软件定义网络抗拒绝服务攻击的流表溢出防护

王东滨[1,2]，吴东哲[1]，智慧[3]，郭昆[1,4,5]，张勖[1]，时金桥[1]，张宇[6,7]，陆月明[1,4]

（1. 北京邮电大学网络空间安全学院，北京 100876；2. 链网融合技术教育部工程研究中心，北京 100876；
3. 中国民航信息网络股份有限公司，北京 100190；4. 移动互联网安全技术国家工程研究中心，北京 100876；
5. 中关村实验室，北京 100094；6. 哈尔滨工业大学网络空间安全学院，黑龙江 哈尔滨 150001；
7. 鹏城实验室网络空间安全研究中心，广东 深圳 518055）

摘 要：针对拒绝服务攻击导致软件定义网络交换机有限的流表空间溢出、正常的网络报文无法被安装流表规则、报文转发时延、丢包等情况，提出了抗拒绝服务攻击的软件定义网络流表溢出防护技术 FloodMitigation，采用基于流表可用空间的限速流规则安装管理，限制出现拒绝服务攻击的交换机端口的流规则最大安装速度和占用的流表空间数量，避免了流表溢出。此外，采用基于可用流表空间的路径选择，在多条转发路径的交换机间均衡流表利用率，避免转发网络报文过程中出现网络新流汇聚导致的再次拒绝服务攻击。实验结果表明，FloodMitigation 在防止交换机流表溢出、避免网络报文丢失、降低控制器资源消耗、确保网络报文转发时延等方面能够有效地缓解拒绝服务攻击的危害。

关键词：软件定义网络；拒绝服务攻击；流表溢出；路径选择

中图分类号：TP393

文献标志码：A

DOI: 10.11959/j.issn.1000–436x.2023036

Preventing flow table overflow against denial of service attack in software defined network

WANG Dongbin[1,2], WU Dongzhe[1], ZHI Hui[3], GUO Kun[1,4,5],
ZHANG Xu[1], SHI Jinqiao[1], ZHANG Yu[6,7], LU Yueming[1,4]

1. School of Cyberspace Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. Engineering Research Center of Blockchain and Network Convergence Technology, Ministry of Education, Beijing 100876, China
3. TravelSky Technology Limited, Beijing 100190, China
4. National Engineering Research Center for Mobile Network, Beijing 100876, China
5. Zhongguancun Laboratory, Beijing 100094, China
6. School of Cyberspace Science, Harbin Institute of Technology, Harbin 150001, China
7. Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518055, China

Abstract: Aiming at denial of service attacks would cause overflow of the limited flow table space of the switch in software defined network, failure to install flow table rules for normal network packets, packet forwarding delay, and packet loss, FloodMitigation was proposed to prevent flow table overflow against denial of service attacks in software defined network. The management of the rate-limit flow rule installation based on available flow table space was adopted to limit the maximum installation speed of flow rules and the number of flow table space occupied by switch ports with denial-of-service attacks, and avoid flow table overflow. In addition, path selection based on available flow table space was adopted to balance flow table utilization of switches among multiple forwarding paths to avoid denial of service attacks on switches with less available flow table in the path. The experimental results demonstrate that FloodMitigation can effectively alleviate the harm of denial of service attacks in terms of preventing switch flow table overflow and packet loss, reducing resource consumption of controllers, and ensuring packet forwarding delay.

Keywords: software defined network, denial of service attack, flow table overflow, path selection

收稿日期：2022–09–16；**修回日期**：2022–12–26

基金项目：国家重点研发计划基金资助项目（No.2020YFB1808100）；中国高校产学研创新基金资助项目（No.2021FNA02004）

Foundation Items: The National Key Research and Development Program of China (No.2020YFB1808100), China University Industry-University-Research Collaborative Innovation Fund (No.2021FNA02004)

0 引言

传统封闭的网络设备内置了过多复杂协议，随着网络规模的不断扩大，网络配置复杂度越来越高、网络的管理和维护越来越复杂，增加了运营商定制优化网络的难度[1]。软件定义网络（SDN，software defined network）将控制平面与数据平面分离解耦，使具有全局网络视图的控制平面能够对数据平面进行集中式管理，实现路由策略集中计算和下发，数据平面按照控制平面下发的流规则进行网络数据的传输转发。软件定义网络使网络管理变得更具灵活性和创新性，并通过多控制器实现网络扩展及对大规模复杂网络的集中管理和维护[2-4]。

目前，OpenFlow[2]作为软件定义网络协议已被广泛采用，当有网络报文到达数据平面交换机时，交换机会先匹配已安装的流规则，如果成功匹配，则按照流规则的动作执行修改、转发和丢弃网络报文等操作；如果未成功匹配（即出现 table-miss），则交换机将网络报文封装到 packet-in 消息中上报给控制平面控制器，控制器计算转发路径后，向交换机下发安装流规则的 flow-mod 消息和转发网络报文到下一跳交换机的 packet-out 消息，交换机执行安装该网络流的流规则，并将网络报文转发给下一跳交换机。该网络流后续到达的网络报文匹配上已安装的流规则后直接被转发，不需要再次上报 packet-in 消息给控制器。在转发路径上的每一跳，交换机都按照上述流程处理收到的网络报文。

为了更加高效地实现流规则的匹配，交换机广泛采用三态内容寻址存储器（TCAM）存储和匹配流规则，但是 TCAM 价格昂贵，仅可支持 8 000 条流规则的存储，有限的交换机流表空间容量和控制器的集中式传输控制管理使软件定义网络更易遭受拒绝服务攻击的威胁[5-9]。软件定义网络拒绝服务攻击如图 1 所示，攻击者会随机生成无法成功匹配交换机流规则的大量拒绝服务攻击流量，与正常网络流量混合在一起进入交换机。由于难以识别和管控拒绝服务攻击，交换机不得不缓存所有网络报文，并向控制器上报所有无法匹配流规则的网络报文的 packet-in 消息，使控制器在计算路径和发送安装流规则的 flow-mod 消息、转发网络报文的 packet-out 消息上耗尽 CPU 和缓存资源；同时在交换机有限的流表空间中安装了大量拒绝服务攻击的流规则，短时间内将导致有限的流表空间溢出，无法为正常的网络流量安装流规则，严重影响软件定义网络服务质量。安全危害如下：1) 没有安装流规则的正常网络流将会触发 table-miss 的处理过程，相较于能够匹配流规则被直接转发的处理过程，增加了网络报文的转发时延，降低了网络可用带宽；2) 无法安装流规则的正常网络流的每个网络报文都将会触发一次 table-miss 处理过程，消耗更多的控制器计算和存储资源，放大拒绝服务攻击效果；3) 网络报文转发时延的增加会使越来越多的网络报文停留在交换机有限的缓存中，甚至导致丢包。同时，软件定义网络交换机的流表空间被所有端口共用，每个端口都会独立地为流入的网络新流报文在共享的有限容量的流表中安装流规则。当一个端口出现拒绝服务攻击流量时，会影响其他端口正常网络流的流规则安装。并且攻击者会通过精心设计生成攻击流量，不仅对攻击流量的接入交换机产生拒绝服务攻击效果，也使分布式攻击流量流经同一个中间交换机，形成攻击汇聚，发生再次拒绝服务攻击，导致更严重的拒绝服务攻击效果。

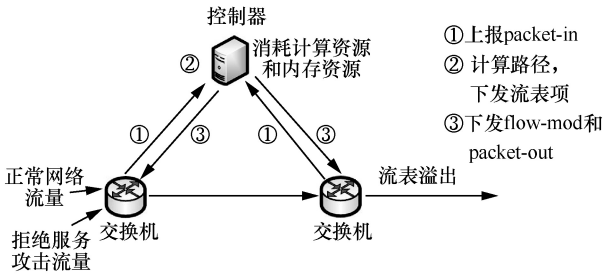


图 1 软件定义网络拒绝服务攻击

为了避免软件定义网络交流机出现流表溢出，确保软件定义网络的稳定运行和网络服务质量，需要解决以下 2 个挑战。

1) 在拒绝服务攻击发生时，如何防止交换机流表空间被攻击流的流规则占满，避免流表溢出。

2) 网络报文在转发路径传输过程中，如何避免在路径上的交换机上出现网络流汇聚导致的再次拒绝服务攻击。

对于第一个挑战，由于与传统网络一样，拒绝服务攻击流量和正常网络流量混在一起，攻击流量很难被准确识别和区分处理[10]，因此无法采用类似攻击

流量清洗的策略来减少安装流规则的数量，并且难以实现安装的流规则都是正常网络流。对于第二个挑战，攻击者会精心构造攻击流量，使分布式的攻击流量汇聚到软件定义网络同一个中间交换机上，可用流表空间少的交换机有可能成为攻击者汇聚攻击流量的目标。

在无法准确识别和消除攻击流量的情况下，本文提出了抗拒绝服务攻击的软件定义网络流表溢出防护技术 FloodMitigation，来避免软件定义网络流表资源消耗和确保网络服务质量，主要的技术贡献如下。

1) 提出了基于可用流表空间的限速流规则管理机制，动态限制出现拒绝服务攻击的交换机端口的流规则安装最大速度，避免交换机流表空间被拒绝服务攻击流规则占满而出现流表溢出的情况，以及影响其他端口正常网络流的流规则安装。优先为出现频次高的网络流安装流规则，减少了上报 packet-in 消息的数量、控制器消耗、网络传输时延和网络报文缓存溢出导致的网络丢包情况。

2) 提出了基于可用流表空间的路径选择，通过在多条转发路径的交换机间均衡流表利用率，避免转发网络报文过程中出现网络新流汇聚在可用流表空间少的交换机上所导致的再次拒绝服务攻击。

3) 对 FloodMitigation 进行了实验验证，实验结果表明，FloodMitigation 能够有效防止交换机流表溢出，降低控制器资源消耗、传输时延和网络丢包，确保网络可用带宽。

1　相关工作

拒绝服务攻击者通常向攻击目标发起大量的伪造网络报文的不同字段、短流的网络攻击流量，消耗目标的服务资源。传统网络的交换机、路由器等传输设备把拒绝服务攻击流量等同于正常网络流量进行转发，不受拒绝服务攻击影响，但是软件定义网络控制器响应式地向交换机下发流规则来转发网络报文，使交换机有限的流表容量和缓存容量、控制器集中式的传输控制管理等成为瓶颈，更易被拒绝服务攻击者利用。很多研究从报文源真实性校验丢弃报文、降低控制器和交换机资源占用、动态调整流表项超时时间等不同角度出发研究拒绝服务攻击防护技术。

Shin 等[11]首次提出了软件定义网络拒绝服务攻击威胁问题，并提出了基于连接迁移的防护机制，通过代理的方式对报文源地址的真实性进行校验，校验通过后转发网络数据报文，能够有效地对有连接状态的传输控制协议（TCP）拒绝服务攻击进行检测和过滤，但是无法应对用户数据报协议（UDP）等无连接状态的拒绝服务攻击，在实际应用中有很大缺陷。Ambrosin 等[12]提出了类似的防御模型，基于代理和黑名单来缓解拒绝服务攻击。面向流表资源和处理资源保护，FloodShield[13]通过源地址验证丢弃伪造地址的网络流量，对于不确定的网络流量，采用基于控制器 CPU 利用率的概率接受方式处理 packet-in 和安装流规则，但是随着网络攻击流的增大，会导致控制器处理资源消耗增加、交换机流表溢出和网络丢包等情况发生。DoSGuard[14]在控制器上保存主机介质访问控制（MAC）地址和交换机的映射关系，当控制器发现从某个端口上报的网络报文的源 MAC 未知时，通过向交换机安装流规则，将未匹配流表的报文全部进行丢弃处理。

面向控制器的资源保护，FloodGuard[15]设置额外的缓存模块，根据网络报文的协议类型采用多队列形式缓存 packet-in 消息，并采用 round robin 算法限速发给控制器处理，保护控制器处理资源。拒绝服务攻击只会影响与其相同类型的协议队列，保证了其他类型协议队列中的 packet-in 正常处理，但是会使与拒绝服务攻击流量在同一个队列的正常网络流量处理时延和丢包。文献[16]设置额外备份控制器，过滤处理疑似恶意流量的 packet-in 消息后，再限速将 packet-in 消息发送给主控制器，从减少 packet-in 消息的角度保护控制器的资源，但无法确保正常网络流量传输时延以及避免丢包。

面向交换机的保护，FloodDefender[17]基于排队论经验公式和链路利用率建立了流量迁移模型，当检测到发生拒绝服务攻击时，交换机根据网络流量信息对 packet-in 消息进行过滤，并依据与其邻接的交换机的可用安全信道容量将网络报文迁移给相邻的交换机，能有效减少该交换机与控制器通过安全信道的数据交互，避免安全信道发生拥塞，但采用的过滤机制将会导致正常网络流量被丢弃。Yuan 等[5]提出在交换机可用流表空间不足时，通过通配符将流量迁移到相邻的伙伴交换机上，利用伙伴交

换机的空闲资源来共同处理网络报文，但无法有效降低控制器的资源消耗。文献[18]处理 packet-in 消息，记录这些报文的 IP，当发往目的 IP 的所有数据包的 Renyi 熵超过阈值时，认为这些报文是攻击流量，对报文进行丢包处理，但无法避免正常网络流量丢包情况。文献[19]将流量分为大象流和老鼠流，不为老鼠流安装流表项，直接通过 packet-out 消息转发，降低对流表的占用，但无法确保转发的攻击流量不流经相同中间交换机，无法避免形成攻击汇聚。

在流表资源方面，目前研究主要采用动态调整超时时间，缩小流表项在 TCAM 中的无效时间，提高流表利用率，但大量的拒绝服务攻击流依然会使流表溢出。文献[20]基于控制器收集流历史信息，对流表项空闲超时时间进行动态设置。HQTimer[21]采用深度强化学习根据当前命中率等信息的反馈进行动态超时时间的决策下发，从而达到更高的流表命中率，但无法避免拒绝服务攻击流使流表溢出。文献[22]通过流表溢出预测将流表项进行主动删除，但会不可避免地将正常网络流表项删除。

现有工作从不同的性能瓶颈角度出发研究拒绝服务攻击防护技术，但不能有效地防止流表溢出，以及防止拒绝服务攻击流量在转发中汇聚导致的再次拒绝服务。本文针对流表溢出防护问题，提出了基于流表可用空间的限速流规则管理，限制出现拒绝服务攻击的交换机端口的流规则最大安装速度和占用的流表空间数量，避免了流表溢出；然后采用基于可用流表空间的路径选择，在多条转发路径的交换机间均衡流表利用率，避免转发网络报文过程中出现网络新流汇聚导致的再次拒绝服务攻击。本文提出的软件定义网络流表溢出防护技术 FloodMitigation 在没有引入额外设备的情况下，实现了在拒绝服务攻击下对流表空间溢出的防护。

2 流表溢出防护

2.1 流表溢出防护系统模型

针对拒绝服务攻击导致的流表溢出问题，本文设计了软件定义网络流表溢出防护 FloodMitigation，并将其作为集成模块运行在控制平面上，包括流监测、限速流规则管理、路径选择这 3 个模块，系统模型如图 2 所示。

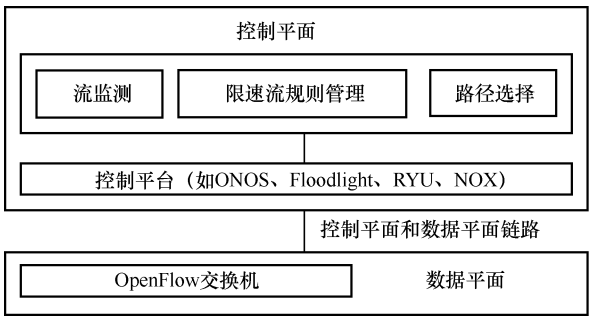


图 2 系统模型

在软件定义网络正常运行时，只有流监测模块在运行，其他拒绝服务攻击防护模块处于待激活状态。当检测到交换机端口上出现拒绝服务攻击时，流监测模块触发限速流规则管理模块，进行流表溢出防护。同时路径选择模块采用负载均衡策略为网络流选择传输路径，避免网络流流经同一个中间交换机导致流表溢出。处理流程如下。

1) 流监测模块统计交换机上报的 packet-in 消息、网络流量统计、计算和内存资源占用率等网络与系统状态，建立基于交换机端口的网络流统计信息表，并进行拒绝服务攻击检测。当出现拒绝服务攻击时，触发其他拒绝服务攻击防护模块。

2) 限速流规则管理模块收到来自发生拒绝服务攻击的交换机端口的 packet-in 消息后，调用路径选择模块计算转发路径，并进行限速安装流规则，通过限制出现拒绝服务攻击的交换机端口安装流规则的最大数量和速度，避免发生拒绝服务攻击的端口的流规则占满整个流表空间，确保其他端口的网络新流的流规则安装使用。

3) 路径选择模块被限速流规则管理模块调用，基于交换机可用流表空间容量来选择转发路径，从多条路径中选取可用流表空间最大的路径。通过均衡多条转发路径的交换机间的流表利用率，避免交换机转发网络报文过程中出现网络新流汇聚再次引发拒绝服务攻击。

2.2 流监测

流监测模块持续监测交换机端口上报的 packet-in 消息速率、网络流量统计，以及控制器内存和计算资源占用率等网络与系统状态，并建立基于交换机端口的网络流统计信息表，结构如图 3 所示。统计信息如下：packet-in 消息中包含的网络流的源地址 scr、目的地址 dst、协议号 protocol、源端口 srcPort、目的端口 dstPort 的五元组信息 tuple5（图 3 中用 tup 表示）；tuple5 内容的哈希值 key 作

为网络流统计项 flow_entry 入口序号；该网络流最近一次 packet-in 到达的时间 time；该网络流的 packet-in 消息的频次 frequency（图 3 中用 freq 表示）。网络流统计信息表用于拒绝服务攻击检测，并被限速流规则管理模块用来决策是否为网络流下发流规则。

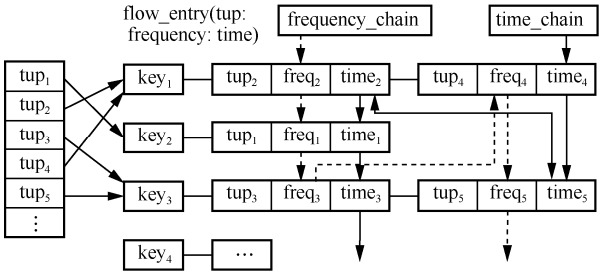


图 3　网络流统计信息表结构

当 packet-in 消息到达时，流监测模块对 packet-in 消息的五元组信息计算哈希值 key，即网络流统计信息表的入口序号。由于多个不同网络流会哈希到同一个哈希值 key，需要精确匹配 packet-in 中的五元组信息，找到对应网络流的流表项 flow_entry，增加该网络流上报 packet-in 消息的频次 frequency，并采用双向链表 frequency_chain 将出现的频次 frequency 进行排序，排序后的 frequency 用于后续限速流规则管理模块对网络流是否下发流规则的决策。同时，更新最近出现时间 time 字段，并采用双向链表 time_chain 将最近更新时间 time 进行排序。如果控制器为网络流向交换机下发安装流规则或者在超时时间内没有新的 packet-in 到达，则该网络流统计项 flow_entry 将被删除。流监测算法伪代码如算法 1 所示。

算法 1　流监测算法

根据交换机上报的 packet-in 消息，查找匹配的流表项 flow_entry

1) 五元组 tuple5 ← src, dst, protocol, srcPort, dstPort in packet-in
2) 五元组哈希值 key = hash (pkt_tuple)
3) for flow_entry in entry [key]
4) if (tuple5 == flow_entry.tuple5)
5) 更新和排序流表项时间 update (flow_entry.time, Now)
6) 更新和排序流表项频次 update (flow_entry. frequency)
7) 返回流表项 flow_entry
8) end if
9) end for
10) 新建流表 flow_entry = build(key, tuple5)
11) 返回流表项 flow_entry

流监测模块采用与 FloodDefender[18]相同的拒绝服务攻击检测方法，通过异常阈值检测识别交换机端口出现的拒绝服务攻击。流监测模块持续统计网络流的 packet-in 消息的频次 frequency，一旦出现拒绝服务攻击，低频次的网络流数量就会快速增加，当超过异常检测阈值时，流监测模块将触发限速流规则管理模块，进行流表溢出防护。当低频次的网络流数量低于异常检测阈值时，表示网络中短流的数量处于安全范围，流监测模块将停止调用限速流规则管理模块，恢复正常处理网络流的状态。

2.3　限速流规则管理

限速流规则管理被触发后，调用路径选择模块对出现拒绝服务攻击流量的交换机端口的 packet-in 计算转发路径，然后根据限速安装速率以及该网络流 packet-in 出现的频次决定是否向交换机下发安装流规则的消息。对于未安装流规则的网络报文，将会向交换机下发 packet-out 消息，直接将网络报文转发给下一跳交换机。

2.3.1　流规则限速安装策略

交换机的流表空间被所有端口共用，为了避免出现拒绝服务攻击的端口的网络流量的流规则完全占满流表空间，影响其他端口的流规则安装，采用基于流表可用空间的限速安装流规则策略，防止流表溢出。t 时刻该端口的流规则最大安装速率 $f_{\max}(t)$为

$$f_{\max}(t)=\frac{\alpha}{T}c(t),0<\alpha<T \tag{1}$$

其中，T 为流规则超时时间，α 为流规则安装速率的调节因子，$c(t)$为 t 时刻交换机可用流表空间大小。

$$c(t)=c(t-1)-f_{\max}(t)+f_{\mathrm{del}}(t) \tag{2}$$

其中，$c(t-1)-f_{\max}(t)$为 $t-1$ 时刻安装流规则后的可用流表空间容量。流规则对应的网络流在超时时间内没有新的网络报文到达时，将会从交换机流表里删除，$f_{\mathrm{del}}(t)$为因流规则超时被删除后 t 时刻可用的流规则数量。在交换机只有一个端口出现拒绝服务攻击流的情况下，由于网络攻击流通常为短流，网络报文数量少，一个拒绝服务攻击持续时间少于单位时间，可认为 $f_{\mathrm{del}}(t)$等于在 $t-T$ 时刻的最大安装速率 $f_{\max}(t-T)$，即

$$f_{\mathrm{del}}(t)=\begin{cases}0,0<t\leqslant T\\ f_{\max}(t-T),t>T\end{cases} \tag{3}$$

设交换机流表空间容量为 C，结合式(1)～式(3)可得

$$c(t)=\begin{cases}\left(\dfrac{T-\alpha}{T}\right)^{t-1}C, 0<t\leqslant T\\ \left(\dfrac{T-\alpha}{T}\right)c(t-1)+\dfrac{\alpha}{T}c(t-T), t>T\end{cases} \tag{4}$$

交换机端口检测到拒绝服务攻击时，根据式(4)，取 T=10 s（控制器 ONOS 的默认超时时间），按照最大速率 $f_{\max}(t)$为该端口的网络流安装流规则，交换机流表空间可用率和流规则安装速率如图 4 所示，流表空间可用率、流规则安装速率保持稳定。随着流规则安装速率的调节因子 α 的增大，流表空间可用率下降、流规则安装速率增长。当流规则安装速率的调节因子 α =1.0 时，有 53%流表可用空间可供其他端口所使用，流规则安装速率约为 5%。限速流规则管理能够有效地限制出现拒绝服务攻击的交换机端口安装流规则的最大数量，防止流表溢出，并确保其他端口的网络流规则的安装使用。

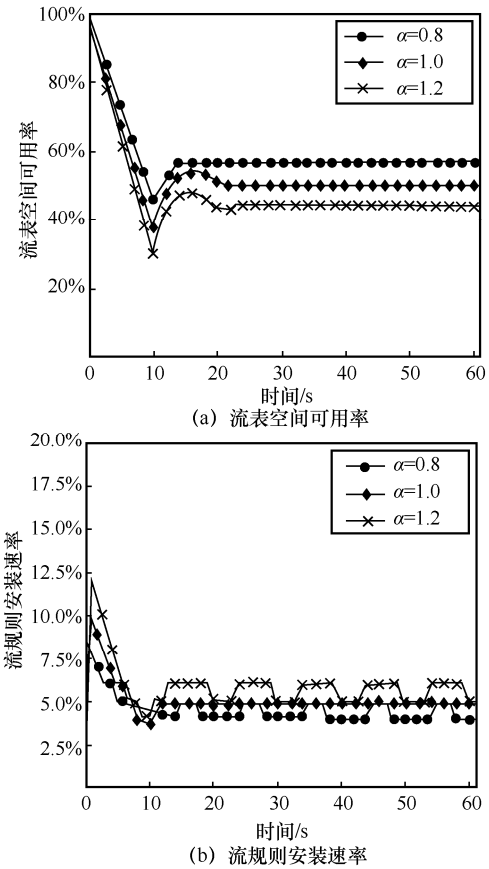


图 4 交换机流表空间可用率和流规则安装速率

2.3.2 流规则安装

控制器收到 packet-in 消息后，首先计算网络报文的转发路径，然后查询该网络流出现的频次。如果该网络流出现的频次位于前 $f_{\max}(t)$位，即该网络流报文出现次数多，则通过 flow-mod 消息向交换机下发安装流规则，并下发 packet-out 消息转发数据报文；否则将不下发安装流规则的 flow-mod 消息，直接通过低开销的 packet-out 转发该网络报文（在 ONOS 官方的性能测试中处理 packet-out 消息的性能比处理 flow-mod 消息的性能高一个数量级），能有效防止流表溢出，并减少网络报文在缓存中的等待时间，确保了网络报文传输时延。流规则安装算法如算法 2 所示。

算法 2 流规则安装算法

根据 packet-in 消息和流监测返回的流表项 flow_entry，给转发交换机下发安装流规则的 flow-mod 消息和转发报文的 packet-out 消息

1) 选择下一跳交换机 next_hop_switch←Path_ Selection(packet-in)
2) if low_entry.frequency ranks top $f_{\max}(t)$
3) 交换机安装流规则 flow-mod(packet-in, next_hop_switch)
4) 删除流表项 remove_entry(flow_entry)
5) end if
6) 转发网络报文给下一跳交换机 packet-out(packet-in，next_hop_switch)

2.4 路径选择

软件定义网络中不同的交换机可用流表空间不同，拒绝服务攻击流量被转发流经剩余流表空间较小的交换机时，更易造成交换机流表溢出、增加网络报文传输时延、消耗控制器资源、丢包等问题。而控制器 ONOS 采用基于最小时延策略的 Djikstra 算法来计算路径，无法确保剩余流表空间较小的交换机不被选进传输路径。在软件定义网络中，源节点至目的节点通常有多条可达路径，路径选择模块计算路径时，从多条路径中选取可用流表空间最大的路径，通过均衡多条转发路径的交换机间的流表利用率，避免转发网络报文过程中出现网络新流汇聚导致的再次拒绝服务攻击，防止再次出现流表溢出情况。

转发路径由路径上的所有交换机组成，根据木桶原理，转发路径的可用流表空间大小取决于路径上所有交换机中可用流表空间容量最小的交换机。为了选出可用流表空间最大的路径，计算每条可选路径的所有交换机的最大可用流表空间容量，从中选出可用流表空间容量最大的路径。

从源地址 src 到目的地址 dst 的可达路径集合为 $P_{src,dst}$，对于一个可达的第 i 条路径 $P_i \in P_{src,dst}$，路径的可用流表空间容量 V_i 为

$$V_i = \min_{S_{i,j} \in P_i}(v(S_{i,j})) \tag{5}$$

其中，$S_{i,j}$ 表示第 i 条路径 P_i 上的交换机集合中的第 j 个交换机，$v(S_{i,j})$表示交换机 $S_{i,j}$ 的可用流表空间容量。

路径选择模块计算源交换机 src 至目的交换机 dst 的路径时，将从多条可达路径中选取可用流表空间最大的路径 P，即

$$P = \{P_i \mid \max(\min(v(S_{i,j}))), S_{i,j} \in P_i, P_i \in P_{src,dst}\} \tag{6}$$

基于可用流表空间的路径选择算法如算法 3 所示。

算法 3　路径选择算法

输入　packet-in 消息中的 src 和 dst

输出　选择的转发路径 P

1) 计算 src 至 dst 的路径集合 $P_{src,dst}$ ← compute_path(src，dst)
2) 初始化最大可用流表空间容量 $v_{max} = 0$
3) for P_i in $P_{src,dst}$
4)　初始化路径最小流表空间容量 $v_{min} = \infty$
5)　for $S_{i,j}$ in P_i
6)　计算路径上的交换机最小流表空间 $v_{min}=\min(v(S_{i,j}), v_{min})$
7)　end for
8)　if $v_{max} < v_{min}$
9)　更新路径最大流表空间容量 $v_{max} = v_{min}$
10)　更新选择的转发路径 $P = P_i$
11)　end if
12) end for
13) 返回选择的转发路径 P

路径选择模块从所有可达路径中，选择可用流表空间最大的路径来传输网络新流，实质上是均衡了所有可达路径的交换机间的可用流表空间，避免交换机转发网络报文过程中出现网络新流汇聚导致的再次拒绝服务攻击，防止再次出现流表溢出情况。

3　实验评估

本节通过实验对提出的抗拒绝服务攻击的软件定义网络流表溢出防护 FloodMitigation 进行有效性验证，验证其在拒绝服务攻击发生时对交换机流表利用率、时延、丢包率、带宽可用率、控制器 CPU 利用率等的保护。使用 ONOS 2.2.1 作为软件定义网络控制器，软件定义网络仿真实验平台 Mininet 创建的 Open vSwitch 2.10.1 作为数据平面交换机，并依据 OpenFlow 硬件交换机 Polaris xSwitch X10-24S2Q 的配置将交换机的流表空间容量设定为 2 000 条。Mininet 和 ONOS 分别独立运行在 Ubuntu 16.04 LTS 系统的 OpenStack 实例中，实例运行环境为 XEON E5-2630 v3 CPU，内存为 20 GB。

在 Mininet 中创建与 FloodShield 类似的实验拓扑，如图 5 所示，拓扑中共有 20 台交换机（S_1～S_{20}）和 16 台主机（H_1～H_{16}）。主机 H_1、H_3、H_5、H_7、H_9、H_{11}、H_{13}、H_{15} 使用 Mawi 流量数据集生成访问其他主机的网络流量。同时由主机 H_2 向 H_{16} 发送拒绝服务攻击流量，源端口号、目地端口号均为随机变化。在实验中，当流规则安装速率的调节因子 $\alpha = 1.0$ 时，将所提 FloodMitigation 与 OpenFlow、FloodShield 在同等实验环境中进行对比实验，验证防护效果。

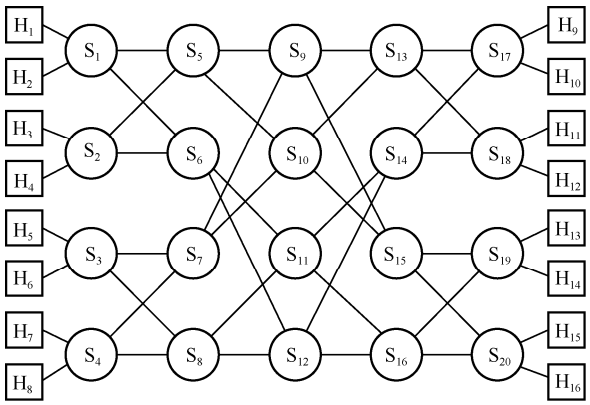


图 5　实验拓扑

3.1　基于限速安装流规则机制的流表利用率

当主机 H_2 向主机 H_{16} 发送的攻击速率分别为 1 000 flow/s、2 000 flow/s 时，交换机 S_1 的流表利用率如图 6 所示。由于 OpenFlow 没有安全防护，出现了流表溢出。FloodShield 采用基于控制器 CPU 利用率以概率接受的方式安装流规则的防护方式，但是没有考虑流表利用率，依然出现了流表溢出。FloodMitigation 由于采用了基于流表可用空间的限速安装流规则机制，能够有效地限制出现拒绝服务攻击的交换机端口安装流规则的最大数量，防止流表溢出，并且在 2 种不同攻击速率下，流表利用率保持稳定。

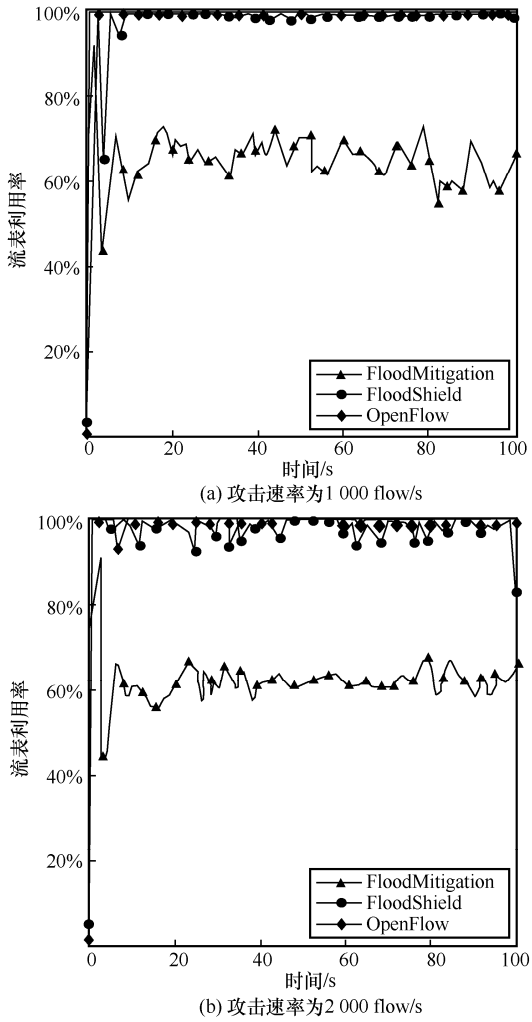


图6　不同攻击速率下交换机 S_1 的流表利用率

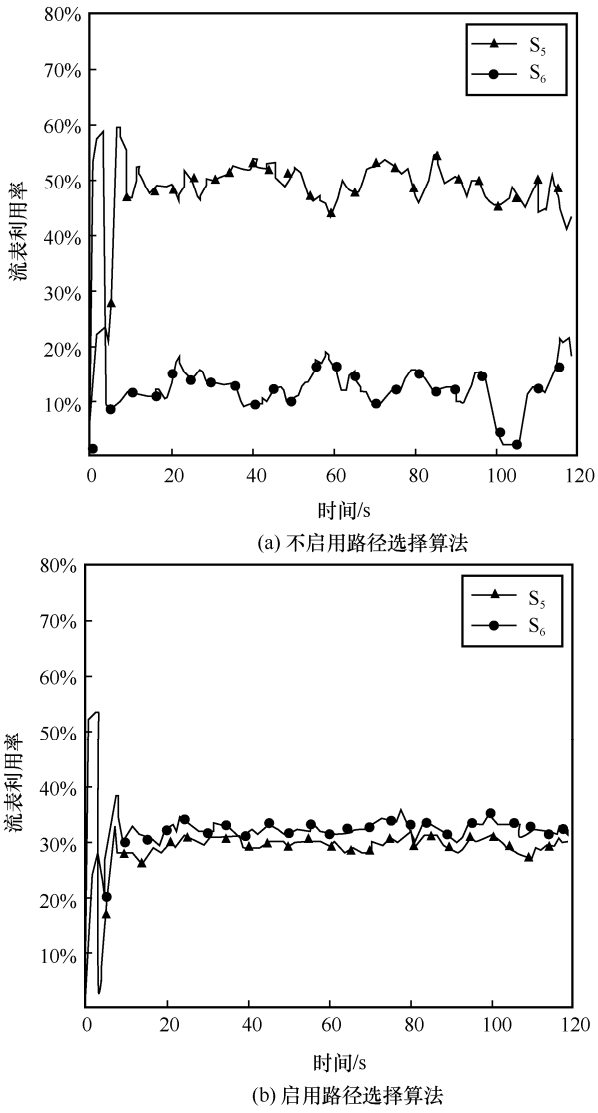


图7　交换机 S_5 和 S_6 的流表利用率

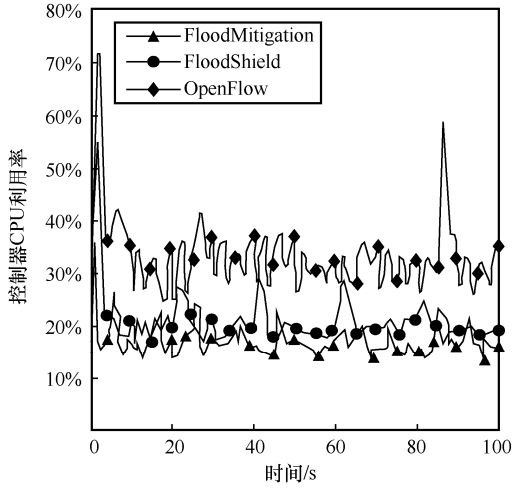
3.2　基于路径选择的流表利用率

本节实验通过交换机间的流表利用率，验证所提路径选择在均衡交换机流表利用率上的有效性。实验中由 H_2 向 H_{16} 发送攻击流量，网络流量将流经路径 H_2—S_1—S_5 或 H_2—S_1—S_6。

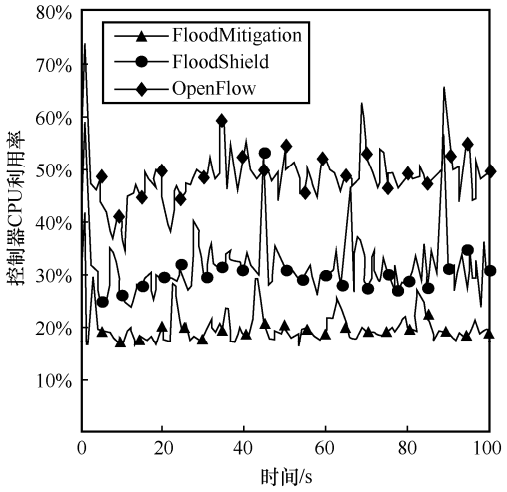
图 7(a)为在不启用路径选择算法情况下交换机 S_5 和 S_6 的流表利用率，S_6 的流表利用率约为 15%，而 S_5 的流表利用率约为 50%。这是因为控制器将 H_1 发送的部分背景测试流量，以及流经交换机 S_1 的攻击流量都转发给了交换机 S_5。图 7(b)为启用路径选择算法情况下交换机 S_5 和 S_6 的流表利用率，S_5 和 S_6 的流表利用率接近。这是由于路径选择模块在每次选择路径时，将从多条路径中选取可用流表空间最大的路径，能够有效地实现多条转发路径的交换机间的流表利用率的均衡，避免交换机转发网络报文过程中出现网络新流汇聚导致的再次拒绝服务攻击，实现对流表的有效保护。

3.3　控制器 CPU 利用率

当拒绝服务攻击速率分别为 1 000 flow/s、2 000 flow/s 时，控制器 CPU 利用率如图 8 所示。由于 OpenFlow 没有防护机制，其控制器 CPU 利用率最高。FloodMitigation 始终比 FloodShield 的控制器 CPU 利用率低，并且随着攻击速率的增加，FloodShield 的控制器 CPU 利用率增加幅度比 FloodMitigation 大。这是因为 FloodShield 基于控制器 CPU 利用率以概率接受的方式通过 flow-mod 安装流规则，随着攻击速率的增加，需要下发流规则的处理增多，CPU 利用率增加。而 FloodMitigation 采用基于流表可用空间的限速安装流规则策略，能够保持稳定的流规则下发安装速率，随着攻击速率的增加，FloodMitigation 的控制器 CPU 利用率增加幅度小。



(a) 攻击速率为1 000 flow/s

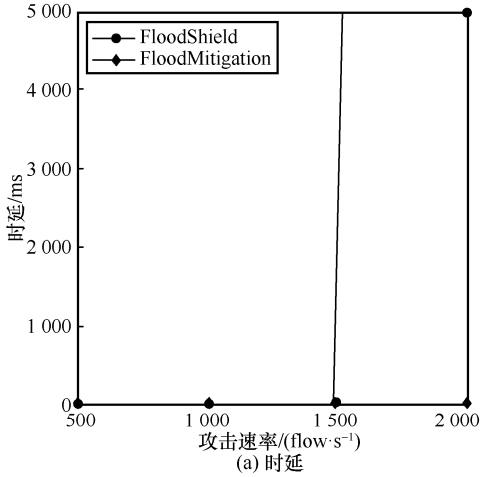


(b) 攻击速率为2 000 flow/s

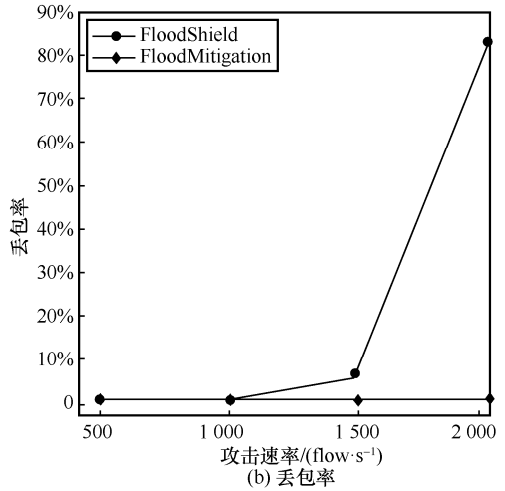
图 8　不同攻击速率下的控制器 CPU 利用率

3.4　时延、丢包率、带宽可用率

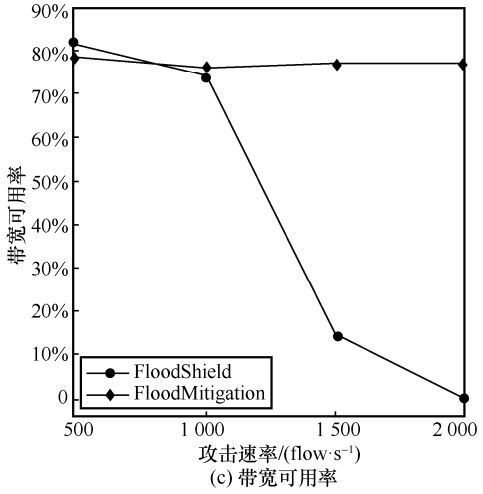
本节实验中 H_2 发送拒绝服务攻击流量，测量 H_1～H_9 的时延、丢包率、带宽可用率，结果如图 9 所示。H_1 发送 20 条 UDP 流，每条 UDP 流有 1 000 个网络报文，统计接收到的网络报文数量以及平均传输时延，并在 H_1 上使用 iperf 工具测试可用带宽。FloodShield 采用基于控制器 CPU 利用率以概率接受的方式安装流规则，随着攻击流量速度的增加，CPU 利用率增加，处理的 packet-in 消息减少，缓存的 packet-in 消息逐渐增多直至缓存溢出，出现丢包率增加和带宽可用率减少的情况。而 FloodMitigation 将无法安装的网络报文直接通过低开销的 packet-out 消息转发，传输时延保持稳定，没有出现因缓存溢出进而导致的丢包率增加和带宽可用率减少的情况。



(a) 时延



(b) 丢包率



(c) 带宽可用率

图 9　时延、丢包率和带宽可用率

4　结束语

现有工作从不同的性能瓶颈角度出发研究拒绝服务攻击防护技术，但未能有效防护流表溢出，以及防止拒绝服务攻击流量在转发中汇聚导致的再次拒绝服务。本文提出了软件定义网络流表溢

出防护技术 FloodMitigation。通过基于可用流表空间的限速流规则管理机制，动态限制出现拒绝服务攻击的交换机端口的流规则安装最大速度，避免拒绝服务攻击流量占用被所有端口共用的流表资源，以及避免影响其他端口的流规则安装；同时基于可用流表空间的路径选择在多条转发路径的交换机间均衡流表利用率，避免网络新流汇聚在可用流表空间少的交换机上所导致的再次拒绝服务攻击。实验结果表明，在没有引入额外设备的情况下，FloodMitigation 在拒绝服务攻击下能够有效防止流表空间溢出的防护，并在避免网络报文丢失、降低控制器资源消耗、确保网络报文转发时延等方面有效抵御拒绝服务攻击。下一步的工作将结合流规则动态管理机制，实现对有限的流表资源的有效利用，并在硬件交换机上验证拒绝服务攻击防护效果。

参考文献：

[1] JAIN R. Internet 3.0: ten problems with current Internet architecture and solutions for the next generation[C]//Proceedings of IEEE Military Communications Conference. Piscataway: IEEE Press, 2007: 1-9.

[2] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[3] 张朝昆, 崔勇, 唐骞骞, 等. 软件定义网络(SDN)研究进展[J]. 软件学报, 2015, 26(1): 62-81.
ZHANG C K, CUI Y, TANG H H, et al. State-of-the-art survey on sofware-defined networking (SDN)[J]. Journal of Software, 2015, 26(1):62-81.

[4] QI Y Z, WANG D B, YAO W B, et al. Towards multi-controller placement for SDN based on density peaks clustering[C]//Proceedings of IEEE International Conference on Communications. Piscataway: IEEE Press, 2019: 1-6.

[5] YUAN B, ZOU D Q, YU S, et al. Defending against flow table overloading attack in software-defined networks[J]. IEEE Transactions on Services Computing, 2019, 12(2): 231-246.

[6] KAUR S, KUMAR K, AGGARWAL N, et al. A comprehensive survey of DDoS defense solutions in SDN: taxonomy, research challenges, and future directions[J]. Computers and Security, 2021, 110: 1-39.

[7] SWAMI R, DAVE M, RANGA V. Software-defined networking-based DDoS defense mechanisms[J]. ACM Computing Surveys, 2020, 52(2): 1-36.

[8] ZHANG P, WANG H Z, HU C C, et al. On denial of service attacks in software defined networks[J]. IEEE Network, 2016, 30(6): 28-33.

[9] 王蒙蒙, 刘建伟, 陈杰, 等. 软件定义网络: 安全模型、机制及研究进展[J]. 软件学报, 2016, 27(4): 969-992.
WANG M M, LIU J W, CHEN J, et al. Software defined networking: security model, threats and mechanism[J]. Journal of Software, 2016, 27(4): 969-992.

[10] TANG D, ZHANG S Q, YAN Y D, et al. Real-time detection and mitigation of LDoS attacks in the SDN using the HGB-FP algorithm[J]. IEEE Transactions on Services Computing, 2022, 15(6): 3471-3484.

[11] SHIN S, YEGNESWARAN V, PORRAS P, et al. AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks[C]//Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. New York: ACM Press, 2013: 413-424.

[12] AMBROSIN M, CONTI M, GASPARI F D, et al. LineSwitch: tackling control plane saturation attacks in software-defined networking[J]. IEEE/ACM Transactions on Networking, 2017, 25(2): 1206-1219.

[13] ZHANG M H, BI J, BAI J S, et al. FloodShield: securing the SDN infrastructure against denial-of-service attacks[C]//Proceedings of the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). Piscataway: IEEE Press, 2018: 687-698.

[14] LI J S, TU T F, LI Y S, et al. DoSGuard: mitigating denial-of-service attacks in software-defined networks[J]. Sensors (Basel, Switzerland), 2022, 22(3): 1061.

[15] WANG H P, XU L, GU G F. FloodGuard: a DoS attack prevention extension in software-defined networks[C]//Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. Piscataway: IEEE Press, 2015: 239-250.

[16] WU P P, YAO L, LIN C, et al. FMD: a DoS mitigation scheme based on flow migration in software defined networking[J]. International Journal of Communication Systems, 2018, 31(9): 1206-1219.

[17] GAO S, PENG Z, XIAO B, et al. Detection and mitigation of DoS attacks in software defined networks[J]. IEEE/ACM Transactions on Networking, 2020, 28(3): 1419-1433.

[18] 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: 2214-2226.

[19] HE H, PENG Z Z, ZHOU X H, et al. LFOD: a lightweight flow table optimization scheme in SDN based on flow length distribution in the Internet[C]//Proceedings of the 23rd Asia-Pacific Network Operations and Management Symposium. Piscataway: IEEE Press, 2022: 1-6.

[20] LI X F, HUANG Y. A flow table with two-stage timeout mechanism for SDN switches[C]//Proceedings of IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). Piscataway: IEEE Press, 2019: 1804-1809.

[21] LI Q, HUANG N Y, WANG D M, et al. HQTimer: a hybrid Q-learning-based timeout mechanism in software-defined networks[J]. IEEE Transactions on Network and Service Management, 2019, 16(1): 153-166.

[22] XIE S X, XING C Y, ZHANG G M, et al. A table overflow LDoS attack defending mechanism in software-defined networks[J]. Security and Communication Networks, 2021, 2021: 1-16.

[作者简介]

王东滨（1978– ），男，黑龙江哈尔滨人，博士，北京邮电大学教授、博士生导师，主要研究方向为软件定义网络与安全、区块链、网络流量分析与模拟等。

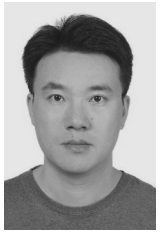
吴东哲（1998– ），男，河南三门峡人，北京邮电大学硕士生，主要研究方向为软件定义网络与安全等。

智慧（1980– ），女，河北邯郸人，中国民航信息网络股份有限公司高级工程师，主要研究方向为物联网、RFID、分布式系统、民航信息系统等。

郭昆（1986– ），男，河北沙河人，博士，中关村实验室助理研究员，主要研究方向为可信路由计算、异常流量检测。

张勖（1973– ），女，北京人，博士，北京邮电大学副教授、硕士生导师，主要研究方向为移动自组织网络安全、物联网安全等。

时金桥（1978– ），男，黑龙江哈尔滨人，博士，北京邮电大学教授、博士生导师，主要研究方向为分布式网络系统、匿名通信与隐私保护、区块链网络、信息智能处理、大数据智能分析、网络测量技术等。



张宇（1979– ），男，黑龙江哈尔滨人，博士，哈尔滨工业大学副教授、博士生导师，主要研究方向为软件定义网络、网络拓扑测量、域间路由和复杂网络等。

陆月明（1969– ），男，江苏苏州人，博士，北京邮电大学教授、博士生导师，主要研究方向为网络安全防护、信任体系等。