硕 士 学 位 论 文

基于 OpenFlow 的数据中心网络路由策略研究与设计

Research and Design on Routing Strategy of Data Center Based on OpenFlow

作 者 姓 名：王珣

学科、专业：计算机应用技术

学　　号：21209175

指 导 教 师：李克秋 教授

完 成 日 期：2015.4.30

大连理工大学

Dalian University of Technology

大连理工大学学位论文独创性声明

作者郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用内容和致谢的地方外，本论文不包含其他个人或集体已经发表的研究成果，也不包含其他已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

若有不实之处，本人愿意承担相关法律责任。

学位论文题目：基于OpenFlow的数据中心网络路由策略研究与设计

作 者 签 名：王珣 日期：2015 年 6 月 9 日

摘　要

互联网经过数十载的发展，已经成为人们生活中的不可或缺的组成部分。近几年云计算技术的发展与大数据概念的提出，各种新兴的互联网应用不断涌现，数据中心在互联网中占据了举足轻重的地位。如今的互联网包括数据中心面临诸多问题，原有的体系架构已经无法与新兴的网络服务和应用相适应。SDN 体系架构和 OpenFlow 技术的出现，将传统网络的控制层与转发层分离，实现了对网络的可编程控制，为解决数据中心网络乃至互联网面临的问题提供了一个新的思路。

本文以 OpenFlow 技术为主要手段，主要研究数据中心网络中的路由方法，以解决数据中心网络中的负载问题。现有的数据中心网络的路由方法，主要采取静态的路由方式，缺少来自于网络的反馈信息，从而导致网络带宽资源分配不合理，不能够充分利用数据中心网络提供的大量的冗余的带宽资源，造成了资源的浪费。

本文首先提出一种基于流分类的动态的数据中心网络路由策略。本文提出的策略使用 OpenFlow 技术，实时监控并收集网络中的链路信息和数据流信息，作为计算路由的参考依据。对进入网络中的新数据流，利用网络的链路信息，采取贪心策略为其进行路由，以达到负载均衡为目的。之后，通过收集网络中数据流的信息将网络中转发的流区分为大小流，并依据大流的转发状态和链路的状态对大流的路由进行动态调整，使大流尽量获得更多的带宽，提高链路的利用率，减小拥塞发生的概率。

其次，本文提出了一种基于 OpenFlow 的拥塞路由管理策略。通过 OpenFlow 实时监控网络中链路的使用状态。当发现一条发生拥塞的链路后，启动拥塞路由管理策略对拥塞链路上的大流进行重路由，为其选择更适合的路由。在进行重路由时，从拥塞的链路起点采取回溯的方式进行，从而减小重路由的开销。这种拥塞路由管理策略在拥塞发生后开始实施，直到拥塞状态结束，以一种被动的方式来缓解网络中出现的拥塞情况，是对本文提出的基于流分类的动态路由策略的补充。

最后本文使用基于 OpenFlow 的 Mininet 仿真实验平台，与已有的数据中心网络路由方法进行实验并对比分析，证明本文所提出的策略能够提高网络的链路利用率，提高网络的吞吐量。

关键词： OpenFlow；数据中心；Fat-Tree；路由策略

Research and Design on Routing Strategy of Data Center Based on OpenFlow

Abstract

After decades of development, Internet has become an indispensable part of our life. In recent years, with the development of cloud computing technology and the proposed concept of Big Data, a lot of new types of Internet applications have emerged. Therefore, the data center occupies a pivotal position in the Internet. Today, the Internet, including the data centers, have face many problems. Traditional architecture of the Internet has been unable to satisfy the new requirement of the new emerging network services and applications. The appearance of SDN architecture and the OpenFlow technology enables the network programmability by decoupling the control plane and the forwarding plane of the network and provides a new way of solving the problems faced by data centers or even by the Internet.

In this paper, we use OpenFlow as our main technology, and focus on the routing strategies used in the data center network, to solve the traffic load problem. Most of the existing routing strategies used in the data center network are static ways, which lack of the feedback information from the network, resulting in irrational distribution of bandwidth resources in the network and not being able to take full advantages of the redundant bandwidth resources provided by the data center, making a waste of resources.

Firstly, we propose a flow classification-based dynamic routing strategy for data center network. The strategy monitors and collects the links' and flows' information in the real time by using the OpenFlow technology. These collected information is used as reference to calculate the route for the flows in the network. For the new coming flows in the network, we take the greedy principle to choose their route with the link information in order to achieve load balancing purpose. Then, by using the collected flow information, we classify the flows being forwarded in the network into small flows and big flows, and reroute for the big flows dynamically to make the big flows getting more bandwidth, improving the utilization of the links and reducing the probability of congestion.

Secondly, we also propose a congested routing management policy based on the OpenFlow technology. By real-time monitoring of the status of links, when a congested link detected, the congested routing management policy starts to work and reroute the big flows on this link to find better routes for them. When rerouting these flows, we take the way back from the starting point of the congested link, thereby reducing the cost of rerouting. This congested routing management policy is triggered after the congestion occurs and works until

the congestion state ends. It is a passive way to relieve the congestion occurred in the network. The congested routing management policy is the complementary for the flow classification-based dynamic routing strategy.

Finally, our implement uses Mininet simulation platform. Compared with existing routing methods, the evaluation results demonstrates that our strategies are superior over other policies which can improve the link utilization and the throughput of the network.

Key Words: OpenFlow; Datacenter; Fat-Tree; Routing Policy

目　录

1 绪论

1.1 课题研究背景

互联网从诞生到今天，已经极大地改变了世界，改变了人类的发展，它已经发展成为社会生活与生产活动离不开的一项重要基础设施。随着计算机网络的发展，尤其是互联网技术的不断革新，计算机网络已由大量不同类型的网络设备构建而成，这其中包括路由器、交换机以及其他各种不同功能的中间设备，伴随而来的是在这些设备上所实现的各种各样复杂的功能，导致现在的网络设备在功能上越来越臃肿。

近几年来，云计算概念的出现以及云服务的兴起，使得全球各个大型软件服务提供商都将服务放在了“云”上，为此云服务提供商们建立了由成百上千台甚至更多的服务器组成的数据中心。每个数据中心中这些数量庞大的服务器通过各种网络互联设备连接在一起，构成了一个个数据中心网络。当各种不同类型的应用与服务运行在数据中心内部时，会在数据中心内部的节点之间产生大量的数据传输，这些数据传输考验着一个数据中心网络的传输能力，而传输能力又直接决定着所提供的应用与服务的质量。如何在有限的网络资源下充分利用现有的数据中心网络资源，是每个数据中心所面临又亟待解决的问题。

最近几年，软件定义网络 SDN 技术的出现与发展，为解决数据中心网络的问题，提供了一个新的思路。SDN 作为下一代互联网体系架构之一，提出将传统网络设备中的控制平面与数据平面分离，并进一步加强控制平面的控制能力，使得单一的软件控制程序能够控制多个数据平面元件。SDN 控制平面通过特定的编程接口直接对数据平面元件（比如路由器、交换机和其他中间设备）的状态进行控制。当前，OpenFlow[1]作为这类接口中表现最突出的一个，逐步演变成一个标准化的协议，开始得到更加广泛的应用。这种数据平面与控制平面相分离的解耦方式，打破了传统的网络设备的专有性和封闭性。对网络的管理者来说，网络设备不再是一个“黑盒子”，而是可以通过一个个控制程序，更灵活的控制它们。

SDN 具有可编程性与灵活性，这也是数据中心对网络资源的管理过程中所需要的。数据中心网络资源的利用，体现在数据中心网络拓扑架构的设计和对数据中心网络中流量的路由策略上。OpenFlow 采取集中控制原则，通过 OpenFlow 可以以全网视图信息为依据，对网络中的流量实现以流为单位的细粒度调度。Google 公司已将 OpenFlow 技术引入它的多数据中心组成的广域网 B4[2]中，用于数据中心间的流量工程。将下一代互联

网体系架构 SDN 及其代表性技术 OpenFlow 与数据中心网络架构相结合，是对 SDN 和数据中心流量工程两者的新的探索，在学术领域与应用领域都具有积极的意义。

1.2 课题研究现状

1.2.1 互联网体系架构研究现状

互联网的发展使当今世界变成了一个数字化的社会，几乎一切信息都是相连的，可以随时随地访问到它们。现如今，互联网主要以 TCP/IP 体系架构作为统一的标准。传统的 IP 网络虽然得到了广泛的采用，但是随着互联网业务与应用的增多，各种网络中间设备的出现和运行在设备上的各种控制程序，使得传统 IP 网络愈加的复杂与难以管理。现有的 TCP/IP 体系架构已无法满足当前大数据环境下的大规模数据传输，传统的 IP 网络与互联网的新兴业务之间出现了相互不适应的表现。为此，业界开始致力于下一代互联网架构的研究工作，以打破现有 IP 网络对互联网发展的阻碍。

目前，世界上各国已经陆续加入到下一代互联网体系架构的研究工作，各国也开展了不同的下一代互联网研究项目，比如美国的 GENI[3]，FIND[4]计划、欧盟的 FIRE 计划、日本的 AKARI[5]以及中国下一代互联网计划 CNGI 计划[6]。

回顾二十年前网络的实现形式，再与今天的网络相比较，很明显网络已经走过了漫长的道路。网络在速度、可靠性、安全性和普遍性等方面都得到了极大的发展。虽然在物理层技术上，网络已经可以提供高容量的链路，网络设备的计算能力也得到显著提升，并且涌现出一大批出色的网络应用，但是网络的架构上还没有看到与成立之初发生太大的变化。在现有的网络基础设施中，例如路由或者网络访问决策这些组成全部网络功能的复杂任务，都委托给各种不同厂商的网络设备来实现，这些不同厂商之间的设备都运行着不同的固件。这种由专有设备所组成的整个网络并没有为创新的研究思路带来太多的空间，比如在真实网络中大规模的测试新的路由协议。本质上的一个主要原因是网络的基础设施一直保持着静态、僵化的特点，并且在这个方向上没有重大的突破，直到 SDN 和 OpenFlow 的出现。

2008 年，美国斯坦福大学 Nick McKeown 教授领导的研究组提出了 OpenFlow 技术。该研究组在文[1]中详细阐述了 OpenFlow 技术的原理与实现方法，并由此开始推广软件定义网络（Software-Defined Networking，SDN）体系架构。OpenFlow 作为 SDN 的具体实现，其核心思想为对传统的网络设备的控制平面与转发平面进行解耦，采用集中控制的方式对网络进行控制与管理[7]。OpenFlow 隶属于美国科学基金会（NSF）在美国斯坦福大学主导的 Clean Slate 项目[8]，该项目旨在打破传统网络的僵化，设计一种全新的网络体系架构，以满足未来 15 年人们对互联网的需求。

OpenFlow 的出现是为了解决当今网络存在的严重缺陷，从而振兴网络的科研工作。OpenFlow 利用了现代以太网交换机和路由器中的查找表技术。这些流表运行在不同的厂商设备之间实现防火墙、网络地址转换（NAT）、服务质量或者收集统计数据。OpenFlow 小组已经确定了一组通用的功能，这些功能得到了多数的交换机和路由器的支持。通过标识这个通用的功能集，一个操作流表的标准方法可以被部署在所有的网络设备上，而不用考虑特定厂商的实施方法。OpenFlow 提供了一个操作流表的标准方法，这个方法允许对网络流量基于流进行划分。通过这种方式，网络流量被组织成各种不同的流，这些流可以被分组和被分离，以便对其进行路由、处理或者以任何期望的方式进行控制。

OpenFlow 可以在校园网中大量使用，其中将科研与生产流量的隔离是一个关键的操作。流通过一个集中式的控制器来创建和保持。控制器可以扩展实现其他诸如路由和网络访问决策等功能。通过移除分散在网络中的网络设备的控制功能，并将这些功能完全或者局部集中后，便可以更容易地控制和改变网络。为了实现这种的要求是网络的交换机要能够支持 OpenFlow 和一个具备网络逻辑的集中式控制器。这样一来，控制与数据平面不再共置与一台单一的网络设备之上，而是相互分离并且动态地连接彼此。控制平面与数据平面功能上的分离，以及采用集中式控制的网络模型已经经历过先前研究者们的讨论和实践。研究者们曾提出 ForCES[9]和 SoftRouter[10]等架构，用以实现网络设备的控制平面与数据平面功能解耦，旨在提供更加有效的分组转发和更加灵活的控制功能。OpenFlow 与这些架构存在许多共同点，但是 OpenFlow 增加了流的概念，以及在交换机中加入流表。

对新网络架构的创新与研究不仅仅是 SDN 与 OpenFlow。近几年又诞生了网络功能虚拟化 NFV[11]（Network Functions Virtualization）的概念。NFV 是一种全新的网络架构概念。NFV 提出利用已有的 IT 虚拟化技术来虚拟化网络节点的功能为相互连接的组成部分，从而建立新的通信服务。例如，可以通过部署一个虚拟化的会话边界控制器功能而无需考虑传统的成本开销和获取和安装物理元件带来的复杂性。其他实例还包括虚拟化的负载均衡器、防火墙、入侵检测设备和 WAN 加速器等。NFV 与 SDN 有着密切的联系。二者互有交集，NFV 可以通过 SDN 来实现，而 NFV 又不全部属于 SDN。但是 SDN 的特点对实现 NFV 具有极大的优势。因此，很多厂商的平台将 SDN 与 NFV 同时纳入在平台的生态系统中，共同发挥作用。

1.2.2 数据中心网络的研究现状

伴随着互联网的疾速发展，近几年云计算与大数据成为了互联网领域最火热的技术名词与概念。越来越多的本地服务与应用被服务提供商搬到了“云”上。接入到互联网的

企业与个人，每时每刻都享受着“云服务”给工作与生活带来的高效与便捷。这些“云服务”包括搜索、电子邮件、多媒体内容分发、文件共享和电子商务等等涵盖各种行业领域。为了提供这些云端服务与应用，各大云服务提供商构建了大型的数据中心来支撑这些业务。每个数据中心都由数以千计的服务器组成，这些服务器通过数据中心的网络相连接。为了提供给用户更好的服务与应用体验，数据中心需要高性能的网络来满足集群服务器之间的通信与信息传输。从应用托管到科学计算，再到网页搜索和 MapReduce[12]，都需要大量的集群内带宽。随着数据中心和运行在其中应用的数量和规模的持续增长，为潜在的所有通信扩展网络架构的性能是一项特殊的挑战。

基于云的服务和应用的很多特性，增加了数据中心网络的设计困难。首先，数据中心网络的流量负载对于网络的设计者来说是未知的，并且会随着时间和空间的变化而变化。从而导致静态的资源分配方式不足以达到高性能网络的目标。其次，网络必须能够分配足够的带宽而不需要软件或者协议的变化，使得应用能够运行在普通的操作系统之上。再者，互联网的云服务提供商普遍使用虚拟化技术来有效地为客户提供跨物理主机的硬件资源，这也导致客户不能够保证他们的虚拟机实例运行在同一个物理机架之中。失去了这种物理的局域性，客户的应用将面临来自传统数据中心网络拓扑中机架间的网络性能瓶颈。

上述所面临的困难并不能归结于应用本身。部署在数据中心网络之中的路由和转发协议按照非常特定的部署设定所设计而成。现如今，数据中心网络的设计依赖于路径的多样性，从而能够达到对主机的横向扩展。一些数据中心应用经常发起不同范围之间主机的连接，并需要显著的聚合带宽的提高。鉴于最高端的商用交换机有限的端口数量，数据中心网络的拓扑通常采用多根树的拓扑架构以及高速的链路，但是减小了聚合带宽[13]。这类多根树的拓扑架构使得每队主机之间存在多条路径。同时并且动态地在这些路径上转发多个流，并且最小化或减小链路的超额利用，从而为这些数据流分配可接受的聚合带宽是一个重要的挑战。

为了提高数据中心网络的性能，让网络的资源得到充分的利用，当前研究者们主要从两个方面对数据中心网络进行研究。第一方面，研究者从数据中心网络的拓扑架构入手，研究并设计出新型数据中心网络拓扑架构。第二方面，研究者从数据中心网络的协议和路由算法等软件方面进行研究，在已有的网络拓扑下改善网络的传输性能和资源利用等。

在数据中心网络拓扑架构研究方面，针对数据中心网络遵循以下几点最基本的设计目标。首先，网络的基础设施必须是可扩展到支持连接大量的服务器，并允许增量扩展；其次，针对网络中可能出现的服务器宕机，链路中断，服务器机柜故障等情况，数据中

心网络要具备容错性；最后，数据中心网络必须能够提供高性能的网络质量，从而更好的支持需求带宽的服务应用。

由于基于树的结构越来越难以满足数据中心的设计目标，近几年研究者们提出了一些创新的数据中心网络架构。这些新兴的数据中心网络拓扑可以粗略分为两大类[14]，一类以交换机为中心，由交换机来控制完成网络的连接与数据的路由等网络功能，这类拓扑包括 Fat-Tree[15]，VL2[16]等拓扑架构；另外一类以服务器为中心，交换机只提供基本的交换功能，连接和路由等网络控制功能主要由接入网络中的服务器来完成，这类拓扑结构包括 DCell[17]，BCube[18]和 FiConn[19]等。

在数据中心网络协议方面，研究者们在原有网络协议的基础上，不断进行研究并提出了大量适用于数据中心网络的协议，如 DCTCP[20]、MPTCP[21]等。除此之外，研究者们还对数据中心网络中的流量的特点进行了研究，对数据中心网络中流量的管理提供了极有价值的参考依据。

现有的网络路由协议为每一对源地址和目的地址选择一条没有故障的路径。这种静态的单路径转发很大程度上没有充分利用多根树拓扑提供的多路径条件。当前应用在大型企业和数据中心环境下的转发方法是 ECMP[22]（Equal Cost Multipath，等代价多路径），该方法通过对每个数据流进行哈希计算，静态地将这些流下发到可用的路径上。

1.3 本文的主要工作

数据中心网络的拓扑架构和路由策略是决定着数据中心网络资源利用的两个重要因素。以 Fat-Tree 为代表的多根树分层拓扑架构因为能够提供等价的冗余路径和更高的聚合带宽，而得到了广泛的应用。在此基础上，本文通过引入 SDN 的 OpenFlow 技术，对数据中心网络的数据流量进行集中控制，在全局网络视图下实现对数据中心网络中以流为单位的细粒度调度。本文的主要工作归纳如下：

（1）研究传统互联网体系存在的不足，并着重阐述数据中心网络在传统 IP 网络体系架构下所面临的问题。介绍下一代互联网体系架构的特点与研究现状，重点说明新一代网络架构 SDN 及其关键技术 OpenFlow 的发展现状和核心思路。

（2）在（1）的基础上，详细研究 SDN 的关键技术 OpenFlow。主要针对 SDN 的架构进行概述；之后详细介绍 OpenFlow 的技术，包括 OpenFlow 的协议内容、OpenFlow 交换机的特性、OpenFlow 控制器的工作机制这几方面。

（3）通过研究当下以 Fat-Tree 为代表的多根树型拓扑结构的数据中心网络的特点，并与 SDN 体系架构相结合，提出基于 OpenFlow 的数据中心网络数据流转发策略与拥塞路由管理策略。

（4）实验方面利用 Mininet[23]仿真平台构建 4 元 Fat-Tree 拓扑的 OpenFlow 网络，编写基于 Pox 控制器的数据流的转发策略和拥塞路由管理模块，从链路利用率、流量负载等方面验证上述策略的正确性与可行性，并对实验结果进行分析。

1.4 论文的组织结构

本篇论文正文共分为五个章节，具体组织结构与各章节内容概述如下：

第一章：绪论。介绍本文的研究背景与研究的重要意义，说明了本文的主要工作。

第二章：相关技术背景。首先对 SDN 体系架构及其核心技术 OpenFlow 进行详细研究与阐述，其次对数据中心网络的拓扑架构和路由技术进行深入的介绍。

第三章：基于流分类的 Fat-Tree 数据中心网络数据流的路由策略。提出在 SDN 体系架构下利用 OpenFlow 技术实现基于流分类的动态数据流路由策略。

第四章：基于 OpenFlow 的数据中心网络拥塞管理策略。提出在网络出现拥塞链路时，在 Fat-Tree 拓扑网络下实现基于 OpenFlow 技术的拥塞路由管理策略。

第五章：实验及结果分析。利用 Mininet 仿真平台进行模拟实验，验证上述两种策略正确性与可行性，并对实验的结果进行分析。

2 相关技术研究

2.1 SDN 架构概述与 OpenFlow 技术

2.1.1 SDN 概述

软件定义网络（SDN）是一种将网络的控制平面与转发平面分离，并能够直接可编程的新兴网络架构。将原本与个体网络设备紧密联系的控制平面，迁移至可访问的计算设备使得底层的基础架构在网络应用与服务面前抽象化。上层的网络应用与服务可以把网络视为一个逻辑的或者虚拟的实体。SDN 架构逻辑图如图 2.1 所示。

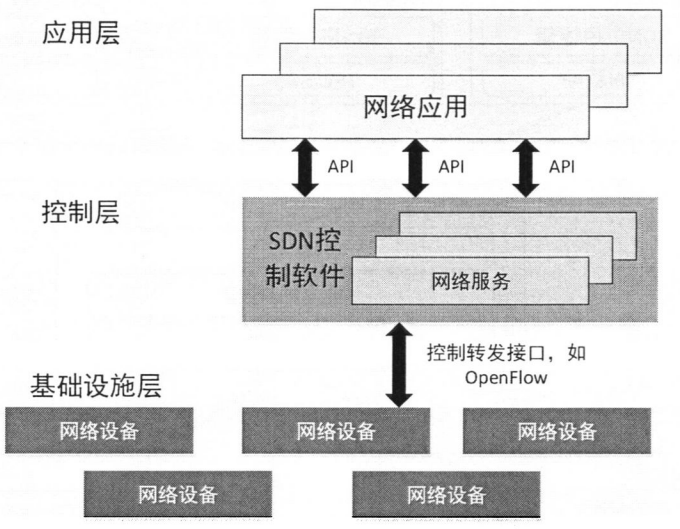

图 2.1 SDN 架构

Fig.2.1 The architecture of SDN

在 SDN 架构中，网络的控制功能集中在基于软件的 SDN 控制器上，同时 SDN 控制器将维护一张网络的全局视图。最终，整个网络在网络应用与策略引擎面前表现为一个单一的逻辑上的交换机。利用 SDN，企业与运营商能够以单一的逻辑角度获得全网络之上并且独立于网络设备厂商的控制权，这在极大程度上简化了网络的设计与操作。SDN 同样能够极大地简化网络设备本身。在 SDN 的架构下，网络设备不再需要理解和处理数以千计的协议标准，它们仅仅需要接受来自于 SDN 控制器的指令。更重要的是，网络的操作与管理者能够以编程的方式配置网络，而不用在分散的数千台网络设备上手动输入成千上万行配置指令。SDN 将网络状态信息集中在控制层，这样使得网络管理者

能够通过动态的、自动化的 SDN 控制程序灵活地配置、管理和优化网络的资源。除了将网络抽象化，SDN 架构支持一系列 API，这些 API 能够实现部署常见的网络服务，比如路由、多播、安全、访问控制、带宽管理、流量工程、QoS 等[24]。因此，SDN 控制层与应用层之间开放 API 的存在，让网络的上层应用能够操作在一个抽象的网络之上，充分利用网络服务和功能，而不局限于其实施细节。最终，能够达到一个计算、存储和网络资源的最优化。

SDN 架构中应用平面、控制平面和数据转发平面分别实现网络的不同功能，因此也分别具备不同的组件模块。图 2.2 所示的是 SDN 架构中每一层具备的不同组件。

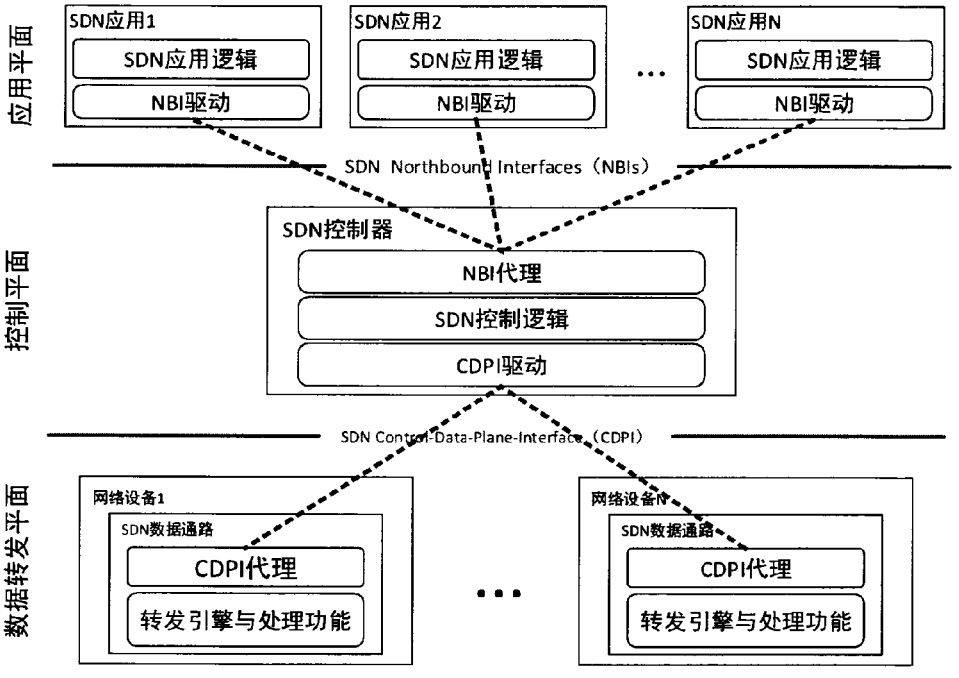

图 2.2　SDN 架构下不同平面的组成组件

Fig.2.2　The components of SDN architecture in different planes

对组件的描述如下：

（1） SDN 应用：SDN 应用是一些通过北向接口向 SDN 控制器明确直接地提出它们的网络需求和期望的网络行为的程序。一个 SDN 应用由 SDN 应用逻辑和北向接口驱动所组成。

（2） SDN 控制器：SDN 控制器是一个逻辑上集中的实体。它负责：

① 将来自 SDN 应用层的请求翻译并向下传送给 SDN 数据通路。

② 向 SDN 应用层提供一个网络的抽象视图，可以包括统计信息和事件。

SDN 控制器的组成包括北向接口代理、SDN 控制逻辑和 CDPI 驱动。

（3） SDN 数据通路：SDN 数据通路是一个逻辑网络设备，它表现出对转发能力和数据处理能力的可见性和控制。这种逻辑的表现可以包含全部的底层物理资源或者它的一个子集。SDN 数据通路包含一个 CDPI 代理、多个流量转发引擎和多个流量处理功能。这些引擎与功能可以包括数据通路外部接口间的简单转发，或者内部数据流量的处理和终止功能。一个或多个数据通路可以被包含在一个单一的物理网络设备中，相当于对多个通信资源的物理集成，从而作为一个单元来管理。一个数据通路也可以被定义为跨越多个物理网络设备。

（4） SDN 控制-数据平面接口（CDPI）：SDN CDPI 是定义在 SDN 控制器和 SDN 数据通路之间的接口，主要提供如下功能：

① 对所有转发操作的可编程控制

② 性能通告

③ 统计信息报告

④ 事件通知

SDN 的一个价值在于对 CDPI 的期望：CDPI 以一种开放的、厂商中立的和客户操作的方式来实现。

（5） SDN 北向接口（NBI）：SDN 北向接口是部署在 SDN 应用和 SDN 控制器之间接口，它通常提供抽象的网络视图，使得网络的行为和要求得到直接的表达。这些可以发生在任何水平上的抽象或者发生在不同的功能集之间。SDN 期望这些北向接口同样能够以一种开放、厂商中立和客户操作的方式来实现。

2.1.2 OpenFlow 技术研究

OpenFlow 是 SDN 架构下控制平面与转发平面之间第一个标准化的通信接口。OpenFlow 允许直接访问与操作网络设备的转发平面，例如交换机或者路由器，无论它们是物理的还是基于虚拟化的。传统网络中，由于转发平面开放接口的缺失，导致今天的网络设备变成了庞大的、臃肿的、封闭的、类似于大型机似的机器。所以，需要将网络的控制功能从网络交换机中分离出来，形成一个逻辑上集中式的控制软件。OpenFlow 协议实现了控制与转发分离，实现了网络控制功能的集中化。

OpenFlow 协议是在网络基础设施的设备和 SDN 控制软件之间的接口的两侧来实现。OpenFlow 使用流的概念来区分网络流量。流的区分基于预先定义的匹配规则并且

匹配规则可以通过 SDN 控制软件静态或者动态地编程实现。基于访问模式、应用类型等因素，网络的管理者可以定义流量应该如何流经各个网络设备。一个基于 OpenFlow 的 SDN 网络架构能够提供以流为单位的细粒度控制，这也使得网络能够回应网络中应用级、用户级和会话级的实时变化。当前的基于 IP 网络的路由技术不能够提供这样水平的控制，因为来自于两个端点的所有的数据流都将沿着相同的路径在网络中进行转发，而忽略了它们个体之间对网络资源的不同需求。

一个最基本的基于 OpenFlow 的 SDN 网络由三部分组成：由支持 OpenFlow 的交换机组成的转发平面；一个集中式的控制器即控制平面；用来实现控制器与交换机之间基于 OpenFlow 协议通信的安全通道[25]。如图 2.3 所示。

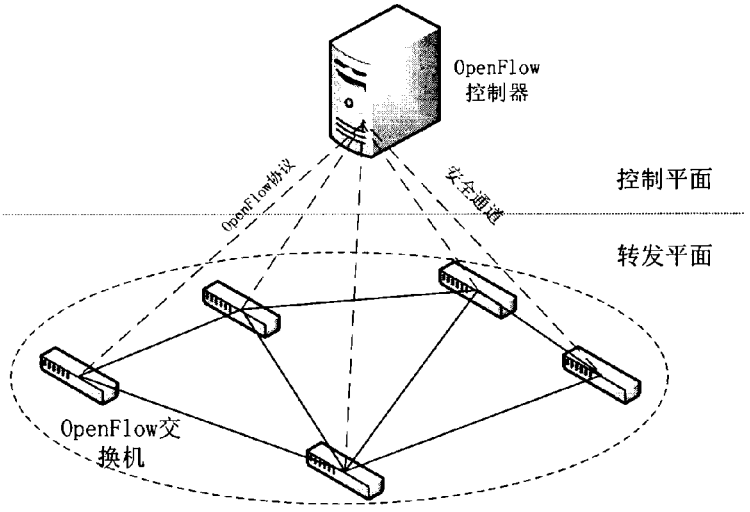

图 2.3　OpenFlow 网络架构

Fig.2.3　The architecture of an OpenFlow network

（1）OpenFlow 交换机。

一台 OpenFlow 交换机由包含流表项的流表和一条与控制器相连的安全通路所组成。流表用来进行包的匹配查找与转发，安全通路用来实现控制器与交换机之间的消息交互。流表包含多个流表项、活动计数器和零个或多个针对已匹配成功数据包的执行动作。所有经过交换机处理后的数据包都会与流表相比较。如果为一个数据包找到了一条匹配的流表项，这条流表项中的执行动作都将执行在这个数据包上；如果没有找到匹配的流表项，则数据包将会通过安全通路被转发到控制器。控制器将会负责决定在没有适

应的流表项时如何处理这个数据包，并通过添加和删除流表项来管理交换机的流表。如图 2.4 所示。

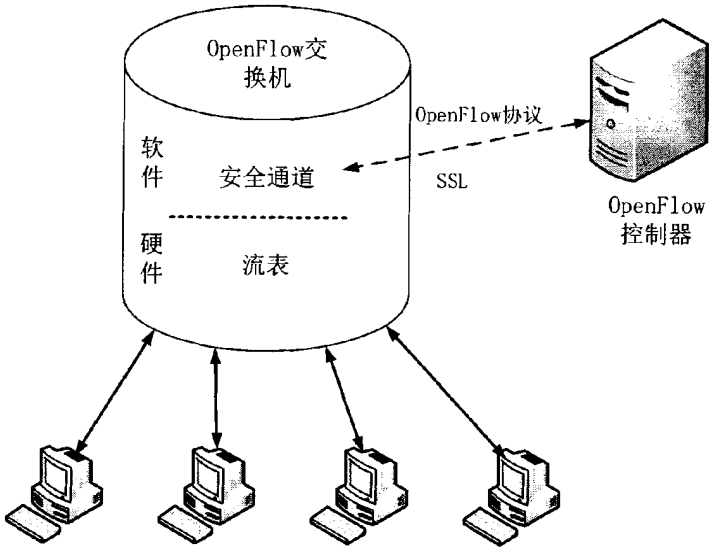

图 2.4　OpenFlow 交换机

Fig.2.4　The architecture of an OpenFlow switch

流表中的每一项都包含三部分，分别是包头字段、计数器和操作。如图 2.5 所示。

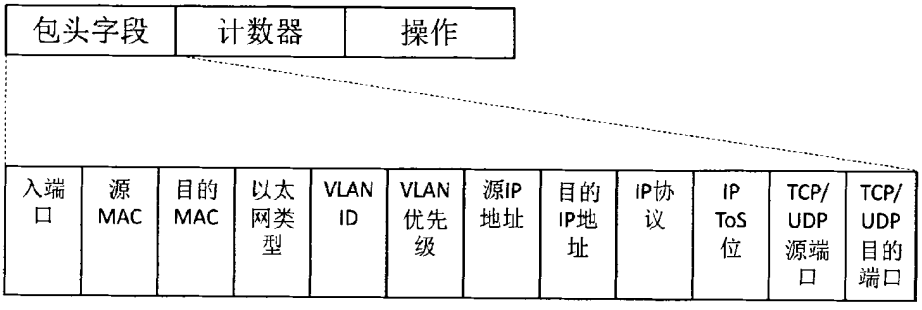

图 2.5　OpenFlow 的流表结构及包头字段的 12 元组

Fig.2.5　The structure of the flow table and the 12-tuple of the header fields

① 包头字段

包头字段用来与数据包进行比较匹配，所要比较的内容是一个十二元组，包括入端口、源 MAC 与目的 MAC、VLAN ID、源 IP 与目的 IP 等。

② 计数器

计数器用来记录并实时更新的网络状态的一些统计信息，这些计数器可以是针对每个流的，每个队列和每个流表的，也可以是针对交换机的每一个物理端口的。兼容 OpenFlow 的计数器可以利用软件实现，并在更有限的范围内通过轮询物理计数器进行保持。

③ 操作集

操作集是针对满足匹配域的数据包所要执行的一组操作，即交换机对满足匹配条件的数据包所要执行的动作。这些动作可以是零个或者多个，其中如果操作集为空，表示交换机将丢弃这个包。所有流表项中的操作集必须按照指定的顺序进行处理。如果一个流表项的操作集在执行时发生逻辑冲突而导致不能够按照指定的顺序进行处理，则这个流表项将会被交换机丢弃。流表项的具体匹配过程如图 2.6、2.7 所示。

操作集可以分为两类：必备的操作和可选的操作。按照交换机对操作集种类支持的不同，可以将交换机分为两类。一类只支持必备的操作集，称为 OpenFlow-only 交换机；另一类除了支持必备动作集还额外支持一些可选操作集，称为 OpenFlow-enabled 交换机。

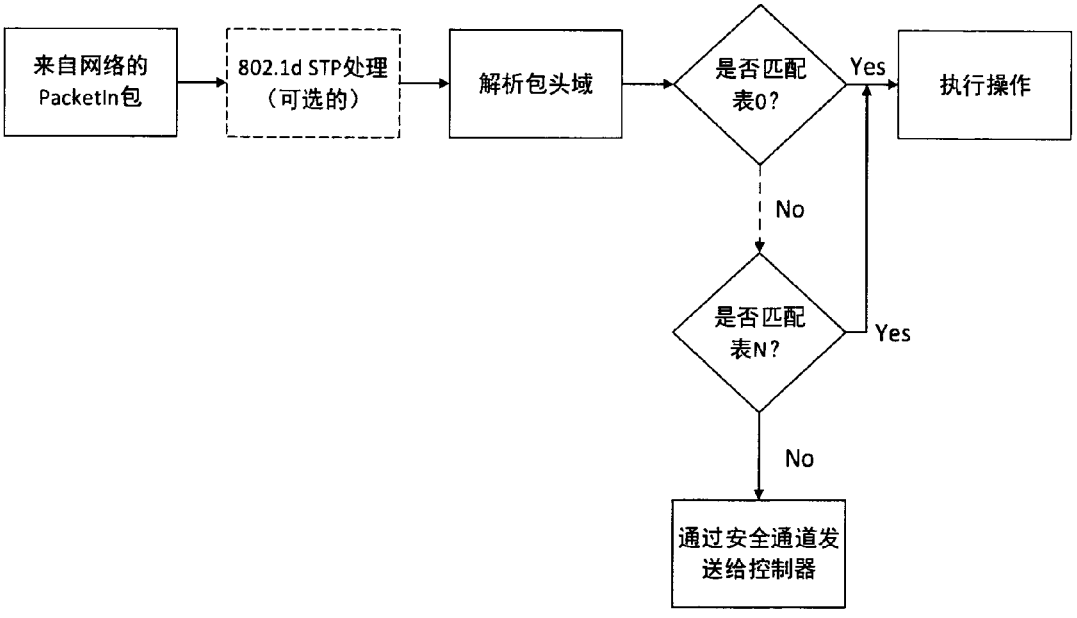

图 2.6　流表项的匹配过程

Fig.2.6　The flow chart of matching a flow entry

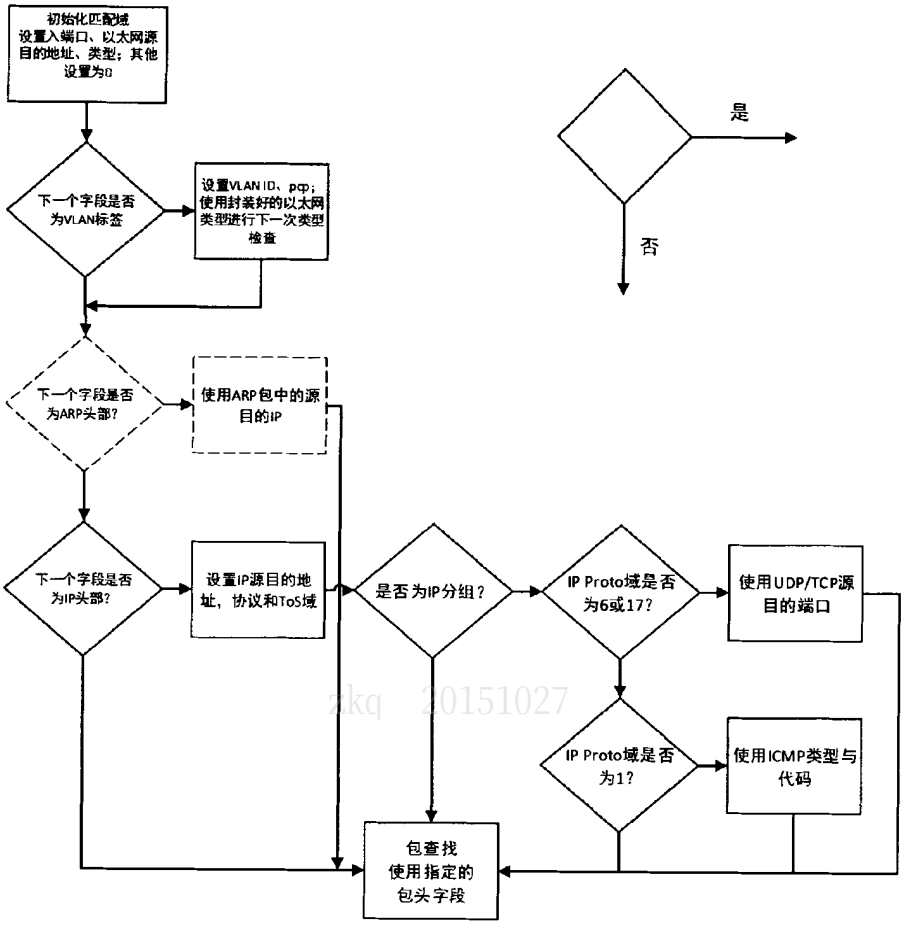

图 2.7　包头域的解析流程图

Fig.2.7　The flow chart of head fields' parsing

（2） OpenFlow 协议

OpenFlow 交换机通过安全通路与 OpenFlow 控制器依据 OpenFlow 协议进行通信。安全通路使用 TLS（Transport Layer Security，安全传输层协议）进行加密连接，默认使用 6633 端口。OpenFlow 协议支持三种消息类型，分别是控制器到交换机的消息、对称消息和异步消息，它们各自都包含多个子类型。

① 控制器到交换机的消息

控制起到交换机的消息由控制器发起，这种类型的消息用来直接管理和检查交换机的状态。例如发送配置消息可实现控制器对交换机配置参数的设置与查询；功能信息查询消息可实现控制器对交换机所支持的功能特点的查询；状态-读/修改消息可实现控制

器管理交换机的状态、添删或修改交换机中的流表向、收集交换机的端口、单个流的统计信息等等；以及包发送消息等。

② 对称消息

对称消息可以由控制器或交换机的任何一方发起而无需发出请求建立连接。如 Hello 消息用来实现交换机与控制器的连接；Echo 消息用来指出交换机与控制器之间的连接带宽、时延等信息。

③ 异步消息

异步消息由交换机发出而无需控制器向交换机发出消息的请求。交换机发送异步消息给控制器来表明一个数据包的到达、交换机状态的改变，或者是发生的错误。例如无匹配流表项时的交换机向控制器发送的 Packet-in 消息；流表项移除消息用来通知交换机流表项因为某些原因（超时、限制、被强制修改）而被删除。初次之外，异步信息还包括端口状态消息和报错消息。

随着 SDN 与 OpenFlow 的火热发展，OpenFlow 协议也在不断的发展与进步。自 2009 年 12 月 31 日推出 1.0.0 版本之后，截至到目前已经发展到最新的 1.5.1 版本，并将继续得到完善与发展。

（3） OpenFlow 控制器

SDN 架构最重要的核心思想就是将传统网络的转发平面与控制平面相分离。控制器集合了控制平面的控制功能，对网络的转发平面进行集中式的控制。对于一个基于 OpenFlow 的 SDN 网络，使用支持 OpenFlow 协议的控制器来控制网络。

OpenFlow 控制器通过安全通路与转发平面的 OpenFlow 交换机进行通信。首先，通过 OpenFlow 协议，控制器能够获得网络的全网视图信息，包括网络的拓扑、网络中链路的状态信息和网络中流的状态。其次，控制器要实现对 OpenFlow 交换机中流表的控制，能够添加、修改、删除流表中的流表项。控制器 OpenFlow 控制器还要能够向上层应用提供必要的接口，使得上层应用能够调用这些接口来控制转发平面，从而实现应用所需要的网络功能。

随着 OpenFlow 的推广，各个公司与组织纷纷推出了各具特点的 OpenFlow 控制器，如 NOX[26]、POX[27]、Floodlight[28]、OpenDaylight[29]、Beacon[30]、Maestro[31]，标志着 OpenFlow 控制器迎来了快速发展的阶段。其中 POX 控制器和 Floodlight 作为所有控制器的两个典型代表，得到了广泛的应用；OpenDaylight 作为后起之秀，但得到大厂商的支持，也得到了迅速的发展。

① POX 控制器

POX 控制器使用 Python 语言开发，是一款跨平台的控制器，可运行在任何支持 Python 的操作系统，可以通过安装 PyPy 运行时完成简单的部署。POX 继承了 Python 语言开发所带来的简单、灵活的特点，与其他语言相比开发实现更加快捷方便。POX 基于模块化设计，可以将数据包交给指定的模块进行处理，用户可根据实际的需要，编写自己的复杂的功能模块。POX 作为第一个 OpenFlow 控制器 NOX 的继任者，较前者具有更好的性能，并且同样支持 GUI；此外，POX 旨在将最初简单的控制器扩展成为控制器平台。POX 可作为一个 OpenFlow 控制器的原型，在此基础上扩展功能，从而实现功能更强更完善的网络控制器。

② Floodlight 控制器

Floodlight 控制器使用 Java 语言开发，是一个跨平台的控制器，它基于一个模块化的加载系统，使得控制器更加易于扩展和加强。Floodlight 还提供一个 Web 图形界面，用户可以通过 Web 图形界面更加直观的管理底层的网络。它还支持 OpenFlow 与传统交换机组成的混合网络。Floodlight 由 Big Switch 公司开发，是 Big Switch 公司旗下商用 OpenFlow 控制器的一个开源版本，Floodlight 由包括 Big Switch 公司的工程师在内的开源社区所维护，并且 Big Switch 的商用控制器完全兼容 Floodlight 的全部功能。这也为将已有的网络控制程序在以后部署在商用环境中带来了方便。

③ OpenDaylight 控制器

OpenDaylight 控制器是实现网络可编程性的一个开放的软件平台，由 OpenDaylight 基金会主导，旨在大规模的网络中实现 SDN 的控制。OpenDaylight 由思科、爱立信、IBM 和微软等公司发起，现如今已经涵盖了几乎所有全球知名的网络设备厂商和软件服务厂商，包括国内的华为、中兴和联想公司也成为了 OpenDaylight 基金会的会员。这也从侧面反映出现如今 SDN 的火热程度以及在工业界获得的重视程度。OpenDaylight 也是一个模块化的系统，除了提供最基本的 SDN 网络功能，如拓扑管理、统计信息管理等以外，还加入了大量其他的功能模块，包括 OpenStack 的服务模块，以及对 NFV 实现的支持等等。可以说，OpenDaylight 是一个大量功能组件的集合体，包括可插拔的控制器、各种接口、协议插件和应用程序。在这个共同的平台上，用户和厂商能够为实现 SDN 的商业化和以 NFV 为基础的解决方案进行合作与创新。

2.2 数据中心网络相关技术

2.2.1 数据中心网络的 Fat-Tree 架构

Fat-Tree 架构是一个以交换机为中心的三层级联的多根树拓扑架构[32]，分别是最上层的核心层，中间的聚合层，以及最下面的边缘层。所有的服务器都与边缘层的交换机

相连。对于一个 k 元 Fat-Tree 拓扑，共有 k 个 Pod，每个 Pod 均由 k 台交换机组成，其中 $k/2$ 为处在中间层的聚合交换机，另外 $k/2$ 为下层的边缘交换机。组成 Fat-Tree 架构的每一台交换机都提供 k 个端口，对于聚合层的每一台交换机，它的 $k/2$ 个端口分别与同一 Pod 内的每一台边缘交换机相连，另外 k/2 个端口分别与最上层的核心交换机相连；同一 Pod 内的聚合交换机的每一个端口都与不同的核心交换机连接，所以整个拓扑有 $k^2/4$ 台核心交换机，总计 $5k^2/4$ 台交换机；每一台边缘交换机的另外 $k/2$ 个端口用来与主机服务器相连，整个拓扑架构可以连接 $k^3/4$ 台服务器。图 2.8 所示的是一个 4 元的 Fat-Tree 拓扑架构。

在 Fat-Tree 拓扑架构下，根据源主机与目的主机所在拓扑内的位置，可以将主机之间的流量分为三类：

（1） 自不同 Pod 的任意两个主机之间的流量都存在$(k/2)^2$条等价路径，并且每一条路径都与一台核心交换机相对应；

（2） 对于同一 Pod 内，任意来自不同边缘交换机的两个主机之间存在 k/2 条等价路径；

（3） 同一边缘交换机下的主机之间，流量直接从对应端口转发，路径唯一。

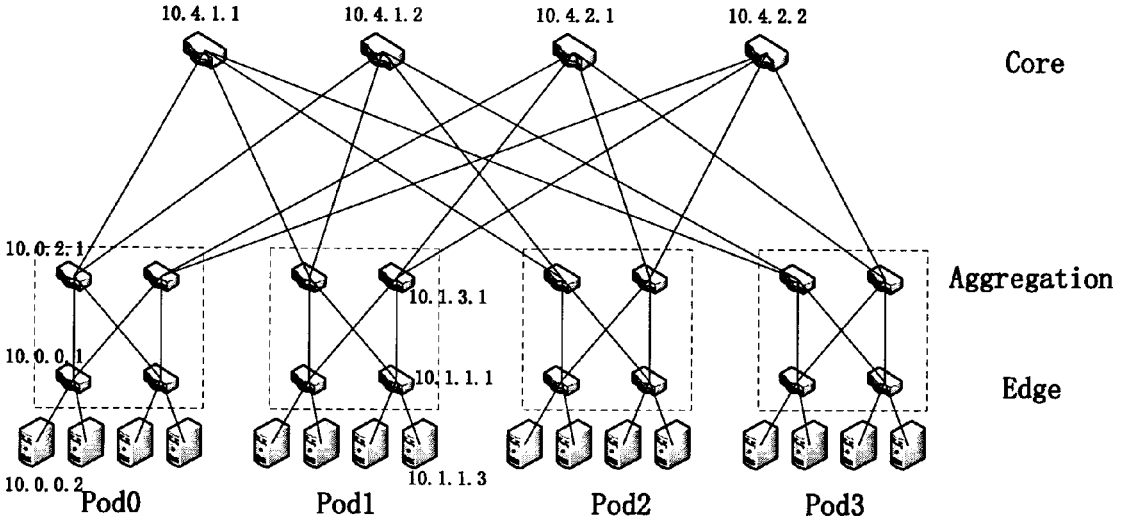

图 2.8 4 元 Fat-Tree 网络拓扑架构

Fig.2.8 The architecture of a 4-ary Fat-Tree network

Fat-Tree 网络拓扑的特点还包括 Fat-Tree 对网络中所有节点，包括交换机与主机的地址分配规则。Fat-Tree 对网络中分配的所有的 IP 地址都在私有的 10.0.0.0/8 地址段。我们在以下条件下遵循点分十进制的地址形式：

（1） 对于处在某一 Pod 内的交换机，所分配的地址形式是10.*pod*.*switch*.1 ，其中 *pod* 表示交换机所在的 Pod 序号（在$[0,k-1]$之间），*switch* 表示交换机在 Pod 内的位置（在$[0,k-1]$之间，顺序按照从左到右，从下到上）。

（2）对于核心交换机，所分配的地址形式是10.*k*.*j*.*i*，其中 j 和 i 表示核心交换机在 $(k/2)^2$ 个核心交换机所组成的矩阵中的坐标（ j,i 都在$[1,(k/2)]$之间）。

（3） 对与连接在 Pod 交换机上的主机的地址分配，主机的得到的地址形式是 10.*pod*.*switch*.*ID*，*ID* 表示主机在每个子网即每个 Pod 下的位置（*ID* 在$[2,k/2+1]$之间，按照从左到右的顺序）。由此可见，每一台边缘层交换机负责一个/24 规模的子网，包括 $k/2$ 台主机。图 2.3 中标注了一个 4 元 Fat-Tree 中部分交换机和主机的 IP 地址。

这种 IP 地址的分配方式为网络拓扑的管理带来巨大的方便。当给定一个节点的 IP 地址后，我们能够通过这个地址定位这个节点所在拓扑的位置，如节点所在的 Pod，所处的层次，这也为网络的路由

Fat-Tree 网络架构的优势在于组成拓扑所有的交换元件都是相同的，这使得数据中心可以在所有的交换机中选择使用便宜的交换机。此外，Fat-Tree 是可重排无阻塞的，这意味着对于任意的通信模式，将会存在一些路径使得到达拓扑中终端主机的可用带宽达到饱和。

2.2.2 数据中心网络的路由技术

现如今，大多数的数据网络拓扑使用生成树技术和 ECMP 技术实现网络的路由功能。在生成树的技术下，所有的流量都流经网络拓扑的一棵生成树，这样会导致很多链路没有被利用。这种方法以牺牲效率为代价，避免了路由中出现环路。在高流量负载的环境下，生成树算法会导致较高的拥塞概率和造成较高的数据包的丢失[33]。

ECMP 通过利用潜在的路径多样性，避免了其中的一些问题。ECMP 在可用的等价路径上随机分配网络中的流。当短流在网络流量中占主导时，ECMP 达到了很好的效果。然而，当网络中出现了多个“大”流时，这种对流的随机分配方式能够导致某些链路上发生了严重的拥塞，而在某些其他链路并没有得到充分的利用。

PROBE[34]（Probe and Reroute Based on ECMP）是一种基于 ECMP 算法的改进算法，PROBE 利用 ECMP 算法实现了基于数据流的重路由并且提供了细粒度的流量控制。它的特点是所有的操作都利用 TCP 协议在终端主机上完成，而不是在交换机上实现的。

VLB[35]（Valiant Load Balancing）同样是一种随机负载均衡路由算法，它的主要思想是过随机选取中间节点的方式进行路由。VLB 路由算法既可以实现面向数据流的应用又可以实现面向数据包的应用。VLB 路由算法主要分两步来分发负载，首先一台源主机通过 VLB 算法随机选取网络中的一个中间“核心”节点，然后将数据流或数据包转发至这个“核心”节点；然后，再从“核心”节点转发到目的节点。当前，为了保证数据包的有序性，面向流的 VLB 算法的实现要比面向数据包的实现广泛的多。并且面向流的 VLB 路由算法与 ECMP 路由算法同等有效。

2.3 本章小结

本章主要从两方面介绍了相关技术的研究。第一方面主要对 SDN 的架构进行了概述，并详细介绍了实现 SDN 架构的 OpenFlow 技术，包括 OpenFlow 交换机、OpenFlow 协议和 OpenFlow 控制器。第二方面主要介绍了数据中心网络的相关的关键技术，阐述了数据中心网络中 Fat-Tree 拓扑结构的特点，以及数据中心网络中多条等价路径条件下的路由技术。

3 基于流分类的数据中心网络动态多径路由策略

当前数据中心网络应用最广泛的 Fat-Tree 拓扑架构，为数据中心网络的流量提供了冗余的等价路径。大部分的数据中心中，80%的数据流持续的时间少于 10 秒。然而少量长时间存在的流（少于 0.1%，持续时间大于 200 秒）却贡献了近 20%的总流量，然而总流量的一半以上都是有持续时间少于 25 秒的流所贡献的[36]。数据中心网络中 85%以上的数据流都是小于 100KB 的流。鉴于数据中心网络中流的特点，数据中心网络要能够兼顾大流和小流，并且能够充分利用 Fat-Tree 拓扑带来的多等价路径，从而达到网络中较高的平均链路利用率。

3.1 现有路由策略存在的问题

3.1.1 ECMP 存在的问题

为了利用数据中心网络多路径的优势，当前最多使用的是 ECMP 路由方法。当一个拥有多条候选路径的包到来时，对这个包的部分头字段内容进行哈希计算，从而从多条候选路径中确定这个包的路径。一个数据流的所有数据包都将使用同一路径，从而保证包的到达顺序。ECMP 方法的一个关键的限制是，当两个或更多的大的长时间保持的流通过哈希计算后结果产生碰撞时，这几个流会从同一个出端口进行转发，导致了部分链路上发生了拥塞和过载，而使一些本可用的链路没有得到充分的利用。这种网络资源的不合理分配，造成了网络传输性能的瓶颈。

ECMP 的性能本质上取决于流的大小和每个主机产生的流的数量。这种基于哈希的转发方法，针对不均匀的通信方式，尤其是包含大数据块的传输形式时，需要避免因哈希碰撞而产生的网络性能瓶颈。ECMP 的不足之处在于，该策略的实施完全脱离网络的运行状态，也与数据流对网络资源的需求无关。ECMP 仅仅实现了数据流在数量上在多等价路径条件下的近似平均分配。

3.1.2 DLB 存在的问题

DLB[37]（Dynamic Load Balancing，动态负载均衡）路由方法是一种应用在 Fat-Tree 拓扑架构下的动态的路由方法。DLB 基于 OpenFlow 技术实现，它以网络链路的负载均衡为目的通过反馈网络的实时链路状态，动态地为网络中的数据流进行路由。该路由方法采取一种类似深度搜索的策略，每次从边缘交换机开始，向上层逐步选取下一跳的交换机，直到确定最顶层的交换机。鉴于 Fat-Tree 拓扑结构的特性，一旦确定了最顶层的交换机，其到达目的主机所连接的边缘交换机的路径唯一，所以 DLB 方法只需向上搜

索到所要到达的最高层的交换机则可以确定完整的路径。在向上选取下一跳的交换机时，采取局部最优的策略，基于最坏适应原理利用贪心算法在多个候选交换机中选取与当前交换机之间链路空闲带宽最大的。

DLB 路由方法的优点在于简单易实现，额外开销小，仅需要额外通过 OpenFlow 来实时获得网络全部链路的使用状态信息。DLB 的不足之处在于，只考虑了局部优化的效果，并没有从全网的整体优化来设计。此外，DLB 没有将数据中心网络中流量的特点纳入参考因素，忽略了数据中心网络中不同大小的流对链路带宽使用的不同影响。

3.2 动态的流分类路由策略

3.2.1 方法设计思想

针对一个采用 Fat-Tree 拓扑架构的数据中心网络，在对其设计路由策略时应考虑以下几个方面：

（1） Fat-Tree 拓扑架构的特点和优势。

Fat-Tree 拓扑架构作为一个多根树的网络拓扑架构，提供了大量的冗余链路。任意来自不同边缘交换机的两个主机之间存在多条等价路径。对于一个 k 元 Fat-Tree 拓扑架构，来自同一 Pod，不同边缘交换机的两个主机间存在 k/2 条等价路径；任意来自不同 Pod 的两个主机间存在 $k^2/4$ 条等价路径。

（2） 数据中心网络中流量的特点。

数据中心网络中存在大量的流，这些流的大小从几 KB 到几 GB。这其中大部分都是生存时间小于 10 秒的小流，占到了全部流数目的 80%。少量生存时间大于 200 秒的大流贡献了全部网络流量的 20%，而数目只占到全部流数目的 0.1%[37]。流的生存时间的长短反映了流的大小，大小不同的流对网络链路带宽的需求自然也是不同的。所以在为流计算路由的时候，要将流的大小的差异性考虑在内。因此，可将网络中的流依据大小，区别为大流和小流两种，并针对二者采取不同的路由算法[39]。

（3） 依据网络的当前状态信息计算路由。

当一个新流进入到网络当中时，在计算路由时若不考虑当前的网络各个链路的使用状态，则计算出来的路径状态是不可知的。这个新流在转发的过程中可能会遭遇到各种状况，例如在某段链路上转发时发生了拥塞；或者与其他流发生了碰撞；存在一条优于当前的路径而没有选择。

（4） 能够适应网络的变化动态调整路由。

在数据中心网络中，每时每刻都有大量的新流进入，同样也有大量旧的流完成转发。网络中各个链路的状态也随着流量的变化而实时变化。在网络状态不断发生变化的过程

中，生存时间长的大流的初始转发路径可能已经不能够满足它们对网络资源的需求，为了更加合理的利用网络资源，更充分地利用等价多路径带来的优势，可以动态地对大流的转发路径进行调整，避免大流因为网络状态的变化而导致的吞吐率的下降和拥塞的发生。

（5） 权衡路由计算带来的额外开销。

路由计算和路由的调整都会带来计算和存储资源以及时间上的开销。如果路由策略过于简单，虽然会减少这部分开销，但是得到的结果可能会不满足性能上的需求，例如ECMP 路由算法；如果路由策略过于复杂，这部分的开销增大，而网络中存在大量的小流，对这些在数量上占大多数的小流执行复杂的路由算法可能会适得其反，反倒增大小流的转发时延。当依据网络状态进行路由时，还需要额外获得网络的全局状态信息，这些也会带来额外的资源的开销[40]。因此，在设计网络的路由策略时，要权衡路由算法的复杂程度与可能带来的网络转发性能的提升。

基于以上几点，本文提出的路由策略的主要思路是，首先引入 OpenFlow 技术，在SDN 的架构下实现 Fat-Tree 拓扑结构的数据中心网络的路由策略。利用 OpenFlow 对数据中心网络集中控制的优势，获得全网的视图信息。根据交换机之间链路的状态信息，为进入到网络中的流选取适合当前网络状态的链路。

其次，针对容易引起网络拥塞的大流，这种流的数据量大，在网络中的生存时间较长，为了适应不断变化的网络状态，对这些大数据流实施动态的调整，即对大流进行动态重路由。对大流进行重路由的核心思路是在大流的源主机与目的主机间的多条等价路径中，选取当前传输能力最高的那条路径。若路径与大数据流当前的路径不同，则改变这个大数据流的路由；若路径相同则保持大流的路由不变。大数据流的重路由是一个基于全局考虑的过程，要综合考虑路径中各段链路的带宽使用情况，来为大数据流选择最优的路径。

而对网络中数量上占据绝大多数的小数据流，路由算法要简单高效，避免为大量的小流进行路由计算所带来的过大的开销。来自同一源主机与目的主机对的多个小数据流要能够充分利用 Fat-Tree 拓扑架构提供的多等价路径，从而提高链路的利用率，减小与大数据流发生碰撞的概率。

当一个流的第一个数据包进入到网络中时，网络的控制模块为这个数据包计算路由，之后这个流的每一个数据包都将按照这个路由进行转发。但是在最开始计算路由的时候，对这个流的大小是不可知的，为此对于新进入网络中的所有数据流都采取相同的路由策略。ECMP 路由算法简单，但在实际应用中缺乏灵活性，完全忽略网络中链路的状态，很容易导致链路带宽分配的不合理，从而导致对大流的动态重路由次数增多，负

担加重，造成更多额外的系统开销。在无法确定数据流的大小的情况下，可以考虑从链路的带宽使用状态出发，每次在可选的路径中选取各链路空闲带宽都较大的那条路径，减少初始情况下大流之间发生碰撞并引发拥塞的概率，从而降低针对大数据流进行重路由的次数与开销。

3.2.2 流分类路由控制的系统框架

本文提出的流分类路由策略基于 OpenFlow 技术实现，要求底层的物理网络全部由支持 OpenFlow 协议的交换机组成。路由功能作为 OpenFlow 控制器的一个模块，集成在所使用的 OpenFlow 控制器当中。路由控制功能的框架图如图 3.1 所示。

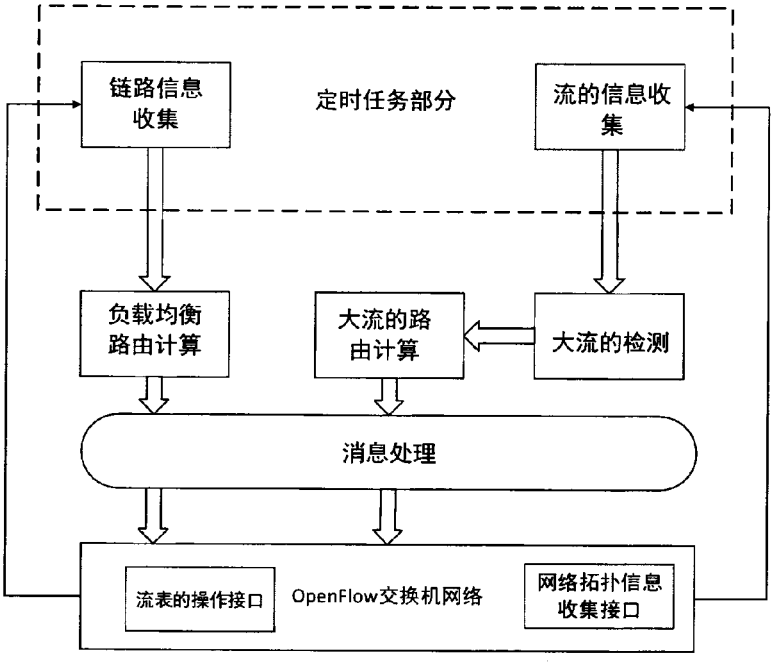

图 3.1 基于 OpenFlow 的流分类路由系统框架图

Fig.3.1 OpenFlow based routing system framework for classified flows

（1） 网络视图信息的收集

本文提出的路由策略中，在转发路径的选择上依据网络的当前状态，从而提高网络的带宽利用率，减小拥塞发生的概率。为网络中的流计算路由所要依据的网络状态信息包括网络中所有链路的状态统计信息，即网络中每条链路当前的传输状态；还包括在网络中进行传输的每个流的状态统计信息，包括每个流的匹配信息、流的已转发字节数等等，并以此为依据，判断每个流是否为大流。

OpenFlow 技术提供了用来收集网络状态信息的接口。通过这些接口，OpenFlow 控制器向 OpenFlow 交换机发送 Read-State 消息可以查询所需的链路状态统计信息和流的状态统计信息。对于网络中的链路，在物理上就是将两个来自不同交换机端口连接在一起的通道。OpenFlow 直接提供了对交换机物理端口的信息查询接口，再结合全网的拓扑视图确定物理端口所到达的目的交换机和对应的物理端口，这样就确定了一条链路的状态统计信息。在某一时刻向交换机发送这一物理端口的统计信息请求，交换机在收到请求后会向 OpenFlow 控制器回馈截至到这一时刻这个物理端口的统计信息，包括这个已经收到和发送出的字节数、包的数量等等。但这些信息不足以反映当前物理端口的使用情况，为此设置一个时间周期 T_1，控制器每隔 T_1 周期，便向交换机发送一次物理端口的统计信息请求。通过与前一时间间隔获得的统计信息进行比较，可以计算出在前一个周期内，物理端口的发送或接受的字节数。控制器将保存一张记录网络中的物理端口在最近一个时间周期内的统计信息，即物理端口在最近一个时间周期内的状态。表中的每一项对应交换机的一个物理端口，每一项中包括以下六个元组，这个六元组可表示为 $\{srcDPID, srcPortNo, dstDPID, dstPortNo, rx\_bytes, tx\_bytes\}$ 其中 srcDPID 为源交换机的 DPID，srcPortNo 为源交换机的物理端口号，dstDPID 为目的交换机的 DPID，dstPortNo 目的交换机的物理端口号，rx_bytes 为收到的字节数，tx_bytes 为发送的字节数。其中 DPID 是 OpenFlow 网络中每台交换机拥有的唯一标识。表中每一个六元组即是网络中以 srcDPID 交换机中的 srcPortNo 端口为源点的一条链路的信息。通过比较链路之间的发送字节数或者收到字节数的大小，可以得出在上一个时间周期内链路之间链路利用率的大小。发送字节数或者收到字节数比较大的那条链路，其链路利用率较高，链路的空闲带宽比较少。

对时间周期 T_1 的设置，查询周期越小，查询结果越接近当前的链路使用情况。但是 T_1 设置的越小，控制器向交换机发送端口信息查询请求的次数就越多，造成系统内过多的额外开销。周期大小的设定还要考虑系统完成以此查询所需要的时间，要保证在这个时间周期内，系统能完成以此完整的查询并将结果记录到六元组的表内。T_1 的值也不能过大，周期设置的过大，虽然会减小状态查询带来的额外开销，但是得到的统计数据不能够准确反映当前的网络链路状态。所以对时间周期的设置要根据网络拓扑的规模，控制器的处理性能，状态信息的精确程度等多方面因素进行综合考虑。鉴于 Fat-Tree 网络拓扑架构的特点，为了获得网络中所有的链路状态统计信息，不需要向全部的交换机发送物理端口统计查询请求。仅仅向 Fat-Tree 拓扑结构中处在中间层的所有的聚合交换机发送物理端口统计查询请求，就能获得当前网络中所有链路的信息。

网络中流的状态信息主要用于对流进行分类，判定一个流是否为大流，从而决定是否对其进行重路由。流的统计信息的获取与网络中物理端口的统计信息的获取方式相似，也是通过控制器 OpenFlow 提供的接口向 OpenFlow 交换机发送流的统计信息请求。OpenFlow 交换机收到请求信息后，将每个流的信息反馈给控制器。这些信息包括流的匹配域、流的已传输的字节数和包的数量、流的持续时间等。流在转发的过程中要经过不只一台 OpenFlow 交换机，从这个流经过的交换机都能获取流的信息，为了获得当前时刻最新的流的统计信息，并且对所有的流不进行重复的查询，只需要向所有的边缘交换机发送流的信息请求，查询从每个边缘交换机与主机相连的端口转发出的流，这样就能保证查询到全部的流，并且没有重复进行查询。由于本路由策略对大流进行实时检测，并动态地对其进行重路由，因此作为判定流大小的依据，流的统计信息的获取也要实时进行更新。为此，为流的状态信息的获取设置一个执行周期 T_2，即每隔一个 T_2 周期，对全网中的流的状态进行一次查询。与 T_1 类似，周期 T_2 的值不能太小，要足够完成一次完整的查询和对所有大流进行重路由的过程，T_2 太小会带来频繁的流查询操作、判断流的大小、对大流进行重路由等操作，造成过多的额外开销；T_2 太大同样不能够精确地反映当前网络中流的状态，不能够及时对网络中的大流的路由进行调整，从而增大了大流引发链路拥塞的概率。

（2） 网络中大流的检测

网络中大流的检测主要以查询到的网络中全部流的信息为依据。不同的 Fat-Tree 拓扑网络中的链路带宽不同，决定着对大流的判定标准也不同。为了适应不同的 Fat-Tree 网络链路带宽，当一个流的转发速率达到带宽的 1/10 时，就认定这个流为一个大流。针对交换机反馈的每一个流的信息，都包含匹配域 match、持续时间 dur_time、已转发的字节数 bytes_count 和包的数量 packets_count。假设当前 Fat-Tree 网络中交换机之间的链路带宽为 B，流的速率占带宽的比例 $\varphi = \frac{bytes\_count}{dur\_time \times B}$ 。若 $\varphi > 0.1$，则记录这个流的匹配域 match。由于控制器对流的统计信息是以 T_2 为周期进行，所以对网络中大流的检测也是以 T_2 为周期，从而实现对网络中大流的实时检测，以便进一步实现对大流路由的动态调整。

（3） 路由计算

系统中的路由计算模块由两部分组成，包括与流的大小无关，与链路带宽相关的局部动态负载均衡路由计算；还包括针对网络中大流的重路由计算。路由的计算模块通过获得网络的全网拓扑结构、链路的使用情况和流的转发情况，为网络中的流选取合适的链路和路径。路由计算模块是控制器的核心模块之一。当路由计算模块完成流的路由计

算之后，控制器将流的转发路径转发成相应的流表项，通过控制器与交换机之间的Modify-State 消息，添加到对应交换机上的流表中。

（4） 消息处理

消息处理模块的主要功能是将上层其他模块向交换机发出的请求和操作按照OpenFlow 协议的格式要求进行封装后发送给交换机，包括获取交换机中物理端口和流状态统计信息的 Read-State 消息，将计算出的路由添加到交换机流表中的 Modify-State 消息等。同样，也会处理交换机发送给控制器的消息，包括当一个流找不到流表项时发送给控制器的 PacketIn 消息，交换机在收到统计信息请求后反馈的统计信息等，这些消息都交给消息处理模块，并从中提取出我们需要的数据，作为控制器各个控制模块控制网络的依据。

3.2.3 路由策略的实现及算法设计

本文提出的基于流分类的动态路由策略分为两部分，第一部分是针对进入到数据中心网络中的新流，在无法事先知道其大小的情况下，从链路的带宽使用状态角度出发，在等价路径的链路间执行以负载均衡为目的的 DLB 路由算法。第二部分是针对已经在数据中心网络中进行转发的流，对这些流进行大小流的的分类，并检测其中的大流，每发现一个大流，对其执行重路由算法，多条等价路径中为大流选择当前最优的路径。大流的检测以一定的时间周期循环执行，以达到动态调整网络中大流的目的。路由策略的执行流程图如图 3.2 所示。

当主机发送一个新的流进入一个 Fat-Tree 拓扑架构的数据中心网络中后，与主机相连的边缘交换机首先查找自身的流表中是否包含相应的流表项，在发现没有相匹配的流表项之后（因为是新流），交换机向 OpenFlow 控制器发送一个 PacketIn 消息，由控制器来决定这个流的路由。对于这个新进入网络的流，由于无法知道该流的大小，使用 DLB 路由算法计算路由；之后将得到的路由通过 OpenFlow 协议在相应的交换机的流表中添加这个流的流表项。至此，完成了路由策略的第一部分。

大流的检测模块周期性地检测网络中的大流。当获得网络中在某一时刻的所有大流后，针对大流的路由模块逐一对这些大数据流重新计算路由，经过重路由计算后，若大流的路由发生改变，重新为这个大流在网络的交换机中添加流表项，大流将按照新的流表项中的操作进行转发。路由决策系统每隔一定的时间周期对网络中存在的大数据流完成一次重路由，从而达到对网络中大流的动态调整。至此，完成了路由策略的第二部分。

（1） DLB 路由算法

对于第一部分采用的 DLB 路由算法，其核心思想是利用贪心原理，以类似深度优先搜索的方式，从边缘交换机开始逐层向上选取下一跳的交换机，直到确定了路径所要到达的最高点。在每次选择下一跳的交换机时，选取和当前交换机之间空闲带宽最大的那条链路。

（2）对大数据流的重路由算法

对于路由策略的第二部分，本文提出的针对大流的路由算法，其核心思想是在选择路径的时候考虑这条路径上所有交换机之间的链路使用情况，而不仅考虑从当前交换机到下一跳的最优链路。

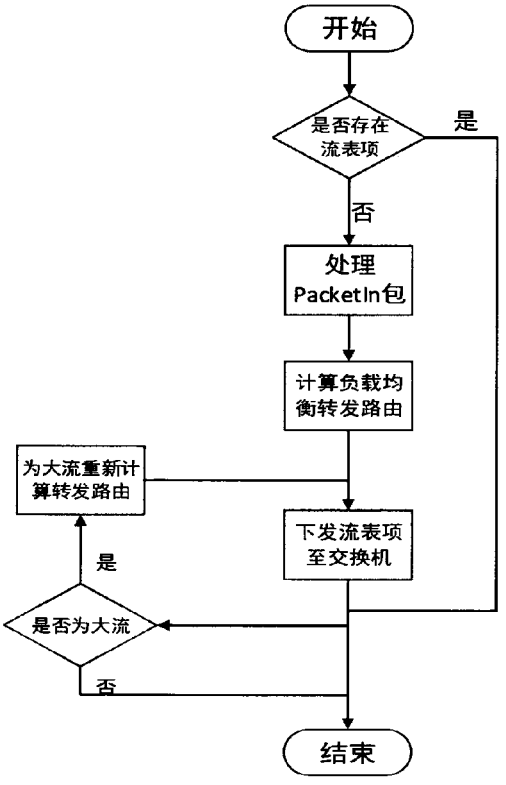

图 3.2 基于流分类的路由的流程图

Fig.3.2 The flow chart of the routing system based on flow classification

如图 3.3 所示，一个由 4 台交换机和 2 台主机组成的一个简单的网络。主机 H1 和 H2 之间存在两条等价路径，分别为 A-C-D 和 A-B-D，其中网络中的链路带宽都相同。当前各个链路的空闲带宽所占的百分比如图中的百分数所示。若使用 DLB 路由算法，

则基于贪心算法，将选择交换机 B 作为 A 的下一跳的交换机，因为 AB 间的空闲带宽比例为 70%，大于 AC 之间的 50%，最终由 DLB 路由算法选择的路径为 A-B-D。但是由于 BD 之间链路的空闲带宽率只有 30%，成为了路径 A-B-D 的传输瓶颈，依据木桶原理，流在路径 A-B-D 上只能以链路带宽的 30%进行转发。而路径 A-C-D 可以提供 50%的带宽，要优于 A-B-D。我们将一条路径中空闲带宽最小的那条链路称作这条路径的瓶颈链路。由此可见，链路 CD（或者 AC）是路径 A-C-D 的瓶颈链路，链路 BD 是路径 A-B-D 的瓶颈链路，一条路径的最大空闲带宽由这条路径的瓶颈链路的空闲带宽所决定。

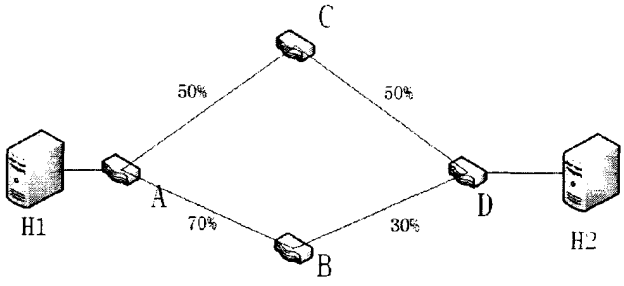

图 3.3　DLB 算法的缺点

Fig.3.3　The drawback of the DLB algorithm

若路由的目标是得到上面例子中 A-C-D 的结果，就不能只考虑局部最优的条件，还要考虑交换机 CD 之间和 BD 之间的链路情况，找出并比较每条等价路径的瓶颈链路。图中的网络拓扑比较简单，为一个流计算路由只需要获得两条等价路径的全部 4 条链路的情况。在 Fat-Tree 结构的数据中心网络，数据中心网络中存在大量的流，主机之间有多条等价路径。若对网络中的所有流在路由时都考虑路径的全局状态，将带来大量的额外开销，对于小数据流来说这样的开销适得其反。为此，我们将这种策略应用在网络中少量的大流上，并且主要从以下几个方面考虑。首先大流在网络中存在的时间很长，会经历网络状态的不断变化，需要一个动态地路由策略让大数据流适应网络的实时变化；大流的存在时间远大于为大流计算路由的时间；大数据流容易造成网络中链路的拥塞，通过调整大流来改变链路的状态，效果最明显。

对于 Fat-Tree 拓扑网络中的一个大流 f，记 $P=\{p_1,p_2,\cdots,p_k\}$ 为大流 f 的 k 条等价路径。对于 P 中的每一条路径 p_i 都由若干段链路组成记为 $p_i=\{l_{i1},l_{i2},\cdots,l_{in}\}$，其中 n 的值为 2 或者 4，当 $n=4$ 时，说明大流 f 的源主机和目的主机来自于 Fat-Tree 网络中的不同 Pod 下；当 $n=2$ 时，表明大流 f 的源主机和目的主机来自于 Fat-Tree 网络中同一 Pod

下的不同边缘交换机。令 u_{ij} 标识链路 l_{ij} 的空闲链路带宽，其中 $j=1,2,\cdots,n$。则路径 p_i 上的瓶颈链路的空闲带宽为

$$B(p_i)=\min u_{ij},\ j=1,2,\cdots,n \tag{3.1}$$

即路径 p_i 整体能保证提供的最大空闲带宽为 $B(p_i)$。在 f 的多条等价路径中，若 $B(p_i)$ 值最大的那条路径的 $B(p_i)$ 大于当前 f 所占的带宽，则这条路径为 f 的新路径。若所有路径的 $B(p_i)$ 都小于 f 当前所占的带宽，则 f 的路径不变。

3.3 本章小结

本章内容主要介绍了基于流分类的 Fat-Tree 拓扑网络下等价多路径的路由策略。已有的 ECMP 路由算法在设计上脱离网络的状态，仅仅通过哈希算法静态地为网络中的数据流选择转发的路径；DLB 路由算法考虑了网络的局部状态，依据最坏适应原理选择局部最优的链路，但 DLB 没有考虑网络的全局状态，所选的路由并不能使全网的效果最优。本策略从链路带宽的使用情况和流的大小两个角度出发，既要兼顾路径中链路的带宽使用情况，又要针对数据中心网络中流量的特点，对不同大小的数据流采取不同的路由算法。对于网络中的小流，使用 DLB 路由算法；对于网络中的大流，本文提出了一种新的考虑路径整体转发能力的路由算法，动态实时地调整网络中大流的路由。两个路由算法相结合，共同完成网络中流的路由，有效的利用 Fat-Tree 拓扑网络的等价多路径优势，提高网络冗余链路的带宽利用率，提高网络的吞吐量。

4 基于 OpenFlow 的数据中心网络拥塞路由管理策略

在上一章，详细阐述了基于 OpenFlow 的数据中心网络动态多等价路径下的路由策略。这种路由策略通过动态实时调整网络中大流的路由，从而提高网络整体的链路了利用率，发挥 Fat-Tree 拓扑架构下多等价路径的优势。这种区别大小流的路由策略可以视为是一种对拥塞的发生所采取的一种主动避免的方式。通过在拥塞发生之前，动态地调整网络中大流的路由，来减少出现拥塞链路的概率。本节之后的内容主要提出了一种被动式的拥塞路由管理策略，即在网络中出现拥塞的链路时，如何调整网络资源的分配，从而使链路状态从拥塞状态切换至非拥塞状态。

4.1 基于 OpenFlow 的拥塞路由管理策略的主要思想

数据中心网络中存在大量的流，主机之间发送的流的不均衡性，路由算法的不完善，都有可能造成流在网络中的某些链路上产生了拥塞，从而造成了网络传输性能的降低。利用 SDN 网络架构和 OpenFlow 技术，能够对网络实施集中控制，并且 OpenFlow 技术可以对网络实现面向流的细粒度管理。对网络的集中控制，OpenFlow 控制器可以获得网络的全局拓扑视图，对网络的全局状态有一个掌握。利用 OpenFlow 协议所提供的接口，控制器可以实时获得网络的状态统计信息，包括网络中所有 OpenFlow 交换机的物理端口的状态信息即网络中链路的信息；网络中所有流的状态信息。根据这些统计信息可以发现网络中出现拥塞的链路，以及在这条链路上转发的流的状态。

定位了发生拥塞的链路之后，对这条链路上的流进行转发控制，从而改变链路的拥塞状态。本文提出一种基于 OpenFlow 的数据中心网络的拥塞控制策略，当网络中的链路发生拥塞时，通过面向流的细粒度控制，改变拥塞链路的拥塞状态。当一条链路发生拥塞后，主要采取两种措施来改变拥塞的情况

（1）对于新进入网络中的数据流，对其进行路由时要尽量避开网络中拥塞的链路，选取其他等价路径。

（2）对于在这条链路上进行传输的数据流，对其中的大数据流进行重路由，将这些大数据流重路由至其他的等价路径上，利用 Fat-Tree 结构多条等价路径的优势

面向流的链路拥塞控制策略基于 OpenFlow 实现，主要应用在 OpenFlow 交换机组成的 Fat-Tree 拓扑网络。拥塞路由管理可作为 OpenFlow 控制器的一个控制模块，集成在 OpenFlow 控制器当中。具体的拥塞路由管理的框架如图 4.1 所示。

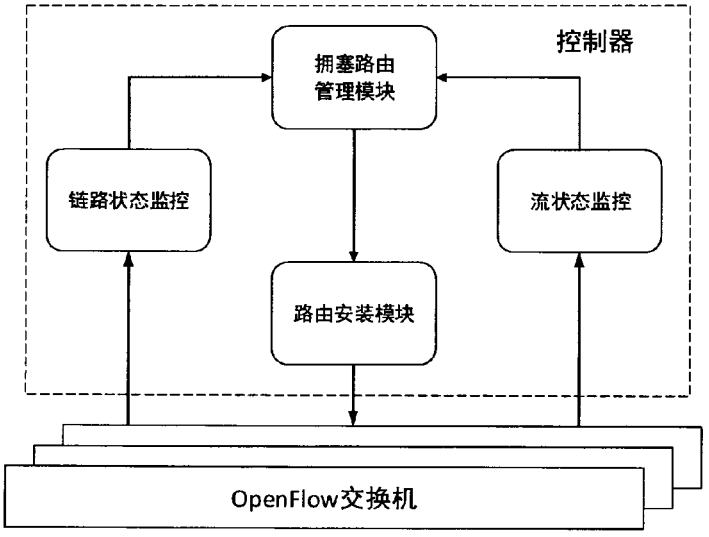

图 4.1 基于 OpenFlow 的拥塞路由管理系统框架图

Fig.4.1 The framework of the OpenFlow based congested routes management system

图中虚线方框内为 OpenFlow 控制器集成的功能模块，包括拥塞路由管理模块、路由计算模块、链路状态监控模块和流状态监控模块。网络的拥塞控制主要利用这几个模块，通过它们之间的共同工作，来处理链路上的拥塞情况。

（1）链路状态监控及拥塞链路检测模块

随着网络中的流量的流入和流出，网络中各条链路的状态也是实时变化的。同一时刻网络中存在过多的大流、路由算法的缺陷和物理链路的故障都有可能造成某些链路上拥塞状况的发生。链路状态控制模块用来实时地监控网络拓扑中所有链路的状态信息，将其中发生拥塞的链路信息发送给拥塞控制模块，进行下一步处理。

（2）数据流状态监控及大流检测模块

数据流状态监控模块主要用来监控发生拥塞的链路上所有流的状态信息。利用这些流的信息，检测其中的大流，选取合适的大流进行路由调整，改变链路的拥塞状态。

（3） 拥塞路由管理模块

拥塞控制模块从链路状态监控模块获得发生链路拥塞的链路的信息，然后通过流状态监控模块，获得拥塞链路上的流的状态信息。拥塞控制模块从两方面对拥塞的链路进行控制。第一方面，在获得了拥塞链路的信息之后，拥塞控制模块要控制进入到网络中的新流，对于这些新流，在进行路由时要尽量避开拥塞的链路，使用其他非拥塞的链路；若无法避开拥塞的链路，则暂时不对这个流进行路由，直到链路的拥塞状态发生变化。

第二方面，在发现拥塞的链路之后，监控拥塞链路上正在转发的流，对其中的大流进行路由调整，将大流重新路由至其他等价路径上，直到拥塞链路的利用率降到某一阈值。

（4） 路由安装模块

路由安装模块负责将拥塞控制模块中进行调整后的流的新的路由部署在 OpenFlow 交换机的流表项中。

以上四个模块作为实现拥塞控制的主要组成部分，与 OpenFlow 控制器的基本模块共同发挥作用，从而实现面向流的链路拥塞路由管理。

4.2 拥塞路由管理逻辑

4.2.1 拥塞路由管理逻辑的总体设计及流程

拥塞路由管理分两部分，第一部分是当发现拥塞链路后，对拥塞链路上的流进行调整；第二部分是网络发生拥塞后，对于新进入网络中的流的路由策略的变化。具体的操作流程如图 4.2 。

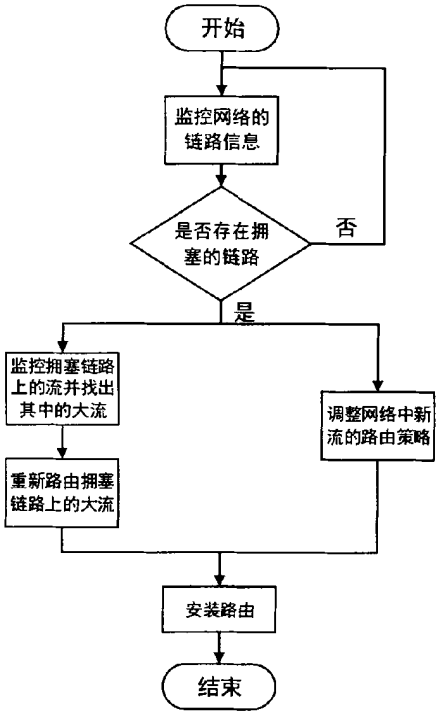

图 4.2 拥塞路由管理逻辑流程图

Fig.4.2 The flow chart of the congested routes management logic

（1） 对于基于 OpenFlow 的 Fat-Tree 数据中心网络，OpenFlow 控制器实时获得网络中的链路状态统计信息。

（2） 在获得链路信息后，判断各个链路是否发生了拥塞，若无拥塞的链路，则继续对链路实时监控；若发现了拥塞的链路，将发生拥塞的链路信息发送给流状态监控模块，监控拥塞链路上流的状态，获得其中大流的转发状态信息。

（3） 对拥塞链路上的大数据流重新计算路由，新的路由要将大数据流调整到其他可用的等价路径上。

（4） 当发现拥塞的链路后，对于进入网络的新流，为它们计算路由时要尽量避开拥塞的链路。步骤（4）与步骤（3）并行执行。

（5） 安装路由操作将步骤（3）与步骤（4）中计算的路由部署在相应的交换机的流表中。

4.2.2 拥塞链路上大流的路由调整策略

对于网络中的一条拥塞的链路，导致其发生拥塞的直接原因是过大的流量经过这条链路之上而超过了这条链路的可提供的带宽能力。为此，最直接的解决方法就是改变拥塞链路上流的转发路径，将某些流迁移至其他非拥塞链路上进行转发，从而减少拥塞链路上的流量负载。

当链路发生拥塞时，采取对这条链路上的大流进行路由的调整的策略。每当检测出一条大流，我们要在这个大流的其他 k-1 条路径中重新为其选定一条路径。重新选择的路径要满足组成这条路径的所有链路的空闲带宽都大于这条检测到的这条大数据流当前所占的链路的带宽，以免这条大流在新的路径上产生拥塞。若在其他等价路径中不存在满足条件的路径，则停止对这条大流进行重路由，对链路上其他的大流进行路由调整，直到对这条链路上所有的大数据流完成路由的调整。在对这条数据流进行重路由的过程，若采取计算这条大数据流其余全部 k-1 条路径，再通过比较组成每条路径的链路的带宽使用情况，求得每条路径的瓶颈链路，再通过比较这些瓶颈链路的大小来决定大数据流的新路径，这种方式会带来大量的额外开销。并且，发生拥塞的链路只是整条路径的一部分，因此考虑对原有路径进行局部调整，从而完成对拥塞链路上大数据流的重路由。下面举例说明对大数据流进行局部调整的思路。

假设大数据流 f 的源主机与目的主机来自于 Fat-Tree 拓扑不同的 Pod。基于流的传输方向，f 的路径以核心交换机为界可分为上行路径和下行路径两部分。链路的拥塞可以发生在这条链路的任何位置，上行路径的边缘交换机与聚合交换机之间，聚合交换机与核心交换机之间；下行路径的边缘交换机与聚合交换机之间，聚合交换机与核心交换

机之间。若发生拥塞的链路在下行路径中，由于从确定的核心交换机到目的主机的路径唯一，需要重新选择核心交换机；若发生拥塞的链路在上行路径中，则从拥塞链路的起点开始重新搜索路径。为此可以分两种情况

（1） 拥塞链路在上行路径

针对这种情况，控制器从拥塞链路的起点开始重新为 f 计算路由，并依据最先适应原理，每次向上层选取第一条空闲带宽大于当前 f 所占带宽的大小的链路，并且在到达核心交换机之后还要判定下行路径的带宽是否满足条件；在某一层若无满足条件的链路时，则回溯至下面一层进行链路搜索。

（2）拥塞链路在下行路径

针对这种情况，需要为 f 重新选择核心交换机，则需要从上行路径的聚合交换机开始进行路由。若拥塞链路发生在下行路径的边缘层和聚合层之间，则基于 Fat-Tree 拓扑结构的特点，可直接回溯到上行路径的边缘交换机层。

表 4.1 拥塞链路上大流的重路由算法

Tab. 4.1 The re-routing algorithm for long-lived flow in the congested link

拥塞链路上大数据流的重路由算法
输入：Fat-Tree 网络全局视图； 拥塞链路 L 上的大流 Flows； 网络中所有链路的状态统计信息。
输出：Flows 中每个大流的路径 Path
1 for f in Flows do:
2 计算 f 的所有等价路径 Paths；
3 计算 f 当前所占的带宽；
4 获取拥塞链路 L 的起点，确定重路由的开始节点；
5 向上一层搜索最先适应的链路；
6 若同层中无满足条件的节点则向下层回溯；
7 到达最高层节点后，判定下行路径的链路空闲带宽， 若链路空闲带宽 > f 当前所占带宽，则返回 f 的路径； 否则 回到步骤 6

4.2.3 拥塞状态下网络中新流的路由策略

数据中心每时每刻都会产生大量的新流，当网络中的某些链路发生拥塞时，新流的到来会对链路的拥塞情况造成进一步的恶化。为此，在网络中的拥塞没有被解决之前，

在为新进入网络中的流计算路由时，要避开这些拥塞的链路。对于一个 Fat-Tree 拓扑结构的数据中心网络，拥塞可以发生在任何一条链路上，可以是核心层与聚合层之间的链路，也可以是边缘层与聚合层之间的链路。在 Fat-Tree 网络中，流的转发从源主机开始先上行到高层的交换机，然后从高层的交换机下行到目的主机。

针对上一章提出的路由算法，对于无法预知大小的网络的新流，所使用的 DLB 路由算法只考虑流的上行路径，所以无法避开下行路径中可能出现的拥塞链路。因此需要采取新的路由算法。新的算法是对 DLB 路由算法的改进，采取类似的深度搜索的方式，并加入回溯机制。当一个新的流进入到网络中，确定需要到达的最高层之后，开始向上逐层进行搜索，确定下一跳的节点和链路。向上逐层搜索的过程与 DLB 路由算法相同。由于 DLB 采用贪心算法选取连通下一跳的链路，每次选取空闲带宽最大的那条链路，自然地避开了发生拥塞链路。因此，当拥塞链路出现在上行搜索的过程中时，依据 DLB 路由算法，不会选择拥塞的链路。当执行 DLB 算法向上搜索到最高点，确定了最高点的交换机之后，基于 Fat-Tree 拓扑结构的特性，最高点到目的节点的路径也唯一确定，DLB 路由算法不会向下搜索找寻下行的路径。若在唯一确定的下行路径中存在拥塞的链路，则说明这条路径没有达到所要的目的，即当前选择的最高点不是适合的节点。针对这种情况进行回溯，从最高点的前一个节点重新搜索。若对当前最高点的前一个节点依旧不存在满足条件的路径，则继续向前回溯，直到回溯到边缘交换机。若遍历了所有的等价路径依旧没有满足条件的路径，则此时将这个流挂起，直到下一个链路监控周期的来到，再为其计算路由。以下图 4.3 为例，进行说明。

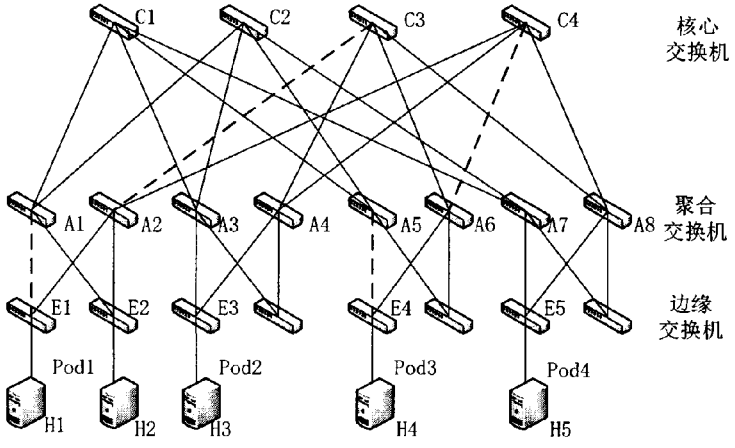

图 4.3 存在拥塞链路的 Fat-Tree 拓扑网络

Fig.4.3 The Fat-Tree network with congested links

图中的虚线的链路表示发生拥塞的链路。假设存在网络中存在三个新流，分别是流 f1 从 H1 到 H5，f2 从 H1 到 H2，f3 从 H3 到 H4。对于流 f1，因为 H1 和 H5 来自 Fat-Tree 拓扑中不同的 Pod，首先确定要到达的最高层是核心层，需要向上搜索两次到达核心层。从边缘交换机 E1 开始，向上搜索空闲带宽最大的链路，选取的上行路径为 E1-A2-C4，最高点为 C4，这时的下行路径唯一确定为 C4-A8-E5 且其中不包含拥塞的链路，所以对于流 f1 的路径为 E1-A2-C4-A8-E5。对于流 f2，H1 和 H2 来自同一 Pod 中的不同边缘交换机，要到达的最高层为聚合层，只需要向上搜索一次到达最高层，路径为 E1-A2-E2，且不存在拥塞链路，满足条件。对于流 f3，因为 H3 和 H4 来自 Fat-Tree 拓扑中不同的 Pod，首先确定要到达的最高层是核心层，需要向上搜索两次到达核心层。假设第一次确定的最高点为 C1，则 f3 的下行路径为 C1-A5-E4，其中 A5-E4 为拥塞链路，不满足条件。这时回溯至节点 A3，重新选择最高层节点为 C2，下行路径为 C2-A5-E4，依旧不满足条件。此时，对于 A3 的所有向上的下一跳节点都不满足条件，继续进行回溯，从 E3 开始搜索。若搜索到的上行的最高点为 C3，则下行路径为 C3-A6-E4 不存在拥塞链路，则整条路径为 E3-A4-C3-A6-E4。基于 Fat-Tree 拓扑网络的结构特点，搜索到的下行路径中存在拥塞链路时，若拥塞链路为聚合交换机与边缘交换机之间的链路，和直接回溯到上行路径中的边缘交换机，这样可以简化回溯的过程。我们称这种通过避开拥塞链路的路由算法为 Backtracking-DLB 路由算法，算法描述如表 4.2 所示。

表 4.2　Backtracking-DLB 算法描述

Tab. 4.2　The Backtracking-DLB algorithm

Backtracking-DLB 算法
输入：数据流 f； Fat-Tree 网络全局视图； 拥塞链路集合 S。
输出：路径 Path
1 获取 f 的源主机 src 与目的主机地址 dst，计算连接二者的边缘交换机 srcEdge 和 dstEdge；
2 计算源主机与目的主机分别所在的 Pod，确定 f 需要到达的最高层 TopLayer；
3 if TopLayer 为边缘层：
4　　返回 Path=srcEdge。
5 if TopLayer 为聚合层：
6　　计算最高点及下行链路 downLink；
7　　if downlink in S；
8　　回溯至上行的边缘层节点重新计算

表 4.2 续

Tab. 4.2 Cont

Backtracking-DLB 算法
9 返回 Path。
10 if TopLayer 为核心层：
11 计算上行路径至最高点 CoreSw；
12 计算最高点 CoreSw 到 dstEdge 的下行路径；
13 if 下行路径中的链路不在 S 中：
14 返回 Path。
15 else if 拥塞链路在边缘层与聚合层之间：
16 回溯到上行路径中的边缘节点重新计算；
17 else 回溯到上行路径中的边缘层重新计算；

4.3 本章小结

本章主要提出了一种基于 OpenFlow 技术的 Fat-Tree 结构数据中心网络下的链路拥塞路由管理策略。本策略利用 OpenFlow 技术，在网络中存在拥塞链路时，对拥塞链路上的大数据流采取重路由的方式，将拥塞链路上的部分流量转移到其他等价路径上，从而减小拥塞链路上的负载；同时对新进入网络中的流，在进行路由时，避开发生拥塞的链路，并提出了一种带回溯的 Backtracking-DLB 路由算法，从而不再增加拥塞链路上新的负载。本章的拥塞路由管理策略是一种针对拥塞状态的被动处理机制，即当拥塞发生之后对网络中的流量做出调整。

5 实验结果与分析

本章节主要针对第三章提出的基于流分类的数据中心网络动态路由策略和第四章提出的基于 OpenFlow 的数据中心网络拥塞路由管理策略进行仿真实验验证并对实验结果进行分析。实验结果分析从链路的带宽利用率和网络中流量的分布情况等方面进行阐述，从而评估所提出的两种策略的性能。

5.1 实验环境搭建与实验设计思路

5.1.1 实验环境的搭建

为了验证本文提出的方法与系统的性能，使用 Mininet 网络仿真平台，用于构建基于 OpenFlow 的 4 元 Fat-Tree 结构的数据中心网络，其中包括 20 台 OpenFlow 交换机和 16 台主机。网络中所有的节点之间的链路设置为全双工链路，链路的带宽设置为 10Mbit，链路的队列最大长度设置为 100。在 OpenFlow 控制器的选取上，使用广泛流行的 POX 控制器，对 Mininet 仿真出的 k=4 的 Fat-Tree 数据中心网络进行集中式的控制。

数据流的生成使用 Iperf 工具，生成的全部数据流都为 TCP 流量。在数据流的转发过程中，使用 bwm-ng 工具来监控网络中全部交换机的物理端口的状态数据。每隔 1 秒收集一次交换机的全部物理端口的状态数据。

以上实验环境的搭建全部在一台物理服务器上实现，物理服务器使用 Ubuntu 12.10 sever 操作系统。

5.1.2 实验设计思路

本文试验的设计思路是将所提出的策略与现有的实施方案进行仿真对比。针对本文提出的基于流分类的动态路由策略（FC-DLB），选取了 ECMP 路由算法和 DLB 算法作为对比实验。在 Mininet 仿真平台上让生成的流量分别在三种路由策略下进行转发。通过比较三者的实验结果，评估所提出算法的性能。

数据流的仿真设计上，为了验证算法的鲁棒性，将采用多种数据中心网络的流量形式，包括：

（1） All_to_All：对于网络中的每一台主机都向其余所有的主机发送数据流。

（2） Random：每台主机都以随机的方式选择网络中的另一台主机作为目的主机，并向这台目的主机发送一个数据流。

（3） Hotspot：对网络中的大部分数据流，都转发给网络中少数几台主机上，从而在这少数几台主机附近节点造成热点。

将这三种形式的网络数据流中的每一种形式都使用三种路由算法进行转发，然后比较每一种数据流形式下，三种路由算法带来的性能的差别。对三种算法性能的评估主要依据每种路由算法下全网的平均二分带宽、网络的平均带宽利用率、网络中的流量负载在各个 Pod 间的分布情况。

面向流的拥塞控制策略作为对路由策略在功能上的补充，以本文提出的路由策略为基础。在实验的设计上与分别执行无拥塞控制功能的路由策略和带有拥塞控制功能的路由策略，评估二者的吞吐量。

下面分别对上述策略进行实验验证与分析。

5.2 基于 OpenFlow 和流分类的数据中心网络的路由策略

（1） 平均二分带宽

如图 5.1 所示，展示了三种不同模式的的数据流在三种不同路由策略下的平均二分带宽。二分带宽指的是用一个截面将一个网络分隔为相等节点的两部分，这两部分在这个截面上的带宽。二分带宽越大，表明网络的通信能力越强。

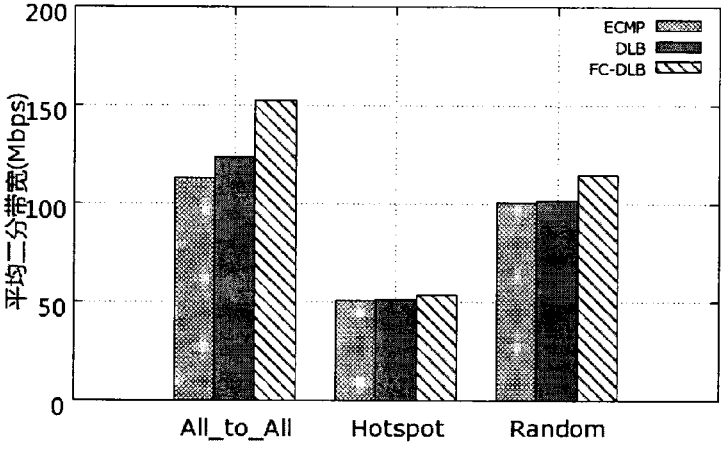

图 5.1 三种流模式在三种不同路由算法下的平均二分带宽对比图

Fig.5.1 Average bisection bandwidth comparison between ECMP ,DLB and FC-DLB routing algorithm

本文提出的基于流分类的动态路由策略在 All_to_All 和 Random 流模式下的平均二分带宽具有较大优势。且对于数量分布较均匀的且流的数目较多时流模式，本文提出的策略优势越大。而对于流分布极不均匀的 Hotspot 模式，由于流分布不均匀，大量的链路没有被使用，因而造成二分带宽的数值较小；可用的冗余路径过少也使三种策略的表现相当，但本文的策略依旧存在少许的优势。

（2） 平均链路利用率

图 5.2 所示的是在不同时刻下，当前网络中所有链路的平均链路带宽利用率，记录了网络中的平均链路带宽随着时间的变化。由于网络中的流初始时刻同时进入网络，所以随着网络中的流陆续完成转发，链路的平均带宽呈下降趋势。

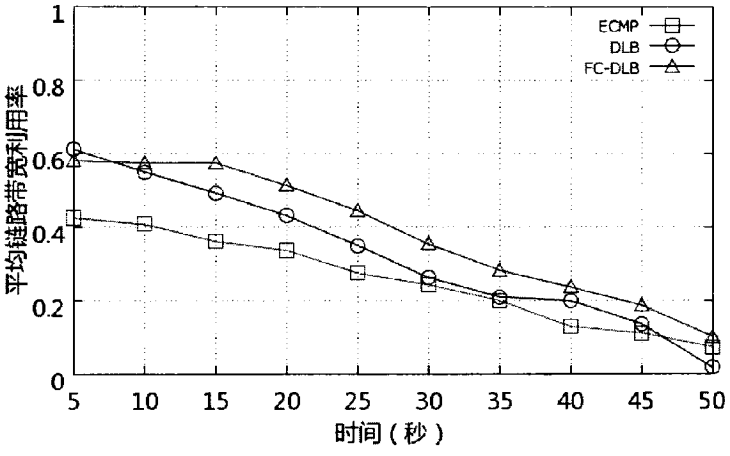

图 5.2 在三种不同路由算法下的平均链路带宽利用率对比图

Fig.5.2 Average link utiliazation comparison between ECMP ,DLB and FC-DLB routing algorithm

本文提出的 FC-DLB 算法在平均链路带宽利用率上均高于另外两种算法 ECMP 和 DLB，并能在高利用率保持较长的时间。说明我们的算法能够提高链路的利用率，在一定程度上实现了链路的负载均衡。

（3） 网络的流量负载分布

图 5.3 中所示的是不同的路由策略下，网络的各个 Pod 之间流量负载的对比情况，可以很明显的看出本文提出的策略在流量负载上的优势。在图中 ECMP 与另外二者之间的差距明显，ECMP 算法下的各 Pod 的流量负载范围为[1153,3548]，平均值为 2304Mbits；DLB 算法下各 Pod 的流量负载范围为[1889,4239]，平均值为 3159Mbits；本文提出的算法下各 Pod 的流量负载范围为[2245,4444]，平均值为 3425Mbits。通过方差的计算，本文所提出算法的方差最小，说明与其他算法相比本文所采用的路由使流量负载更加平均。在流量负载的大小上，我们的算法远高于 ECMP，说明本文提出的路由策略能为网络带来更大的吞吐量和带宽利用。

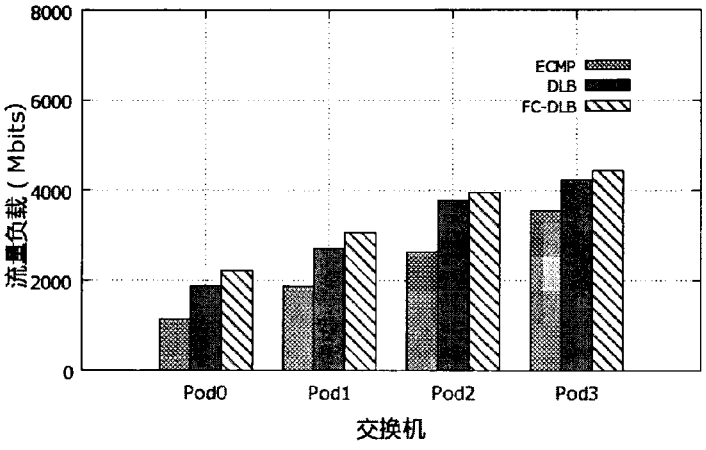

图 5.3　在三种不同路由算法下不同 Pod 间的流量负载对比图

Fig.5.3　Traffic load comparison towards deferent Pods under 3 routing algorithms

5.3　数据中心网络拥塞路由管理策略

数据中心网络的拥塞路由管理策略是对本文提出的路由策略的功能补充，用于网络中出现拥塞链路的情况，通过将拥塞链路上的流量分担给其他存在空闲带宽的链路来摆脱拥塞的状态。为了让网络中产生拥塞的链路，我们在实验过程中加大了主机发送的流的数量，并通过控制网络中大流所占流总数的比例，调整拥塞发生的概率。实验的结果分析主要比较有无拥塞路由管理机制的网络的吞吐量。如图 5.4 所示。

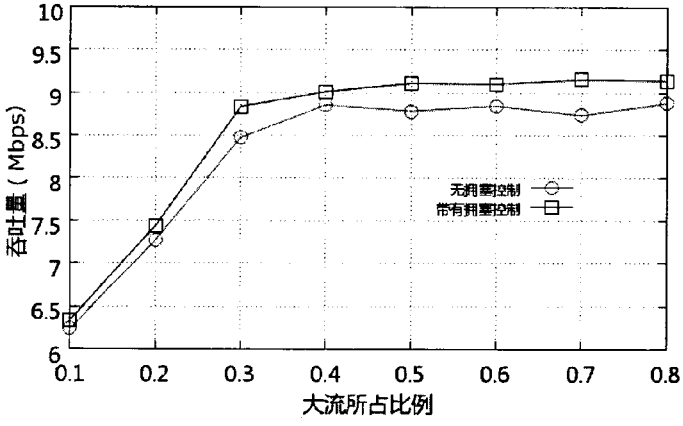

图 5.4　伴随大流增长的两种策略下吞吐量的比较

Fig.5.4　Throughput comparison between two strategies with increasing big flows

横轴表示容易造成拥塞的大流占网络中总流数的比例。与不采取拥塞控制的策略相比，带有拥塞控制的策略会提高网络的吞吐量。随着网络中大流数量的增加，网络的吞吐量在开始阶段快速增加；当大流所占比例达到0.3左右的时候，网络发生拥塞，吞吐量增速开始放缓。带有拥塞控制的策略在0.4左右吞吐量依旧可以有小幅度的增长；随着大流的比例越来越大，网络的链路吞吐量达到饱和。

5.4 本章小结

本章是本文的实验部分，对三四章所提出的路由策略和拥塞策略进行实验验证，并对结果进行分析。首先，介绍了实验平台的搭建，然后阐述了实验的设计思路及实验参数设置。最后对实验的结果进行分析。实验通过对链路利用率、流量负载分布和吞吐量等方面，比较了三种路由策略的性能，以及拥塞控制的性能。实验的最终结果表明，本文提出的基于流分类的动态路由策略要由于另外两种策略；面向流的拥塞控制策略对于网络的吞吐量有一定的提升。

结　论

随着互联网的快速发展，云计算的出现变革了互联网应用的服务模式。各大互联网应用提供商纷纷建立数据中心为用户提供各种各样的网络应用。大量新兴的互联网应用的出现对网络需求提出了新的要求。传统的网络体系架构对这些新的要求表现出了不适应。数据中心网络也同样暴露了一些问题，如带宽利用不足，无法达到更高的网络吞吐两。本文主要针对数据中心网络的路由策略进行研究，以达到对网络中的数据流进行更合理的带宽分配，主要工作内容为以下几个方面。

首先，本文对新一代互联网体系架构 SDN 和其关键技术进行了研究，了解 SDN 架构对解决现有网络中的问题的技术优势。并进一步对数据中心网络的现有的路由策略进行研究，发现现有路由策略的优势与不足。

其次，针对现有数据中心网络路由策略存在的不足，本文提出了一种基于流分类的数据中心网络动态路由策略。本文的路由策略以 OpenFlow 技术为基础来实现，通过 OpenFlow 技术获得网络的链路和流的信息，为数据流计算路由。在对流进行路由时，对流进行分类处理，对网络中新的数据流依据贪心原理选择路由。针对网络中的大数据流做动态地调整，为大数据流分配更加合适的带宽，从而提高网络的链路利用率，降低拥塞发生的概率。

再者，在提出的基于流分类的数据中心网络动态路由策略的基础上，提出了一种面向流的拥塞路由管理策略，作为路由策略的功能补充，以一种主动的方式尽最大努力摆脱拥塞的状态。当拥塞发生时，对网络中的新数据流路由时采取回溯的方式进行路由，来避开发生拥塞的链路；对在拥塞链路上转发的流，对其中的大流进行重路由，将负载转移到其他有足够空闲的链路上，同样采取回溯的方式进行新路由的计算。

最后，本文在 Mininet 仿真实验平台上对本文提出的两种策略进行实验和结果分析。通过与现有的两种路由策略 ECMP 和 DLB 路由策略的实验比较，本文提出的基于流分类的动态路由策略在平均二分带宽、链路带宽利用率和网络流量负载分布方面都要优于上述两种路由策略。对本文提出的面向流的拥塞路由管理策略的实验，通过增大大数据流再留总数中占的比例，比较有无拥塞路由管理策略两种条件下的网络的吞吐量，实验结果表明在加入拥塞路由管理策略后，网络的吞吐量得到了一定的提升。

参 考 文 献

[1] 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.

[2] JAIN S, KUMAR A, MANDAL S, et al. B4: Experience with a globally-deployed software defined WAN[C]//ACM SIGCOMM Computer Communication Review. ACM, 2013, 43(4): 3-14.

[3] WROCLAWSKI J. GENI: Global Environment for Network Innovations [EB/OL]. http://www.geni.Net, 2011-03.

[4] FIND [EB/OL]. http://www.nets-find.net/, 2011-01-12.

[5] 吴建平,李星,刘莹.下一代互联网体系结构研究现状和发展趋势[J]. 中兴通讯技术, 2011, 02:10-14.

[6] CNGI 背景 [EB/OL]. http://www.cngi.cn/cngijj/cngijs/cngibj/271327.shtml, 2011-02-18.

[7] 王文东,胡延楠. 软件定义网络:正在进行的网络变革[J]. 中兴通讯技术,2013,01:39-43.

[8] GREENBERG A, HJALMTYSSON G, MALTZ D A, et al. A clean slate 4D approach to network control and management[J]. ACM SIGCOMM Computer Communication Review, 2005, 35(5): 41-54.

[9] YANG L, DANTU R, ANDERSON T, et al. Forwarding and control element separation (ForCES) framework[R]. RFC 3746, April, 2004.

[10] LAKSHMAN T V, NANDAGOPAL T, RAMJEE R, et al. The softrouter architecture[C]//Proc. ACM SIGCOMM Workshop on Hot Topics in Networking. 2004.

[11] 薛淼,符刚,朱斌,等. 基于SDN/NFV的核心网演进关键技术研究[J]. 邮电设计技术, 2014, 03:16-22.

[12] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[13] AL-FARES M, RADHAKRISHNAN S, RAGHAVAN B, et al. Hedera: Dynamic Flow Scheduling for Data Center Networks[C]//NSDI. 2010, 10: 19-19.

[14] 李丹,陈贵海,任丰原,等. 数据中心网络的研究进展与趋势[J]. 计算机报, 2014, 02: 259-274.

[15] LEISERSON C E. Fat-trees: universal networks for hardware-efficient supercomputing[J]. Computers, IEEE Transactions on, 1985, 100(10): 892-901.

[16] GREENBERG A, HAMILTON J R, JAIN N, et al. VL2: a scalable and flexible data center network[C]//ACM SIGCOMM computer communication review. ACM, 2009, 39(4): 51-62.

[17] GUO C, WU H, TAN K, et al. Dcell: a scalable and fault-tolerant network structure for data centers[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(4): 75-86.

[18] GUO C, LU G, LI D, et al. BCube: a high performance, server-centric network architecture for modular data centers[J]. ACM SIGCOMM Computer Communication Review, 2009, 39(4): 63-74.

[19] LI D, GUO C, WU H, et al. FiConn: Using backup port for server interconnection in data centers[C]//INFOCOM 2009, IEEE. IEEE, 2009: 2276-2285.

[20] ALIZADEH M, GREENBERG A, MALTZ D A, et al. Data center tcp (dctcp)[J]. ACM SIGCOMM computer communication review, 2011, 41(4): 63-74.

[21] RAICIU C, PAASCH C, BARRE S, et al. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP[C]//NSDI. 2012, 12: 29-29.

[22] HOPPS C E. Analysis of an equal-cost multi-path algorithm[EB/OL]. 2000.

[23] HANDIGOL N, HELLER B, JEYAKUMAR V, et al. Reproducible network experiments using container-based emulation[C]//Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, 2012: 253-264.

[24] 左青云,陈鸣,赵广松,等. 基于 OpenFlow 的 SDN 技术研究[J]. 软件学报, 2013, 05:1078-1097.

[25] OpenFlow Switch Spwcification Version 1.0.0 Implement[EB/OL]. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf, 2009-12-31

[26] NOX[EB/OL]. http://www.noxrepo.org/nox/about-nox/, 2014-03-16.

[27] POX[EB/OL]. http://www.noxrepo.org/pox/about-pox/, 2014-03-18.

[28] Floodlight[EB/OL]. http://www.projectfloodlight.org/floodlight/, 2015-3-20

[29] Opendaylight[EB/OL]. http://www.opendaylight.org/, 2015-3-22

[30] ERICKSON D. The beacon openflow controller[C]//Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013: 13-18.

[31] Maestro: A System for Scalable OpenFlow Control [EB/OL]. http://www.cs.rice.edu/~eugeneng/papers/TR10 - 11.pdf, 2014-02-21.

[32] AL-FARES M, LOUKISSAS A, VAHDAT A. A scalable, commodity data center network architecture[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(4): 63-74.

[33] BENSON T, ANAND A, AKELLA A, et al. MicroTE: Fine grained traffic engineering for data centers[C]//Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies. ACM, 2011: 8.

[34] XI K, LIU Y, CHAO H J. Enabling flow-based routing control in data center networks using probe and ECMP[C]//Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. IEEE, 2011: 608-613.

[35] KODIALAM M, LAKSHMAN T V, SENGUPTA S. Efficient and robust routing of highly variable traffic[C]//In Proceedings of Third Workshop on Hot Topics in Networks (HotNet). 2004.

[36] KANDULA S, SENGUPTA S, GREENBERG A, et al. The nature of data center traffic: measurements & analysis[C]//Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. ACM, 2009: 202-208.

[37] LI Y, PAN D. OpenFlow based load balancing for Fat-Tree networks with multipath support[C]//Proc. 12th IEEE International Conference on Communications (ICC' 13), Budapest, Hungary. 2013: 1-5.

[38] BENSON T, ANAND A, AKELLA A, et al. Understanding data center traffic characteristics[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(1): 92-99.

[39] KANAGAVELU R, MINGJIE L N, MI K M, et al. OpenFlow based control for re-routing with differentiated flows in Data Center Networks[C]//Networks (ICON), 2012 18th IEEE International Conference on. IEEE, 2012: 228-233.

[40] 樊自甫,伍春玲,王金红. 基于 SDN 架构的数据中心网络路由算法需求分析[J]. 电信科学, 2015,02:42-51.

攻读硕士学位期间发表学术论文情况

研究生期间，申请国家发明专利一项，并已获得受理。
《一种面向 Fat-Tree 数据中心网络架构的数据流转发方法》李克秋，**王珣**，齐恒
申请号 201510162959.X （本硕士论文第 3 章）

致　谢

写到这里，论文即将完成，我的三年硕士研究生的生活也接近尾声。回顾这三年的研究生生活，有过成功也有过失败；有过欢笑也有过泪水。感谢在我身边陪我一路走来的老师、同学、亲人、朋友。

首先要感谢的，是我的导师李克秋教授。李老师治学严谨，在学术研究方向上为我严格把关，总能够准确地发现问题的所在，并给出正确的方向。除了在学术研究上，李老师在生活中的其他方面也会给我很多宝贵的建议与教诲，从李老师身上我学到了很多做人的道理。感谢李老师这三年来给予的教导与帮助。

同时感谢实验室的齐恒老师。齐恒老师作为我所在课题组的负责老师在科研工作上给予了我跟多的帮助，提出过很多建议与指导。齐老师每周都带领课题组的同学进行学术上的讨论，建立了良好的组内学术氛围。齐老师为人谦逊和善，总是很认真的和我讨论论文中的问题，给予细心的指导。

感谢李文信博士，在论文的选题和内容上给予我的建议与指导；感谢喻海生博士，在论文的实验过程中给予我的帮助与建议。他们的帮助与建议对论文的完成起到了至关重要的作用。

感谢软件定义网络课题组的同学们，我们在一起完成项目的经历是我宝贵的人生财富。我们一起经历过项目过程中的困难，大家共同努力奋斗排除障碍，让我感受到了团队合作的能力与精神。

感谢网络与云计算实验室的全体同学，在这三年时光我感受到了同学们营造的融洽的学习氛围；感谢这三年来在学校里帮助过我的所有同学与老师，谢谢你们。

最后感谢我的父母、家人和朋友，是他们在我背后对我的支持，让我顺利完成了三年的硕士学业。

大连理工大学学位论文版权使用授权书

本人完全了解学校有关学位论文知识产权的规定，在校攻读学位期间论文工作的知识产权属于大连理工大学，允许论文被查阅和借阅。学校有权保留论文并向国家有关部门或机构送交论文的复印件和电子版，可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、或扫描等复制手段保存和汇编本学位论文。

学位论文题目：基于OpenFlow的数据中心网络路由策略研究与设计

作者签名：王珣　　日期：2015年6月9日

导师签名：李七丹　　日期：2015年6月9日