

基于软件定义网络的网络故障诊断方法研究

李嘉伟, 杨知举, 林喜军, 曲海鹏

(中国海洋大学信息科学与工程学院, 山东 青岛 266100)

摘要: 基于软件定义网络环境下故障诊断方法的研究现状, 提出了一种在软件定义网络下故障诊断定位的方法。通过发送具有匹配识别标志字段的 TCP 测试数据分组, 利用软件定义网络环境下控制器的可编程和流表可扩展匹配的特性, 设计匹配流表和指令, 匹配并按特定格式写入具备定位作用的转发信息到测试数据分组。监听分析模块根据返回数据分组中的历史转发信息, 提取数据并据此诊断和定位网络故障。在模拟环境下, 实验结果证明了该方法的有效性, 具备一定的应用价值。

关键词: 网络管理; 软件定义网络; 故障诊断; 可扩展匹配

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2016.00118

Research on networking troubleshooting method based on software defined network

LI Jia-wei, YANG Zhi-ju, LIN Xi-jun, QU Hai-peng

(College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China)

Abstract: Based on the reaserch status of networking troubleshooting in software defined network, a new networking troubleshooting method was proposed. A TCP packet marked in head field was sent and detailed forward information in every hop would be written in its data filed once the test packet was matched. Taking advantages of the programmable feature of the controller and OpenFlow extensible match, special flow table was designed for matching test-used packet. The analysis module extracted and split the forward information, and thus printed the suspicious network problem which can be used to track and located the problematic network node. The experiment results indicate that the proposed method is fairly efficient.

Key words: network management, software defined network, networking troubleshooting, OpenFlow extensible match

1 引言

随着互联网的日益发展, 网络规模变得越来越大, 网络用户和业务需求不断增长, 网络中存在的问题也逐渐增多^[1]。

根据统计, 在网络管理人员遇到的最常见的网络故障中, 排名前 5 位的依次为: 目的主机不

可达、吞吐量/延迟过大、间歇连接、路由器 CPU 利用率过高和拥塞问题。最常被用来诊断这些故障的 3 个工具依次为: ping、traceroute 和 SNMP^[2]。然而随着网络规模的增大, 使用这些工具来解决问题的效率变得非常不理想, 传统的方法和工具已经不能满足网络管理人员的需求, 网络管理人员实现对网络的管理与监控也变得近乎不可能。

收稿日期: 2016-08-20; 修回日期: 2016-10-30。通信作者: 曲海鹏, quhaipeng@ouc.edu.cn

基金项目: 国家自然科学基金资助项目 (No.61379127)

Foundation Item: The National Natural Science Foundation of China (No.61379127)

与此同时, 网络的管理具有一定的困难性, 这是由网络故障定位中的 3 个难点导致的。

1) 转发状态难以观察。如果需要查看, 网络管理人员必须登录到对应的每一个网元设备, 查看并梳理庞大的日志文件。

2) 转发状态分散度高。网络通信数据分组从源主机到目的主机需要经历若干个路由器和交换机等网元设备, 每个网元设备都可能涉及数据分组的转发状态。

3) 有过多的应用程序、协议或网络管理人员可能会更新转发状态。

因此, 解决目前网络管理所面临的困境的一个关键因素就是获取数据分组在其生存周期内的所有转发信息。

随着软件定义网络 (SDN, software defined networking) 的兴起, 可编程控制器的介入为网络管理人员提供了一个可以更加灵活地通过编程控制数据分组转发的方法, 从而使获取数据分组转发信息变得可行。本文研究了在软件定义网络下的网络故障诊断方法, 该方法能有效地解决当前网络故障诊断中存在的问题, 提高网络管理人员的工作效率。为达到这一目的, 本文设计了具有测试标志的数据分组, 该数据分组是在保留字段填充特定标志位的 TCP 数据分组, 同时设计了数据分组的监听分析模块, 可以通过分析测试返回数据分组中的转发信息做出相应的诊断输出。

2 背景知识和相关工作介绍

2.1 SDN 简介

SDN 技术起源于美国国家自然科学基金会资助的斯坦福大学 Clean Slate 项目。该项目的研究目标是设计开发新型互联网架构, 淘汰现有存在弊端的网络架构, 随着 SDN 技术的不断发展, 基于 OpenFlow 的 SDN 技术逐渐被大家认可并推广, 还曾被美国麻省理工学院主办的 Technology Review 评为十大前沿技术^[3], 成为当前学术界的热点。

目前, 计算机的网络架构可以分为 3 层: 数据层、控制层和管理层。网元设备负责转发数据层的通信数据, 再通过相应的协议指导控制层对数据的转发做出决策, 管理层则是利用基于

SNMP 的工具对网元设备的功能进行监控和配置。该架构是高度耦合的, 特别是数据层和控制层, 它们基本上都位于同一网元设备中。这一设计在最初 Internet 的发展过程中很好地保证了网络弹性, 但随着网络规模的急剧增大, 这一架构的弊端也开始显现, 数据层与控制层的高关联性使传统的网络过于僵硬, 不易于管理和控制, 而及时地对故障进行判断与诊断更是不易。

SDN 是旨在改善上述弊端的一种动态的灵活的新型互联网架构。如图 1 所示, SDN 技术将数据层和控制层进行解耦合设计, 把传统网络中的控制层与数据层分离出来, 这时的网元设备着重负责数据层的转发工作, 专门的控制器负责管理整个网络。SDN 开放了控制层的软件模式, 提供了丰富的 API 接口, 赋予控制器可编程特性, 实现集中式的控制, 对网络进行管理和配置, 同时在应用层通过不同的软件来满足各式各样的网络管理需求, 由此, 在当前静态网络下的很多平台就可以演变成为一个可扩展的服务交付平台来应对快速变化的业务需求。

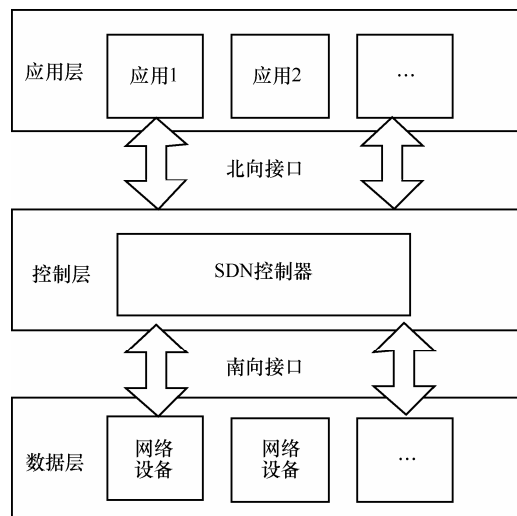


图 1 软件定义网络架构

SDN 的基本思想有以下 4 个方面。

- 1) 控制层与数据层解耦合。
- 2) 转发决策基于流表而非基于目的地址。流表是一组匹配标准和操作的集合, 流表的可编程性给网络管理带来了前所未有的灵活性。
- 3) 控制逻辑分离到外部实体, 即 SDN 控制器或网络操作系统^[4] (NOS, network operating

system)。

4) 通过 NOS 上可与数据层交互的应用程序实现网络可编程。

SDN 同样会遭遇网络节点失效等问题,但它相比于传统的互联网具有更快恢复节点的能力,这种低容错性为其在网络中有效地检测出故障提供了保障^[5]。

2.2 OpenFlow 简介

2008 年,斯坦福大学的 Nick McKeown 教授在 ACM Sigcom 上首次提出 OpenFlow 的概念。OpenFlow 协议实现了控制器和转发设备的控制和管理,是主流的 SDN 南向接口协议^[6]。它开放了可编程的通信网络,为用户提供了可以动态编程的功能。

OpenFlow 交换机至少由 3 部分组成,如图 2 所示。

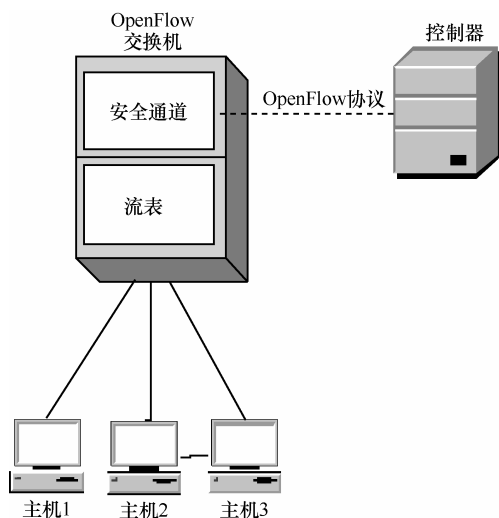


图 2 OpenFlow 交换机的构成

- 1) 含有对应操作指令的流表。
- 2) 负责连接控制器和交换机的安全通道。
- 3) OpenFlow 协议。

其中,控制器完全负责流表的生成、更新和删除,交换机功能被简化,只负责根据流表项的指令转发数据分组^[7]。每个流表项对数据分组的操作指令必须包含但不限于以下 3 项:转发到指定端口、转发到控制器和丢弃。

在 OpenFlow v1.3 中,流表项共包含 7 部分,分别是负责数据分组匹配的匹配域、优先级、负责统计的计数器、决定数据分组如何处理的

指令、超时设定、cookie 和标志。其中匹配域又分为 12 个字段,包括 OSI 模型中的 1~4 层的网络控制信息。匹配域的可扩展匹配 (OXM, OpenFlow extensible match) 技术是从 OpenFlow v1.2 开始实现的,这使用户可以自定义数据分组匹配字段,大大增加了数据分组匹配的灵活度和精细度。

2.3 相关研究进展

随着 SDN 的快速发展,因其灵活的可编程性,网络故障定位技术也逐渐获得了更多的关注,当前在这一领域已取得了一些进展和成果。文献[2]提出了一种可自动生成测试数据分组的工具,但其网络带宽的利用率过高,会影响网络的正常使用。

斯坦福大学的研究人员在文献[8]中提出了 NetSight 平台,该平台集成了调试器、监视器、历史日志和分析器这 4 个应用来定位和分析网络中的故障。其核心思想是记录下每一个数据分组的转发历史信息,一旦出现网络故障,可通过分析器分析相关数据分组的转发历史信息以诊断故障。但是该方法需要单独的服务器支撑该平台的运行,同时因为需要记录每一个数据分组的所有转发历史信息,其对系统资源的要求也很高。因此该方法部署成本高,在中小型网络环境下并不是理想的诊断工具。

文献[9]提出了一个 SDN 故障诊断方法,该方法的工作原理是实时收集来自控制器和各交换机的告警信息与错误信息,通过匹配知识库产生诊断结果,其工作方式属于被动式,因此应用范围和效果欠佳。通过文献[10]提出的方法,网络管理人员可以在控制器发送测试数据分组和收集流量信息。这种方法设定了若干故障场景以供测试,但它只能运行在控制器端,这降低了其使用范围,从而降低了实用性。

3 网络故障诊断方法

为实现网络故障诊断,本方法共设计 3 个功能模块:测试数据分组发送模块、测试数据分组转发处理模块和测试返回数据分组监听分析模块。其工作模式可简要概括为:预先通过控制器向交换机下发用于匹配测试数据分组的流表,发

送模块生成并发送具有测试标志的数据分组，当交换机匹配到该数据分组之后写入相应的转发信息并转发，通信完成之后监听分析模块接收到测试返回数据分组，提取分析转发信息并将诊断结果输出。

3.1 发送模块设计

与正常的业务流量不同，测试网络故障的数据分组需要通过编程填充首部构造测试数据分组。本方案要求测试任务不能影响到正常的业务流量，即需要有特殊标识区分测试数据分组和正常业务通信数据分组。这些功能和需求就是通过发送模块满足的。

该模板使用 python 语言编写，名为 `diagnose.py`，通过原始套接字构造测试数据分组。为区分正常的 TCP 数据分组和测试用途的 TCP 数据分组，在构造数据分组时，做如下处理。

- 1) 将其首部的 6 位保留字段置为“111100”。
- 2) 将源端口号置为“30021”。
- 3) 其余字段均模仿普通数据分组进行填充。
- 4) 数据分组的数据部分在发送时不做任何填充。

该模块的使用命令为

```
python send.py <destination IP>
```

3.2 转发处理模块设计

对于测试数据分组的转发处理是本方案的重要组成部分，网络故障诊断可行的基础就在于在 SDN 网络环境中，对数据分组的处理更加灵活。SDN 的网络可编程特性与开源的控制器使网络管理更加容易。

转发处理主要包括写入转发信息和处理丢弃包操作。在诊断任务进行前，需要修改当前网络的配置，通过控制器向其控制的每一个交换机发送 `OFPP_FLOW_MOD` 消息，添加用于匹配测试数据分组的流表。

该流表的匹配设置为

```
match = oxm { ip_p roto = "6", tcp_res = "111100", in_port = "1", eth_type = "0x800", }
```

该流表配置完毕后，控制器和交换机便可对

测试数据分组按以下流程进行处理。

```
function process(packet,switch,controller)
    foreach flowset do
        if packet ∈ switch.matchset then
            write actions
            if action! = drop then
                write forward info
                forward to next hop
            else
                sent to controller
            else
                sent to controller
```

当交换机接收到测试数据分组之后对数据分组进行匹配操作，匹配成功时，向数据分组写入转发信息。

获得数据分组的转发历史信息对网络故障的诊断起着至关重要的作用。因此，在每一个网元设备对数据分组执行指令动作之前，均按照如图 3 所示的字符格式写入对应 27 B 的转发信息。

1) 第 0~5 字节：写入长度为 6 B 的交换机 MAC 地址，用于唯一标识定位网元设备。

2) 第 6~9 字节：写入交换机的入端口标识，用于分析数据分组转发经历。

3) 第 10~13 字节：写入交换机的出端口标识，用于分析数据分组转发经历。

4) 第 14~15 字节：写入数据分组在该网元设备的优先级信息，用于判断其合理性。

5) 第 16~19 字节：写入数据分组的持续时间，即出端口时间与入端口时间的差值，用于判断拥塞和延迟情况。

6) 第 20~23 字节：写入出现丢弃指令的表项。若未出现该情况，该字段置 0。

7) 第 24~26 字节：记录当前数据分组将会执行的指令类型，依次为：转发、入队列、修改 Tag 和设置报头域，若存在以上指令，则在对应字节写入 1，否则写入 0。

例如，如果交换机对数据分组执行的是转发和修改 Tag 操作，则该情况下的写入数据为“1010”。

0	5	9	13	15	19	23	26
交换机MAC	入端口	出端口	优先级	持续时间	表项	指令	

图 3 转发信息存储格式

并非所有的测试数据分组都能正确匹配, 对于因流表错误配置等问题造成的丢弃操作, 触发数据分组丢弃之后将该数据分组转发至控制器, 由控制器写入转发信息之后发送给源主机。综上, 数据分组在每个交换机的处理流程如图 4 所示。

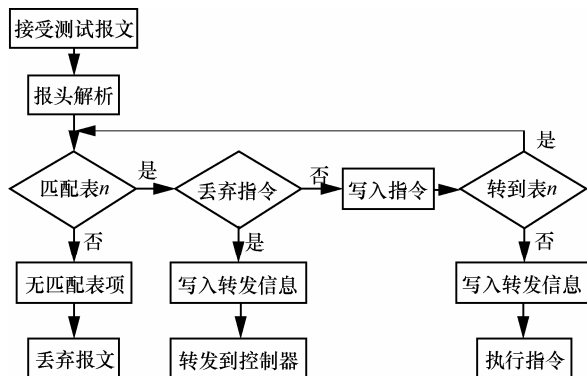


图 4 测试数据分组处理流程

3.3 监听分析模块设计

监听分析模块采用 Python 语言编写实现, 属于 diagnose.py 文件的另一个代码块。当发送模块完成数据分组发送之后, 调用该模块在 30021 端口监听接收测试返回数据分组。

对接收到的数据分组根据原始套接字技术进行解码操作, 以每一跳转发信息为单位字符串, 即 27 B, 将其放入到字符串 list 中。然后遍历该字符串并按照诊断需求输出对应项, 处理逻辑如下所示。

```

function diagnose(raw packet)
    for gap=27 in range(40:len(packet))
do
    dataList ← info of each hop
    for item in dataList do
        #获得数据分组经历的指令集
        actionList ← (item(0:5), item
(23:26))
        #输出拥塞时间最长的前 5 个
交换机
        #用于判断拥塞延迟情况
        DurationList ← (item(0:5),item
(15:19))
        print the five maximum duration

```

```

print actionList
#输出所有表项字段填充非 0

#用于诊断出现丢弃指令的表项
tableList←(item(0:5),item(15:19))
print tableList
print actionList

```

交换机

该模块可实现的主要功能包括: 转发规则诊断、分组丢失操作诊断和拥塞延迟诊断。对于诊断转发规则, 其实现原理为: 遍历转发历史信息 dataList 可获得数据分组经历的指令集 actionList, 因此便可查看数据分组在每一跳的经历转发规则。对于测试正常的数据分组, 其历史信息应是无错误规则的, 也就意味着该功能需要配合经历网络故障的数据分组使用。诊断由转发规则配置错误引起的故障, 可通过诊断从后向前输出的 actionList 中的数据实现。

分组丢失现象分为 2 种情况: 成功匹配流表和未找到匹配流表^[7]。这 2 种情况均可触发交换机的分组丢失操作。对于诊断分组丢失故障, 可选择输出所有出现在 tableList 中的交换机地址和出入端口定位到该交换机。

对于诊断拥塞延迟^[11], 首先对数据分组在所有交换机的持续时间进行排序, 然后输出持续时间最长的前 5 位、优先级及交换机的 MAC 地址和出入端口。持续时间排在 5 位并不能决定数据分组在该交换机经历了延迟问题, 判断是否出现拥塞延迟需要管理人员配合优先级的设置大小及持续时间的具体数值进行判断。一般来讲, 如果持续时间大于毫秒级或者远高于平均持续时间, 则任务在该交换机出现延迟问题, 需要管理人员进一步查看该交换机的配置情况。

4 仿真实验

本节采用 Mininet 模拟 SDN 实验环境用来验证上述故障诊断方法的可行性和有效性。Mininet 采用虚拟化技术提供了轻量级的 SDN 研究平台, 它支持 OpenFlow、OpenvSwitch 等各种协议, 通过 Mininet, 开发人员可以灵活地进行相关测试, 并能轻松地实现实验环境到真实硬件环境的部署工作。

4.1 实验环境

操作系统采用 Ubuntu 14.04 LTS 系统，Mininet 运行虚拟机采用 Virtualbox，控制层采用 Floodlight 作为 SDN 控制器^[12]。

平台细节如下。

- 1) 个人计算机。
- 2) 操作系统：Ubuntu 14.04 LTS。
- 3) 处理器：Intel Core i5-4460。
- 4) 内存：金士顿 8 G 1600 MHz。
- 5) 网卡：Realtek PCIe GBE 系列控制器。
- 6) Floodlight：运行在物理主机上，监听端口 6653。
- 7) Mininet：通过安装在物理主机上的 Virtualbox（虚拟机）软件运行。

为便于测试，搭建具有 3 个交换机 2 台虚拟主机的线性网络环境，拓扑结构如图 5 所示。

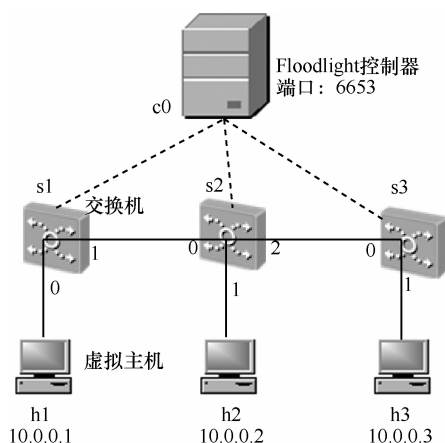


图 5 实验环境拓扑结构

该实验环境下共有 3 个 OpenFlow 交换机，为了直观地观察实验结果，将 s1、s2、s3 的 MAC 地址分别设置为以 S1、S2、S3 结尾。交换机 s1 连接主机 h1，其 IP 地址为 10.0.0.1；交换机 s2 连接主机 h2，其 IP 地址为 10.0.0.2；交换机 s3 连接主机 h3，其 IP 地址为 10.0.0.3。同时控制器已向各个交换机下发了匹配测试数据分组的流表。

4.2 设定场景及实验结果

通过模拟具有故障的网络环境，可以测试本文提出的网络故障诊断方法的可行性与有效性。以下分 2 个场景模拟实现故障定位诊断。

场景 1 指定交换机含有配置不当的流表通过 Floodlight 的 Rest API 加入一条含有错

误转发规则的流表，使本应通过 s2 的端口 2 发往 s3 的测试数据分组被 s2 主动丢弃，最终导致数据分组无法被 s3 接受并转发。添加流表 curl 命令如下。

```
curl -d '{"switch": "00:00:00:00:00:S2",
"name": "flow-mod-2", "cookie": "0",
"priority": "32768", "ingress-port": "0",
"active": "true", "actions": "drop"}'
http://1.1.1.1:8080/wm/staticflowentrypusher/json
```

在该网络环境下，从主机 h1 发送测试数据分组到 h3，主机 h1 上的监听分析模块会接收到控制器返回的带有转发信息的数据分组，按照 3.3 节中的处理逻辑得到如图 6 所示的实验结果。

Drop at 00:00:00:00:00:S2, table 15

图 6 相关运行结果

测试返回数据分组中包含 s1 和 s2 的转发信息，判断出丢弃操作发生在 s2，故其转发信息的第 19~23 字节填充非 0，从而可以输出配置错误的流表项。

场景 2 指定交换机出现网络拥塞

在主机 h2 使用 iperf 工具向主机 h3 满负载发送 TCP 数据分组共 5 s，导致交换机 s2 的端口 2 出现拥塞延迟。与此同时运行主机 h1 的发送与监听模块，分析测试数据分组的延迟情况。iperf 操作命令为

```
iperf -c 10.0.0.3 -M 1024-t 5-I 1
```

实验对照组分别为运行 iperf 1 s 和取消 iperf 操作，在保持网络环境正常的情况下运行主机 h1 的发送与监听模块，分析测试数据分组的延迟情况。实验结果如图 7 所示。

在不运行 iperf 的正常网络环境下，测试数据分组的平均持续时间为 16 ms，其在交换机 s2 的持续时间为 20 ms；在运行 iperf 5 s 的情况下，测试数据分组在 s1 和 s3 的平均持续时间为 19 ms，属于正常范围，但其在 s2 的持续时间为 4 500 ms；在运行 iperf 1 s 的网络环境下，测试数据在 s2 的持续时间为 950 ms。可见，数据分组的持续时间有效地反映了交换机的网络状态，监听分析模块的正确性与有效性得到了印证。

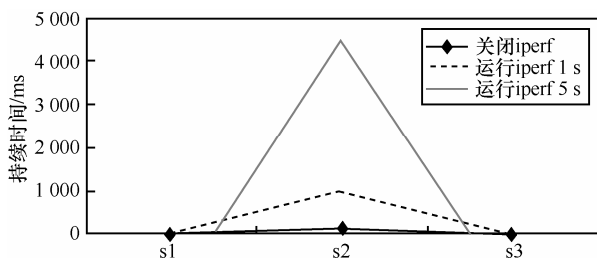


图7 3种不同网络环境的实验结果

5 结束语

在传统网络诊断工具,诸如 ping 和 traceroute 已不能高效地完成故障诊断的情况下,本文结合 SDN 可编程特性,提出了一种高效的网络故障诊断和定位的方法,通过发送测试数据分组记录每一个交换机上执行的指令集合和相关转发信息,通过 python 监听分析模块,在接收到返回数据分组之后提取和分析数据,从而达到诊断和定位故障的目的。模拟实验也证明了该方法的可行性和有效性。SDN 作为新兴的网络体系,其应用前景是巨大的,值得进一步学习和研究。

参考文献:

- [1] GILL P, JAIN N, NAGAPPAN N. Understanding network failures in data centers: measurement, analysis, and implications[J]. Sigcomm ACM, 2011, 41(4): 350-361.
- [2] ZEND H, KAZEMIAN P, VARGHESE G, et al. Automatic test packet generation[J]. IEEE/ACM Transactions on Networking, 2012, 22(2): 554-566.
- [3] MIT technology review[EB/OL][2013-12-28].<http://www.technologyreview.com/article/412194/tr10-software-defined-networking/>.
- [4] 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.
- [5] 张朝昆, 崔勇, 吴建平, 等. 软件定义网络(SDN)研究进展[J]. 软件学报, 2015(26): 62-81.
ZHANG Z K, CUI Y, WU J P, et al. Evelopment of SDN Stucly[J]. Journal of Software, 2015(26): 62-81.
- [6] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM Sigcomm Computer Communication Review, 2008, 38(2): 69-74.
- [7] 左青云, 陈鸣, 赵广松, 等. 基于 OpenFlow 的 SDN 技术研究[J]. 软件学报, 2013(5): 1078-1097.
ZUO Q Y, CHEN M, ZHAO G S, et al. Stady of SDN based on OpenFlow[J]. Journal of Software, 2013(5): 1078-1097.
- [8] HANDIGOL N, HELLER B, JEYAKUMAR V, et al. I know what

your packet did last hop: using packet histories to troubleshoot networks[C]//Llisenix Conference on Networked Systems Design and Implementation. 2014: 71-85.

- [9] BAJPAY P, BIENFAIT R, DARDASHTI M, et al. Method and apparatus for providing automated processing of a network service alarm: US20100014431[P]. 2010.
- [10] ZHANG Y. An adaptive flow counting method for anomaly detection in sdn[C]//The 9th ACM Conference on Emerging Networking Experiments and Technologies. 2013: 25-30.
- [11] KREUTZ D, RAMOS F, ESTEVES P, et al. Software-defined networking: a comprehensive survey[J]. Proceedings of the IEEE, 2014, 103(1): 10-13.
- [12] LE F, LEE S, WONG T, et al. Detecting network wide and router specific misconfigurations through data mining[J]. IEEE/ACM Transactions on Networking, 2009, 17(1): 66-79.

作者简介:



李嘉伟 (1992-), 男, 山东青岛人, 中国海洋大学硕士生, 主要研究方向为密码学与信息安全。



杨知举 (1990-), 男, 山东济宁人, 硕士, 主要研究方向为网络安全、隐私保护。



林喜军 (1978-), 男, 山东青岛人, 博士, 中国海洋大学讲师, 主要研究方向为密码学与信息安全。



曲海鹏 (1972-), 男, 山东莱西人, 博士, 中国海洋大学副教授, 主要研究方向为网络安全攻防与工业控制安全。