基于 OpenFlow 的 SDN 架构研究与实践

于天放,芮兰兰
(北京邮电大学 网络与交换技术国家重点实验室,北京 100876)

摘 要:随着互联网数据流量在规模上的快速增长,交换机和路由器等设备的软硬件结构日趋复杂化,增加了网络管理的难度。传统网络的控制平面与转发平面的高度耦合不仅让网络节点中全局策略的部署变得困难,而且限制了网络的灵活性与可扩展性。SDN(software-defined networking,软件定义网络)打破了传统网络分层的思想,它将控制和转发解耦,通过提供开放式应用程序接口实现对网络设备的自动化配置和网络流量的实时监测,用以满足日益变化的网络需求。在研究了 SDN 架构和主要实现方式 OpenFlow 技术的基础上,深入探讨了基于 OpenFlow 协议的底层网络拓扑发现机制和网络性能监测机制,并搭建了实验平台进行仿真测试。实验结果表明,SDN 架构通过逻辑上集中控制的方式实现底层网络对上层应用的透明化并根据全局网络视图进行精细的流量监测,从而提高网络整体的服务质量。

关键词:软件定义网络;OpenFlow;流表;拓扑发现;Mininet;服务质量
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2018)07-0159-06
doi:10.3969/j.issn.1673-629X.2018.07.034

Research and Practice on SDN Architecture Based on OpenFlow

YU Tian-fang, RUI Lan-lan
(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: With the rapid growth of Internet traffic, both software and hardware components of switches and routers are becoming more complex, which makes it difficult for network administrators to maintain effective networks. In IP-based networks, the high level of interdependence between the control layer and the forwarding layer not only increases the difficulty of deploying integrated strategies on nodes, but also restricts flexibility and extendibility of these networks. An emerging technology named SDN puts forward a novel architecture against the present multi-layered architecture of the Internet, which innovatively separates out the control layer from the forwarding layer and meets ever-increasing network requirements by means of providing open APIs to realize both automatic configurations on network devices and real-time monitoring on traffic. On the basis of research on the SDN architecture and OpenFlow technology, we deeply explore topology discovery mechanism of underlying network and network performance monitoring mechanism based on OpenFlow. An experiment platform is built to carry out simulation tests, which shows that SDN architecture hides details of underlying networks by means of centralized control, provides end-user applications with an overall view, and accurately monitors traffic according to the view. All of these results will be key factors for raising the quality of service of networks.

Key words: SDN; OpenFlow; flowtable; topology discovery; Mininet; quality of service

0 引 言

随着 IT 技术的高速发展,云计算、虚拟化等新兴业务的不断涌现,对网络的承载能力提出了更高的要求。在传统网络中,控制平面功能是分散地运行在各个节点中,控制逻辑和数据转发被紧耦合在网络设备上,这种分布式的控制方式增加了网络管理的难度,不仅带来了网络互操作性的问题,而且难以开展基于真实网络流量的新技术研究和实验。SDN[1] 和 OpenFlow[2] 的理念就是在上述背景下得以快速发展的。SDN 的基本思想是采用相互分离的控制平面和转发平面,通过将控制功能抽象到控制平面中实现集中化的网络状态控制,同时为用户提供开放式可编程接口,使得用户能够根据上层业务需求对网络资源进行灵活调度。OpenFlow 是 SDN 控制器与交换机之间的主要

收稿日期:2017-08-31 **修回日期**:2018-01-11 **网络出版时间**:2018-03-07
基金项目:辽宁省教育科学研究一般项目(LJQ2014102)
作者简介:于天放(1980-),男,硕士,高级工程师,CCF 会员(E200076461M),研究方向为软件定义网络、医学信息检测与处理。

通信协议[3],其规定了 OpenFlow 交换机的基本组件与性能要求,为实现网络设备的数据转发与路由控制的解耦合提供了一种全新的方法。

文中首先介绍了 SDN 的发展现状以及基于 OpenFlow 的 SDN 架构实现方案。在此基础上,详细探讨了 OpenFlow 协议下的底层网络拓扑发现机制和网络性能监测机制,并搭建了实验平台进行仿真实践。

1 SDN 发展现状及实现方案

1.1 SDN 与 OpenFlow 的发展历程

SDN 的前身来自于斯坦福大学提出的用于企业集中安全控制的 Ethane[4]架构。该架构在链路层和网络层之间定义了一个保护层,由一台逻辑中央控制器对网络主机进行安全绑定和认证。研究人员可以定义基于网络流的安全策略,并将这些策略运用在各种网络设备上,由控制器对每个流检测是否违反了相关的策略,并为每个流确定路由。

在上述相关工作基础上,Nick Mckeown 教授等提出了 OpenFlow 概念。OpenFlow 定义了控制器与 OpenFlow 交换机之间的通信协议,允许控制器来决定网络数据流的路径,并且能够实现细粒度的数据包优先级划分和创建端口级别的流量控制规则。OpenFlow 技术后经由 Clean State 项目的不断推广,以及在 GENI 计划中的应用,逐渐发展为 SDN。2011 年,Google、Microsoft 等企业联合成立了开放网络基金会联盟,共同致力于推动 SDN 技术的发展,并进行相关标准制定和市场推广,使其成为全球开放网络架构与网络虚拟化领域的研究热点。

1.2 基于 OpenFlow 的实现方案

基于 OpenFlow 的 SDN 架构由控制器和 OpenFlow 交换机(文中简称为 OF 交换机)构成。OF 交换机的功能包括安全通道、组表、多级流表和 OpenFlow 协议等几个部分。

安全通道是连接控制器和 OF 交换机的接口,控制器通过该接口对 OF 交换机进行管理和控制。安全通道采用 TLS(transport layer security)技术对控制器和 OF 交换机之间的控制和管理信息进行加密传输。

OF 交换机进行数据转发的依据是流表,多个流表串联起来形成流水线,流水线对数据包的处理过程如图 1 所示。

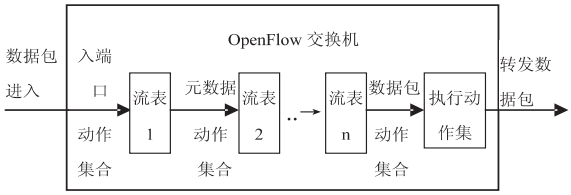


图 1 多级流表的数据包匹配过程

当数据包进入交换机后,将从流表 1 开始匹配。流表中的表项按照优先级的高低依次与数据包进行匹配,当匹配到一条表项时,会触发该表项指令集的执行,这些指令会进行计数器的更新、修改数据包的动作集合等操作。当数据包跳转到最后一个流表时,如果不存在匹配项,则该数据包将转发给控制器,否则,执行数据包对应的动作集合中所有的动作指令。

OF 交换机中的组表可以将每个数据流划分到相应的组中,对属于同一组标识的所有数据包执行相同的动作集,从而实现广播和多播功能。

2 基于 OpenFlow 的拓扑发现机制

2.1 链路层发现协议

链路层发现协议[5](link layer discovery protocol,LLDP)是 802.1ab 中定义的链路层协议,它的帧格式包括目的 MAC 地址、源 MAC 地址、帧类型、链路层发现协议数据单元(link layer discovery protocol data unit,LLDPDU)以及校验位等字段。其中,LLDPDU 字段用于承载要发送的消息,基本的信息单元采用 TLV(Type – length – value)形式表示,TLV 类型如表 1 所示。

表 1 基本的 TLV 类型

TLV 类型	含义	状态
End of LLDPDU	标识 LLDPDU 结束	必选
Chassis ID	发送设备的 MAC 地址	必选
Port ID	标识 LLDPDU 发送端的端口	必选
Time To Live	设备信息在邻居设备上的生存时间	必选

LLDP 将本地设备的主要功能、设备编码、管理地址和接口标识等信息构造成 TLV 封装在 LLDPDU 中,周期性地发送给邻居设备,同时将从邻居设备接收的 LLDPDU 以标准的管理信息库(management information base,MIB)形式进行存储。网络管理程序可以通过 SNMP[6](simple network management protocol)协议访问获取这些信息,从而发现和模拟网络拓扑结构。

2.2 OF 交换机发现机制

OpenFlow 协议允许一台交换机同时连接多个控制器,因此,当交换机启动时会进行主控制器和从属控制器的 IP 地址和 TCP 端口的初始化设置。文中仅讨论单控制器的情形。控制器和 OF 交换机通过建立 TLS 会话实现 OpenFlow 消息的传递,具体的流程描述如下:

步骤 1:安全通道的建立。

OF 交换机首先尝试连接控制器默认开启的 TCP 6633 端口,并和控制器交换证书进行认证。当认证通过后,双方通过发送 OFPT_HELLO 消息确认能够支持

的 OpenFlow 协议版本进而建立安全通道。

步骤 2:OF 交换机的发现。

安全通道成功建立后,控制器将向 OF 交换机发送 OFPT_FEATURES_REQUEST 消息要求交换机上报自己的功能特性。OF 交换机则用 OFPT_FEATURES_REPLY 消息回复控制器,消息中包含了交换机的 DPID(DATA path identity)、所支持的流表数、可用的端口和端口的 MAC 地址等信息。至此,控制器就可以精确掌握成功连接到网络的交换机信息。

2.3　OpenFlow 链路发现机制

依据初始化阶段获得的 OF 交换机信息,控制器为每个 OF 交换机的所有可用端口发送一个 OFPT_PACKET_OUT 消息。该消息携带了一个 LLDP 数据包,数据包由 Chassis ID 和 Port ID 构成。交换机收到消息后按照 Port ID 将消息转发给相邻的交换机,邻居交换机将消息中的 LLDP 数据包连同本身的元数据一起封装到 OFPT_PACKET_IN 消息中发送给控制器,控制器对收到的消息进行解析,得出交换机之间的链接关系,同时进行网络拓扑状态的更新操作。

假设网络由 M 个 OF 交换机构成,每个交换机的可用端口数为 N,交换机之间的连接数为 H,则控制器需要发送的 OFPT_PACKET_OUT 消息数为:

$$\mathrm{SUM}_{\mathrm{packet-out}} = \sum_{i=1}^{M} N_i \tag{1}$$

控制器完成网络中所有 OF 交换机之间链路发现过程需要接收的 OFPT_PACKET_IN 消息数为:

$$\mathrm{SUM}_{\mathrm{packet-in}} = 2H \tag{2}$$

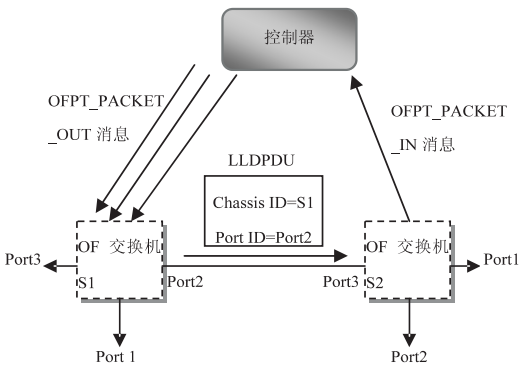


图 2　基于 LLDP 的链路发现过程

图 2 为控制器进行链路发现的具体过程。控制器通过 OFPT_PACKET_OUT 消息向交换机 S_1 的 $Port_1$、$Port_2$ 和 $Port_3$ 端口各发送一个 LLDP 数据包,交换机 S_1 按照 Port ID 分别将数据包在相应的端口转发给邻居交换机。以 $Port_2$ 端口为例,交换机 S_1 将 LLDP 数据包通过 $Port_2$ 端口转发给交换机 S_2 的 $Port_3$ 端口,按照初始的流规则定义,除去交换机的 Controller 端口外的所有端口只要收到 LLDP 数据包都转发给控制器,因此交换机 S_2 将收到的 LLDP 数据包连同自己的元数据一起封装到 OFPT_PACKET_IN 消息并发送给控制器。控制器对该消息进行解析,得到如下转发信息:①LLDP-DU:{Chassis ID = S_1, Port ID = $Port_2$};②Meta-Data:{Chassis ID = S_2, Port ID = $Port_3$}。根据这些信息,控制器能够确定交换机 S_1 和 S_2 之间存在链路关系。

3　基于 OpenFlow 的网络性能监测机制

3.1　网络 QoS 的主要性能指标

网络 QoS 用于为指定的网络通信提供更好的服务能力,它可以用一系列可度量的指标来描述。

(1)吞吐量:对于网络、端口或其他设施,单位时间内成功传送的数据量。

(2)时延:数据包从链路的一端传送到另一端所需要的时间。

(3)丢包率:在数据传输过程中,丢失的数据包数与所有传送数据包数的比率。

3.2　OpenFlow 网络的性能监测

在 OpenFlow 网络中,控制器定期向 OF 交换机发送 OFPT_STATS_REQUEST 消息,该消息中的 Type 字段定义了多种类型的请求信息,包括单流、多流、流表、端口和队列等。控制器通过交换机回复的 OFPT_STATS_REPLY 消息获取不同类型请求的计数信息,根据这些信息,控制器能够计算出吞吐量、时延和丢包率等性能指标。

3.2.1　吞吐量测量分析

OF 交换机向控制器回复的单流类型 OFPT_STATS_REPLY 消息是数据流吞吐量测量的依据。该消息的结构如下:

```
struct ofp_flow_stats{
…
uint32_t duration_sec; //数据流持续时间
uint32_t duration_nsec; //数据流额外生存时间
…
uint64_t packet_count; //已发送的数据包数
uint64_t byte_count; //已发送的字节数
…
}
```

控制器在 T_1 时刻和 T_2 时刻分别统计数据流发送的字节数和持续活动时间,通过求得 T_2 _ T_1 时间间隔内的数据增量,进而计算出数据流的吞吐量值,用公式表达为:

$$\mathrm{Throughput} = \frac{\Delta \mathrm{byte\_count}}{\Delta \mathrm{duration\_sec} + \dfrac{\Delta \mathrm{duration\_nsec}}{10^9}} \tag{3}$$

3.2.2　链路时延测量分析

链路时延测量的过程用图 2 加以说明。

(1)控制器到交换机的时延。

控制器分别向交换机 S_1 和 S_2 发送带有时间戳的 OFPT_ECHO_REQUEST 消息,交换机 S_1 和 S_2 收到消息后即刻向控制器回复带有时间戳的 OFPT_ECHO_REPLY 消息,控制器利用前后两个时间戳的差值计算得到控制器到交换机 S_1 和 S_2 的往返时延[7](round-trip time,RTT),记为 RTT_1 和 RTT_2。

(2)交换机间的链路时延。

首先,控制器产生一个可以识别的时延探测包下发到交换机 S_1,探测包的数据段携带着控制器下发探测包的时间戳。交换机 S_1 将探测包通过 $Port_2$ 端口转发给交换机 S_2,交换机 S_2 将探测包封装到 OFPT_PACKET_IN 消息通过 $Port_3$ 端口发送给控制器。控制器收到消息后,对当前时间与探测包中的时间戳求差值,记为 T_a。同理,控制器向交换机 S_2 发送探测包,重复上述过程,所求得的时间差值记为 T_b。由控制器链路发现过程可知,T_a 与 T_b 之和等于 RTT_1、RTT_2、$RTT_{S_1-S_2}$ 三者之和,因此,交换机 S_1 与 S_2 之间的链路单向时延可表示为:

$$\text{Latency}(S_1, S_2) = \frac{T_a + T_b - RTT_1 - RTT_2}{2} \quad (4)$$

3.2.3 丢包率测量分析

当控制器向 OF 交换机发起端口状态请求信息时,OF 交换机回复的 OFPT_STATS_REPLY 消息中包含了端口发送和接收的数据包数、出错的数据包数等信息。端口类型的 OFPT_STATS_REPLY 消息结构如下:

```
struct ofp_port_stats{
…
uint64_t rx_packets; //已接收的数据包数
uint64_t tx_packets; //已发送的数据包数
…
uint64_t rx_dropped; //接收时丢弃的数据包数
uint64_t tx_dropped; //发送时丢弃的数据包数
…
}
```

根据这些信息,控制器就能够计算出具体的丢包率值,用公式表达为:

$$PLR = \frac{tx\_dropped + rx\_dropped}{tx\_packets + rx\_packets} \times 100\% \quad (5)$$

4 实验环境搭建与仿真分析

采用 Floodlight[8] 和 Mininet[9] 搭建一个小型 SDN 网络,便于进一步理解控制器和 OF 交换机之间的工作机制。

4.1 仿真拓扑

Mininet 与 Linux 系统完全兼容,可以快速部署到硬件环境中,其内置的 OVS(Open vSwitch)虚拟交换机能够模拟出包含多终端和网络设备的复杂大规模计算机网络。在本例中,采用 Mininet 虚拟一个包含三个主机节点、一个 OF 交换机和一个远程控制器的拓扑结构,如图 3 所示。启动网络包分析工具 Wireshark[10],能够看到控制器和交换机之间定时发送的 OF_ECHO_REQUEST 和 OF_ECHO_REPLY 消息。在 Mininet 命令行界面运行“pingall”命令,执行结果如下:

$h_1 \rightarrow h_2 h_3$

$h_2 \rightarrow h_1 h_3$

$h_3 \rightarrow h_1 h_2$

由上述结果可知,在初始阶段,主机间处于相互连通状态。

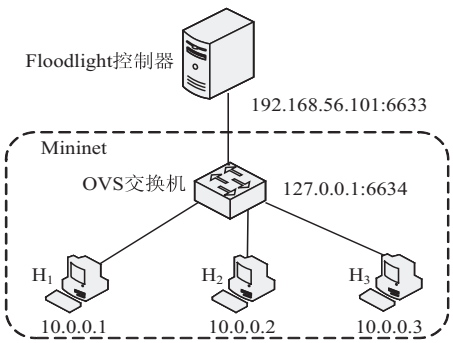


图 3 Mininet 创建的拓扑结构

Floodlight 控制器中的 LinkDiscoverManager 模块实现链路发现功能,该模块通过向 OVS 交换机下发 LLDP 数据包和 BDDP(broadcast domain discovery protocol)数据包来获取网络中每条链路的信息。TopologyManager 模块监听 LinkDiscoverManager 模块的更新信息,如果有链路状态更新,则采用 Dijkstra 算法[11] 进行节点间的路径计算并生成对应的拓扑实例。

由于 Floodlight 控制器中负责流表下发的 Forwarding 模块采用了反应式流表安装方法,当数据包到达 OVS 交换机后,初始状态下交换机内没有任何匹配的流,则 Forwarding 模块产生一些临时流表指导数据包继续转发。因此,当执行“pingall”命令后,主机之间依然能够通信。除了上述反应式流表模式,Floodlight 控制器还支持静态流表写入方法(static flow pusher)。定义一条测试流规则实现 H_1 到 H_2 之间的链路不可达,流规则描述如下:

```
FlowTable(H1-H2)={
    "switch":"00:00:00:00:00:00:00:01",
    "cookie":"0",
    "priority":"2",
    "ingress-port":"1",
    "active":"true",
    "actions":" output=3"
```

}

通过 curl 命令将该流规则安装到控制器中,并再次运行“pingall”命令,测试结果如下:

$h_1 \rightarrow xh_3$

$h_2 \rightarrow xh_3$

$h_3 \rightarrow h_1 h_2$

在 Floodlight 的 UI 页面能够看到命令执行过程中 OVS 交换机各端口的流量统计数据,如表 2 所示。流规则通过控制器的 8080 端口所支持的 REST[12](representational state transfer)接口下发到 OVS 交换机中,H_1 到 H_2 之间的链路执行“数据包直接丢弃”策略,因此无法 ping 通。H_1 到 H_3 以及 H_2 到 H_3 之间的链路执行“数据包正常转发”策略,因此链路是可达的。

表 2　OVS 交换机不同端口的数据统计

TX Bytes	RX Bytes	TX Pkts	RX Pkts
22 726	1 078	428	15
27 236	1 316	419	22
26 970	1 022	414	15

4.2　网络性能监测实验与分析

(1)吞吐量测量。

采用网络性能测试工具 iperf[13] 以不同的速率发送 UDP[14](user datagram protocol)数据包进行测量。选择 H_1 到 H_3 之间的链路作为测试路径,链路带宽设置为 100 M,时延为 10 ms。iperf 发包速率变化范围设置为 10 ~ 80 Mbps,共进行 8 次测试,每次测试时间为 60 s。测试 OVS 交换机在不同的负载流量速率下的吞吐量,测试结果如图 4(a)所示。由图可见,随着负载流量的不断增大,链路吞吐量也随之增大,当负载流量超过 40 Mbps 时,吞吐量维持在 23 Mbps 左右,是因为 OVS 交换机在进行查找流表并转发 UDP 数据包时,在交换机的缓冲队列中存在大量的数据包导致队列溢出而被丢弃。

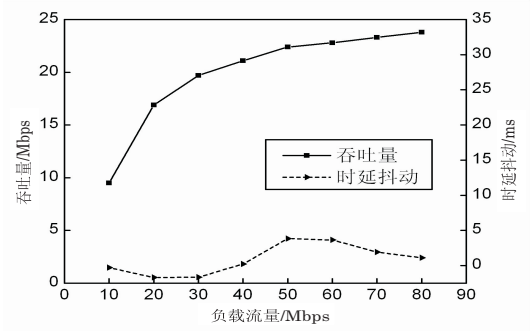
(2)链路时延测量。

利用 Ping 命令测试不同大小的数据包对链路时延的影响。选择 H_2 到 H_3 之间的链路作为测试路径,链路带宽设置为 10 M,数据包大小变化范围设置为 200 ~ 1 400 Byte,共进行 5 次测试,每次测试发送数据包数为 5 个,取平均时延值,测试结果如图 4(b)所示。

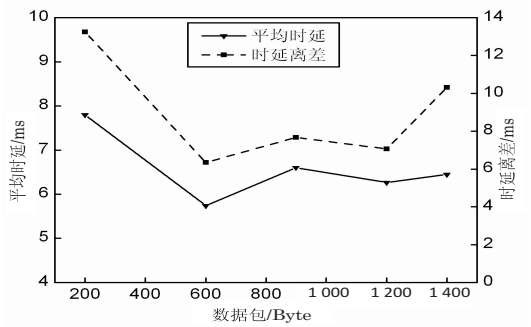
图 4(b)中的时延离差反映了数据包的往返时延距离平均时延的偏离程度,该值越大,表明响应时间的变化范围越大。当发送 200 Byte 的数据包时,H_2 首先通过 OFPT_PACKET_IN 消息向控制器发送 ICMP[15](Internet control message protocol)报文,控制器收到该报文后,添加流规则定义将该报文转发到 H_3。H_3 收到报文后,同样通过 OFPT_PACKET_IN 消息向控制器发送 ECHO 回显应答消息,该消息最终被转发给 H_2。至此,一个 Ping 命令的通信过程结束。上述过程的执行影响了数据包的往返时延,由于交换机中已存在相应的流规则,因此,后续发包的平均时延显著减少,趋于平稳状态。

(3)丢包率测量。

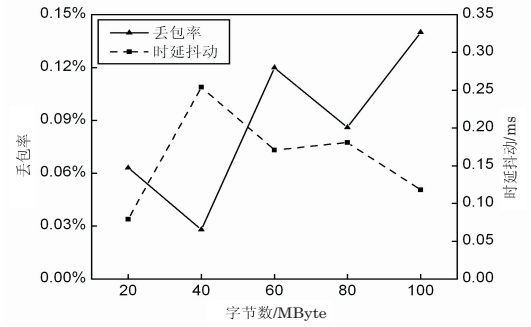
利用 iperf 工具传输不同长度的字节数进行链路丢包率的测量。选择 H_1 到 H_2 之间的链路作为测试路径,链路带宽设置为 100 M,时延为 10 ms,发包速率设置为 20 Mbps。待传输的字节数长度范围设置为 20 ~ 100 MB,共进行 5 次测试,测试结果如图 4(c)所示。由图 4(c)可见,所测得的丢包率值均在 2% 以下,说明网络处于正常状态。当传输数据量为 60 MB 时,丢包率陡然上升,表明网络拥塞程度增大。当传输数据量为 100 MB 时,丢包率达到峰值,此时 OVS 交换机端口缓冲区队列溢出大量数据包被丢弃,在这种过载的状态下,网络性能出现大幅下降。



(a)不同负载流量下的 UDP 吞吐量



(b)不同大小数据包的平均时延



(c)传输不同长度数据的丢包率

图 4　网络性能监测实验结果

5 结束语

SDN 是一种数据控制分离、具有灵活软件编程能力的新型网络架构,它通过提供开放式接口,实现对网络状态的集中控制并根据业务需求进行资源的全局调配和优化。文中介绍了 SDN 的主要实现方式 OpenFlow 技术,详细探讨了基于 OpenFlow 协议的链路发现机制和网络性能监测机制并搭建了基于 Floodlight 和 Mininet 的实验平台进行仿真测试。SDN 网络的目标是服务于多样化的业务应用创新,因此,在下一步的工作中,主要开展 SDN 应用编排与资源管理技术的研究。

参考文献:

[1] 左青云,陈 鸣,赵广松,等. 基于 OpenFlow 的 SDN 技术研究[J]. 软件学报,2013,24(5):1087-1097.

[2] 张团利,吕光宏,杨沛霖. 基于 OpenFlow 的 SDN 可靠性综述[J]. 电子科技,2016,29(2):177-181.

[3] 王 鹃,王 江,焦虹阳,等. 一种基于 OpenFlow 的 SDN 访问控制策略实时冲突检测与解决方法[J]. 计算机学报,2015,38(4):872-883.

[4] LI Wenjuan,MENG Weizhi,KWOK L F. A survey on OpenFlow-based software defined networks:security challenges and countermeasures[J]. Journal of Network and Computer Applications,2016,68:126-139.

[5] ALSMADI I, XU Dianxiang. Security of software defined networks:a survey[J]. Computers & Security,2015,53(9):79-108.

[6] VALLET J,BRUN O. Online OSPF weights optimization in IP networks[J]. Computer Networks,2014,60:1-12.

[7] 周 波,张代远. TCP 连接往返时延的被动测量算法实验验证[J]. 计算机技术与发展,2012,22(9):83-86.

[8] 高翔宇,王 雷,王振法,等. 支持协议无感知转发的 floodlight 控制器实现[J]. 微电子学与计算机,2015,32(11):92-96.

[9] 江国龙,付斌章,陈明宇,等. SDN 控制器的调研和量化分析[J]. 计算机科学与探索,2014,8(6):653-664.

[10] LAVADO L, PANIZO L, GALLARDO M D M, et al. A characterisation of verification tools for software defined networks [J]. Journal of Reliable Intelligent Environments, 2017,3(3):189-207.

[11] 王树西,李安渝. Dijkstra 算法中的多邻接点与多条最短路径问题[J]. 计算机科学,2014,41(6):217-224.

[12] 王仲洲,杨晓洪,王剑平,等. 基于 REST 风格的 WEB API 架构研究[J]. 微处理机,2016,37(5):52-55.

[13] TURULL D,SJÖDIN P,OLSSON R. Pktgen:measuring performance on high speed networks[J]. Computer Communications,2016,82:39-48.

[14] 张 新,饶若楠,郭 宁,等. 多端口自适应 UDP 通信协议的设计与实现[J]. 计算机工程与设计,2017,38(4):846-851.

[15] 史振华,刘外喜,杨家烨. SDN 架构下基于 ICMP 流量的网络异常检测方法[J]. 计算机系统应用,2016,25(4):135-142.

(上接第 158 页)

[3] 邹优嘉. 移动设备增强现实技术与书本交互的应用[J]. 计算机技术与发展,2013,23(8):227-229.

[4] 相茂英,马纯永,韩 勇,等. 基于 Unity3D 的化工设备虚拟培训系统研究[J]. 计算机技术与发展,2014,24(7):196-200.

[5] 周泽伟,冯毅萍,吴玉成,等. 基于虚拟现实的流程工业过程模拟仿真系统[J]. 计算机工程与应用,2011,47(10):204-208.

[6] JIANG Yongmei. Research on the application of virtual reality technology on industry automatic simulation [J]. Advanced Materials Research,2012,463-464:1155-1159.

[7] 陈 彬,邱晓刚,郭 刚. 多范式人工社会建模与多智能体仿真平台框架[J]. 系统仿真学报,2011,23(8):1702-1707.

[8] 邓兴升. 虚拟仿真平台下摄影测量实践教学模式探索[J]. 测绘工程,2015,24(9):74-76.

[9] 宋广东,王 昌,王金玉,等. 基于 DLL 技术和 COM 组件技术实现 LabVIEW 和 MATLAB 混合编程[J]. 计算机应用与软件,2013,30(1):287-289.

[10] 陆悌亮,龚声蓉. 基于 Web 的实时虚拟仿真系统研究与实现[J]. 计算机应用与软件,2008,25(1):275-276.

[11] 张凯选,李 鹏,马 强. VRML 与 3DS MAX 建模的互补性研究[J]. 计算机系统应用,2012,21(2):85-88.

[12] 侯 颖,许威威. 增强现实技术综述[J]. 计算机测量与控制,2017,25(2):1-7.

[13] PALETTA L,SANTNER K,FRITZ G,et al. FACTS-a computer vision system for 3D recovery and semantic mapping of human factors[M]//Lecture notes in computer science. Berlin:Springer,2013:62-72.

[14] IMBERT N,VIGNAT F,KAEWRAT C,et al. Adding physical properties to 3D models in augmented reality for realistic interactions experiments [J]. Procedia Computer Science, 2013,25:364-369.

[15] 王 华,徐明亮,毛天露,等. 三维汽车群组动画仿真研究综述[J]. 计算机辅助设计与图形学学报,2017,29(2):211-220.

[16] LIN Fuhua,YE Lan,DUFFY V G,et al. Developing virtual environments for industrial training [J]. Information Sciences,2002,140(1-2):153-170.

[17] 王庆月. 基于 CAVE 的虚拟现实关键技术研究[J]. 现代电子技术,2017,40(1):140-144.