

支持协议无感知转发的 Floodlight 控制器实现

高翔宇, 王 雷, 王振法, 张云灿, 刘 静

(中国科学技术大学 信息科学技术学院, 安徽 合肥 230027)

摘 要: 无协议感知转发(POF)交换机在转发数据时去除了对于特定协议的依赖,使得 SDN 控制器具有更高的灵活性和可编程性. 现有 SDN 控制器不支持 POF 协议,通过扩展 Floodlight 开源控制器的功能,添加了链路发现、拓扑管理、设备管理、数据转发等模块,实现了支持 POF 协议的控制器. 实验结果表明,该控制器可以有效管理 POF 交换机并支持自定义转发协议的制定和执行.

关键词: 软件定义网络; OpenFlow; 支持协议无感知转发; 控制器

中图分类号: TP309

文献标识码: A

文章编号: 1000-7180(2015)11-0092-05

Implementation of the Floodlight Controller with
Protocol Oblivious Forwarding

GAO Xiang-yu, WANG Lei, WNAG Zhen-fa, ZHANG Yun-can, LIU Jing

(School of Information Science and Technology, University of
Science and Technology of China, He'fei 230027, China)

Abstract: Switch with Protocol-Oblivious Forwarding doesn't rely on the support of special protocol, which makes the controller more flexible and more programmable. Current SDN controller doesn't support Protocol-Oblivious Forwarding. In order to let the SDN controller support Protocol-Oblivious Forwarding, Modules are added to expand Floodlight controller's function. The modules are LinkDiscovery, TopologyManager, DeviceManager and Forwarding. Experiment shows that the controller can manage POF switch and can support the development and execution of custom-defined protocol with POF.

Key words: software defined networks; OpenFlow; POF; controller

1 引言

随着网络规模的不断增大以及网络应用的不断增多,传统网络架构越来越不能满足需求. 网络安全等问题也愈发突出^[1-2],而且网络研究者对已有网络进行的创新性实验很难在真实网络中部署. SDN 网络架构的出现使得网络的控制功能与数据转发功能分离^[3],在 SDN 架构下通过软件编程的方式实现了网络的智能性,因此也被称作“网络操作系统(NOS, Network Operating System)^[4]”. 下层的转发平面主要负责转发数据的功能,需要保持简单、灵活的特性.

OpenFlow^[5]是专为 SDN 设计标准接口,实现了 SDN 控制与转发分离的设计目标. OpenFlow 交换机通过 OpenFlow 协议和控制器交互,控制器通过 OpenFlow 协议控制和管理 OpenFlow 交换机. 一个 OpenFlow 交换机一组流表,以实现数据包的查找、匹配、转发^[5]. OpenFlow 每个流表都包含一组流条目,流条目包括匹配域、计数器和一组指令. 其中匹配域是现有 OpenFlow 协议中实现的、与传统网络中的协议相匹配的包的头域,被称为元数据(Metadata). 在现有 OpenFlow 1.0 协议中实现的元数据有 12 个,在 OpenFlow 1.3^[6]版本中增加到了 40 个,随着对网络协议栈的实现的完善,匹配域

收稿日期: 2014-12-14; 修回日期: 2015-02-04

基金项目: 中央高校基本科研业务费专项资金(WK2100100022); 中科院先导专项课题(XDA06011202)

的数目仍会增加. 这样的模式下, OpenFlow 只能一步步地增加对现有协议的支持. 每次增加对新协议的支持, 交换机都要做出相应修改, 这样的缺陷阻碍了 SDN 的可编程性和灵活性^[7]. 因此, 文献^[7]在 OpenFlow 的基础上提出了 POF, 改进了 OpenFlow 的实现, 使转发平台完全协议无感知. 并在此基础上开发了简单的 POF 控制器和 POF 交换机.

POF 是对于 OpenFlow 协议的改进, 其处理流程和 OpenFlow 的定义没有不同, 但在 OpenFlow 中要明确指明所要匹配的数据包的元数据(如 IPv4 目的地址). 这需要转发平面能够解析数据包, 根据解析的数据包进行匹配. 而在 POF 中使用一种统一的方式来表示匹配域, 不需要在 OpenFlow 中预先定义元数据, 也就不需要在交换机中实现具体的协议的表示与解析. POF 使用一个或多个 {offset, length} 这样的元组来表示匹配的一个或多个域. 其中 offset 表示偏移量, length 表示域的二进制位数. 为了使协议更加通用, 删除了 push MPLS label、decrement IP TTL 这样的 OpenFlow 指令. 使用 insert、delete、modify 这样指令作为替代以修改域的值, {offset, length} 则用来定位所要修改的域的位置和长度.

这样做的好处显而易见, 比如在 OpenFlow 1.3 定义了三种打标签的操作能够添加 Vlan、MPLS 和 PBB 头, 但像 VPLS 这样的网络协议需要添加以太网头, OpenFlow 就不能实现, 而这样的情形在未来会持续增加. 在 POF 中通过 AddField 指令便能完成三种操作, 只需通过参数 {offset, length} 来指定位置和长度. 由此文献^[7]中提出了 POF 协议, 同时为了支持 POF 控制器, 开发 POFSwitch 和简单的 POF 控制器^[8].

本文的主要工作基于开源的 Floodlight 控制器以实现支持 POF 协议的 SDN 控制器, 实现对 POF 交换机的管理并支持用户自定义转发规则的制定与执行, 实现了 OpenFlow 控制器核心模块.

2 POF 控制器需求分析

POF 控制器基于 Floodlight 开发而成, 但 POF 控制器只是一个原型系统. 目前 POF 控制器主要关注于无协议感知转发, 而对于传统 OpenFlow 控制器(如 Floodlight、POX 等)中的其他功能和模块并不感兴趣. 为了简化系统开发, POF 控制器时仅仅保留了原 Floodlight 控制器的核心功能. POF 控制器保留的 Floodlight 的主要功能模块包括模块及资

源加载模块、FloodlightProvider 模块. 资源加载模块用于系统的初始化以及模块的加载; FloodlightProvider 模块主要有两个功能, 一是处理和交换机的连接, 二是把交换机发送的 OpenFlow 消息转化成事件并调用相应模块处理 OpenFlow 消息^[7]. 另外 POF 实现了 GUI 功能, 用于为交换机制定流表, 但只能通过手动方式下发流表.

为了快速实现一个能支持 POF 核心功能的控制器, 以便对 POF 协议进行验证以及实现一个 POF 实验平台. POF 控制器仅实现了通过 GUI 生成流表的功能, 而没有自动生成流表为新的流提供转发支持的模块. 因此 POF 控制器除了能够与交换机建立连接以及通过手动方式建立和删除流表外, 不包含 OpenFlow 控制器需要实现的任何其他功能, 如链路发现、拓扑管理、设备管理、数据转发等 OpenFlow 核心功能. 由此需要 POF 控制器对进行扩展和完善.

为了完善和扩展 POF 控制器的功能, 我们基于 POF 与 Floodlight 实现了相关功能模块, 使其成为一个具有完整功能的能支持无协议感知转发的控制器. 我们实现的模块包括链路发现、拓扑管理、设备管理、数据转发等.

3 系统设计

POF 控制器是对于 Floodlight SDN 控制器的简化, 同时增加无协议感知功能. POF 控制器仅仅保留了对 OpenFlow 消息的接受发送功能, 删除了 Floodlight^[9]控制器处理 OpenFlow 消息的模块. POF 控制器通过 Floodlight 控制器的通信模块与交换机建立连接并通信, 同时在其上增加了 POF 模块和 GUI 模块, 通过 GUI 模块可以通过一步步手动配置流表以实现转发功能, 也可以通过 GUI 直接加载配置文件到交换机上. 可以使用 POFManager 提供的 API 开发其他应用. 图 1 是 POF 控制器的系统框图.

Controller 模块为 Floodlight 原有模块, 是实现与交换机通信的基础模块, 有交换机连接控制器时其通过发送 OpenFlow 消息与交换机建立 OpenFlow 连接, 建立连接后监听交换机发送的 OpenFlow 消息, 也能把通过交换机生成的 OpenFlow 消息发送给交换机. BypassManager 模块提供了一个 Controller 到 POFManager 的桥梁, 他通过检查 OpenFlow 消息的类型以调用 POFManager 中相应的处理函数. POFManager 则是 POF 的关键模块, 它

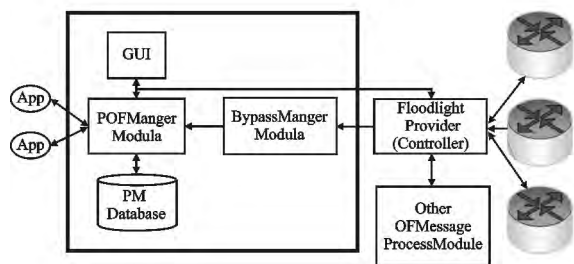


图 1 POF 控制器系统框图

提供接口供 BypassManager 和 GUI 模块调用, BypassManager 模块调用 POFManage 接口处理交换机发送的 OpenFlow 消息,把相应信息储存在数据库里同时调用 GUI 接口更新显示信息,另外也能处理出错信息;POFGUI 模块则能通过调用 POFManage 相应接口添加、修改、删除包类型、流表、流表项等。POFGUI 为用户提供了一个图形化的用户接口,通过 GUI 用户能添加、修改、删除包类型、流表、流表项等,它还包括信息展示功能,能够展示设备 ID、端口数、最大流表数目、流表大小、交换机型号等信息。原 POF 控制器中 OtherOFMessageProcessModule 里面没有任何实现代码,即除了 POFManageModule 外没有任何其他模块会处理 OFMessage。

正因为现有 POF 控制器仅是为了验证 POF 协议的可行性而开发的原型系统,控制器下发给交换机的所有流表都是手工通过 GUI 点击生成的,当仅具有几台交换机时还能使用这种方式进行流表的管理,而当交换机再增加时就不可能再使用这种方式进行管理。而且,对比 Floodlight 控制器,POF 控制器还缺乏很多支持控制器工作的基础功能和模块。由于缺少这些基础模块,对于陌生的未经网络管理人员通过 GUI 手工配置的流,在交换机发送给控制器后没有相应的处理逻辑产生流表以支持流的转发,而只能在用户通过 GUI 给定流表的时候才能实现转发功能,这样的 SDN 控制器并不是一个现实可用的控制器。所以我们就对 POF 控制器进行了扩展与二次开发,实现了这些基础模块以支持转发功能。开发过程中参考了现有 Floodlight、POX 等控制器的相关模块。

我们的工作是对 POF 控制器的功能扩展和完善,是使用 POF 提供的接口实现了一些功能模块,并未对其原有架构有较大改变。在原有 POF 控制器架构中,用户所添加的模块一般是以应用模块的形式呈现的,但我们添加的模块本身就属于正常控制

器应该具有的核心模块,所以将我们实现的模块放到 POF 控制器核心模块中。如图 2 所示。

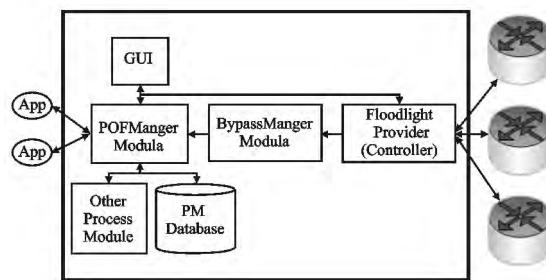


图 2 扩展的 POF 控制器框图

实现的模块包括以下模块:LinkDiscovery、TopologyManager 和 DeviceManager 三个核心模块和应用模块 Forwarding 模块这四个模块,加上原 POF 中的 OfSwitchManager 和 Controller 模块,实现 OpenFlow 控制器核心功能的所需的模块就已经全部具备。

4 主要模块设计

4.1 链路发现模块

LinkDiscoveryManager 模块即链路发现模块,它负责发现 OpenFlow 网络中的链路和维护链路的状态。

链路发现模块使用 LLDP 包和广播包(即 BBDPs)侦测链路,LLDP 的目的 MAC 地址是 01:80:c2:00:00:0e, BBDP 的目的 MAC 地址是 ff:ff:ff:ff:ff:ff。以太网帧协议类型分别是 0x88cc 和 0x8999。

如果拓扑中仅仅存在 OpenFlow 设备,仅仅使用 LLDP 即可正确发现链路。在存在的非 OpenFlow 交换机时,为了拓扑能被正确的发现,算法必须满足两个条件:(1)任意的交换机都(包括 OpenFlow 交换机)不会转发 LLDP 包,(2)交换机能够转发二层广播包。

传统的网络设备(交换机)对 LLDP 的处理方式:针对某个端口开启、不开启 LLDP 服务,即是否主动发送 LLDP 数据包,发送与接收一般无逻辑关系。如果传统设备已主动发送 LLDP 包,OpenFlow 交换机收到以后会上传到控制器,控制器可以判断出这个 LLDP 包是非 OpenFlow 设备发出的,但不能判定该非 OpenFlow 设备存在于 OpenFlow 区域中。因此不管传统设备对 LLDP 如何处理,都按统一的算法进行判断。

算法流程是控制器利用 OpenFlow 消息终端

Output 指令要求 OpenFlow 交换机发送 LLDP 包和 BDDP 包到所有的端口. 对于 LLDP 包:

(1)网络设备收到 LLDP 包后,支持 OpenFlow 协议的交换机会把 LLDP 包封装到 OpenFlow 的 PacketIn 消息中,并上传到控制器. 控制器解析 PacketIn 消息里面的 LLDP 包,判断其是否为 OpenFlow 设备产生的 LLDP 包,如果是则提取发送者的 DPID(datapath id),从 OpenFlow 消息中获取接收者的 DPID,这样控制器就发现了两个交换机间的一条链路. 通过两个 DPID 来确保唯一性.

(2)如果没收到 LLDP 包,链路对端的网络设备有可能是主机,也可能是非 OpenFlow 设备,需要通过 BDDP 包判断. 对于 BDDP 包作以下判断.

通过 LLDP 包可以判断出两个 OpenFlow 设备直连相连,因此若交换机 A 从一个网络 B 收到 BDDP 包而没有收到 LLDP 包,证明两个 OpenFlow 设备间有非 OpenFlow 的区域存在. 否则,该交换机 A 端口连接的是主机(即 A 在网络边缘),或者是内部包含的非 OpenFlow 区域.

对于 OpenFlow 交换机,如果收到 LLDP 包与 BDDP 包,则对端是 OpenFlow 交换机;仅收到 BDDP,说明对端设备是非 OpenFlow 设备(普通交换机). 都收不到,则对端是主机或是边缘非 OpenFlow 区域.

在系统中我们通过一个 4 元组来表示链路,其格式为 {SrcId, SrcPort, DstId, DstPort}, 其中 SrcId 为发送 LLDP 消息的交换机的 ID, SrcPort 为发送端口; DstId 为接收 LLDP 消息的交换机的 ID, DstPort 为接收端口. 由链路发现机制,同一条链路会被发现两次,在发现链路的时候会做一次查询,查看链路是否存在,若不存在保存,已经存在则不做处理.

4.2 拓扑管理模块

拓扑管理模块(TopologyManager)为控制器维护整个网络的拓扑信息,为网络中网络设备间的路由路径规划提供服务. 拓扑管理模块根据拓扑发现模块中获取到的链路信息计算拓扑结构. 这里一个重要概念就是 OpenFlow“孤岛”(island)的概念,“孤岛”是指一组强连接 OpenFlow 交换机,每一个“孤岛”即一个 OpenFlow 域,“孤岛”间可以使用普通 2 层交换机域相互连通. 像图 3 所示的拓扑:被划分成两个孤岛,孤岛 1 包含 OpenFlow 交换机 1 和 2,而孤岛 2 仅包含 OpenFlow 交换机 3.

每次链路发送变化时,将创建一个新的拓扑的

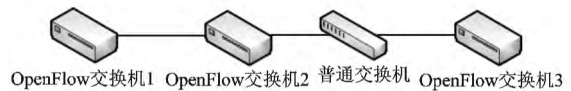


图 3 孤岛拓扑示例

实例,然后重新计算拓扑路径. 具体算法如下:

```
Compute(){
    identifyOpenflowDomains();
    addLinksToOpenflowDomains();
    calculateShortestPathTreeInClusters();
    calculateBroadcastTreeInClusters();
}
```

拓扑路径计算的算法流程如下所示.

(1)根据链路互联信息把交换机拓扑划分成多个 OpenFlow 域,每个域为一个“孤岛”. 使用 Tarjan 算法通过深度优先遍历的方式对域进行划分,每一个交换机属于一个域.

(2)在同一个域内的两个 OpenFlow 交换机如果有链路相连,则链路属于同一个簇,每个域对应一个簇. 遍历所有链路,如果链路两端的两个交换机在同一个域中,则将链路加入该簇,该方法统计了所有簇内的链路. 这么做是为了减少路由计算的开销,不然若干个大二层网络中共有 n 个节点,就需要 n 平方的存储开销,计算效率也会下降,将不相连的节点分开就减少了上述开销.

(3)遍历所有簇,对于每个簇内的每一个交换机使用 Dijkstra 算法计算簇内任意两个交换机间的最短路径. 每一个交换机节点在其域内生成一个相应的 BroadcastTree,具体实现中即每一个 switchDPID 对应一个 BroadcastTree. 则对于同一个域内的两个交换机节点,可根据目的交换机节点 BroadcastTree 计算得到路径.

4.3 设备管理模块

设备管理模块(DeviceManager)跟踪设备在网络中的移动并且能为目的设备定义新流. 设备管理器通过网络节点的请求数据包学习设备,它从包中提取信息并根据设置的节点分类器分类设备. 默认情况下节点分类器使用 Mac 地址和 Vlan 标识一个网络设备,这两个属性可以标识唯一的一个设备. 设备管理模块也会学习其他的属性,例如设备的 IP 地址. 如果 OpenFlow 交换机接收到一个相连设备的数据包,继而产生一个 PacketIn 消息发送给控制器,控制器就会为这个设备创建一个连接点,设备连接点是包括设备所连接的交换机 DPID 和连接的端

口. 一个设备在每个 OpenFlow 孤岛上可以有一个或者多个连接点. 设备自身、连接点及 IP 等都会设置失效时间, 每个设备及其连接点有一个最近活跃的时间戳, 过了失效时间控制器就会删除连接点及设备信息.

连接点的数据结构如图 4 所示.

mac	ip	vlan	DPID	port	timeStamp
-----	----	------	------	------	-----------

图 2 连接点数据结构

DPID 和 port 分别表示与设备连接的 OpenFlow 交换机的 ID 和与其相连接端口的端口号. 如果需要向设备发送数据, 即可先发送到 DPID 指明的交换机上, 交换机再通过 port 通过相应端口转发到设备上. MAC(Vlan)地址和 IP 地址用来支持二层和三层的转发模块为控制器维护整个网络的拓扑信息, 为网络中网络设备间的路由路径规划提供服务.

4.4 数据转发模块

数据转发模块(Forwarding)能够在两个设备间转发数据包.

在 OpenFlow 中, 一个新的流到达交换机收到时, 会将数据包发送给控制器. 转发模块解析数据包的格式, 如果是 IP 包, 则提取其目的 IP 地址. 根据目的 IP 地址遍历查找与目的 IP 地址相同连接点, 找到则可获取到目的 switchID 和 switchPort, 没有找到则泛洪(或丢弃). 根据源 switchID(即上传流的交换机的 ID)和获取到的目的 switchID, 再由拓扑管理模块的生成的目的 switchID 的 Broadcast-Tree, 可得到源交换机到目的交换机的一条路径. 为路径上的每一跳交换机配置流表, 即可完成转发.

5 结束语

本文基于开源 SDN 控制器, 增加了链路发现、拓扑管理、设备管理、数据转发等核心模块, 实现了一个自动支持数据转发的真正可用的 POF 控制器,

为今后面向 POF 协议的研究与开发提供了一个基础实验平台. 由于 POF 协议标准尚处于完善阶段, 该控制器仍有改进空间, 下一步的工作包括增加负载均衡、流量规划等功能以优化转发路径.

参考文献:

- [1] 周健, 章倩. 网络安全态势量化模型研究[J]. 微电子学与计算机, 2014, 31(11): 32-35.
- [2] 梁海军. 人工蜂群优化支持向量机算法在网络安全中的应用[J]. 微电子学与计算机, 2013, 30(2): 95-98.
- [3] 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.
- [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] Open Networking Foundation. OpenFlow switch specification v1. 0. 0. [EB/OL]. (2014-01-10). <http://www.opennetworking.org/>.
- [6] Open Networking Foundation. OpenFlow switch specification v1. 3. 0. [EB/OL]. (2014-01-10). <http://www.opennetworking.org/>.
- [7] Song H. Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane [C] // Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. China, Kongkong, ACM, 2013: 127-132.
- [8] Huawei Technologies. Protocol oblivious forwarding. [EB/OL]. (2014-01-10). <http://www.poforwarding.org>.
- [9] Big switch networks floodlight[EB/OL]. (2014-01-10). <http://floodlight.openflowhub.org/>.

作者简介:

高翔宇 男, (1991-), 硕士研究生. 研究方向为未来网络、SDN. E-mail: renxianggao@163.com.