计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(9):93−99 [doi: 10.15888/j.cnki.csa.006561]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于协议无感知转发的服务功能链①

刘正印[1,2], 葛敬国[2], 李　佟[2], 韩春静[2], 吴嘉磊[3]

[1](中国科学院大学 网络空间安全学院, 北京 100093)
[2](中国科学院 信息工程研究所, 北京 100093)
[3](南京理工大学 计算机科学与工程学院，南京 210094)
通讯作者: 韩春静, E-mail: hanchunjing@iie.ac.cn

摘　要: IETF 提出的服务功能链 (Service Function Chain, SFC) 解决了服务功能在部署过程中网络拓扑与硬件设备紧密耦合、灵活性差等问题, 其中, NSH 协议用于支持服务功能链的实现. 然而, 标准的 OpenFlow 协议对 NSH 协议支持不足、实现过程复杂且实现后造成兼容性等问题. 本文基于软件定义网络 (Software Defined Network, SDN) 和网络功能虚拟化 (Network Function Virtualization, NFV) 技术, 根据 IETF 规定的相关标准, 提出一种基于协议无感知转发 (Protocol Oblivious Forwarding, POF) 的服务功能链, 利用 POF 在数据平面深度可编程的能力实现 NSH 协议. 文中基于 FloodLight 控制器和 POF 交换机实现了该服务功能链, 实验结果表明, 基于协议无感知转发的服务功能链可以高效地实现服务功能的部署.

关键词: 软件定义网络; 网络功能虚拟化; 服务功能链; NSH; 协议无感知转发

引用格式: 刘正印,葛敬国,李佟,韩春静,吴嘉磊.基于协议无感知转发的服务功能链.计算机系统应用,2018,27(9):93–99. http://www.c-s-a.org.cn/1003-3254/6561.html

Service Function Chain Based on Protocol Oblivious Forwarding

LIU Zheng-Yin[1,2], GE Jing-Guo[2], LI Tong[2], HAN Chun-Jing[2], WU Jia-Lei[3]

[1](School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China)
[2](Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)
[3](School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China)

Abstract: The Service Function Chain (SFC) proposed by the IETF solves the problems that the service functions are tightly coupled with the hardware devices and have poor flexibility in the deployment process. The NSH protocol is used to support the implementation of the SFC. However, the standard OpenFlow protocol is not sufficient to support the NSH protocol and causes the problems of compatibility after implementation. Based on Software Defined Network (SDN) and Network Function Virtualization (NFV) technologies, this study proposes an SFC based on Protocol Oblivious Forwarding (POF), which implements the NSH protocol using the ability of POF that can be programmed deeply in the data plane. In this study, we implements the SFC based on FloodLight controller and POF switch. The experimental results show that the SFC based on POF can efficiently implement the deployment of service functions.

Key words: Software Defined Network (SDN); Network Function Virtualization (NFV); Service Function Chain (SFC); NSH; Protocol Oblivious Forwarding (POF)

① 基金项目: 国家重点研发计划 (2017YFB0801801); 国家科技重大专项 (2017ZX03001019-003)
Foundation item: National Key Research and Development Plan of China (2017YFB0801801); National Science and Technology Major Project of China (2017ZX03001019-003)
收稿时间: 2018-01-31; 修改时间: 2018-03-13; 采用时间: 2018-03-14; csa 在线出版时间: 2018-08-16

服务功能在部署时, 功能与专属硬件紧密耦合, 每个功能都嵌入在特定的硬件设备中, 导致运营成本不断提高, 网络灵活性变差, 服务部署十分困难[1]. 为此, IETF 提出一种服务功能链 (Service Function Chain, SFC) 架构来解决服务功能在部署过程中拓扑独立性和配置复杂性等问题[2]. NSH 以一种报头的格式被添加到网络流量中, 用于支持服务功能链架构的实现[3].

当前, 利用软件定义网络[4](Software Defined Network, SDN) 和网络功能虚拟化 (Network Function Virtualization, NFV) 技术实现服务功能链已经成为了研究热点. SDN 和 NFV 技术高度互补, 彼此互利但并不相互依赖. SDN 被视为一种全新的网络技术, 通过分离网络设备的控制平面与数据平面, 将网络能力抽象为应用程序接口提供给应用层, 从而构建开放可编程的网络环境, 在对底层各种网络资源虚拟化的基础上, 实现对网络的集中控制和管理. NFV 利用 IT 虚拟化技术定义网络功能, 通过通用硬件以及虚拟化技术, 来承载相关网络功能, 从而降低网络成本并提升业务开发部署能力, 将硬件和软件解耦, 使网络功能不再依赖于专用硬件.

到目前为止, SFC 已经在多个平台上实现, 包括 OpenDaylight (ODL)[5], ONOS[6] 和 OpenStack Networking[7]. 其中, 只有 ODL 平台完全采用 IETF SFC 架构, 包括所有核心组件、NSH 协议等. ODL 平台通过对 OpenFlow 协议的私有扩展实现了 NSH 协议, 然而, 这种实现方式造成了 OpenFlow 的兼容性问题. 由于 ODL 平台中 OpenVSwitch(OVS) 交换机[8]通过 OpenFlow 协议与控制器通信, 因此, OVS 交换机需要修改自身代码以实现 NSH 协议, 实现过程复杂. OVS 交换机首先对 L2 或 L3 层的数据包添加 NSH, 然后利用现有封装协议, 如 MPLS、VXLAN/VXLAN-GPE 在 L3 或 L4 层对数据包再次封装. 利用现有协议封装、转发 NSH 报文, 增加了不必要的报文处理过程, 导致网络传输效率降低.

POF 协议是对 SDN 南向接口协议 OpenFlow 的扩展, 其提供了 SDN 控制器直接指定规则匹配字段和指令操作字段的偏移量和长度的能力, 定义了字段修改、插入、删除等指令集用于对报文进行各种操作, 从而实现了数据平面的深度可编程, 更加有效地支持新协议的实现.

本文根据 IETF 提出的有关服务功能链的标准, 提出了一种基于协议无感知转发的服务功能链, 该服务功能链主要是基于 SDN 和 NFV 技术, 基于 Floodlight 开源控制器, 利用 POF 协议数据平面深度可编程的能力, 提供了对 NSH 协议的支持, 从而构建了服务功能链. 实验结果表明, 该架构有效地实现了服务功能的部署.

本文其余小节组织如下: 第 1 节介绍服务功能链的相关背景; 第 2 节提出基于 SDN 和 NFV 技术的服务功能链的架构设计; 第 3 节中, 详细介绍该服务功能链架构的实现过程; 第 4 节, 通过设计实验并验证了该架构的可行性和系统性能. 最后, 对下一步工作进行相关介绍.

1 背景

1.1 服务功能链概述

SFC 是指定义和实例化一组有序的服务功能, 例如防火墙、深度报文检测系统等, 通过控制网络流量的处理顺序, 高效灵活地为用户提供所需的服务[2].

IETF 对服务功能链进行了标准化定义, 对架构框架、使用场景、路由转发和报文格式等进行了详细阐述[2]. 图 1 展示了 IETF 提出的 SFC 架构, 其包括 4 个核心组件: 分类器 (Classifier)、服务功能转发器 (SFF)、服务功能 (SF) 和 SFC 代理 (SFC Proxy). 当数据包进入到一个支持 SFC 报文的网络区域即 SFC-Enabled Domain 中后, Classifier 根据匹配策略匹配数据包, 将其分配到不同的 SFC 中; SFF 根据 SFC 报文中携带的信息将数据包转发到一个或多个 SF 中, 并且处理从 SF 返回的数据包; SF 负责对收到的数据包做特定功能的处理, SF 在具体实现的上可以是一个虚拟的元素, 或者是嵌入在具体网络设备上的某种功能, 如防火墙, 深度报文检测系统等; 对于不能识别 SFC 报文的设备, 可以使用 SFC 代理.

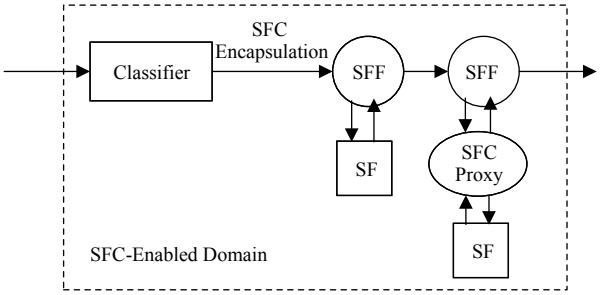

图 1 SFC 逻辑组件

在服务功能链中, 服务功能路径 (Service Function Path, SFP) 是指数据包必须按照指定顺序转发的限制规范, 而数据包在网络中实际访问 SFF 和 SF 的顺序称为呈现服务路径 (Rendered Service Path, RSP). 从 SFP 到 RSP, 是一个从抽象定义到具体实现的逐步细化的过程.

1.2 NSH 协议

为引导网络流量在服务功能链的各个逻辑组件中正确传递, 解决 SF 之间信息交互, 拓扑独立性等问题, IETF 提出了用于创建动态服务功能链的 NSH 协议. 通常, 数据包在网络入口处被添加 NSH, 数据平面根据 NSH 报文完成服务流量的控制和转发. 如图 2 所示, 一个 NSH 包含 3 部分: 四个字节的基本报头、四个字节的服务路径信息和上下文信息 (固定长度和可变长度两种). 其中, 四个字节的基本报头提供有关 NSH 报文本身的信息以及有效负载的信息; 服务路径信息主要是提供服务路径中的路径标识和位置; 上下文信息用来携带元数据 (metadata).

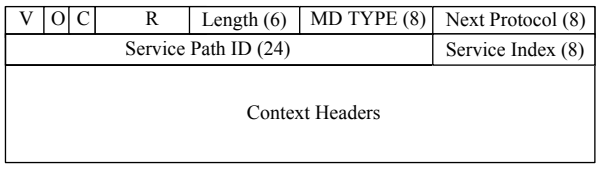

图 2 NSH 报文

2 系统架构设计

在 IETF 关于服务功能链的 RFC 文档中提出了服务功能链架构的实现原则, 其中包括拓扑独立性、平面分离、数据包分类和共享元数据等[2]. 本文基于 SDN 和 NFV 技术, 将服务功能链的系统架构分为控制平面和数据平面 (图 3). 控制平面通过北向接口获取用户对于服务功能链的描述, 生成相应的策略和流表信息, 通过控制平面和数据平面间的南向接口传递到数据平面, 数据平面根据控制平面下发的策略和流表, 将各个逻辑组件实例化, 并使数据包按照指定的顺序经过各个 SF, 实现用户所需的服务.

控制平面主要功能包括: (1) 提供北向接口, 接收用户对于服务功能链的描述信息; (2) 管理底层网络基础设施、链路状态、拓扑结构; (3) 负责资源的统一分配, 各个逻辑组件的创建和生命周期的管理; (4) 管理 SFC 的相关信息, 初始化各个监听服务, 持久化相关信息; (5) 构建 SFP, 生成相关策略和流表; (6) 提供南向接口与数据平面进行信息传递. 控制平面通过北向接口, 获取到用户对于服务功能链的描述, 资源管理模块根据描述分配相应的计算和网络资源, SFP 路径规划模块根据服务功能链的描述和资源管理模块资源分配结果, 利用 Floodlight 控制器中原有的对底层网络拓扑管理的模块, 初始化 SFP, NSH 流表生成模块根据实例化的结果, 生成相应的流表, 通过控制平面和数据平面之间的南向接口下发至数据平面, 控制平面的数据库模块对 SFC 的相关信息做持久化工作.

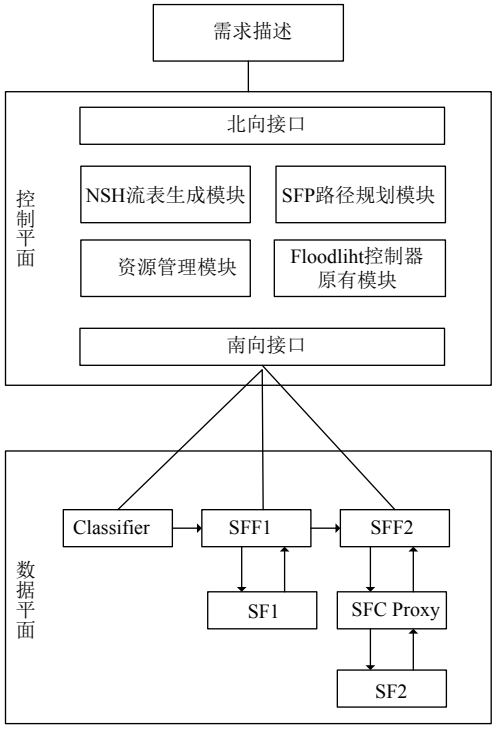

图 3 SFC 架构

当数据包进入数据平面的 SFC-Enabled Domain 时, 入口处 Classifier 根据控制平面下发的流表决定哪些包能够进入 SFC-Enabled Domain, 并根据下发的匹配规则为这些数据包添加不同的 NSH, 转发至相应的 SFF 中. SFF 会根据 NSH 报文中服务路径标识 (Service Path Identifier, SPI) 和服务序号 (Service Index, SI) 确定下一跳地址, 其中, SPI 是 SFP 的唯一标识, SI 表示 SF 在 SFP 中的位置. 当 SF 不能识别 NSH 时, SFF 会将数据包发送给相应的 SFC Proxy 进行处理.

3 系统实现

根据系统的架构设计, 将系统实现分为了控制平

面和数据平面的实现两个部分. 控制平面采用 Floodlight 开源控制器, Floodlight 不仅是一款 SDN 控制器, 它也包含一系列模块化应用, 而这些应用可以向上提供 REST API, 从而帮助应用层的应用更好地管控整个网络. POF 交换机协议无感知转发的特点, 有效的支持了 NSH 协议在数据平面的实现, Classifier 和 SFF 等网络功能组件是基于 POF 交换机开发完成的. 由于控制器主要通过流表的形式对 POF 交换机进行管理, 因此, 流表生成模块的构建在控制器中十分关键.

3.1 基于 Floodlight 控制器的控制平面的实现

SFC 控制平面主要是基于 Floodlight 控制器实现的, 用户可以通过控制器提供的北向接口使用 SFC 的功能, 可以创建、更新或者删除 SFC, 自定义非透明的 metadata 数据字段用来实现 SF 间的数据共享, Floodlight 控制器还可以对底层网络抽象, 获取网络的拓扑结构.

Floodlight 控制器提供了一个模块加载系统, 可以方便的利用 IOFMessageListener 和 IFloodlightModule 这两个接口进行功能扩展[9]. 通过实现 IFloodlightModule 接口, 我们增加了 NSH 流表生成模块, 将用户对于服务功能链的路径描述信息, 通过流表的形式下发至数据平面, 数据平面的 POF 交换机根据流表信息对报文添加不同的 NSH, 实现数据包在数据平面的传递. 为了实现 SFC 逻辑组件的自动化部署和管理, 利用 python 语言编写了资源管理模块. 由于 SFC 的生成是根据用户的需求来定义的, 因此我们采用了 JSON 类型的接口格式, 通过 JSON 接口, 控制平面可以接收来自用户层面的 SFC 需求描述. 另外, 控制器中还定义了与多个与 SFC 相关的数据结构, 例如 NSH 报文信息等.

3.2 NSH 流表设计

服务功能链各个逻辑组件对数据包的处理模式可以归结为 match-action 的处理过程, 即通过匹配数据包中的某个字段, 匹配成功后则用 action 进行处理, 如 Classifier 匹配到目的地址 A 的数据包, 则封装 NSH 并转发. 因此, 流表的设计主要是对 match-action 策略的设定.

由于服务功能链中不同的逻辑组件实现的功能不同, 因此下发的流表也不一样. Classifier 需要提供对数据包的识别、封装和转发等功能. 本文中 Classifier 支持对数据包中五元组 (目的 IP、源 IP、目的端口号、源端口号和协议) 的识别, 由于 POF 交换机一次性执行的指令数量的限制, 无法一次性完成五元组的识别和匹配功能, 需要利用多个流表对同一数据包进行处理, 每个流表处理后将结果暂存到 POF 交换机中的 metadata 字段中, metadata 字段设计如图 4 所示, 当 POF 交换机依次将数据包中的五元组写入交换机的 metadata 字段后, Classifier 中负责处理 NSH 报文的流表利用 metadata 中的五元组, 根据 POF 交换机中设定的匹配域, 实现 NSH 报文的解析与传递.

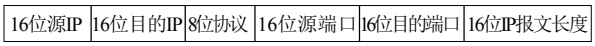

图 4 交换机中的 metadata 字段

根据上述分析, 控制平面共下发至 Classifier 中 6 个流表, 其处理过程如图 5 所示, 由于 POF 交换机接收到的是 MAC 帧报文, 包括携带 VLAN 和不携带 VLAN 字段的两种格式, 因此 Classifier 中的流表首先是对 MAC 帧的处理, 从而确定 IP 报文开始的位置, 接着依次是 IP 报文和 TCP/UDP 报文的处理. 五元组写入到 POF 交换机的 metadata 中后, 将流表的匹配域与 metadata 中的五元组相对应, 从而实现对数据包的匹配. 匹配完成后, 利用 POF 协议无感知转发的特点, 只需要通过 AddField、ModifyField 和 DelField 等 POF 交换机指令便能完成对数据包添加 NSH 报文头等操作, 不需要对其他代码进行修改, 添加完成后将带有 NSH 报文转发到不同的 SFC 中.

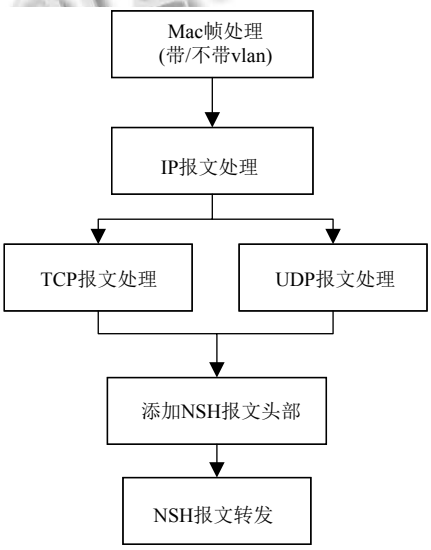

图 5 Classifier 流表处理过程

在 SFC 中, SFF 根据 NSH 报文中携带的信息将数

据包转发至一个或多个 SF 中, 并且处理从 SF 返回的数据包. SFF 报文处理流程如图 6 所示, 当 SFF 收到报文后, 首先根据 NSH 报文中的 SPI 和 SI 字段确定下一跳地址, 转发到相应的 SF 或 SFF 中, 如果 SF 不支持 NSH 报文, 则需要先转发到相应的 SFC Proxy 中将 NSH 去掉, 转换成 SF 可以识别的报文, 当 SF 处理完毕后, SF Proxy 需要重新对数据包封装 NSH 再由 SFF 转发到下一跳. 由于 Classifier 不会对匹配策略之外的数据包封装 NSH 报头, 因此在 SFF 还需要支持对常规报文的转发.

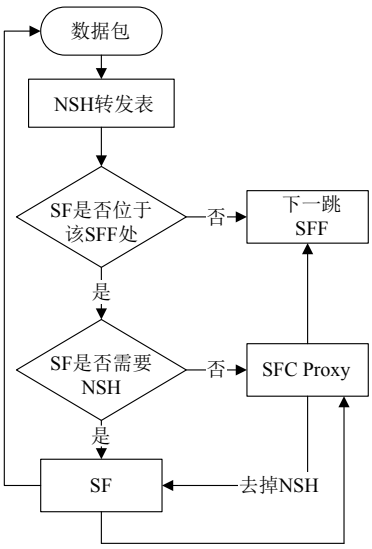

图 6 SFF 处理过程

4 实验设计及验证

为了对该系统的功能和性能进一步分析和研究, 本文设计并搭建了与实际网络环境相近的网络拓扑. 本实验的实验环境使用的是 ubuntu14.04 的系统, 内核版本是 3.13, 网络拓扑利用 mininet 搭建而成 (mininet 是一个轻量级的 SDN 网络研发、仿真以及测试平台). 控制器采用 Java 语言编写的 Floodlight 开源控制器, 安装的 JDK 版本是 1.8.

4.1 系统功能测试

本文首先对服务功能链的功能效果进行了实验验证, 实验模拟的是 Client 请求 Web Server 服务器的 HTTP 服务, 其中, Classifier 和 SFF 均基于 POF 交换机设计而成. 本实验设计了两条服务功能链, 一条功能链 SFC1 经过的服务功能路径是 SF1、SF2、SF3, 另一条经过的服务功能路径 SFC2 是 SF2、SF3. NSH 包含 3 部分: 基本头信息、服务路径信息和上下文信息. 实验中两条服务功能链的 SPI 分别设为为 1 和 2, SI 的初始值设为 255, Next protocol 设为 4(4 表示 NSH 协议), 上下文信息的初始化长度为 4 个字节固定长度, 初始值为 0. 在 NSH 报文转发表中, 利用 SPI、SI 和 Next Protocol 确定下一跳地址, 下一跳地址可以是 SF 或者 SFF, 每次经过一个 SF 后, SFF 负责将 SI 减 1. 在节点通信过程中存在的 ARP 请求报文, Classifier 不会对其进行封装. 表 1 是控制器下发至 SFF1 中的 NSH 报文转发表, 其中*号代表通配符.

表 1 SFF1 中的 NSH 报文转发表

Index	Match filed				Action filed		Priority
	Protocol	SPI	SI	Input_port	Action	Next hop	
0	NSH	01	255	S2-eth1	output (port: S2-eth2)	SF1	100
1	NSH	01	255	S2-eth3	modify (SI-1)output (port: S2-eth4)	SFF2	100
3	NSH	02	255	S2-eth1	output (port: S2-eth4)	SFF2	100
4	ARP	*	*	S2-eth1	output (port: S2-eth4)		20
5	ARP	*	*	S2-eth4	output (port: S2-eth1)		120

实验的网络拓扑如图 7 所示, 其中 Client1 主机的 IP 地址为 10.0.0.1, Client2 主机的 IP 地址为 10.0.0.2, Server 服务器的地址为: 10.0.0.3, 控制器的地址为 10.10.28.37. 实验使源 IP 地址为 10.0.0.1 的数据包经过 SFC1, 源 IP 地址为 10.0.0.2 的数据包经过 SFC2. 由于 mininet 网络中的网络设备默认只能添加一条链路, 因此对 linux 主机添加了虚拟网卡对, SFF 和 SF 间分别关联了虚拟网卡对. 当数据包进入 SFC-Enabled Domain 后, Classifier 根据流表, 匹配数据包并添加 NSH, 分别将发自 Client1 和 Client2 的 NSH 中 SPI 字段设为 1 和 2, 添加完成后, 将 NSH 报文从 S1-eth3 端口转发至 SFF1, SFF1 根据流表中的 Next protocol、SPI 和 SI 确定下一跳地址, 如果匹配到的 SPI 为 1, 则将匹配到的数据包从 S2-eth2 端口转发之 SF1, SF1 处理完毕后, 将数据包从 F1-eth2 端口转发回 SFF1, SFF1 将数据包中的 SI 减 1 后从 S2-eth4 端口转发到 SFF2 处, 依次类推, 当 SF3 将处理完的数据包返回给 SFF3 时, SFF3 去掉数据包的 NSH 报头后, 转发至 web

Server 服务器. 当 SFF1 匹配到到数据包中的 SPI 为 2 时, 直接从端口 S2-eth4 转发至 SFF2. 在实验中, SF1 起到了防火墙的功能, 本文将防火墙规则设置为不允许目的 IP 地址为 10.0.0.3 的请求数据包通过, 因此 SPI 为 1 的数据包无法收到 HTTP 响应报文. 此外, Client 的发起的 HTTP 请求报文采用 NSH 协议进行转发, Web Server 返回的数据由于不需要再经过 SF, 直接按照常规的数据包进行返回.

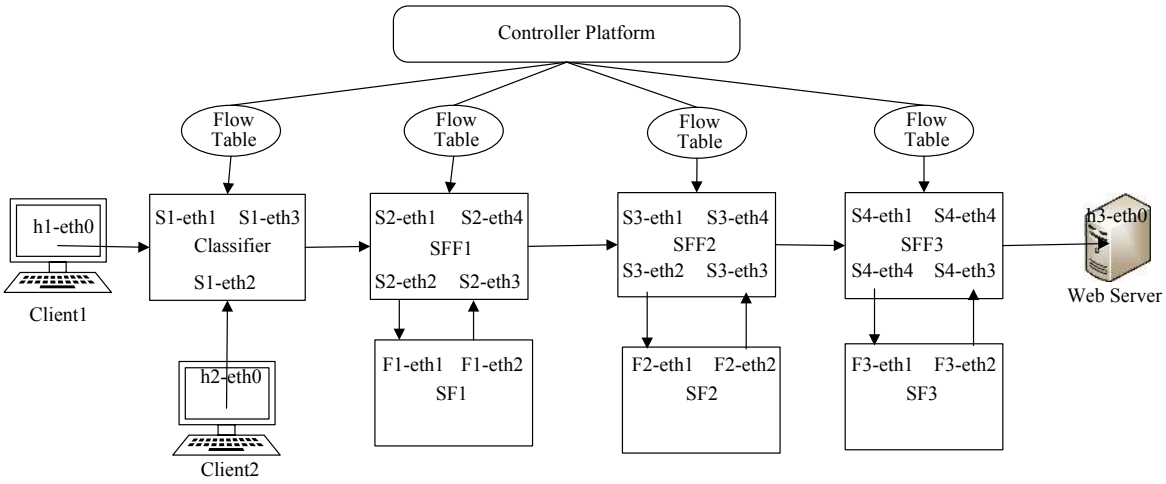

图 7 实验拓扑

实验结果如表 2 所示, 表中第一条记录是 SFC1 的 HTTP 请求结果, 第二条记录是 SFC2 的 HTTP 请求结果.

表 2 Client 请求结果

Index	HTTP request	HTTP response
1	Connectting to 10.0.0.3:80	Failed: Connectiontimed out.
2	Connectting to 10.0.0.3:80	200 OK

上述实验证明, 基于协议无感知转发的服务功能链实现了 IETF 提出的有关 SFC 的标准, 各个逻辑组件功能完备, 并基于 POF 实现了 NSH 协议.

4.2 系统性能测试

在性能测试时, 为了更加接近真实的网络环境, 实验利用了一台物理服务器和 IXIA 网络测量仪对服务功能链的丢包率和时延进行了测试. 实验设计了对照实验, 由于 Classifier 和 SFF 是基于 POF 交换机设计实现的, 因此, 对比了服务功能链环境下和 POF 交换机环境下, 常规 TCP 报文和 NSH 报文的在相同网络环境下的丢包率和时延情况, 实验中数据包的发送速率 50 Mbit/s. 图 8 中左框内是服务功能链性能测试的实验拓扑图, 右框内是 POF 交换机性能测试的实验拓扑图. 在服务功能链性能测试的实验中, IXIA 测试仪的 Port0 端口不停发送 TCP 数据包, Classifier 对据包封装 NSH 后转发至 SFF, SFF 收到 NSH 报文后, 修改 NSH 中的 SPI 字段, 然后去掉数据包中的 NSH, 转发至 IXIA 测量仪的 Port1 端口. 在对照试验中, TCP 数据包直接通过 POF 交换机转发, 不经过 Classifier 和 SFF.

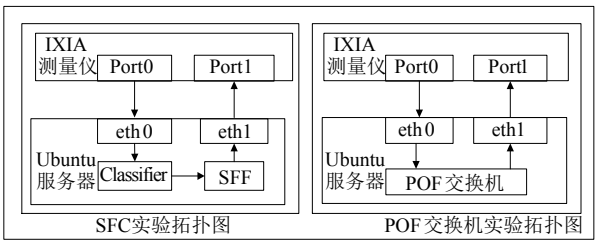

图 8 丢包率和时延实验拓扑图

两组对照实验的报文丢包率均接近于 0, 数据包的接收时延如图 9 所示, 基本趋于一致.

接着, 实验利用 iPerf3 网络性能测量工具和 mininet 仿真工具对服务功能链的吞吐量进行了测试, 测试分为两组, 将服务功能链与 POF 交换机设置为对照实验, 利用 iPerf3 将数据包的发送速率设置成不同的值, 采用 UDP 模式进行了测试. 图 10 的上下两框内分别是服务功能链和 POF 交换机测试吞吐量时的逻辑拓扑图.

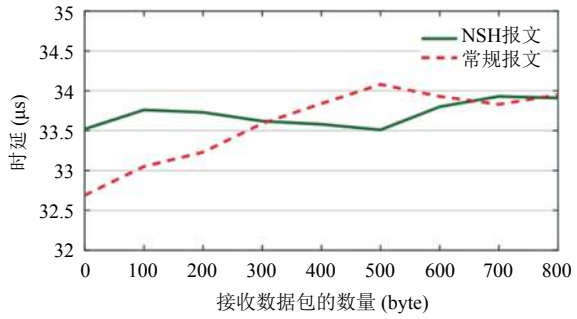

图 9 时延数据

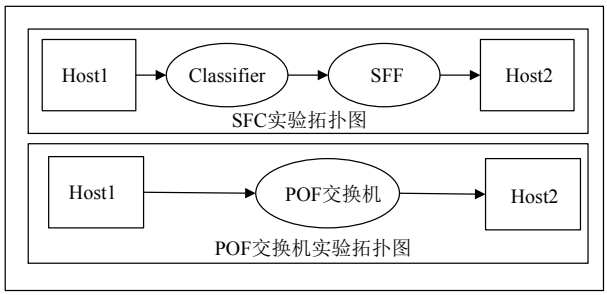

图 10 吞吐量实验拓扑图

吞吐量的实验结果如图 11 所示. 从实验结果可知. 在一定范围内, 服务功能链和 POF 交换机的吞吐量无明显差异, 随着发送速率的提高, 服务功能链网络吞吐量会逐渐下降, 维持在 380 Mbit/s 左右.

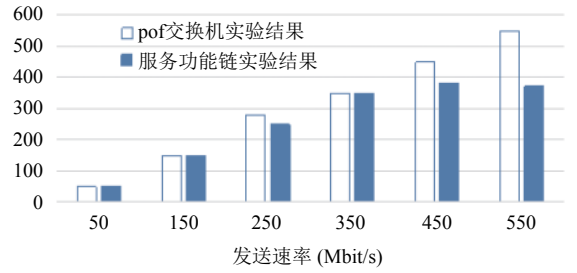

图 11 吞吐量实验拓扑图

由于 Classifier 和 SFF 均是基于 POF 交换机设计实现的, 根据丢包率、时延和吞吐量的测试结果可知: 在一定的数据包发送速率下, 利用 POF 交换机实现 NSH 报文协议后对交换机本身性能没有明显影响, 说明了 POF 协议能够高效的提供对 NSH 协议的支持, 避免了 OpenFlow 协议在扩展过程中的兼容性和实现复杂性等问题. 说明了基于协议无感知转发的服务功能链在性能方面表现优异.

5 结束语

本文根据 IETF 提出的关于服务功能链的标准, 利用 POF 协议无感知的特点, 提出基于 SDN 和 NFV 架构的服务功能链, 实现了 NSH 协议, 避免对 OpenFlow 协议的私有扩展和实现复杂性等问题. 利用 Floodlight 控制器和 POF 交换机实现了该服务功能链, 并对其功能和性能分别进行了验证, 实验结果表明, 基于协议无感知转发的服务功能链功能完备, 在性能方面表现优异. 由于本文侧重 SDN 的实现模块, 对 NFV 模块的研究不够深入, 也没有考虑到包括故障处理在内的其他因素, 下一步工作将围绕着云计算环境中协议无感知转发的服务功能链的应用进行深入研究.

参考文献

1 Han B, Gopalakrishnan V, Ji LS, *et al*. Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine, 2015, 53(2): 90–97. [doi: 10.1109/MCOM.2015.7045396]

2 Halpern J, Pignataro C. Service Function Chaining (SFC) Architecture. Internet-Draft draft-ietf-sfc-architecture-07, Alia Atlas. 2015.

3 Quinnand P, Elzur U. Network service header (NSH). Internet-Draftdraft-ietf-sfc-nsh-07, IETF Secretariat, August 2016. https://tools.ietf.org/html/draft-ietf-sfc-nsh-07.

4 McKeown N, Anderson T, Balakrishnan H, *et al*. OpenFlow: Enabling innovation in campus network. ACM Sigcomm Computer Communication Review, 2008, 38(2): 69–74. [doi: 10.1145/1355734]

5 Service function chaining: Main, OpenDaylight Project, Wiki. Opendaylight.org, 2016. https://wiki.opendaylight.org/view/ServiceFunctionChaining:Main. [2016-09-09].

6 Open network operating system, ONOS, 2017. https://www.sdxcentral.com/projects/on-lab-open-network-operating-system-onos/. [2017-08-01].

7 Service function chaining documentation networking-sfc 2.0.1. dev1 documentation, docs. Openstack. org, 2016. https://docs.openstack.org/networking-sfc/latest/. [2016-09-09].

8 Open vSwitch, Open vSwitch. org, 2016. http://OpenVSwitch.org. [2016-09-09].

9 周环, 刘慧. 基于 Floodlight 的 SDN 控制器研究. 计算机工程与应用, 2016, 52(24): 137–147. [doi: 10.3778/j.issn.1002-8331.1603-0330]