计算机网络实验期末考查实验报告

学院（系）名称：信息学院

姓名	高星杰	**学号**	2021307220712	**专业**	计算机科学与技术
班级	21 级 2 班	**实验项目**	SDN 实际应用实践		
课程名称		计算机网络实验		**课程代码**	3103009336
实验时间		2023 年 12 月 21 日		**实验地点**	逸夫楼 C207

考核内容	目的和原理 40	内容及过程分析 30	实验方案设计 10	实验结果（结论正确性以及分析合理性）20	**成绩**	
各项得分						
考核标准	○原理明确 ○原理较明确 ○原理不明确	○分析清晰 ○分析较清晰 ○分析不清晰	○设计可行 ○设计基本可行 ○设计不可行	○结论正确，分析合理 ○结论正确，分析不充分 ○结论不正确，分析不合理	**教师签字：**	

1. 实验目的：

1.1 了解 SDN 的实现原理
1.2 掌握基本 SDN 的配置
1.3 掌握 RYU 的使用
1.4 掌握 Mininet 的使用
1.5 使用 SDN 与 RYU 设计一个有实际意义网络**（本次实验决定设计一个具有防火墙、实现 STP、实现 DHCP 分配 IP 的、能够使用 OSPF 路由的、可以实时检测状态的、支持 SDN 的网络）**

2. 实验创新点

由于 SDN 的实验不像之前的实验一样发挥空间很小，而本次实验我们可以尝试使用一些新的有趣的东西，所以为了更好更直观的介绍本次实验，新增了本节。

本次实验有一些我认为对于我来说有一些比较新的应用：

① 基于 RYU 控制器和 OpenFlow 下发流表（类似之前实验的 ACL）实现防火墙功能
② 基于 sFlow 实现实时的可视化的网络监视
③ 考虑到实际情况使用环状网络，并且实现 STP
④ 使用 DHCP 动态分配 IP
⑤ 实现动态路由 OSPF 的配置

3. 实验原理：

我们要彻底了解一个技术就要了解他的来龙去脉。在之前的实验中我们使用的网络设备是华为路由器与

交换机，我们需要学习该类型的网络设备的指令，去实现我们想要他实现的功能，我们是与华为的网络设备的操作系统打交道的，但是出现一个问题，如果我们用其他厂商的网络设备，与其他厂商的操作系统打交道，那么就又要学习新的一套指令的，如果一个巨大的网络有多种的网络设备，那么我们需要记住的指令要有上千条，所以 SDN 就针对这个问题应运而生了。

3.1 SDN 的产生背景是什么？（SDN 解决了什么问题？）

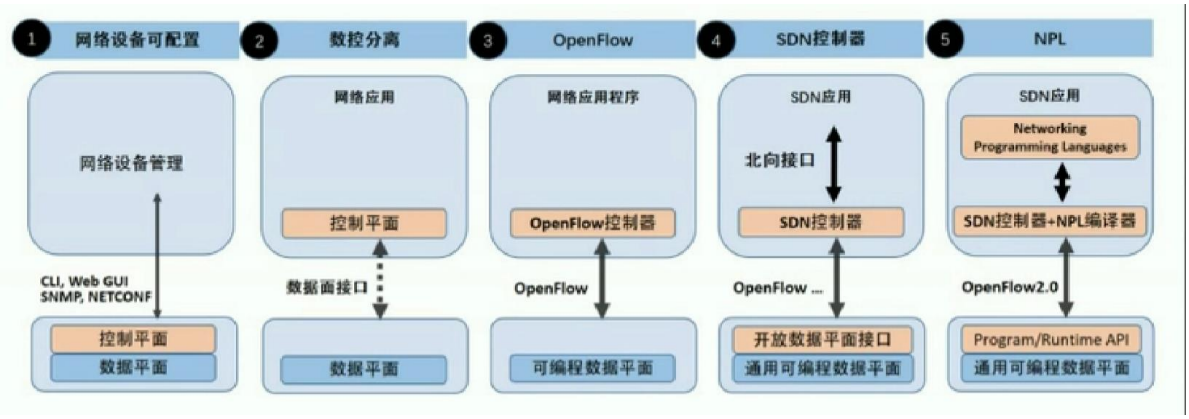
在最初计算机网络设备的一般是这样子的：

我们不难发现这样的网络设备是高度耦合的，就是一个网络设备是一个整体，就是要一个厂商来生产的，并且用户想要实现一些功能只能使用厂商提供的接口和指令，**我们不能自定义自己想要的逻辑**，想要使用自己的安全控制策略，并将这些安全策略应用到各种网络设备中，从而实现对整个网络通讯的安全控制，非常困难。具体有以下缺点：

- **垂直集成的封闭系统**：厂商负责制
- **网络功能的简单堆砌**：一个问题，一个协议
- **行业创新基本停滞**：设备厂商独家话

Martin 和他的导师 Nick McKeown 教授（时任 Clean Slate 项目的 Faculty Director）发现，如果将 Ethane 的设计更一般化，**将传统网络设备的数据转发（data plane）和路由控制（control plane）两个功能模块相分离，通过集中式的控制器（Controller）以标准化的接口对各种网络设备进行管理和配置，**那么这将为网络资源的设计、管理和使用提供更多的可能性，从而更容易推动网络的革新与发展。于是，他们便提出了 OpenFlow 的概念。基于 OpenFlow 为网络带来的可编程的特性，Nick 和他的团队（包括加州大学伯克利分校的 Scott Shenker 教授）进一步提出了 SDN（Software Defined Network，目前国内多直译为“软件定义网络”）的概念。

SDN 的相关发展历史：



3.2 什么是 SDN？

开放的架构
Intel＋linux＋APP
问题－应用

如果将网络中所有的网络设备视为被管理的资源，那么参考操作系统的原理，可以抽象出一个网络操作系统（Network OS）的概念—这个网络操作系统一方面抽象了底层网络设备的具体细节，同时还为上层应用提供了统一的管理视图和编程接口。这样，基于网络操作系统这个平台，用户可以开发各种应用程序，通过软件来定义逻辑上的网络拓扑，以满足对网络资源的不同需求，而无需关心底层网络的物理拓扑结构。

从上面的描述中，可以看出 OpenFlow/SDN 的原理其实并不复杂，从严格意义上讲也很难算是具有革命性的创新。然而 OpenFlow/SDN 却引来了业界越来越多的关注，成为近年来名副其实的热门技术。截止 2012 年，包括 HP、IBM、Cisco、NEC 以及国内的华为和中兴等传统网络设备制造商都已纷纷加入到 OpenFlow 的阵营，同时有一些支持 OpenFlow 的网络硬件设备已经面世。2011 年，开放网络基金会（Open Networking Foundation）在 Nick 等人的推动下成立，专门负责 OpenFlow 标准和规范的维护和发展；同年，第一届开放网络峰会（OpenNetworking Summit）召开，为 OpenFlow 和 SDN 在学术界和工业界都做了很好的介绍和推广。2012 年年初召开的第二届峰会上，来自 Google 的 Urs Hölzle 在以 OpenFlow@Google[7]为题的 Keynote 演讲中宣布 Google 已经在其全球各地的数据中心骨干网络中大规模地使用 OpenFlow/SDN，从而证明了 OpenFlow 不再仅仅是停留在学术界的一个研究模型，而是已经完全具备了可以在产品环境中应用的技术成熟度。而后，Facebook 也宣布其数据中心中使用了 OpenFlow/SDN 的技术。

3.3 SDN 的特点与优势是什么？

SDN 的特点：

- **网络开放可编程**
- **数控分离**
- **逻辑上集中控制**

SDN 是一种新的网络体系结构，我认为给传统网络带来最大的改变是网络**可编程和开放性**。网络用户追逐 SDN 的关键是想获得更多的网络可编程能力，获得更多的网络定制开发能力和自主权，所以一开始 SDN 的发展并不是网络设备的厂商推动的，而是用户如大学教授等推动的。其次 SDN 的开放分层架构加速了网络产业的参与度，越来越多的网络用户、网络软件公司和初创公司都加入到网络产业中来，**这种开放竞争进一步加速整个产业的创新。**

3.4 RYU 是什么？和 SDN 有什么关系？

我们了解了 SDN 是什么后，再来了解 RYU 是什么。SDN 只是一种标准一种思想，具体对 SDN 的实现由以下几种开源 SDN 控制器。

	NOX (2008)	Ryu (2012)	Floodlight (2012.1)	ONOS (2014.12)	ODL (2013.12)
系统架构	Centralized multi-threaded	Centralized multi-threaded	Centralized	Distributed	Distributed
北向接口	C++ API	REST API	RESTful API/Java API	RESTful API	**REST/RESTCONF/Java**
编程语言	C++/Python	Python	Java	Java	Java
管理	--	CLI	Web UI	Web GUI/CLI	Web GUI/CLI
南向接口	OpenFlow	Full	OpenFlow	Full	Full
OpenStack支持	No	Yes	Yes	--	Yes
OpenFlow支持	v1.0	**v1.{0,2,3,4,5} OF++**	V1.{0,3}	v1.0	v1.{0,3}
一致性	No	No	No	Strong	Weak
容错能力	No	No	No	Yes	No
学习曲线	Medium	**Easy**	Medium	Medium	Hard
License	GPLv3	Apache 2.0	Apache	Apache 2.0	EPL v1.0
社区活跃度	28 / 6	2460 / 57	2511 / 53	4453 / 55	**6384 / 60**

其中初学者最常用的一般是 RYU。RYU 很好的实现了 SDN，具体来说向上给用户提供了编程能力，向下实现了 Open flow，支持使用 Open flow 协议与下层硬件通信。所以本质上 RYU 就是一种 SDN 的控制器。

RYU 是一种基于 **Python 语言的软件定义网络（SDN）控制器**，它提供了一个开放的应用程序接口（API），使网络管理员和开发人员能够轻松地编写新的网络控制应用程序来进行网络流量的控制和管理。Ryu 通过 OpenFlow 协议与网络交换机通信，可以对网络设备进行配置、监控和管理。Ryu 是一个开源项目，由日本 NTT 实验室开发和维护，被广泛应用于 SDN 应用程序的开发和部署。

所以 RYU 与 SDN 是什么关系呢？"RYU" 和 "SDN"（软件定义网络）之间的关系可以通过一个通俗的比喻来解释。

想象一下，SDN 是一种创新的建筑设计理念，这种理念将建筑物（在这个比喻中，指的是网络结构）的设计和管理方式彻底改变了。在传统的网络结构中，网络设备（如交换机和路由器）的控制功能和数据转发功能是紧密结合在一起的。但在 SDN 的理念中，这两个功能被分离开来。控制功能（即决定数据如何流动）被集中到一个称为“控制器”的地方，而网络设备则仅负责数据的实际转发。这就像是在建筑物中，将设计决策（控制功能）和实际的建筑施工（数据转发）分开。

RYU 就好比是实现这种新建筑设计理念的一套工具或者框架。它是一个用来开发和管理 SDN 控制器的软件平台。使用 RYU，开发者可以编写程序来控制网络流量的流动方式，就像建筑师使用特定的工具和语言来设计建筑物一样。RYU 提供了一种方式，让开发者能够更加灵活和创造性地管理网络资源，这在传统网络结构中是很难实现的。

所以 SDN 是一种新的网络设计和管理理念，而 RYU 是在这个理念下用来开发和管理网络的一个具体工具或平台。

3.5 Mininet 是什么?和 SDN 有什么关系？和 RYU 又是什么关系？

Mininet 是一个强大的网络模拟工具，它可以在单台计算机上创建一个虚拟的网络环境。这个工具非常适合那些需要研究、教学或测试网络系统的人员使用。想象一下，Mininet 就像是一个网络实验室，但它完全存在于计算机软件中。

为了通俗易懂地解释 Mininet，我们可以把它比作一个虚拟的“乐高积木”套装，专门用于构建网络。就像乐高积木可以用来构建不同的结构一样，使用 Mininet，你可以“搭建”各种网络结构，包括交换机、路由器、计算机等网络设备，甚至是整个网络的连接方式。这一切都在你的电脑上虚拟进行，无需实际的物理设备。

Mininet 的一个主要优势是它的灵活性和低成本。由于它是虚拟的，你可以轻松地实验和测试不同的网络配置和场景，而不必担心实际设备的限制或成本。例如，如果你想测试一个新的网络协议或配置，只需在 Mininet 中设置你的虚拟网络，然后运行实验，就可以看到结果。这对于学习网络原理、开发新的网络技术或测试网络应用来说非常有用。

所以 Mininet 是一个模拟真实网络环境的强大工具，它通过在单台电脑上虚拟化网络设备和链接，提供了一个灵活、易于使用、成本低廉的网络测试和研究环境

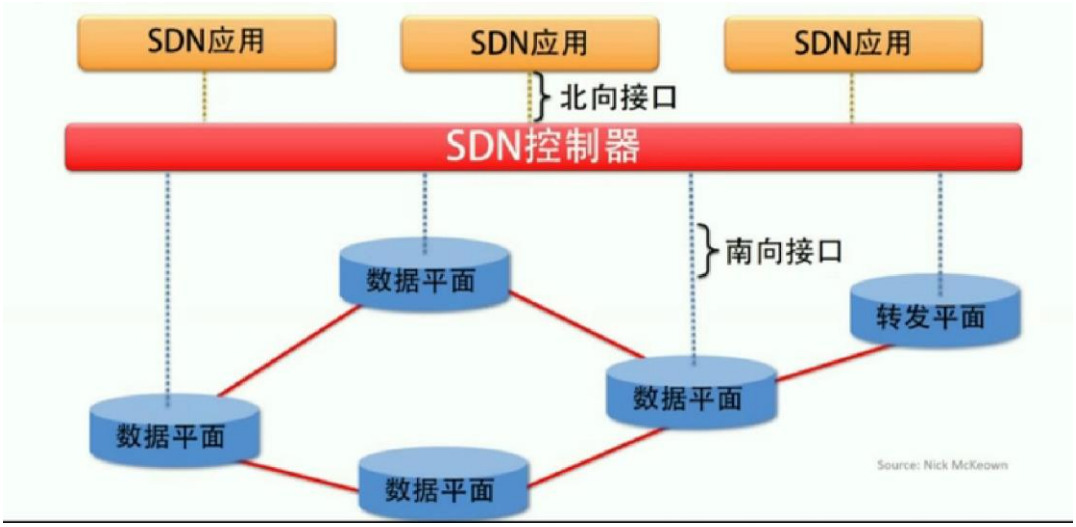
Mininet 和 SDN 有什么关系呢？SDN 全名为（Software Defined Network）即软件定义网络，是现互联网中一种新型的网络创新架构，其核心技术 OpenFlow 通过网络设备控制面与数据面分离开来，从而实现网络流量的灵活控制，为网络及应用提供了良好的平台。而 Mininet 是一个轻量级软件定义网络和测试平台；它采用轻量级的虚拟化技术使一个单一的系统看起来像一个完整的网络运行想过的内核系统和用户代码，也可简单理解为 SDN 网络系统中的一种基于进程虚拟化平台，它支持 OpenFlow、Open vSwith 等各种协议，Mininet 也可以模拟一个完整的网络主机、链接和交换机在同一台计算机上且有助于互动开发、测试和演示，尤其是那些使用 OpenFlow 和 SDN 技术；同时也可将此进程虚拟化的平台下代码迁移到真实的环境中。

所以换句话说就是 Mininet 为可以为 SDN 在真实部署之前提供模拟环境，我们不需要真正的找到那么多网络设备，而使用 Minine 模拟他们。

Mininet 和 RYU 有什么关系呢？我们了解了 Mininet 和 SDN 的关系后，很容易就知道了和 RYU 的关系，我们可以先使用 mininet 搭建一个虚拟的网络，然后使用 RYU 控制器来控制这些设备，以实现在部署之前的检查和仿真。

3.6 Open Flow 是什么？和 SDN 有什么关系？

OpenFlow 是 SDN 的基础，如果没有 OpenFlow 就没有 SDN。OpenFlow 起源于斯坦福大学的 Clean Slate 项目组 。CleanSlate 项目的最终目的是要重新发明英特网，旨在改变设计已略显不合时宜，且难以进化发展的现有网络基础架构。在 2006 年，斯坦福的学生 Martin Casado 领导了一个关于网络安全与管理的项目 Ethane，该项目试图通过一个集中式的控制器，让网络管理员可以方便地定义基于网络流的安全控制策略，并将这些安全策略应用到各种网络设备中，从而实现对整个网络通讯的安全控制。受此项目（及 Ethane 的前续项目 Sane)启发，Martin 和他的导师 Nick McKeown 教授(时任 Clean Slate 项目的 Faculty Director）发现，如果将 Ethane 的设计更一般化，**将传统网络设备的数据转发（data plane）和路由控制（control plane）两个功能模块相分离，通过集中式的控制器（Controller）以标准化的接口对各种网络设备进行管理和配置，**那么这将为网络资源的设计、管理和使用提供更多的可能性，从而更容易推动网络的革新与发展。于是，他们便提出了 OpenFlow 的概念，并且 Nick McKeown 等人于 2008 年在 ACM SIGCOMM 发表了题为 OpenFlow: Enabling Innovation in Campus Networks[4]的论文，首次详细地介绍了 OpenFlow 的概念。该篇论文除了阐述 OpenFlow 的工作原理外，还列举了 OpenFlow 几大应用场景，包括：1）校园网络中对实验性通讯协议的支持（如其标题所示）；2）网络管理和访问控制；3）网络隔离和 VLAN；4）基于 WiFi 的移动网络；5）非 IP 网络；6）基于网络包的处理。当然，目前关于 OpenFlow 的研究已经远远超出了这些领域



这里我们可以根据 SDN 分层层次结构的这张图看出一个 SDN 不仅要有面向用户的北向接口，还有面向硬件的南向接口，而 SDN 与这些硬件是解耦合，那么就需要一种通讯协议来让双方都遵守，也就是 OpenFlow，

所以说有了 OpenFlow 才让数据转发和路由控制两个功能模块相分离，才有了 SDN 的基础。

3.7 其他额外的知识

DHCP 是什么？

DHCP（动态主机配置协议）是一种在网络上自动分配 IP 地址和其他网络配置信息的技术。为了更通俗易懂地解释它，我们可以将其比作是一家餐厅的服务员。

想象一下，当你走进一家餐厅时，服务员会带你到一个空闲的桌子上坐下。在网络世界中，DHCP 服务器就像这个服务员一样。当一台电脑或其他设备（比如智能手机、平板电脑）连接到一个网络（比如你家里的 WiFi 或公司的网络）时，它需要一个 IP 地址来识别自己并与网络上的其他设备通信。

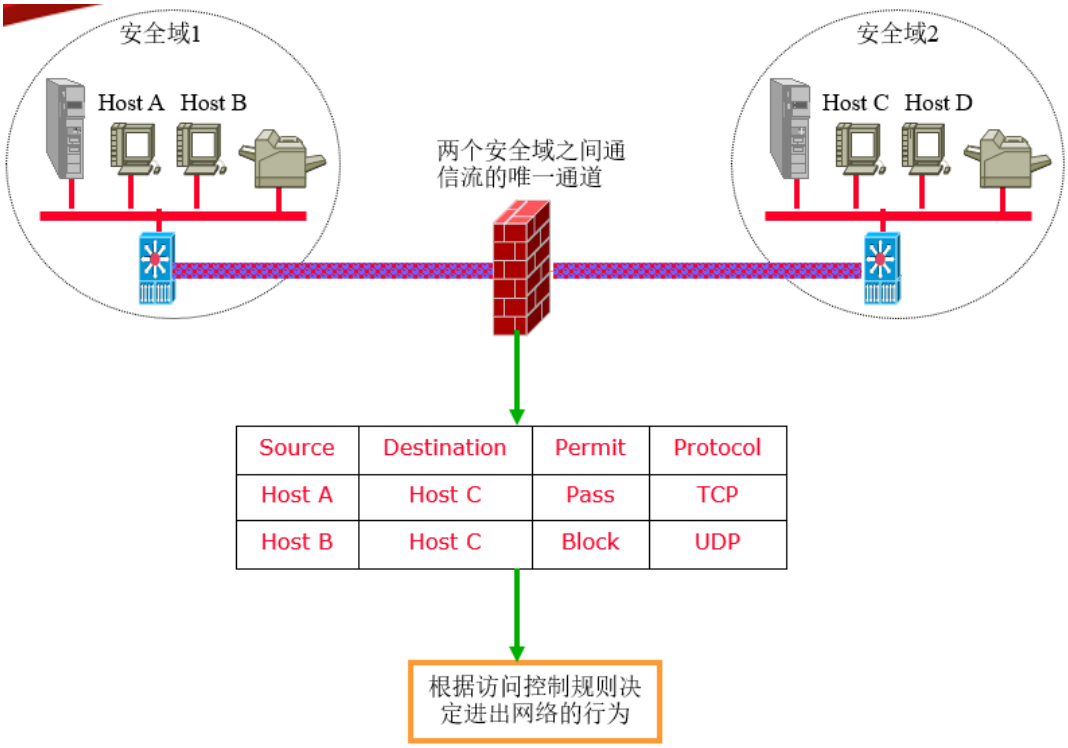
这台电脑或设备会向网络上的“服务员”（DHCP 服务器）发送请求，询问可以使用的 IP 地址。DHCP 服务器会从它管理的地址池中选择一个未被占用的 IP 地址，就像服务员会选择一张空闲的桌子一样。然后，它会“分配”这个 IP 地址给请求的设备，同时还会提供其他一些必要的信息，比如子网掩码、默认网关和 DNS 服务器地址。

这个过程的优点是自动化和简单。设备用户不需要手动设置 IP 地址，网络管理员也不需要手动为每台设备分配固定的 IP 地址。DHCP 使得设备可以轻松地加入网络，获得所需的所有配置信息，从而快速开始通信。

简单来说 DHCP 就像是网络上的服务员，它自动地为设备提供所需的“座位”（IP 地址）和其他“餐具”（网络配置信息），从而使设备能够顺利地“用餐”（加入并使用网络）。

防火墙是什么？

防火墙（Firewall），也称防护墙，是由 Check Point 创立者 Gil Shwed 于 1993 年发明并引入国际互联网它是一种位于内部网络与外部网络之间的网络安全系统。是一项信息安全的防护系统，依照特定的规则，允许或是限制传输的数据通过。



在网络的世界里，要由防火墙过滤的就是承载通信数据的通信包。

在网络中，所谓“防火墙”，是指一种将内部网和公众访问网（如 Internet）分开的方法，它实际上是一种隔离技术。防火墙是在两个网络通讯时执行的一种访问控制尺度，它能允许你“同意”的人和数据进入你的网络，同时将你“不同意”的人和数据拒之门外，最大限度地阻止网络中的黑客来访问你的网络。换句话说，如果不通过防火墙，公司内部的人就无法访问 Internet，Internet 上的人也无法和公司内部的

人进行通信。

网络层防火墙可视为**一种 IP 封包过滤器**，运作在底层的 TCP/IP 协议堆栈上。可以**以枚举的方式只允许符合特定规则的封包通过，其余的一概禁止穿越防火墙**（病毒除外，防火墙不能防止病毒侵入）。这些规则通常可以经由管理员定义或修改，不过某些防火墙设备可能只能套用内置的规则。

STP 是什么？

生成树协议（英语：Spanning Tree Protocol，STP），是一种工作在 OSI 网络模型中的第二层(数据链路层)的通信协议，基本应用是防止交换机冗余链路产生的环路。用于确保以太网中无环路的逻辑拓扑结构。从而避免了广播风暴,大量占用交换机的资源。

在 SDN 中，如果 Mininet 建立的拓扑中存在交换机环路，则如果利用普通的 Ryu Learning Switch APP 进行 ryu-manager 部署，会出现 ping、pingall 不通的问题，其原因在于环路中出现了广播风暴。为了在 Mininet 中使用带有环路的拓扑，需要让交换机开启 STP 协议。

什么是 sFlow？

sFlow（Sampled Flow）是一种基于报文采样的网络流量监控技术，主要用于网络流量的统计分析。sFlow 提供基于接口的流量分析，可以实时监控流量状况，及时发现异常流量以及攻击流量的源头，为企业用户的日常巡检维护提供了极大的方便。

为什么要用 sFlow？

相对于电信级网络，企业级网络通常具有规模相对较小、组网灵活、易受攻击等特点，因此企业级网络更容易出现由组网或者攻击导致的流量业务异常。故而企业用户更需要一种以设备接口为基本采样单元的流量监控技术来实时监控流量状况，及时发现异常流量以及攻击流量的源头，从而保证企业网络的正常稳定运行。在这样的背景下，sFlow 应运而生。

4. 实验分析：

当我们在了解以上的实验原理后，我们可以针对本次实验进行分析。

本次实验决定设计一个本次实验决定设计一个具有防火墙、实现 STP、实现 DHCP 分配 IP 的、能够使用 OSPF 路由的、可以实时检测状态的、支持 SDN 的网络，接下来我们将要一个一个进行分析。

4.1 防火墙

防火墙本质上就是一些规则，控制数据包的流入和流出，我们基于 ryu 和 mininet 可以很轻松的实现它，通过 RYU 控制器控制 Mininet 中的网络设备，然后使用 OpenFlow 下发规则然后就可以基本实现防火墙的功能。

4.2 STP

为什么要用 STP？因为我们设计的场景中学校需要保证网络满足实验室考试网络拥堵和网络故障特殊情况，保证网络的质量和考试的网络质量。所以需要三个交换机成环连接，在此需要实现 STP。在 SDN 中，如果 Mininet 建立的拓扑中存在交换机环路,则如果利用普通的 Ryu Learning Switch APP 进行 ryu-manager 部署，会出现 ping、pingall 不通的问题，其原因在于环路中出现了广播风暴。为了在 Mininet 中使用带有环路的拓扑，需要让交换机开启 STP 协议。

4.3 DHCP

为什么要用 DHCP？网络配置采用自动配置有诸多的优点，下面列举了一些：

- 可以很容易地在网络中添加新的客户端。
- IP 地址是由 DHCP 集中管理的。
- IP 地址可以重复使用，从而减少了对 IP 地址总数的要求。
- DHCP 服务器上的 IP 地址空间可以很容易地进行重新配置，而不需要单独重新配置客户端。
- 网络管理员可以利用 DHCP 协议提供的方法，从集中区域配置网络。

综上，我们网络采用 DHCP 的配置方式。

4.4 OSPF

为什么使用 OSPF？开放式最短路径优先（Open Shortest Path First，OSPF）是广泛使用的一种动态路

由协议，它属于链路状态路由协议，具有路由变化收敛速度快、无路由环路、支持变长子网掩码（VLSM）和汇总、层次区域划分等优点。

在网络中使用 OSPF 协议后，大部分路由将由 OSPF 协议自行计算和生成，无须网络管理员人工配置，当网络拓扑发生变化时，协议可以自动计算、更正路由，极大地方便了网络管理。

4.6 sFlow 分析

为什么要使用 sFlow? 我们知道 sFlow 是一种网络监视技术，更方便去监测和分析流量和 CPU 情况，在实际情况中选择如果没有一个可以是监控的工具，出错了我们需要排查很久，并且不能即时的检查到问题和预防问题。

4.5 网络拓扑图设计分析

我们想要设计的比较有实际应用就要考虑到一些实际可能发生的事情，比如网络的入侵和网络的稳定性，但是又不能太过于复杂因为我们只是抽象的解决问题，而不是真正的要应用在现实中的。所以我们可以考虑一两个现实的问题并解决它，然后并抽象出一个简单点的拓扑模型。这样不仅有一定的现实意义也是符合我们的水平的。

5. 实验设计：

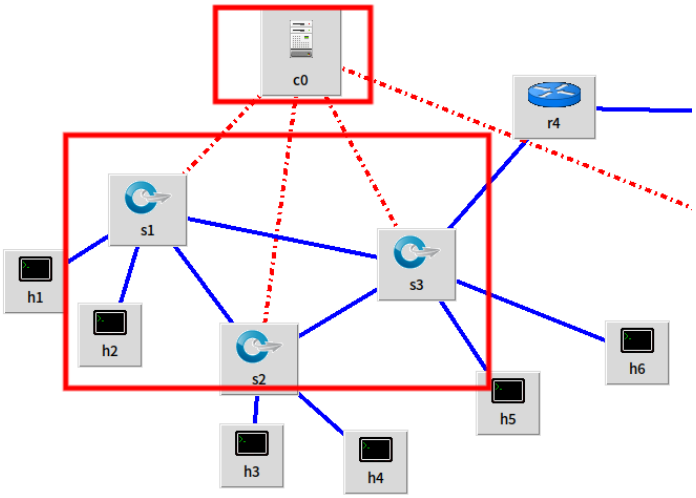
当我们在了解以上的实验分析后，我们可以针对本次实验进行设计。**本次实验决定设计一个具有防火墙、实现 STP、实现 DHCP 分配 IP 的能够使用 OSPF 路由的计算机网络，接下来我们将要一个一个功能进行设计。**

5.1 防火墙设计

防火墙的设计我们采用的是 Mininet+RYU 的方式，RYU 运行防火墙程序。交换机 S7 运行防火墙程序，阻塞不能通过的流表。其中运行防火墙的控制器和另一个 RYU 控制器 C0 不是同一控制器，因为同一个控制器运行两个程序会存在冲突，所以采用了两台控制器。

5.2 STP 设计

我们设计的校园网络中，涉及到实验室场景的模拟。我们希望实验室区域的网络拥有充足的健壮性，所以我们设计了环路，希望正常情况能防止环路形成，特殊情况（某条链路断开）能恢复连通性，保证网络畅通。



如图所示的形成环路的三台交换机，都连接了同一个 ryu 控制器 c0，那么我们只需要 ryu 对应的模块上进行设定就可以实现 stp 协议。

5.3 DHCP 设计

创建两台 unbantu18.0 的虚拟机 A，B，在一台虚拟机 A 上面配置 DHCPservever，使其成为 DHCP 服务器，而在虚拟机 B 上搭建拓扑，使用添加网卡的方式使得两台虚拟机实现互通，使得虚拟机 B 上面中 mininet 的主机能够访问虚拟机 A，从而实现 DHCP 的功能。

5.4 OSPF 设计

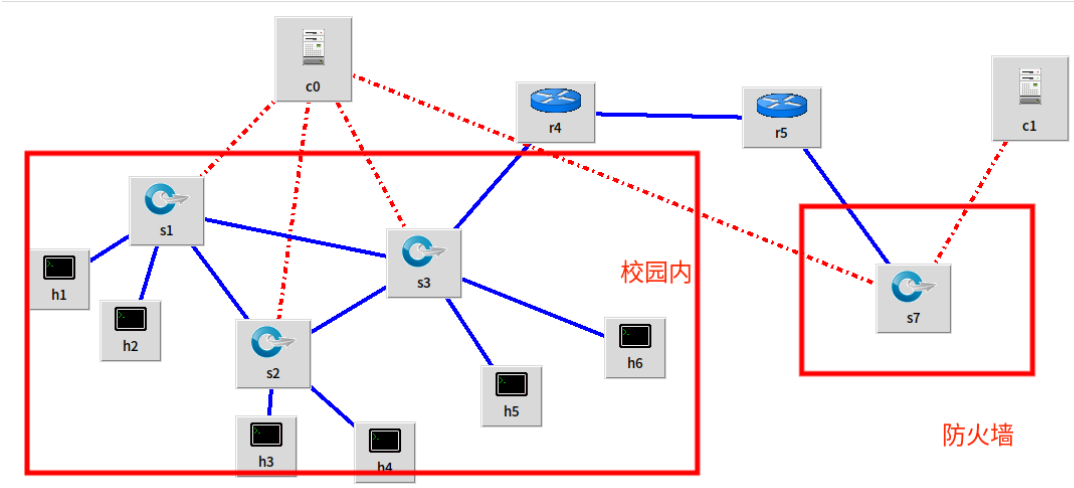
使用一个 linux 的软件 quagga 它可以将 linux 设备变成一个功能完整的路由器。支持 rip，ospf，bgp 等协议。这里我们只使用它的 ospf 功能，然后再 mininet 中的所有网络设备中配置这个软件，开启 ospf 即可实现 OSPF。

5.5 sFlow 设计

首先我们先选择交换机进行配置，本次我们选择的负责防火墙的交换机和内部的交换机。其次就是我们要对他们进行配置。

sFlow 的配置不算复杂，具体配置 sFlow 一共分为两步，一步是配置远端的 sFlow Collector（也就是在本机电脑上）另一步是配置嵌入在设备中的 sFlow Agent（也就是在 mininet 中的虚拟交换机上），配置完了就可以在本机电脑 127.0.0.1：8008 中查看流量的大小等来实时监控网络状态

5.6 总体网络拓扑图设计



经过上面的分析，你肯定对这个总体的网络拓扑为什么要设计成这样的不再迷惑，**因为每一部分都是有来由的，**那么接下来我们将对设计进行实现。

6. 实验过程：

6.1 SDN 配置与实现

SDN 采用 RYU 控制器和 mininet 的方式，交换机采用的是 openflow13 协议。

配置：

主要是涉及 Mininet 和 RYU 控制器的安装，路由定义软件 Quagga 的安装和配置。

详细的配置步骤在网络上都可以找到，在此不在叙述。

实现：

第一步：打开 RYU 控制器，运行相应的程序（比如 simple_switch_13.py 默认路由，rest_firewall.py 防火墙程序）

```
packet in 3 5e:83:0f:ad:01:3a 33:33:00:00:00:01 4
^C(genui) new@new-Jiaolong-Series-GK5NPFO:~/桌面/ryu1/ryu/app$ ryu-manager example_switch_13.py
loading app example_switch_13.py
loading app ryu.controller.ofp_handler
instantiating app example_switch_13.py of ExampleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

第二步：运行拓扑代码，连接到控制器

```
(base) new@new-Jiaolong-Series-GK5NPFO:~/桌面/mininet/examples$ sudo /usr/bin/python3.10 /home/new/桌面/mininet/examples
/miniedit.py
topo=none
Getting Hosts and Switches.
Getting controller selection:remote
Unable to contact the remote controller at 127.0.0.1:6633
Getting controller selection:ref
Getting Links.
*** Configuring hosts
h1 h2 h3 h4 h5 h6 r4 r5
**** Starting 2 controllers
```

6.2 防火墙实现

防火墙采用的是 RYU 控制器，通过在 RYU 控制器中的防火墙程序（rest_firewall.py）文件进行实现。

```python
import requests  # 导入requests库，用于发送HTTP请求

# 定义四条规则的数据，每条规则指定源地址和目的地址
data1 = '{"nw_src":"192.168.1.1","nw_dst":"192.168.2.1"}'
data2 = '{"nw_src":"192.168.2.1","nw_dst":"192.168.1.1"}'
data3 = '{"nw_src":"192.168.2.2","nw_dst":"192.168.1.1"}'
data4 = '{"nw_src":"192.168.1.1","nw_dst":"192.168.2.2"}'

def firewall():
    # 启用名为 '0000000000000004' 的交换机的防火墙模块
    response = requests.put('http://localhost:8080/firewall/module/enable/0000000000000004')

    # 为交换机 '0000000000000004' 添加规则，限制流量根据data1定义的规则进行过滤
    response = requests.post('http://127.0.0.1:8080/firewall/rules/0000000000000004', data=data1)

    # 添加第二条规则，限制流量根据data2定义的规则进行过滤
    response = requests.post('http://127.0.0.1:8080/firewall/rules/0000000000000004', data=data2)

    # 添加第三条规则，限制流量根据data3定义的规则进行过滤
    response = requests.post('http://127.0.0.1:8080/firewall/rules/0000000000000004', data=data3)

    # 添加第四条规则，限制流量根据data4定义的规则进行过滤
    response = requests.post('http://127.0.0.1:8080/firewall/rules/0000000000000004', data=data4)

# 调用firewall函数，执行防火墙规则设置
firewall()
```

这里只模拟限制了几个网址和生成的几个规则和之前访问控制流表是一样的。

然后用 RYU 启动这个文件，通过远程控制交换机来遵循这些规则。

6.3 STP 实现

STP 要在 RYU 中进行配置，然后下发给交换机。

具体代码：

```python
from ryu.lib import stplib  # 引入stp包

1 个用法
class SimpleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    _CONTEXTS = {'stplib': stplib.Stp}  #添加STP功能
```

然后再运行 RYU 控制器(ryu-manager example_switch.py)就可以实现 STP 的功能了

6.4 DHCP 实现

在实验的过程中，我们使用了实验设计中方法，但是发现 DHCP 无法给 mininet 中的主机自动分配地址，思考其原因是：由于一开始 IP 地址的缺失，导致虚拟机 B 上的网络处于 ping 不通的未完成状态，因此，即使虚拟机 A 能够实现与虚拟机 B 的通信，但无法进行对未完成状态的网络进行 DHCP。所以我们要先配置一个初始的 IP 然后再进行 DHCP 获取 IP。

具体代码：启动 DHCP 服务器

```
1 个用法
def startDHCPserver(host, gw, dns):
    "Start DHCP server on host with specified DNS server"

    info('* Starting DHCP server on', host, 'at', host.IP(), '\n')

    dhcpConfig = '/tmp/%s-udhcpd.conf' % host

    makeDHCPconfig(dhcpConfig, host.defaultIntf(), gw, dns)

    host.cmd('udhcpd -f', dhcpConfig,

             '1>/tmp/%s-dhcp.log 2>&1  &' % host)
```

为每台主机启动 DHCP 客户端

```
1 个用法
def startDHCPclient(host):
    "Start DHCP client on host"

    intf = host.defaultIntf()

    host.cmd('dhclient -v -d -r', intf)

    host.cmd('dhclient -v -d 1> /tmp/dhclient.log 2>&1', intf, '&')
```

获取 IP：

```
1 个用法
def waitForIP(host):
    "Wait for an IP address"
    info('*', host, 'waiting for IP address')
    while True:
        host.defaultIntf().updateIP()
        # print(host.IP())
        if host.IP():
            break
        info('.')
        sleep(1)
    info('\n')
```

6.5 OSPF 实现

使用 quagga 来实现 OSPF 动态路由，也就是说，要对每台路由器进行配置 quagga 软件，然后启动后分配完成 IP 后就启动 quagga 软件。

具体代码：

```
r1.cmd('zebra -f /etc/quagga/r1zebra.conf -d -z /tmp/r1zebra.api -i /tmp/r1zebra.interface')
r2.cmd('zebra -f /etc/quagga/r2zebra.conf -d -z /tmp/r2zebra.api -i /tmp/r2zebra.interface')

time.sleep(1)  # time for zebra to create api socket
r1.cmd('ospfd -f /etc/quagga/r1ospfd.conf -d -z /tmp/r1zebra.api -i /tmp/r1ospfd.interface')
r2.cmd('ospfd -f /etc/quagga/r2ospfd.conf -d -z /tmp/r2zebra.api -i /tmp/r2ospfd.interface')
```

使用后就可以实现每台路由器配置了的 OSPF 动态路由，其过程和我们实验课中配置动态了路由的过程相似只不过使用代码控制路由器执行命令，生效的。

6.6 sFlow 实现

① 首先配置 sFlow Collector

```
(base) new@new-Jiaolong-Series-GK5NPFO:~/桌面/sflow-rt$ ./start.sh
2024-01-10T14:42:54+08:00 信息: Starting sFlow-RT 3.0-1697
2024-01-10T14:42:56+08:00 信息: Version check, running latest
2024-01-10T14:42:56+08:00 信息: Listening, sFlow port 6343
2024-01-10T14:42:56+08:00 信息: Listening, HTTP port 8008
```

② 然后配置 sFlow Agent

首先打开虚拟交换机的终端

```
*** Starting CLI:
mininet> xterm s1
mininet>
```

```
root@new-Jiaolong-Series-GK5NPFO:/home/new/面/mininet/examples#
```

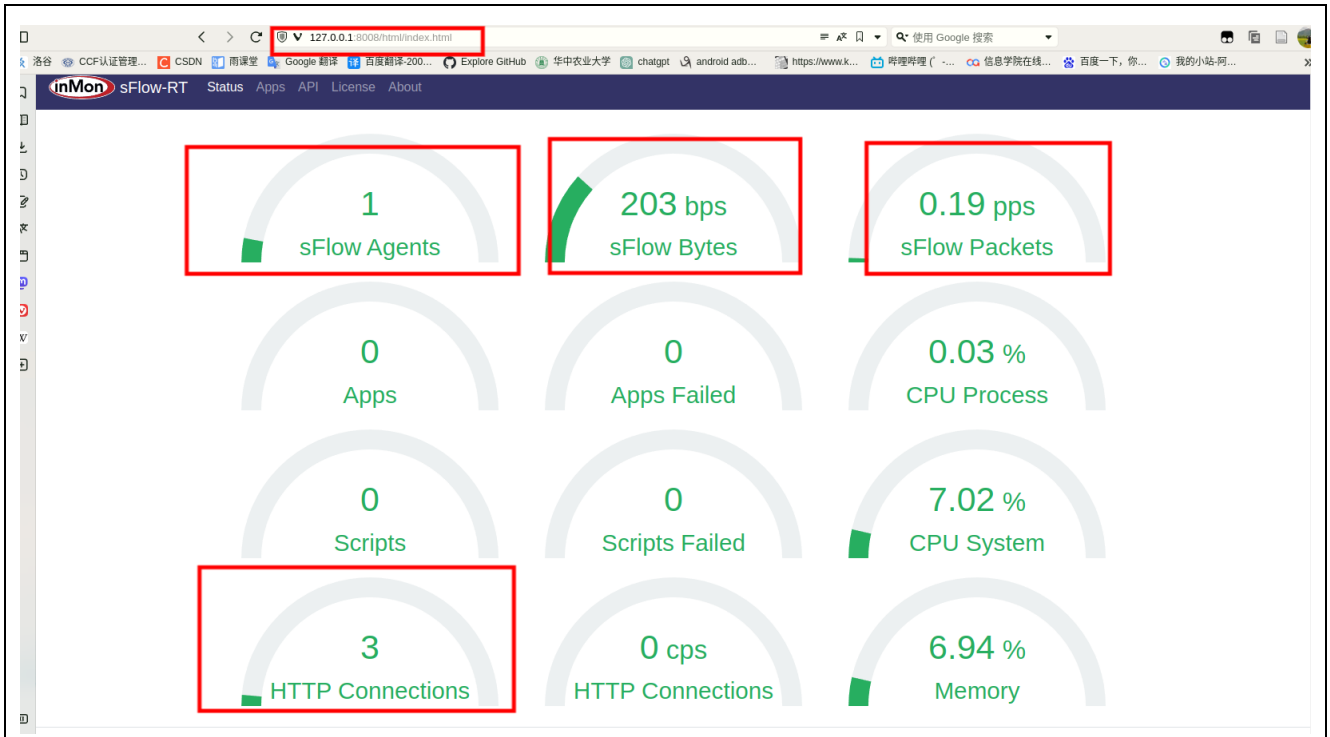
然后打开 输入

sudo ovs-vsctl -- --id=@sflow create sflow agent=s1-eth0 target=\"127.0.0.1:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow

```
root@new-Jiaolong-Series-GK5NPFO:/home/new/面/mininet/examples# sudo ovs-vsct
l -- --id=@sflow create sflow agent=s1-eth0 target=\"127.0.0.1:6343\" sampling=
10 polling=20 -- -- set bridge s1 sflow=@sflow
4def7604-2f83-4ead-9418-30c74634e957
root@new-Jiaolong-Series-GK5NPFO:/home/new/面/mininet/examples#
```

表示 sFlow agent 建立成功了。

③ 查看结果



然后重复以上几部知道所有交换机都部署了 sFlow agent

6.7 总体实现

将之前的代码整合在一起，然后启动 ryu 控制器

```
(genui) new@new-Jiaolong-Series-GK5NPFO:~/桌面/mininet/examples/myapp$ ryu-manager myryu.py
loading app myryu.py
loading app ryu.controller.ofp_handler
instantiating app None of Stp
creating context stplib
instantiating app myryu.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
[STP][INFO] dpid=0000000000000001: Join as stp bridge.
[STP][INFO] dpid=0000000000000001: [port=4] DESIGNATED_PORT      / LISTEN
[STP][INFO] dpid=0000000000000001: [port=1] DESIGNATED_PORT      / LISTEN
[STP][INFO] dpid=0000000000000001: [port=2] DESIGNATED_PORT      / LISTEN
[STP][INFO] dpid=0000000000000001: [port=5] DESIGNATED_PORT      / LISTEN
[STP][INFO] dpid=0000000000000002: Join as stp bridge.
[STP][INFO] dpid=0000000000000001: [port=3] DESIGNATED_PORT      / LISTEN
[STP][INFO] dpid=0000000000000002: [port=1] DESIGNATED_PORT      / LISTEN
```

启动 mininet 进行仿真模拟

```
(base) new@new-Jiaolong-Series-GK5NPFO:~/桌面/mininet/examples/myapp$ sudo /usr/bin/python3.10 net.py
Unable to contact the remote controller at 127.0.0.1:6633
Unable to contact the remote controller at 127.0.0.1:6634
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 nat0 r1 r2 r3
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(r1, nat0) (r1, r2) (r3, r1) (s1, h1) (s1, h2) (s1, r2) (s1, s2) (s1, s3) (s2, h3) (s2, h4) (s3, h5) (s3, h6
s4, h7) (s4, h8) (s4, h9) (s4, h10) (s4, h11) (s4, r3)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 nat0 r1 r2 r3
*** Starting controller
c0 c1
*** Starting 4 switches
s1 s2 s3 s4 ...
* Starting DHCP server on h1 at 192.168.1.1
* h2 waiting for IP address.....
```

最终我们可以 pingall 测试联通，发现是可以互相联通的。

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results: 0% dropped (30/30 received)
```

7. 结论与分析：

7.1出现的问题与解决方法

首先是配置环境过程中出现了各种问题：

使用 conda 环境中的 python 不能安装依赖：新建了环境使用 python 3.10 成功安装依赖

安装 mininet 报错：

```
-DVERSION=\"`PYTHONPATH=. python -B bin/mn --version 2>&1`\" mnexec.c -o
<command-line>: warning: missing terminating " character
mnexec.c: In function ?main?:
<command-line>: error: missing terminating " character
mnexec.c:208:28: note: in expansion of macro ?VERSION?
  208 |             printf("%s\n", VERSION);
      |                            ^~~~~~~
mnexec.c:208:35: error: expected expression before ?)? token
  208 |             printf("%s\n", VERSION);
      |                                   ^
make: *** [Makefile:50: mnexec] Error 1
```

通过建立软连接的方式解决

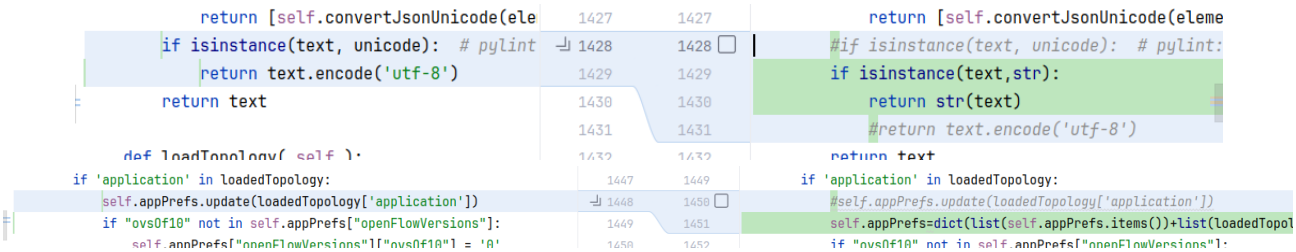
```
sudo ln -s /usr/bin/python3 /usr/bin/python
```

使用 mininet 也有出现了各种问题：

无法将保存的拓扑图打开报错：

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "/usr/lib/python3.8/tkinter/__init__.py", line 1883, in
    return self.func(*args)
  File "miniedit.py", line 1447, in loadTopology
    loadedTopology = self.convertJsonUnicode(json.load(f))
  File "miniedit.py", line 1426, in convertJsonUnicode
    return {self.convertJsonUnicode(key): self.convertJsonUni
  File "miniedit.py", line 1426, in <dictcomp>
    return {self.convertJsonUnicode(key): self.convertJsonUni
  File "miniedit.py", line 1429, in convertJsonUnicode
    elif isinstance(text, unicode):
NameError: name 'unicode' is not defined
```

修改 miniedit.py 源代码解决



7.2实验感想

“学习一个技术就要知道他的来龙去脉”，本次实验是真正的拓宽了我的视野，我之前从没有注意到这块的知识，当我再了解了 SDN 的前世今生后，感觉已经形成了对 SDN 的大概认识，之后的学习都是在此基础上，就容易许多，所以的以后学习一个东西就要了解它的前世今生，这可以说是一个影响深远的学习方法吧。大学不仅教我们如何解决问题的，更多的是培养我们解决问题的能力的，从这次实验来看，我已经在解决诸多问题的过程中不知不觉收获了许多解决问题的方法和经验，这都会成为我解决问题的能力的一部分。

本次实验算是为这 8 次计算机网络实验加上了圆满的句号，这一学期计算机网络从一开始的网络边缘边的探索，然后逐渐到计算机网络核心，从小开始到整体，我们从小的细节开始学习，然后逐渐到扩展到整体，所以我们学习到到知识是系统，是成体系的。并且回看这么多次实验，全都是以实际问题为导向进行的，所以可以说我们计网实验很有现实关照，很符合计算机这么实践中的学科。总结来说收获颇丰，我们已经在计网实验中成长了许多，现实了许多，清晰了很多，不再是之前的幼稚的混沌的迷茫的了，如果许年后再回忆大学时光，那么学习计算机网络，做计算机网络实验这段时光一定还是如现在记忆这般清晰与深刻。