SDN 控制器研究与微控制器开发

学生姓名：李呈
指导老师：黄韬　　职称：副教授

摘　要

本论文通过对 POX，FloodLight 等多种控制器的研究和总结，提出了微控制器模型。微控制器是控制器功能最小功能集，包含基础功能层和网络基础服务层，能实现控制器的最基本功能。我们还设计研发了微控制器实例 Miracle，并使用 Cbench 对 Miracle 进行性能测试。在 Cbench 延时模式中，单交换机的测试结果为 287.85Responses/s，而 POX 的测试结果为 882.15 Responses/s，可知 Miracle 的性能仅仅为 POX 的 1/3。从测试结果看 Miracle 性能不够理想，还需要在数据处理流程上进一步优化。

关键字　软件定义网络　OpenFlow　控制器　微控制器

ABSTRACT

In this paper, inspired by the design philosophy of these classical SDN controllers we proposed a Micro-controller model which includes basic functions layer and network infrastructure services layer. We design and develop Miracle as an instance of Micro-controller model. Finally, we also use Cbench to evaluate Miracle's performance. Miracle can handle 287.85 new flows per second in single switch topology using latency model of Cbench, while POX can handle 882.15 new flows per second under the same condition. As the performance of Miracle is just 1/3 of POX, we can know that Miracle is not good enough. So we need to improve process of packets processing that make Miracle better.

KEY WORDS　software defined networking　openflow　controller　micro-controller

第一章 SDN(Software Defined Networking)与 OpenFlow

1.1 SDN 定义

SDN（Software Defined Networking，软件定义网络）[1]是一种新型的基于软件可编程思想的网络架构，它有一个集中式的控制平面和分布式的转发平面（转发平面没有控制逻辑，只专注于转发功能的实现），实现了控制平面与转发平面的分离，可以做到集中化的控制，并且提供开放的编程接口，为网络提供灵活的可编程能力，具备以上特点的网络架构都可以被认为是一种广义的 SDN。

1.2 OpenFlow

Ethane 项目于 2006 年开始部署，致力于企业网架构的创新，OpenFlow[2]协议的雏形就诞生于这个项目。2008 年，Nick McKeown 教授的一篇重要论文“OpenFlow: Enabling Innovation in Campus Networks”使得 OpenFlow 开始正式进入人们的视野，继而成为了标准化组织 ONF 主推的南向接口协议。经过多年的发展，OpenFlow 目前已成为 SDN 的主流技术之一。

第二章 SDN 控制器架构研究

在 SDN 数据平面与控制平面相分离的体系架构中，控制器作为整个网络的操作系统具有举足轻重的地位，它是连接底层交换设备与上层应用的桥梁。一方面，控制器通过南向接口协议对底层网络交换设备进行集中管理、状态监测、转发决策以处理调度数据平面的流量；另一方面，控制器通过北向接口向上层应用开放出多个层次的可编程能力，允许其根据特定的应用场景灵活地制定网络策略。经过对 POX[3]，FloodLight[4]，OpenDaylight[5]，RYU[6]和 OpenContrial[7]等多款控制器的研究，我们总结出层次化控制器架构如图 2-1。

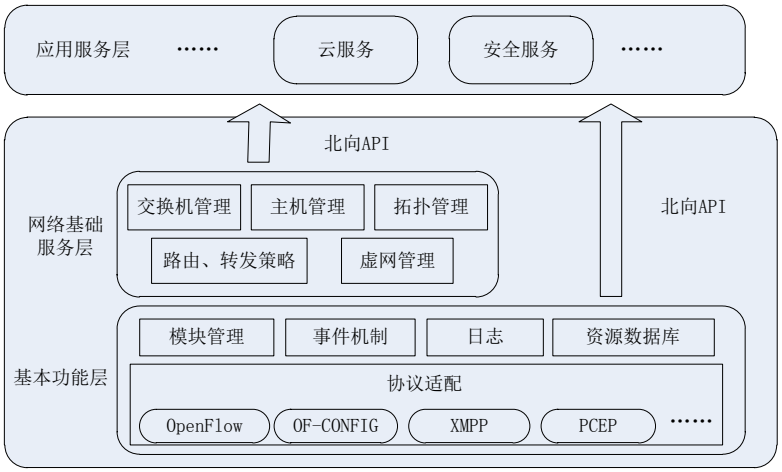

图 2-1 控制器层次化体系架构

由图 2-1 可知层次化控制器架构可分为基本功能层和网络基础服务层。其中基本功能层可包含协议支持，协议适配，资源管理等多个功能模块；网络基础服务层可包含转发，路由决策，拓扑管理功能等多个功能模块。

通过抽象总结，我们提出微控制器的概念。微控制器是层次化结构、最小功能集的控制器。与其他控制器一样，微控制器可分为基础功能层和网络基础服务层。其中，微控制器的基础功能层有 OpenFlow1.0 协议接口功能；网络服务功能仅有二层转发功能等基础网络功能，也称核心网络应用。我们将在下一章详细介绍微控制器的实现实例，并将微控制器实例命名 Miracle。

第三章　微控制器设计与实现

本章介绍的微控制器 Miracle 是在研究多种控制器的架构、功能之后，设计研发的一个最简单的控制器。Miracle 实现了 OpenFlow 协议解析、ARP 代理和二层交换等功能，并具有可编程拓展能力。本章将对 Miracle 主要功能进行简要介绍，并讲解 Miracle 的具体实现，主要包括开发环境准备、代码实现和分析。

Miracle 微控制器采用 Python 语言编写，采用基于 Tornado[8]的底层通信，利用开源软件 Scapy[9]进行协议数据解封装。其主要模块有底层网络通信模块、OpenFlow 协议处理模块、ARP 代理和二层交换模块。Miracle 模块图如 4-1 所示。

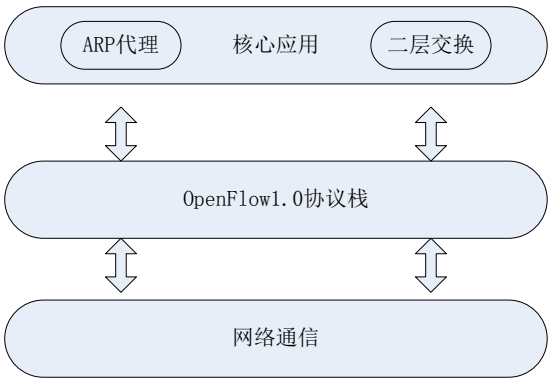

图 3-1　Miracle 模块图

3.1　环境准备

（1）Tornado 安装

Tornado 是 FriendFeed 使用的可扩展、非阻塞式 web 服务器及其相关工具的开源版本。Tornado 每秒可以处理数以千计的连接，对于实时 Web 服务来说，Tornado 是一个理想的 Web 框架。所以我们选择 Tornado 搭建控制器的网络通信模块。具体安装由于篇幅原因此处不进行介绍。

（2）Scapy 安装

Scapy 是一个功能强大的交互式数据包处理程序，可用来发送、嗅探、解析和伪造网络数据包，常常被用到网络攻击和网络测试中。所以我们选择 Scapy 进行数据的封装。由于篇幅原因，安装细节不再介绍。Scapy 封装了许多函数，可用于数据结构的封装，这大大方便了 Miracle 的开发。

3.2　网络通信

网络通信模块基于 Tornado 架构搭建，主要实现控制器的底层数据通信的收发功能，具体包括创建控制器监听套接字，建立交换机的连接并管理 io_loop 的状态。控制器接收数据之后，调用对应协议解析函数对报文进行处理，并向交换机发送数据。在搭建网络通信底层时，我们使用到了 Tornado 的 io_loop 类。通过 io_loop 的状态变化，可以决定读写数据的操作，从而完成底层的数据收发。

3.3　协议解析

协议解析部分主要从 OpenFlow1.0 数据结构定义和 OpenFlow1.0 协议解析两部分进行介绍。数据结构定义部分主要介绍如何使用开源软件 Scapy 封装

OpenFlow 报文的数据结构，而 OpenFlow1.0 协议解析部分将分为数据解析和数据封装两部分讲解。

3.3.1 OpenFlow1.0 数据结构定义

新建 libopenflow.py 文件用于定义数据结构。然后根据 OpenFlow 协议规定的数据结构类型，使用 Scapy 进行封装即可。如 ofp_header 报文有 version，length，type 和 xid 四个关键的字段。我们可以使用 scapy 的对应函数对这些不同长度或者不同类型的数值进行赋值，如 BitField 函数可用于封装指定若干 bit 长度的数据。依次类推将所有的 OpenFlow 协议报文的数据结构封装完成即可。

3.3.2 OpenFlow1.0 协议解析

OpenFlow1.0 协议的处理函数主要负责报文的处理，可分为数据解析和报文封装两部分。数据解析部分主要用于解析接受的交换机信息，如 OFPT_PACKET_IN 报文等。而报文封装部分主要是用于完成控制器下发给交换机的消息的封装，如 OFPT_PACKET_OUT。

在数据解析过程中，我们将收取的字节流数据的首 8 字节截取，并解析为 OFP_HEADER 报文，并通过判断 Type 字段决定调用何种解析函数对报文进行解析。如 OFPT_FEATURES_REPLY 报文的解析函数需将收到的 features 数据解析成 switch_features 和 ofp_phy_port 数据。

在报文封装过程中，需要将对应的信息封装成指定的报文。如根据 OFPT_PACKET_IN 报文的信息，封装 OFPT_PACKET_OUT 报文信息，OFPT_PACKET_OUT 报文数据结构如图 3-2。其中 in_port 等重要需从 OFPT_PACKET_IN 报文获取。依次类推可完成其他报文的封装。

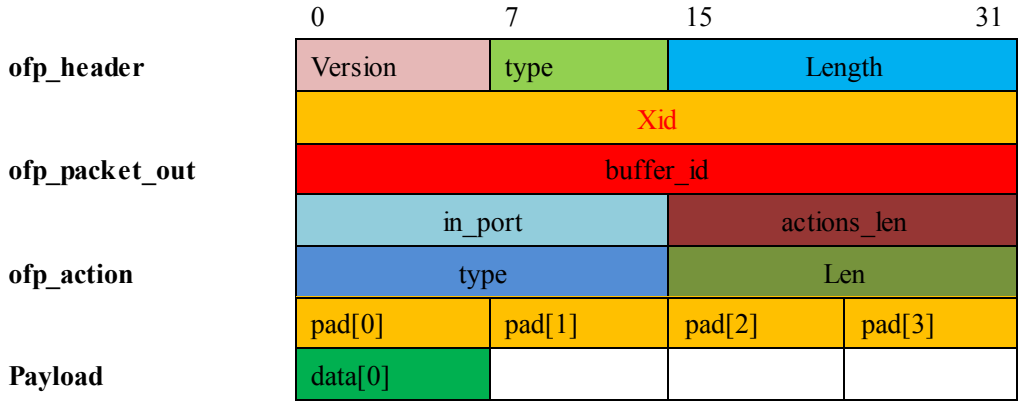

图 3-2 ofp_packet_out 数据结构

3.4 核心应用

Miracle 目前实现的核心应用有 ARP 代理服务和二层交换。

3.4.1 ARP 代理

在传统网络中，主机发送 ARP 请求，将会在局域网进行广播。过多的 ARP 广播数据将占据过多带宽进而影响网络效率。在 SDN 网络中，通过控制器记录主机 IP/MAC 信息，用于回复 ARP 请求，这样能避免大量 ARP 请求在局域网内广播，从而减少网络负载。因此 ARP 代理服务成为许多控制器的功能之一，Miracle 也实现 ARP 代理服务功能。其主要的工作流程为： ARP 代理应用通过学习、记录主机 ARP 信息，并通过查询 ARP 表响应 ARP 请求。

3.4.2 二层交换

二层交换是网络通信中重要的功能，在传统网络架构的局域网内，数据交换往往是由二层交换机完成。首先交换机记录数据包的源 MAC 地址和入端口的对应关系，然后判断目的 MAC 地址是否为广播地址，若是，则将数据包泛洪，否则以目的 MAC 地址为键，查询出端口，若查找成功，则将数据转发到出端口，否则把数据包泛洪。在 Miracle 中处理 ARP 请求类型的广播包采用了 ARP 代理解决方案，使二层交换应用得到了优化。

第四章 微控制器测试与评估

本章将对微控制器 Miracle 进行测试与评估。测试工具采用专门测试控制器的 Cbench[10]。单纯测试 Miracle 的意义不大，我们将在同等条件下对 POX 进行测试。通过对比两者的性能，得出 Miracle 的性能数据。

在 Cbench[11]的延时模式中，对单交换机拓扑进行测试，POX 的测试数据为平均 882.15 Responses/s，而 Miracle 则是 287.85Responses/s。对比之下，Miracle 为 POX 的 0.326，换言之，Miracle 的单交换机性能仅为 POX 的 1/3。

4.1 测试评估

从测试结果看，单交换机对比之下，Miracle 性能欠佳。作为一个控制器，Miracle 的性能不够理想，但是作为一个微控制器的初步尝试，Miracel 的性能是可以接受的。Miracle 性能不理想的原因可能有：1、在 Miracle 的实现过程中，仅注重功能实现缺乏性能考虑。2、Miracle 为单线程，导致在处理数据时，不能

进行下一个数据的读取，且没有完成数据处理时，始终无法发送数据。

由此可知，微控制器 Miracle 还有很多优化工作需要完成。主要有：1、引入订阅分发的事件系统，由事件驱动上层应用，从而解除底层协议解析与上层应用的耦合关系，提高 Miracle 的编程可拓展性。2、使用多线程完成数据收发和处理，充分利用数据处理时间完成输入输出工作，提高整体的处理效率。

第五章　总结

本篇论文首先介绍了 SDN 和 OpenFlow 的背景知识；然后对 SDN 控制器架构进行了深入的研究，提出微控制器的概念；在提出微控制器概念之后，介绍微控制器实例 Miracle 的实现过程。最后还使用 cbench 对 Miracle 进行测试，并与 POX 进行对比，得出 Miracle 的性能评估结论。在完成 Miracle 基础之上，我们将继续优化微控制器 Miracle。

参考文献

[1] OPEN NETWORK FOUNDATION. Software-Defined Networking: The New Norm for Networks. ONF White Paper. April 13, 2012

[2] Nick McKeown, Tom Anderson, Hari Balakrishnan. OpenFlow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer Communication Review. Volume 38, Number 2, April 2008

[3] “Nox”,[Online] Available:http://www.noxrepo.org/ (2014/5/21)

[4] ”FloodLight”,[Online] Available:http://www.projectfloodlight.org (2014/5/21)

[5] ”Opendaylight”,[Online] Available:http://www.opendaylight.org/ (2014/5/21)

[6] ”OpenContrail”,[Online] Available:http://opencontrail.org/about/ (2014/5/21)

[7] ”Ryu”,[Online] Available:http://osrg.github.io/ryu (2014/5/21)

[8] “Tornado”,[Online] Available: http://www.tornadoweb.org (2014/5/21)

[9] ”Scapy”, [Online] Available: http://www.secdev.org/projects/scapy/ (2014/5/21)

[10] Guillermo Romero de Tejada Muntaner. Evaluation of OpenFlow Controllers. October 15, 2012

[11] Ogden, J. Cbench: A Software Toolkit for Testing, Benchmarking, and Qualifying HPTC Linux Clusters. Whitepaper. Sandia National Laboratory. http://sourceforge.net/projects/cbench-sf