面向协议无感知转发技术的 SDN 试验床①

谈小冬, 邹 山, 郭浩然, 田 野

(中国科学技术大学 计算机科学与技术学院, 合肥 230027)

摘 要: 基于 OpenFlow 的软件定义网络(SDN)中的转发设备不便于支持新协议的转发, 因此协议无感知转发(Protocol Oblivious Forwarding, POF)技术被提出. 本文基于 POF 技术和 OpenFlow 控制器 POX, 设计并实现了一种支持 POF 技术的 SDN 控制器, POF 控制器可以充分利用 POF 转发设备并且体现了 SDN 的可编程性. 同时, 我们基于 POF 控制器搭建了 POF 试验床, 实验结果表明 POF 控制器能够有效地管理 POF 网络, 并提供高效的控制功能.

关键词: 软件定义网络; OpenFlow; 协议无感知转发; 试验床

SDN Testbed for Protocol Oblivious Forwarding

TAN Xiao-Dong, ZOU Shan, GUO Hao-Ran, TIAN Ye

(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

Abstract: The OpenFlow-based software-defined Networking could not easily support forwarding of the new protocols, thus Protocol-Oblivious Forwarding technology is proposed. On the basis of POF technology and OpenFlow controller POX, this paper designs and implements the controller which supports POF technology, employs the full potentials of POF devices and reflects the programmability of SDN. Meanwhile, we construct a network testbed based on POF controller, and experimentally illustrate that POF controller can effectively manage the POF network and provide the controlling functionality with high performances.

Key words: software-defined networking; OpenFlow; protocol oblivious forwarding; testbed

软件定义网络(Software Defined Networking, SDN)是将控制平面与转发平面分离并直接可编程的网络架构[1], 它的思路是: 将传统网络设备的数据转发和路由控制两个功能模块相分离, 通过集中式的控制器以标准化的接口对各种网络设备进行管理和配置.

OpenFlow 是由斯坦福大学提出的可用于 SDN 中的核心技术[2], 它已逐渐成为 SDN 中的控制器与网络设备通信协议的标准. 然而, 基于 OpenFlow 的 SDN 只能支持现有的数据报文协议转发, 不能支持新的协议, 若需要支持新协议, 设备制造商需要修改设备代码. 所以为了支持更多的新协议, OpenFlow 协议变得越来越复杂.

开放网路基金会(Open Networking Foundation, ONF)在中提出了协议独立转发层和协议独立转发(Protocol Independent Forwarding, PIF)[3], 并在白皮书中提出使用 P4[4]和协议无感知转发技术(Protocol-Oblivious Forwarding, POF)[5]两种方案互补的方式来解决协议独立转发的问题. POF 技术定义了底层的原始指令集, P4 需要来编写和编译使用 POF 指令集的数据包处理程序.

POF 技术是由华为公司提出的 SDN 转发平面的一项创新技术, 即转发硬件设备对数据报文协议和处理转发流程没有感知, 网络行为完全由控制面负责定义. POF 技术可以作为对 ONF OpenFlow 协议的增强, 支持任意转发协议和分组数据格式, 使 SDN 的控制平面和转发平面分离更加彻底. 目前, 华为公司只发布

① 基金项目:国家自然科学基金(61202405);中国科学院先导专项子课题(XDA06011202)
收稿时间:2015-07-30;收到修改稿时间:2015-10-09

了支持 POF 技术的交换机和控制器的 Linux 版本代码[6], 但是华为提供的 POF 控制器只能手工配置网络设备的流表, 不支持编程模型, 而 SDN 最大的特色就是可编程性. 所以本文中, 我们在 POX 控制器[7]的基础上, 提出了一个新的支持 POF 技术的 SDN 控制器. 同时, 我们利用新的 POF 控制器和 POF 软交换机搭建了第一个 POF 试验床, 并进行了相关的功能和性能测试.

本文余下的内容安排为: 第 2 节介绍我们开发 POF 控制器的工作; 第 3 节展示了在 POF 试验床上的相应测试工作; 第 4 节总结我们的工作, 并说明后续工作的安排.

1 POF控制器

1.1 基于 POF 的 SDN 架构

基于 POF 的 SDN 架构如图 1 所示, 类似于基于 OpenFlow 的 SDN 机构. POF 控制器中包括通信引擎、拓扑发现、最小生成树、MAC 地址学习等模块, 控制器通过扩展的 OpenFlow 协议下发流表来控制 POF 交换机的转发行为. 在设计 POF 控制器时, 我们借助了 POX 控制器的编程模型和基础结构. 同时, 我们利用 POF 控制器和 POF 软交换机搭建了第一个 POF 试验床并进行测试.

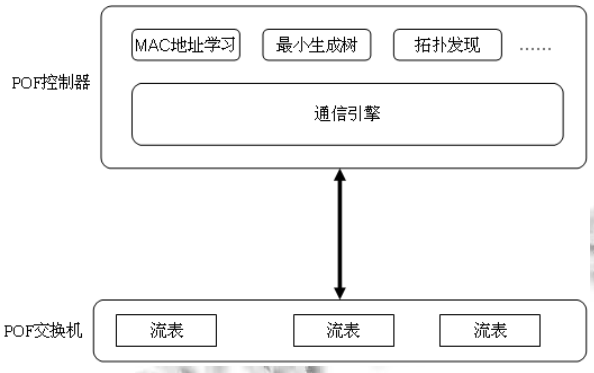

图 1　基于 POF 的 SDN 架构

1.2 通信引擎

POF 控制器与 POF 交换机通信主要是通过消息的交互, 这些消息可以是控制器与交换机建立连接、向 POF 交换机下发流表、改变 POF 交换机的状态等. 通信引擎模块主要功能是实现这些消息的数据结构、封装收发和解析处理, 是整个控制器框架的基本模块, 也是我们设计过程中的关键之处. 我们根据 POF 技术设计和定义基本的消息类型, 消息的解析过程以及异常处理过程.

1.3 拓扑发现

拓扑发现模块实现了在POF网络中发现POF交换机间(本文假设 POF 网络中所有的网络设备都支持 POF 技术)的网络拓扑信息的功能, 主要使用了 LLDP(Link Layer Discovery Protocol, 链路层发现协议)[8]作为链路发现协议. LLDP 协议是一种邻居发现协议, 它提供了一种标准的链路发现方式, 可以将本端设备的主要能力、管理地址、设备标识、接口标识等信息表示成不同的 TLV(Type/Length/Value, 类型/长度/值)形式, 并封装在 LLDPDU(链路层发现协议数据单元)中, 封装有 LLDPDU 的报文称为 LLDP 报文, 报文中包含了特定的组播、目的 MAC 地址, 以及特定的以太网类型, 可以将 LLDP 数据包与其他 MAC 数据帧区分. POF 控制器进行链路发现的过程如图 2 所示.

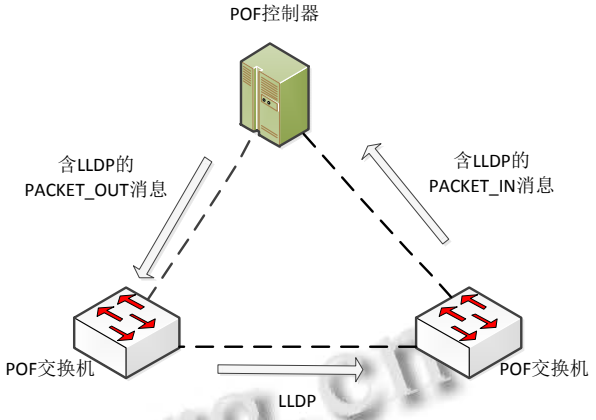

图 2　POF 交换机间的链路发现示意图

当 POF 交换机连接 POF 控制器时, POF 控制器会发送一个包含 LLDP 数据包的 PACKET_OUT 的消息给该交换机, 该消息使 POF 交换机通过所有自身端口发送 LLDP 数据包. 当 POF 交换机收到邻居 POF 交换机发送的 LLDP 数据包时, 由于 POF 交换机初始时并没有专门的流表项用于处理 LLDP 数据包, 所以它会通过一个 PACKET_IN 消息将 LLDP 数据包发送给 POF 控制器. POF 控制器在收到 PACKET_IN 消息时, 通过解析消息和 LLDP 数据从而发现两台 POF 交换机间的链路信息. 网络中其他 POF 交换机也都采用相同的方式发现与邻居 POF 交换机间的链路, 因此 POF 控制器通过拓扑发现模块能够发现完整的网路拓扑图.

1.4 生成树协议

如果基于 POF 的 SDN 网络拓扑中存在回路时可能会产生网络风暴, 最小生成树模块能有效地防止了网络中回路的出现, 避免了由于帧的无限循环和重复接收所导致网络风暴的发生. 该模块的实现需要拓扑发现模块探测的网络拓扑信息, 实现算法如下:

算法 1: 基于 POF 的 SDN 中的生成树算法

Input: a dictionary *tree_dict* about topology information and a set *switches* contains all of POFSwitches get by Topology Discovery Module//由拓扑发现模块得到的保存拓扑信息的字典和保存 POF 交换机的集合

Define: POFSwitches have visited(*done* = ()),port in spanning_tree(*tree_port* = ())

Steps:

```
1.  for S1 in switches do
2.      if S1 not in done then
3.          done.add(S1)
4.      else//若 POFSwitch 已经遍历, 则遍历下一个
5.          Continue
6.      end if
7.      for S2 in switches do
8.          if there is a link between S1 and S2 in tree_dict then//若 S1、S2 间存在链路
9.              done.add(S2)
10.             tree_port.add(S1.port, S2.port)//S1.port1, S2.port2 是连接 S1、S2 的链路上的两端端口
11.         end if
12.     end for
13.   end for
14. for S in switches do
15.   for port in S do
16.     if port in tree_port then//若端口在生成树里
17.         S.port.is_flood = True//开启端口的洪泛功能
18.     else
19.         S.port.is_flood = FALSE
20.     end if
21.   end for
22. end for
```

由拓扑发现模块得到所有连接 POF 控制器 POF 交换机的集合 *switches* 和 POF 交换机间的链路关系 *tree_dict*, 并用 *done* 保存已遍历的 POF 交换机, 用 *tree_port* 保存生成树里的端口. 按序遍历 *switches* 里的每个 POF 交换机, 将该交换机保存到 *done* 中, 若 *tree_dict* 中存在该 POF 交换机与另一 POF 交换机间的链路, 则将另一交换机和存在的链路两端端口加入到 *tree_port* 中. 最后, *done* 中已包含所有 POF 交换机时, 将 *tree_port* 中对应的端口开启洪泛功能.

1.5 MAC 地址学习

MAC 地址学习模块是通过维持 MAC 地址表来保存 MAC 地址和端口的映射关系, 实现 MAC 地址学习功能. 如果网络拓扑中存在回路, 则需要启动最小生成树模块打破环路, 避免广播风暴. MAC 地址学习模块的实现算法如下:

算法 2: 基于 POF 的 SDN 中 MAC 地址学习算法

Input: a Ethernet packet which has fields of *dst_mac,src_mac* and *type_mac*

Define: MAC address table(*mac_dict* = {})//MAC 地址表, 保存 MAC 地址和端口的对应关系

Steps:

```
1.  when POFSwitch receives a Ethernet packet from port p do
2.  mac_dict[src_mac] = p //更新 MAC 地址表
3.  if type_mac is LLDP or dst_mac is Bridge Filtered address then
4.      drop the packet
5.  elif dst_mac is multicast then//目的 MAC 地址是广播地址则洪泛数据包
6.      flood the packet
7.  else
8.      if dst_mac not in mac_dict then//目的 MAC 地址不在 MAC 地址表内则洪泛数据包
9.          flood the packet
10.     else
11.         if mac_dict[dst_mac] == p then//输入端口和输出端口是同一端口则丢弃数据包
12.             drop the packet
13.         else
14.             forward the packet to mac_dict[dst_mac] and install a flow entry//转发并安装流表
15.         end if
16.     end if
17. end if
```

当 POF 交换机收到数据包时，首先根据输入端口 *p* 和源 MAC 地址 *src_mac* 更新 MAC 地址表 *mac_dict*，如果数据包是 LLDP 类型或目的 MAC 地址 *dst_mac* 是 Bridge Filtered 地址，则丢弃数据包；如果 *dst_mac* 是广播地址或者 *mac_dict* 中没有 *dst_mac* 对应的键值(即输出端口)时，则广播该数据包；如果 *mac_dict* 中有 *dst_mac* 对应的输出端口是 *p*，则丢弃数据包，否则将收到数据包转发到合适端口并安装相应的转发流表.

2　POF试验床

基于我们实现的 POF 控制器和 POF 交换机，我们搭建了第一个 POF 试验床，如图 3 所示. 其中，8 台 POF 交换机通过以太网交换机连接，POF 控制器也与以太网交换机连接. POF 控制器运行在一个配置为酷睿 i7, 1T 硬盘和 1G 内存的主机上, POF 交换机运行在 10 个千兆网口的主机上，如图 4 所示. 同时，我们也对我们的试验床进行了相应测试.

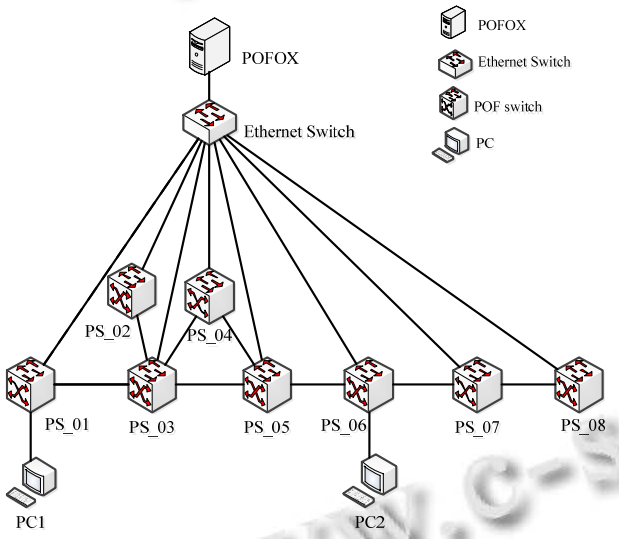

图 3　POF 试验床

图 4　安装 POF 交换机的 10 网口主机

2.1 性能分析

吞吐量和时延是网络性能的重要衡量指标. 这里，我们通过对 POF 交换机进行了简单的修改，以统计 POF 控制器每秒中能够下发流表的数量. 经过我们测量统计，我们得到了 POF 控制器下发流表的速率和 POF 交换机建立一个流表的平均时间，并与 POX 控制器进行了比较，如表 1 所示. 由结果可知, POF 控制器的性能是可观的，基本达到了与 POX 控制器一致的性能.

表 1　流表下发速率与建立流表的平均时间

	POF	POX
POF 控制器流表下发速率(flows / s)	33000	31000
POF 交换机建立流表的时间 (ms)	0.050	0.050

2.2 拓扑发现

为了测试 POF 试验床的功能，我们运行拓扑发现模块来展示. 这里，我们借用第三方工具 poxdesk[9]来显示我们的测试结果，如图 5 所示. 图中每个节点代表一个 POF 交换机，与图 3 对比可知，我们通过 POF 控制器的拓扑发现的模块得到的网络拓扑与我们试验床的拓扑一致.

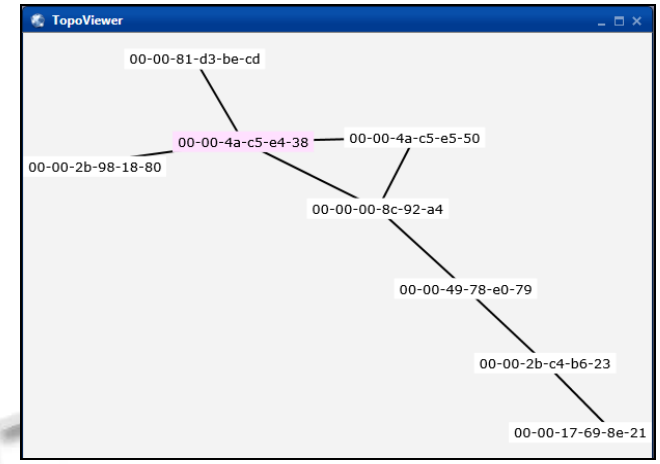

图 5　网络拓扑图

2.3 生成树协议

POF 技术对转发的数据包协议无感知，能支持任何协议的数据包转发. 我们通过工具 Ostinato[10]组建并发送自定义数据包来测试试验床的性能. 我们设置自定义数据包的类型为 0x0988，TTL 字段值为 10. 同时，在 POF 控制器端运行 MAC 地址学习模块. 当数据包到达 POF 交换机时，交换机在转发前会通过相应流表操作将 TTL 值会减 1，当 TTL 值为 0 或者数据包到达目的交换机时数据包将不会转发. 我们在 PC_1 向 PC_2 连续发送 50 个自定义数据包，并在相应链路上的 POF 交换机用 wireshark 抓包分析.

在表 2 中, 我们统计比较是否运行生成树协议模块这两种情景下 POF 交换机上经过的数据包数量: 当没有运行生成树协议模块时, POF 交换机 PS_04 和 PS_05 上转发的数据包数量是运行生成树协议模块时的 6 倍, PS_01、PS_03 和 PS_06 上是运行生成树协议模块的 5 倍. 因此, 该模块可有效减少冗余传输, 有效地提高网络的稳定性.

表 2 POF 交换机转发数据包的数量

POF 交换机	运行生成树协议	未运行生成树协议
PS_01	50	250
PS_03	50	250
PS_04	50	300
PS_05	50	300
PS_06	50	250

3 总结

本文基于 POF 技术设计并实现了 POF 控制器, 主要包括通信引擎、拓扑发现、最小生成树、MAC 地址学习等模块. 在此基础上, 我们利用新的 POF 控制器和 POF 交换机搭建了基于 POF 的 SDN 试验床, 证实了我们的 POF 控制器具有良好的特性和一定的灵活性和扩展性. 我们的工作为 POF 技术的研究提供了很好地思路, 极大的体现了 SDN 的可编程性这一特点. 但是, POF 控制器的功能和性能亟待丰富提升, 下一步的工作我们将修改 Mininet 平台使其支持 POF 技术, 并进行不同规模的仿真实验. 同时我们将考虑 POF 控制器的并行化[11]以及性能提升, 以及将 POF 软交换机移植到 OpenWRT[12]系统中以便于支持无线接入技术.

参考文献

1 Casado M, Freedman MJ, Pettit J, Luo J, McKeown N, Shenker S. Ethane: taking control of the enterprise. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 1–12.

2 McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69–74.

3 PIF. https://www.opennetworking.org/protocol-independent-forwarding.

4 Bossharty P, Daly D, Gibb G, et al. P4: Programming protocol-independent packet processors, ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87–95.

5 Song H. Protocol-Oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane. ACM SIGCOMM Workshop on HotSDN. 2013. 127–132.

6 Huawei's POF Controller and switch. http://www.poforwarding.org/vdownload/.

7 POX. http://www.noxrepo.org/pox/about-pox/.

8 LLDP. http://en.wikipedia.org/wiki/Link_Layer_Discovery_Protocol.

9 Poxdesk. https://github.com/MurphyMc/poxdesk/wiki/Getting-Started.

10 Ostinato. https://code.google.com/p/ostinato/wiki/UserGuide

11 Cai Z, Cox AL, Ng TSE. Maestro: A system for scalable OpenFlow control[Technical Report], TR10-08, Rice University, 2010.

12 OpenWRT. http://www.openwrt.org.cn.

.