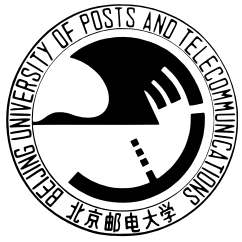
密级：　　保密期限：

北京郵電大學

硕士学位论文



题目：**xxxxxxxxxxxxxxx**

学　　号：**xxxxxxxxx**

姓　　名：**xxxx**

专　　业：信息与通信工程

导　　师：**xxxxx**

学　　院：网络技术研究院

2015 年 12 月 26 日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

本学位论文不属于保密范围，适用本授权书。

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

xxxxxxxxxxxxxxx

摘 要

中、英文摘要位于声明的次页，摘要应简明表达学位论文的内容要点，体现研究工作的核心思想。重点说明本项科研的目的和意义、研究方法、研究成果、结论，注意突出具有创新性的成果和新见解的部分。

关键词是为文献标引工作而从论文中选取出来的、用以表示全文主题内容信息的术语。关键词排列在摘要内容的左下方，具体关键词之间以均匀间隔分开排列，无需其它符号。

关键词：TEX LATEX xeCJK 模板 排版 论文

EXAMPLE OF BUPT GRADUATE THESIS LaTeX 2ε TEMPLATE

ABSTRACT

The Chinese and English abstract should appear after the declaration page. The abstract should present the core of the research work, especially the purpose and importance of the research, the method adopted, the results, and the conclusion.

Key words are terms selected for documentation indexing, which should present the main contributions of the thesis. Key words are aligned at the bottom left side of the abstract content. Key words should be seperated by spaces but not any other symbols.

KEY WORDS: TeX LaTeX xeCJK template typesetting thesis

目 录

第一章 绪论

1.1 研究背景及意义

随着互联网的发展，云数据中心的规模不断扩大，业务流量不断变化，如何为租户提供可编程的云数据中心网络，如何对云数据中心的流量进行有效的控制，提高带宽的利用率，降低成本，成为目前急需解决的问题。

软件定义网络（Software Defined Network，SDN）作为一种新兴的可编程网络架构，有动态配置、可编程及快速响应的特点。其核心思想是将网络控制平面与数据转发平面分离，实现控制平面对数据平面的全局集中化控制；同时对外提供开放的可编程接口，为网络提供可编程能力。控制权的迁移使得底层构架能够抽象出来，各种应用和网络服务因此能将网络当作一个逻辑或虚拟实体，不再依赖于底层网络设备[1]，使得网络配置的自动化程度得到极大提高。通过应用 SDN，除了网络的设计和操作变得简化，网络设备也得到简化，这些设备无需理解或处理成千上万的协议，只需要接受 SDN 控制器的指令即可。利用集中控制，网络管理员可以实时改变网络的行为，并且在几小时或几天内就可以部署新的应用和网络服务。

网络虚拟化[2] 是一种将底层网络中的硬件以及配套的软件资源进行整合，形成统一管理实体的技术，通过虚拟网络资源到物理网络资源的映射，使得多个逻辑虚拟网络共享底层物理网络基础设施, 为用户提供差异化服务。网络虚拟化技术是当今网络革新的重要技术之一。从概念上，网络虚拟化与 SDN 是互相独立的，但随着近几年网络技术的发展与融合，二者之间的联系变得越来越紧密，SDN 的技术相关专题常会提及网络虚拟化技术，网络虚拟化问题的研究也时常会运用到 SDN 的概念，可见基于 SDN 的网络虚拟化技术已经成为网络技术研究领域的一个专门课题。

OpenStack[3] 是由 Rackspace 和美国国家航空航天局（NASA）合作研发的用于搭建 Iaas 平台的云计算管理软件。旨在为公共及私有云的建设与管理提供软件的开源项目，主要提供计算、存储、网络服务。OpenStack 支持几乎所有类型的云环境，项目目标是提供实施简单、可大规模扩展、丰富、标准统一的云计算管理平台。OpenStack 通过各种互补的服务提供了基础设施即服务（Infrastructure as a Service，IaaS）的解决方案，每个服务提供 API 以进行集成[4]。

在现有 OpenStack 云平台中，租户网络的创建与隔离仅限于服务器内部，Open-

Stack 无法进行物理服务器之间数据中心网络的管控，跨服务器的通信通过隧道技术实现，该模式无法满足用户多样性的需求，同时无法实现云数据中心带宽资源的有效利用, 时常会出现有些链路阻塞严重，而有些链路则处于空闲状态。

本文以此作为出发点，提出了一种多租户虚拟网络定制化管理方案，运用虚拟化技术，为租户提供相互隔离的虚拟 SDN 网络（virtual Software Defined Network，vSDN），完成云数据中心物理资源有效利用的同时，SDN 网络由租户自有的控制器实现集中管控，租户可以实时监测当前的流量状况并根据流量状况自定义转发路径，实现对网络的灵活管控，在不降低云平台性能的前提下，实现了 OpenStack 云平台中租户网络的定制化操作，既提高了安全性，又可以根据当下链路的剩余带宽，进行链路的定制化，提高带宽的利用率。对于租户本身而言，真正实现了租户对全局网络的可控性。对于运营商来说，对物理网络的集中控制可以大大较小网络配置的繁琐性，可以对网络故障实现快速的排查，提高可扩展性。

1.2 主要研究内容及创新点

1.2.1 研究内容

本课题为实现基于 SDN 的云平台多租户虚拟网络定制化方案，将 SDN 集成到云数据中心，通过虚拟化技术，为租户提供了可编程的 vSDN 网络，vSDN 网络由租户自由的 SDN 控制器进行集中管控，租户可以根据当前的链路状况，进行定制化流表的下发，真正实现对网络的灵活定制化操作。主要包括以下三个方面的研究内容：

第一、实现虚拟网络的创建：通过虚拟化技术，实现跨物理服务器，数据中心网络的虚拟化，虚拟网络支持 SDN 模式，由租户自己的 SDN 控制器实现集中管控，本文选用网络虚拟化平台（OpenVirteX，OVX）[5] 实现虚拟网络的创建。对于底层物理设备来说，OVX 是一个控制器；对于租户控制器来说，OVX 可以看做是 OpenFlow 交换机的集合，租户控制器看到的只是一张虚拟网络。OVX 最大的优势是紧密结合了 SDN，可以发挥 SDN 的控制与转发分离的强大优势，将创建的虚拟网络指定 SDN 控制器，从而可以运用 SDN 的优势，实现对租户虚拟网络的管控。该部分主要对 OVX 进行二次开发，为其封装手动创建虚拟网络的 API 接口，完成与 OpenStack 的集成，实现 OVX 对 OpenStack 虚拟机的集中管控。

第二、完成虚拟网络的集中控制：针对租户创建的 vSDN 网络，由租户自己的 SDN 控制器进行集中管控，本文为用户提供了集成 Ryu 控[6] 制器的定制化镜像。本文为该控制器开发了剩余带宽、已用带宽、链路时延的测量以及定制化流表的下发

功能。租户可以进行网络的链路带宽、时延，选取一条最优的数据传输链路，下发定制化流表，从而实现数据传输的可控性。提高带宽利用率的同时，数据传输速率得到了很大的改善。

第三、云平台前端可视化操作：本文为实现租户的便捷操作，在 OpenStack 云平台上开放了可视化界面操作。其一是用来显示物理数据中心网络，并在数据中心网络上进行租户虚拟网络的创建操作，以及数据中心网络链路时延的显示工作。其二用于实现租户虚拟网络的拓扑展示，虚拟网络带宽、时延的展示，以及定制化链路的选取和流表的下发操作。使用户切身感受到操作的便捷性、对网络可控性。

1.2.2 创新点

针对现有云平台数据中心的不足，本文主要实现了 SDN 与云平台的结合，将 SDN 的集中控制、网络的虚拟化集成到云平台中，实现了租户网络隔离的同时，租户可以对网络实现自由控制。相应的创新点主要涉及以下几个方面：

1. 对于云平台运营商来说，物理网络的集中控制，可以有效的减小网络配置的繁琐性，为数据中心规模的伸缩性提供了便利，对网络的状况可以实现实时监控，同时大大加速了网络的故障修复速度，通过控制侧的日志输出，以及错误模拟，实现快速故障排除。

2. 将虚拟化技术集成到云数据中心，为租户提供相互隔离的 vSDN 网络，该网络由租户自己的控制器实现集中控制。集中控制的实现，可以让租户指定灵活的数据包转发路径，以及灵活的转发策略。对于自身网络的变化也可以实现即时的应对措施。从而对现有的网络资源进行最有效的利用，减少了网络资源的浪费。租户之间的隔离性，保证了租户数据传输的安全性。

3. 基于包对技术，完成了虚拟网络的链路带宽测量，实现了细粒度的剩余带宽测量，精确度远远高于以前的模式；基于数据包统计的方法，实现了粗粒度的已用带宽的测量；完成了 SDN 架构下的时延测量。租户根据带宽、时延实现定制化链路的选取，可以有效的利用剩余带宽，将降低数据传输时延，大大加快了数据中心网络的传输速度。

综上所述，基于 SDN 的云平台多租户虚拟网络定制化方案，对提高云数据中心的带宽利用率，为租户提供可编程的云网络是非常必要的，有助于实现租户网络的定制化操作，同时为运营商进行云数据中心的维护和配置提供了便利。

1.3 研究生期间主要工作

在攻读硕士研究生期间，本文作者认真学习专业相关课程，积极参与实验室的科研工作。认真学习了网络智能研究中心要求的基本课程并取得了满意的成绩，先后参与了实验室多个科研项目。主要的研究方向为智慧云平台的开发工作，在老师和师兄师姐的指导下，对研究课题进行了深入的探索，通过阅读书籍和学术论文夯实理论基础，并进行系统设计、实现和功能测试。主要科研经历和学术成果如下：

参与了国家重点实验室自主课题仪器设备研制类——面向无线业务的网络智能承载验证环境，开发智慧云平台，为租户提供自主可控的 SDN 云网络；在国家重点基础研究发展计划（973 计划）课题——智慧服务机理与理论中，负责新一代网络架构的调研和撰写工作。

以第一作者的身份发表了两篇 EI 检索论文，具体为：通过对多租户虚拟网络定制化机制的研究，发表了学术论文《MVNC: A SDN-based Multi-tenant Virtual Network Customization Mechanism in Cloud Data Center》，在论文中提出了一种 MVNC 的多租户虚拟网络定制化机制，论文已发表于 IEEE。通过对粒子群算法（Particle Swarm Optimization，PSO）算法及网路虚拟化的研究，撰写了学术论文《A PSO-Based Virtual Network Customization for Multi-tenant Cloud Services》，在论文中，提出了一种基于 PSO 算法的网络虚拟化方案，实验证明相对于传统的基于最短路径的虚拟化算法，性能得到很大提升。同时，在此基础上，完成了多租户云服务中虚拟网络的定制化。论文已被录用，即将发表于 ACM。

1.4 论文组织结构

论文总共包括五章，具体内容组织如下：

第一章主要概述了研究背景、意义以及研究的主要内容，包括 SDN、OpenStack 的背景介绍，本文的主要研究内容以及基于 SDN 云平台多租户虚拟网络定制化的意义、创新点。

第二章主要介绍了 SDN、网络虚拟化及 OpenStack 的相关技术。首先介绍了 OpenFlow 协议,OpenFlow 交换机的组成以及 OpenFlow 流表的字段和匹配过程。对于网络虚拟化技术, 由于本文需要实现 SDN 模式下的网络虚拟化, 所以仅对适用于本文的 SDN 模式下的网络虚拟化技术做了简单的概括。随后阐述了 OpenStack 的模块构成, 而对于本文用到的 Neutron 模块, 进行了详细的架构剖析。最后对本章内容做了

总结。

第三章

第四章

第五章

1.5 本章小结

本章对课题的研究背景进行了描述，阐述了 SDN、OpenStack、网络虚拟化等相关技术，并在此基础上确定了研究内容，对研究生期间的主要工作做了简单的概括，最后对论文的组织结构进行了介绍。

第二章　相关技术概述

SDN 作为新一代的网络架构，具有动态配置、可编程及快速响应的特点，通过解耦控制平面与数据转发平面，实现了对网络底层基础设施的抽象。OpenStack 是当下最流行的开源云平台，为用户提供计算、存储、网络服务。本章对 SDN、OpenFlow、OpenStack 等相关技术做了详细的介绍。

2.1　SDN、OpenFlow 概述

2.1.1　SDN 简介

SDN 是一种时下热门的网络架构，在这种网络架构中，网络的控制与转发解耦，具有很强的可编程行和可扩展性。传统模式下，网络的控制权与网络设备紧密捆绑，现有模式下，控制权的迁移使得底层构架能够抽象出来，各种应用和网络服务因此能将网络当作一个逻辑或虚拟实体。通过应用 SDN，网络的设计和操作变得简化，网络设备也得到简化，这些设备无需理解或处理成千上万的协议，只需要接受 SDN 控制器的指令即可。利用控制集中，网络管理员可以实时改变网络的行为，并且在几小时或几天内就可以部署新的应用和网络服务。

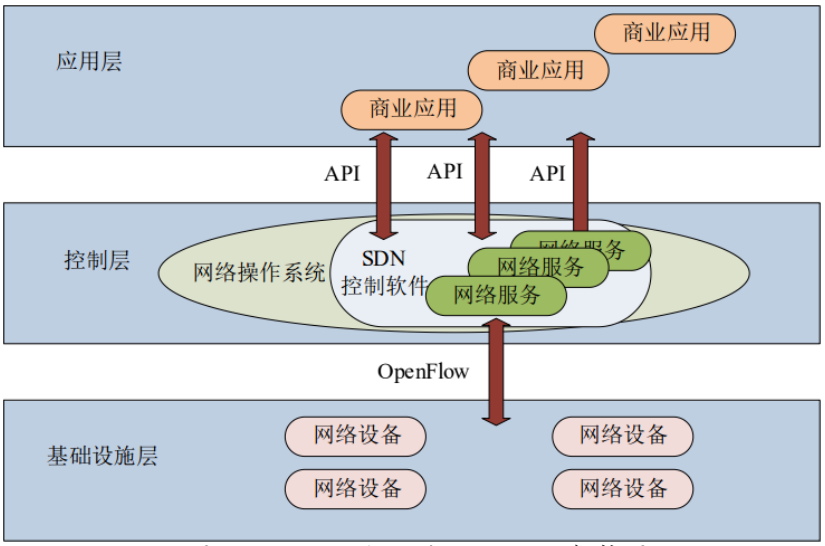


图 2-1　ONF 定义的 SDN 三层架构图

传统网络设备紧耦合的构架在 SDN 体系中被拆分成应用、控制、转发三层分离

的，可编程的开放构架。控制功能被转移到服务器（软件），上层应用、底层转发设施被抽象成多个逻辑实体。图2-1为 ONF 定义的 SDN 三层架构示意图。

应用层为网络各种应用需求，如移动视频、云存储、企业应用商店、桌面云、物联网等，通过北向接口灵活、可编程地调用控制层提供的网络抽象模型与业务功能。北向接口因为涉及业务较多，开放的标准化过程还处于研究阶段。

控制层为整个 SDN 构架的核心，也称为网络操作系统，可实现网络拓扑的集中控制和设备管理，进行流表的控制和下发。其主要功能包括路由优化、网络虚拟化、质量监控、拓扑管理、设备管理、接口适配等。

基础设施层包括标准化的网络设备和虚拟的网络设备，负责多级流表处理和高性能的数据转发，并作为硬件资源池，为控制层提供按需的网络拓扑、数据处理和数据转发。目前主流的网络设备和芯片产商已经提供了支持 OpenFlow 的网络设备。

南向接口定义了控制层与数据转发层（基础设施层）之间的交互协议，通过将转发过程抽象为流表，控制器可直接控制流表、屏蔽硬件，实现了网络虚拟化。物理硬件被淡化为资源池，可按需进行灵活分配和相互隔离。目前主流的控制层与转发层之间交互的协议为 OpenFlow 协议。

2.1.2 OpenFlow 协议

OpenFlow 协议是 OpenFlow 交换机和控制器之间交互信息的标准，也是 OpenFlow 交换机和控制器之间的接口标准。OpenFlow 协议主要支持三种类型的消息：Controller-to-Switch 消息，Asynchronous 消息和 Symmetric 消息。每种类型消息都有多个子类型。Controller-to-Switch 消息由控制器发起直接管理和监视交换机状态，可能需要交换机做出响应；Asynchronous 消息由交换机发起用于更新交换机的状态和控制器的网络事件；Symmetric 消息可由控制器和交换机任一方发起。具体协议消息的类型及描述如下：

1. Controller-to-Switch 消息

   - Features：在建立传输层安全会话（Transport Layer Security Session）的时候，控制器发送 feature 请求消息给交换机，交换机需要应答自身支持的功能。一般在 OpenFlow 通道建立时使用。
   - Configuration：控制器设置或查询交换机上的配置信息。交换机仅需要应答查询消息。

- Modify-state：控制器管理交换机流表项和端口状态等。
- Read-state：控制器发送该消息来管理交换机状态。主要用于增加、删除或改变组/流表项和设置交换机端口属性。
- Packet-out：控制器通过交换机指定端口发出网包。控制器按指定交换机端口发送数据包, 并且转发经由 Packet-in 消息接收到的包。Packet-out 消息必须包含完整包或一个指向存储在交换机中的完整包的缓冲区 ID。该消息包含一系列按指定顺序执行的行为, 若为空则丢弃该包。
- Barrier：控制器确保消息依赖满足，或接收完成操作的通知

2. Asynchronous 消息

- Packet-in：交换机收到一个网包，在流表中没有匹配项，则发送 Packet-in 消息给控制器。如果交换机缓存足够多，网包被临时放在缓存中，网包的部分内容（默认 128 字节）和在交换机缓存中的的序号也一同发给控制器；如果交换机缓存不足以存储网包，则将整个网包作为消息的附带内容发给控制器。
- Flow-removed：交换机中的流表项因为超时或修改等原因被删除掉，会触发 Flow-removed 消息，通知控制器删除流表。
- Port-status：交换机端口状态发生变化时（例如 down 掉），触发 Port-status 消息。
- Error：交换机发生问题时触发消息。

3. Symmetric 消息

- Hello：交换机和控制器用来建立连接。
- Echo：交换机和控制器均可以向对方发出 Echo 消息，接收者则需要回复 Echo reply。该消息用来测量延迟、带宽、以及是否连接正常等。
- Vendor：交换机提供额外的附加信息功能。为未来版本预留。

OpenFlow 协议的主要交互过程如下所示：

1. 控制器与交换机的连接建立: 通过安全通道建立连接，所有流量都不经过交换机流表检查。当连接建立起来后，两边必须先发送 OFPT_HELLO 消息给对方，该

消息携带支持的最高协议版本号，接收方将采用双方都支持的最低协议版本进行通信。一旦发现共同支持的协议版本，则连接建立，否则发送 OFPT_ERROR 消息，描述失败原因，并中断连接。

2. 控制器与交换机的连接中断：当连接发生异常时，交换机应尝试连接备份的控制器。当多次尝试均失败后，交换机将进入紧急模式，并重置所有的 TCP 连接。此时，所有包将匹配指定的紧急模式表项，其他所有正常表项将从流表中删除。此外，当交换机刚启动时，默认进入紧急模式。

3. 连接加密：安全通道采用传输层安全（Transport Layer Security，TLS）连接加密。当交换机启动时，尝试连接到控制器的 TCP 6633 端口。双方通过交换证书进行认证。因此，每个交换机至少需配置两个证书，一个是用来认证控制器，一个用来向控制器发出认证。

4. 流表项修改：该交互是核心交互过程，通过控制器下发的流表项修改指令完成。每条指令可能触发一系列 OpenFlow 协议消息，主要包括流表的增加、修改和删除。

5. 流表项移除：流表项的移出包括控制器主动模式和被动模式，控制器被动模式下，定时器计时结束：每个表项均有一个 idle_timeout 定时器和一个 hard_timeout 定时器（两者的计量单位都是秒），前者计算的是没有 Flow 匹配发生的时间，而后者则计算的是表项在流表中的总时间。一旦到达时间期限，交换机将自动删除该表项，同时发出一个流删除的消息；控制器主动模式下，控制器通过下发 DELETE_STRICT、DELETE 等指令相关的协议消息主动删除流表项

2.1.3 OpenFlow 交换机

OpenFlow 交换机负责数据转发功能，主要技术细节由流表 (flow table)、安全信道 (secure channel) 组成[7]，如图2-2所示。

OpenFlow 的转发策略主要保存在流表中，每个流表中都有很多表项 (FlowEntry), 这些表项可由外部控制器通过 OpenFlow 协议来写入、更新和删除。对于 OpenFlow v1.3, 每一个流表项都由匹配域、优先级、计数器、指令、超时以及 Cookie 组成，如表2-1所示。当一个流进入交换机后，交换机会选择匹配的流表中优先级最高的一个执行相应的指令并更新计数器、超时等项。

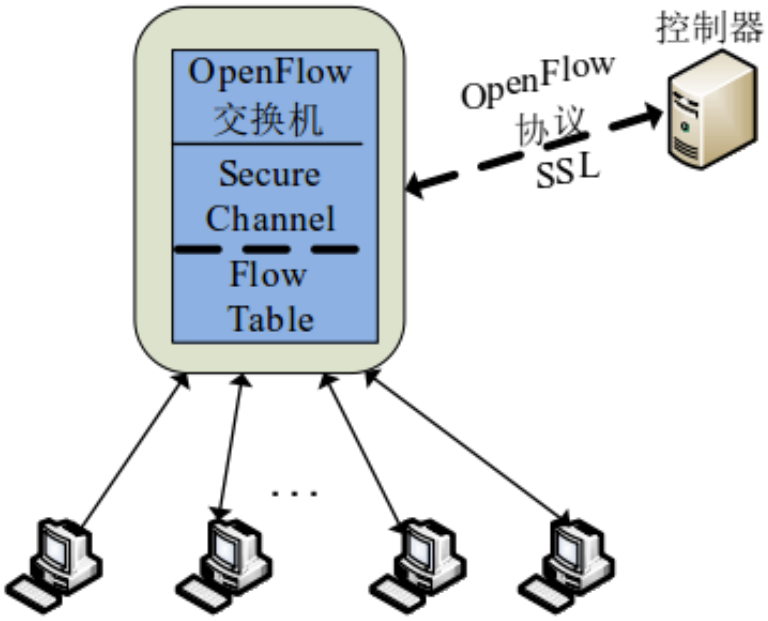


图 2-2 OpenFlow 交换机结构

表 2-1 OpenFlow 流表项结构

匹配域 (Match Field)	优先级 (Priority)	计数器 (Counters)	指令 (Instructions)	超时 (Timeouts)	Cookie

匹配域用于进行对流的匹配。包括 13 个必须支持的域 (Required)：入端口、Ethernet 目的地址、源地址、类型、IP 协议号、IPv4 源地址、目的地址、IPv6 源地址、目的地址、TCP 源地址、目的地址、UDP 源地扯、目的地址。每一个域包含一个确定的值或者任意值 (Any，表示任何值都能匹配)，更准确的匹配可以用位掩码实现，但是每个域对掩码的支持与否不同。

优先级用于在流匹配到多个表项时的选择策略，这里优先级最高的被优先匹配。

计数器用于维护每个流表、每个流、每个队列等的统计数据，例如活动表项、包查找次数、包匹配次数、发送包数、接收字节数等。指令执行于匹配到的包，每个表项包括一系列指令，这些指令可能会直接对包进行改变，也可能改变行动集，或者管道处理。其中行动集是在包被转发前按照指定顺序执行的一系列行动，管道处理是包在各流表之间进行传递时，包与流表之间的互相作用关系。

超时机制用于交换机对于超时表项的自动删除，进而提高交换机的空间利用率。包括硬超时 (指超过该值即删除，绝对时间) 和软超时 (指超过该值也没有匹配的流即删除，相对时间)。

匹配的具体过程如图2-3所示，具体流程如下。

1. 数据匹配字段从数据包中提取。用于表查找的数据包匹配字段依赖与数据包类型，这些类型通常包括各种数据包的报头字段，如以太网源地址或 IPv4 目的地址。除了通过数据包报头中进行匹配，也可以通过入口端口和元数据字段进行匹配。元数据可以用来在一个交换机的不同表里面传递信息。报文匹配字段表示报文的当前状态，如果在前一个表中使用 Apply-Actions 改变了数据包的报头，那么这些变化也会在数据包匹配字段中反映。

2. 数据包匹配字段中的值用于查找匹配的流表项。如果流表项字段具有该值，它就可以匹配包头中的所有可能的值。如果交换机支持任意的的位掩码对特定的匹配字段，这些掩码可以更精确地进行匹配。优先级最高的流表项首先被匹配上，此时与选择流表项相关的计数器也会被更新，选定流表项的指令集也被执行。如果有多个匹配的流表项具有相同的最高的优先级的，所选择的流表项被确定为未定义表项。

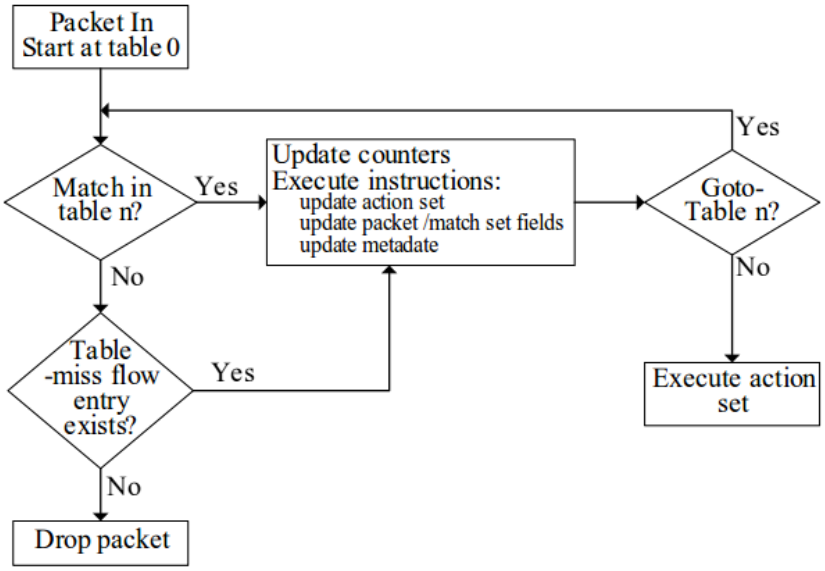


图 2-3　数据包匹配流程图

安全通道是连接 OpenFlow 交换机和控制器的接口，控制器通过这个接口，按照 OpenFlow 协议规定的格式来配置和管理 OpenFlow 交换机，同时接收交换机传来的事件信息并发送数据包等。所有的 OpenFlow 通道信息必须用 OpenFlow 协议打包，OpenFlow 通道通常用 TLS 来加密，但也能直接用 TCP 协议来实现。

目前，基于软件实现的 OpenFlow 交换机主要有两个版本[8]，都部署于 Linux 系统：基于用户空间软件 OpenFlow 交换机操作简单，便于修改，但性能较差；基于内核空间的软件 OpenFlow 交换机[9] 速度较快，同时提供了虚拟化功能，使得每个虚拟机能够通过多个虚拟网卡传输流量，但实际的修改和操作过程较复杂。另外，斯坦福大学基于 NetFPGA 实现了硬件加速的线速 OpenFlow 交换机[10]，而网络硬件厂商如 NEC、HP 等公司也已相继推出了支持 OpenFlow 标准的硬件交换机[7]。

2.1.4 OpenFlow 控制器

控制器是 OpenFlow 网络的核心部分，是整个网络的大脑，网络中所有的控制指令，数据流的转发操作都由它来完成，控制器中组件的好坏直接影响了整个网络的运行效率。

控制器负责控制其所管辖的 OpenFlow 交换机中的流表，包括流表内容的添加、修改以及删除等基本操作。操作的具体逻辑由控制器上运行的控制程序制定。网络使用者可以根据具体需求在控制器上编写控制程序，使得 OpenFlow 网络可以提供丰富的应用，体现了 OpenFlow 网络的灵活性。在数据流处理的过程中，控制器的控制程序决定其所在网络中交换机上的流表内容。网路初始运行时，交换机上的流表为空，控制器必须具有全局网络视图，如网络的拓扑结构、网络的当前运行状态等信息，才能针对不同数据流做出正确的决策。因此，控制器的设计是 OpenFlow 网络整体功能和效率的重点之一。

本文选取 Ryu 控制器进行二次开发，Ryu 控制器是由日本最大的电信服务提供商 NTT 主导开发的基于 Python 语言的开源控制器, 它提供了丰富的 API 接口。Ryu 是一个基于组件的软件定义网络架构，是一个基于 Apache 协议的完全开源的 SDN 控制器，对于软件组件提供了良好的 API 接口，便于开发人员创建新的网络管理与控制中的应用，Ryu 支持管理网络设备的多种网络协议，诸如：OpenFlow,Netconf,OF-config 等，Ryu 架构和主要组件如图2-4所示。OF-conf、Netconf 等组件主要提供了对 OpenFlow 交换机的控制能力，Topology 用于对 SDN 网络的拓扑进行管理,REST 提供了上层的 API 接口。

2.2 网络虚拟化技术概述

互联网不断地向前发展, 必然会出现越来越多的应用和服务。然而现有的网络结构比较僵化, 而且可扩展性很差, 随着网络的发展, 这些缺点日益突出, 但是研发和部

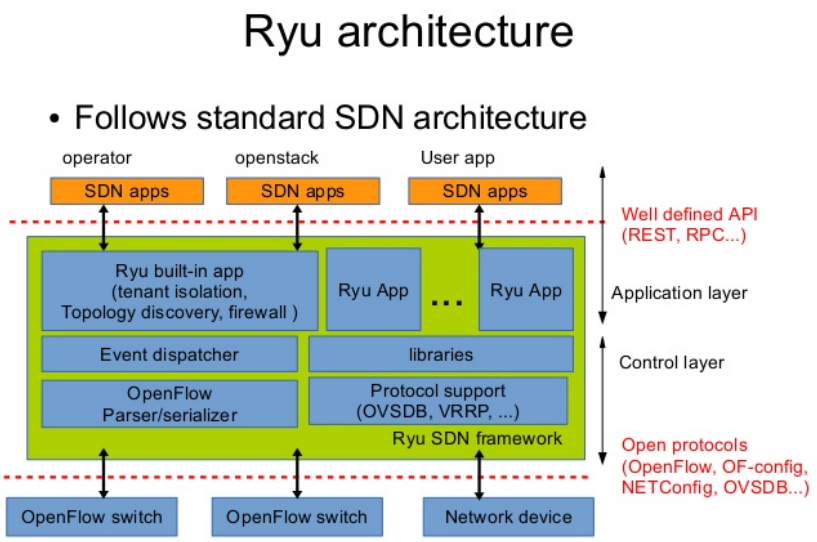


图 2-4　Ryu 架构图

署新型的网络体系架构比较困难, 难以更新, 而且由于各个设备生产商自身利益的考虑, 更加加剧了这种困难。在这种背景下, 网络虚拟化成为当前互联网面向未来网络体系架构过渡的一种有效解决方案, 同时网络虚拟化也是未来网络体系架构应该具备的关键特性之一。

网络虚拟化技术[2]，通过对共用的底层网络进行抽象并提供统一的可编程接口，将多个相互隔离且具有不同拓扑的虚拟网络映射到共用的基础设施上，为用户提供差异化服务。网络虚拟化允许多个租户占据相同的网络基础设施，每个租户会有一种错觉，认为可以对整个网络进行完整的处理. 传统的网络虚拟化配置和操作都比较复杂, 难以管理。原因就在于路由器和交换机变得越来越复杂，其配置操作的代价越来越大。而 SDN 的出现为网络虚拟化开启了另外一条道路，在 SDN 网络中，集中控制的优势使得对网络的管理变得灵活和高效。基于 SDN 的网络虚拟化是 SDN 思想的典型应用。本文选用 OVX 实现网络虚拟化模块，OVX 通过给每个租户提供访问一个虚拟网络拓扑和一个完整的网络头空间来达到这样的目标，前者允许租户自定义拓扑结构, 而后者保证功能性和流量隔离，即使是在不同的租户选择重叠的寻址方案。OVX 用作控制信道内的一个代理，呈现 OpenFlow 网络给租户，同时经由南向的 OpenFlow 接口控制底层的物理基础设施。图2-5描述了此结构。通过暴露 OpenFlow 网络给租户，OVX 允许租户使用自己的网络控制器控制自有的虚拟网络资源。换句话说，OVX 基于同一底层物理网络创建多个 vSDN 网络，并且 vSDN 网络之间是相

互隔离的，保证了租户虚拟网络的安全性。这样的方法会产生两个后果:，允许 OVX 呈现支持 OpenFlow 的可编程虚拟网络，租户可以使用自己的 NOS 控制虚拟网络，使 OVX 是透明的——从底层网络的角度，OVX 的表现为一个控制器，从租户的角度上看，OVX 作为网络中具有 OpenFlow 能力的交换机集合[11]。

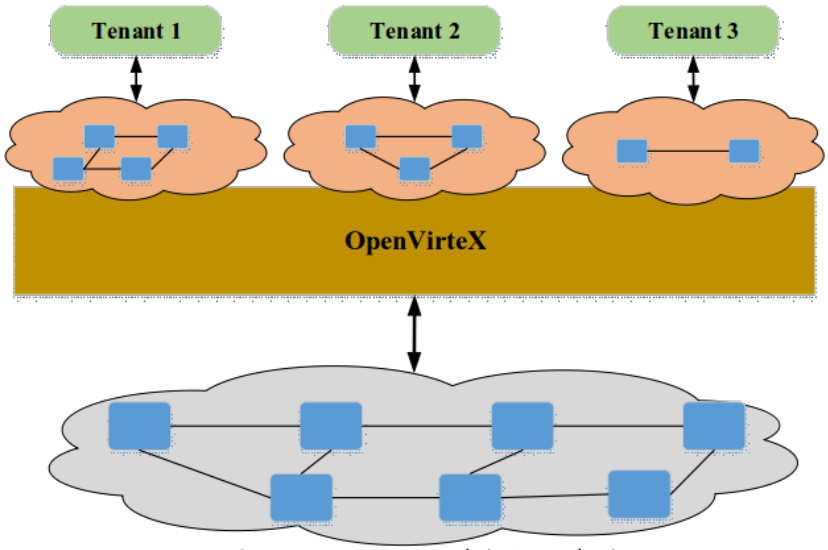


图 2-5 OVX 网络虚拟化示意图

2.3 OpenStack 概述

OpenStack 是一个由 NASA（美国国家航空航天局）和 Rackspace 合作研发并发起的，以 Apache 许可证授权的自由软件和开放源代码项目，主要为企业或者厂商提供了一种云服务的解决方案。OpenStack 既可以被企业使用，在企业局域网内给员工提供私有云；也可以被厂商使用，进行二次开发，给云用户提供通过 Internet 访问的公有云服务。OpenStack 通过一系列相关的服务提供 IaaS 解决方案。每个服务提供一个应用编程的 API，这样有利于集成。管理员可以根据自己的需求安装部分或者全部服务。

2.3.1 OpenStack 架构

OpenStack Juno 版本的构架由 9 个服务组成，分别是 Dashboard、Compute、Networking、Object Storage、Block Storage、Identity、Image Registry and Delivery Service、Telemetry 和 Orchestration。分别对应的各组件及各组件之间的协同关系如图2-6所示。

主要组件的功能介绍如下[12]。

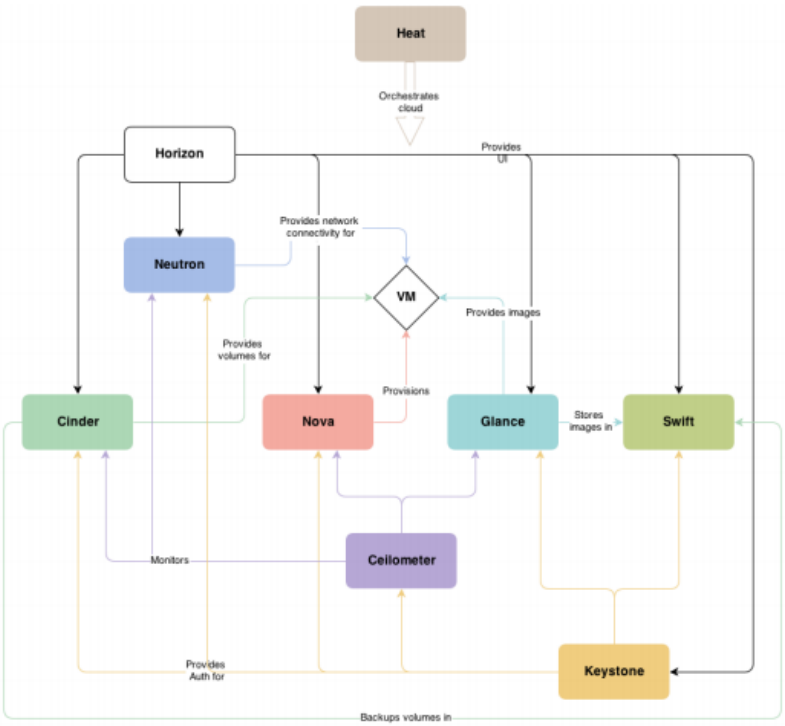


图 2-6　OpenStack 概念架构图

1. Nova: 计算服务是 OpenStack 的核心服务，它由 nova-compute 模块通过 libvirt、XenAPI 等管理 hypervisor，从而管理虚拟机，此外它还通过 nova-api 服务向外提供如 EC2 兼容、管控功能等的接口，通过 nova-scheduler 模块提供虚拟机调研逻辑等，这些模块间的通信全部通过消息队列完成

2. Swift: 对象存储服务是 OpenStack 最早期的两个服务之一（另一个是计算服务），在 OpenStack 平台中，任何的数据都是一个对象，swift-proxy 模块对外提供如 HTTP(S)、OpenStack Object API 及与 S3 兼容的存取接口。对象的存取经 swift-proxy 接入后，还需要经三个模块进行定位，即 account、container、object，这是因为在 OpenStack 中一个对象被描述为：某个帐户下某个容器中的某个对象。

3. Neutron: 经过一定时间的演变,网络管理也抽离成一个独立的服务。在 OpenStack 的网络管理流程中，通常需要经过以下几个步骤：创建一个网络；创建一个子网；启动一个虚拟机，将一块网卡对接到指定的网络上；删除虚拟机；删除网络端口；删除网络。

4. Glance:Glance 的出现是解决虚拟机镜像的管理问题，在生成一个镜像后，需要

将镜像注册到系统的数据库中。当要实例化一个虚拟机时，需要将镜像分派到一台具体的实机上用来以启动虚拟机。因而 Glance 最重要的接口是镜像的注册和分派。

5. Cinder:Essex 将 nova 的卷管理 api 独立化后，Folsom 终于将卷管理服务抽离成了 Cinder，Cinder 管理所有的块存储设备，块设备可以挂接在虚拟机的实例中，然后虚拟机里的 guest 系统可以像操作本地卷一样操作块存储设备。Cinder 需要处理的主要问题应该是接入各种块设备，如本地磁盘、LVM 或各大厂商提供的设备如 EMC、NetApp、HP、华为，还有如 Vmware 提供的虚拟块设备等。

6. KeyStone: 身份服务需要进行认证凭证的验证及关于用户、角色等的信息，及所有相关的元数据，这些数据全都由 Keystone 服务管理，并提供 CRUD 的操作方法，另外这些信息可以从另一个认证服务中获取，例如使用 LDAP 做 Keystone 的后端。

7. Heat:Heat 是一套业务流程平台，旨在帮助用户更轻松地配置以 OpenStack 为基础的云体系。利用 Heat 应用程序，开发人员能够在程序中使用模板以实现资源的自动化部署。Heat 能够启动应用、创建虚拟机并自动处理整个流程。

8. Ceilometer:Ceilometer 项目创建时最初的目的是实现一个能为计费系统采集数据的框架。社区后来更新了他们的目标，新目标是希望 Ceilometer 成为 OpenStack 里数据采集（监控数据、计费数据）的唯一基础设施，采集到的数据提供给监控、计费、面板等项目使用。

9. Horizon:Horizon 是一个用以管理、控制 OpenStack 服务的 Web 控制面板，它可以管理实例、镜像、创建密匙对，对实例添加卷、操作 Swift 容器等。除此之外，用户还可以在控制面板中使用终端（console）或 VNC 直接访问实例。

2.3.2 Neutron 模块详解

本文研究的主要内容，涉及最多的是 OpenStack 的 Neutron 模块，故本节对 Neutron 模块做详细的介绍，方便大家对论文的理解。

Neutron 是 OpenStack 项目中负责提供网络服务的组件，它基于软件定义网络的思想，实现了网络虚拟化下的资源管理。在实现上充分利用了 Linux 系统上的各种网

络相关的技术。一般的，OpenStack 中网络实现包括 VLAN、GRE、VXLAN 等模式，本文中选取 GRE 模式搭建 Neutron 服务，GRE 模式下的 Neutron 架构图如图2-7

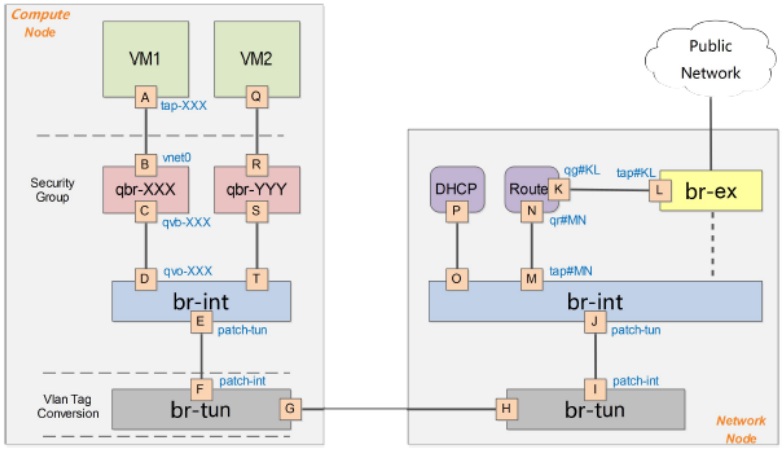


图 2-7　Neutron 组件 GRE 模式架构图

在 OpenStack 中，所有网络有关的逻辑管理均在网络节点中实现，例如 DNS、DHCP 以及路由等。计算节点上只需要对所部属的虚拟机提供基本的网络功能支持，包括隔离不同租户的虚拟机和进行一些基本的安全策略管理（即安全组）。以抽象系统架构图为例，计算节点上包括两台虚拟机 VM1 和 VM2，分别经过一个网桥（如 qbr-XXX）连接到 br-int 网桥上。br-int 网桥再经过 br-tun 网桥（物理网络是 GRE 实现）连接到物理主机外部网络。网络节点上配置有 DHCP、Router 服务，对于访问外网的数据包，经由 br-ex 发出。各网桥功能及具体的服务如下所示。

1. qbr: 在 VM1 中，虚拟机的网卡实际上连接到了物理机的一个 TAP 设（即 A 常见名称如 tap-XXX）上，A 则进一步通过 VETH pair（A-B）连接到网桥 qbr-XXX 的端口 vnet0（端口 B）上，之后再通过 VETH pair（C-D）连到 br-int 网桥上。一般 C 的名字格式为 qvb-XXX，而 D 的名字格式为 qvo-XXX。注意它们的名称除了前缀外，后面的 id 都是一样的，表示位于同一个虚拟机网络到物理机网络的连接上。之所以 TAP 设备 A 没有直接连接到网桥 br-int 上，是因为 OpenStack 需要通过 iptables 实现安全组的功能。目前 openvswitch 并不支持应用 iptables 规则的 Tap 设备。因为 qbr 的存在主要是为了辅助 iptables 来实现 security group 功能，有时候也被称为安全网桥。

2. br-int:br-int 是一个 openvswitch 网桥。br-int 在 GRE 模式中作为一个 NORMAL 交换机使用，因此有效规则只有一条正常转发。如果两个在同一主机上的 vm

属于同一个 tenant 的（同一个 vlan tag），则它们之间的通信只需要经过 br-int 即可。

3. br-tun:br-tun 将带有 vlan tag 的 vm 跟外部通信的流量转换到对应的 gre 隧道，这上面要实现主要的转换逻辑，规则要复杂，一般通过多张表来实现。这些规则组成的整体转发逻辑如图2-8所示。

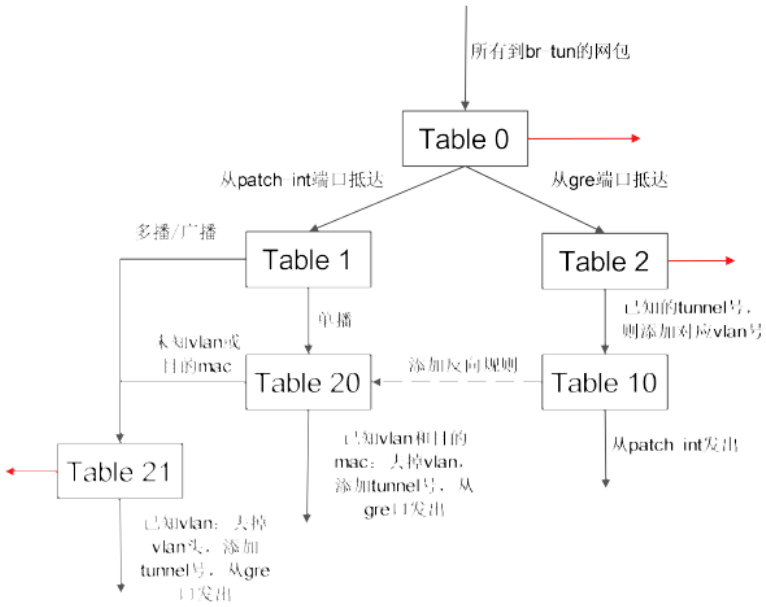


图 2-8 br-tun 网桥转发规则

4. br-ex:br-ex 直接连接到外部物理网络，一般情况下网关在物理网络中已经存在，则直接转发即可。

5. dhcp 服务:dhcp 服务是通过 dnsmasq 进程（轻量级服务器，可以提供 dns、dhcp、tftp 等服务）来实现的，该进程绑定到 dhcp 名字空间中的 br-int 的接口上。可以查看相关的进程。

6. router 服务: 首先，要理解什么是 router，router 是提供跨 subnet 的互联功能的。比如用户的内部网络中主机想要访问外部互联网的地址，就需要 router 来转发（因此，所有跟外部网络的流量都必须经过 router）。目前 router 的实现是通过 iptables 进行的。同样的，router 服务也运行在自己的名字空间中。防火墙服务就是在 router 命名空间中实现。

2.4 本章小结

本章主要介绍了 SDN、网络虚拟化及 OpenStack 的相关技术。重点介绍了 OpenFlow 协议，OpenFlow 交换机的组成以及 OpenFlow 流表的字段和匹配过程。对于网络虚拟化技术，由于本文需要实现 SDN 模式下的网络虚拟化，所以仅对适用于本文的 SDN 模式下的网络虚拟化技术做了简单的概括。随后介绍了 OpenStack 的模块构成，而对于本文用到的 Neutron 模块，进行了详细的架构剖析。本研究的系统，主要在以上各部分的基础上，进行了集成开发。后续会对整体的系统架构做详细的介绍。

第三章 系统架构

附录 A 不定型（0/0）极限的计算

定理 A.1 （L'Hospital 法则 ） 若

1. 当 $x \to a$ 时，函数 $f(x)$ 和 $g(x)$ 都趋于零；
2. 在点 a 某去心邻域内，$f'(x)$ 和 $g'(x)$ 都存在，且 $g'(x) \neq 0$；
3. $\lim\limits_{x\to a}\dfrac{f'(x)}{g'(x)}$ 存在（或为无穷大），

那么

$$\lim_{x\to a}\frac{f(x)}{g(x)}=\lim_{x\to a}\frac{f'(x)}{g'(x)}. \tag{A-1}$$

证明： 以下只证明两函数 $f(x)$ 和 $g(x)$ 在 $x=a$ 为光滑函数的情形。由于 $f(a)=g(a)=0$，原极限可以重写为

$$\lim_{x\to a}\frac{f(x)-f(a)}{g(x)-g(a)}.$$

对分子分母同时除以 $(x-a)$，得到

$$\lim_{x\to a}\frac{\dfrac{f(x)-f(a)}{x-a}}{\dfrac{g(x)-g(a)}{x-a}}=\frac{\lim\limits_{x\to a}\dfrac{f(x)-f(a)}{x-a}}{\lim\limits_{x\to a}\dfrac{g(x)-g(a)}{x-a}}.$$

分子分母各得一差商极限，即函数 $f(x)$ 和 $g(x)$ 分别在 $x=a$ 处的导数

$$\lim_{x\to a}\frac{f(x)}{g(x)}=\frac{f'(a)}{g'(a)}.$$

由光滑函数的导函数必为一光滑函数，故 (A-1) 得证。 □

附录 B 缩略语表

IaaS Infrastructure as a Service，基础设施即服务

OVX OpenVirteX，网络虚拟化平台

PSO Particle Swarm Optimization，粒子群算法

SDN Software Defined Network，软件定义网络

TLS Transport Layer Security，传输层安全

vSDN virtual Software Defined Network，虚拟 SDN 网络

参考文献

[1] Porras P, Shin S, Yegneswaran V. A security enforcement kernel for Open Flow networks[A]. // The 1st Workshop on Hot Topics in Software Defined Networks[C]. New York: ACM Press, 2012: 121–126.

[2] Wang A, Iyer M, Dutta R, et al. Network Virtualization: Technologies, Perspectives, and Frontiers[J]. Journal of Lightwave Technology, 2013, 31(4): 523–537.

[3] Hua-YingChang, Shie-YuanWang. Using SDN technology to mitigate congestion in the OpenStack data center network[A]. // (ICC)[C]. IEEE, 2015: 401–406.

[4] Feller E, Rilling L, Morin C, et al. A scalable and autonomic virtual machine management framework for private clouds[A]. // Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster[C]. IEEE Computer Society, 2012: 482–489.

[5] OpenVirteX web site[EB/OL]. http://ovx.onlab.us/.

[6] Ryu web site[EB/OL]. https://osrg.github.io/Ryu/.

[7] 左青云, 陈鸣. 基于 OpenFlow 的 SDN 技术研究 [J]. 软件学报, 2013, 24(5): 1081–1083.

[8] OpenFlow setup website[EB/OL]. [4]http://www.openflow.org/wp/deploy-labsetup.

[9] Pfaff B, Pettit J, Koponen T, et al. Extending networking into the virtualization layer[A]. // Proc. of the 7th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)[C]. New York: ACM Press, 2009.

[10] Naous J, Erickson D, Covington G, et al. Implementing an OpenFlow switch on the NetFPGA[A]. // Proc. Of the 4th ACM/IEEE Symp. on Architectures for Networking and Communications Systems (ANCS)[C]. San Jose: ACM Press, 2008: 1–9.

[11] Al-Shabibi A, Leenheer M, Gerola M, et al. OpenVirteX: A Network Hypervisor[A]. // Open Networking Summit 2014[C]. 2014.

[12] OpenStack web site[EB/OL]. http://docs.openstack.org/.

[13] 北京邮电大学研究生院培养与学位办公室. 关于研究生学位论文格式的统一要求 [EB/OL]. http://www.bupt.edu.cn/, 2014–11.

[14] 北京邮电大学研究生院培养与学位办公室. 关于研究生学位论文格式的统一要求 [EB/OL]. http://www.bupt.edu.cn/, 2004.

致　谢

感谢 Donald Ervin Knuth.

```
def create():
    pass
class Test():
    def __init__():
        pass
```

脚注使用带圈数字的表示方法，此处为示例 1① 和示例 2②。参考文献可以使用[13] 和 [14] 的表示方法。

① 测试脚注一
② 测试脚注二

攻读学位期间发表的学术论文目录

期刊论文

[1] **Zhang San** , Newton I, Hawking S W, et al. An extended brief history of time[J]. Journal of Galaxy, 2079, 1234(4): 567–890. （SCI 收录，检索号：786FZ）.

会议论文

[2] McClane J, McClane L, Gennero H, et al. Transcript in Die hard[A]. // Proc. HDDD 100th Super Technology Conference (STC 2046)[C]. Eta Cygni, Cygnus: 2046: 123–456. （EI 源刊）.

专利

[3] 张三 , 李四. 一种进行时空旅行的装置 [P]. 中国: 1234567, 2046–01–09.