结果作为环境反馈，引导强化学习模型动态生成防御策略，并在奖励函数中引入动作代价约束，以降低频繁策略切换对网络稳定性的影响。实验部分基于仿真平台构建双栈 SDN 网络环境，并结合 CIC-DDoS-2019 与 IDOS6 数据集进行验证。通过与 A2C、DQN 以及 DDQN 等方法的对比分析可以看出，本文方法在检测准确性、响应效率以及网络运行稳定性等方面表现出更加均衡的性能，能够在复杂攻击场景下更快恢复系统状态并维持较高的运行质量。

综上所述，本文提出的双栈 SDN 环境下洪泛攻击检测与动态防御方法能够兼顾检测准确性、响应效率以及网络运行稳定性，在复杂多协议网络场景中具有较好的泛化能力与应用潜力，可为多协议网络环境中的智能安全防护提供参考。

关键词：软件定义网络；DDoS 攻击；DHCP 饥饿攻击；信息熵；深度强化学习；双栈网络

ABSTRACT

With the continuous expansion of network scale and the increasing dependence of modern services on availability and reliability, Distributed Denial of Service (DDoS) attacks have become one of the major threats affecting network service availability. During the long-term transition from IPv4 to IPv6, dual-stack network architectures will remain prevalent for an extended period. However, most existing studies focus on single-protocol environments and pay insufficient attention to unified protection mechanisms in dual-stack scenarios. In addition, many existing methods mainly pursue high detection accuracy while neglecting resource consumption and network stability during defense processes, which limits their practical applicability. To address these issues, this thesis conducts systematic research on flooding attack detection and mitigation in dual-stack Software Defined Networking (SDN) environments from both protocol-layer and data-layer perspectives.

At the protocol layer, a DHCP starvation attack defense framework based on the SDN controller is proposed to address the problem of IP address pool exhaustion caused by DHCP starvation attacks. The framework consists of a malicious IP locator, a malicious request filter, and an intelligent DHCP request dynamic processing algorithm, which continuously analyzes DHCP behaviors and adaptively adjusts defense strategies to rapidly identify abnormal hosts and mitigate malicious requests. Experimental results show that the proposed framework achieves approximately 20% higher detection performance than DHCPGuard in different network scenarios. Meanwhile, controller CPU utilization only increases to 5.72%, while RAM utilization rises to 27.4%, demonstrating effective attack mitigation with relatively low system overhead.

At the data layer, a DDoS attack detection and mitigation method integrating information entropy and deep reinforcement learning is proposed to address the complex traffic characteristics and insufficient adaptability of traditional defense strategies in dual-stack environments. IPv4 and IPv6 traffic features are uniformly processed to achieve consistent modeling of multi-protocol traffic, while a multidimensional information entropy system is constructed to perceive network state variations in real time and improve abnormal traffic detection capability. During the defense stage, attack detection results are used as environmental feedback to guide the reinforcement learning model in dynamically generating defense strategies. In addition, action cost constraints are introduced into the reward function to reduce network instability caused by frequent policy switching. Experiments conducted on a simulated dual-stack SDN platform using the CIC-DDoS-2019 and IDOS6 datasets demonstrate that the proposed method achieves more balanced performance than A2C, DQN, and DDQN in terms of detection accuracy, response efficiency, and network stability.

In summary, the proposed flooding attack detection and dynamic mitigation methods for dual-stack SDN environments can effectively balance detection accuracy, response efficiency, and network operational stability. The proposed methods exhibit strong generalization ability and practical application potential in complex multi-protocol network scenarios, providing valuable references for intelligent security protection in future network environments.

KEY WORDS:Software Defined Networking; DDoS Attack; DHCP Starvation Attack; Information Entropy; Deep Reinforcement Learning

第一章　绪论

1.1　研究背景与研究意义

随着互联网技术的快速发展，网络规模不断扩大，云计算、物联网以及移动互联网[1]等新型应用对网络服务的连续性、可靠性与实时性提出了更高要求。然而，网络规模的扩大也使得网络安全问题日益突出，其中 DDoS 攻击已成为威胁网络基础设施稳定运行的主要安全威胁之一。攻击者通常通过控制大量受感染主机向目标服务器发送海量请求，从而耗尽网络带宽、计算资源或服务能力，最终导致正常用户无法获得网络服务。

近年来，网络威胁呈现出高频化、自动化和多源异构化的发展趋势。一方面，DDoS 攻击的规模和频次持续上升，攻击者借助僵尸网络、云基础设施和自动化工具，可以在较短时间内形成高强度流量冲击。Cloudflare 发布的 2025 年第四季度 DDoS 威胁报告显示，2025 年 DDoS 攻击总量达到 4710 万次，同比增长 121%，其中网络层 DDoS 攻击数量增长尤为明显，说明网络层洪泛攻击仍是影响网络基础设施稳定运行的重要威胁。另一方面，攻击持续时间更短、流量峰值更高，超大规模 DDoS 攻击在 2025 年持续增加，单次攻击峰值已达到 31.4Tbps，且该攻击仅持续约 35s，这使依赖人工响应或静态规则的防御机制难以及时生效。

从更广泛的网络安全态势看，网络攻击还表现出漏洞利用快速化、攻击服务产业化和人工智能辅助攻击等特征[2]。ENISA 发布的 2025 年威胁态势报告指出，漏洞利用仍是攻击者获取初始访问权限的重要方式，且部分漏洞会在披露后较短时间内被快速武器化；同时，人工智能辅助钓鱼、勒索软件服务化和网络犯罪生态专业化等趋势进一步提高了攻击活动的隐蔽性与自动化程度[3]。这些变化表明，当前网络防御不仅需要提高攻击检测准确率，还需要关注检测实时性、策略自适应性以及防御动作对网络稳定性的影响。

在传统网络架构中，网络控制与数据转发通常紧密耦合于设备内部，缺乏统一的管理机制与全局视角，这在一定程度上增加了网络安全防护的复杂性。软件定义网络[4]通过将控制平面与数据平面解耦，使控制逻辑集中于控制器，从而实现对网络资源的统一调度与灵活管理。这种架构改变为网络安全提供了更加宏观的观察视角，使管理者能够基于全局信息对网络状态进行分析，并据此制定动态防御策略。不过，这种集中化特性也使控制器成为关键节点，一旦其受到洪泛攻击等威胁，可能直接影响整个网络的运行性能，严重时甚至导致系统瘫痪[5]。

与此同时，在互联网从 IPv4 向 IPv6 逐步演进的过程中[6]，双协议栈共存已成为一种长期现象[7]。在这种环境下，IPv4 与 IPv6 在地址表示、报文结构及扩展机制等方面存在明显差异，使得网络流量呈现出更加复杂且多样的特征分布。

这种协议层面的异构性不仅提高了流量分析的难度，也对传统依赖单一协议设计的检测与防御方法提出了新的挑战。

表 1-1　当前网络威胁趋势、研究热点与本文切入点

威胁趋势	具体表现	防御要求	研究热点	本文切入点
DDoS 攻击高频化	攻击数量持续增长，网络层洪泛攻击占比高	提高实时检测与快速响应能力	DDoS 检测、流量异常识别	构建基于信息熵的 DDoS 攻击检测方法
攻击流量超大规模化	Tbps 级攻击增多，攻击持续时间短	防御策略需要自动化、低延迟	自动化防御、动态缓解	引入强化学习实现动态防御决策
攻击来源多样化	僵尸网络、云主机、IoT 设备共同参与	适应复杂异构流量环境	Botnet 检测、流量溯源	利用 SDN 全局视角分析网络状态
网络环境异构化	IPv4 与 IPv6 长期共存，流量特征差异明显	统一建模多协议流量	双栈网络安全、IPv6 入侵检测	设计 IPv4/IPv6 双栈流量统一处理机制
防御稳定性要求提高	过度防御可能影响正常业务	兼顾攻击抑制与服务质量	代价敏感防御、自适应安全	在强化学习奖励函数中引入动作代价约束

在双栈 SDN 环境中，网络系统面临的威胁具有多层次特征：既包括协议层面的资源消耗类攻击，也涉及数据平面的流量型洪泛攻击。例如，通过构造大量伪造的 DHCP 请求，攻击者可以迅速耗尽地址池资源，影响合法用户的正常接入；与此同时，大规模 DDoS 流量会占用带宽资源并增加控制器负载，从而降低整体服务质量。在这种背景下，如何在双栈 SDN 环境中实现兼顾效率与适应性的防御机制，逐渐成为一个值得关注的问题。

从另一方面看，SDN 的集中式控制结构也为攻击检测与防御提供了新的实现路径。控制器能够实时掌握全网运行状态，并根据需要向数据平面下发策略，这为安全防护提供了良好的支撑条件。通过持续采集流量特征、连接状态以及设备行为等多维信息，可以为异常流量的识别提供较为充分的数据基础。在此基础上，对相关信息进行分析处理，有助于定位潜在的恶意主机并识别其行为模式。一旦发现异常，控制器可以通过限速、优先级调整、流表更新或流量重定向等方式进行干预，从而在较短时间内对攻击作出响应并减轻其影响。得益于其全局视角与可编程特性，SDN 为解决传统网络中较难处理的安全问题提供了新的思路。

围绕 SDN 环境下的入侵检测与防御，已有研究从多个角度展开探索。常见方法包括基于信息熵的分析、统计建模，以及机器学习[8]与深度学习[9]等技术手段。此外，也有工作尝试引入区块链、联邦学习及强化学习等新兴方法，与 SDN

架构相结合，以期提升防御机制的灵活性与鲁棒性。这些研究在一定程度上拓展了 DDoS 防护的技术路径，并展示出良好的应用潜力。

需要指出的是，IPv4 向 IPv6 的过渡仍将持续较长时间，双栈共存将成为常态[10]。然而，现有研究多针对单一协议环境展开，通常分别为 IPv4 或 IPv6 设计防御机制，对双协议共存场景的统一处理关注不足。在实际应用中，这种方式往往需要并行部署多套检测系统，不仅增加了资源开销，也可能由于系统之间的交互影响整体性能。

基于上述背景，本文以双栈 SDN 网络中的洪泛攻击防御为研究对象[11]，从协议层与数据层两个角度出发，设计相互协同的防御机制，并结合信息熵方法与深度强化学习模型[12]，实现对攻击行为的实时识别与动态响应，从而在复杂网络环境中提升系统的安全性与运行稳定性。

1.2 研究现状

1.2.1 SDN 环境下协议层面洪泛攻击防御研究现状

在现有的 SDN 环境下的 DHCP 攻击检测工作中有很多检测方式，例如请求速率检测、端口安全检测以及深度数据包检测等。单一的检测方式并不能达到很好的效果，因此现有的工作中都结合了多种检测方式。Younes 等人[13]提出了一种新的 P-DHCP（Protect DHCP）方案，用于缓解 DHCP 饥饿攻击。P-DHCP 方案通过结合强认证机制和速率限制策略来抵御攻击。一方面 P-DHCP 利用密钥分发机构（KDA）以验证客户端的真实性，确保只有经过认证的客户端才能完成 DHCP 过程。另一方面通过监控和控制每个端口上的 DHCP 消息速率以检测并阻止可疑活动。Tok 等人[14]提出了一种名为 DHCPguard 的攻击防御模块。DHCPguard 利用 DHCP 嗅探记录相应的请求的信息结合限制速率检测相应的攻击，但在释放 IP 池的步骤中只是验证了服务本身的 offer 列表和 lease 列表中的 IP 是否一一对应，这样的检测只能针对于特定的只发起 request 包的攻击方式，对于其他的攻击方式并没有效果。Aldaoud 等人[15]提出了一种名为 DHCPWatcher 的多阶段检测机制。DHCPWatcher 虽然采用多阶段的检测，但是规格到底仍然属于结合多种检测方式。DHCPWatcher 结合了签名验证和过程完整性验证等深度数据检测手段。值得一提的是，DHCPWatcher 在做验证的时候存储行为并引入了行为检测和分析，通过对 DHCP 流量行为分析提取从而找到类似攻击流量行为。Toprak 等人[16]同样结合了两种检测方式以应对不同的攻击类型。一种是具有相同 MAC 地址的攻击，应对方式是采用端口安全限制每个端口的最大 IP 数量。另一种是不同 MAC 地址的攻击，应对方式是测量特定时间窗口内的 DHCP 请求数量。该方法存在很大缺点，即对端口的限制不适合应用于云计算环境中。于 oprak 等人[16]相似，Mukhtar 等人[17]提出了在多个层面防御 DHCP 饥饿攻击

的方式。在物理层面使用射频指纹技术或者无线局域网定位技术来验证身份，在链路层上使用结合端口安全和 MAC 地址过滤技术来限制每个端口 MAC 地址数量。这也会照成相同的问题，导致网络的灵活性变差从而不适用云计算等领域。Ishtiaq 等人[18]结合了速度阈值检测和黑白名单过滤的方式。通过检测每个主机特定时间内的 DHCP 消息的数量来进行判断并加入相应的黑白名单，不同的点在分配 IP 之后引入了流量检测进行第三阶段的检测。引入了第三阶段的检测进一步提高了检测的精度。Urama 等人[19]使用加密客户端身份验证、DHCP 嗅探、对称密钥身份认证等方式结合从而检测攻击。不同的是他们引入了基于概率的异常检测方式，这种方式同 Aldaoud 等人[15]的方式异曲同工。两者都在以往数据中找到攻击方式的模式，利用该模式的特性从而找到攻击之间的共性。

在现有有关 SDN 环境下的 DHCP 攻击检测工作中还有一些间接的检测方式。Jony 等人[20]则利用端口扫描来防御 DHCP 饥饿攻击。针对以及分配的 IP 进行端口扫描，如果只是一个常用端口开放则认为其合法，反之不合法。这种检测方式不同于上述直接针对有关 DHCP 数据信息进行判断，而是利用其他的数据信息间接进行判断。然后这种方式只能针对已经分配出去的 IP，并不能做到预防攻击的效果。

此外，Tripathi 等人[21]使用机器学习算法来对 DHCP 数据请求进行处理。该工作在 IPv4 和 IPv6 环境中使用多种单类分类技术构建异常检测框架。然而, 虽然使用了先进的机器学习算法，但是训练数据仍然使用的是单一的 DHCP 请求数据的请求频率。一旦环境发生变化，该算法就要重新使用匹配当前环境的数据进行训练。虽然该方式能很好得分类针对训练环境，但是该算法的扩展性和兼容性很差。

1.2.2 SDN 环境下数据层面洪泛攻击防御研究现状

(一) SDN 环境下基于熵特征的异常检测研究现状

当前的研究中，一些学者将信息熵作为检测 DDoS 攻击的重要手段。通过计算网络数据包或流量特征的多种类型熵值，研究人员能够有效识别攻击流。结合这些熵值结果与其他检测或防御方法，能够实现更加精准的 DDoS 攻击检测与有效的缓解策略。这种基于信息熵的检测方法，因其在处理大规模网络流量中的高效性和准确性，已成为网络安全领域中防御低速 DDoS 攻击的一项重要技术。例如，Kalkan 等人[22]提出了一种高效、低复杂度且兼容现有 SDN 架构的 DDoS 防御方法称为 JESS，通过联合熵的动态应用实现了对已知和未知攻击的全面防护。实验证明该算法对几种已知的攻击都能给出理想的结果。此外，对于不熟悉的攻击类型，该算法的成功率可达 80%。它还为混合攻击提供了 70% 的成功率。Ding 等人[23]提出了在 P4 中实现熵估算和 DDoS 检测的完整方法链，

该方法在执行操作时不需要与控制平面进行交互。因此方法在确保了更小的内存使用和资源消耗的同时可以保证控制器遭受攻击的极端情况下数据平面能自行执行策略。Liu 等人[24]提出了一种基于信息熵分析和深度学习优化的两级检测方法，该方法利用信息熵分析快速筛选可疑端口，利用控制器的 PACKET_IN 消息进行速率检测和特征熵值计算，从而实现粗粒度的可疑流量标记。实验结果表明，该方法在 CICIDS2017 数据集上的检测准确率达到 98.98%，显著优于传统统计分析方法和部分单一深度学习模型。Pandey 等人[25]提出了一种基于信息熵变化的 DDoS 攻击检测方法，系统分析了三种熵指标的性能，采用滑动窗口机制和两种阈值设定来动态检测流量中的异常模式。实验结果表明 ϕ-熵与四分位数阈值的结合在精度和一致性方面表现最佳。Shirsath 等人[26]一种新的检测 SDN 环境下 TCP SYN DDoS 攻击的框架—SYNTROPY。该框架相对于传统的信息熵相比提供了灵活性，可以根据不同的网络条件和攻击模式调整敏感度并采用最小-最大阈值来准确识别攻击模式，从而提高检测准确性。Ahalawat 等人[27]提出一种提出了基于 Rényi 熵的低速率 DDoS 检测方法，并通过信息距离度量网络流量特征分布的变化。该方法结合分组丢弃机制，设计了高效的攻击防御策略。实验表明，该算法的性能优于 Shannon 熵、广义熵和其他统计距离度量。Hassan 等人[28]提出了一种结合熵的统计方法与机器学习技术的混合方法。该方法在 CIC-IDS2017、CSECIC2018 和 CICDDoS2019 等数据集上的实验表明，其检测准确率超过 99.9%，特别是在减轻误报和实现高准确率方面显著优于其他基于深度学习和传统机器学习的方法。

(二) SDN 环境下洪泛攻击动态防御中强化学习的应用研究现状

在当前的 DDoS 攻击检测与缓解研究中，强化学习的应用仍处于初步探索阶段，相关工作相对较少。一些学者将强化学习作为辅助工具，通过动态调整信息熵阈值或机器学习模型参数，以适应不断变化的网络环境。另一些学者则直接利用强化学习算法，根据网络状态动态调整响应策略，以更有效地应对 DDoS 攻击。例如，Feng 等人[29]提出了一种基于强化学习的协作 DDoS 检测方法，设计了一种基于统计学的轻量级无监督分类器。该检测方法利用部署在边缘服务器上的软演员-评论家强化学习模型来动态调整底层无监督分类器的参数配置，以此应对物联网环境的动态变化使其可用于不同类型的物联网设备。此外，该方法在强化学习模块引入协作机制，通过衡量单个节点分类器的检测效果以及基于协作节点的整体检测表现进行奖励或惩罚显著提升了强化学习模型的检测效果和效率。Li 等人[30]提出了一种名为 FAST 的基于强化学习的 DDoS 缓解方法，特别针对车联网（IoV）环境下 DDoS 攻击的检测与缓解。该方法利用 IoV 流量的时空规律和强化学习技术，实现了对动态环境中未知攻击的快速检测与缓解。该方法利用网络连接特征构建 Q-Learning 与 Double Deep Q-Network 相结合的混合强化模型，综合 Kalman 预测偏差和历史正常流量偏差构建奖惩机制，以此提高提升了学习效率和模型性能。Yungaicela 等人[31]一种基于深度强化学

习的灵活 SDN 框架，用于检测和缓解慢速 DDoS 攻击，该方法包括基于 LSTM 的入侵检测系统和基于 DRL 的入侵防御系统。入侵防御系统基于连接的特征、流量类型构建 DRL 模型以实现最大化累积奖励，即学习到对恶意流量的最优缓解策略，同时最小化对正常流量的影响。Rezapour 等人[32] 一种基于强化学习的 LFA 防御系统 RL-Shield，该防御框架在防御 LFA 的同时可以有效地转发数据流量。RL-Shield 设计了两种基于强化学习的路由算法 Efficient Routing Algorithm（ERA）和 Detective Routing Algorithm（DRA），两种算法都采用基于网络状态构建的 DQN。其中 ERA 基于带宽利用率和网络延迟构建奖惩机制，而 DRA 则根据是否有效引诱攻击流量构建奖惩机制。通过强化学习使得 DRA 不断更换路径提高攻击暴露的概率的同时使得 ERA 最大限度地提高网络利用率并最小化网络延迟。Yungaicela 等人[33]在先前工作的基础上提出另一种基于 SDN 和网络功能虚拟化的框架，结合深度学习、强化学习以及动态目标防御机制，用于检测和缓解慢速 DDoS 攻击。该框架在强化学习处理攻击流量时引入 MTD 机制，利用 NFV 的思想在动作空间中增加将攻击流量迁移到影子服务器中，并在检测机制中状态转移和执行动作的成本。以此可以实现最小化成本的限制下最大化地保护合法流量以及防御攻击流量。Li 等人[34]在先前工作的基础上提出了一种基于 Kalman 滤波和迁移学习的强化学习方法以应对 IoV 场景的 DDoS 攻击。该方法根据基站之间的相似性通过将已学习的知识从一个基站迁移到另一个基站，以加速新基站的学习过程，达到快速检测和缓解攻击的目的。此外，该方法利用卡尔曼滤波器估算攻击偏差使得 DDQN 可以不依赖原始数据中的标签完成学习。Xia 等人[35]提出了一种基于集中式通信机制的集中式强化学习路由器节流方法 CRLRT-LC 以解决 DDoS 攻击中的部分可观测马尔可夫决策过程问题。该方法通过深度确定性策略梯度算法动态调整每个控制每个路由器的流量，避免服务器过载，同时尽可能保证合法流量通过。

1.2.3 研究现状小结

上述针对协议层洪范攻击的监测和缓解工作中，大部分都是直接针对 DHCP 数据进行分析和检测[36]。这种检测的数据太过于单一，并且随着环境和攻击方式的改变很容易造成手段失效。即使是存在间接的检测方式，也单单只是在检测阶段有所成效，并不能有效于预防阶段。先前的工作中，并没有任何工作将 DHCP 攻击的问题的解决视角扩展全局网络情况，都只是针对 DHCP 这一特定的数据。同时，也没有工作去将 DHCP 恶意工作转化成在受攻击情况下维持 DHCP 持续运行的优化问题，只是简单的针对 DHCP 请求分析而忽略了最终的目标。

尽管现有研究在 DDoS 攻击的检测与缓解方面取得了显著进展，但仍存在一些不足之处。基于信息熵的攻击检测方法虽有效，但其实时性尚未得到充分验证。此外，大多数研究在设计信息熵指标时，侧重于个体流量特征而非整体系统

的行为模式。这种设计的优点在于能够直接定位攻击个体，但缺点在于难以保障检测过程的实时性和整体性。特别是在 SDN 环境中，基于个体的信息熵计算并不完全适用。由于 SDN 的数据平面主要负责数据转发，信息熵的计算通常需要在控制平面中完成，从而导致数据平面与控制平面之间的数据传输增多，增加了额外的负载和时延。针对这些问题，本研究提出了一种适用于 SDN 网络的整体性信息熵体系，设计了多种全局熵指标，并优化了其计算过程，使其在攻击检测中具有更高的实时性。这一方法能够有效降低控制平面与数据平面之间的冗余传输负担，同时提升 DDoS 攻击检测的效率与准确性。

现有研究利用强化学习防御 DDoS 攻击已取得显著进展并表现出良好效果，但仍存在一定的改进空间。在少数直接通过强化学习动态调整网络的工作中，防御效果依然高度依赖于检测与缓解的准确性或检测阶段的数据标签。本研究旨在优化上述不足，通过将 DDoS 攻击的整体检测任务与个体攻击的检测任务解耦，并将个体检测的责任完全交由强化学习模型完成。这种方法有效地消除了对数据标签的依赖，同时降低了检测算法准确性对防御效果的限制，从而提升系统的适应性和泛化能力。此外，在强化学习对动作进行评估和奖励惩罚的阶段，本研究创新性地引入了动作评估机制。这一机制不仅确保强化学习模型能够最大限度地缓解攻击，还能够在评估中优化策略，最小化对网络环境的长期负面影响。通过这一改进，强化学习模型在提升防御性能的同时，能够更好地维护网络的整体稳定性。

1.3 研究内容

针对双栈 SDN 网络环境中洪泛攻击防御所面临的挑战，本文从协议层攻击检测与数据平面 DDoS 攻击缓解两个方面展开研究，主要研究内容如下：

（1）DHCP 饥饿攻击防御框架设计。针对 DHCP 饥饿攻击导致 IP 地址资源耗尽的问题，设计一种基于 SDN 控制器的 DHCP 攻击防御框架。该框架由恶意 IP 定位器、恶意请求过滤器以及智能 DHCP 请求动态处理算法组成，通过对网络全局流量进行分析，实现对恶意主机的快速定位，并通过多阶段过滤机制对异常 DHCP 请求进行处理，从而有效缓解地址资源耗尽攻击。

（2）双栈网络流量统一建模方法。针对 IPv4 与 IPv6 协议在报文结构和特征维度上的差异，提出一种双栈流量特征统一处理机制，通过特征提取与归一化映射，实现 IPv4 与 IPv6 流量特征的一致性表示，为后续攻击检测算法提供统一的数据输入。

（3）基于信息熵的 DDoS 攻击检测方法。利用 SDN 控制器的全局可视性，构建多维网络熵指标，包括请求数量熵、CPU 使用率熵、带宽熵以及网络时延熵等，通过分析网络状态变化实现对异常流量的实时检测。

（4）基于强化学习的 DDoS 动态防御算法。将 DDoS 攻击缓解问题建模为马

尔可夫决策过程，引入双重深度 Q 网络算法，通过智能体与网络环境的持续交互学习最优防御策略，并在奖励函数中引入动作成本评估机制，以在有效抑制攻击流量的同时降低对正常业务的影响。

1.4 研究章节安排

本文围绕双栈 SDN 环境下基于强化学习的洪泛攻击动态防御算法展开研究，全文共分为六章，各章内容安排如下图1-1所示。

第一章为绪论，主要介绍研究背景与研究意义，综述国内外相关研究现状，并详细阐述本文的研究内容、主要工作以及论文的整体结构安排。

第二章介绍与本文研究相关的理论基础与关键技术，包括软件定义网络的体系结构与工作机制、洪泛攻击的基本模型、双栈网络环境的特点，以及信息熵理论和强化学习等相关理论，为后续研究工作奠定理论基础。

第三章研究协议层攻击的检测与缓解方法，针对 DHCP 饥饿攻击提出一种基于 SDN 控制器的防御框架，并设计恶意 IP 定位器、恶意请求过滤器以及智能 DHCP 请求动态处理算法，实现对异常 DHCP 请求的检测与过滤。

第四章研究数据平面 DDoS 攻击的检测与缓解方法，通过构建双栈网络流量特征处理机制以及基于信息熵的攻击检测模型，并结合强化学习算法实现动态防御策略学习。

第五章对全文研究工作进行总结，分析本文的不足之处，并对未来研究方向进行展望。

1.5 本章小结

本章围绕双栈 SDN 环境下洪泛攻击检测与缓解问题，对研究背景、研究意义、国内外研究现状以及本文研究内容进行了系统阐述。首先，从网络规模持续扩大、DDoS 攻击威胁加剧以及 IPv4/IPv6 双栈长期共存等方面说明了本文研究的现实背景，指出复杂网络环境下洪泛攻击防御需要同时关注检测准确性、响应实时性和系统运行稳定性[10]。其次，结合现有研究进展，对 SDN 环境下协议层洪泛攻击防御、数据层 DDoS 攻击检测与缓解方法进行了梳理，分析了现有方法在单一协议适应性、实时性、全局状态建模以及动态防御策略方面存在的不足。最后，明确了本文从协议层和数据层两个角度展开研究的总体思路，提出面向 DHCP 饥饿攻击的多阶段协同防御框架，以及融合信息熵与深度强化学习的 DDoS 攻击检测与动态缓解方法，并对全文章节安排进行了说明。本章为后续相关理论介绍、算法设计与实验验证奠定了问题基础和逻辑起点。

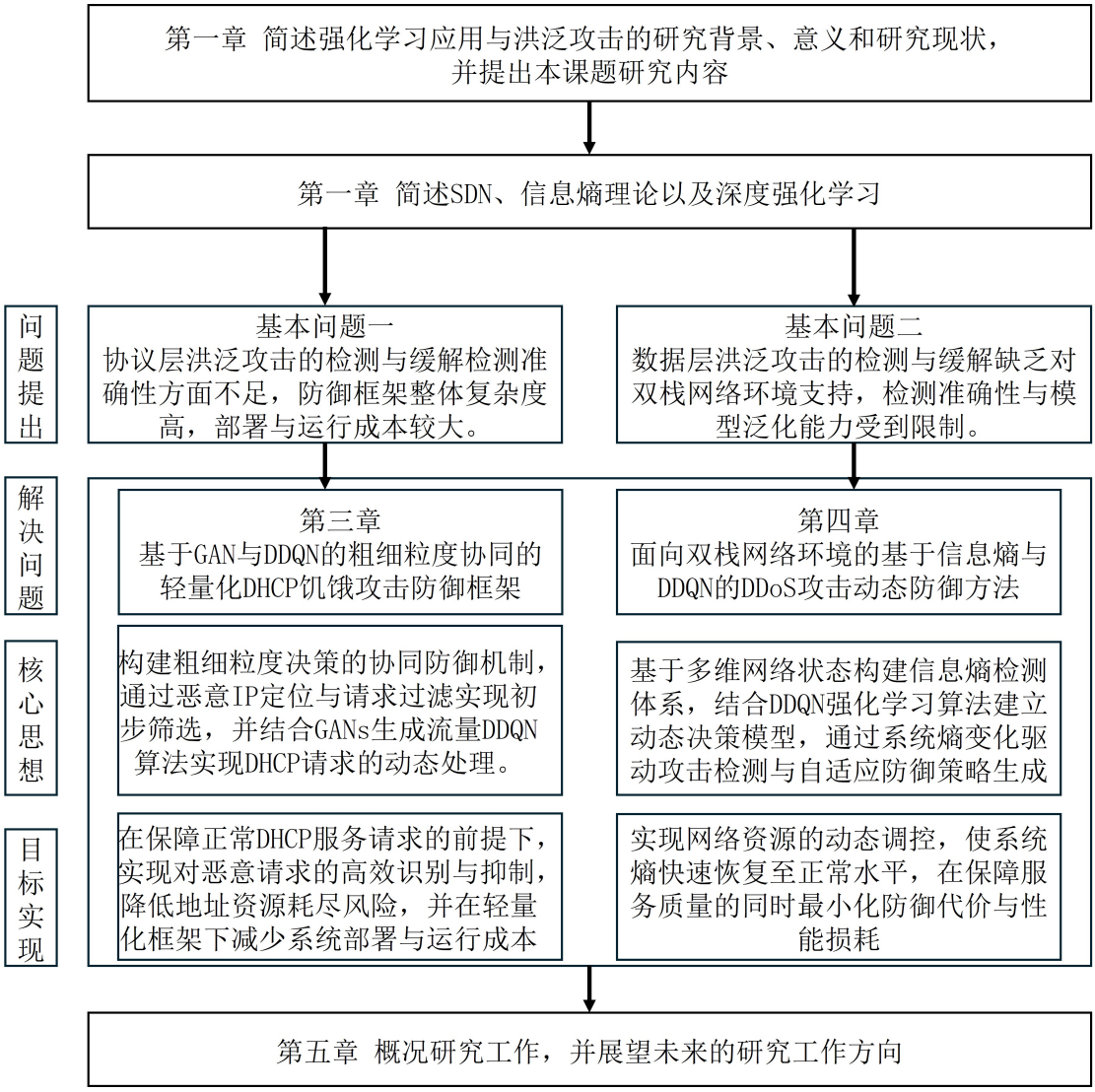


图 1-1 论文的组织结构与章节安排

第二章　相关理论基础

本节围绕本文所涉及的关键技术与理论基础展开说明，从网络控制与管理的整体架构入手，逐步过渡到具体的服务实现与优化方法，对软件定义网络等相关技术进行简要介绍。在此基础上，对信息熵检测方法以及深度强化学习算法进行整理与说明，为后续研究内容的展开提供必要的理论支撑。

2.1　软件定义网络

SDN 通过将网络的控制功能与数据转发功能进行解耦，形成了一种以集中控制为特征的网络架构[37]。与传统网络设备同时承担控制与转发任务不同，这种方式将控制逻辑从底层设备中抽离出来，由集中式控制器统一负责，从而在一定程度上缓解了配置复杂、扩展受限等问题，也更容易适应不断变化的业务需求。在这一架构下，转发设备主要专注于数据包的高速处理，而控制器则基于对全网状态的掌握进行统一决策，并通过标准化接口向数据平面下发控制策略。这样的设计使网络能够以软件方式进行配置与调整，为后续的自动化运维和智能化管理提供了基础条件。

从结构上看，SDN 通常由应用平面、控制平面和数据平面构成。应用平面承载各类网络功能与服务，例如流量调度、安全防护以及负载均衡等，这些应用通过北向接口与控制层交互，将具体需求转化为可执行的策略。控制平面作为核心部分，一般由控制器实现，其主要职责包括维护网络的整体视图，并完成路径计算、策略制定及资源分配等工作，同时通过南向接口将决策结果传递给底层设备。数据平面则由交换机、路由器等转发设备组成，依据控制器下发的流表规则完成数据转发任务。相较于传统网络设备，这些转发节点不再承担复杂决策功能，而是按照既定策略执行操作，从而在一定程度上降低了系统复杂度。

2.2　信息熵检测理论

信息熵最早由 Shannon 提出[38]，用于衡量随机变量不确定性的大小。在信息论中，熵值越大，表示系统的不确定性越高；熵值越小，则表明系统状态越集中、有序。由于信息熵能够从统计分布的角度刻画系统状态变化，因此在异常检测和网络安全领域得到了广泛应用。设随机变量 X 具有 n 种可能取值，其概率分布为 $P(X = x_i) = p_i$，则随机变量 X 的信息熵定义为：

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \tag{2-1}$$

其中，log 通常以 2 为底，熵值的单位为比特。当随机变量取值分布均匀时，熵值达到最大；当取值高度集中时，熵值较小。

在网络环境中，数据包的源地址、目的地址、端口号以及协议类型等字段都可以看作具有随机性的特征变量。对这些特征的分布情况进行统计，并进一步计算其信息熵，可以从整体上刻画网络流量的变化状态。在正常情况下，各类流量特征通常维持在相对稳定的分布范围内，对应的熵值波动也较为平缓；而一旦出现洪泛类攻击，流量结构往往会被打破，例如源地址变得高度离散或目的地址出现集中现象，这些变化会直接反映在熵值的异常波动上。因此，通过观察熵值变化，可以在一定程度上识别潜在的异常行为。

围绕这一思路，基于信息熵的检测方法通常以流量统计结果作为输入，通过对熵值变化的持续跟踪来判断网络状态是否偏离正常范围。实际过程中，一般先在固定时间窗口内对流量进行采集，并统计相关特征的分布情况，随后依据熵的定义完成计算，再将当前结果与正常状态下的参考值进行对比。当偏离程度超过预设范围时，便可认为网络中可能存在异常。与依赖特定攻击特征的检测方式相比，这类方法对具体攻击模式的依赖较弱，在面对未知攻击或变种时仍具有一定的适应能力。

为了降低单一特征带来的误判风险，实际应用中往往不会只依赖某一个熵值指标，而是从多个维度对流量进行综合刻画，例如同时考虑源地址、目的地址以及端口等信息的熵变化情况。多维特征的引入有助于从不同角度反映流量结构的变化，从而提升判断的可靠性。在 SDN 环境下，控制器能够获取较为完整的全局流量信息，这为此类方法的实现提供了便利条件。通过合理设置时间窗口及特征维度，可以在检测效果与系统开销之间取得相对平衡，使其更适用于实时监测场景。

总体来看，基于信息熵的检测方法具有实现相对简单、计算成本较低以及对攻击特征依赖性较弱等特点，在大规模网络环境中具有一定应用价值。不过，这类方法在实际使用中也存在局限，例如对时间窗口和阈值参数较为敏感，在流量波动较大的情况下容易受到干扰。因此，在实际系统中，通常需要将其与其他检测或决策机制结合使用，以提高整体的稳定性与可靠性。基于这一考虑，本文在后续工作中结合强化学习方法，对防御策略进行动态调整，以弥补单一熵检测手段的不足。

2.3 强化学习算法理论

2.3.1 强化学习基本概念

强化学习是一类通过与环境不断交互来获取决策经验的学习方法，其核心在于根据反馈信息逐步调整策略。与依赖标注数据的监督学习或侧重数据结构挖掘的无监督学习不同，强化学习更强调“试探—反馈—修正”的过程[39]，通过环境返回的奖励信号引导行为优化，从而在长期意义上获得更优的决策结果。

在这一过程中，智能体会依据当前所处的状态选择相应动作，环境则在动作执行后给出新的状态反馈，并附带一定的奖励或惩罚信息。随着交互的不断进行，智能体逐渐积累经验，对不同状态下的行为效果形成认知，进而趋向于选择更有利的决策方式。

通常，强化学习问题可以用一个五元组进行描述：

$$\langle S, A, P, R, \gamma \rangle \tag{2-2}$$

其中，S 表示状态空间，A 表示动作空间，P 表示状态转移概率，R 表示奖励函数，γ 为折扣因子，用于衡量未来奖励的重要性。

2.3.2 马尔可夫决策过程

强化学习问题通常被建模为马尔可夫决策过程（Markov Decision Process，MDP）。在 MDP 中，系统在任一时刻的状态转移仅与当前状态和当前动作有关，而与历史状态无关，即满足马尔可夫性。

在给定状态 $s_t \in S$ 下，智能体选择动作 $a_t \in A$，环境以概率 $P(s_{t+1}|s_t, a_t)$ 转移至下一状态 s_{t+1}，并产生即时奖励 $r_t = R(s_t, a_t)$。智能体的目标是在长期交互过程中最大化累积折扣奖励，其定义如下：

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2-3}$$

2.3.3 价值函数与策略函数

为评估策略的优劣，强化学习中通常引入价值函数的概念。状态价值函数 $V^\pi(s)$ 表示在策略 π 下，从状态 s 出发所能获得的期望累积回报，其定义为：

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right] \tag{2-4}$$

相应地，动作价值函数 $Q^\pi(s, a)$ 表示在状态 s 下执行动作 a 后，遵循策略 π 所能获得的期望回报，其定义为：

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right] \tag{2-5}$$

策略函数 $\pi(a|s)$ 描述了在状态 s 下选择动作 a 的概率分布，是强化学习中决策行为的核心。

2.3.4 基于值函数的强化学习算法

基于值函数的方法通过估计动作价值函数来指导策略优化，其中 Q-learning 是最具代表性的算法之一。Q-learning 通过迭代更新动作价值函数，使其逐步逼近最优动作价值函数 $Q^*(s,a)$，其更新公式为：

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right] \tag{2-6}$$

其中，α 为学习率。

在状态空间和动作空间较大时，传统 Q-learning 难以直接应用。DDQN 算法相较于 DQN 算法，其主要针对后者的过估计问题，改变了目标值的计算方法，其他地方与 DQN 算法完全一致。DQN 利用深度神经网络不断逼近表示给定状态和动作的期望累计回报的参数化值函数 Q。DQN 具有两个网络结构，分别为主网络和目标网络。主网络的参数用 θ 表示，目标网络则用 θ^- 表示。在 DQN 中，使用主网络来计算当前动作的 $Q(S_t,a_t;\theta)$。动作 a_t 使用 ε-greedy 策略进行选择，这能保证进行一定的探索量。因此，我们可以得到：

$$a_t = \begin{cases} \arg\max_{a_t} Q\left(S_t,a_t;\theta\right), & 1-\varepsilon \\ random & \varepsilon \end{cases} \tag{2-7}$$

随后将 a_t 输入到环境中可以得到 (S_{t+1},r_t)。通过这个相互作用可以到四元组 (S_t,a_t,r_t,S_{t+1}) 并存储到经验区中。DQN 可以利用小批量处理从经验区中随机小批量选择一组经验进行训练。在训练过程中，将 S_{t+1} 传入到目标网络中，我们可以得到不同动作的 $Q(S_{t+1},a_{t+1};\theta^-)$，并从中选择最大的 $\max_{a+1} Q(S_{t+1},a_{t+1};\theta^-)$。从而可以计算实际值 $\hat{y}$,$\hat{y}$ 可定义为：

$$\hat{y} = r + \gamma \cdot \max_{a_{t+1}} Q(S_{t+1},a_{t+1};\theta^-) \tag{2-8}$$

其中，γ 是折扣因子。

这时，我们可以根据预测值与实际值之间的损失来进行方向传播从而优化深度神经网络的参数，使得之后的预测值更加接近于真实值。计算损失函数的公式如下：

$$Loss = (\hat{y} - \max_{a_t} Q(S_t,a_t;\theta))^2 \tag{2-9}$$

DDQN 与 DQN 大部分都相同，只有一步不同，那就是在选择 $Q(S_{t+1},a_{t+1};\theta^-)$ 的时候。DQN 总是选择最大值 $\max_{a+1} Q(S_{t+1},a_{t+1};\theta^-)$。而 DDQN 不同,DDQN 首先从主网络网络中找到 $\max_{a_{t+1}} Q(S_{t+1},a_{t+1};\theta)$ 的 a_{t+1}, 然后再找到 a_{t+1} 对应的目标网络的输出值 $Q(S_{t+1},a_{t+1};\theta^-)$。因此，在 DDQN 中 $\hat{y}$ 的计算如下：

$$\hat{y} = r + \gamma \cdot Q(S_{t+1},a_{t+1};\theta^-) \tag{2-10}$$

与 DQN 中相同，我们利用真实值与预测值之间的 TD 误差来进行反向传播从而更新主网络中的参数 θ,θ 更新过程的公式如下：

$$\theta_{t+1} \leftarrow \theta_t + \alpha[\hat{y} - Q(S_t, a_t; \theta)]\nabla_\theta Q(S_t, a_t; \theta) \tag{2-11}$$

其中，α 是深度神经网络的学习率。因此，可以如上不断更新主网络的参数 θ，并在一定的迭代次数之后同步参数到目标网络。循环往复，直到损失函数收敛之后，便可以完成对模型的训练。

2.3.5 强化学习在网络安全防御中的适用性分析

在网络安全防御场景中，网络状态往往呈现出动态变化和一定的不确定性，攻击行为也会随时间不断演变，这使得防御策略难以依赖固定规则长期有效[40]。相比之下，强化学习通过与环境的持续交互来调整决策策略，能够在变化的环境中逐步形成更合适的应对方式，因此在处理未知或复杂攻击场景时具有一定优势。

基于这一特点，将强化学习引入双栈 SDN 环境下的洪泛攻击监测与缓解问题，有助于在不同网络状态下对防御策略进行动态调整，从而提升整体防护的灵活性与适应能力，也为后续面向数据平面的 DDoS 防御方法提供了相应的理论支撑。

2.4 本章小结

本章围绕本文所涉及的相关理论与关键技术进行了梳理。从网络架构角度出发，对软件定义网络的基本思想及其结构进行了说明，并结合其集中控制的特点，讨论了在提升网络灵活性与管理能力的同时可能带来的安全问题，为后续研究提供了背景基础。

在此基础上，对信息熵检测方法进行了整理，结合网络流量特征分析其在异常行为识别中的作用。通过对熵建模方式及多维特征的讨论，可以看出该方法在反映流量分布变化方面具有一定效果，但在复杂环境下仍存在一定局限。随后，对强化学习相关方法进行了介绍，重点关注 DDQN 算法[41]在策略稳定性方面的表现。通过对其机制的分析，可以看出其在动态决策问题中具有一定优势，这也为其在网络安全防御场景中的应用提供了依据。

总体来看，本章内容为后续双栈 SDN 环境下洪泛攻击的检测与防御方法研究提供了必要的理论基础与技术支撑。

第三章　协议层攻击的检测与缓解

3.1　引言

在 SDN 网络环境中，控制器能够集中掌握网络拓扑、主机接入状态和流量转发规则，这为协议层攻击检测与防御提供了传统网络中难以具备的全局视角。DHCP 服务作为网络地址自动分配的重要基础服务，直接关系到合法用户能否正常接入网络。当攻击者伪造大量 DHCP 请求并持续占用地址池资源时，DHCP 服务器中的可分配 IP 地址会被快速耗尽，正常用户将无法获得合法地址，从而影响网络服务的连续性和可用性。因此，如何在 SDN 环境下及时识别 DHCP 饥饿攻击，并在较低系统开销下保障 DHCP 服务持续运行，是协议层洪泛攻击防御中需要重点解决的问题。

现有 DHCP 饥饿攻击防御方法多从请求速率、端口安全、身份认证、黑白名单过滤或异常行为识别等角度进行设计。这类方法虽然能够在一定程度上识别异常 DHCP 请求，但仍存在一定不足。一方面，部分方法主要依赖 DHCP 请求本身的局部特征，缺乏对全局网络状态、主机行为关联性以及地址池变化过程的综合分析，面对攻击方式变化时适应性较弱。另一方面，单纯依靠规则过滤或阈值判断容易出现误判或漏判，尤其是在正常请求与恶意请求混杂的情况下，难以同时满足攻击抑制和合法用户接入保障的要求。

针对上述问题，本章提出一种面向 DHCP 饥饿攻击的多阶段协同防御框架。该框架依托 SDN 控制器的全局可视能力，首先通过自学习表记录主机 IP、端口、总流量和 DHCP 流量等信息，并利用流量相关性分析定位疑似恶意 IP 和恶意端口；随后通过恶意请求过滤器安装保护规则，对明显异常的 DHCP 请求进行初步过滤；最后引入基于 DDQN 的智能 DHCP 请求动态处理算法，对粗粒度检测后的请求进行细粒度判别与动态处理。通过上述设计，本章将 DHCP 饥饿攻击防御从单一请求检测扩展为面向地址池持续运行的动态优化问题，在释放恶意占用地址、降低攻击影响和保护合法请求之间实现协同平衡。

3.2　DHCP 饥饿攻击防御框架

为有效应对 SDN 环境下的 DHCP 饥饿攻击，本文设计了一种 DHCP 饥饿攻击防御框架。该框架面向数据平面与控制平面之间的协同防御过程，充分利用 SDN 控制器具备全局网络视图和集中策略下发能力的特点，对 DHCP 请求行为、主机流量状态以及地址池变化情况进行综合分析，从而实现对异常请求的检测、过滤和动态处理。与传统仅依赖单一规则或固定阈值的防御方式不同，本文所提出的框架并不只关注单个 DHCP 数据包本身，而是从网络整体运行状态出发，结合主机流量特征、端口行为和 DHCP 地址池资源变化情况，对攻击行为进行

分阶段识别与处理。

通过上述设计，本文将 DHCP 饥饿攻击防御问题从传统的请求级异常检测扩展为面向 DHCP 服务持续运行的动态优化问题。该框架不仅能够识别和过滤明显的恶意请求，还能够根据网络状态变化自适应调整处理策略，在攻击流量抑制、地址资源恢复和合法请求保护之间取得平衡。总体而言，该防御框架具有较好的轻量性、透明性和可扩展性，适用于 SDN 环境下协议层洪泛攻击的动态防护场景。

3.2.1 系统架构

DHCP 饥饿攻击防御框架的架构如图3-1所示。该框架引入了三个模块以实现对攻击的防御：恶意 IP 定位器、恶意请求过滤器以及智能 DHCP 请求动态处理算法。其中，恶意 IP 定位器通过分析全局流量并结合触发式检测机制来识别恶意攻击。在触发攻击检测时，对已分配 IP 的全局流量进行分析。由于恶意流之间通常具有较强的相关性，因此通过挖掘流量间的共性特征，可以构建强相关的攻击流候选集合。基于该候选集合，可进一步定位对应的恶意端口，并针对其 DHCP 数据包进行检测。在此基础上，保护规则激活器将特定的防护规则下发至交换机，以过滤较为明显的恶意 DHCP 请求。一旦恶意 IP 定位器识别出恶意 IP，即可进一步定位对应的恶意端口。对于来自该端口的 DHCP 请求，将建立专用的防护规则并加载至保护表中，同时结合自适应超时重传机制，对明显的恶意流量进行有效过滤。在完成粗粒度的恶意 IP 定位与请求过滤后，剩余流量将交由智能 DHCP 请求动态处理算法进行处理。该算法基于细粒度分析对数据包进行判别与决策，并采取相应的处理策略。同时，利用 DDQN 的经验回放机制，将处理过程中的数据存储至经验池中，从而不断优化模型性能，提高整体防御效果。

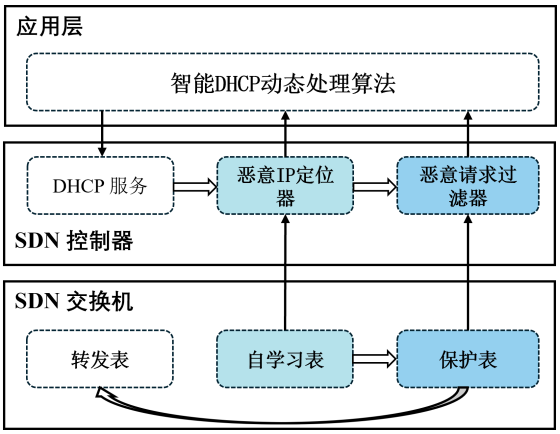


图 3-1 DHCP 饥饿攻击防御框架的系统架构

3.2.2 恶意 IP 定位器

恶意 IP 定位器不同于传统的实时检测器，传统的实时检测器由于需要实时监听并实时处理数据包，往往会消耗大量的带宽，这会对网络性能照成一定的影响。本文设计的恶意 IP 定位器采用激活性的检测方法，在不触发激活条件时，网络正常运行。这个阶段不会对网络性能照成影响，不影响网络的正常运行。同时，为了防止固定的激活条件导致恶意 IP 定位器的灵活性较差，采用动态调整激活条件的策略。

当系统触发激活条件时，利用 SDN 控制器一般处理能力，可以获取当前网络的状态。同时由于 SDN 框架带来的优势，DHCP 应用可以实时与 SDN 控制器实时共享网络信息以处理网络事务。

利用 SDN 控制器的一般处理能力，构建自学习表。如图3-2所示，自学习表记录 mac 地址对应的主机获取的 IP、所在端口、总流量以及 DHCP 数据流量。在 DHCP 服务启动时，获取 IP 池的数量，设置 3/4 池的数量作为阈值。当池数量降低到该阈值时触发检测条件。分析自学习表中的信息，找出疑似恶意 IP 列表并交由智能 DHCP 请求动态处理算法处理。如果未发现恶意 IP 则对以同样的方法调整新的阈值，进入下一轮的检测。问题的核心在于对于自学习表中的信息的

自学习表

Mac ID	IP	Port	Traffic	DHCP_Traffic
02:00:00:11:17:8a	10.1.1.11	1	High	Common
02:00:00:0d:a1:6c	10.1.1.12	2	Medium	Common
02:00:00:22:ac:21	10.1.1.13	3	Low	High
⋮				
02:22:33:22:ac:66	10.1.1.168	3	Low	High
02:44:55:22:ac:77	10.1.1.169	3	Low	High

图 3-2　自学习表示例

分析。如何分析表中的信息可以找到疑似的恶意 IP。针对 DHCP 服务的 IP 池的攻击，无疑乎在于获取非法 IP。而非法 IP 最大的特点是不像合法 IP 一般会根据 TCP/IP 协议簇的需要自主发出数据包以保证在网络中的正常运行。而 DHCP 饥饿攻击攻击以及其他 DHCP 饥饿攻击也是如此，不同的是 DHCP 饥饿攻击会产生混淆流量。就算 DHCP 饥饿攻击会产生混淆流量，也是人为构造的数据包。而大量的非法 IP 共用一套生成混淆流量的方案，则其流量则具有极强相关性。同时，DHCP 饥饿攻击攻击以及其他 DHCP 饥饿攻击一样，都是需要进行无限轮

的攻击流，这也为我们进行流量分析提供了思路。可以将每个主机的流量分成其他流量以及 DHCP 数据流量，表示如下：

$$\text{Traffic}\ {}^{i}_{ALL} = \text{Traffic}\ {}^{i}_{\text{else}} + \text{Traffic}\ {}^{i}_{DHCP} \tag{3-1}$$

其中 $Traffic^{i}_{ALL}$ 表示 i 主机的总流量，$Traffic^{i}_{else}$ 表示 i 主机的除了 DHCP 外的其他流量，$Traffic^{i}_{DHCP}$ 表示 i 主机的 DHCP 数据流量。我们将每个主机的流量分成两个部分进行分析，相比于分析总体流量将流量划分成两部分进行分析更加简单且明确。

对于非 DHCP 数据部分的分析。我们假设非 DHCP 数据由每一轮的混淆数据组成，可以表示如下：

$$\text{T}raffic_{else} = \sum_{m=1}^{\infty} Traffic^{m}_{confusing} \tag{3-2}$$

其中 $Traffic^{m}_{confusing}$ 表示第 m 轮的混淆流量。当 DHCP 饥饿攻击攻击进行到第 k 轮时，触发了激活条件，则这个时候会产生两张情况，一种是已经运行了 k 轮的混淆流量，另一种是还没运行 k 轮的混淆流量。它们之间的关系可以表示如下：

$$Tr_{else} = \frac{\sum_{m=1}^{k-1} Traffic^{m}_{confusing}}{\sum_{m=1}^{k} Traffic^{m}_{confusing}} \tag{3-3}$$

其中 Tr_{else} 为除去 DHCP 数据流量的其他流量的相识比，而 $\sum_{m=1}^{k-1} Traffic^{m}_{confusing}$ 表示第 k-1 轮的混淆流量，$\sum_{m=1}^{k} Traffic^{m}_{confusing}$ 表示第 k 轮的混淆流量。当 k 为第一轮时，由于 k-1 轮为 0，即尚未发起攻击所有不会存在自学习表中。因此 Tr_{else} 都为 1。当 k 为第二轮时，存在发起两轮以及未完成第二轮的攻击流。同时在不考虑丢包的情况下 $\sum_{m=1}^{2} Traffic^{m}_{confusing}$ 为 $\sum_{m=1}^{1} Traffic^{m}_{confusing}$ 的两倍。因此 Tr_{else} 都为 0.5。依次类推，当 k 大于第二轮时，随着轮数的增大 Tr_{else} 只会比 0.5 大。当 k 足够大的时候，会发现 Tr_{else} 会无限趋近于 1。因此可以推断出 Tr_{else} 的范围为 0.5 到 1 之间。

对于 DHCP 数据部分的分析。对于 DHCP 数据流量的分析相对来说更加容易，因为 DS 的 DHCP 遵循合法的 DHCP 的过程。当 DHCP 饥饿攻击攻击进行到第 k 轮时，触发了激活条件，则这个时候仍然会产生两张情况，一种是已经进行 k 轮的攻击流，另一种是还没运行 k 轮的混淆流量。它们之间的关系可以表示如下：

$$Td_{DHCP} = \sum_{m=1}^{k} Traffic^{m}_{DHCP} - \sum_{m=1}^{k-1} Traffic^{m}_{DHCP} \tag{3-4}$$

其中 Td_{DHCP} 为同一轮不同主机 DHCP 数据流量的差值，$\sum_{m=1}^{k-1} Traffic^{m}_{DHCP}$ 表示第 k-1 轮的攻击流量，$\sum_{m=1}^{k} Traffic^{m}_{DHCP}$ 表示第 k 轮的攻击流量。根据 Td_{DHCP} 的定义可以很容易发现 Td_{DHCP} 恒定为进行一次完整的 DHCP 过程的流量。

Algorithm 1 基于流量分析的恶意 **IP** 定位算法

输入**:** 自学习表 SLT; 阈值 $threshold$; 地址池 $pool$; 流量差异阈值 TD;

输出**:** 恶意 IP 集合 MI; 恶意端口集合 MP;

$MI \leftarrow [\,], MP \leftarrow [\,]$
RankByTraffic(SLT)
$S \leftarrow 0, E \leftarrow 0, Cur \leftarrow 0$
$Flag \leftarrow 0$
while $Cur <$ len$(SLT) - 1$ **do**
  $TrE \leftarrow \dfrac{SLT[Cur][Traffic]}{SLT[Cur+1][Traffic]}$
  $TrD \leftarrow SLT[Cur+1][Traffic] - SLT[Cur][Traffic]$
  if $(0.5 \leq TrE \leq 1)$ **and** $(TrD \leq TD)$ **and** $(SLT[Cur][Port] = SLT[Cur+1][Port])$ **then**
    $E \leftarrow E + 1$
  else
    if $E - S \geq 2$ **then**
      $Flag \leftarrow 1$
      for $i = S$ **to** $E - 1$ **do**
        MI.add($SLT[i][IP]$)
        MP.add($SLT[i][Port]$)
      end for
    end if
    $E \leftarrow Cur + 1$
    $S \leftarrow Cur + 1$
  end if
  $Cur \leftarrow Cur + 1$
end while
SendToIDRDPA(MI)
if $Flag = 1$ **then**
  $threshold \leftarrow \dfrac{\text{len}(pool)}{2}$
end if
output(MI, MP)

算法1展示出了恶意 IP 定位器的伪代码。输入 SLT 是自学习表的集合，threshold 为检测触发阈值，pool 为 DHCP 服务的 IP 池，TD 是一轮完整的 DHCP 所需的流量。步骤 1 到步骤 4 为初始化参数的步骤，其中包括创建恶意 IP 集合 MI、恶意端口集合 MP。随后对 SLT 的表项依据总流量进行排序，并初始化滑动窗口

的参数以及检测结果标识。主循环是从步骤 5 到步骤 22。在每次循环迭代中该算法判断当前位置的总流量、DHCP 流量以及端口是否满足强相关性，如果满足强相关性则将其加入滑动窗口并继续下一轮循环。如果不满足强相关性，则判断前面的滑动窗口的大小，如果滑动窗口不止一个对象，则将滑动窗口中的条项记录到 MI 和 MP 中并将窗口的前后指针移动到当前位置继续下一轮循环。循环结束后到步骤 23，该步骤为根据上述循环获得的恶意 IP 集合递交给智能 DHCP 请求动态处理算法处理。步骤 24 到 26 为经过智能算法处理完之后根据当前 IP 池的状态调整当前的检测触发阈值。

复杂度分析：设自学习表 SLT 中的记录数为 n。该算法首先需要调用 $RankByTraffic(SLT)$ 按流量大小对自学习表进行排序，若采用常规比较排序方法，该步骤的时间复杂度为 $O(n\log n)$。排序完成后，算法通过变量 Cur 对 SLT 进行一次线性遍历，在遍历过程中主要执行流量比值、流量差值和端口一致性判断等操作，这些操作均为常数时间，因此该部分时间复杂度为 $O(n)$。当发现满足条件的连续可疑区间时，算法会将相应的 IP 地址和端口加入 MI 与 MP，最坏情况下所有记录都可能被加入集合，其时间开销仍为 $O(n)$。同时，$SendToIDRDPA(MI)$ 的执行开销与恶意 IP 集合规模有关，最坏情况下也为 $O(n)$。因此，算法总体时间复杂度主要由排序过程决定，为 $O(n\log n)$。在空间复杂度方面，算法主要需要维护恶意 IP 集合 MI 和恶意端口集合 MP，最坏情况下二者存储规模均与 SLT 记录数成正比；若排序过程采用原地排序，则额外空间主要来自这两个集合，若采用非原地排序，则可能引入额外的线性辅助空间。综合来看，该算法的空间复杂度为 $O(n)$，整体上能够在自学习表规模增大时保持较好的可扩展性。

3.2.3 恶意请求过滤器

在恶意 IP 定位器以及智能 DHCP 请求动态处理算法的作用下，可以定位到恶意攻击流所在的恶意端口。恶意请求过滤器的作用在于根据恶意 IP 定位器提供的恶意端口，将相应的保护规则安装到保护表中以初步过滤后续的攻击流。恶意请求过滤器会创建两种形式的保护规则，一种规则是基于所有端口和 MAC 的，对来自于所有端口以及所有 MAC 的 DHCP 请求会不做任何处理，这条规则的优先级最低。一种规则是针对恶意端口的 DHCP 数据包安装实时的过滤规则，这种规则有一个 1s 的硬超时时间。当第一次接收到恶意端口的 DHCP 数据包时会记录其 MAC 地址并创建相应的保护规则安装到交换机中。当后续接收到数据包时，会判断其超时重传次数，当其超时重传的次数没达到阈值时，则仍然让该 DHCP 数据进入重传超时并重置硬超时时间。反之，则将数据包交由智能 DHCP 请求动态处理算法处理。如果在硬超时时间内没有接收到改数据包，则会将改保护规则移除。

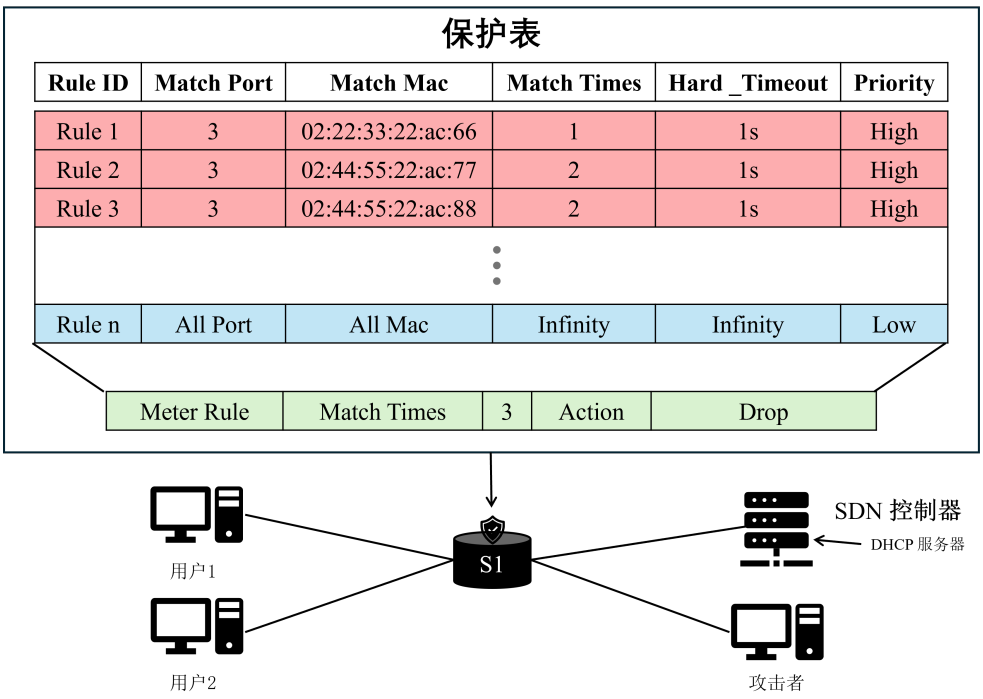


图 3-3　保护表示例

如图3-3所示，攻击流通过 S1 中的端口 3，而 user1 和 user2 的正常流量通过 S1 的 1 和 2 端口。当恶意 IP 定位器检测到 Attacker 发起的攻击流之后，会在释放被恶意获取的 IP 之后记录器所在的 3 端口为恶意端口。当后续 Attacker 仍然发起攻击时，保护规则激活器会记录从恶意端口 3 传输的 DHCP 数据包，并安装相应的规则到保护表并安装到交换机中。由于安装的来自恶意端口的保护规则属于高优先级，所以会对攻击流进行拦截验证。如果 Mactch Times 没有达到超时重传的次数要求则会强行让该 MAC 的 DHCP 数据包进入重传超时。则 Attacker 后续的数据包会被强行要求进入超时重传，因此 Attacker 发送的 DHCP 数据在未达到超时重传要求时数据包不会被处理。而当下一轮发起相应的攻击流时，前一轮产生的表项已经因为硬超时被移除则会重新载入表项进行重新计数。因此，攻击流则会一直被强行进入超时重传而无法将数据包传输到 DHCP 服务器中，从而阻止攻击流的进一步攻击。

对超时重传的次数要求需要一个权衡的值，如果数组太高则会过滤掉良性流，相反则会导致很多的恶意流未被过滤。为了满足上述的要求，本文通过数据分析的方式确认合适的超时重传次数。具体来说，我们利用 DHCP 饥饿攻击以及本文的 DHCP 饥饿攻击攻击产生大量的虚假请求以及大量正常用户的大量的正常请求，利用控制器强行使得 DHCP 请求无限超时重传，记录这些请求的超时重传请求次数。分别统计虚假请求和正常请求超时重传的数据分布，可以发现虚假请求的超时重传次数大部分为 1，极少部分会超过 3。而正常的请求中几乎所有请求的次数都会大于等于 3。因此我们设定超时重传次数为 3，因为在该阈值下可以屏蔽大量的虚假请求的同时可以保证良性流不会被过滤掉。

Algorithm 2 恶意 **DHCP** 请求过滤流程

输入**:** 保护表 PT; 阈值 α;

输出**:** 处理后的数据包 $Packet$;

1: $Packet \leftarrow$ ListenFlow()
2: $MAC \leftarrow Packet.src$
3: **if** IsDhcpPacket(Packet) **and** IsMaliciousPort(Packet) **then**
4: **if** $MAC \in PT$ **then**
5: **if** $PT[MAC][MatchTimes] \geq \alpha$ **then**
6: handle_PacketIn(Packet)
7: **else**
8: $PT[MAC][MatchTimes] + +$
9: Drop(Packet)
10: **end if**
11: **else**
12: InstallProtectionRule(Packet)
13: **end if**
14: **end if**
15: **output**(Packet)

算法2展示出了过滤恶意攻击的伪代码。输入 PT 为保护表，α 为超时重传阈值。步骤 1 和步骤 2 为数据平面监听接收的数据包，并获取该数据包的 MAC 地址。步骤 3 到步骤 14 为算法的主体部分，接收到数据包判断该数据包是不是属于 DHCP 数据包并判断是否来着于恶意端口。如果均不是 DHCP 包或者不是来自恶意端口则不对数据包进行处理直接输出该数据包。如果属于 DHCP 数据包且来自恶意端口，则判断保护表中是否有该表项。如果没有该表项则安装保护规则到表项中，如果有该表项则判断其是否满足超时重传要求。符合要求则转发该数据包，如果不符合则丢弃该数据包。

复杂度分析：设保护表中的规则数量为 m。恶意请求过滤器对每个到达数据包进行一次 DHCP 类型判断、恶意端口判断以及保护表匹配操作。若保护表采用哈希结构存储，则单个数据包的平均查询与更新复杂度为 $O(1)$；若采用顺序表存储，则最坏情况下需要遍历保护表，时间复杂度为 $O(m)$。因此，在哈希实现下，算法对单个数据包的平均时间复杂度为 $O(1)$，最坏时间复杂度为 $O(m)$。在空间复杂度方面，算法主要存储保护表及其匹配次数、超时时间等信息，空间复杂度为 $O(m)$。

3.2.4 智能 DHCP 请求动态处理算法

（一）问题描述

每个网络中的 IP 数量有限，因此需要 DHCP 服务去动态分配 IP 给网络中的主机使其能进行通信。但是网络环境复杂多变，主机的数量随时都可能发生改变，尤其是在云计算环境中。如果无法合理分配 IP 会影响云计算用户的体验，严重会导致网络陷入僵化[42]。当控制器接收到网络中传输过来的 DHCP 请求时，如果不加以检测和控制也导致 DHCP 服务中的 IP 被恶意用户消耗殆尽。同时检测无法做到百分之百防范，如果不能及时释放恶意 IP，同样会导致上述的结果。综上所述，无论什么阶段检测恶意请求的结果都是为了使得 DHCP 服务能够被合法用户使用。因此，我们可以将检测任务转化为在受攻击情况下维持 DHCP 服务持续运行的优化问题。在 DHCP 饥饿攻击防御框架结构中，首先使用恶意 IP 定位器和恶意请求过滤器进行粗粒度的检测。然而粗粒度的检测会导致正常的请求被误判为虚假请求。因此，我们的目标对根据粗粒度检测后的请求进行细粒度的分析和判别，从中找出正常请求与虚假请求并对其做出相应的释放或者保留。最终目标是为了保障正常的请求不能被误判的前提下保证虚假的请求尽可能被全部释放。

（二）系统模型

物理网络。物理网络是一个无向连通图，用 $G=(N,P)$ 表示，其中 N 表示该网络中的一个正常用户或者恶意用户，P 表示该用户所属交换机端口。T_{ut} 表示用户 $(u \in N)$ 的 t 时刻的吞吐量。F_{upt} 表示端口 $p \in P$ 的某一时刻的用户 u$(u \in N)$ 所属端口 DHCP 请求频率。R_{ut} 表示用户 u$(u \in N)$ 的某一时刻的 DHCP 请求数据包。RI_t 表示 t 时刻网络中的 DHCP 服务 IP 池中未分配 IP 数量。每个网络中的 IP 数量有限，这限制了网络中主机的 DHCP 请求数量。

DHCP 请求。受网络中的真实主机数量有限以及 DHCP 饥饿攻击的请求方式单一限制，以此为基础训练的模型往往容易针对特殊的情况造成过拟合的效果。另一方面，由于无论在常规的 DHCP 饥饿攻击还是 DHCP 饥饿攻击攻击中攻击速率会根据环境和攻击者习惯的不同而发生，因此可以通过 GAN 生成不同速率不同形式的攻击数据以增加攻击场景的多样性和完整性。为了避免上述的限制，本文引入生成式对抗网络 (GAN) 到整个 SDN 网络中以优化对智能 DHCP 请求动态处理算法的训练。通过控制器的一般处理能力，可以获取到网络中所有的 DHCP 请求。按照 R_{ut} 的 DHCP 请求所属类型与 T_R 绑定形成二元组 (R_{ut},T_R)，T_R 表示请求 R_{ut} 的类型。将该二元组的请求传入 GAN 中生成相应的请求 (R'_{ut},T_R)。

（三）问题公式化

这一部分提出 DHCP 请求动态处理的数学定义。首先，对问题进行了相关的约束。其次，明确了问题的优化目标。

约束。一个有效的处理战略必须确保一下条件。由于每个环境中的 IP 数量有限，因此网络中已分配的 IP 数量加上未分配的 IP 数量不能超过可分配的 IP 数量。具体如下所示：

$$\sum_{n\in N} UI_{ut} + RI_t \leq NI_t \tag{3-5}$$

同时，要求不能将正常用户的 IP 误判为恶意 IP 释放。具体如下所示：

$$\sum A(r_{ut}, 0) \leq 0, \forall r \in R \tag{3-6}$$

目标。本优化问题的关键目标在与受攻击情况下，维持 DHCP 服务的 IP 池不被消耗殆尽并且释放恶意 IP。换句话说，就是在受到攻击的情况下让 DHCP 的服务能一直持续工作，这是本文需要研究的问题。因此，我们使用受攻击情况下 DHCP 服务持续运行时间 T_{ALL} 来作为度量算法的标准。T_{ALL} 可以计算为：

$$T_{ALL} = [T_{TN} \times f_{TN} + T_O \times (1 - f_{TN})] + T_F \tag{3-7}$$

其中 f_{TN} 为将正常 IP 释放的误判标识，T_{TN} 则为因为将正常 IP 释放的误判标识而终止的情况下接受正常请求所用的时间，T_O 则为因为 IP 溢出而终止的情况下接受正常请求所用的时间。T_F 处理恶意请求所用的时间。T_{TN} 和 T_O 可以用公式计算为：

$$T_{TN} \mid T_O = \sum_{r\in R} a(r_{ut}, 1) \times F_{upt} \tag{3-8}$$

T_F 可以用公式计算为：

$$T_F = \sum_{r\in R} a(r_{ut}, 0) \times F_{upt} \tag{3-9}$$

因此优化目标可以定义为：$max\ T_{ALL}$ s.t (3 – 5)-(3 – 6)

$$f_{TN} \in \{0, 1\} \tag{3-10}$$

其中 f_{TN} 是二进制变量，当 DHCP 服务因为将正确的 DHCP 请求错误判断为恶意的 DHCP 请求时，它的值为 1，否则为 0。

（四）Markov 决策过程

马尔可夫决策过程[43]是强化学习[44]中的一个核心概念，它提供了一种数学模型来描述决策者在不确定环境中进行决策的过程。它基于马尔可夫性质[45]，通过定义状态、行动、转移概率函数和奖励函数来描述决策过程，并通过最优化方法来求解最优策略[46]。MDP 可以用五元组 $(S, A, P_{SA}, R, \gamma)$ 来表示，其中 S 表示一组状态空间，A 是一组动作空间，P_{SA} 是在某一状态 S_i 下，采取动作 A_i，转移到 S_{i+1} 转态的概率，γ 是折扣因子。

状态空间：表示网络在某一时刻的具体状态。状态空间包含智能 DHCP 请求动态处理算法中的四种类型的信息。他可以定义为等式 (3 – 11）。其中

$th(t) = [th_{u1}(t), th_{u2}(t), ..., th_{u|N|}(t)]$ 表示 t 时刻每台主机的吞吐量，$pf(t) = [pf_{u1}(t), pf_{u2}(t), ..., pf_{u|P|}(t)]$ 表示 t 时刻每个端口的 DHCP 请求的频率，$rf(t) = [rf_{u1}(t), rf_{u2}(t), ..., rf_{u|N|}(t)]$ 表示到 t 时刻为止每台主机的 DHCP 请求的频率，$RI(t)$ 表示 t 时刻整个网络中未分配 IP 的数量。$|N|$ 表示网络中主机的数量，$|P|$ 表示网络中交换机端口的数量，$|NI|$ 表示网络中可分配的 IP 数量，因此空间维度为 $2|N| + |P| + |NI|$。这些信息的整合提供了对当前状态更全面的描述，这对决策工作至关重要。

$$S(t) = \{th(t), pf(t), rf(t), RI(t)\} \tag{3-11}$$

动作空间：智能体根据当前状态 $S(t)$ 为控制器传输过来的 SL 表或者过滤之后的请求列表做出最优的处理策略。因此智能 DHCP 请求动态处理算法问题中的行为是一个 $|I|$ 维向量，它可以定义为：

$$A(t) = \{a_1(t), ..., a_{|I|}(t)\} \tag{3-12}$$

其中 $a(t)$ 为二进制变量，当做出保留或分配 IP 给用户的动作时值为 0，否则为 1。$|I|$ 是根据由恶意 IP 定位器还是由恶意请求过滤器传输的数据而改变。它可以表示为：

$$|I| = |CMIS_{SLT}| \quad | \quad |FR| \tag{3-13}$$

其中 $|SLT|$ 是恶意 IP 定位器根据 SL 表传输的候选恶意 IP 集合的数量，$|FR|$ 是由恶意请求过滤器粗粒度过滤之后的请求的数量。

转换概率：这意味着网络在给定状态下执行动作后从一种状态转换到下一个状态的概率分布。在本文中，状态转移的概率是未知的，因为下一个状态的转移不仅取决于所选择的动作，同时取决于环境中的各种因素。

奖励：在 MDP 中，每个状态都伴随着一个奖励信号，这代表对该转化的智能体的评价。奖励可以立即获得，它可以定义为：

$$r(t) = \begin{cases} 1 - 2(T_r \oplus a_r(t)), & \text{succeed} \\ -\sigma, & \text{otherwise} \end{cases} \tag{3-14}$$

在等式 (3 – 14) 中，我们认为算法做出的处理是在满足公式 (3 – 5)-(3 – 6)。我们的目标是为了让算法能正确区分出正常请求与恶意请求，同时让算法不能将正常的请求当成恶意的请求而被处理。因此，在常规情况下，我们可以根据 GAN[47] 生成的请求对应的标签与智能体对该请求做出的动作形成对应的奖励。而且奖励的正方向反馈是根据 GAN 生成的请求对应的标签与智能体对该请求做出的动作是否相应，相应则做出正向奖励，反之则是负向奖励。此外，我们设置了惩罚 $-\sigma$，其中 σ 远大于常规情况下的正负向反馈奖励值。当算法做出的选择不能满足公式 (3 – 5)-(3 – 6) 以及当算法将正常的请求当成恶意的请求而被处理时终止智能体的决策并给予惩罚 $-\sigma$。

在强化学习中，智能体的目标是最大化长期积累回报，而不仅仅是即使回报。单个训练轮次的积累奖励可以定义为：

$$R_n = r_n(0) + \gamma r_n(1) + \gamma^2 r_n(2) + ... = \sum_{k=0}^{\infty} \gamma^k \cdot r(k) \qquad (3-15)$$

其中 γ 是折扣因子，k 是迭代次数。

（五）基于 GAN 环境下的深度学习智能 DHCP 请求动态处理算法

MDP 通常通过使用动态规划 (DP)[48] 或者深度强化学习 (DRL)[49] 来求解。但是，由于我们问题的场景具有较大的状态空间，这将导致维度灾难，要求系统需要存储和计算大量的状态和策略，而 DRL 可以通过神经网络在不记录大量状态对应的策略的情况下根据经验以及当前状态做出策略。另一方面，由于网络环境本身的复杂性导致该场景下的问题不能预先构建出完整的模型以解决问题，因此可以利用 DRL 的无模型性质来处理本文的问题。同时可以通过收集经验到经验区后进行经验回放的形式不断更新网络参数以更新 Q 值[50] 从而学习最优策略[51]，这更加适合本文中随机拜变化且错综复杂的网络场景。因此。我们选择 DRL 来解决问题。

如图3-4所示，我们利用部署具有 GAN 来生成相似攻击流的环境以及深度强化学习我们提出了基于 GAN[52] 环境和深度学习的智能 DHCP 请求动态处理算法来解决在粗粒度检测之后细粒度的细分正常流与恶意流并做出相应的策略。我们介绍了算法的数学模型以及详细的 Markov 处理过程，包括状态空间、动作空间、状态转移概率以及奖励机制。此外我们还详细介绍了 DDQN 的训练过程，包括智能体与环境交互以及经验存储与经验回放过程。接下来，我们来看看智能 DHCP 请求动态处理算法的具体细节。如算法3所示，第 1 至第 28 行描述了智能

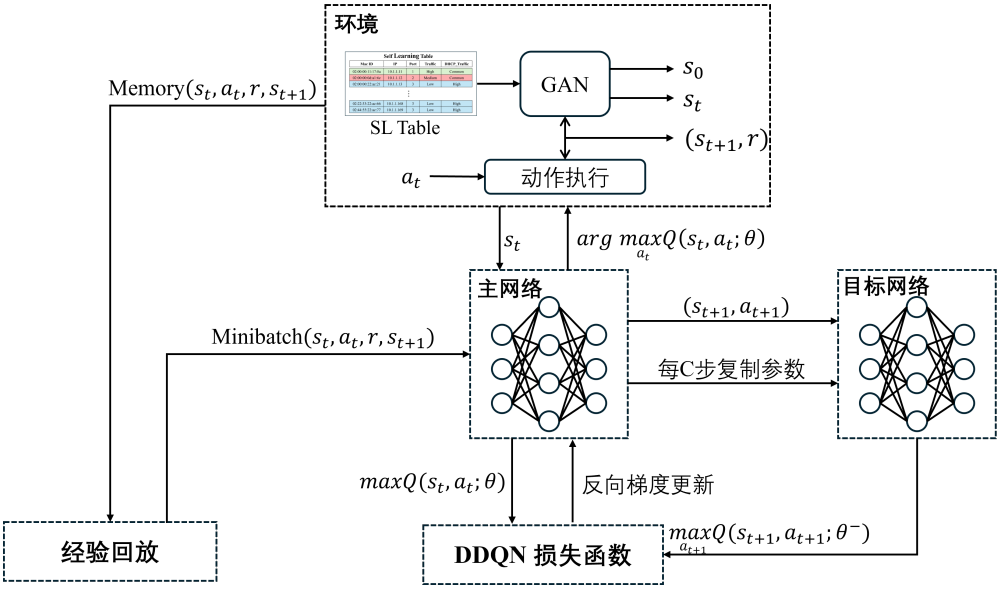


图 3-4　智能 DHCP 请求动态处理算法的训练过程

DHCP 请求动态处理算法的训练过程。在算法开始时，初始化经验池 E 以存储 (S_t, a_t, r_t, S_{t+1})(第 1 行)。同时，初始化主网络和目标网络的参数 θ 和 θ^- 以及结束标志 η。然后对智能 DHCP 请求动态处理算法进行 Episode 轮次的训练。对于每一个 episode，首先初始化状态，然后执行以下步骤：

1. 通过训练完成的 GAN 以及根据网络中情况获取正常请求或者恶意请求并打上标识 (第 8 行)。
2. 使用贪婪算法选择主网络中的动作 $action$(第 9 行)。
3. 根据上述两个步骤中获取的请求标识对以及 $action$ 在网络中执行并获取下一步状态 S_{t+1} 和奖励 $reward$(第 10 行)。
4. 判断下一步状态 S_{t+1} 中的主机数量是否超过网络中最大主机容量以及是否将正常请求的数据包丢弃。如果是，则将在原本奖励的基础做出惩罚。反之照常进行奖罚 (第 11-21 行)。
5. 将上述步骤依次得到的四元组 (S_t, a_t, r_t, S_{t+1}) 存储到经验池 E 中 (第 22 行)。
6. 当经验池累计到足够的经验后，从地址池中取出小批量数据训练主网络以及目标网络并使用梯度下降法更新主网络中的参数 (第 23 行)。
7. 判断当前 episode 是否到了更新目标网络的轮次。如果满足条件，则将主网络的参数复制到目标网络中。

训练完成之后保存训练的模型，将其部署到网络的控制器中已处理从粗粒度检测之后传输来的请求。

复杂度分析：设训练轮数为 E，每轮训练步数为 T，经验回放批量大小为 B，主网络一次前向传播的时间复杂度为 C_f，一次反向传播更新的时间复杂度为 C_b。在每个训练步中，智能体需要根据当前状态选择动作、执行动作并存储经验，当经验池满足训练条件后，还需要从经验池中采样并更新网络参数。因此，智能 DHCP 请求动态处理算法的总体训练时间复杂度可表示为 $O(ET(C_f + BC_b))$。若仅考虑模型部署后的在线决策过程，则每次请求处理主要涉及一次前向传播，时间复杂度为 $O(C_f)$。在空间复杂度方面，算法需要存储经验回放池、主网络参数和目标网络参数。设经验池容量为 M，单条经验状态维度为 d_s，动作维度为 d_a，神经网络参数量为 W，则空间复杂度为 $O(M(d_s + d_a) + W)$。

Algorithm 3 智能 **DHCP** 请求动态处理算法

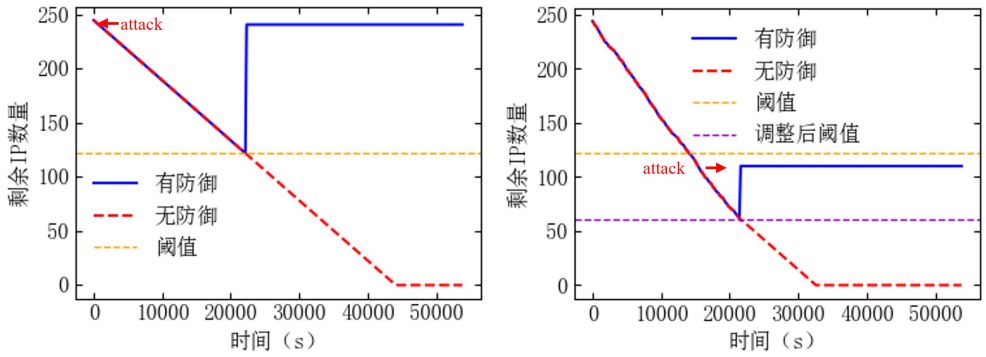
输入**:** 惩罚值 σ; 初始探索率 ϵ; 折扣因子 γ; 学习率 α; 目标网络更新频率 T; 网络环境 $Network$; 训练轮数 $Episode$; 训练步数 $Epoch$; 生成模型 GAN;

输出**:** DHCP 请求处理策略 π;

初始化经验回放池 E
初始化主网络参数 θ
初始化目标网络参数 $\theta^{-} \leftarrow \theta$
初始化结束标志 η
for $episode \leftarrow 1$ **to** $Episode$ **do**
  初始化状态空间 $state \leftarrow [th(0), pf(0), rf(0), RI(0)]$
  for $epoch \leftarrow 1$ **to** $Epoch$ **do**
    $(Request, Target) \leftarrow GetRequestByGAN(GAN, Network)$
    $action \leftarrow SelectAction(episode)$
    $(s_{t+1}, reward) \leftarrow ExecuteAction(Network, Request, Target, action)$
    if $action = 1$ **then**
      $f_{TN} \leftarrow IfMisJudged(Request, Target, action)$
      if $f_{TN} = 1$ **then**
        $\eta \leftarrow True$
      end if
    end if
    if $s_{t+1}.RI \geq Network.Pool.AllNum()$ **then**
      $\eta \leftarrow True$
    end if
    if $\eta = True$ **then**
      $reward \leftarrow reward - \sigma$
    end if
    StoreExperience(S_t, action, reward, S_{t+1})
    $\theta \leftarrow NetworkTrain(Minibatch(E))$
    if $episode$ mod $T = 0$ **then**
      updateTargetNetwork(θ)
    end if
  end for
end for
通过主网络得到 DHCP 请求处理策略 π
return π

3.3 DHCP 饥饿攻击防御框架评估

实验设置。基于 Mininet 和 POX 控制器构建一个虚拟的 SDN 测试环境来进行我们的实验。Mininet 版本为 2.3.1b4，POX 控制器版本 gar，该版本支持 python3。POX 控制运行在一个拥有 Intel 六核 CPU 以及 40G RAM 的主机上。为在 POX 控制器的应用层中部署 DHCP 饥饿攻击防御框架。其中有两台合法的用户机，用于进行流量活动，模拟真实环境中的情况。一台攻击主机，用于模拟非法用户，发起非法的攻击。此外，我们基于 Pytorch 完成对 IDRDRA 的训练。实验中，使用上述搭建的 SDN 测试环境构结合 GAN 构建 DHCP 请求作为 IDRDRA 模型中的环境，调节用户占比数量以满足实验要求。

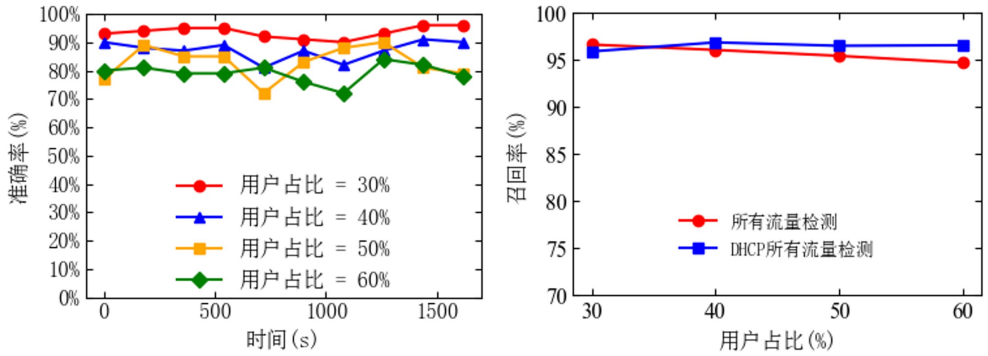


(a) 场景I　　(b) 场景II

图 3-5　**DHCP** 饥饿攻击防御框架对 **IP** 池的保护作用

对于 IP 池的影响。如图3-5所示。对于 DHCP 饥饿攻击防御框架对于 IP 池的防御效果，本文通过两个场景展示其防御效果。在场景 1 中恶意主机在 IP 池剩余 240 时就发起了攻击，随着 IP 池数量逐渐减少至阈值时触发检测系统，检测系统检测到该攻击并定位到恶意 IP 并释放。在恶意 IP 定位器定位到该恶意端口后，触发保护规则激活器使得后续攻击流被过滤无法发起对 DHCP 服务的攻击效果。在场景 2 中，恶意主机是在 IP 池剩余 170 的时候发起的攻击，因此在正常 DHCP 服务被获取至第一个阈值的位置时，恶意 IP 定位器没有检测到恶意 IP 以及恶意端口，所有动态调整了阈值。在调整阈值之后，恶意主机发起攻击并触发恶意 IP 定位器与保护规则激活的步骤如场景 1 一样。

恶意 IP 定位器的评估。该部分评估在没有智能 DHCP 请求动态处理算法情况下的恶意 IP 定位器性能。由于聚合流量统计的粒度会影响恶意流量定位的准确性，因此我们在不同用户数量下的网络进行实验。如图3-6(a) 中所示，依据总流量判断恶意 IP 的准确率随着网络中正常用户的数量的增加的下降。因为随着网络主机的增加，网络中的聚合流量变成更加错综复杂，同时大量的数据会导致混淆流量的丢包率的上升，这也将导致混淆流量产生变化，脱离原本设想的范围之内。这也就是为什么当网络中正常主机数量增加时会导致恶意 IP 定位器的准



(a)准确率　　(b) 召回率

图 3-6　恶意 **IP** 定位准确性评估

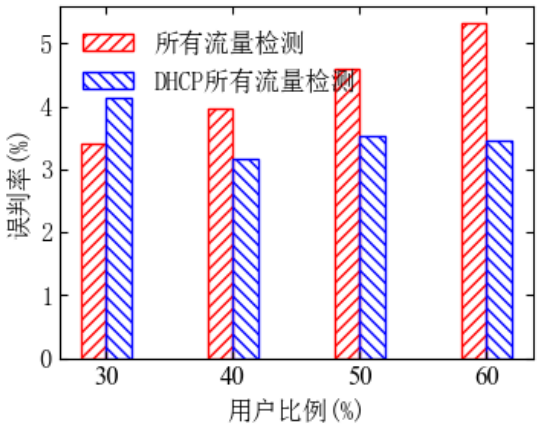


图 3-7　恶意 **IP** 定位器误判率

确率下降的原因。图3-6(b) 中总流量检测的召回率同样随着网络中正常主机数量增加而下降，这也说明当网络中主机变多时网络会带来很多不确定的因素。在没有智能 DHCP 请求动态处理算法的情况下，粗粒度的恶意 IP 定位器会因为检测条件的宽泛性导致一定程度的误判。由于网络主机增多会导致网络更加错综复杂，这在导致恶意 IP 定位器的准确率下降的同时也会导致对良性主机的误判。如图3-7所示，随着网络中良性主机数量的增加，利用 DHCP 数据流量判别的误判率会维持在一个稳定的水平，这是因为无论网络中主机的数量如何，DHCP 数据包的请求频率以及数量不会发送大幅度变化。但是总流量的误判率会不断升高，这是因为随着网络中主机数量的增加，网络中的流量水平将会发送大幅度变化，同时会产生更多种类的数据流量。这也会导致与混淆流量相同的流量水平的主机的概率会大大提升，这将导致误判率上升。综合网络中主机数量对于恶意 IP 定位器准确度和误判率的影响，可以发现当网络中的主机增大时会使得恶意 IP 定位器的性能产生影响。因此网络的规划对于恶意 IP 定位器的性能有很大的影响因素。

恶意请求过滤器的过滤效果。该部分评估在没有智能 DHCP 请求动态处理算法情况下恶意请求过滤器的效果，本文采用监听恶意主机所在端口的总流量

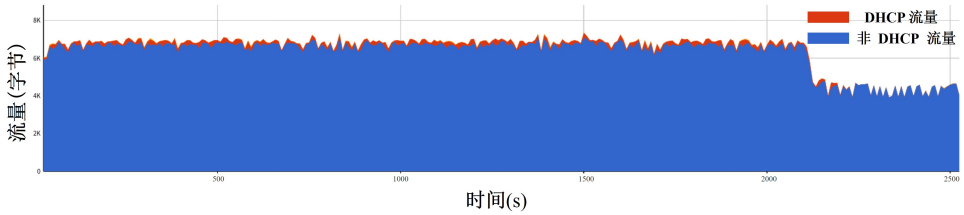


图 3-8　恶意请求过滤器的效果

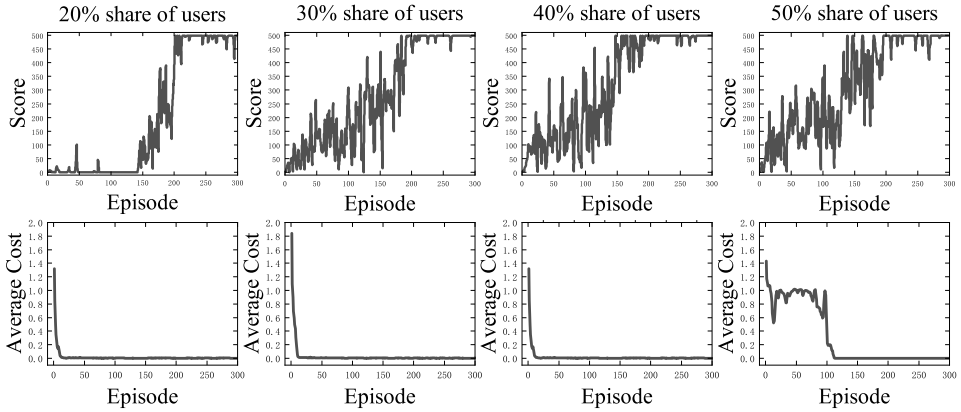


图 3-9　智能 **DHCP** 请求动态处理算法的训练结果

以及 DHCP 数据流量的吞吐量来评估其过滤效果。如图3-8所示，在图中可以看到，在过滤的保护规则被安装之后，总流量会突然发送一个下降，这表明有一部分的流量被过滤掉了。同时在保护规则被安装之后，少部分的 DHCP 数据包被过滤掉，可能是因为仍有一些较为不明显的攻击无法被过滤或者该端口的恶意主机本机的 IP 属于正常的获取的 DHCP 中的 IP，因此不会被检测和过滤。所有仍然有大量的 DHCP 流量存在。因此仍然需要智能 DHCP 请求动态处理算法做出更为细粒度的检测

智能 DHCP 请求动态处理算法的评估。如图3-9所示，该图上半部分以及下半部分依次为在用户数量占总体网络用户容量的 20%、30%、40% 以及 50% 的环境下每个 Episode 的的训练得分以及平均损失。从图中可以看出，用户占比在 20% 时，由于正常的用户请求数量远低于恶意请求数量，在这种情况下深度神经网络会较容易做出将数据包丢弃的决策。因此在前 150 轮训练中可以看出，由于容易丢弃正常的请求而被智能 DHCP 请求动态处理算法而判断失败而终止，因此得分几乎为 0. 随着用户数量占比的增加，在后续三种情况中可以看出，随着用户占比的增加前一半的 episode 的得分越来越高并且越来越稳定。同时，分析后 150 个 episode 可以看出，随着用户占比的增加，模型的收敛时间越来越快。可以看出在用户占比为 20%、30%、40% 以及 50% 时模型分别在第 200、180、150 以及 140 个 episode 的时候完成了收敛。这是因为当网络中正常请求与恶意请求的数量越来越接近时，深度神经网络更容易学习到更多不同的策略并且做出正常的动作。同时，我们可以看到当用户占比为 20%、30% 和 40% 时，每个 Episode 的的平均损失都呈现出线性的下降趋势，可以见在前几个轮次中，目标

网络与主网络就已经收敛完成。而当用户占比为 50%，在前 100 个 episode 主网络和目标网络的损失仍然很大没有收敛，并且从每个 Episode 的的训练得分我们可以看出其变化形势不同于前三种情况，呈现了一个先增高在下降的趋势。这种不稳定的变化也是由于在用户占比为 50%，正常用户的请求更加复杂多样同时网络的情况也更加错综复杂。虽然正常请求与恶意请求数量相当时会导致模型前期的训练不稳定，但是也能让模型更好的学习到各种不同的情况，相应的会导致模型更快的收敛。

对比现有防御机制。为了评估我们的 DHCP 饥饿攻击防御框架在 DHCP 饥饿攻击中的改进与性能优势，我们将其与以下四种基准防御架构进行对比：

- P-DHCP[13]：一种结合客户认证与消息速率检测的 DHCP 攻击防御机制。P-DHCP 利用客户端与服务器之间的通信确保 DHCP 请求的完整性和真实性。
- DHCPGuard[14]：一种结合网络嗅探记录与限制速率检测的 DHCP 攻击防御机制。在对 DHCP 请求做出分析检测的同时，DHCPGuard 还增加了 IP 地址池恢复功能来维持 DHCP 服务持续运行。
- DHCPWatcher[15]：一种多阶段的检测和缓解机制。DHCPWatcher 使用前面检测、验证检测和行为检测等多种机制来识别恶意 DHCP 恶意流量。
- K-Means[21]：一种基于机器学习的异常检测架构，作者采用了多种机器学习进行实验。在本文中，我们选择其中最好的 K-Means 方法进行实验对比。

对比指标，我们采用各个防御机制对请求的检测结果的准确度进行对比。其中，智能 DHCP 请求动态处理算法的最终目标是维持 DHCP 服务长时间运行的长期收益，这与其他机制着重针对接收请求的检测结果的即使收益不同。所以对于智能 DHCP 请求动态处理算法的准确度，我们采用在受攻击情况下 DHCP 服务运行足够久的时间就认定其获得了维持 DHCP 服务长时间运行的长期收益。

首先，我们在不同用户占比的网络中，利用五种不同的防御机制进行实验。如图3-10所示，DHCP 饥饿攻击防御框架在所用场景中的效果均优于其他防御机制，而 DHCPGuard 次之，其他三种防御机制的效果较差。从图中我们可以看出，P-DHCP、DHCPWatcher 以及 K-Means 三种防御机制随着网络中用户数量占比的增长性能逐渐下降，很大概率是因为当用户数量占比增加之后网络中的数据包更加复杂多样，同时恶意请求数量占比下降，防御机制不能做出更加准确的判断。而 DHCP 饥饿攻击防御框架和 DHCPGuard 的检测准确率随着用户数量占比的提升都呈现了上升的趋势，这也能反映出两种防御机制更加适合错综复杂的网络场景。而不同的是 DHCP 饥饿攻击防御框架相比于 DHCPGuard 的变化趋势更加平缓，甚至在大占比的时候呈现出了平稳的状态，这也能看出 DHCP 饥饿攻击防御框架相比于 DHCPGuard 具有更强大的稳定性。总体而言，DHCP 饥饿攻击防御框架均优于现存的 DHCP 饥饿攻击防御机制，相比于现存的最好的防

御机制仍然高出 20%。

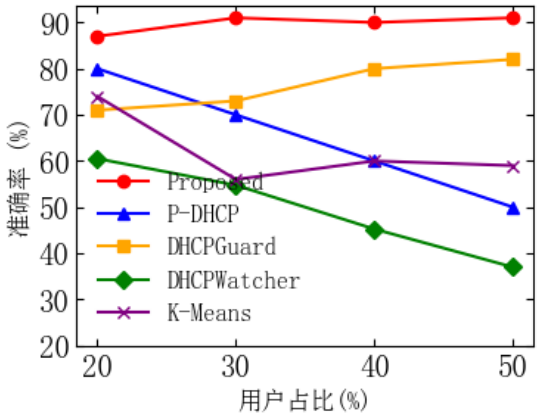
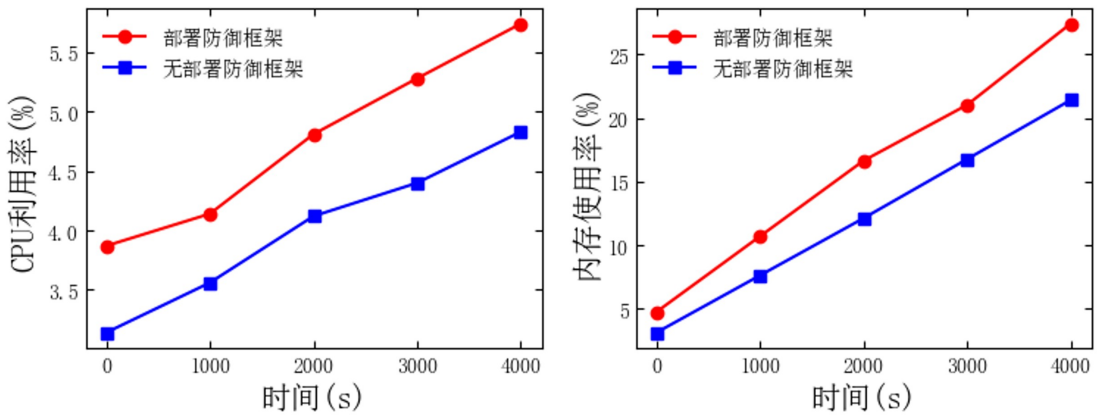


图 3-10　与现有工作的检测精度对比

系统开销。我们评估了我们所提出框架的平均 CPU 使用率。平均 CPU 使用率是从攻击开始到识别攻击结束这段时间内计算的。如图3-11(a) 所示，DHCP 饥饿攻击防御框架随着运行时间的增长略微增加了 CPU 使用率。结果是合理的，因为运行时间越久，交换机和控制器的 CPU 将处理更多的自学习表以及保护表。一般来说，控制器中的 CPU 使用率不到 5%，在引入 DHCP 饥饿攻击之后 CPU 使用率增长到 5.72%，这表明增加 14% 左右的开销。如图3-11(b) 所示，DHCP 饥饿攻击防御框架随着运行时间的增长略微增加了 RAM 使用率，这同 CPU 一样。因为随着时间运行，交换机和控制器将存储更多的自学习表以及保护表的内容以及其他的信息，所有会导致更多的 RAM 的使用。一般来说，控制器中的 RAM 使用率不到 20%，在引入 DHCP 饥饿攻击之后 CPU 使用率增长到 27.4%，这表明增加 30% 左右的开销。结果表明，DHCP 饥饿攻击防御框架通过引入较小的开销来抵抗 DHCP 饥饿攻击攻击。



(a) CPU　　(b) RAM

图 3-11　**DHCP** 饥饿攻击防御框架开销

进一步分析上述实验结果可以发现，本文所提出的 DHCP 饥饿攻击防御框架的优势并不单纯体现在检测准确率的提升上，而是体现在攻击识别、资源恢复和后续阻断三个环节的协同作用。图 3-5 表明，当 IP 地址池数量下降至触发阈值后，恶意 IP 定位器能够及时启动检测过程，并在识别恶意 IP 后释放被异常占用的地址资源，使 IP 池数量恢复到相对稳定状态。这说明本文方法不是仅对单个 DHCP 请求进行静态判别，而是将地址池剩余量、主机流量行为和端口状态纳入统一分析框架，从而能够从服务持续运行的角度缓解地址池耗尽问题。

从图 3-6 和图 3-7 可以进一步看出，恶意 IP 定位器的性能受到网络中正常用户数量的影响。当正常主机数量增加时，网络总体流量更加复杂，部分正常流量可能在总流量规模上接近攻击混淆流量，从而导致基于总流量的判别准确率下降、误判率上升。相比之下，DHCP 数据流量的误判率变化较小，说明 DHCP 请求行为本身具有更稳定的协议特征。因此，单纯依赖总流量特征容易受到网络规模和业务复杂度影响，而将总流量特征与 DHCP 流量特征结合，可以在一定程度上提高检测的稳定性。这也解释了本文采用“恶意 IP 定位器 + 恶意请求过滤器 + 智能处理算法”多阶段结构的必要性：前两个模块用于快速缩小可疑范围，后续强化学习模块则用于降低粗粒度检测带来的误判风险。

图 3-8 显示，保护规则安装后恶意端口的总流量和 DHCP 流量均出现下降，说明恶意请求过滤器能够对后续攻击请求形成持续抑制。然而，过滤后仍然存在少量 DHCP 流量，表明仅依靠端口级过滤难以完全区分攻击请求和正常请求，特别是在恶意端口上混合存在合法主机请求或攻击流量伪装程度较高时，固定过滤规则可能出现漏判或误判。因此，本文进一步引入智能 DHCP 请求动态处理算法，对过滤后的请求进行细粒度判别。该结果说明，多阶段防御并非简单叠加多个模块，而是形成了由粗到细的处理链条：先通过全局相关性分析定位异常范围，再通过保护规则减少攻击流量，最后通过强化学习完成动态决策。

从图 3-9 的训练结果可以看出，智能 DHCP 请求动态处理算法在不同用户占比下均能够逐步收敛，但收敛速度和前期波动程度存在差异。当正常用户占比较低时，恶意请求在训练样本中占主导，模型容易形成偏向丢弃请求的策略，因此前期得分较低；随着正常用户占比提高，模型能够接触到更多正常请求与恶意请求并存的状态，策略学习更加充分，收敛速度随之提升。当用户占比达到 50% 时，训练前期波动较大，说明复杂环境会增加策略探索难度，但后期收敛速度较快，表明多样化样本有助于模型学习更稳健的处理策略。这一现象说明，强化学习模型的训练效果与网络环境复杂度密切相关，在实际部署中应尽可能覆盖不同用户规模和攻击强度下的训练场景，以提高模型泛化能力。

图 3-10 的对比结果表明，本文方法在不同用户占比场景下均优于 P-DHCP、DHCPGuard、DHCPWatcher 和 K-Means 等方法。其原因在于，P-DHCP 和 DHCP-Watcher 主要依赖认证、规则验证或多阶段检测机制，对攻击行为变化的适应性有限；K-Means 虽然引入机器学习方法，但其检测效果依赖训练数据分布，当网

络环境变化时容易出现性能下降；DHCPGuard 具有地址池恢复能力，但缺乏面向复杂流量状态的智能决策机制。相比之下，本文方法同时利用 SDN 全局视角、触发式检测、保护规则过滤和 DDQN 动态决策，使得防御过程能够根据网络状态变化自适应调整。结合图 3-11 的开销结果可以看出，本文框架在提高防御效果的同时仅带来有限的 CPU 和内存开销，说明该方法在检测性能与系统资源消耗之间具有较好的平衡。

3.4 本章小结

本章针对 SDN 环境下 DHCP 饥饿攻击可能导致地址池资源耗尽、合法用户无法正常接入等问题，设计并验证了一种面向协议层洪泛攻击的多阶段协同防御框架。首先，基于 SDN 控制器的全局可视能力，构建了由恶意 IP 定位器、恶意请求过滤器和智能 DHCP 请求动态处理算法组成的系统架构，将传统单一请求检测问题扩展为面向全局网络状态的动态防御问题。其次，恶意 IP 定位器通过自学习表记录主机 IP、端口、总流量和 DHCP 流量等信息，并结合流量相关性分析定位疑似恶意 IP 和恶意端口；恶意请求过滤器则根据恶意端口安装保护规则，对后续异常 DHCP 请求进行初步过滤，从而降低攻击流量对 DHCP 服务的持续影响。在此基础上，本文进一步将粗粒度检测后的请求处理建模为马尔可夫决策过程，引入基于 DDQN 的智能 DHCP 请求动态处理算法，对正常请求与恶意请求进行细粒度判别与动态决策。

实验结果表明，本文提出的 DHCP 饥饿攻击防御框架能够在不同攻击场景下有效保护 IP 地址池资源，及时释放被恶意占用的地址，并阻断后续攻击请求。与 P-DHCP、DHCPGuard、DHCPWatcher 和 K-Means 等基准方法相比，本文方法在检测效果和稳定性方面具有更优表现，同时仅引入较低的 CPU 和内存开销。由此说明，基于 SDN 全局视角、粗粒度过滤与强化学习细粒度决策相结合的协议层防御机制，能够在保证 DHCP 服务持续运行的同时提升系统对洪泛攻击的自适应防护能力，也为第四章进一步研究数据层 DDoS 攻击检测与动态缓解提供了方法基础。

第四章　数据控制层攻击的检测与缓解

4.1　引言

在 SDN 架构中，控制平面与数据平面分离使网络具备集中管理和灵活调度能力，也使控制器成为网络运行中的关键节点。控制器能够从全局角度感知网络状态，并根据流量变化动态下发控制策略，这为 DDoS 攻击检测和缓解提供了有利条件。然而，控制器的集中化特性也带来了新的安全风险。一旦大量异常流量触发频繁的控制交互，控制器可能因处理压力过大而出现响应延迟，严重时甚至影响正常业务流量的调度。因此，在 SDN 环境下，DDoS 攻击防御不能只关注数据平面的异常流量过滤，还需要考虑控制器资源消耗、控制信息传输能力以及防御动作对网络稳定性的影响。

在 IPv4 向 IPv6 长期过渡的背景下，IPv4 与 IPv6 双栈共存将持续存在。双栈网络虽然能够兼顾现有 IPv4 业务和未来 IPv6 网络发展需求，但也使网络流量表现出更加复杂的协议异构特征。IPv4 与 IPv6 在地址长度、报文结构、扩展机制、字段含义和流量分布等方面存在明显差异，导致两类协议流量难以直接采用完全相同的特征表示方式进行建模。传统面向单一协议栈设计的检测方法通常只能针对 IPv4 或 IPv6 中的一类流量进行分析，当其直接应用于双栈环境时，容易出现特征维度不一致、流量表示不统一和检测结果不稳定等问题。因此，如何消除 IPv4 与 IPv6 之间的特征差异，构建统一的双栈流量表达方式，是双栈 SDN 环境下开展 DDoS 攻击检测与缓解的基础问题。

现有 DDoS 攻击检测与缓解方法主要包括流量统计分析、信息熵检测、机器学习分类和强化学习动态决策等类型，已在异常流量识别和攻击缓解方面取得一定效果，但在双栈 SDN 环境中仍存在局限。首先，部分方法主要面向单一协议环境，缺乏对 IPv4/IPv6 共存场景的统一建模能力。其次，传统检测方法多依赖单一流量字段或局部统计特征，难以全面反映攻击对控制器资源、链路带宽、网络时延和控制信息传输能力等系统状态的综合影响。再次，部分动态防御方法对动作代价、策略切换频率和网络稳定性考虑不足，可能造成过度防御或服务质量下降。因此，DDoS 攻击缓解需要在攻击抑制效果与网络稳定运行之间取得平衡。

针对上述问题，本章提出融合双栈数据处理、多维信息熵检测和深度强化学习的 DDoS 攻击动态防御方法。首先，设计双栈数据处理机制，对 IPv4 与 IPv6 流量进行特征提取、数值化转换和统一降维，使不同协议流量能够形成一致的特征表示。其次，构建包含请求数量、控制器 CPU 使用率、全网带宽、网络时延和控制信息传输能力的多维信息熵体系，从系统整体状态角度刻画 DDoS 攻击造成的异常变化。最后，将攻击缓解过程建模为马尔可夫决策过程，利用 DDQN

根据熵检测结果和网络状态反馈动态生成防御策略，并在奖励函数中引入动作代价约束，使模型在抑制攻击流量的同时兼顾网络稳定性、资源消耗和服务质量。

4.2 基于强化学习的 DDoS 攻击监测

4.2.1 双栈数据处理

在双协议栈网络环境中，IPv4 与 IPv6 数据包共存，两者在协议格式上存在明显差异。为了在分析中保持特征维度的一致性，同时避免影响信息熵的计算精度，我们对 IPv4 和 IPv6 数据包分别进行特征提取与归一化映射。这样做的目的是，在不丢失关键信息的前提下，使得两种协议的数据流量在后续检测阶段能够统一表达，从而提升 DDoS 攻击检测与缓解算法在双栈环境下的适应性和准确性。

（一）特征处理概述

传统网络中，要获取交换机上的数据流信息，往往需要配置交换机或接入额外设备。这些操作有时会增加交换机的处理负担，对网络性能造成潜在影响。而在软件定义网络环境下，控制器的集中管理机制结合 OpenFlow 南向接口协议，使得管理员可以从全局视角动态管理流量，通过控制器直接与交换机交互，相对高效地获取数据包信息。

在双栈环境中，IPv4 与 IPv6 数据包共存带来了协议异构性问题。两者的报头结构、地址空间、扩展机制存在本质差异，导致提取出的特征在内容和维度上都不一致。这种不匹配使得构建统一的流量分析模型变得困难。因此，如何有效消除 IPv4 与 IPv6 之间的特征差异，实现特征空间的一致映射，是双栈网络流量统一建模中一个需要解决的问题。

如图4-1所示，IPv4 与 IPv6 协议的特征在内容和维度上都有所不同。针对这一问题，本研究在双栈协议特征处理器中首先将两类特征按语义分类，确保特征含义对齐。然后采用数据摘要迭代压缩技术对特征进行处理，在保持原始信息熵的前提下，将其转化为数字化表示，以适配强化学习算法的输入格式。最后，对于同类但维度不一致的特征，通过降维方法进行统一化处理，实现 IPv4 与 IPv6 特征在数量和维度上的对齐。

（二）特征唯一数字化

在计算机网络环境中，许多特征并非以数值形式呈现，无法直接应用于强化学习模型[53]。因此，非数值特征的数据转换成为必要步骤。如图4-2所示，我们采用双栈特征处理器来实现特征的唯一数值化及同类不同维度的降维优化[54]。Daru 等人[55]研究并验证了利用 SHA-1[56]算法将任意数据类型转换为唯一数值表示的可行性。在此基础上，我们选择哈希长度更短的 MD5[57]算法，以在后续

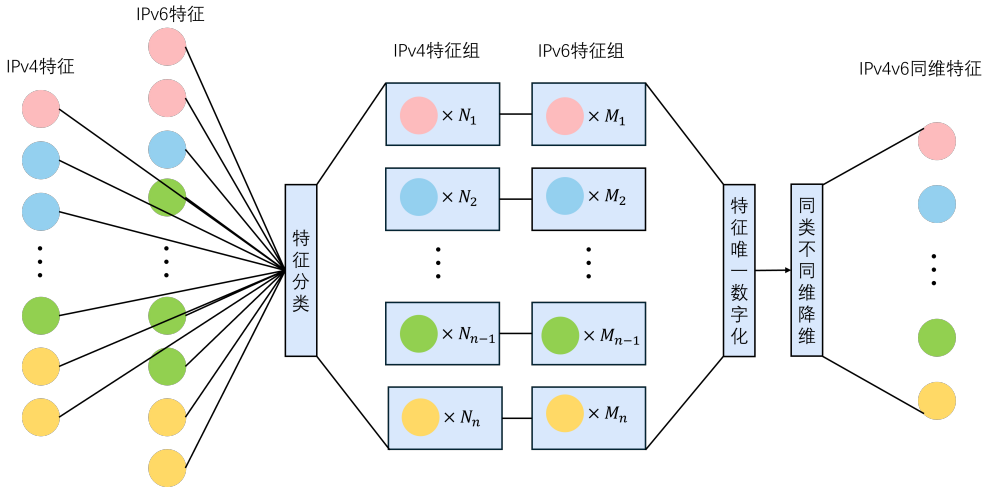


图 4-1　双栈特征处理流程

特征处理过程中尽可能保留更多原始信息。由于哈希转换结果通常以十六进制表示，为适配深度学习与强化学习模型，我们需将其进一步转换为十进制数据。尽管 MD5 相较于 SHA-1 更为紧凑，但其仍具有固定的 128 位长度。为进一步压缩特征表示，同时保持信息的有效性，我们采用 CRC32[58]算法对 MD5 哈希值进行提炼，将其转换为 32 位的数值表示，从而使其能够高效用于深度学习和强化学习任务。

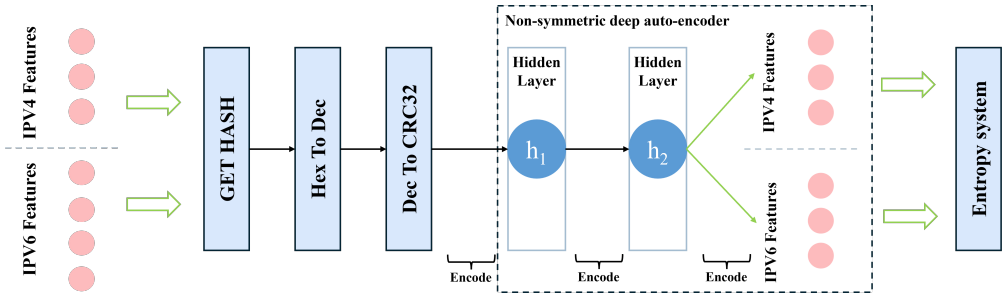


图 4-2　一个唯一数字化和降维过程的例子

（三）非对称深度自编码器

非对称深度自编码器（Non-symmetric Deep Auto-Encoder，NDAE）[59] 通过其编码器部分实现对数据的降维处理。NDAE 通过多个隐藏层逐步将高维输入数据映射到低维潜在表示空间，从而实现特征压缩与降维。NDAE 通常包含多个隐藏层，且每一隐藏层中的神经元数量通常少于前一层，从而实现逐层递减的维度压缩。

NDAE 的输入为高维向量 $x \in \mathbb{R}^d$，并通过式（1）所示的确定性函数逐步映射为低维潜在表示 $h_i \in \mathbb{R}^{d_i}$：

$$h_i = \sigma(W_i \cdot h_{i-1} + b_i), \quad i = 1, 2, \cdots, n \tag{4-1}$$

其中，$h_0 = x$，$\sigma(\cdot)$ 表示非线性激活函数，n 表示隐藏层的层数。

NDAE 的非对称性体现在其仅保留编码器部分，而省略了解码器结构。其输出向量可由式（2）表示：

$$y = \sigma(W_{n+1} \cdot h_n + b_{n+1}) \tag{4-2}$$

其中，W_{n+1} 和 b_{n+1} 分别表示输出层的权重矩阵和偏置向量。

NDAE 采用无监督学习方式进行训练，其目标是最小化输入数据 x 与输出数据 y 之间的重构误差。通过最小化重构误差，NDAE 能够学习输入数据中最具代表性的特征，并将其压缩至低维空间。重构误差可定义为：

$$E(\theta) = \sum_{i=1}^{m} \left(x^{(i)} - y^{(i)}\right)^2 \tag{4-3}$$

NDAE 的降维机制依赖于隐藏层神经元数量的逐层递减、非线性激活函数以及无监督学习策略，使其能够在保留输入数据主要特征的同时，高效地将高维数据压缩至低维空间。在网络入侵检测场景中，NDAE 的降维机制能够显著提升特征提取和数据压缩的效率，从而提高检测精度和整体性能。

Algorithm 4 双栈数据处理流程

输入：数据包流 s_k;

输出：双栈数据 D_d;

```
V4Packet, V6Packet ← PacketClassification(s_k)
V4Feature[], V6Feature[] ← FeatureExtraction(V4Packet, V6Packet)
V4FeatureGroup[], V6FeatureGroup[] ← FeatureClassification(V4Feature[], V6Feature[])
MD5(V4FeatureGroup[], V6FeatureGroup[])
HexToDec(V4FeatureGroup[], V6FeatureGroup[])
CRC32(V4FeatureGroup[], V6FeatureGroup[])
for (V4FeatureGroup, V6FeatureGroup) in (V4FeatureGroup[], V6FeatureGroup[]) do
    dimension ← min(shape(V4FeatureGroup, V6FeatureGroup))
    NDAE(V4FeatureGroup, V6FeatureGroup, dimension)
end for
D_d ← MergeData(V4FeatureGroup[], V6FeatureGroup[])
return D_d
```

复杂度分析：设输入的数据包数量为 N，单个数据包提取的平均特征数量为 f，NDAE 降维模型一次前向处理的复杂度为 C_e。双栈数据处理流程首先需要对数据包进行分类和特征提取，该部分时间复杂度为 $O(Nf)$；随后对非数值

特征进行 MD5、十进制转换和 CRC32 处理，由于这些操作对每个特征执行一次，其时间复杂度仍为 $O(Nf)$；最后对同类特征进行 NDAE 降维，时间复杂度为 $O(NC_e)$。因此，算法整体时间复杂度可表示为 $O(N(f+C_e))$。在空间复杂度方面，算法需要存储 IPv4 与 IPv6 特征组、中间哈希结果以及降维后的统一特征表示，空间复杂度为 $O(Nf)$。

4.2.2 熵体系

DDoS 攻击具有明显的时间特性，其持续的大规模恶意流量会在一段时间内对网络系统产生可观测的影响。基于这一特性，本文提出了一种基于信息熵理论的 DDoS 攻击检测方法。通过构建针对关键网络状态参数的信息熵系统，对受攻击影响的网络状态参数的信息熵值在时间序列上的演化过程进行系统分析。通过研究不同时间尺度下信息熵的波动特性，实现对 DDoS 攻击发生的准确检测。

本文构建的信息熵分析系统，和传统基于数据流特征的检测方法不太一样——我们更关注网络系统自身特性的变化。具体来说，就是从系统内部的状态特征入手，挖掘并量化其动态变化。这样做的好处有两个：一是能更准确地捕捉网络在遭受攻击时的异常行为，二是泛化能力相对更强，对于未知的新型攻击也具备一定的检测能力。

围绕系统动态变化这一思路，我们设计了五个网络层面的信息熵指标，分别是：网络请求总量、控制器 CPU 利用率、全网带宽、全网时延，以及 SDN 控制信息的传输能力。这五个指标基本覆盖了网络运行状态的主要方面，能为 DDoS 攻击检测提供相对全面、可靠的分析基础。

其中，网络请求总量可以直接通过统计数据流的数量获得。控制器 CPU 利用率和全网带宽，则可以通过在控制器中进行定制化编程，再借助南向接口把数据传到应用层来获取。至于全网时延，需要测量数据流在整个持续时间内，数据包的总传输时间来计算，具体定义如下：

$$N_D = \frac{\left(\sum_{i=1}^{N} \frac{FD_i}{FP_i+BP_i}\right)}{N} \tag{4-4}$$

其中，N_D 表示全网时延，FD_i 表示第 i 条数据流的持续时间，FP_i 表示第 i 条数据流的前向数据包数量，BP_i 表示第 i 条数据流的反向数据包数量。

在实际网络中，SDN 控制信息的传输能力很容易被链路拥塞“拖累”。特别是在 DDoS 攻击期间，数据平面链路往往会出现严重拥塞，控制器没法及时把关键控制命令下发给交换机，这样一来，网络系统的自适应防御能力就会打折扣，攻击带来的影响也会被进一步放大[60]。所以说，保证控制信息的高效传输，是做好 DDoS 缓解工作的前提。也正因如此，我们把 SDN 控制信息传输能力作为一个关键检测指标——这样可以在控制命令真正被阻塞之前，就察觉到潜在威

胁，提前采取防御措施。

不过，我们没法直接根据某个特征就断定一条数据流是不是控制流。所以只能换个思路，通过分析控制器发送数据流的标志位、传输速率、包大小这些信息，来间接评估控制信息的传输效率。基于此，SDN 控制信息传输能力可以定义为：

$$C_C = \frac{\sum_{i=1}^{N} [\alpha_1 (F_{fin} + F_{SYN} + F_{RST} + F_{ACK})_i + \alpha_2 IAT_i + \alpha_3 PL_i]}{N_C} \tag{4-5}$$

其中，F_{fin}、F_{SYN}、F_{RST} 和 F_{ACK} 分别表示 FIN、SYN、RST 和 ACK 标志位的数量，IAT 表示数据包到达间隔时间，PL 表示控制器发送的数据包数量。

为便于分析与比较，需要根据各指标对系统运行状态影响程度的不同，对五个指标进行加权求和。SDN 控制信息传输能力是攻击缓解的基础，必须确保在攻击发生初期控制器仍具备及时下发控制信息的能力。同时，控制器 CPU 利用率、全网带宽和全网时延在 DDoS 攻击发生时均会出现显著变化。当攻击发起时，网络流量带宽会在初始阶段迅速增加；随着带宽逐渐趋于拥塞，全网时延随之升高；相应的网络拥塞将促使控制器频繁调整路由策略并加快流表下发过程，从而导致控制器 CPU 利用率上升。

攻击检测的首要目标是尽可能早地识别攻击行为，因此指标变化的先后顺序也是衡量其影响程度的重要因素。尽管网络请求总量在攻击初期能够被快速检测，但网络中往往存在正常的突发流量行为，单独使用该指标容易产生较高的误报率。然而，该指标具有较强的实时性，仍具有重要参考价值。通过赋予其较小权重并提高检测阈值，可以有效降低由小规模突发流量引起的误报。综合上述因素，系统总体信息熵定义如下：

$$E_S = \beta \cdot [N_R, U_C, N_B, N_D, C_C]^\intercal \tag{4-6}$$

其中，β 为权重向量，N_R 表示网络请求总量，U_C 表示控制器 CPU 利用率，N_B 表示全网带宽，N_D 表示全网时延，C_C 表示 SDN 控制信息传输能力。

4.2.3 DDoS 攻击检测

我们做 DDoS 攻击检测，核心是信息熵理论。基本思路是：先长期累积熵数据，再看这些数据的分布特征有没有出现明显偏离，据此判断网络系统是不是正在遭受攻击。整个检测过程主要分成两个阶段：一是攻击检测阶段，二是攻击发生之后的实时响应部署阶段。

在攻击检测阶段，我们设计了一个基于信息熵的分析框架，并引入了 Z-score 方法。这个方法的用处是衡量当前系统的某个熵指标，相对于过去一段时间的历史数据，到底偏离了多少。通过这个偏离程度，就能评估系统熵的异常情况，进

而判断是否发生了 DDoS 攻击。Z-score 的具体定义如下：

$$Z^{(T_n)} = \frac{E_S^{(T_n)} - \mu^{(T-T_n)}}{\sigma^{(T-T_n)}} \tag{4-7}$$

其中，$Z^{(T_n)}$ 表示给定时间点的系统熵 Z-score 值，$E_S^{(T_n)}$ 表示该时刻的系统总体信息熵，$\mu^{(T-T_n)}$ 表示固定时间窗口内历史系统熵的均值，$\sigma^{(T-T_n)}$ 表示对应的标准差。

我们在检测中引入了一个滑动窗口机制。做法不算复杂：把网络流表项按时间顺序依次放入一个固定大小的窗口里，窗口长度设成了 5000 条流。然后每隔一秒，就重新算一下当前窗口内的 Z-score。这么设计的好处是，既保证了统计上的稳定性，又能兼顾实时性——短时的流量波动也能被抓到，在检测时效和鲁棒性之间找到了一个还算不错的平衡点。

至于攻击发生之后的实时响应部署阶段，我们把模型训练过程和系统的实际运行过程拆开了，互不干扰，目的就是防止模型训练影响到正常网络资源的调度。因为整个检测框架跑在应用层，模型训练不会占用 SDN 的计算或传输资源，只是在检测完成之后，需要把对应模型快速部署上去。 为了提升检测与响应的

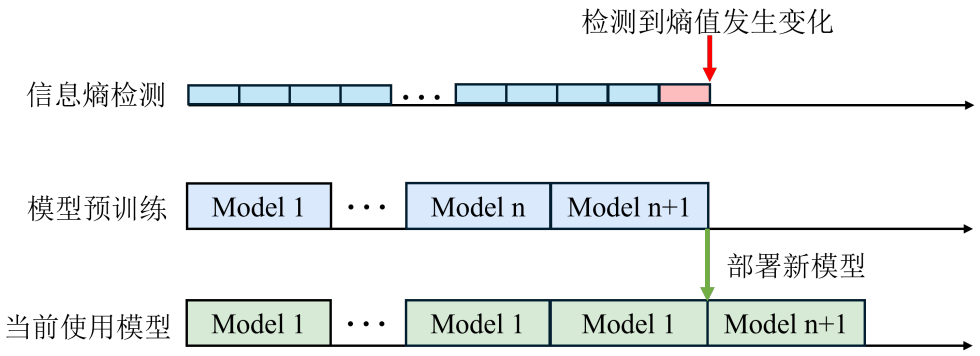


图 4-3　实时响应部署方法常规情况

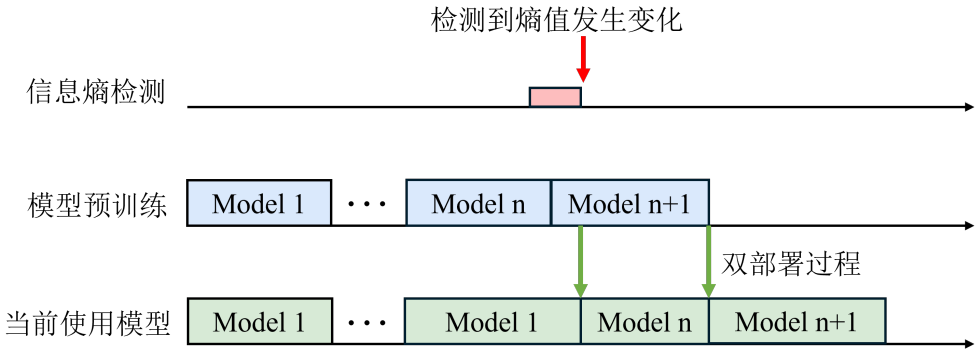


图 4-4　实时响应部署方法双重部署情况

实时性，同时解决模型训练时间较长带来的部署滞后问题，我们设计了一种新的

检测与部署协同机制。如图4-3所示，理想情况下，检测模块发现 DDoS 攻击的时候，新模型已经训练好了，可以直接部署上去，快速缓解攻击。但实际场景往往没那么顺利，如图4-4所示，很多时候攻击已经被检测到了，当前模型却还没训练完。这时候我们的做法是：先临时部署上一轮已经训练好的模型，维持住防护能力，等新模型训练完成后再替换上去。这个机制能有效缓解训练延迟造成的部署滞后问题，同时也能保证模型持续更新，跟上不断变化的攻击模式。

如图4-3所示，我们这套框架在实时响应能力和模型自适应能力之间做了一个折中，即便模型训练和攻击检测不同步，系统也能先拿历史模型顶上，维持持续防御，等训练完成再更新到最新模型。这样一来，防御能力不会断档，框架在面对动态演化的攻击场景时也更有韧性。

Algorithm 5 DDoS 攻击检测流程

输入**:** 双栈数据 D_d，系统熵列表 L_E，权重参数 β，Z-score 阈值 Z_h，当前时间 T_C，最新模型训练完成时间 T_L，时间偏移阈值 T_o;

输出**:** 攻击标记 A_t;

1: $A_t \leftarrow 0$
2: $[N_R, U_C, N_B, N_D, C_C] \leftarrow$ GetEntropyIndex(D_d)
3: $E_C \leftarrow \beta \cdot [N_R, U_C, N_B, N_D, C_C]$
4: $\mu \leftarrow$ mean(L_E)
5: $\sigma \leftarrow$ std(L_E)
6: $Z_S \leftarrow \frac{E_C - \mu}{\sigma}$
7: **if** $Z_S > Z_h$ **then**
8: $\quad A_t \leftarrow 1$
9: $\quad$ **if** $T_C - T_L \leq T_o$ **then**
10: $\quad\quad$ 初始部署模型 ()
11: $\quad$ **else**
12: $\quad\quad$ 二次部署模型 ()
13: $\quad$ **end if**
14: **end if**
15: **return** A_t

如算法 5 所示，该算法给出了 DDoS 攻击检测与模型部署的整体流程。第 2 步从双栈数据处理模块输出的数据包中提取熵指标，构建系统信息熵体系；第 3 步利用所获得的熵指标计算当前系统总体信息熵。第 4 至第 6 步基于历史系统熵列表计算当前系统熵的 Z-score 值。第 7 至第 14 步根据 Z-score 判断网络系统是否遭受攻击：若检测到攻击，则将攻击标记设为 1，并根据当前时间与最新模

型训练完成时间之间的差值，判断是否执行单模型部署或双阶段部署策略。

复杂度分析：设滑动窗口长度为 L，熵指标数量为 k。DDoS 攻击检测流程需要从双栈数据中提取请求数量、CPU 使用率、带宽、网络时延和控制信息传输能力等熵指标，并计算系统总体信息熵。由于本文使用的熵指标数量固定，因此 $k=5$。在每次检测过程中，算法需要对滑动窗口内的数据进行统计，并计算均值、标准差和 Z-score 值，时间复杂度为 $O(Lk)$。由于 k 为常数，整体可简化为 $O(L)$。在空间复杂度方面，算法主要维护历史系统熵列表和当前滑动窗口数据，空间复杂度为 $O(L)$。

4.2.4 仿真设计结果及分析

如图 4-5 所示，系统中的攻击流量主要集中分布在时间轴的中间区域，而信息熵检测系统的核心目标在于判断系统是否处于 DDoS 攻击状态。如图 4-6 所示，熵检测系统能够准确识别系统受到攻击的时间点。通过对比图 4-5 与图 4-6 中的时间位置可以发现，本文所提出的信息熵检测系统具有较高的攻击识别准确率。进一步分析图 4-5 和图 4-6 可以发现，攻击流量集中出现时，系统熵曲线

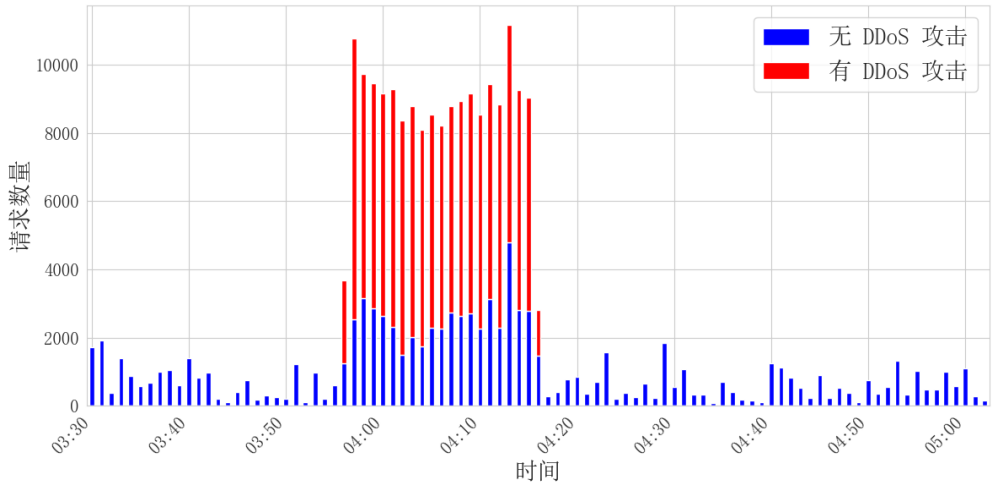


图 4-5　系统流量分布图

能够在相应时间段产生明显偏离，说明本文构建的多维信息熵体系能够从整体网络状态层面反映 DDoS 攻击对系统运行状态的影响。与直接基于单一流量字段进行检测的方法不同，本文并未仅依赖源 IP、目的 IP 或端口等局部特征，而是综合考虑请求数量、CPU 使用率、带宽、网络时延和控制信息传输能力等指标。因此，即使攻击流量在局部特征上存在伪装，仍可能通过系统资源占用、链路拥塞或控制信息传输能力下降等方式表现为整体熵值异常。这也是本文方法能够适应双栈 SDN 环境的重要原因。

从检测机制角度看，Z-score 方法的作用在于刻画当前系统熵相对于历史正

常状态的偏离程度。由于 DDoS 攻击通常会在较短时间内改变流量分布和资源占用状态，因此系统熵的突变能够作为攻击发生的有效信号。与此同时，滑动窗口机制能够平衡检测实时性和稳定性：窗口过小容易受短时正常突发流量干扰，窗口过大则可能降低攻击响应速度。本文采用固定窗口持续更新历史熵分布，使检测过程能够在保持统计稳定性的同时及时捕捉异常变化。该结果说明，本文检测方法的有效性不仅来源于熵指标本身，还来源于多维熵建模与时间窗口异常判别机制的结合。

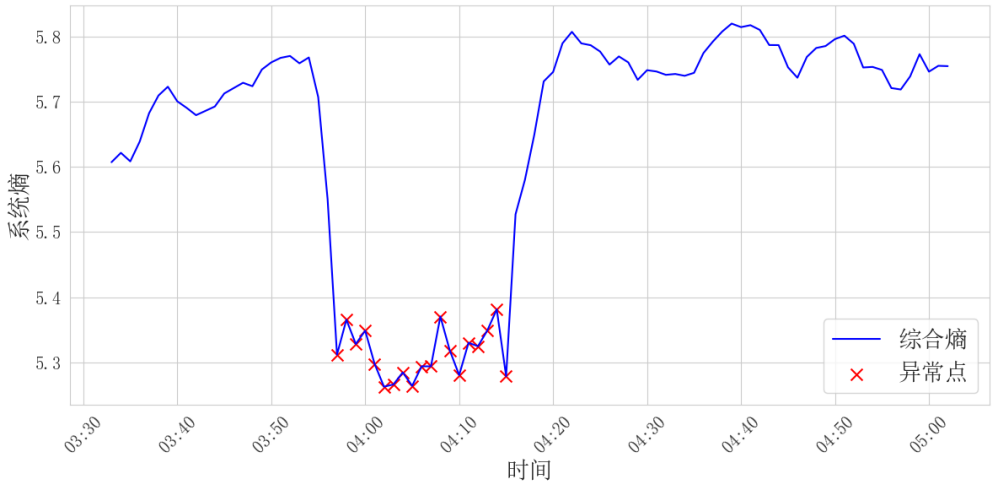


图 4-6　系统熵变化曲线及异常检测点

4.3　基于强化学习的 DDoS 攻击缓解

4.3.1　问题描述与定义

当网络系统检测到 DDoS 攻击发生后，必须及时采取有效的缓解措施，以防止攻击对网络系统造成进一步破坏。然而，在复杂且动态变化的网络环境中，网络流量特征高度复杂，不同强度和不同类型的攻击往往需要采取具有针对性的缓解策略。此外，不同层级的缓解措施对网络性能和可用性产生的影响程度也存在显著差异。因此，亟需设计一种兼顾防御效果与系统稳定性的代价最优缓解策略，在有效抵御 DDoS 攻击的同时，尽可能降低对正常网络服务的影响。该策略的核心目标是使系统信息熵在最短时间内恢复至正常水平，从而实现对 DDoS 攻击的有效缓解。

本节从数学角度对 DDoS 攻击的动态处理问题进行形式化定义。首先给出问题所需满足的相关约束条件，随后明确该问题的优化目标。

约束条件。一个有效的攻击处理策略必须满足以下约束条件。首先，为确保智能调节算法能够平稳地对流量进行限制，防止系统熵长期维持在异常状态，有

必要约束系统熵在一段时间内不能保持不变。其形式化描述如下所示：

$$\sum_{t=T_0}^{T_n} Z^t \quad \text{subject to} \quad \forall [t_1, t_2] \subseteq [T_0, T_n],\ \exists t' \in (t_1, t_2),\ Z^{t'} \neq Z^{t_1} \tag{4-8}$$

同时，为避免智能算法过度调整而将正常流量误判为恶意流量，需要对算法施加约束，使其在相邻时刻对网络流量的调整幅度不能过大，如下所示：

$$\left| \frac{N_B^t - N_B^{t-1}}{N_B^{t-1}} \right| \geq 0.5, \quad \forall t \in [T_0, T_n] \tag{4-9}$$

其中，N_B^t 表示时刻 t 的整体网络流量，N_B^{t-1} 表示时刻 $t-1$ 的整体网络流量。

在保障网络系统正常运行的前提下，还需要确保缓解策略能够使网络状态逐步恢复至正常水平，而不是因过度限制而导致网络不可用。相应约束条件如下所示：

$$\left| \frac{E_S^t - \mu_{E_S}^{T-t}}{\sigma_{E_S}^{T-t}} \right| \geq Z_h, \quad \forall t \in [T_0, T_n] \tag{4-10}$$

其中，E_S^t 表示时刻 t 的系统熵值，$\mu_{E_S}^{T-t}$ 表示在时间区间 T 内（不包含时刻 t）的系统熵均值，$\sigma_{E_S}^{T-t}$ 表示对应的标准差，Z_h 为算法允许的最大 Z-score 偏差阈值。

最后，还需保证系统熵不会退化至所有值均为零的极端状态，该状态意味着系统拒绝所有流量或处于不可接受的异常状态，其约束条件如下所示：

$$\sum_{t=T_0}^{T_n} Z^t > 0 \tag{4-11}$$

优化目标。为应对不断变化的网络环境，系统通过信息熵系统对网络状态进行动态监测，并据此对当前网络请求进行相应调整。然而，本文亟需解决的核心问题在于：如何在最小代价下选择最优缓解策略，使系统熵值尽快恢复至正常水平，从而实现对 DDoS 攻击的有效缓解。

首先，当检测到系统熵发生显著变化时，需要定位引发该变化的关键请求 IP 地址。相应的计算方式定义如下：

$$\text{IP\_}N_R^{(T_n)} = \arg\max_{\text{IP}_i} \left(\frac{N_R(\text{IP}_i)^{(T_n)} - \mu_{N_R(\text{IP}_i)}^{(T-T_n)}}{\sigma_{N_R(\text{IP}_i)}^{(T-T_n)}} \right) \tag{4-12}$$

$$\text{IP\_}U_C^{(T_n)} = \arg\max_{\text{IP}_i} \left(\frac{U_C(\text{IP}_i)^{(T_n)} - \mu_{U_C(\text{IP}_i)}^{(T-T_n)}}{\sigma_{U_C(\text{IP}_i)}^{(T-T_n)}} \right) \tag{4-13}$$

$$\text{IP\_}N_B^{(T_n)} = \arg\max_{\text{IP}_i} \left(\frac{N_R(\text{IP}_i)^{(T_n)} - \mu_{N_B(\text{IP}_i)}^{(T-T_n)}}{\sigma_{N_B(\text{IP}_i)}^{(T-T_n)}} \right) \tag{4-14}$$

$$\text{IP\_}N_D^{(T_n)} = \arg\max_{\text{IP}_i} \left(\frac{N_D(\text{IP}_i)^{(T_n)} - \mu_{N_D(\text{IP}_i)}^{(T-T_n)}}{\sigma_{N_D(\text{IP}_i)}^{(T-T_n)}} \right) \tag{4-15}$$

$$\text{IP\_}C_C^{(T_n)} = \arg\max_{\text{IP}_i} \left(\frac{C_C(\text{IP}_i)^{(T_n)} - \mu_{C_C(\text{IP}_i)}^{(T-T_n)}}{\sigma_{C_C(\text{IP}_i)}^{(T-T_n)}} \right) \tag{4-16}$$

其中，$\text{IP\_}N_R^{(T_n)}$、$\text{IP\_}U_C^{(T_n)}$、$\text{IP\_}N_B^{(T_n)}$、$\text{IP\_}N_D^{(T_n)}$ 和 $\text{IP\_}C_C^{(T_n)}$ 分别表示在时刻 T_n，网络请求数、控制器 CPU 利用率、网络带宽、网络时延以及 SDN 控制信息传输能力这五个指标中变化最为显著的请求 IP 地址。

由上述五个指标所对应的异常 IP 可构成当前时间段内的恶意 IP 列表，其定义如下：

$$\text{BL}^{(T_n)} = \left[\text{IP}_{NR}^{(T_n)},\ \text{IP}_{UC}^{(T_n)},\ \text{IP}_{NB}^{(T_n)},\ \text{IP}_{ND}^{(T_n)},\ \text{IP}_{CC}^{(T_n)} \right] \tag{4-17}$$

进一步地，整个网络在多个时间段内的恶意 IP 列表可表示为：

$$\text{BL} = \left[\text{BL}^{(T_0)},\ \text{BL}^{(T_1)},\ \cdots,\ \text{BL}^{(T_n)} \right] \tag{4-18}$$

DDoS 攻击检测与缓解通常以检测准确率作为评价指标。在此基础上，结合本文框架的特点，为保障网络系统的稳定运行，需要确保系统熵在一定时间范围内保持相对稳定。因此，引入系统熵稳定性指标（Entropy Stability Index，ESI），其定义如下：

$$\text{ESI}^{(T_n)} = \frac{1}{n} \sqrt{\sum_{t=T_0}^{T-T_n} \left(Z^{(T_n)} - Z^t \right)^2} \tag{4-19}$$

最后，为在保证缓解效果的同时尽量降低对网络系统的长期影响，本文引入缓解动作代价指标，其定义如下：

$$C_a = \sum_{t=T_0}^{T_n} f(a^t) \tag{4-20}$$

其中，$f(\cdot)$ 为动作评估函数，用于根据缓解动作的强度为其分配相应的代价值。缓解动作强度越大，其代价越高；反之，动作越温和，其代价越低。

综上所述，本文的优化目标可定义为：

$$\min \left(\sum_{t=T_0}^{T_n} \text{ESI}^t + C_a \right) \tag{4-21}$$

s.t. (4 – 8)–(4 – 11)

4.3.2 马尔可夫决策过程

马尔可夫决策过程（Markov Decision Process，MDP）[43] 是强化学习中的核心理论之一，为不确定环境下的决策问题提供了一种严格的数学建模方法。MDP 基于马尔可夫性质，通过对状态、动作、状态转移概率以及奖励函数的形式化定义，对决策过程进行描述，并借助优化方法求解最优策略。一般而言，MDP 可表示为五元组 $(S, A, P_{SA}, R, \gamma)$，其中，$S$ 表示状态空间集合，A 表示动作空间集合，P_{SA} 表示在状态 S_i 下执行动作 A_i 后转移至状态 S_{i+1} 的概率分布，R 为奖励函数，γ 为折扣因子。

状态空间：状态空间用于刻画网络在某一时刻的运行状态，其定义依赖于信息熵系统，因此包含熵检测框架中的五个关键指标。状态空间的形式化定义如式（21）所示。其中，$N_R(t) = [N_R^{r_1}(t), N_R^{r_2}(t), \ldots, N_R^{r_{|N|}}(t)]$ 表示在时刻 t，各个 IP 所对应的数据流请求数量；$U_C(t) = [U_C^{r_1}(t), U_C^{r_2}(t), \ldots, U_C^{r_{|N|}}(t)]$ 表示控制器在处理各 IP 数据流时的资源消耗情况；$N_B(t) = [N_B^{r_1}(t), N_B^{r_2}(t), \ldots, N_B^{r_{|N|}}(t)]$ 表示各 IP 在时刻 t 占用的网络带宽；$N_D(t) = [N_D^{r_1}(t), N_D^{r_2}(t), \ldots, N_D^{r_{|N|}}(t)]$ 表示各 IP 在时刻 t 所发送数据流的平均传输时延。此外，$C_C(t)$ 表示控制器在时刻 t 的控制信息传输能力。

因此，状态空间的维度为 $4|N|+1$。上述多维信息的综合能够更加全面、准确地描述当前网络状态，为后续决策过程提供重要依据。状态空间定义如下：

$$S(t) = \{N_R(t), U_C(t), N_B(t), N_D(t), C_C(t)\} \tag{4-22}$$

动作空间：智能 DDoS 处理算法中的智能体根据当前网络状态选择最优的数据流处理策略。因此，其动作空间由 I 维动作元组构成，可定义为：

$$A(t) = \left[(ip_0, r_0),\ (ip_1, r_2),\ \ldots,\ (ip_{|I|}, r_{|I|})\right] \tag{4-23}$$

其中，$(ip_{|i|}, r_{|i|})$ 表示对 IP 地址为 i 的数据流执行相应的缓解动作 $r_{|i|}$。需要指出的是，动作执行后系统状态必须满足式（8）–（11）所给出的约束条件。

状态转移概率：状态转移概率描述了在给定状态下执行某一动作后，网络系统从当前状态转移至下一状态的概率分布。在本文研究场景中，由于网络环境复杂多变，下一状态不仅依赖于所选动作，还受到多种环境因素的影响，因此状态转移概率是未知的。

奖励函数：在 MDP 框架中，每一次状态转移都会伴随一个奖励信号，用于评价智能体当前决策的优劣。即时奖励函数可定义如下：

$$r(t) = \begin{cases} |Z^t| - |Z^{t-1}|, & \text{TP} \\ |Z^t - Z^{t-1}|, & \text{FP} \\ -\max(|Z^t|, |Z^{t-1}|), & \text{FN} \\ -|Z^t - Z^{t-1}|, & \text{TN} \end{cases} \tag{4-24}$$

在式（24）中，假设算法的处理过程始终满足式（8）-（11）所给出的约束条件。本文的目标是在最大化 DDoS 攻击缓解效果的同时，使系统熵尽快恢复至正常水平，并尽量降低缓解动作对网络系统造成的长期影响。在奖励机制设计方面，本文引入 Z-score[61] 来判断是否需要执行缓解操作，并依据处理结果在混淆矩阵中的位置分配相应奖励或惩罚。该奖励设计方式无需依赖人工标注标签，使算法在真实网络环境中具有更强的可行性。

通过对混淆矩阵四种不同结果进行精细化奖励设计，不仅能够提升智能体的学习效率，还能有效缓解奖励稀疏问题[62]。同时，精细化奖励机制有助于智能体在实际生产环境中应对多样化场景，从而提升整体策略性能。

在强化学习过程中，本文进一步引入周期性的动作奖励与惩罚机制。智能体的目标不再局限于最大化单步即时奖励，而是通过一个完整训练周期内动作代价的累计回报来学习最优策略。给定时刻 t 的累计回报定义如下：

$$R(t) = r(t) + \gamma \cdot r(t+1) + \gamma^2 \cdot r(t+2) + \cdots = \sum_{n=0}^{\infty} \gamma^n \cdot r(t+n) \qquad (4-25)$$

如图 4-7 所示，在每一轮训练周期结束后，系统根据各动作的执行代价对其进行排序，并分配递减的奖励或惩罚。因此，在该奖惩机制约束下，每个状态对应的动作集合会动态变化。随着训练过程的推进，高代价动作将持续受到惩罚，而低代价动作则会获得相应奖励。最终，系统在代价—奖励机制的约束下，实现攻击缓解效果与缓解代价之间的平衡，从而达到相对最优的动态均衡状态。

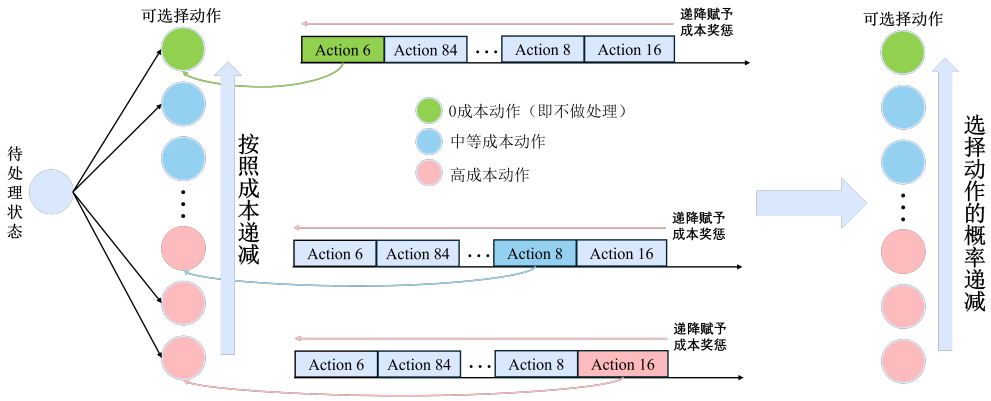


图 4-7　动作代价奖励与惩罚机制原理示意图

4.3.3　智能化 DDoS 攻击动态处理算法

马尔可夫决策过程通常可通过动态规划（Dynamic Programming，DP）[63] 或深度强化学习（Deep Reinforcement Learning，DRL）[64] 方法进行求解。然而，在本文所研究的网络场景中，由于状态空间规模巨大，传统 DP 方法将面临严重的

维度灾难问题，需要存储和计算大量状态—策略映射，难以在实际系统中高效实现。相比之下，DRL 能够借助深度神经网络基于当前状态和历史经验进行决策，而无需显式记录所有状态—动作映射关系。

此外，由于实际网络环境本身具有高度复杂性和不确定性，难以事先构建精确的系统模型，因此本文采用无模型（model-free）的 DRL 方法来解决该问题。同时，通过将交互经验存储至经验池并进行经验回放，能够不断更新网络参数，修正 Q 值估计并逐步学习最优策略，使其更适用于本文所涉及的随机性强、动态变化快的网络环境。因此，本文选择采用 DRL 框架来解决 DDoS 攻击的动态处理问题。在智能化 DDoS 动态处理算法的深度强化学习模型中，本文采用双重

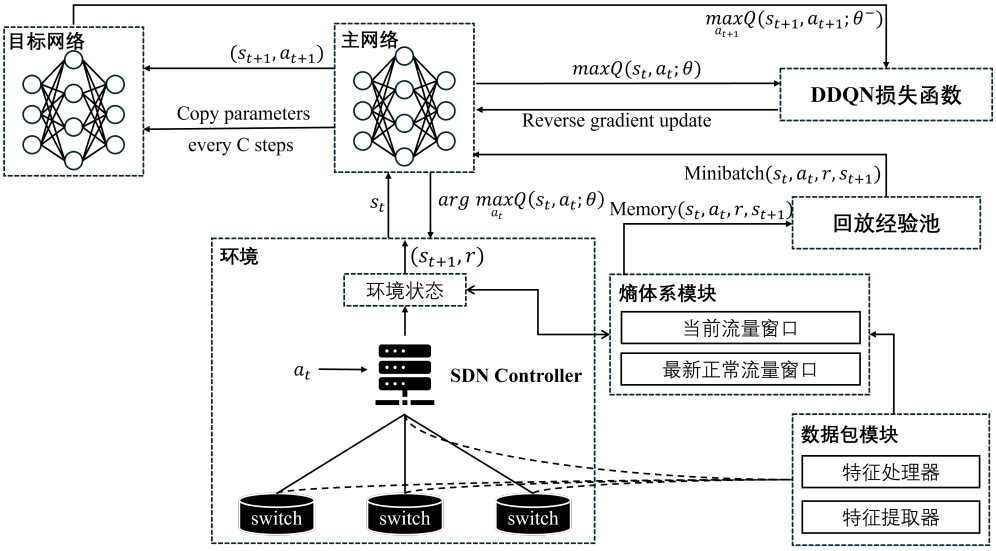


图 4-8　智能 **DDoS** 攻击动态处理算法训练流程

深度 Q 网络（Double Deep Q-Network，DDQN）算法。尽管深度 Q 网络（DQN）在诸多问题中取得了良好效果，但其在目标网络价值计算过程中始终选择最大 Q 值，容易引发过估计问题。在网络安全场景下，DQN 的过估计问题可能导致对熵系统状态的错误评估，从而做出不恰当的缓解决策，进而造成整体策略不稳定、攻击缓解效果不佳。因此，本文采用 DDQN 算法以有效缓解 DQN 中存在的过估计问题。

如图 4-8 所示，通过该过程，主网络参数可不断更新，并在一定迭代次数后同步至目标网络。该过程不断重复，直至损失函数收敛，模型训练完成。

在本文所构建的 DDQN 训练模型中，信息熵系统模块与环境及 DDQN 结构相互融合，使得 DDQN 的奖励机制能够在无需人工标签的情况下独立运行。前文已对算法的数学模型及马尔可夫决策过程进行了详细描述，包括状态空间、动作空间、状态转移概率和奖励机制。智能化 DDoS 攻击动态处理算法的完整训练流程如算法6所示。

复杂度分析：设训练轮数为 E，每轮迭代次数为 T，经验回放批量大小为 B，

动作空间规模为 $|A|$，主网络一次前向传播的复杂度为 C_f，一次反向传播更新的复杂度为 C_b。在每个训练步中，智能体需要基于当前状态通过 ϵ-greedy 策略选择动作，并与环境交互获得下一状态和奖励。动作选择过程需要对候选动作进行 Q 值估计，其时间复杂度可表示为 $O(|A|C_f)$；网络训练阶段则需要对小批量经验样本进行反向传播更新，其时间复杂度为 $O(BC_b)$。因此，智能 DDoS 攻击动态处理算法的总体训练时间复杂度为 $O(ET(|A|C_f + BC_b))$。模型部署后，在线缓解阶段主要执行一次动作选择，时间复杂度为 $O(|A|C_f)$。在空间复杂度方面，算法需要维护经验回放池、动作代价评估信息、主网络参数和目标网络参数。设经验池容量为 M，状态维度为 d_s，动作维度为 d_a，神经网络参数量为 W，则空间复杂度为 $O(M(d_s + d_a) + W)$。

Algorithm 6 智能 **DDoS** 攻击动态处理算法

输入：初始探索率 ϵ，折扣因子 γ，学习率 α，目标网络更新频率 T，网络结构 Network，训练轮数 Episode，迭代次数 Epoch，环境 Env；

输出：攻击处理策略 π；

初始化经验池 E；
初始化主网络参数 θ；
初始化目标网络参数 $\theta^- = \theta$；
初始化终止标志 η；
for $episode \leftarrow 1$ **to** $Episode$ **do**
  初始化状态空间 $state = [N_R[0], U_C[0], N_B[0], N_D[0], C_C[0]]$
  for $epoch \leftarrow 1$ **to** $Epoch$ **do**
    $action \leftarrow \epsilon$-greedy$(episode)$
    $(s_{t+1}, reward) \leftarrow$ ExecuteAction(action)
    $reward \leftarrow reward+$ actionEvaluation(reward)
    if $Env.T_C > Env.T_E$ **and** constraint(s_{t+1}) **then**
      $\eta \leftarrow$ True
    end if
  end for
end for
StoreExperience(S_t,action,reward,S_{t+1})
$\theta \leftarrow$ NetworkTrain(Minibatch(E))
if episode/T == 0 **then**
  updateTargetNetwork(θ)
end if
通过主网络获得 DDoS 攻击处理策略 π；

4.3.4 仿真设计结果及分析

（一）实验结果与性能评估

本节基于构建的仿真测试平台，对所提出的安全防御框架在攻击与防御场景下进行一系列实验，以验证算法的有效性、稳定性与泛化能力。

（二）实验环境与参数设置

在实验平台方面，实验采用的硬件环境为 Higon C86 3250 八核处理器（主频 3.2 GHz）、NVIDIA RTX A4000 GPU，操作系统为 Ubuntu 20.04 与 Windows 11，开发与实验环境为 PyCharm 2023。在数据集选择方面，IPv4 流量数据采用 CIC-DDoS-2019 数据集，IPv6 流量数据采用 IEEE 提供的 IDOS6 数据集。此外，基于 PyTorch 框架完成了智能 DDoS 动态处理算法的训练与测试。实验中所采用的神经网络超参数配置如表 4-1 所示。

表 4-1 超参数配置

参数	数值
初始探索率 $\varepsilon(t)$	1
最小探索率 ε	0.01
探索衰减率	0.005
Batch size	64
折扣因子 γ	0.95
经验回放池容量	2000
优化器	Adam（学习率 0.001）
目标网络更新周期	10

（三）对比算法介绍

为了全面评估本文所提出算法在相同实验环境下的性能优势，选取三种经典深度强化学习算法以及两种引入动作代价奖励机制的改进算法作为对比基准，具体如下：

1. **DQN**[65]：DQN 是一种基于价值函数的深度强化学习算法，通过深度神经网络对 Q 值函数进行近似。
2. **A2C**[66]：A2C 是一种基于策略梯度的深度强化学习算法，由策略网络与价值网络组成，并引入优势函数以减少策略更新的方差，从而提高训练稳定性。
3. **DDQN**[67]：DDQN 在 DQN 的基础上将动作选择与 Q 值评估分离，有效缓解了 DQN 中的 Q 值过估计问题。
4. **ACRDQN**：在 DQN 算法基础上引入动作代价奖励与惩罚机制的改进算法。

5. **ACRA2C**：在 A2C 算法基础上引入动作代价奖励与惩罚机制的改进算法。

（四）性能对比分析

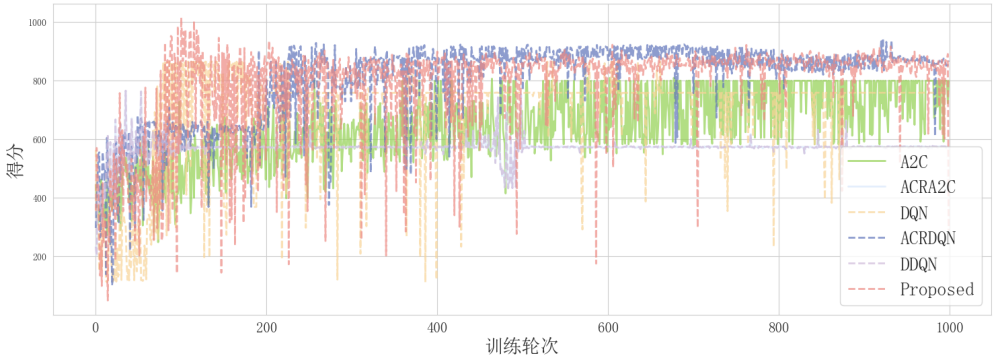


图 4-9　本文算法与对比算法训练结果对比

本文在 IPv4 网络环境下，基于请求数量熵、CPU 使用率熵、带宽熵、网络时延熵以及控制信息传输能力熵五个指标，对 A2C、ACRA2C、DQN、ACRDQN、DDQN 以及本文提出算法在 DDoS 攻击场景下的防御性能与系统稳定性进行了综合评估。

如图 4-9 所示，给出了 A2C、ACRA2C、DQN、ACRDQN、DDQN 以及本文提出算法的训练过程对比结果。可以看出，六种算法均能够实现收敛，但本文提出的算法在收敛后获得了较高的奖励值。尽管 ACRDQN 在初期训练阶段取得了最高的奖励值，但这是由于 DQN 结构本身存在 Q 值过估计问题所致。在训练超过 800 轮后，DQN 的过估计问题逐渐显现，导致后续奖励值持续下降，并最终稳定在较低水平。与此同时，ACRDQN 在长时间训练过程中同样表现出明显的性能衰减。因此，可以得出结论，本文提出的算法在整体稳定性与长期性能方面均显著优于其他对比算法。

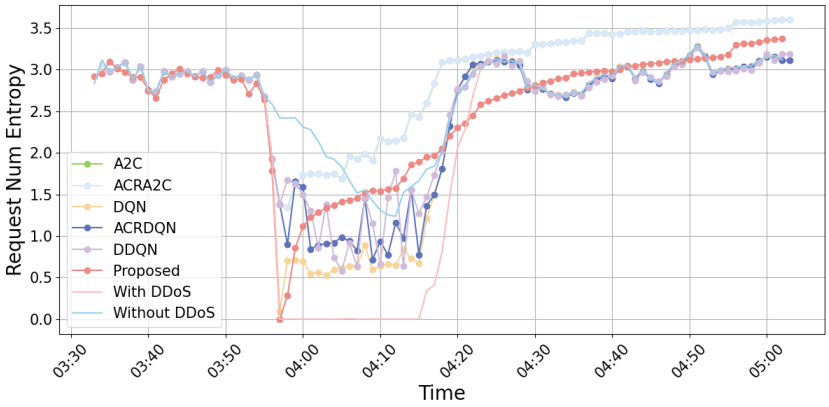


图 4-10　IPv4 环境下不同算法的请求数量熵对比

从图 4-9 的训练曲线可以进一步看出，本文算法在收敛稳定性方面优于

DQN、ACRDQN 等方法。DQN 在训练初期可能获得较高奖励，但由于其目标 Q 值计算中直接选取最大动作价值，容易产生过估计问题，导致模型在后续训练中对部分高收益动作形成不稳定偏好，进而造成奖励下降。ACRDQN 虽然引入动作代价奖励机制，但其基础结构仍受到 DQN 过估计问题影响，因此在长时间训练后仍可能出现性能衰减。相比之下，本文方法采用 DDQN 结构，将动作选择与动作评估过程分离，降低了 Q 值过估计对策略学习的影响；同时，动作代价机制使模型在追求攻击缓解效果的同时考虑防御动作对网络状态的长期影响，从而避免频繁或过强的调节行为。训练结果说明，本文算法不仅能够获得较高奖励值，而且能够维持更稳定的策略更新过程，这对于动态网络安全防御场景具有重要意义。

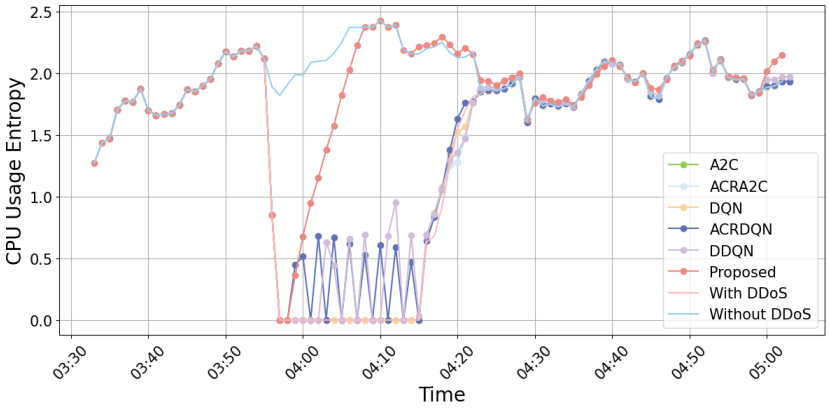


图 4-11　IPv4 环境下不同算法的 CPU 使用率熵对比

如图 4-10 所示，当 DDoS 攻击发生时，各基线算法的请求数量熵均出现显著下降，并伴随一定程度的回升与剧烈波动，反映出系统稳定性较差。其中，ACRA2C 与 DQN 的波动尤为明显。相比之下，本文提出的算法在攻击过程中呈现出明显的熵值回升趋势，并保持较高的稳定性。

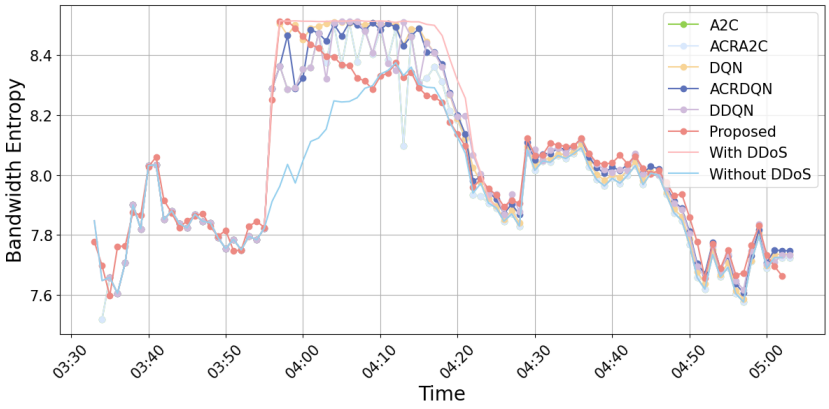


图 4-12　IPv4 环境下不同算法的带宽熵对比

如图 4-11 所示，在 DDoS 攻击期间，各基线算法在 CPU 使用率熵方面的表现与图 4-10 类似，整体稳定性较差。而本文提出的算法在遭受攻击时，CPU 使用率熵能够显著且稳定地回升，体现出较强的资源调度能力。

如图 4-12 所示，在攻击初期，所有算法的带宽熵均出现一定程度的上升，反映了网络中带宽使用不确定性的增加。然而，随着攻击持续，ACRDQN 与 DQN 出现明显的剧烈波动，而本文提出的算法能够快速适应攻击环境，并将带宽熵稳定在较高水平，体现出其在带宽分配与拥塞控制方面的优势。

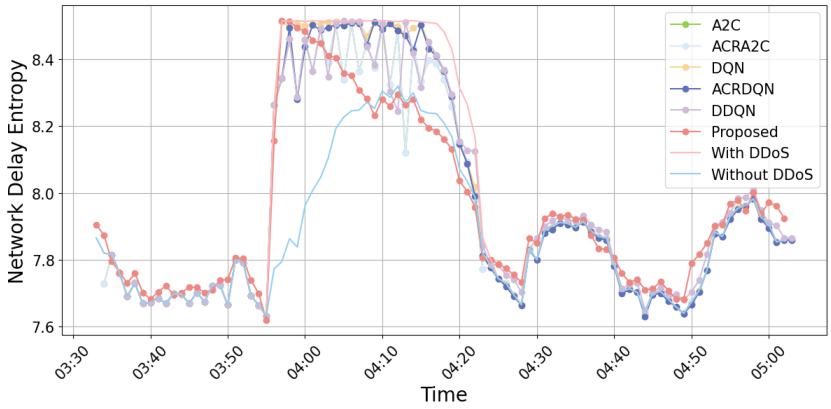


图 4-13　IPv4 环境下不同算法的网络时延熵对比

如图 4-13 所示，除本文提出的算法外，其余基线算法在网络时延熵方面均表现出较大的波动幅度且恢复速度较慢。相比之下，本文提出的算法在攻击过程中具有更快的收敛速度和更小的熵值波动，有效降低了系统时延的不确定性，表明其在保障服务质量方面具有明显优势，尤其适用于对时延敏感的网络场景。

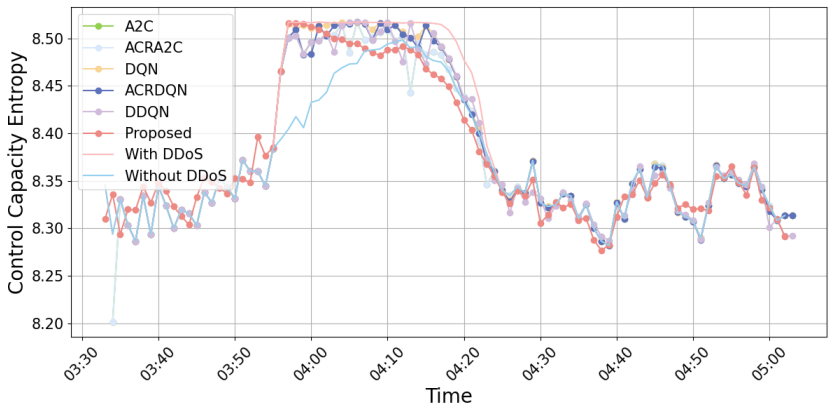


图 4-14　IPv4 环境下不同算法的控制信息传输能力熵对比

如图 4-14 所示，本文提出的算法在整个攻击过程中均保持较高的控制信息传输能力熵，其性能接近于无 DDoS 攻击时的理想状态，而其他基线算法则表现出不同程度的熵值下降。这表明本文提出的算法在控制平面通信与指令传输稳

定性方面具有更强的鲁棒性。

如图 4-15 所示，系统熵综合反映了上述五项指标所表征的系统整体稳定性与自适应能力。实验结果表明，本文提出的算法在 DDoS 攻击期间能够维持较高的系统熵水平，并在攻击结束后快速恢复至正常状态，展现出显著的自恢复能力。而其他基线算法在攻击过程中系统熵值较低，恢复速度缓慢，暴露出其在动态复杂环境下的不足。

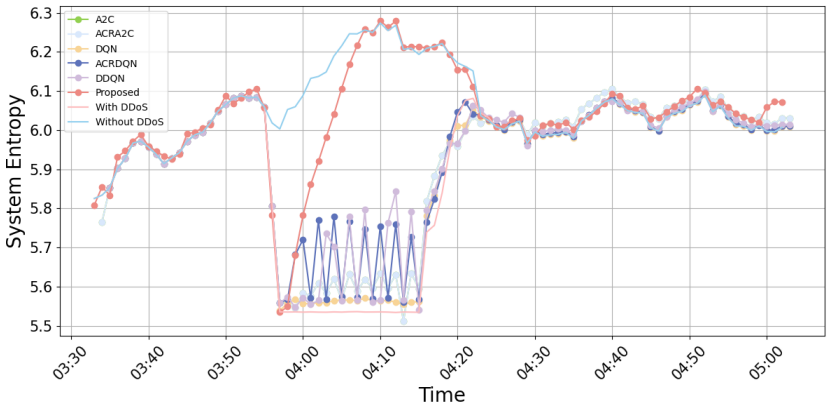


图 4-15　IPv4 环境下不同算法的系统熵对比

综上所述，实验结果充分表明，本文提出的算法在 IPv4 网络环境下应对 DDoS 攻击时，具有较高的稳定性、自适应能力以及综合防御性能，能够有效提升复杂网络环境中的服务连续性与系统安全性。

对 IPv4 环境下的各项实验结果进行综合分析可以发现，不同熵指标反映了 DDoS 攻击对网络系统不同层面的影响。请求数量熵主要反映流量分布是否集中，攻击发生时大量请求集中指向目标节点，导致请求分布不均衡，因此多数基线算法的请求数量熵明显下降。CPU 使用率熵反映控制器资源调度是否均衡，攻击流量会引发大量流表处理和控制消息交互，使控制器资源消耗集中于异常流处理过程。带宽熵和网络时延熵分别反映链路负载分布和传输状态稳定性，当攻击造成链路拥塞时，带宽占用和时延波动会出现明显异常。控制信息传输能力熵则反映 SDN 控制平面对数据平面的调度能力，是攻击缓解能否及时生效的重要基础。

本文算法在上述五类指标中均表现出较好的稳定性，说明其并非仅在某一单项指标上取得改善，而是在请求分布、控制器资源、链路负载、时延状态和控制信息传输等多个方面同时发挥作用。这与本文算法的设计目标一致：检测阶段利用系统熵刻画整体状态，缓解阶段通过强化学习动态选择防御动作，并借助动作代价约束减少过度调节。相比之下，部分基线算法虽然能够在某些时间段抑制攻击流量，但其防御动作缺乏对长期系统状态的约束，容易造成熵值剧烈波动或恢复速度较慢。因此，IPv4 实验结果表明，本文方法在攻击抑制效果和系统稳定

性之间取得了更好的平衡。

本文在 IPv6 网络环境下，基于请求吞吐量熵、CPU 使用率熵、带宽熵、网络时延熵以及控制信息传输能力熵五个指标，对 A2C、ACRA2C、DQN、ACRDQN、DDQN 以及本文提出的算法在 DDoS 攻击场景下的攻击缓解效果和系统稳定性进行了综合评估。

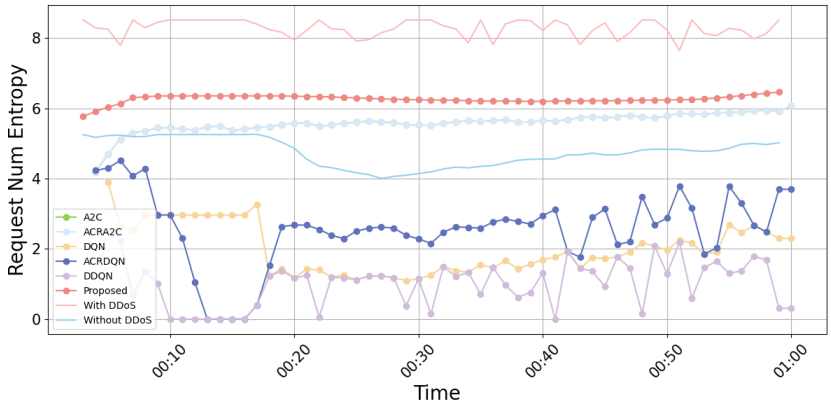


图 4-16　IPv6 环境下不同算法的请求数量熵对比

如图 4-16 所示，在 IPv6 网络环境中，当遭受 DDoS 攻击时，大多数算法的请求数量熵迅速下降，尤其是 ACRDQN 和 DDQN，其熵值波动剧烈，甚至在部分时间段接近于 0，表明在异常流量冲击下系统对请求分布的控制能力显著下降。相比之下，本文提出的算法在攻击期间能够保持较高且相对稳定的请求数量熵，说明其在 IPv6 环境下仍具备良好的请求调度能力和服务均衡能力。

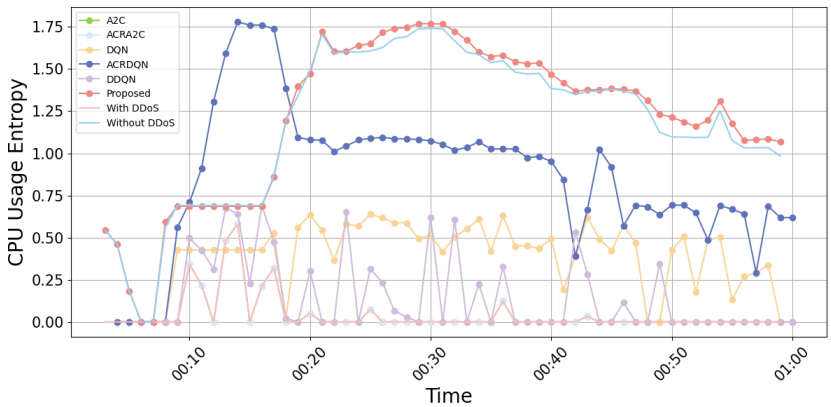


图 4-17　IPv6 环境下不同算法的 CPU 使用率熵对比

如图 4-17 所示，CPU 使用率熵的差异更加明显。DDQN 和 ACRA2C 在攻击期间的变化趋势高度相似，并在部分时间段出现熵值为 0 的情况，表明其资源分配机制较为集中，系统容易陷入不可控状态。而本文提出的算法在攻击期间的曲线形态与未受攻击状态下基本一致，表现出更优的资源调度稳定性。

如图 4-18 所示，ACRDQN 和 DQN 在带宽熵方面的表现依然较差，其熵值始终围绕受攻击状态波动。A2C、ACRA2C 和 DDQN 的表现略有改善，但在部分时间段内带宽使用仍明显偏离理想状态。相比之下，本文方法在攻击过程中能够维持与无攻击状态相近的带宽熵水平，充分体现了其在流量控制和负载均衡方面的优势。

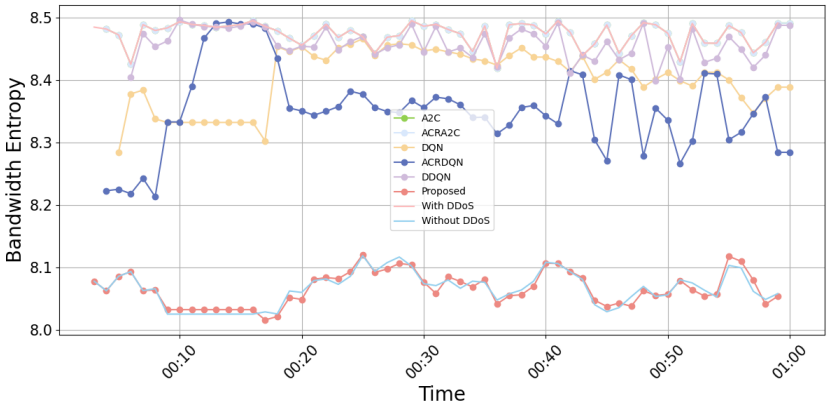


图 4-18　IPv6 环境下不同算法的带宽熵对比

如图 4-19 所示，在网络时延熵方面，除本文提出的方法外，其余基准算法在攻击期间均表现出熵值显著下降且波动剧烈的现象，尤其是 ACRDQN。较低的时延熵意味着网络响应不确定性增加，时延波动频繁。本文算法的时延熵始终稳定在接近无 DDoS 攻击时的水平，且波动范围较小，表明其具备良好的时延控制能力，适用于对实时性要求较高的业务场景。

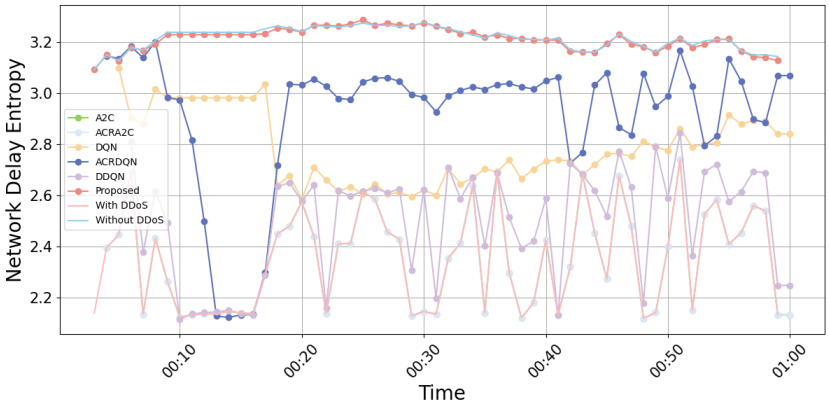


图 4-19　IPv6 环境下不同算法的网络时延熵对比

如图 4-20 所示，在控制信息传输能力熵方面，本文算法表现最为优越，其熵值在攻击期间始终接近无 DDoS 攻击时的水平，而其他基准算法的熵值则明显向攻击状态靠拢，并伴随较大幅度的波动。

如图 4-21 所示，从系统熵的整体结果来看，本文算法在攻击期间的系统熵

变化趋势始终与无 DDoS 攻击场景保持一致，且数值差异较小，体现出较强的系统稳定性。相比之下，其余基准算法的系统熵波动剧烈，与无攻击状态下的系统熵差异较大，暴露出其在动态复杂环境中的稳定性不足。

与 IPv4 环境相比，IPv6 环境下各算法之间的性能差异更加明显。这主要与 IPv6 协议自身特点有关。IPv6 具有更大的地址空间和更复杂的扩展机制，流量特征分布更加稀疏，攻击流量在地址和报文结构上的变化空间更大，使得基于固定规则或单一特征的防御方法更难保持稳定性能。从实验结果可以看出，ACRDQN、DQN 和部分传统强化学习算法在 IPv6 环境下的熵值波动更加剧烈，说明其在面对更复杂的协议特征时容易出现策略不稳定或状态评估偏差。

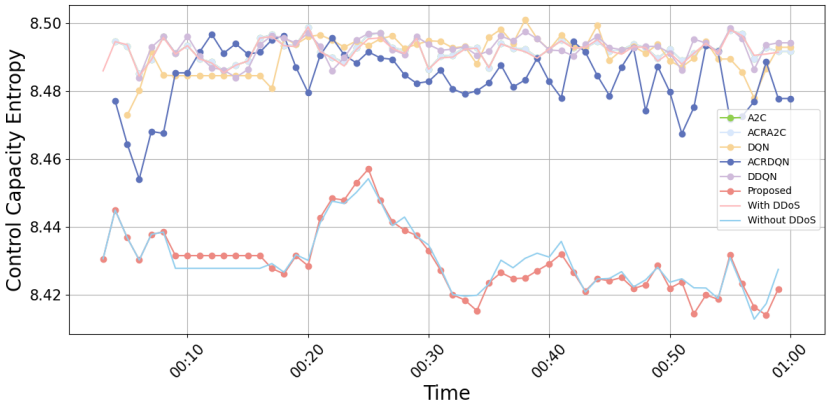


图 4-20　IPv6 环境下不同算法的控制信息传输能力熵对比

本文算法在 IPv6 环境下仍能保持与无攻击状态较为接近的熵变化趋势，说明双栈数据处理机制和多维系统熵建模能够在一定程度上降低 IPv4 与 IPv6 之间的协议差异对模型判断的影响。通过对 IPv4 和 IPv6 特征进行统一表示，模型能够在同一状态空间下处理不同协议流量；通过引入请求数量、CPU 使用率、带宽、时延和控制信息传输能力等系统级指标，算法能够避免过度依赖某一类协议字段，从而增强对 IPv6 复杂流量环境的适应能力。由此可见，本文方法在 IPv6 环境中的优势不仅来自强化学习模型本身，也来自前端双栈特征统一处理和系统级熵指标设计。

综上所述，实验结果表明，本文提出的算法不仅在传统 IPv4 网络环境中具有良好的攻击缓解性能，在更为复杂的 IPv6 网络环境下同样表现出较强的泛化能力和鲁棒性，是一种适用于现代双栈网络安全防御场景的有效方法。

如图 4-22 所示，对本文提出的算法与五种基准算法的缓解准确率进行了对比分析。可以看出，在 IPv4 环境下，六种算法均表现出较高的缓解准确率，其差异主要体现在 IPv6 环境中。实验结果表明，未引入动作代价奖励与惩罚机制的基准算法在 IPv6 环境下的性能明显劣于引入该机制后的改进算法。

然而，由于 DQN 算法本身存在较为严重的价值函数过估计问题，在引入动

作代价奖励与惩罚机制后，该问题进一步被放大，其整体性能仍然劣于改进后的 ACRA2C 算法以及本文提出的算法。综合来看，本文提出的算法在双栈网络环境下整体性能优于其余五种基准算法。

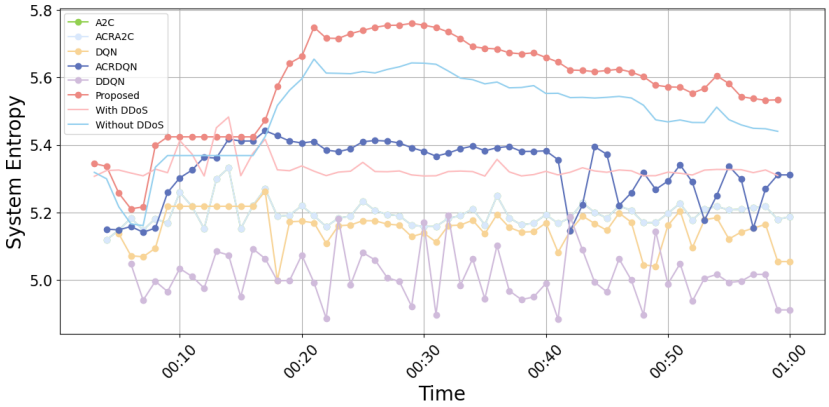


图 4-21　IPv6 环境下不同算法的系统熵对比

在对 IPv4 与 IPv6 双协议栈环境下的实验结果进行对比分析时，可以观察到部分算法在 IPv6 环境中的检测性能低于其在 IPv4 环境中的表现。该现象主要由以下两个方面的因素造成：首先，当前公开可用的 IPv6 流量数据集规模相对较小，这在一定程度上限制了模型对 IPv6 特有流量特征分布的充分学习，从而削弱了其泛化能力与稳定性；其次，IPv6 协议本身具有更大的地址空间以及更为复杂的报文首部结构，使得相关统计特征在分布上更加稀疏且波动性更强，进一步增加了攻击检测与缓解的难度。

相比之下，IPv4 流量数据规模更大，特征分布相对集中，有助于模型在训练过程中更快收敛并取得更优性能。上述实验现象与已有研究中关于 IPv6 网络入侵检测复杂性更高的结论一致，也进一步凸显了在多协议栈网络环境下进行算法优化设计的必要性。

总体来看，本文实验结果从检测准确性、策略稳定性、协议适应性和系统开销等方面验证了所提方法的有效性。首先，基于多维信息熵的检测方法能够从整体网络状态变化中捕捉 DDoS 攻击，避免了单一流量特征容易受协议差异和攻击伪装影响的问题。其次，基于 DDQN 的动态缓解算法能够在攻击发生后根据环境反馈持续调整防御动作，相比固定规则或传统强化学习算法具有更好的自适应能力。再次，动作代价奖励机制使模型在抑制攻击的同时考虑防御动作对网络性能的影响，从而降低了过度防御导致系统波动的风险。最后，IPv4 与 IPv6 实验结果表明，本文方法在双栈环境下具有较好的泛化能力，尤其是在 IPv6 这种特征更稀疏、状态更复杂的场景中，仍能够保持较高的系统熵水平和缓解准确率。

需要说明的是，本文实验主要基于仿真平台和公开数据集完成，虽然能够验

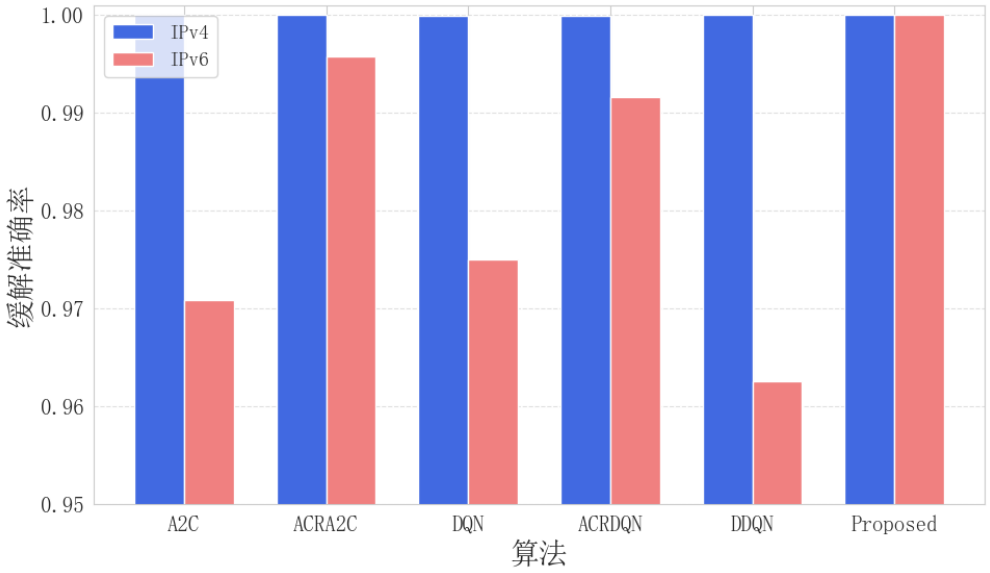


图 4-22　不同算法缓解准确率对比

证所提方法在典型攻击场景下的有效性，但与真实大规模网络环境相比，仍存在业务流量类型、攻击源分布和网络拓扑复杂度不足等限制。因此，后续研究可进一步在更大规模的真实或半实物 SDN 测试环境中验证算法性能，并引入更多类型的攻击流量和正常业务流量，以进一步评估模型在复杂网络场景下的鲁棒性和实际部署价值。

4.4　本章小结

本章针对双栈 SDN 网络环境中 DDoS 攻击检测困难、防御策略自适应能力不足等问题，从控制层角度出发，设计并实现了一种基于信息熵与深度强化学习的 DDoS 攻击检测与动态缓解方法。在攻击检测方面，利用 SDN 控制器的全局可视能力，构建多维网络状态信息熵模型，从请求数量、CPU 使用率、链路带宽、网络时延和控制信息传输能力等特征刻画网络运行状态，并通过熵值变化的综合分析实现攻击的实时检测与定位，从而提升复杂双栈环境下检测的准确性与稳定性。

在攻击缓解方面，本文将 DDoS 防御过程建模为马尔可夫决策过程，引入深度强化学习实现防御策略的自适应优化。通过构建基于熵反馈的环境评估机制，并在奖励函数中考虑动作代价，使智能体在抑制攻击流量的同时兼顾资源消耗与服务质量，降低过度防御带来的性能损失。实验结果表明，在 IPv4/IPv6 双栈 SDN 仿真环境中，所提方法在攻击检测准确率、响应速度和网络稳定性方面表现良好，相较于 A2C、DQN 和 DDQN 等算法，能够更有效提升系统整体熵水平并加快网络恢复速度，验证了该方法的有效性与优越性

第五章　总结和展望

5.1　全文总结

本文围绕双栈 SDN 环境下 DDoS 攻击防御问题，从协议层与数据层两个方面开展研究，并在第三章与第四章提出了针对性防御方法。

（1）在协议层，针对 DHCP 饥饿攻击问题，提出了一种基于 SDN 全局视角的多阶段协同防御框架。该方法通过构建自学习表并引入流量相关性分析，实现基于全局行为特征的恶意 IP 定位，突破传统局部特征检测的局限；同时设计触发式检测机制，在降低系统开销的基础上提高检测效率；进一步结合多阶段过滤策略与基于 DDQN 的智能处理算法，将 DHCP 攻击防御建模为动态优化问题，实现防御策略的自适应调整与持续优化。

（2）在数据层，针对双栈网络流量异构性强、防御策略自适应能力不足的问题，提出了一种融合信息熵与深度强化学习的动态防御方法。该方法通过统一 IPv4/IPv6 流量特征表示，解决多协议环境下的数据建模问题；构建多维全局信息熵体系，实现对网络整体状态的实时感知；并将熵检测结果作为强化学习环境反馈，结合动作成本约束奖励机制，构建无需依赖标签的自适应防御模型，在提升攻击抑制能力的同时保障网络长期稳定性。

综上，本文从协议层与数据层分别围绕“全局行为分析”和“智能决策优化”展开研究，实现了双栈 SDN 环境下 DDoS 攻击的高效检测与动态防御，验证了所提方法在复杂网络场景中的有效性与实用价值。

5.2　未来展望

尽管本文围绕双栈 SDN 环境下的 DDoS 攻击检测与防御开展了较为系统的研究，并取得了一定进展，但仍存在进一步深化空间。本文实验主要基于仿真环境，后续有必要在更大规模或多域实际网络场景中验证所提方法，以更全面评估其适应能力与稳定性。同时，当前研究重点聚焦于 DHCP 饥饿攻击与 DDoS 洪泛攻击，对于端口扫描、ARP 欺骗及应用层攻击等多类型威胁的协同防御仍有待拓展，可进一步构建统一的多攻击检测与防御框架。在模型层面，深度强化学习方法在提升性能的同时也带来较高计算开销，未来可结合模型压缩、迁移学习或联邦学习等技术，降低资源消耗并增强实际部署能力。此外，现有信息熵特征建模方式仍相对有限，后续可引入图神经网络及时序建模方法，对网络流量进行更深层次的特征刻画，提升检测准确性与鲁棒性。随着网络智能化与自动化运维的发展，进一步构建融合检测、决策与执行的闭环安全机制，将有助于提升双栈 SDN 环境下智能安全防护的整体效率与可靠性。

参考文献

[1] 罗军舟, 吴文甲, 杨明. 移动互联网: 终端、网络与服务[J]. 计算机学报, 2011, 34(11): 2029-2051.

[2] 桂畅旎, 徐国庆. 2025 年国际网络安全态势回顾与趋势展望[J]. 中国信息安全, 2026(01): 25-29.

[3] 本刊采编中心. 2025 年网络安全态势回顾及 2026 年趋势展望[J]. 中国信息安全, 2026 (01): 19.

[4] KREUTZ D, RAMOS F M, VERISSIMO P E, et al. Software-defined networking: a comprehensive survey[J]. Proceedings of the IEEE, 2014, 103(1): 14-76.

[5] 左志斌, 杨凯, 邓淼磊, 等. 基于可编程软件定义网络的动态网络防御方案[J]. 计算机应用, 2025, 45(01): 144-152.

[6] 王晓峰, 吴建平, 崔勇. 互联网 IPv6 过渡技术综述[J]. 小型微型计算机系统, 2006(03): 385-395.

[7] 杨斌. IPv4 和 IPv6 共存策略及过渡技术探讨[J]. 通信管理与技术, 2019(06): 41-42.

[8] LIU Y, YU F R, LI X, et al. Blockchain and machine learning for communications and networking systems[J]. IEEE communications surveys & tutorials, 2020, 22(2): 1392-1431.

[9] LUONG N C, HOANG D T, GONG S, et al. Applications of deep reinforcement learning in communications and networking: a survey[J]. IEEE communications surveys & tutorials, 2019, 21(4): 3133-3174.

[10] ZHUANG T, SHU Z, CHEN S, et al. Intelligent algorithm for dynamic handling of DDoS based on action cost in a dual-stack environment[J]. Computer Networks, 2026, 278: 112105-112105.

[11] 谢丽霞, 丁颖. 链路洪泛攻击的 SDN 移动目标防御机制[J]. 清华大学学报 (自然科学版), 2019, 59(01): 36-43.

[12] NEJAD M S, GAZAFRUDI S M M, POUR H N. Enhancing deep reinforcement learning through fuzzy reward granulation: a strategy for reducing agent-environment interactions[J]. Neurocomputing, 2026, 693: 133692-133692.

[13] YOUNES O S. Securing ARP and DHCP for mitigating link layer attacks[J]. Sādhanā, 2017, 42: 2041-2053.

[14] TOK M S, DEMIRCI M. Security analysis of SDN controller-based DHCP services and attack mitigation with DHCPguard[J]. Computers & Security, 2021, 109: 102394.

[15] ALDAOUD M, AL-ABRI D, AL MAASHRI A, et al. Detecting and mitigating DHCP attacks in OpenFlow-based SDN networks: a comprehensive approach[J]. Journal of Computer Virology and Hacking Techniques, 2023, 19(4): 597-614.

[16] TOPRAK C, TURKER C, ERMAN A T. Detection of DHCP starvation attacks in software defined networks: a case study[C]//2018 3rd international conference on computer science and engineering (UBMK). IEEE, 2018: 636-641.

[17] MUKHTAR H, SALAH K, IRAQI Y. Mitigation of DHCP starvation attack[J]. Computers & Electrical Engineering, 2012, 38(5): 1115-1128.

[18] ISHTIAQ H U, BHUTTA A A, MIAN A N. DHCP DoS and starvation attacks on SDN controllers and their mitigation[J]. Journal of Computer Virology and Hacking Techniques, 2024, 20(1): 15-25.

[19] URAMA G C, UKWUOMA C C, HEYAT M B B, et al. SDN-based cryptographic client authentication: a new approach to DHCP starvation mitigation[C]//International Conference on Innovations in Data Analytics. Springer, 2022: 519-532.

[20] JONY A, MIAH A S M, ISLAM M N. An effective method to detect DHCP starvation attack using port scanning[C]//2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM). IEEE, 2023: 1-6.

[21] TRIPATHI N, HUBBALLI N. Detecting stealth DHCP starvation attack using machine learning approach[J]. Journal of Computer Virology and Hacking Techniques, 2018, 14: 233-244.

[22] KALKAN K, ALTAY L, GÜR G, et al. JESS: joint entropy-based DDoS defense scheme in SDN[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(10): 2358-2372.

[23] DING D, SAVI M, SIRACUSA D. Tracking normalized network traffic entropy to detect DDoS attacks in P4[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 19 (6): 4019-4031.

[24] LIU Y, ZHI T, SHEN M, et al. Software-defined DDoS detection with information entropy analysis and optimized deep learning[J]. Future Generation Computer Systems, 2022, 129: 99-114.

[25] PANDEY N, MISHRA P K. Performance analysis of entropy variation-based detection of DDoS attacks in IoT[J]. Internet of Things, 2023, 23: 100812.

[26] SHIRSATH V A, CHANDANE M M, LAL C, et al. SYNTROPY: TCP SYN DDoS attack detection for software defined network based on Rényi entropy[J]. Computer networks, 2024, 244: 110327.

[27] AHALAWAT A, BABU K S, TURUK A K, et al. A low-rate DDoS detection and mitigation for SDN using renyi entropy with packet drop[J]. Journal of Information Security and Applications, 2022, 68: 103212.

[28] HASSAN A I, EL REHEEM E A, GUIRGUIS S K. An entropy and machine learning based approach for DDoS attacks detection in software defined networks[J]. Scientific Reports, 2024, 14(1): 18159.

[29] FENG Y, ZHANG W, YIN S, et al. A collaborative stealthy DDoS detection method based on reinforcement learning at the edge of internet of things[J]. IEEE Internet of Things Journal, 2023, 10(20): 17934-17948.

[30] LI Z, KONG Y, WANG C, et al. DDoS mitigation based on space-time flow regularities in IoV: a feature adaption reinforcement learning approach[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(3): 2262-2278.

[31] YUNGAICELA-NAULA N M, VARGAS-ROSALES C, PÉREZ-DÍAZ J A, et al. A flexible SDN-based framework for slow-rate DDoS attack mitigation by using deep reinforcement learning[J]. Journal of network and computer applications, 2022, 205: 103444.

[32] REZAPOUR A, TZENG W G. Rl-shield: mitigating target link-flooding attacks using sdn and deep reinforcement learning routing algorithm[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 19(6): 4052-4067.

[33] YUNGAICELA-NAULA N M, VARGAS-ROSALES C, PEREZ-DIAZ J A. SDN/NFV-based framework for autonomous defense against slow-rate DDoS attacks by using reinforcement learning[J]. Future Generation Computer Systems, 2023, 149: 637-649.

[34] LI Z, KONG Y, JIANG C. A transfer double deep Q network based DDoS detection method for internet of vehicles[J]. IEEE Transactions on Vehicular Technology, 2023, 72(4): 5317-5331.

[35] XIA S M, GUO S Z, BAI W, et al. A new smart router-throttling method to mitigate DDoS attacks[J]. IEEE Access, 2019, 7: 107952-107963.
[36] 贾小东, 孙向辉, 彭四伟. DHCP 协议缺点及其解决方案[J]. 计算机工程, 2007(23): 138-139+147.
[37] LLORET J, ROMERO O, LEON A. Energy-efficient SDN routing with deep learning for data center networks[J]. Sustainable Computing: Informatics and Systems, 2026, 51: 101368-101368.
[38] ALI A, ANAM S, AHMED M M. Shannon entropy in artificial intelligence and its applications based on information theory[J]. J. Appl. Emerg. Sci, 2023, 13(1): 9-17.
[39] 孙长银, 穆朝絜. 多智能体深度强化学习的若干关键科学问题[J]. 自动化学报, 2020, 46(07): 1301-1312.
[40] 陈雪. 基于人工智能的大数据网络安全防御技术[J]. 信息与电脑, 2026, 38(06): 126-128.
[41] 江宇楠, 刘琳岚, 舒坚. 基于改进 DDQN 算法的复杂网络关键节点识别方法[J]. 计算机应用研究, 2025, 42(04): 1122-1127.
[42] SHARKH M A, JAMMAL M, SHAMI A, et al. Resource allocation in a network-based cloud computing environment: design challenges[J]. IEEE Communications Magazine, 2013, 51(11): 46-52.
[43] KALLENBERG L. Markov decision processes[J]. Lecture Notes. University of Leiden, 2011, 428.
[44] BRADTKE S, DUFF M. Reinforcement learning methods for continuous-time markov decision problems[J]. Advances in neural information processing systems, 1994, 7.
[45] GETOOR R, SHARPE M. Markov properties of a markov process[J]. Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete, 1981, 55(3): 313-330.
[46] SUTTON R S, BARTO A G. Reinforcement learning: an introduction[M]. MIT press, 2018.
[47] 王坤峰, 苟超, 段艳杰, 等. 生成式对抗网络 GAN 的研究进展与展望[J]. 自动化学报, 2017, 43(03): 321-332.
[48] 王洪斌, 尹鹏衡, 郑维, 等. 基于改进的 A* 算法与动态窗口法的移动机器人路径规划[J]. 机器人, 2020, 42(03): 346-353.
[49] ARULKUMARAN K, DEISENROTH M P, BRUNDAGE M, et al. Deep reinforcement learning: a brief survey[J]. IEEE Signal Processing Magazine, 2017, 34(6): 26-38.
[50] WATKINS C J, DAYAN P. Q-learning[J]. Machine learning, 1992, 8: 279-292.
[51] KIRAN B R, SOBH I, TALPAERT V, et al. Deep reinforcement learning for autonomous driving: a survey[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(6): 4909-4926.
[52] 杨行, 张招弟, 赵利华, 等. 基于生成对抗网络的网络攻击样本增强技术研究[J]. 网络安全技术与应用, 2026(05): 21-24.
[53] 吴亮. 基于原始流量特征融合的入侵检测方法研究[D]. 燕山大学, 2024.
[54] 殷玉玲, 罗兰花. 高维数据降维算法综述[J]. 电脑知识与技术, 2025, 21(06): 12-14+26.
[55] DARU A F, HARTOMO K D, PURNOMO H D. IPv6 flood attack detection based on epsilon greedy optimized Q learning in single board compute[J]. International Journal of Electrical and Computer Engineering, 2023, 13(5): 5782-5791.
[56] DE CANNIERE C, RECHBERGER C. Finding SHA-1 characteristics: general results and applications[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer, 2006: 1-20.

[57] GUPTA S, GOYAL N, AGGARWAL K. A review of comparative study of md5 and ssh security algorithm[J]. International Journal of Computer Applications, 2014, 104(14).

[58] HADIATNA F, SUSANA R, SUDEWA B. Data transmission system via bluetooth using validation cyclic redundancy check 32 (CRC32) method[C]//AIP Conference Proceedings: Vol. 2744. AIP Publishing, 2024.

[59] SHONE N, NGOC T N, PHAI V D, et al. A deep learning approach to network intrusion detection[J]. IEEE transactions on emerging topics in computational intelligence, 2018, 2(1): 41-50.

[60] XIE R, CAO J, LI Q, et al. Disrupting the SDN control channel via shared links: attacks and countermeasures[J]. IEEE/ACM Transactions on Networking, 2022, 30(5): 2158-2172.

[61] YARO A S, MALY F, PRAZAK P, et al. Outlier detection performance of a modified z-score method in time-series rss observation with hybrid scale estimators[J]. IEEE Access, 2024, 12: 12785-12796.

[62] RIEDMILLER M, HAFNER R, LAMPE T, et al. Learning by playing solving sparse reward tasks from scratch[C]//International conference on machine learning. PMLR, 2018: 4344-4353.

[63] WANG Y, ZHANG Z, ZHANG Y, et al. A novel online adaptive dynamic programming algorithm with adjustable convergence rate[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2024, 71(3): 1371-1384.

[64] ZHU C, DASTANI M, WANG S. A survey of multi-agent deep reinforcement learning with communication[J]. Autonomous Agents and Multi-Agent Systems, 2024, 38(1): 4.

[65] FAN J, WANG Z, XIE Y, et al. A theoretical analysis of deep Q-learning[C]//Learning for dynamics and control. PMLR, 2020: 486-489.

[66] BABAEIZADEH M, FROSIO I, TYREE S, et al. Reinforcement learning through asynchronous advantage actor-critic on a gpu[A]. 2016.

[67] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double q-learning [C]//Proceedings of the AAAI conference on artificial intelligence: Vol. 30. 2016.

攻读学位期间的学术论文与研究成果

■ 学术论文

[1] Zhuang T,Shu Z,Chen S,et al. Intelligent algorithm for dynamic handling of DDoS based on action cost in a dual-Stack environment[J]. Computer Networks,2026,278112105-112105.（CCF-B，SCI 期刊，第一作者，已发表）

[2] 徐泽鹏, 舒兆港, 陈淑武, 等. 面向 SDN 流表多模态感知与 DRL 协同防御 DDoS 方法 [J]. 计算机应用研究,2026,43(02):596-603.（中文核心，第五作者，已发表）

■ 科研项目

[1] 福建省科技厅福厦协同创新平台项目，“Research and industrial application of high reliability intelligent gateway devices and systems based on LLM and SD-WAN”, No.2025E3012, 2025.09-2027.09（项目主要成员，负责智能 DDoS 检测和缓解算法设计相关任务）

[2] 来自网络通信公司的学研项目，“Research on intelligent monitoring and optimization of power information network and its industrial application”, No.KH240131A, 2024.04-2027.04（项目主要成员，负责智能 DDoS 检测和缓解算法设计相关任务）

■ 获奖情况

[1] 2023-2024 学年学业二等奖学金

[2] 第十四届 MathorCup 数学应用挑战赛全国（三等奖）

[3] 2024 中国高校计算机大赛-网络技术挑战赛省赛（三等奖）

[4] “华为杯”第二十一届中国研究生数学建模竞赛（三等奖）

[5] 2024-2025 学年学业二等奖学金

[6] 校数学建模大赛（三等奖）

[7] 2023 中国高校计算机大赛-网络技术挑战赛省赛（三等奖）

[8] 第九届华为 ICT 大赛实践赛省赛网络赛道高教组（三等奖）

[9] 第 22 届全国大学生信息安全与对抗技术竞赛（三等奖）

致谢

岂不闻光阴如骏马加鞭，日月如落花流水，三年研究生生涯在提笔写下致谢之时也将要匆匆落幕。读研的这三年，我总是在寻常时光里想着，要在毕业论文的致谢中穷尽这三年所有的酸甜苦辣，可真到了写致谢之时，才发觉终是无法以言语道尽这三年的时光，一时无语凝噎。

回望这三年的研究生时光，心中有太多难以言说的感慨。一路走来，有过收获时的喜悦，也有过迷茫时的无措；有过坚持后的释然，也有过深夜独自面对压力与挫折时的泪水。那些看似平凡却又漫长的日子，如今回想起来，早已成为我人生中弥足珍贵的一段经历。正是在这段时光里，我逐渐学会了独立思考，学会了在困难面前调整自己，也学会了以更加从容和坚定的态度面对学习、科研与生活中的种种挑战。

在此，我要衷心感谢我的导师在学业和科研道路上给予我的悉心指导与耐心帮助。无论是在论文选题、研究思路，还是在论文写作与修改过程中，导师都给予了我宝贵的意见和严格的要求，使我不断发现自身不足，并在一次次修改与完善中获得成长。导师严谨的治学态度、踏实的工作作风和宽厚的为人品格，都对我产生了深远影响，也将成为我今后继续前行的重要力量。

同时，我也要感谢一路陪伴我的同门、师兄师姐以及师弟师妹们。感谢你们在我遇到困难时给予的鼓励与帮助，感谢你们在学习和生活中带来的温暖与陪伴。那些一起讨论问题、修改论文、准备汇报的日子，那些彼此鼓励、共同进步的瞬间，都让原本枯燥而紧张的研究生生活多了许多温情与色彩。因为有你们的陪伴，许多艰难的时刻不再那么孤单，许多平凡的日子也因此变得闪闪发光。

三年时光匆匆而过，曾经以为漫长的旅程，如今已悄然走到尾声。感谢这段经历给予我的磨炼与成长，也感谢所有在我前行路上给予关心、支持和帮助的人。正是你们的出现，让我原本暗淡的研究生生涯变得熠熠生辉，也让我更加坚定地相信，所有的努力与坚持，终将在岁月中留下温暖而清晰的印记。