基于 NFV 的防范 SDN 控制器中 UDP 控制分组冗余的机制

薛 昊 陈 鸣 钱红燕
(南京航空航天大学计算机科学与技术学院 南京 211106)

摘 要 尽管软件定义网络(Software Defined Networking,SDN)的安全性得到了极大的关注,但 SDN 控制器受大流 UDP 冗余分组威胁的问题并没有得到有效解决。对此,基于 SDN 和网络功能虚拟化(Network Function Virtualization,NFV)技术的特点,结合 SDN 控制器处理 UDP 和 TCP 两种数据流时的负载状况,首先提出了一种新型的基于 NFV 的防范 SDN 控制器中 UDP 冗余分组的机制,前置于 OpenFlow 交换机口的检测中间盒能够有效地检测并滤除 UDP 流冗余分组;其次,提出了一种经济有效的基于 NFV 的检测中间盒的实现方法,使用 Linux 容器实现检测中间盒,在 SDN 控制器下发流表之前只允许 UDP 流首分组通过中间盒,保证后续 UDP 流分组在到达 OpenFlow 交换机时已经有相关的流表项存在;最后,在 Linux 服务器中实现了基于该机制的原型系统并进行实验。结果表明,当非首分组的时延 t 大于或等于控制器处理单个分组的时间时,该方法能够有效地解除 UDP 冗余分组的威胁。

关键词 网络安全,软件定义网络,网络功能虚拟化,检测中间盒,UDP
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/jsjkx.180901659

NFV-based Mechanism to Guard Against UDP Control Packet Redundancy in SDN Controller

XUE Hao CHEN Ming QIAN Hong-yan
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

Abstract Although the security of software-defined networking (SDN) obtains great attention, the threat of SDN controllers from the UDP duplicate packets in a heavy flow has not been eliminated yet. In response, based on the features of SDN and network function virtualization (NFV) technology, combining the load condition of SDN controller in handling both UDP and TCP data streams, firstly, this paper proposed a new NFV-based mechanism to guard against UDP control packet redundancy in SDN controller. The detection middlebox located in front of the OpenFlow switch interface can detect and filter UDP duplicate packets effectively. Secondly, this paper put forward a cost-effective NFV-based implementation method of detection middlebox. The detection middlebox is implemented by the Linux container and only the first UDP flow packet is allowed to pass through before a path is established by the SDN controller, ensuring that subsequent UDP flow packets already have relevant flow table entry when they reach the OpenFlow switch. Finally, this paper implemented and tested the prototype system of the mechanism in Linux server. The experimental results demonstrate that the method can effectively free from threat of the UDP redundant packets when the setting of the delay t of non-first packets is larger than or equal to the time for controller processing a single packet.

Keywords Network security, Software defined network, Network function virtualization, Detection middlebox, UDP

1 引言

软件定义网络[1]是一种将控制功能和转发功能相分离的网络体系结构,其集中控制方式使得网络管理与应用业务的配置更加灵活和便捷,也便于部署网络新技术、新协议,促进了网络创新。然而,SDN 在获得了空前发展的同时,其安全性也引起了越来越多的关注[2-3]。由于 SDN 网络的运转完全依赖于控制器执行集中式的决策,如果控制器由于某种原因而无法正常工作甚至瘫痪,整个网络都将随之崩溃。因此,作为 SDN 网络的核心,SDN 控制器的安全性成为了研究的焦点之一。

网络中的所有应用流都可以划分为 TCP 流和 UDP 流两类,但一种看似简单却尚未得到解决的问题是 SDN 网络如何有效地处理 UDP 流。根据 SDN 网络的工作原理,当某 UDP 流的首个分组到达某 SDN 网络(如进入一个 OpenFlow 交换机)时,该 OpenFlow 交换机没有与该流相匹配的流表项,此时交换机就会将该分组封装成 Packet_In 报文转发至 SDN 控制器,交由控制器处理。由于 UDP 流具有无连接特性,在控

到稿日期:2018-09-05 返修日期:2019-03-15 本文受国家自然科学基金(61772271,61379149)资助。
薛 昊(1991-),男,硕士生,主要研究方向为软件定义网络和网络安全,E-mail:xhlcxx@163.com;陈 鸣(1956-),男,博士,教授,博士生导师,CCF 高级会员,主要研究方向为网络结构、网络测量和未来网络,E-mail:mingchen@nuaa.edu.cn(通信作者);钱红燕(1973-),女,博士,副教授,硕士生导师,CCF 会员,主要研究方向为无线网络和信息安全。

制器为该流下发并安装传输路径的流表之前，该流可能已经向网络中发送了大量分组，这些分组也都将被 OpenFlow 交换机以 Packet_In 报文的形式转发给控制器，因此控制器会收到大量冗余的 Packet_In 报文。然而，控制器处理这些冗余 UDP 分组将严重浪费资源，降低网络的性能。当多条 UDP 大流（无论是正常的业务流还是恶意的攻击）同时进入该网络时，将导致 SDN 控制器的缓存被大量冗余 UDP 分组填满，SDN 网络根本无法为其他正常的流提供服务，从而呈现出拒绝服务（DoS）攻击的状况。当 SDN 控制器处理 TCP 流的首分组时，该流的后继分组会等待控制器建立转发路径，当路径建好后，后继分组才会被继续发送。相比 SDN 网络对 TCP 流的处理流程，UDP 控制分组冗余问题的关键在于：UDP 流的后继分组不会等待，而是直接进入控制器缓存形成冗余分组，显然这些 UDP 冗余分组对控制器工作是无用且有害的。因此，设计一种类似处理 TCP 流的完美工作机制来处理 UDP 流是本文的目标。

目前，解决 UDP 冗余分组的方法大概包括：在控制平面的控制器处消除冗余分组，或限制流的初始速率，以及在数据平面控制冗余分组[4]。然而，这些方法要么要求修改交换机软、硬件结构，与 SDN 标准不相符合，要么要求限制流的初始速率，降低了网络性能，且无法避免后继大量冗余分组的侵扰。首先，如何创新性地设计一种理想检测控制机制是一项技术挑战；其次，采用何种技术来实现这种机制也是一个技术难题。本文的主要贡献如下：通过分析 SDN 控制器的工作过程，提出了一种新型的防范 UDP 冗余分组的机制；根据网络功能虚拟化[5]和 SDN 的特征，提出了一种基于 NFV 实现上述机制的技术，通过将 SDN 和 NFV 相结合给出了防范 UDP 冗余分组的方案。原型系统的实验结果表明了该机制和技术的有效性。

2 相关工作

在 SDN 网络中，当有新流出现时，控制器就要采用集中式控制决策方式分析新流并为其安装流表。当到达的新流数量很大时，控制器可能会由于负载过大而处于低效或者瘫痪状态，成为整个 SDN 网络的瓶颈。文献[6]指出 SDN 控制器很难对 10 Gbps 网络中产生的大量新流进行合理和及时的处理；文献[7]指出在遇到攻击时 SDN 控制器受到的威胁更大，需要提高 SDN 控制器资源的可扩展性以应对攻击；文献[8]指出无连接数据流会向控制器发送大量的冗余分组等待控制器处理，从而消耗大量控制器资源，危害 SDN 网络的安全。

为了减小无连接数据流对控制器性能的影响，文献[4]提出了在控制平面和数据平面分别消除冗余控制分组的方法。在控制平面通过维护最新 Packet_In 报文视图，使冗余控制分组在控制器端直接被丢弃。该方法在流量以及网络规模较大时，不能避免大量冗余控制分组占用控制器资源；在数据平面，通过对 OpenFlow 交换机进行修改，将同一条流的冗余控制分组暂时缓存至 SDN 交换机，使每一条新流仅向控制器发送一个 Packet_In 报文，但该方法要求修改 OpenFlow 交换机，需要在交换机上增加复杂的流识别功能，将部分控制功能下放至交换机中，同时需要交换机缓存大量的冗余分组，通过牺牲交换机性能来缓解控制器压力。因此，该方法的可部署性与可扩展性不强。文献[9]指出数据平面流数目的急剧增加将会导致交换机向控制器发送大量的流请求分组，并最终导致控制器崩溃。文献[10]针对单个 SDN 控制器不能正常工作的问题，提出使用备用控制器来保障整个网络的恢复能力，但是使用备用控制器仍会收到大量 Packet_In 报文。该方法只能缓解但无法解决大量冗余分组问题。文献[11]提出了一种基于预装转移流表并滤除冗余分组（Preinstalling Flow-Tables & Filtering Redundant-Packets，PFFR）的机制，通过预装转移流表来限制 UDP 控制分组的初始速率，并通过冗余分组滤除算法快速消除冗余分组报文。该方法限制了大流的初始速率。

近年来，NFV 和相关技术在很多新兴领域都展现出了独特的魅力和良好的应用前景[5]。NFV 的重点在于解耦网络中原有的提供各种网络功能的专用设备，将设备中的硬件部分与提供网络功能的软件部分相分离，软件都以虚拟网络功能的形式运行在统一标准的大容量服务器上。NFV 可以动态地配置、启动网络功能，分配基础设施的资源，调节各个网络功能的位置而无须添加新的硬件设备，有效地提高了网络环境的开放性和可扩展性，降低了网络设备投资和网络运营的成本，缩短了新技术的引进周期，加快了网络创新和新服务部署的速度。

3 控制器防护 UDP 冗余分组的机制

3.1 SDN 网络的工作流程

为了便于分析，图 1 给出一个典型的 OpenFlow 网络实例，其中 SDN 控制器负责处理来自交换机的用户报文并下发路径流表项，使用户能够访问 NFV 网络中的某主机（如 Web 服务器）。

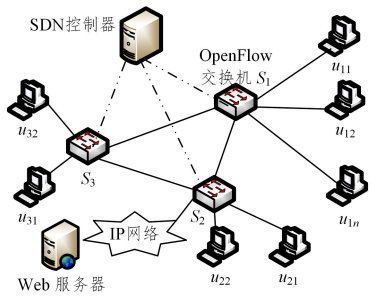

图 1 OpenFlow 网络实例

Fig. 1 Example of OpenFlow network

图 2 为在上述 SDN 网络中处理某用户的 UDP 流首分组的时序图。1）用户 u_{11} 发送流首分组到接入 OpenFlow 交换机 S_1 某端口的时延为 t_1。2）S_1 处理该分组的时延为 t_2。3）SDN 控制器处理该分组的时延为 t_3。一旦该首分组被拒绝，该流将终止；若该首分组被接受，控制器将向相关交换机（如 S_1 和 S_2）下发流表。4）该流的后继分组将以时延 t_4 到达 IP 网络中的某 Web 服务器。其中，t_1 是用户发送分组的时延加上分组传输到 S_1 的时延，该时延为微秒量级；t_2 是交换机将分组与流表项进行匹配并将其向控制器转发的时延，该时延为微秒量级；t_3 是控制器处理该分组的时延，该时延为毫秒量级；t_4 为微秒量级。t_1，t_2 和 t_4 发生在数据平面，t_3 发生在控制平面，一般有 $t_1 < t_4 < t_2 \ll t_3$。

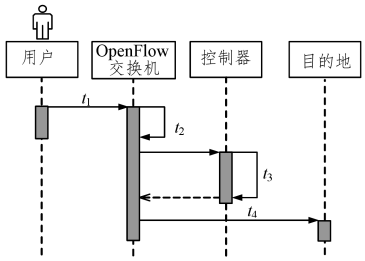

图 2　流首分组的时延分析

Fig. 2　Time delay analysis of first packet of flow

当 SDN 网络遇到大量 UDP 流时,从 SDN 控制器处理第一个 UDP 分组至下发流表到 OpenFlow 交换机的这段时间内,由于交换机尚无法为该流提供转发路径,该流的大量 UDP 分组由交换机不断地转发给 SDN 控制器,致使 SDN 控制器中积聚了大量冗余 UDP 分组。如果这些冗余分组过多,后续其他流的控制分组将不得不在控制器缓存中被丢弃,从而极大地增加了这些流在控制器中分组处理的时延 t_3。

3.2　处理 UDP 冗余分组的优势位置

本文将控制器检测处理 UDP 冗余分组的相关功能称为检测中间盒,从上述分析可见,将该检测中间盒放置在控制器中并非一种好的选择。因为控制器是整个 SDN 网络的中枢,所有到达 SDN 网络新流的首分组都会进入 SDN 控制器,并由控制器来分析处理。因此将检测中间盒部署在控制平面的好处是:正常情况下方案的性价比高,无须改变已有设备,也无须添加额外的网络设备,只须增加相关的软件程序即可。但对于 UDP 大流,该方案势必造成控制器缓存的溢出和控制器的 CPU 重负荷。将检测中间盒部署在数据平面,通常有两种方法:1)修改 OpenFlow 交换机的代码,将检测中间盒部署在 OpenFlow 交换机之中;2)在数据传输路径中增加检测中间盒,通过分析传输的流的分组,检测冗余 UDP 分组并进行处理。前一种方法要修改交换机的硬件和软件,使得交换机不再符合标准规范,因此不考虑这种方案;对于后一种方法,若将检测中间盒置于用户主机与交换机端口之间,中间盒检测到某流首分组后,将其转发到控制器,而用缓存区缓存后继 UDP 分组并延时 Δt_1,等待 SDN 控制器处理完首分组并建立起该流的传输路径后,再将缓存中的后继 UDP 分组放行,从而实现类似于处理 TCP 流的理想结果。后一种方法通过将检测处理功能转移到每条数据流的源头,以并行中间盒的方法消除 SDN 网络存在的集中式控制器的性能瓶颈。为了便于讨论,本文将此方法称为中间盒前置法。表 1 比较了中间盒前置法与其他典型相关机制和方法的主要差异。

表 1　相关机制和方法的比较

Table 1　Comparison of relative mechanisms and methods

方法	处理位置	对交换机的影响	对控制器的影响
交换机防御[4]	交换机	大	无
控制器防御[4]	控制器	无	大
PFFR 机制[11]	交换机+控制器	小	小
中间盒前置法	前置中间盒	无	无

虽然中间盒前置法能够较好地消除对交换机和控制器的影响,但是采用传统技术实现中间盒前置法将存在严重的性能、价格问题:n 个用户接入 SDN 网络需要配置 n 个检测中间盒。本文将在第 4 节讨论解决该问题的具体技术方案。

3.3　中间盒前置法的工作流程

在用户主机与 SDN 网络交换机端口之间放置检测中间盒,所有发送至 SDN 网络的 UDP 分组都会通过该中间盒进行处理。该检测中间盒的工作流程如算法 1 所示。

算法 1　前置检测中间盒处理

输入:流 F 的分组 p,时延时间 Δt

输出:整形的 UDP 流序列

```
1. UDP_View←Ø
2. for 每一条新到的分组 p do;
3.    if p 是一条 UDP 分组 then
4.    通过分组 p 提取 F 的元组散列值 h;
5.    if h 不属于 UDP_View then
6.       UDP_View←UDP_View∪h;
7.       continue;
8.    else
9.       该流延时 Δt1;
10.            转发该流所有缓存分组;
11.      end if
12.   end if
13. end for
```

该算法要求中间盒维护一张 UDP 分组的元组视图 UDP_View,每个新流的首分组的 UDP 元组信息的散列值将存储于该视图中。每当该流 UDP 分组到达中间盒后,通过比对该视图是否包括该散列值 h 来确定该 UDP 分组是否是首分组。若 UDP_View 中没有该散列值,表明它是该流的首分组,则将该值保存在视图中,并直接将该 UDP 分组转入 SDN 网络,OpenFlow 交换机将其发送至控制器,再由控制器计算路径、下发路径流表;若 UDP_View 中有新到达 UDP 分组的散列值,则表明它不是该流首分组,中间盒对该 UDP 分组进行缓存,控制器为该流建立好传输路径之后,中间盒向 SDN 网络转发该流的所有后续 UDP 分组。考虑到 UDP 流标识可以重用,当某 UDP 流闲置一段时间 Δt_2(如 18 s)后,就将 UDP_View 中的相应散列值删除。具体处理方法是:为 UDP_View 中的散列值设置计时器,若经 Δt_2 时长仍无 UDP 分组的散列值与之相匹配,则删除视图中的散列值。Δt_2 等于 OpenFlow 交换机中流表项保持的时间。算法 2 给出了具体操作步骤。

算法 2　UDP_View 视图无效项处理

输入:UDP_View 中所有散列值 h,保存时间 Δt_2

输出:已去除无效项的 UDP_View

```
1. 对于每个散列值 h,有等待计时器 t,每当 h 有匹配项时,将 t 置 0;
2.    while (1) do;
3.       if t<Δt2 then
4.             t=t+1;
5.             sleep(1);
6.       else
7.             UDP_View←UDP_View-h;
8.       end if
9.    end while
```

算法 1 为每个 UDP 流首分组的后续分组设置了时延,增加了该 UDP 流在中间盒的传输时延,正是该时延使得到达控制器的 UDP 冗余分组减少为 0,其将控制平面的压力分散到数据平面的各个输入端口,从源头上保证了控制器免受 UDP

大流的冗余分组侵扰或拒绝服务攻击,增强了 SDN 网络的安全性与稳定性。

4 一种基于 NFV 的中间盒前置技术

采用何种技术来实现中间盒前置法是本文研究的另一个重要问题。理论上,采用专用设备运行特定软件来实现检测中间盒是可行的,但其成本可能无法接受。NFV 能够利用服务器强大的计算能力以软件形式来实现原本需要依赖于硬件的网络功能,显著地降低了成本并简化了管理[12]。因此,本节提出一种 NFV 与 SDN 相结合的技术来实现中间盒前置技术。首先,设计了一种 NFV 与 SDN 综合的网络环境;然后,设计并实现了一种检测 UDP 流的中间盒。

4.1 设置 NFV 与 SDN 综合的环境

NFV 与 SDN 综合的环境在各个领域都有广泛的应用[13-14],通常存在两种情况:1)基于专用设备的 SDN 网络与基于虚拟化的 NFV 的综合,属于虚拟设备与实体设备之间的综合;2)基于虚拟化的 NFV 和 SDN 之间的综合,属于虚拟设备之间的综合。由于现有的虚拟化技术都提供了虚拟设备与实体设备之间交互的方法,因此从交互方式而言,上述两种情况并无本质不同,下面以虚拟化环境为例进行讨论。

(1)选用计算能力强大的服务器作为承载虚拟计算环境的宿主机,在其上运行 Linux 操作系统以提供虚拟化计算环境,用以支撑 SDN 和 NFV 的网络环境。

(2)生成配置网络虚拟机。优选高效、低耗的虚拟机(如 Linux 容器,LXC[15]),通过为虚拟机配置 IP 地址、路由选择协议、性能参数等,使其成为某种网络功能的虚拟网络设备,如虚拟路由器、虚拟主机等。

(3)生成网络安全挑战之一的网络中间盒[16]。在上述虚拟网络设备上安装特定网络功能软件,使其成为具有特定网络功能的虚拟网络中间盒,如防火墙、入侵检测系统或本文所需的 UDP 分组检测中间盒等。

(4)部署 SDN 网络、互联网络虚拟设备和网络中间盒。根据应用需求,通过在虚拟计算环境部署虚拟 OpenFlow 交换机,将所有虚拟网络设备与相应的 OpenFlow 交换机相连;安装 SDN 控制器,设置 OpenFlow 交换机到 SDN 控制器的通信链路,使 SDN 控制器与 OpenFlow 交换机保持连接,形成 NFV 与 SDN 综合的网络环境。注意:SDN 控制器既能安装在实体主机上,也能安装在虚拟主机上。

NFV 的灵活性,使得不仅可以将虚拟网络中间盒部署在虚拟网络的各个部分,还可以将虚拟网络中间盒与真实网络进行虚实互连。将运行虚拟网络功能中间盒的服务器与真实的 OpenFlow 交换机、服务器、主机等网络设备用网线连接起来,并配置中间盒的虚拟网卡,与服务器相连,就能实现外部实际网络设备与服务器中的虚拟网络中间盒的连接。NFV 网络中的虚实互连使得虚拟网络功能能够与真实网络融为一体,提高了其实用价值。

4.2 检测中间盒的设计与实现

用软件实现检测中间盒的功能也是本文研究的要点之一。以 LXC 容器作为检测中间盒的运行环境,基于 Netfilter 框架[17]和 HOOK 技术设计检测功能的程序。常用的 HOOK 点有 5 种,分别是 PRE_ROUTING,LOCAL_IN,FORWARD,LOCAL_OUT 和 POST_ROUTING。其中,FORWARD 是数据包转发的 HOOK 点,因此选用它作为设计检测中间盒的 HOOK 点。

libnetfilter_queue 是一个用户态库,本文的用户态程序可以使用它来处理 NF_QUEUE 队列传输来的分组。检测中间盒程序使用 libnetfilter_queue 依次处理 NF_QUEUE 中的 UDP 数据包,为 UDP 流中的非首分组设置时延,用于控制 SDN 控制器在处理首分组时这些分组不会进入交换机,从而消除到达 SDN 控制器的冗余 UDP 分组。

图 3 给出了检测中间盒的工作过程。当某 UDP 流进入检测中间盒后,Linux 的 iptables 指令使所有经过 FORWARD 点的 UDP 分组进入 NF_QUEUE 队列中。接着,中间盒调用 libnetfilter_queue 用户态库函数来处理所有 NF_QUEUE 中的 UDP 分组,检测程序分析 UDP 分组的元组信息,判断其是否为该 UDP 流的首分组。如果是,直接将其转发至外部的 SDN 交换机;如果不是,进行延时后再转发。最后,回调函数返回值为 NF_ACCEPT 的其他分组,这些分组将被转发。

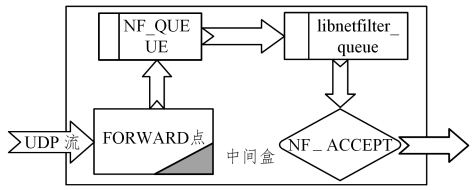

图 3 检测中间盒的功能模块及其工作过程

Fig. 3 Function module of detection middlebox and its working process

检测中间盒的工作需要一个关键参数即非首分组的时延 Δt_1。其计算公式如下:

$$\Delta t_1 = t_{pi} + t_c + t_{po} \tag{1}$$

其中,t_{pi} 为 OpenFlow 交换机发送 Packet_In 报文到 SDN 控制器的时间;t_c 为 SDN 处理 Packet_In 报文的时间;t_{po} 为控制器下发流表到 OpenFlow 交换机的时间。由于上述时间都具有不稳定性,因此理想状态下 Δt_1 的值不是固定不变的,需要一种方法来具体计算该参数。

本文提出了一种通过网络测量测定该参数的方法:1)在检测中间盒的 LXC 不启动检测中间盒功能模块的情况下,流量发生器产生一个新的 UDP 流;2)在控制器处运行分组分析仪(如 Wireshark),记录俘获的首个 UDP 分组的时刻 T_1;3)分组分析仪记录俘获到的 Flow_Mod 报文的时刻 T_2;4)估算控制器从处理首个分组到下发流表所用的时间,大约为 $T_2 - T_1$。

5 实验与性能评价

根据第 3 节的模型和第 4 节的设计,本节在 LXC 环境下采用 NFV 与 SDN 综合的方法,实现了一个防范 SDN 控制器 UDP 冗余分组攻击的原型系统。通过多次实验来测试该系统的性能指标,并对结果进行分析。

5.1 原型系统的实现

原型系统是一台 Intel(R) Xeon(R) X5647 服务器,其主

频为 2.93 GHz,内存为 32 GB,操作系统为 Ubuntu 16.04 server。以 LXC 作为虚拟机,构建了 3 台基于 Open vSwitch[18]生成的 OpenFlow 交换机 S_1,S_2 和 S_3,SDN 控制器由运行在虚拟主机上的 ONOS 控制器软件充当,将该控制器与 S_1,S_2,S_3 相连,宿主机上安装协议分析仪 Wireshark 来获取并分析到达各个端口的流控制报文。在虚拟主机上安装流量发生器软件 Iperf,得到虚拟用户主机 c_1,c_2,…,c_{10};在 LXC 上安装控制 UDP 分组的检测中间盒功能模块作为检测中间盒 Mid_1,Mid_2,…,Mid_{10},并将它们部署在虚拟主机与 OpenFlow 交换机的某个端口之间。基于 NFV 的原型系统的配置如图 4 所示。

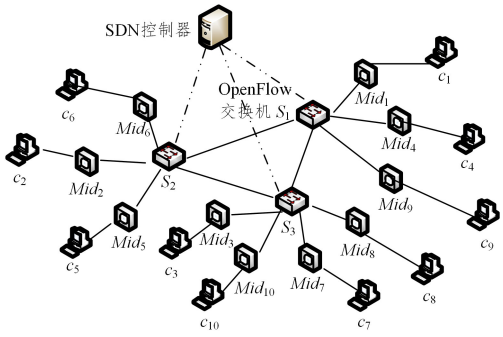

图 4　基于 NFV 的原型系统

Fig. 4　Prototype system based on NFV

测定的过程:1)在检测中间盒不启动的情况下,用户主机向 SDN 网络发送一条带宽为 100Mbps 的 UDP 流。在 ONOS 控制器处俘获分组,得到并记录该 UDP 流到达控制器的第一个 Packet_In 报文的时刻 $T_1=1.19306$ s,同时记录 ONOS 控制器为该 UDP 流下发 Flow_Mod 消息的时刻 $T_2=1.19428$ s。考虑主机到交换机的传播时延以及交换机处理并转发到控制器的时延,可以估算出正常情况下控制器在收到 Packet_In 报文到下发流表所需的时间大于 t,其中 $t=T_2-T_1=1.22$ ms。2)由于在有负载的情况下,流表下发的时间会略有增加,因此将检测中间盒的 Δt_1 设置为 1.5 ms,并启动检测中间盒,在相同的条件下再次让 Iperf 向 SDN 网络发送分组。此时,ONOS 控制器只会收到 1 个 Packet_In 报文,这表明 $\Delta t_1=1.5$ ms 是一个较好的检测中间盒的时延参数。

5.2　实验与结果分析

为了测试文中提出的前置检测中间盒方法的效果,让图 4 中的虚拟用户主机向 SDN 网络发送 16 条 UDP 流,并且让每流发送速率从 50 Mbps 开始,依次递增 50 Mbps,直至 250 Mbps 为止。为了进行分析对比,分别在启动和不启动检测中间盒功能模型的情况下,统计在 ONOS 控制器端俘获到的 UDP 分组的数量。重复实验 10 次,取平均值作为最终结果。实验结果如图 5 所示。

从图 5 可知,当不采用检测中间盒时,到达控制器的 UDP 分组的数量随着 UDP 流带宽的增加而显著上升,例如当 UDP 流的发送速率为 250 Mbps 时,UDP 分组多达 17 756 个。相比之下,当采用检测中间盒时,控制器中的 UDP 分组数量最少仅有 16 个。图 5 还给出了检测中间盒的 Δt_1 分别选取 0.6 ms,1 ms,1.3 ms,1.5 ms 和 1.6 ms 时延时,控制器俘获到的 UDP 分组数量。当 $\Delta t_1=0.6$ ms 时,由于该参数值明显偏小,为该流设置的时延量不足,控制器尚未为该流建立完路径,这时仍会有大量 UDP 冗余分组到达控制器;当 $\Delta t_1=1$ ms 时,该流的时延量依然不够,但在流带宽相同的情况下,产生的 UDP 冗余分组明显少于设置 0.6 ms 时延时所产生的 UDP 冗余分组;当 $=1.3$ ms 时,由于该流的时延量已经很接近 OpenFlow 交换机与控制器处理分组的时延,冗余分组的数量较少;当 $\Delta t_1=1.5$ ms 时,为该流设置的延时量已经可以有效地缓存 UDP 的冗余分组,使得控制器中 UDP 的分组数量最少仅有 16 个,即每条 UDP 流只产生一个分组,无冗余分组;当 $\Delta t_1>1.5$ ms 时,对分组数量的控制效果与 $\Delta t_1=1.5$ ms 时无异,但是却增加了该流的端到端时延。当在控制器性能降低、处理分组时间增加时,可以适当增加 Δt_1 来保证对冗余 UDP 分组的控制。

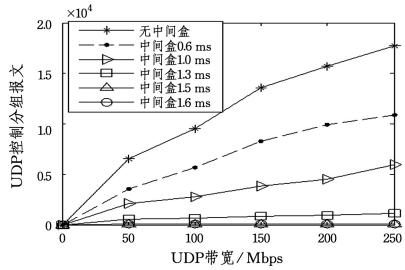

图 5　检测中间盒对 UDP 冗余分组数量的影响

Fig. 5　Influence of detection middlebox on number of duplicated packets of UDP

该实验表明:当检测中间盒的时延 Δt_1 设置合理时,到达控制器处的冗余 UDP 分组会被极大地抑制且不会造成明显的时延,这充分表明了本文所提的前置检测中间盒方法能够有效地防范 UDP 冗余分组对 SDN 控制器的攻击,提升了系统的安全性和健壮性。

TCP 流的友好性对于网络的稳定起到决定性作用,而随着 UDP 流在网络流量中比例的增大,网络的稳定性难以得到保证。这是因为 UDP 流不具备优良的 TCP 流的友好特性,因此通常认为网络中的 UDP 流对于 TCP 流是不公平的[19]。本实验的目的是测试分析为 UDP 流增加了检测中间盒后,对 TCP 流的公平性是否有影响。让 3 台虚拟主机发送 TCP 流,而其他虚拟主机仍以不同带宽发送 UDP 流。图 6 分别给出了采用检测中间盒和不采用检测中间盒两种情况下,UDP 流影响 TCP 流建立连接的时延数据。由于 TCP 流在控制器中的排队序列有一定的随机性,导致时延波动性较大,因此在统计数据时分别在最小端和最大端各去掉多个极端数据后再进行统计。由于 TCP 流在传输数据分组之前,首先要通过 3 次握手来建立源和目的地之间的连接,因此 TCP 源主机仅当第 2 次握手分组返回时,才会发送第 3 次握手分组。第 1 个握手分组与大量 UDP 冗余分组一同到达控制器时,需要在控制器缓存按先到先服务的规则排队。由于在缓存中 TCP 首分组与 UDP 冗余分组的数量相差甚远,因此 TCP 流得到控制器服务的概率较低,如果 UDP 分组数量很大,则 TCP 握手分组因缓存满而丢失的概率较大,从而极大地增加了建立 TCP 连接的时延。而当虚拟主机前端连接检测中间盒时,其有效

地控制了控制器下发流表之前进入 OpenFlow 交换机的 UDP 分组的数量，使得 TCP 流与 UDP 流在控制器缓存排队时更为公平，从而极大地减少了建立 TCP 连接的时延。图 6 所示的结果表明，当不采用检测中间盒时，建立 TCP 连接的平均时延随着 UDP 流带宽的增加而上升，当 UDP 流的发送速率为 250 Mbps 时，该时延可达 500 ms 以上，而且还存在建立连接失败的情况。相比之下，当采用前置检测中间盒时，无论多条 UDP 流的发送速率如何变化，几条 TCP 流的建立连接时延均比较稳定而且较低，约为 2.5 ms。

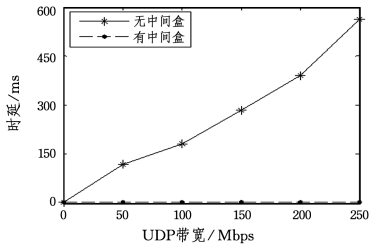

图 6 TCP 连接建立的时延

Fig. 6 Time delay established by TCP connection

该实验表明：检测中间盒实现技术能够以经济的手段提升 UDP 流的 TCP 友好性，增强了 NFV 和 SDN 综合网络的稳定性。

原型系统的实验表明了在 NFV 和 SDN 网络中采用前置检测中间盒方法的可行性和可用性。

结束语 尽管 SDN 网络技术与应用已经取得了空前的进展，但 SDN 控制器仍可能受到大量 UDP 流冗余分组的威胁。控制器缓存中的大量 UDP 冗余分组一方面会消耗大量的控制器资源，另一方面会影响 TCP 流的正常运行，严重时将造成整个网络效率低下甚至瘫痪。文中提出了一种前置检测中间盒机制，其能够有效地防御 UDP 冗余分组攻击 SDN 控制器；并且，提出了一种基于 NFV 和 SDN 综合网络的技术，该技术能够经济有效地完成设计的目标。原型系统的实验结果表明，这种机制和技术具有可行性，能够有效地防范 UDP 冗余分组对 SDN 控制器的攻击，提升了系统的安全性和健壮性；检测中间盒实现技术能够以经济有效的手段提升 UDP 流的 TCP 友好性，增强了 NFV 和 SDN 综合网络的稳定性。下一步将研究把这种机制和技术用于 NFV 和 SDN 综合网络的其他安全方面，并推进该技术的实用化。

参 考 文 献

[1] MOUSAVI S M, ST-HILAIRE M. Early Detection of DDoS Attacks Against Software Defined Network Controllers[J]. Journal of Network & Systems Management, 2018, 26(3): 573-591.

[2] ABDOU A R, OORSCHOT P C V, WAN T. Comparative Analysis of Control Plane Security of SDN and Conventional Networks[J]. IEEE Communications Surveys & Tutorials, 2018, 20(4): 3542-3559.

[3] DARGAHI T, CAPONI A, AMBROSIN M, et al. A Survey on the Security of Stateful SDN Data Planes[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1701-1725.

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

[5] MIJUMBI R, SERRAT J, GORRICHO J L, et al. Network Function Virtualization: State-of-the-art and Research Challenges[J]. IEEE Communications Surveys & Tutorials, 2017, 18(1): 236-262.

[6] TOOTOONCHIAN A, GORBUNOV S, SHERWOOD R, et al. On controller performance in software-defined networks[C]// Usenix Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. San Jose: USENIX Association, 2012: 10-10.

[7] JARSCHEL M, OECHSNER S, SCHLOSSER D, et al. Modeling and performance evaluation of an OpenFlow architecture [C]// Teletraffic Congress. San Francisco: IEEE, 2011: 1-7.

[8] ZUO Q Y, CHEN M, DING K, et al. Eliminating Redundant Control Messages in OpenFlow Networks[J]. Journal of Computer Research & Development, 2014, 51(11): 2448-2457.

[9] GUDE N, KOPONEN T, PETTIT J, et al. NOX: towards an operating system for networks[J]. Acm Sigcomm Computer Communication Review, 2008, 38(3): 105-110.

[10] FONSECA P, BENNESBY R, MOTA E, et al. A replication component for resilient OpenFlow-based networking[C]// Network Operations and Management Symposium. Maui: IEEE, 2012: 933-939.

[11] HU H, CHEN M, LIU B, et al. Mechanism of eliminating UDP redundancy control packets in OpenFlow network[J]. Journal on Communications, 2017, 38(9): 167-175.

[12] JIA Y, WU C, LI Z, et al. Online Scaling of NFV Service Chains Across Geo-Distributed Datacenters[J]. IEEE/ACM Transactions on Networking, 2018, 26(2): 699-710.

[13] HOFFMANN M, JARSCHEL M, PRIES R, et al. SDN and NFV as Enabler for the Distributed Network Cloud[J]. Mobile Networks & Applications, 2018, 23(3): 521-528.

[14] GUAN J, WEI Z, YOU I. GRBC-based Network Security Functions placement scheme in SDS for 5G security[J]. Journal of Network & Computer Applications, 2018, 114(15): 48-56.

[15] BERNSTEIN D. Containers and Cloud: From LXC to Docker to Kubernetes[J]. IEEE Cloud Computing, 2015, 1(3): 81-84.

[16] QIU X, ZHANG K, REN Q. Global Flow Table: A convincing mechanism for security operations in SDN[J]. Computer Networks, 2017, 120: 56-70.

[17] FIESSLER A, LORENZ C, HAGER S, et al. HyPaFilter+: Enhanced Hybrid Packet Filtering Using Hardware Assisted Classification and Header Space Analysis[J]. IEEE/ACM Transactions on Networking, 2017, 25(6): 3655-3669.

[18] EMMERICH P, RAUMER D, GALLENMULLER S, et al. Throughput and Latency of Virtual Switching with Open vSwitch: A Quantitative Analysis[J]. Journal of Network & Systems Management, 2018, 26(2): 314-338.

[19] FLOYD S, FALL K. Promoting the use of end-to-end congestion control in the Internet[J]. IEEE/ACM Transactions on Networking, 1999, 7(4): 458-472.