基于软件定义网络的流量工程*

周桐庆+, 蔡志平, 夏 竟, 徐 明

(国防科学技术大学 计算机学院,湖南 长沙 410073)
通讯作者:周桐庆, E-mail: zhoutongqing@nudt.edu.cn

Traffic Engineering for Software Defined Networks*

ZHOU Tong-Qing+, CAI Zhi-Ping, XIA Jing, XU Ming

(Department of Computer Science and Technology, National University of defense technology, Changsha 410073, China)
+ Corresponding author: E-mail: zhoutongqing@nudt.edu.cn

Abstract: Software Defined Networks (SDN) is an emerging network paradigm that proposes to decouple the control and data forwarding plane. Leveraging the centralized control ability and open programming interfaces SDN provides, network management can be dramatically simplified, and network services can be dynamically controlled by applications. Traffic engineering (TE) is a category of mechanisms that promises to optimize network performance through analyzing, predicting and regulating data flow in network. The unique features of SDN provide TE with a unified and real-time global network view, flexible control manner and better policy for scalable traffic management, exhibiting profound research significance. This paper surveys the state-of-art works on TE for SDN. With regard of the methods, application and deployment of measurement, works on traffic measurement architecture, network invariant detection and measurement resource management with SDN are concluded, respectively. The problems in traditional network traffic management are analyzed, and SDN-based data traffic management methods for efficient load balance and control traffic management methods for centralized bottleneck relief are presented. Moreover, network failure resilience technologies in SDN are concluded. Finally, the technological approaches are summarized and future research issues are discussed.

Key words: SDN; traffic engineering; traffic measurement; traffic management; network failure recovery

摘 要: 作为一种新型网络架构,软件定义网络(Software Defined Networking, SDN)将网络的数据层和控制层分离,通过集中化控制和提供开放控制接口,简化了网络管理,支持网络服务的动态应用程序控制.流量工程通过对网络流量的分析、预测和管理,实现网络性能的优化.在 SDN 中开展流量工程,可以为网络测量和管理提供实时集中的网络视图,灵活抽象的控制方式以及高效可扩展的维护策略,具有突出的研究意义.本文对基于 SDN 的流量工程相关工作进行了综述.分别从测量的方法、应用和部署角度出发,对 SDN 中流量测量的基本框架、基于测量的正确态检测以及测量资源的管理进行概述.分析传统网络流量调度方案的问题,介绍 SDN 中数据流量和控制流量调度的主要方法.从数据层和控制层两个方面概述 SDN 中故障恢复方法.最后,总结并展望了未来工作.

关键词: 软件定义网络;流量工程;流量测量;流量调度;网络故障恢复
中图法分类号: TP393 **文献标识码**: A

* 基金项目:国家自然科学基金61379144, 61379145, 61402513.

近年来,随着互联网规模不断扩大,互连设备数目和种类不断增多,传统网络在管理难度、可扩展性以及实验研究等方面的局限逐渐凸显出来[1],具体体现在:研究者很难在大规模网络环境中进行实验研究的验证;网管人员无法有效地根据自身的应用需求进行网络的配置和优化;设备厂商不能快速地进行研发和部署以满足客户需求[2].

作为一种新型的网络架构,SDN 基于网络抽象的思想,将网络中的控制层和数据层分离开来,提供对分布式网络的集中管理和动态维护能力,从而有效解决传统 IP 网络在维护、扩展和实验创新上的弊端.如图 1 所示,SDN 网络主要包含应用、控制和数据三个层面:控制层将网络各节点的控制功能从逻辑上集中;应用层利用控制层提供的可编程接口实现各种网络应用;数据层负责进行数据流的转发;OpenFlow[3]是控制层和转发层之间通信的标准接口,状态、数据以及控制信息通过这一接口实时地传递.

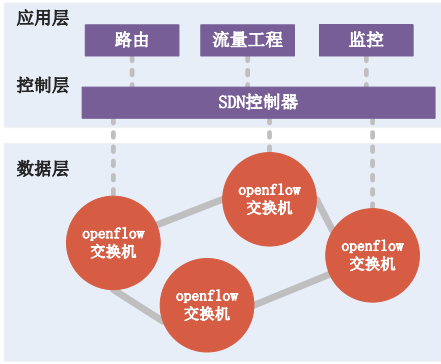


Fig.1 SDN architecture

图 1 SDN 分层结构

如图 1 所示,流量工程是 SDN 网络中的一类典型应用.流量工程是网络管理者优化网络性能和流量传输的一系列重要方法,主要内容包括对网络中的数据流进行动态的分析、预测和管理,其相应技术广泛地应用在公共电话交换网、局域网、广域网和互联网中.在网络快速发展和网络应用不断出现的新形势下,基于传统网络技术的流量工程具有局限性,具体可从服务的使用者和提供者两个角度加以概括:

1) 服务质量与可扩展性:网络应尽可能满足不同应用的传输需求,而且能够针对各个应用对延迟等性能指标的不同要求提供不同优先级的服务.例如,不断增多的多媒体应用对于端到端延迟、抖动以及丢包率等服务质量参数都提出更高要求[5].

2) 网络资源利用率和容错:云计算的发展使得大规模数据中心的需求愈发明显,网络管理过程中应更有效地利用网络资源,以降低成本、提高系统的总体性能[5].与此同时,网络设备的不断增多使得网络中出现故障的概率不断变大,需要流量工程为网络提供足够的容错和快速恢复能力.

SDN 的提出为集中和高效的网络管理提供了更好的途径,为流量工程的开展提供了全局的网络视野、实时的状态信息和清晰的网络流模式,易于实现及时准确的测量和动态灵活的调度.另一方面,传统网络的流量工程技术能否与 SDN 有效地兼容尚不明确[4].因此,开展基于 SDN 的流量工程研究是极有理论意义和应用价值的.

本文针对现阶段 SDN 中流量工程相关工作进行介绍和分析,第 1 节首先介绍基于 SDN 的流量工程的概念和内容,并分析存在的挑战,在此基础上分别利用第 2、3 和 4 节对流量工程中的流量测量、流量管理和故障恢复三方面工作进行详细阐述,最后,第 5 节对全文进行总结并对未来工作提出展望.

1 SDN 中流量工程概述

1.1 流量工程定义

流量工程,或称流量管理,指的是针对网络性能进行优化的一系列方法,即针对网络中数据流的行为进行动态的分析预测和有目的性地管理.图 2 给出了一个使用流量工程进行路由优化的例子,2 个用户流由节点 A 流向

节点 C,在图 2(a)中,使用最短路径路由(OSPF),导致链路 AC 拥塞,图 2(b)中利用链路 AB 和 BC 分摊 AC 间的流量,减轻或消除链路 AC 的拥塞状况,可见流量工程可以有效改善网络性能.

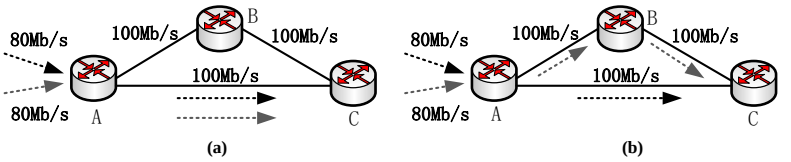


Fig.2 A toy example of using traffic engineering for routing optimization

图 2 使用流量工程进行路由优化的一个例子

针对不同的网络类型,流量工程技术不断发展以满足不同的需求.在上世纪 80 年代的 ATM 网络中,流量工程技术主要用来解决拥塞控制问题[6];90 年代末 IP 网络迅猛发展,流量工程被用来在 IP 网络中进行路由优化以均衡路径上的流量,并为应用提供服务保证[7];针对 IP 网络中优化路由的限制,研究者进一步提出了通过标签而非 IP 头部进行转发决策的多协议标签交换技术(MPLS)[8],实现更有效的流量管理.但是,MPLS 的控制层的实现和管理过于复杂,SDN 的提出恰好有效解决了这一问题.

SDN 的出现引起了学术界和产业界的广泛重视,例如,Google 期望通过使用 SDN 将网络资源的利用率提升 20-30%,同时降低网络的延迟和丢包[2].相比传统方式,SDN 为流量工程的开展提供了更好的基础:首先,提供了集中的全局视图,控制器能够实时得到全局的网络状态、拓扑信息和应用需求信息;其次,提供了可编程的数据层接口,操作者可以根据当前状态动态的对网络资源进行动态重分配;此外,多个设备厂商的交换机使用统一的编程接口,可以提供充分的开放性.总结起来,基于 SDN 开展流量工程的优势可概括如下:

1) 在流量的测量方面,可以灵活的部署可扩展的全局测量任务,实时收集网络状态信息,对流量进行准确的监控和统计分析;

2) 在流量的管理方面,可以综合考虑网络状态和网络应用的需求,以流为单位进行细粒度的、动态的流量控制,实现负载的合理调度;

3) 在网络的资源利用和维护方面,支持对包括带宽、存储等资源的动态分配,实现有效合理的资源利用.基于集中的网络状态反馈,可以透明的进行故障地应对和处理.

1.2 基于SDN的流量工程研究内容

流量工程的工作可以按照不同的角度进行划分[5].在研究基于 SDN 的流量工程时,本文按照功能将其分为:流量测量,流量管理和故障恢复三个部分,如图 3 所示.

1) 流量测量:流量测量是对一个特定网络中流量的规模、特征进行测量的过程,是有效实现网络管理的基础.一般说来,网络管理的预算至少占整个信息技术行业预算的 80%,而流量测量承担的任务至少占网络管理工作的一半（另一部分是控制）,因为明确发生了什么要比决定如何处理更为关键和困难.现有的许多 SDN 框架使用传统的 IP 架构的流量测量方案,部署复杂,硬件资源开销较大[14],而在控制器上实现复杂的监控和分析逻辑引入了额外处理延迟.所以,基于 SDN 的流量测量方法应该综合考虑复杂性、开销和准确性三方面因素.测量的目的在于获取流量特征,捕获网络的异常,如何利用流量测量进行有效的网络正确态检测也是该方向的重要内容.最后,流量测量过程中的硬件资源调度问题也是文中关注的一个研究点.

2) 流量调度:流量调度通过对网络中流量的控制和管理实现负载均衡,获得在延迟、吞吐率和带宽利用率等性能指标上更好的表现.利用 SDN 集中的网络视图和动态的规则配置能力,流量可以依据后续路径的占用情况得到合理的转发,实现网络负载的有效均衡.在 SDN 网络中,流的默认管理操作[9]为:首先,如果一个流与交换机流表中的任何规则都不匹配,其首个报文将由交换机送至 SDN 控制器;控制器在接收到报文后,计算对应流在网络中的转发路径;控制器向转发路径上的各个交换机安装更新规则,交换机相应的更新自身的流表;流的后续报文以及其它的所有能与该规则匹配的流按照此规则在数据层进行转发,不再需要控制器介入.然而,如果在同一阶段,大量需要进行规则更新或安装的流到达交换机,会带来控制器南向接口路径的拥塞,另一方面,控制

器需要为这些流计算新的规则,会引起额外的延迟.所以,基于 SDN 的流量调度工作主要围绕的是:如何发挥 SDN 优势进行有效地负载均衡,同时降低集中控制引入的延迟和管理开销.

Fig.3 The scope of traffic engineering in SDN

图 3 SDN 网络中流量工程的主要内容

3) 故障恢复.可靠性是当前云计算和数据中心网络中很关键的一项性能.如果网络中设备（控制器、交换机、链路等）出现故障,SDN 需要提供透明地、快速地故障恢复.在 OpenFlow1.1 中,针对链路和节点的故障提出了一个快速故障恢复机制,该机制允许在当前规则之外指明一条备份的路径,交换机可以在不引起控制器介入的情况下使用备用路径继续转发.尽管 SDN 为故障的检测和恢复提供了集中的视图,在 SDN 中实现快速恢复依然具有挑战性,因为控制器需要计算新的转发策略并更新所有受影响的交换机.此外,故障恢复过程需要考虑交换机中存储和流表的限制.

以上三个部分构成了网络管理的一个闭环,底层的测量工作负责监控流量,将网络信息反馈给调度模块进行决策和具体的流量控制,故障恢复作为与调度并行的模块利用实时监测信息进行故障的发现、补救和及时处理.

在接下来的 3 节中,将分别对流量工程中的流量测量,流量调度和故障恢复这三方面的技术进行综述.

2 基于 SDN 的流量测量

流量工程的开展依赖于对网络的准确实时的流量测量,而云端、数据中心中涌现的多种类、大规模网络应用为测量任务有效运行提出了挑战.基于 SDN 的流量测量的研究工作主要围绕如何支持、设计和部署灵活准确可扩展的测量任务,为相关应用提供有效状态信息,这里对相关工作进行如下分类:

1) 从流量测量的实现角度,设计抽象通用的测量框架.OpenFlow 为集中分配任务和监控提供了良好的接口,然而传统 IP 网络中的测量方法为控制器带来了流量压力,所以需要设计面向 SDN 的灵活、通用的测量方式.

2) 从流量测量的应用角度,设计准确、及时的网络正确态检测方法,应对路由配置和规则兼容性方面可能存在的问题.

3) 从流量测量的开销控制角度,设计有效的资源管理方案.网络设备中用于测量工作的计算、存储资源有限,所以需要对其进行合理的利用和有效的管理.

本节首先对网络流量测量的传统方法进行简述,然后介绍和分析基于 SDN 的流量测量框架,正确态检测方法和资源分配管理方案.

2.1 流量测量框架

流量测量主要可以分为主动和被动两类方法,主动测量需要额外的探测流量,被动测量通过采样的方式对流经测量设备的流量进行观察.许多 SDN 测量系统使用传统 IP 网络中基于包采样的流量监控方法,利用网络中交换机以一定概率抓取流经本地的包进行信息统计.其中,应用最为普遍的就是思科的 NetFlow[10],NetFlow 规定使用 5 元组标识一个流,交换机维护一个流缓存记录每个流的信息,同时对到达的流的头部与本地缓存中的表项进行比较,如果匹配则更新该流对应条目的包计数以及比特计数,否则在缓存中为新的流增设一个条目.另一个流采样方法是 InMon 的 sFlow[11],sFlow 以一定比例在路由的每个入口处进行包采样,进而将采样数据包头

与时间戳等原信息封装之后发送给中心收集者.JFlow[12]是一个专用的采样测量方案,思路基本与 NetFlow 一致.大规模网络的测量与信息收集增加了中心控制器的负担与开销,所以以上方法应用于 SDN 网络中效率低.

Table1 Traffic measurement methods in SDN

表 1 SDN 中流量测量方法

方法名称	目标	方法类别	流量工程技术	优点
OpenTM	利用直接询问测量流量信息	基于询问的测量方式	对活跃流的信息进行周期询问以维护一个流量矩阵	对于短周期:高准确性和大的开销;对于长周期:低准确性和小开销
PayLess	设计通用的测量框架	基于询问的测量方式	根据实时的流量情况进行频率自适应的交换机询问	高的准确性和大的开销
FlowSense	利用控制消息设计低开销的测量方法	基于流量的被动测量方式	利用 OpenFlow 的控制消息进行网络链路利用率的评估	高的准确性和较低开销
OpenNetMon	网络参数的准确测量	基于询问的测量方式	使用自适应询问的方式对流的吞吐率等具体参数进行细粒度的测量计算	细粒度的参数测量方法
OpenSketch	设计抽象、可配置的测量框架	基于询问的测量方式	将测量分离为转发和控制两部分,前者提供可配置的数据层功能流水,后者提供任务库,提供通用的测量框架	低存储开销,高准确性,需要硬件支持
MicoTE	利用数据中心流量特征提升流量管理性能	基于流量的被动测量方式	在数据中心机组的服务器中增加监控模块,提供对流量变化的前瞻性响应,并提供充分的扩展能力	低的网络开销
OpenSample	利用协议特征设计低开销、细粒度的测量方法	基于流量和协议特征的被动测量方式	在 sFlow 方法中引入 TCP 数据包头部的序号这一新特征,进行速率估计,实现对 elephant 流的快速发现	低测量延迟,高准确性

针对传统测量方案的诸多局限,SDN 中流量测量方法的研究主要围绕如何设计通用的测量框架,如何有效的汇总数据层流信息,以及如何进行网络参数估计三方面工作,表 1 总结了相关工作.以下分别展开分析:

2.1.1 网络流信息的采集

交换机利用自身资源进行网络中流信息的收集和记录,控制器对收集的信息进行汇总、分析,为流量调度提供参考.折衷考虑信息汇总的准确性和开销,研究者提出了两类信息采集方法:主动式[13][14]和被动式[15][18][20],其中被动式又包括基于事件触发[15][18]和基于时序触发[20]两种.具体的:

(1) 主动式信息采集

作为 SDN 的南向接口协议,OpenFlow 支持控制器对交换机发起主动查询获取流信息.利用这个特性,文献[13]提出 OpenTM,一个针对 SDN 的 TM(traffic matrix,流量矩阵)估计系统——TM 表示的是网络中一个源节点和目标节点对(OD 对)之间的流量大小,对 TM 的准确评估是负载均衡,异常检测,路由配置等网络操作和任务的基础.OpenTM 对网路中所有活跃的流进行探测,根据控制器中的路由信息和流的转发路径信息,提出多种有选择性的路由节点询问方法,从而获得准确的流量大小评估和数据包计数信息,并通过将一个 OD 对的所有流累加起来更新 TM.

文章指出:流量测量的准确性与测量过程中使用询问消息所引入的测量开销之间存在一个权衡关系,即越准确的测量结果,意味着越大的测量开销.文中方法的缺点在于:控制器主动的进行周期询问带来开销,而另一方面,主观的选择一部分交换机进行询问会损失测量的准确性.

围绕着测量开销与测量时效性和准确性之间的权衡,文献[14]对直接探测和被动采样测量进行折衷,提出一种灵活的主动式信息采集方法 Payless.该方案中,控制器使用动态频率的询问方式进行统计数据的请求,而非 OpenTM 中的使用的固定询问频率.此外,针对多个同时超时的流的询问,提出使用批处理的方式进行合并优化,使得询问的开销得以进一步降低.实验中使用链路利用率(瞬时的吞吐率)和开销(单位时间的询问消息数目)对方案进行了评价,分析发现相比于固定频率方案中 13.3 个/s 的询问开销,Payless 引入的询问开销为 6.6 个/s.总的来看,Payless 通过引入可观的测量开销,提高了测量的准确率.

(2) 被动式信息采集

主动式流量测量在保证准确性的同时,不可避免的引入测量开销,降低了带宽利用率.相比而言,被动式测量使用时间或事件触发机制,由交换机自主的向控制器反馈所采集的流量信息.

文献[15]提出一种被动测量方案 FlowSense,利用 SDN 中的控制流进行流量变化的实时监测.基于流的网络监控要求能够准确及时的探测,同时要求测量任务带来对网络尽可能小的开销.FlowSense 利用 OpenFlow 协议中规定的南向消息接口的异步消息 PacketIn 和 FlowRemoved(前者表示一个新的流的到达,后者表示一个流的结束),进行数据层中流状态变化的监控,进而计算交换机之间链路的利用率.文中采用一个由两个 OpenFlow 交换机和一个控制器构成的简单测试环境对方案进行了验证,结果表明相比于如 OpenTM 采用的询问方式,FlowSense 有更高的准确性,没有引入任何额外流量开销,同时可以在 10s 之内完成几乎全部的链路利用率评估.然而,采取这种基于特定消息的被动监控方式,使得方法只能在几个离散时间点进行速率的估计,对于一个持续时间较长的流,无疑降低了获取流信息的时效性.

文献[18]中提出的方法 MicroTE 使用机组中的服务器进行流状态监测和被动式的流信息统计,并在流状态发生显著变化时触发一次信息的汇总行为,根据流的平均流量值和瞬时流量值之间的偏差大小,决定其是否趋于平稳可预测[19],进而由控制器集中的制定路由转发决策.

在文献[20]中,IBM 提出一种利用协议信息改进 sFlow 的被动式流量测量方案 OpenSample.文章指出使用包采样的 sFlow[11]技术在测量的准确性上受到采样率的限制,而在提高准确率的两种直接方法中,提高采样概率会增加网络设备 CPU 的负担,降低采样间隔与方案设计的低延迟的初衷相违背.OpenSample 基于一个观测统计的事实——数据中心的流量中 99%都是 TCP 流[21],提出通过计算同一个流中相邻两个采样包中 TCP 头部的序号差,进一步除以 sFlow 采样数据中自带的对应采样时间戳的差值,从而得到 TCP 流在该测量窗口中的准确速率值.实际过程如图 4 所示,其中 S_i 和 T_i 分别表示测量获得两个流在时间点 i 上的 TCP 序号,t 是采样的时间,两个流的速率可以分别表示为$(S_2-S_1)/(t_3-t_1)$和$(T_2-T_1)/(t_4-t_2)$.通过被动测量以及引入协议相关信息,方案有效的提高了准确率并降低了延迟.OpenSample 在监控时将任何具有超过两个样本的 TCP 流视作一个 elephant 流,而 elephant 流是流量工程所关注的重点,也是 DDoS 等安全检测的关键,所以 OpenSample 相比于 MicroTE[18]等方法而言对更多的流进行了测量、优化与管理.

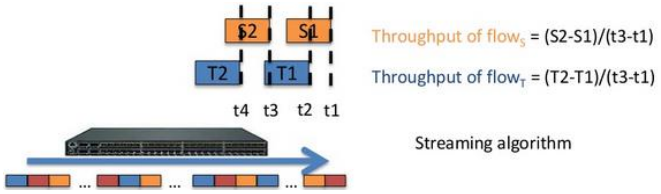


Fig.4 Throughput estimation based on TCP-specific information in OpenSample

图 4 OpenSample 中利用 TCP 信息进行吞吐率评估

2.1.2 网络参数评估方法

网络延迟、带宽和丢包率等是网络管理和网络服务质量保障所需的重要性能指标.文献[16]提出的 OpenNetMon 给出几种网络参数的测量方法,它被设计为 POX 控制器的一个模块,可以支持细粒度的流量工程.模块以自适应的频率询问交换机获取流信息,同时保证获得准确的测量结果和最小化交换机 CPU 的开销.OpenNetMon 持续的对所监控网络的吞吐率、丢包率和延迟进行测量.具体的:

1) 测量吞吐率时,仅对路径的最后一个交换机进行自适应频率的询问,数据层的计数器返回采样间隔 T 中各个流的包数目 S,通过计算 S/T 得到路径上一个流的吞吐率;

2) 计算丢包率时,分别询问一个路径的第一个交换机和最后一个交换机,获得两者计数器的统计数据,用测量周期中第一个计数器的增量减去最后一个计数器的增量,除以时间就是该阶段以两个交换机为端点路径的丢包率;

3) 计算延迟时,OpenNetMon 向路由的数据层注入探测报文,探测报文通过路径、节点和控制器与路径两个端路由之间的链路,最后再次返回控制器,利用这个延迟减去控制器和交换机之间的通信延迟即获得所探测路径的通信延迟.

FlowSense[15]在 SDN 基础上增加了监控模块,通过解析控制器接收的消息,动态分析流量变化情况.例如,控制器在收到对应于某一个流的 FlowRemoved 消息之后,利用统计得到的流的大小值除以对应流在网络中存在的时间,即可得到流的吞吐率信息.

总的来看,传统的网络测量可以位于 OSI 的各个层次,即各个层次都对应着可以反应网络情况的参数,其中应用层和网络层是两个比较普遍的选择,而一般的参数评估都采取网络层测量,使用路由器或交换机进行监测.然而,网络层获得的测量数据都是针对设备端口计数器而言的(如 SNMP 中,通过周期性的询问交换机,可以获得每个端口的计数信息),不能针对单独的应用流进行测量.OpenFlow 支持针对每一个流进行数据统计,从而决定端到端网络的性能.另一方面,已有的测量技术都基于额外的硬件或软件配置的支持,开展测量任务需要较大的开销.SDN 将传统的分布式控制逻辑进行集中,无需额外的配置与硬件开销,即可高效的完成测量工作.

2.1.3 通用流量测量框架

如表 1 所示,面向不同的应用需求和测量目标,研究者提出多种基于 SDN 的流量测量方法,为了有效地支持多种测量任务和应用程序,设计一个通用的测量框架是有必要的.

文献[14]中提出的 PayLess 是一个基于询问的,低成本的流量测量框架.PayLess 框架位于 OpenFlow 控制器的北向接口之上,并为应用提供 RESTful 风格的接口.具体的,测量框架负责解析应用发出的请求级的任务,将其解析为对部分交换机的询问规划,同时负责将反馈到控制器中的统计信息进行汇总.所以,Payless 为应用维护了一个抽象网络信息视图,为多种网络应用提供了一个良定义的编程接口——应用以 JSON 格式将任务下达,并支持将用户自定义的构件加入到测量框架中.

文献[17]提出一个通用的、抽象的测量框架 OpenSketch,使用一种类似于 OpenFlow 中自定义转发的软件定义方法进行流量测量.文章指出:基于流的测量存在计数器开销和准确率的问题,且无法满足不同用户的需求;另一方面,基于 sketch 的方法针对用户定制,然而任务与设备的耦合度太高,不同任务对硬件有不同要求,所以很难将多个任务的解决方法部署到同一硬件设备中.为了充分权衡通用性和效率,OpenSketch 提出将测量的控制和数据层进行分离:数据层设计为一个可动态配置的 3 阶段流水线,每一个阶段都使用商用交换机组建构成.如图所示,第一阶段提供多种哈希算法将数据包映射为少量的测量数据,第二阶段利用 TCAM(Ternary Content Addressable Memory)进行规则存储和通配符式的匹配,第三阶段通过 SRAM 提供灵活的计数.通过这种流水线的设计,OpenSketch 将多种测量算法抽象成若干通用的步骤;另一方面,在控制层中,提供一个测量任务库,根据用户或应用的选择调整数据层流水线的各个环节,支持更多种询问类型,使得用户可以在商用交换机组件上应用新的测量算法.需要注意的是,OpenSketch 的实现需要对交换机的部分硬件进行重新设计,这是网络运营商不愿意选择的.

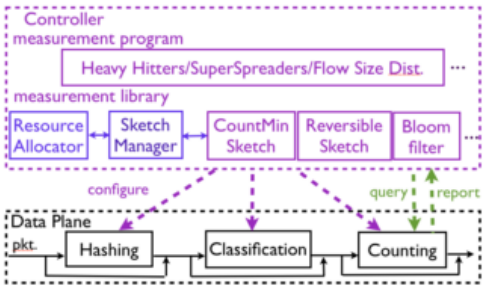


Fig.5 OpenSketch Architecture

图 5 OpenSketch 的框架图

2.2 网络正确态检测

网络正确态指的是在对数据进行转发的过程中,网络所表现出的正确的行为和状态,如数据包不会陷入路由环路.数据在网络中的转发依赖于各类网络设备所提供的多种功能,过程复杂且容易受到潜在的配置错误,软件缺陷等问题的影响,导致包括循环路由在内的各种错误.SDN 简化了网络应用,但是软件自身具有的复杂性导

致错误依然难以避免.此外,SDN 支持多个应用对同一个物理网络的流量进行共享的配置,导致配置规则存在冲突的可能,影响网络性能.因此,对 SDN 的正确态检测与规则验证是测量工作中的重要部分,目前的研究主要集中在设计、运用不同的检测方法进行正确态的测量.

2.2.1 模型检测方法

通过设计维护一个问题搜索模型,模型检测法对网络应用、配置规则中潜在的破坏正确态的问题进行检测.文献[22]提出一个结合模型检查和符号化执行（程序中有变量符号）的网络正确态验证工具 NICE,目的是找出 OpenFlow 应用中的错误（举例而言,控制器响应新到达网络的流,产生一条规则,此时如果多余的包也到达则可能超出程序的预期,引起应用的错误）,维护网络应用的可靠性.NICE 将 OpenFlow 程序、网络状态和一个正确态作为输入,利用模型检查对状态空间进行搜索,验证各个状态下的正确性,最后输出破坏正确态报告.通过使用 NICE,OpenFlow 程序可以对路由循环,空洞等问题进行检查.在实现中,NICE 采用 Python 语言以无缝的支持 OpenFlow 控制程序.

2.2.2 静态检测方法

静态检测的方法将收集的网络状态形式化,分析判断其是否破坏指定的正确态.文献[23]指出已有的正确态检测工作中,分析过程对控制器下发的配置文件进行检查,无法分析路由软件中的问题,且很难应付不同厂商的多种设备配置语言.文章提出可以将诊断过程尽可能的靠近网络的数据层次,更直接的观察网络行为,从而发现配置生效之前无法发现的问题,并支持对多种协议和语言环境的统一分析.

根据以上分析,[23]提出 Anteater,一个基于数据层静态分析的正确态检查工具.它的主要工作流程为:首先将网络的连接状态和交换机中的转发信息(FIB)转化为布尔表达式,然后将三类正确态——无环路由、网络连通性、规则的一致性表达成 SAT 问题,选择一个正确态,与转化为布尔表达式的网络状态与转发配置一起带入 SAT 求解器中进行分析,如果存在一个破坏正常态的配置,Anteater 会发出一个特殊报文来触发相应的问题.在真实的测试中,Anteater 成功的在所部署的校园网中发现 23 个网络问题.

2.2.3 实时检测方法

实时的对流转发规则实现检测,对于发现规则冲突,维护正确转发状态是非常重要的.文献[24]提出 VeriFlow,一个面向 SDN 网络进行实时检查和调试的工具.设计中将 VeriFlow 插入在 SDN 的控制层和数据层之间,实现对配置到网络中的每一条规则的监控和检查,更新的预先检查使得冲突可以提前被发现并发出及时的警告.在更新和删除规则时,VeriFlow 通过 3 个步骤实现对规则的快速验证:首先,将网络中的包分类为一个对等类（经历相同的规则-动作的包）集合,并以对等类为原子单位进行对规则的验证;然后为每一个对等类建立转发图;最后,设计一个检测算法,将更新规则与对应的对等类转发图作为输入（Nice 和 Anteater 将网络状态和一个待检查的正确态作为输入）,通过深度优先的方式遍历所有受更新影响的对等类中的路径,最后判断一次规则更新是否会产生网络异常.实验结果显示,使用 VeriFlow 可以在数百微秒的时间开销内完成对网络正确态的检查.

2.3 测量资源的管理

如前文所述,流量信息的采集和汇总是流量测量的两个重要部分,信息采集过程中需要来自数据层的硬件支持—TCAM(存储对流进行监控的规则).然而现有交换机的片上资源往往是有限的,此外,网络应用有同时进行多个流量监控任务的需求,而非在一段时间内仅支持一个测量任务.所以,如何合理的调度分配硬件资源,提供准确地、有效地的流量测量是十分重要的.

2.3.1 测量信息的存储调度

针对资源有限和多任务需求,文献[27]提出 DREAM,一种自适应的硬件资源分配调度方案,可以有效权衡测量的准确性和资源开销.方案依赖两个观察:1)流量测量任务的准确性依赖于分配给任务的资源量(TCAM 数).随着分配给任务的资源数目的增加,会出现一个收益递减点,在该点之后,增加资源量对任务准确性的提升能力变小,换言之,准确性提升相同程度需要的资源量更多;2)测量任务所需要的资源数目随着时间和所关心的流的大小在不断变化,所以,可以对各个探测任务进行时间和空间上资源(TCAM)分配的复用.

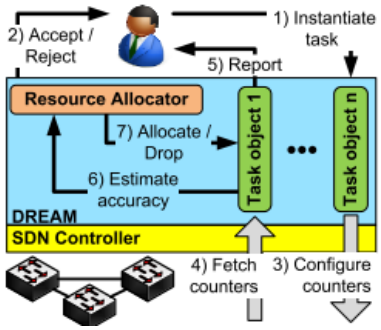


Fig.6 DREAM Architecture

图 6 DREAM 方案的框架图

DREAM 框架主要包括三个层次,最上层的是用户层,负责生成一个测量任务,包括任务类型、具体的流量阈值和精确度要求等,并作为实时检测结果汇总的终点;中间层次是运行于 SDN 控制层之上的 DREAM 算法,它负责接收来自用户的任务并创建对应的任务对象,通过 SDN 的控制接口将抽象任务部署到多个交换机中,部署过程就是在交换机中请求TCAM 资源,并将测量逻辑存储在TCAM 中完成实时的测量.根据实时反馈的流量监测情况,算法对测量的准确性进行评估,调整为各个任务分配的资源,进行多个任务之间的资源复用;另一方面,根据当前所有任务可以达到的准确性情况,决定是否接受一个新的任务;最底层是 SDN 网络,控制层进行集中控制和动态配置,转发设备为测量提供存储资源,并反馈实时流计数的结果.

2.3.2 流量计数的优化

文献[28]指出:交换机中的集成电路使用的硬件计数器缺乏灵活性、占用较大空间且无法有效开展新的测量方法,针对这一问题,文章提出利用软件自定义的思想改进 ASIC(Application specific Integrated Circuit)的设计,将传统的 ASIC 内部硬件计数改为利用基于 CPU+DRAM 的软件自定义计数（SDC）.

SDC 是一个软件自定义的计数方案.[28]指出一个“聪明控制器,愚蠢交换机”的 SDN 模型过于极端,尤其是在交换机中搭载高性能嵌入式 CPU 成本越来越低的情况下,所以提出利用交换机本地的 CPU 降低 ASIC 的复杂性.SDC 的计数流程为:ASIC 利用触发的方式将小容量流信息记录缓冲中的信息通过总线发送给 ASIC 外部的通用 CPU,CPU 对信息流进行处理并利用 DRAM 保存流计数,控制器访问 CPU 而非 ASIC 获得统计信息.SDC 的收益可以分为 3 方面:首先,增加了处理计数器相关信息的灵活性;其次,降低了控制器获取计数器信息的开销;最后,由于将计数部件移除所以节省了 ASIC 宝贵的空间,并降低了复杂性.利用这些优势,SDC 为灵活的测量提供了更有效的硬件资源支持.

3 基于 SDN 的流量调度

随着数据中心基础设施的不断扩展,以及网络技术的不断变化,新兴网络应用越来越依赖有效地网络管理来提供高可用性、高质量、低成本的服务.SDN 不仅为快速的实验创新提供了一个合适的平台,也改变了管理网络的方式——取代以往对网络“片”式的管理,SDN 通过集中控制器进行面向全局的策略部署,同时动态的收集网络状态.此外,SDN 提供了对网络事件更快反应的能力,从而降低了网络管理的延迟[28].

网络管理旨在维护网络可用性和提高网络性能,对网络流量的合理调度(路由优化),是有效分配网络资源,提升网络性能和服务质量的重要途径.数据中心网络中源和目的节点之间存在多条路径,为流量调度提供了空间,SDN 中控制器维护的全局视图反映了网络中各路径的使用情况,流量可以据此进行动态的转发以实现负载的均衡,另一方面,对流量转发规则的实时安装与更新增加集中控制器的压力,也带来了相应的处理延迟.所以,SDN 中的流量调度可以分为对数据层流量的调度和控制层内部流量的调度,前者主要利用 SDN 的集中控制功能改进传统网络中的流量调度方法,后者主要围绕如何缓解 SDN 中集中控制器带来的流量调度瓶颈问题.

3.1 数据层流量调度

合理的网络流量调度可以提高链路的利用率,避免网络拥塞,为应用提供有效的带宽支持,满足用户的需求.传统IP网络中使用最短路径路由(如OSPF)进行链路间负载的分配,交换机之间通过交换静态的链路权重计算出转发的最短路径,然而,基于权重的路由策略不能将流量分配到多个后续路径上,无法有效地进行负载均衡.ECMP（Equal-Cost Multi-Path）通过对数据包头部进行哈希运算,对数据流进行转发路径的映射,使用类似于 ECMP 的调度方案可以将到达一个节点的流‘分散’到后续的多条可用路径上[29],然而,该类方案根据局部信息进行流量的调度,往往会造成网络中某链路或节点的拥塞.

SDN 使用集中控制器计算流量转发规则,实现路径之间的负载均衡.与以往流量调度工作相比的主要优势为:转发决策是集中式而非分布式制定,有效的结合多个可选链路的占用率和流量的大小特征.按照调度的分配实体,可以将基于 SDN 的数据层流量调度工作分为路径和服务流量的负载均衡两大类,进一步可将前者分为基于 elephant 流识别的调度、面向 QoS 的调度以及其它算法,具体如图 6 所示.本节将对该分类中的调度工作进行逐一介绍.

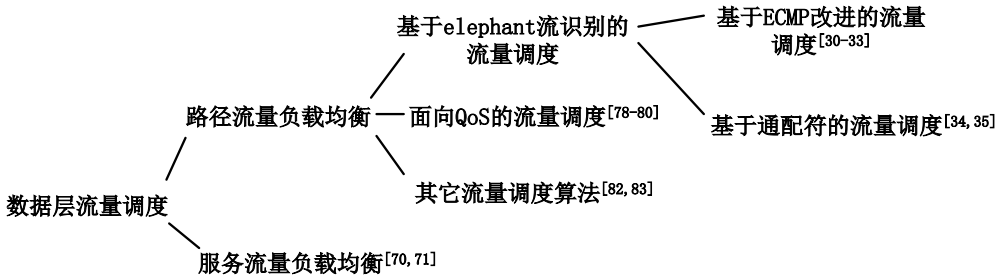


Fig.7 Classifications for traffic management schemes in data plane based on SDN

图 7 基于 SDN 的数据层流量调度分类

3.1.1 基于 elephant 流识别的流量调度

(1) 基于 ECMP 改进的流量调度

在ECMP路由中,拥有相同IP的流会沿着相同后续路径进行转发,可能导致多个elephant流(超过一定大小且持续时间长的流)映射到同一路径上,从而引起负载的不均衡和链路吞吐量的浪费[30~31],图 7 中给出了一个示例,图中到达聚合交换机的流 A 和流 B 被映射到同一后续路径,造成聚合交换机处的流长时间排队,而另一后续路径却无任何转发流量.没有合理参考当前链路的利用率和流的大小是引起大量数据包拥塞到一个端口,造成传输延迟长的主要原因.直接的解决方法是从包级别（原来是流级别）进行 ECMP 的流量切分,虽然极大的增加了负载之间的均衡性,却加重 TCP 流中报文的重排序问题,引起 TCP 发送窗口的不必要缩减[32].下面介绍基于 SDN 的 ECMP 改进方案,该类方案的主要思路是识别 elephant 流,进而利用控制器为其选择合适的路径.

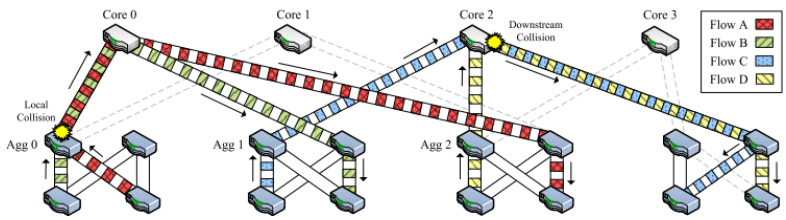


Fig.8 A case that multi-elephant flows are mapped to the same path in ECMP

图 8 ECMP 策略将多个大流量映射到同一路径

文献[30]提出的 Hedera 是一个可扩展的、动态的流量管理系统,它面向数据中心的多级路由结构,提出通过自适应的调度流以有效利用网络资源,实现等分带宽（穿过网络截面的最大传输率）的最大化.文章指出为了有效利用数据中心服务器之间的多条路径,需要及时地探测并管理 elephant 流（含大量数据的流）.基于这个

出发点,Hedera 开展包含三个基本步骤的调度策略:首先在边界交换机（端服务器的出口路由）中对大流量数据流进行探测.默认情况下 Hedera 使用 ECMP 进行流转发,同时通过向交换机周期性的询问（实现中为 5s）收集流信息,如果流的速率超过特定的阈值（实现中指定为 100Mbps）,则标记为 elephant 流;进一步,系统对判断为 elephant 的流的带宽需求进行估计,结合后续路径的负载情况,动态的为其计算合理的转发路径.

[30]指出为 elephant 流计算转发路径的算法是开放的（可以由用户自己定义）,文中选择全局首次适应算法(Global First Fit)和模拟退火算法(Simulated Annealing),并对两者进行了比较.以前者为例,已知 elephant 流的带宽需求 D,全局首次适应算法线性的搜索所有可选路径,将遍历过程中发现的空闲容量可以满足 D 的首个路径作为转发路径,进而在构成该路径的各个链路中进行容量的预留.最后将转发策略（规则-动作）更新至路径上的各个交换机中.

文章通过实验指出 Hedera 可以达到最优带宽利用率的 96%,同时,相比于静态的调度方法,有效地管理 elephant 流使得网络总体吞吐量增加超过 100%.

文献[33]提出的 Mahout 引入端服务器检测降低了流量管理的开销,它同样提出通过优化对 elephant 流的管理提高路径带宽的利用率.首先,文章指出 Hedera 方案中的 elephant 流探测方法利用的是基于网络内部的探测,往往引入较长的探测延迟、较高的交换机探测资源开销、较大的控制报文开销或处理开销.所以,Mahout 提出使用端服务器而非转发设备进行 elephant 流的探测,实现高效、低开销的流量管理.在 Mahout 的实现中,每个端服务器操作系统中抽象出一个 shim 层,shim 层负责通过本地的套接字缓冲监视该服务器产生的流,当缓冲超过指定阈值时,则将对应流判定为 elephant 流,然后对该流的后续数据包进行标记.Mahout 控制器通过高、低两个优先级设置流表,缺损情况下数据包匹配低优先级规则,交换机利用 ECMP 对其进行转发;标记为 elephant 流的数据包会匹配高优先级规则,交换机会将其发送至控制器,控制器进行最优路由的计算,并更新交换机中对应流的转发规则.

对于 elephant 流的管理,Mahout[33]采用离线首次适应算法(Offline Increasing First Fit),遍历所有可选路径并利用如下公式计算所有路径的拥塞度,算法选择拥塞度最小的路径作为最优的路由.其中 f 表示流,f.rate 表示其速率,path 表示一个可选路径,path.load 表示路径上的背景负载,path.bw 表示路径的带宽,而 congest 表示评估的拥塞度.

$$congest = (f.rate + path.load) / path.bandwidth \quad (1)$$

Mahout 方案的优势在于:对 elephant 流的测量过程不消耗交换机的 CPU 和存储资源,不产生集中的带宽开销;此外,将对 elephant 流的判断放在端节点中,使得数据只会缓冲在应用层而非堆积在网络层设备中.实验表明,Mahout 有效地增加了网络中路径带宽的利用率,同时,相比于询问测量方式,降低了检测 elephant 流的延迟.

(2) 基于通配符的流量调度

OpenFlow 协议规定交换机将未匹配流的第一个数据包发送至控制器,控制器通过计算将一个新的匹配规则安装到交换机中,流量依赖规则进行转发路径的映射.大量的流的到达以及流量信息收集的开销,使得集中的控制器成为瓶颈,为了降低数据层向控制层传输的数据量,支持高性能的网络需求,文章[34]和[35]中提出使用基于通配符的规则同时匹配多个流,避免控制器频繁的介入（由于此类方案目的是减少控制流量,所以一并归入表 2 进行总结和对比）:

文献[34]中提出 DevoFlow,一种资源节约的、可扩展的流量调度方案,它对 OpenFlow 模型进行修改,提出仅将必要的流从数据层转发至控制层,从而放松集中化控制引入的交互开销.具体的,DevoFlow 提出控制器应该在发现一个流为 elephant 流后对其进行重路由,而非将所有到达网络的流都视为一个潜在的 elephant 流,以避免由此带来的控制延迟和开销.文章设计了一个基于通配符的多路径匹配规则,初始情况下交换机使用这一规则进行流的转发,同时引入一种流量信息统计的方法识别 elephant 流,被识别的流需要控制器的介入进行最小拥塞路径的重路由.仿真结果显示,较以往的需求,DevoFlow 平均少使用 10-53 倍的流表项和 10-42 倍的控制信息.

与 DevoFlow[34]类似,文献[35]提出的 DIFANE 是一个基于通配符的、高效率的流量调度方案.DIFANE 的核心思路可以概括为两方面:首先,控制器将规则以最小划分的方式分发给一部分称为授权子集的交换机;其

次,交换机在数据层中处理所有的包,如果流无法与入口路由本地的规则匹配,则将它封装、重定向到合适的授权路由,授权路由对包进行处理并反馈规则.整个管理过程涉及 3 种通配规则:常规转发规则、定期更新的授权路由规则和重定向规则.实现方面,文章提出的方案易于使用商用交换机实现,且仅需要少量的软件扩展,因为所需要的数据层功能都可以由通配符规则来表示.

3.1.2 面向 QoS 的流量调度

网络中大量的实时业务流对传输过程中的延迟、丢包敏感[79],所以合理的调度网络资源为实时业务提供 QoS 保证是流量工程的一项重要工作.SDN 网络提供的控制接口支持灵活的流量调度策略制定,以充分兼顾不同网络应用的需求,并缓解网络中交叉节点或链路上的拥塞.

文献[78]提出 OpenQoS 方案,为多媒体的传输分发提供 QoS 保证.基于多媒体业务流的数据包头部与其它流中包头部的不同,OpenQoS 利用 OpenFlow 配置匹配规则将流量分为多媒体流和数据流两组.针对多媒体流,OpenQoS 结合后续路径在延迟和丢包两方面的表现,为其选择一条 QoS 路径,其它数据流保持在最短路径上.该方案仅对多媒体流的调度进行优化,没有考虑多个有 QoS 需求的业务流同时进行调度的场景.

UCLA 大学的 Mario 的研究小组提出一个 SDN 网络中基于集中式流量调度的 QoS 增强框架[79,80].方案考虑的具体问题为:在一个有交叉节点或链路的网络中,如何为多个应用的业务流量选择合适的路径,在满足流量的 QoS 需求的条件下,分配到每个路径中各个链路上的流量大小不会超过链路的容量. 文中提出的框架包含 6 个关键组件: 网络拓扑映射模块和网络状态收集模块负责监控数据层,动态的捕获拓扑的更新和收集网络参数,最后生成网络中各路径的权值图;路径选择模块结合业务流量的需求和权值图为不同应用选择合适的后续路径;动态路径配置模块在 QoS 管理模块的指导下在合适的时机将新的路由规则更新到数据层.作为框架的核心模块,路径选择模块利用文中提出的 MCFCSP(Multi-Commodity Flow and Constrained Shortest Path)模型进行调度决策,模型考虑在网络 G=(N,A)中为业务流量集合 K 建立转发策略,模型的目标函数为:

$$\min: \sum_{(i,j)\in A}\sum_{k\in K} c_{ij} x_{ij}^{k} \tag{2}$$

其中, $x_{ij}^{k} > 0$ 表示业务 k 转发至链路(i,j)上的流量大小,而

$$c_{ij} = \alpha d_{ij} + \beta p_{ij} \qquad \forall (i,j) \in A \tag{3}$$

表示链路(i,j)的开销,其中 d_{ij} 和 p_{ij} 分别表示网络状态收集模块通过实时测量获得的链路的延迟大小和丢包率统计值.结合模型给出的关于参数和变量的限制,可以对这一 NP 完全问题进行求解(文中指出该方案适用于规模较小的网络),获得全局的调度策略.文中基于 Mininet 平台对方案进行了验证,实验结果显示方案可以在链路出现拥塞时,为包括视频和文件传输在内的业务流量提供高于设定阈值的带宽资源,满足实时业务的 QoS 需求.

3.1.3 其它流量调度算法

OpenFlow 协议的出现为权衡数据中心网络中与日俱增的业务流量的带来的压力提供了途径,然而,文献[82]中指出在已有的基于 OpenFlow 的协议中,仅仅在流初始化时利用全局信息为其建立一条静态路由规则,流在传输过程中网络的配置可能发生变化,导致规则不适用,网络性能受到影响.针对以上问题,文中提出一种路径上的负载均衡路由算法 LABERIO, LABERIO 利用如下公式决定是否进行重调度,

$$\delta^{*} < \delta(t) = \frac{\sum_{1}^{N} [\overline{load}_{i,j}(t) - load_{i,j}(t)]^2}{N} \tag{4}$$

其中 $load_{i,j}(t)$ 表示链路(i,j)上的负载, $\delta(t)$ 给出了所有链路负载的方差,该值与网络的稳定性成反比,当其超过给定的阈值 δ^{*} 时,说明有必要进行转发规则的重计算.调度算法的核心是将网络中最拥塞的链路上占据最大

带宽的流量调度到其它后续路径上.具体的,通过对最拥塞链路(i,j)上各个业务流的负载占总体的比例进行排序,进而选择以 i 为起点,j 为终点的所有路径中开销最小的,将排序出的最大流通过该路径转发.

文献[83]中提出一种基于模糊评估模型指导负载平衡决策的流量调度方案.方案的执行分为初始模式和周期模式两个阶段,初始模式下,文中利用经典的 Floyd 算法为网络中的两点(s,e)计算出最短的 K 条路径,目的是在进行动态选择时可以提高效率.当网络中有业务流之后,方案进入周期模式并周期性的执行路径的模糊评估方法.模糊评估方法首先获取 K 条路径的跳数 h,传输的包计数 p,关键交换机的字节计数 b,以及关键端口的转发速率 r 几类特征,经过转换之后得到各条路径的影响向量 $R_i=(r_1,r_2,r_3,r_4)$,从而构成模糊关系矩阵 $R=(R_1,\ldots,R_k)$,最后利用一个给定的权值行向量 W 与 R 求积,得到各路径的评分向量 S,选择评分最高的路径作为最短路径,分配给对应业务流.

3.1.4 服务请求流量调度

现今的网络服务中,往往采用多个功能对等的服务器提供服务,服务请求首先到达一个负载均衡器,均衡器通过预定义的策略将请求分发至不同服务器,使得服务器的处理压力得到分摊.然而,现有的均衡器存在价格昂贵,单点时效问题,而且均衡器的调度逻辑固定,无法有效的进行调度策略的更新和优化.基于 OpenFlow 的控制器可以动态的收集服务器的负载情况,为解决传统负载均衡器存在的问题提供了一种可行的方案.

由斯坦福大学提出的 Plug-n-Serve 方案（现在更名为 Aster*x）是一个基于 OpenFlow 进行 web 流量负载均衡的应用[70] .它的目标是降低服务请求的响应时间,即通过合理选择网络路径和服务器降低发起请求到收到回复的时间.方案的主要逻辑架构如下图所示,Aster*x 通过 OpenFlow 控制器监视网络和服务器状态.图中控制器主要由流量管理者、网络管理者和主机管理者三个部件构成.其中,网络管理者以探测的形式对路径的占用率进行实时的评估,同时发现意外的拓扑变化;主机管理者监视系统中每个服务器的运行状态和负载情况;网络和主机管理者将收集的网络拥塞信息和服务器负载信息报告给流量管理者,流量管理者使用 LOBUS 算法进行请求流量的调度,最小化请求的延迟.

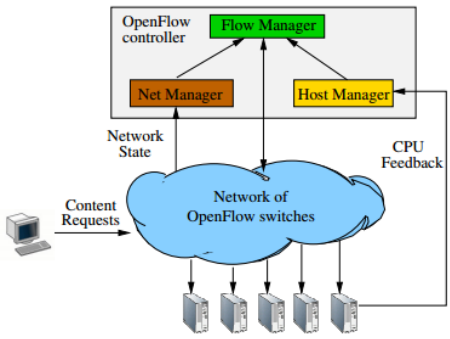


Fig.9 The main control logic of Aster*x

图 9 Aster*x 的主要控制逻辑示意图

文章[71] 提出为每个到达的服务请求计算、安装一个转发规则会产生大量的流规则,并造成控制器的处理拥塞,所以提出基于 OpenFlow 的服务负载均衡器应该利用通配符规则将服务请求进行聚合处理,避免流表的膨胀,降低控制器的压力.一般情况下,负载均衡器扮演一个 NAT 服务器的角色,收到请求之后将流的目的地址改写为某个服务器的 ip 地址,交由该服务器进行处理,每个服务器有单独的 ip 和一个标志其预期应处理总体请求比重的权值.文章针对的是需要预先安装在交换机中的部分规则,在原有调度算法生成的以单个请求为单位转发规则的基础上,将多个 ip 前缀匹配的转发规则聚合成一个含有通配符的规则,有效的控制了转发规则个数.

Table2 Methods in traffic management

表 2. 流量调度方法及模型

分 类	方 法	解决方案	优缺点分析
数据层流量调度——降低控制请求流量	**DevoFlow**	控制器仅处理 elephant 流, 其它流使用通配符规则在数据层处理	减少控制交互, 降低了实时视图的准确性

		DIFANE	控制器将规则分发给授权交换机,授权交换机处理各个入口无法处理的流,将流全部留在数据层	极大的降低控制请求;转发的开销较大
控制层流量调度——控制器部署	逻辑集中物理分布式	**HyperFlow**	交换机与最近的控制器通信,控制器之间维持视图同步	将控制分布化;额外开销用于同步控制器
	物理分布式	**Onix**	使用发布/订阅方法,向下提供分布式管理,向上提供集中视图与接口	额外的维护开销
		BalanceFlow	使用一个超级控制器管理控制器,实现控制层的负载均衡	需要两级控制层之间的额外交互
	层次化分布式	**Kandoo**	同时维护一个集中和本地的控制层	本地控制层没有集中视图

3.2 控制层流量调度

SDN 的集中化控制器需要处理来自数据层的大量数据包,面临处理瓶颈和缺乏扩展性在内的一系列问题[36].文献[37]指出,一个典型的 NOX 平台（首个 OpenFlow 控制器,使用最广泛）每秒可以对 30K 个流进行初始化,平均每个流 10ms,然而,这无法满足 SDN 的应用需求,尤其对于数据中心场景.所以,控制层流量调度的主要工作为:协调部署多个控制器以缓解集中控制的处理压力,降低延迟开销,维护可扩展性.表 2 中总结了控制层调度方案,具体的:

(1) 逻辑集中物理分布式控制器结构

逻辑集中的分布式控制器结构可以有效缓解集中控制的压力,同时利用逻辑上的抽象保留原有的控制器南向接口.文献[36]提出的 HyperFlow 是一个基于事件的分布式 OpenFlow 网络控制平台,它利用分布式的控制器结构对网络进行逻辑上集中的控制,从而在解决扩展性问题的同时保留集中控制的优势.文章提出的设计方案中,HyperFlow 规定每一个交换机使用 OpenFlow 协议与它临近的控制器通信,临近指的是具有最小的通信开销,另一方面,多个控制器之间采用事件传播系统进行同步,维护一个共享的、一致的网络视图.

(2) 物理分布式控制器结构

与逻辑集中的结构相比,直接利用物理分布的控制器结构同样可以缓解流量对集中控制器的冲击.文献[38]提出的 Onix 是一个运行在一个或多个服务器上的分布式控制平台.Onix 的核心是一个通用的网络控制接口,运行在多个控制服务器上的应用可以通过接口读取和修改网络中任何设备,从而保持设备和应用的一致性.Onix 将收集的网络状态保存在一个称为 NIB 的数据结构中,控制器实例们通过复制和分发 NIB 共享全局的视图.值得一提的是,Google 的 B4 项目使用的就是 Onix 控制平台.

与 Onix 类似,[39]提出的 BALANCEFLOW 是一个基于 OpenFlow 网络的控制器负载平衡框架.[39]指出发送给控制层的请求流应该被动态的分配给多个控制器,一个控制器上过载的流应该自动的被转移到另一个控制器中,以维持最大的控制器利用率.不同于 HyperFlow 中在交换机级别进行控制器分配,[39]提出在流级别进行分配,方案在 OpenFlow 协议基础上扩展了一个 X 动作,该动作将流转发至 X 对应的控制器.控制器提前或实时的将含有 X 动作的流表项安装到交换机中,这就为需要转发到控制层流量提供被灵活调度的能力.同时,BALANCEFLOW 在设计中引入一个超级控制器,用于分析其它控制器周期性推送的流请求信息,如果发现流量不均衡则进行集中式的流量重调度,并将修改的 X 规则发送至对应控制器,进一步更新到数据层.

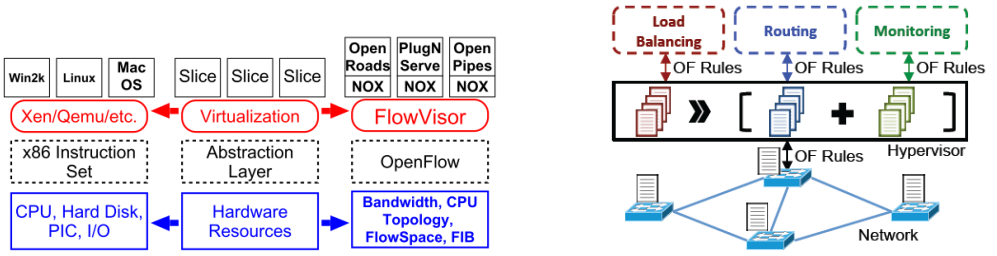
(3) 层次分布式控制器结构

层次化分布式控制器结构尝试将部分控制功能移至引入的本地控制器,以解决集中控制器南向接口的拥塞问题.文献[40]中提出的 Kandoo 利用层次化的控制器部署方法,缓解了控制集中化的扩展性问题,以及控制请求对控制器的压力.文章指出将流量控制在数据层需要对交换机进行修改,且牺牲了一定的全局视野,所以提出层次化的分布式控制器结构,保持集中控制的优势,也避免了对交换机的修改.Kandoo 框架中包含两类控制器:本地控制器和逻辑上集中的根控制器,本地控制器以一对多的形式管理交换机,处理仅需要本地信息的应用——一部分如链路层发现协议的应用（可以分解为）仅需要本地操作;而根控制器则用于支持需要全局信息的应用,并通过本地控制器更新规则.

3.3 流量调度的虚拟化

SDN 为网络应用提供统一编程接口,应用通过接口对数据流进行管理,对网络资源实行调配,通过虚拟化技术可以实现网络流量和资源实体在多个应用之间的高效共享.

作为一种典型的流量虚拟化调度方案,FlowVisor[41]是一个基于 OpenFlow 的、插入在交换机和控制器之间的虚拟化层,也可理解为一个逻辑分布的控制器或一个资源代理,目的是取得与计算机虚拟化相同的效果.FlowVisor 的结构如图 8(a)所示,该层次针对带宽、拓扑、流量、CPU 以及转发表 5 方面资源进行虚拟化,将总的网络流量分片之后交由多个控制逻辑管理,每个控制器面向一个客户自定义的策略,仅针对对应的流量分片进行操作.通过这种方式,FlowVisor 允许多种网络配置以不相交的方式运行于多种硬件之上,向上为用户提供独立的自定义和优化能力,向下抽象覆盖多种速率、介质的网络形式.



(a) (b)

Fig.10 Architecture of FlowVisor and hypervisor

图 10 FlowVisor 和 hypervisor 结构图

如上的传统方式将网络中的流量分成了多个不相交的“片”,每个应用只能选择一个“片”进行操作和调度,虽然提供了多用户共享网络的能力,然而多个用户的应用无法同时生效作用于同一个分片上.文章[42]指出未来 SDN 网络中的超级管理器应该允许将多个应用的策略同时映射到一个流上,所以提出并设计了一种复合式的管理器以简化网络管理,支持不同平台、不同语言同时对流量进行处理.图 8(b)中展示了位于控制器和应用之间的复合式超级管理器的逻辑效果.最上面一层是进行负载均衡、路由和测量功能的 3 个应用,3 个应用产生 OF 规则并送至超级管理器,并由它判断规则之间是否可以并行(+),还是需要串行(>>),然后将规则安装到网络中的交换机上.

通过设计超级管理器实现对网络的虚拟化管理过程中,最大的挑战和主要的研究工作是如何为多个用户应用提供正确,有效和实时的规则更新算法.

4 基于 SDN 的故障恢复

故障管理是为数据中心以及企业级网络提供通信服务的基本要求.有效的故障管理需要及时响应网络组件的故障,对可用的网络资源进行重分配,以保证服务的运行.在传统的 IP/MPLS 网络架构中,通过传输层和网络层的分布式协议进行故障恢复,协议提前针对可能的故障(如节点失效)计算资源分配方案,在出现故障之后利用消息机制激活备份的分配方案[72] .

SDN 将网络流量的控制和转发逻辑解耦,提供了传统分布式网络结构所不具备的可编程性,但也造成传统故障恢复机制在实施过程中缺乏灵活性和效率,具体的问题可概括为:

1) 由集中的控制器负责响应失效事件,增加了失效处理的延迟和通信的开销.实现快速的故障恢复([43]中给出的量化指标为 50ms)是有挑战的,因为检测到错误之后,控制器必须计算一个新的路由并将恢复规则更新到所有受影响的交换机中

2) 集中控制器本身存在单点失效的隐患,控制功能的故障将进一步导致转发功能非正常运行.

所以,SDN 中故障恢复相关的研究工作可以分为数据层的故障恢复和控制层的故障恢复两个方面.

4.1 数据层故障恢复

传统网络中针对网络设备和链路的故障恢复机制可分为两类[44]:

1) 非预留型.此类方案中,用来替代故障路径的备份路径可以是提前或动态计算分配的,但是相应的链路资源没有预留.当路径故障出现之后,需要一个特殊信号来通知建立恢复路径.

2) 预留型.用于故障恢复的路径是提前计算并预留好的,即工作路径和备份路径对应的规则是同时计算和安装的,所以在发生错误时不需要触发额外的信号.

不难发现,非预留型方案采用的是实时应激性恢复策略,而预留型方案是前瞻性策略.两类机制在 SDN 中都得到了运用:一方面,文献[45]和[46]中使用非预留型策略,在控制器收到链路故障的通知（OpenFlow 协议中的 PortStatus 消息）之后,会先确定受到影响的路径的列表,然后使用 OSPF 算法为受影响路径计算恢复路径,最后将新的表项安装到交换机;对于既处在故障路径上也处在恢复路径上的交换机,需要对流表进行修改,否则删除或添加相应的表项.另一方面,文献[47]和[48]中采用预计算和预留恢复路径的策略,控制器将计算得到的预留路径与正常路径一同安装在交换机中.

Table3 Qualitative overview of different schemes of fault tolerance for data plane

表 3 不同故障恢复方案的量化比较

名 称	故障恢复方法	非预留方案的最大延迟	预留方案的最大延迟
快速故障恢复[45][46]	数据层非预留	(80 – 130ms)>50ms	N/A
电信级故障恢复[47]	数据层预留与非预留	60ms>50ms	(42 – 48ms)<50ms
OSP[48]	数据层预留	N/A	64ms>50ms

两类方案的延迟对比如上表所示.通过分析可知,SDN 中采用非预留策略的缺点在于:网络需要在出现故障时引入控制器,并且修改流表,增加了故障恢复的延迟.另一方面,预留策略以增加交换机存储开销为代价,降低恢复延迟和恢复过程的通信开销.所以,对于重视无故障连续运行的大规模 SDN 系统,更倾向于使用预留方案进行有效的故障恢复.

文献[45-48]的研究工作为 SDN 网络提供了基本故障恢复功能,在此基础上,针对如何进一步提升故障恢复的速率,如何降低访问集中式控制器的开销,以及如何将故障管理逻辑从 SDN 应用中解耦,展开了一系列相关研究:

(1) 降低故障恢复延迟

通过降低生效延迟和控制更新延迟可以实现更快速的故障恢复.生效延迟指的是恢复路径等待原表项超时的时间,[45]提出通过直接删除失效链路的表项,再添加更新的表项,从而降低生效延迟.另一方面,鉴于更新延迟应该尽可能小,提高故障恢复的效率,避免不必要的转发浪费带宽资源,[49]提出在交换机之间传递简单的链路失效消息,缩短受影响交换机获悉失效的时间,同时不引入控制器进行全网络的泛洪,然而这种方法存在扩展性上的限制.

文献[73] 中提出使用 BFD（双向转发探测协议）算法进行链路和路径故障的发现.BFD 算法使用控制和回复消息监视链路的状态,文章为每个链路建立 BFD 会话,从而降低探测消息的复杂度（每条链路单个会话）,同时降低了故障发现的间隔（使用缩减的会话 RTT）.实验证明,当选择的 BFD 传输间隔为 15ms 时,方案可获得的平均故障恢复时间为 42.1ms,当间隔降低为 1ms 时,恢复延迟降低到 3.4ms,达到了电信级别网络 50ms 的要求.

(2) 降低访问控制器开销

SDN 将网络的控制逻辑集中到控制层,提供灵活的全局视图和动态配置能力,然而网络事件的处理需要控制器频繁的介入,增加了处理延迟,对于延迟敏感的故障管理影响明显.如何将故障管理的逻辑下移至数据层,成为有效进行网络故障管理的关键.

文献[73]中将 BFD 算法逻辑加入到 OpenFlow（1.1 版本）提供的组表功能中,一个组表可以对应多个动作.

故障恢复功能可以由组表来提供:将主路径及故障恢复路径合并到一个组表中,并增加一个默认规则——如果没有合适的备选路径,将流从进入端口回溯至上游交换机,这个操作可以一直迭代到具备有效备选路径的交换机.如下图所示,BFD 组件负责监视链路状态并控制故障恢复组表的动作选择,动作 1 和动作 2 分别对应转发至主路径和备选路径,动作 3 进行流量的回溯.所有动作的配置由控制器预先定义,发生链路或节点故障之后,数据层内部进行流的重路由或回溯到上游节点,无需额外的控制器访问.

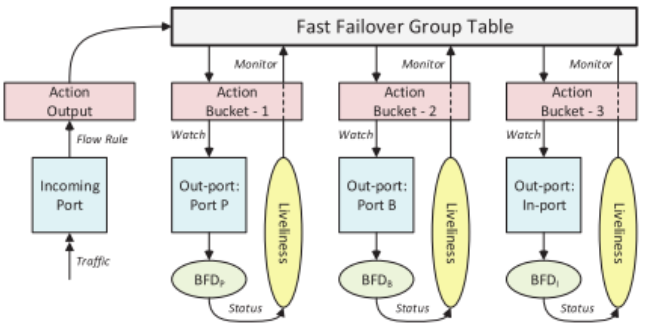


Fig.11 OpenFlow fast failure group table

图 11 OpenFlow 的故障管理组表

类似的，文献[74]也提出使用组表进行故障恢复规则的管理,同时指出通过预先将备选路径规则安装至交换机增加了存储开销,所以创新的提出将多个转发端口相同的备选路径对应的规则进行合并,使用通配符形成一个规则以减小存储压力.

文献[74] 中提出在数据层中利用 OpenState 进行包级别的事件响应,在发生链路失效之后快速的将流重定向到可用的路径上.OpenState 扩展了对 OpenFlow 数据层的抽象,其中交换机可以根据状态进行流规则的匹配,状态由状态机根据包级别事件的触发而转化.基于 OpenState 的故障管理流程为:当下行链路上的交换机节点探测到链路失效后,对缓冲中得数据包加标志,并将其转发至上行链路的节点;具有标志的数据包流到达可以重路由的节点 N_r 之后,触发该交换机的状态转换,进一步将流通过预计算安装的迂回路径进行转发; N_r 对后续到达的流加标志并使用迂回路径转发;标志的流通过合并交换机 N_m 重新回到主路径之后,标志会被去掉.通过状态转化指导转发规则,方案降低了频繁访问远程控制器的开销.

(3) 应用中故障管理逻辑的解耦

文献[74] 中指出 SDN 中使用控制器进行故障恢复策略的集中制定,而对于当今的主流控制器（POX, Floodlight 等）,所有运行在上面的控制器应用中都需要包含单独的故障处理逻辑,增加了程序代码编写的难度,和潜在的错误数目.针对以上问题,文章提出 AFRO 运行时系统,进行故障的自动恢复以简化应用程序的设计和编写.AFRO 提供两种运行模式:记录模式和恢复模式.网络正常运行时系统处于记录模式,对所有 PacketIn 消息进行记录,并通过保存 FlowMod 和 FlowRem 信息记录所有安装的规则.当探测到故障时,系统切换至恢复模式,该模式包含两个步骤:a)系统模拟故障发生之后的网络拓扑,并将记录模式下保存的故障发生之前的消息作为输入反馈至模拟的系统,重放得到正确的转发逻辑;b)系统将当前故障网络的状态迁移至正确的网络状态,分别将不一致的规则安装至交换机中,并对控制器的状态进行转化.通过以上过程,AFRO 成功的将故障恢复逻辑从应用程序中解耦,并保证故障管理的正确性.

4.2 控制层故障恢复

分布式的结构为容错性提供了良好的基础,而 SDN 将控制功能从传统网络中抽象出来形成一个集中化的层次,在带来了网络维护的收益的同时也潜在的增大了故障的影响.所以,为了保证控制层的可靠性,必须有效地解决控制器单点故障的问题.最直观有效的解决方法就是进行双控制器备份[50],备份控制器可以在主控制器故障之后继续提供服务.SDN 中实施双控制器备份主要需要解决两方面问题:

1) 主和备份控制器的同步、协作协议.OpenFlow 提供设置配置多个备份控制器的机制,但没有提供任何协

同机制.所以,需要协同机制保持主和备份控制器之间的一致性,降低切换的代价[51].

2) 备份控制器的部署.在引入尽可能小的维护开销和同步延迟的条件下,实际在何处部署、部署多少备份控制器可以保证足够的可靠性[52].具体的,应该明确控制器数目对可靠性的影响,以及可靠性与加入备份引入的延迟之间如何进行权衡.

针对主、备份控制器之间的同步操作,文献[51]设计了 CPRecovery 组件,用于支持双控制器备份机制,CPRecovery 可以独立的运行在多个网络操作系统之上,具有较好的可扩展性.文章设计的备份机制包括两个运行阶段:复制阶段和恢复阶段,分别对应主控制器正常和故障两个情形.系统运行于复制阶段时,主控制器负责处理来自数据层的控制请求,如果控制器维护的流表中不包含对应流的表项,则需要计算一个新的流表项,并通过 CPRecovery 发送消息将其同步到备份控制器,同时使用消息确认机制确保同步成功.如果控制请求发送到备份控制器,CPRecovery 会将备份控制器标识为主控制器;NOX 的处理故障以及控制与交换机的链路失效都会触发系统进入恢复阶段,其中,链路的失效是通过交换机周期性的发送控制器探测报文来捕获的,如果控制器在一定时间内未对探测报文进行响应则认为当前的主控制器无法连接.处于恢复阶段时,交换机向备份控制器发起请求,备份控制器的状态转化为主控制器,接管网络并不断地尝试与原控制器通信.方案的缺点在于增加了控制器响应交换机请求的延迟,因为需要确保同步操作的完成.

文章[52]对控制器的部署位置问题进行了讨论.文中假设控制器与数据层之间通过 in-band 的形式进行交互,即控制路径基于的是交换机之间的物理链路.部署位置问题可以形式化为:针对于一个网络拓扑 G=<V,E>,其中 V 表示交换机节点,E 表示节点间的链路,设备和链路的失效概率相同,如果有 K 个控制器,在|V|个可选的位置中应如何进行放置,以最大化总体的可靠性,即降低控制链路失效的比率.文中提出随机放置、模拟退火和贪婪法等算法对问题进行求解,通过实验分析指出模拟退火算法可以输出可靠性最高的部署方案.

文章[52]和[53]中,详细分析了控制器部署的数目对延迟和可靠性的影响,即两者之间的相互权衡.通过在 OS2E 和 Rocketful 拓扑上的实验分析,[52]指出增加控制器的数目,可以分摊控制请求的压力,降低控制请求的响应延迟,但是却由于在控制层与数据层之间引入更多的通信链路而造成链路失效概率的上升,降低了总体的可靠性.另一方面,[53]提出一种(3+1)的控制器配置方案,即 3 个用于负载均衡的主控制器配合 1 个备份控制器提供控制层功能.文章通过实验指出 10ms 的控制器延迟可以满足具有 8 到 200 个节点的网络,并且提出一个最合适的控制器数目应该位于区间[0.035n,0.117n]之间,其中 n 是网络节点的个数.

5 总结与展望

5.1 总结

SDN 具有集中控制、控制管理分离以及开放接口等特点,为有效的网络管理、网络负载均衡和安全分析工作提供了基础支持.在此基础上,流量工程技术能充分利用 SDN 的特性,更好的发挥 SDN 在应用层面优势.

开展流量工程的目标是利用实时、动态的网络信息对网络进行监控、维护和管理.总结来看,SDN 中流量工程的研究主要包括以下三个方面:

1) 流量的动态测量分析,准确的测量是流量工程相关服务的基础.流量测量的主要工作为收集流量统计信息,分析网络流量的特征,发现特殊流量(如 elephant 流)和不合理的配置.文献[13-18,20]采用采样或计数器的方式统计流量信息,使用控制器主动询问或被动接收的方式收集交换机中存储的统计结果,各个方法在开销和准确性间展开权衡.文献[22-26]提出了若干网络正确态进行检查的方案,对规则配置进行静态或动态的监控和评估,避免循环路由等问题的产生.此外,测量的过程应合理的利用有限的网络资源,交换机的片上空间有限,流表的存储单元(TCAM)成本高、数目少,应当在保证测量正确性的基础上尽量减少 TCAM 的占用,文献[27-28]设计方法在测量准确性和空间开销上进行了有效的权衡.

2) 流量的管理调度,是流量工程提高网络性能的一个重要途径,管理的目标是实现网络路径或者服务器之间的负载均衡,提高链路利用率,和增加网路的总吞吐量(或对等带宽).数据中心的流量具有突发和动态特点,合理调度大数据量流量的可以明显的改善网络性能,文献[30,33-35]在交换机或端服务器中设置阈值检测

elephant 流,然后通过集中控制器根据当前网络状态计算、更新流的转发策略.另一方面,采用分布式或层次化的控制器结构可以缓解 SDN 中控制请求和信息汇总的瓶颈问题,文献[36,38-40]在提出控制器架构基础上设计控制流量的管理方法,多个控制器采取就近、集中调度、层次化响应等策略对控制请求流量进行管理和调度.最后,将流量虚拟化为多个分片,可以支持应用对流量的灵活管理,文献[41,42]采用完全分离和组合的方式对操作分片之后的网络流量进行调度管理.

3) 网络中错误及故障的恢复,用于保证 SDN 可用性和可靠性.软件和硬件问题导致的故障是不可避免的,针对数据层的设备或链路出现的故障,可以采用传统的预留或非预留两类恢复机制保证网络继续在未受影响的路径和设备上继续运作[45-47],在此基础上,可以进一步利用包括 BFD 在内的数据层技术降低恢复的延迟,利用组表等技术缓解集中控制器在恢复过程中引入的开销[73-75].另一方面,使用备份控制器可以应对集中控制器单点故障,利用交换机与控制器间的心跳消息发现故障,从而切换主、备份控制器角色,是进行控制器间有效同步协作的一种可选方案,文章[52-53]对多控制器结构的部署以及可靠性和延迟之间的权衡进行了讨论.

5.2 展望

作为一项新兴网络架构,SDN 是当前网络领域最具发展前景的技术之一,备受业界和研究界的关注[68-69].在网络顶级会议 SIGCOMM 关于 SDN 的 workshop——HotSDN 中,涉及流量管理、流量测量、资源调度、状态更新等流量工程技术的文章比例的一个粗略统计如下表:

Table4 Statics for traffic engineering related papers in HotSDN

表 4 HotSDN 中流量工程文章统计

年份	2012	2013	2014
比例	11/22	7/24	13/35

围绕 SDN 的流量工程的研究项目也有许多,如 Google 开展的 B4[54],Microsoft 提出的 SWAN 架构[55],以及华为提出的 ADMCF-SNOS 系统[56]等.以 B4 为例,SDN 被用来改造 Google 数据中心之间互联的 G-Scale 网络,该网络的链路成本很高,链路的利用率却只有 30%左右,B4 采用有效的流量管理方法,实现链路的负载均衡,增加了网络稳定性和简化了网络管理,经过改造的大部分链路的利用率可以达到 100%.

最后,基于 SDN 的流量工程在以下几个方面有待于进一步改进或深入研究:

1) 快速且资源节约的故障恢复

针对数据层的可靠性维护机制中,采用路径预留法进行故障恢复可以保证短延迟,和控制器与路由器之间通信的低开销.然而,将备份路径转发规则提前保存在数据层的交换机中,会增加存储资源的开销.同时,预留方法属于静态范畴,转发路径是提前计算的,缺乏对动态网络状态的适应性,而对规则的动态更新还需要额外的操作对备份规则进行相同的更新.因此,如何进一步优化故障恢复方法,以保证在快速的前提下,有效利用交换机中得存储资源,并通过引入尽可能少的控制-数据层交互开销,进行预定义备份规则的动态更新,是值得深入研究的.

2) 多控制器结构

控制器是 SDN 的核心,集中化的设计在可扩展性、可靠性和处理能力上限制了网络的总体能力.采用多控制器结构可以缓解集中控制器的瓶颈问题,然而,究竟需要部署多少控制器? 控制器之间如何协同分工? 仍是需要进一步探究的问题.另一方面,SDN 的北向接口上“百花齐放”,为了满足自身应用和利益需求,不同生产商提供了自己的商业或开源控制器,如何维护和协调多种控制器共存的场景,使得同一控制域中控制器能正确同步,而不同控制域之间的控制信息能有效交互,需要进一步加以探讨.文献[57]提出通过观察控制请求总量维护一个弹性的控制器池,并将控制请求无缝的在多个控制器间迁移,通过协调分工实现负载均衡,这是一种可以借鉴的思路,但仍有待更多的深入研究.

3) 流量调度对传输协议效率的影响

开展对网络流量的有效调度,尤其是对 elephant 流而言,可以实现网络路径间负载的均衡,提高网络总体吞吐量.数据中心中 TCP 流量占主要成分,而流量的多路径传输会导致目标服务器处出现数据包的乱序到达, TCP

协议会根据滑动窗口协议限制发送端的窗口,进而影响发送端的数据吞吐率.所以,流量调度对传输协议效率的影响值得深入的分析和研究.

4) 基于测量的软件定义安全

网络流量的测量以及网络正确状态的检查工作应该与安全问题充分结合.SDN 采用的集中控制为安全策略的制定和部署提供了良好支持,克服了传统分布式网络结构的动态、不可控性对维护一个安全网络的影响.传统的安全问题在 SDN 中仍然存在,如以消耗网络带宽和处理能力为目标的 Dos 和 DDos 攻击.现有的静态阈值流量测量方式缺乏灵活性,限制了进一步分析发现异常流量的能力.基于 SDN 的流量测量的过程是测量和分析的回环控制形式,测量为分析提供统计数据,分析结合网络拓扑和实时采集的网络参数信息动态的对测量的配置进行修改,这种自适应的方式可以更有效的应对复杂的网络环境,更快速的发现网络中的流量、拓扑异常,并从全局的角度制定安全措施.另一方面,OpenFlow 提供的可编程能力为不同需求的服务自定义网络安全策略提供了充分、灵活的支持.所以,利用 SDN 提供的细粒度、实时测量结果指导安全应用,实现软件定义安全是一个有意义的研究课题.同时,SDN 集中控制的结构使得控制器成为了安全的薄弱环节,在实际部署中,如何通过测量分析对异常请求流量的模式进行有效的甄别也具有重要的实际意义.

5) 基于 SDN 的新型网络中的流量工程

SDN 将数据转发与控制逻辑分离的特性可以促进网络技术的创新和新型网络的发展.例如,用户中心网络(User-provided Network,简称 UPN)是一种依赖于用户之间共享可用的网络接入资源提升无线接入能力的新兴网络架构,[77]提出使用将 SDN 架构与 UPN 结合,利用 SDN 集中控制的优势实现对底层终端-终端、终端-接入点(无线 AP 或基站)之间转发路径可用性、利用率情况的监测,并基于实时的测量信息指导应用流量的转发.在以上过程中,有效的利用流量工程技术,可以对数据流量进行合理的、动态的调度.包括 D2D、移动群感知在内的新型网络中,都存在将控制和转发逻辑解耦,通过集中网络视图制定决策,实现数据流合理调度的需求.所以,如何在基于 SDN 框架的新型网络中开展有效的流量工程,对于发挥 SDN 优势,驱动技术发展是十分重要的.

References:

[1] ZHANG Chao-Kun, CUI Yong, TANG He-Yi, WU Jian-Ping. State-of-the-Art Survey on Software-Defined Networking (SDN). Journal of Software, 2015, 26(1):62−81 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4701.htm

[2] "OPEN NETWORKING SUMMIT 2012, " http://opennetsummit.org/archives/apr12/site/why.html

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.

[4] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in INFOCOM, 2013 Proceedings IEEE. IEEE, 2013, pp. 2211–2219.

[5] D. Awduche et al., "Overview and Principles of Internet Traffic Engineering," IETF RFC 3272, May 2002.

[6] J. J. Bae and T. Suda, "Survey of traffic control schemes and protocols in atm networks," Proceedings of the IEEE, vol. 79, no. 2, pp. 170–189, 1991.

[7] N. Wang, K. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," Communications Surveys & Tutorials, IEEE, vol. 10, no. 1, pp. 36–56, 2008

[8] D. O. Awduche and J. Agogbua, "Requirements for traffic engineering over mpls," 1999.

[9] The Openflow Switch, openflowswitch.org.

[10] "Netflow," http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod\white\ paper0900aecd80406232.html.

[11] "sflow," http://www.sflow.org/sFlowOverview.pdf

[12] A. C. Myers, "Jflow: Practical mostly-static information flow control," in Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. ACM, 1999, pp. 228–241.

[13] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: traffic matrix estimator for openflow networks," in Passive and Active Measurement. Springer, 2010, pp. 201–210.

[14] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba. PayLess: A low cost network monitoring framework for Software Defined Networks[J]. IEEE Network Operations & Management Symposium, 2014:1 - 9.

[15] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "Flowsense: monitoring network utilization with zero measurement cost," in Passive and Active Measurement. Springer, 2013, pp. 31–41.

[16] van Adrichem N L M, Doerr C, Kuipers F A. Opennetmon: Network monitoring in openflow software-defined networks[C]//Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014: 1-8.

[17] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in Proceedings 10th USENIX Symposium on Net- worked Systems Design and Implementation, NSDI, vol. 13, 2013.

[18] T. Benson, A. Anand, A. Akella, and M. Zhang, "Microte: fine grained traffic engineering for data centers," in Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies. ACM, 2011, p. 8.

[19] T. Benson, A. Akella, and D.Maltz. Network Traffic Characteristics of Data Centers in theWild. In Proceedings of IMC, 2010.

[20] J. Suh, T. Kwon, C. Dixon, W. Felter, and J. Carter, "Opensample: A low-latency, sampling-based measurement platform for sdn," in IBM Research Report, January 2014.

[21] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in SIGCOMM, 2010.

[22] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, "A nice way to test openflow applications," NSDI, Apr, 2012.

[23] Mai H, Khurshid A, Agarwal R, et al. Debugging the data plane with anteater[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 290-301.

[24] A. Khurshid,W. Zhou, M. Caesar, and P. Godfrey, "Veriflow: Verifying network-wide invariants in real time," ACM SIGCOMM Computer Communication Review, vol. 42, no. 4, pp. 467–472, 2012

[25] E. Al-Shaer and S. Al-Haj, "Flowchecker: Configuration analysis and verification of federated openflow infrastructures," in Proceedings of the 3rd ACM workshop on Assurable and usable security configuration. ACM, 2010, pp. 37–44.

[26] Al-Shaer E, Alsaleh M N. ConfigChecker: A tool for comprehensive security configuration analytics[C] //Configuration Analytics and Automation (SAFECONFIG), 2011 4th Symposium on. IEEE, 2011: 1-2.

[27] Moshref M, Yu M, Govindan R, et al. DREAM: dynamic resource allocation for software-defined measurement[C]//Proceedings of the 2014 ACM conference on SIGCOMM. ACM, 2014: 419-430.

[28] Mogul J C, Congdon P. Hey, you darned counters!: get off my ASIC![C]//Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012: 25-30.

[29] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," 2000.

[30] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in NSDI, vol. 10, 2010, pp. 19–19.

[31] A. A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks." in Proc. INFOCOM, 2013.

[32] Traffic Engineering with Equal-Cost-MultiPath: An Algorithmic Perspective

[33] A. R. Curtis,W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in INFOCOM, 2011 Proceedings IEEE. IEEE, 2011, pp. 1629–1637.

[34] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in ACM SIGCOMM Computer Communication Review, vol. 41, no. 4. ACM, 2011, pp. 254–265

[35] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with difane," ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, pp. 351–362, 2011.

[36] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in Proceedings of the 2010 internet network man- agement conference on Research on enterprise networking. USENIX Association, 2010, pp. 3–3.

[37] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying nox to the datacenter." in HotNets. Citeseer, 2009.

[38] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama et al., "Onix: A distributed control platform for large-scale production networks." in OSDI, vol. 10, 2010, pp. 1–6.

[39] [39] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "Balanceflow: Con- troller load balancing for openflow networks," in Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on, vol. 2. IEEE, 2012, pp. 780–785.

[40] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 19–24

[41] Sherwood R, Gibb G, Yap K K, et al. Flowvisor: A network virtualization layer[J]. OpenFlow Switch Consortium, Tech. Rep, 2009.

[42] Jin X, Rexford J, Walker D. Incremental update for a compositional SDN hypervisor[C]//Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014: 187-192.

[43] B. Niven-Jenkins, D.Brungard, M. Betts, N. Spreche, "MPLS-TP Re-quirements draft-ietf-mpls-tp-requirement-10", 2009

[44] Jean Philippe Vasseur, Mario Picavet, Piet Demeester, "Network Recovery: protection and restoration of optical, SONET-SDH, IP and MPLS", Morgan Kaufmann, 2004.

[45] Sharma S, Staessens D, Colle D, et al. Enabling fast failure recovery in OpenFlow networks[C]//Design of Reliable Communication Networks (DRCN), 2011 8th International Workshop on the. IEEE, 2011: 164-171.

[46] Staessens D, Sharma S, Colle D, et al. Software defined networking: Meeting carrier grade requirements[C]//Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on. IEEE, 2011: 1-6.

[47] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Openflow: meeting carrier-grade recovery requirements," Computer Communications, 2012.

[48] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "Openflow-based segment protection in ethernet networks," Optical Communications and Networking, IEEE/OSA Journal of, vol. 5, no. 9, pp. 1066–1075, 2013.

[49] M. Desai and T. Nandagopal, "Coping with link failures in centralized control plane architectures," in Communication Systems and Networks (COMSNETS), 2010 Second International Conference on. IEEE, 2010, pp. 1–10.

[50] N. Budhiraja, K. Marzullo, F. B. Schneider, and S. Toueg, "The primary-backup approach," Distributed systems, vol. 2, pp. 199–216, 1993.

[51] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient openflow-based networking," in Network Op- erations and Management Symposium (NOMS), 2012 IEEE. IEEE, 2012, pp. 933–939

[52] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability- aware controller placement for software-defined networks," in Inte- grated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE, 2013, pp. 672–675.

[53] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 7–12.

[54] 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.

[55] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. ACM, 2013, pp. 15–26.

[56] M. Luo, Y. Zeng, and J. Li, "An adaptive multi-path computation framework for centrally controlled networks," in To Be Submitted, 2013.

[57] Dixit A, Hao F, Mukherjee S, et al. Towards an elastic distributed SDN controller[C]//ACM SIGCOMM Computer Communication Review. ACM, 2013, 43(4): 7-12.

[58] Wegner P, Zdonik SB. Inheritance as an incremental modification mechanism or what like is and isn't like. In: Gjessing S, Nygaard K, eds. Proc. of the ECOOP'88. LNCS 322, Heidelberg: Springer-Verlag, 1988. 55-77.

[59] Waxman BM. Routing of multipoint connections. IEEE Journal on Selected Areas in Communications, 1988,6(9):1617-1622.

[60] Yonezawa A. ABCL: An Object-Oriented Concurrent System. Cambridge: MIT Press, 1990.

[61] Matsuoka S, Yonezawa A. Analysis of inheritance anomaly in object-oriented concurrent programming languages. In: Agha G, Wegner P, Yonezawa A, eds. Research Directions in Concurrent Object-Oriented Programming. Cambridge: MIT Press, 1993. 107-150.

[62] Hemige V. Object-Oriented design of the groupware layer for the ecosystem information system [MS. Thesis]. University of Montana, 1995.

[63] Rose A, Perez M, Clements P. Modechart toolset user's guide. Technical Report, NML/MRL/5540-94-7427, Austin: University of Texas at Austin, 1994.

[64] Keene SE. A Programmer's Guide to Object-Oriented Programming in Common LISP. Boston: Addison-Wesley Longman Publishing Co., Inc., 1988.

[65] Guo L, Tang ZS. Specification and verification of the triple-modular redundancy fault-tolerant system. Journal of Software, 2003,14(1):28~35 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/28.htm

[66] Schutze, H. Dimensions of meaning. In: Whitelock P, ed. Proc. of the Supercomputing'92. Los Alamitos, 1992. 787~796. ftp://parcftp.parc.xerox.com/pub/qca/papers/

[67] Wang XW. Research on quality-of-service management and group communication mechanisms in distributed multimedia systems [Ph.D. Thesis]. Shenyang: Northeastern University, 1998 (in Chinese with English abstract).

[68] Zuo QY, Chen M, Zhao GS, Xing CY, Zhang GM, Jiang PC. OpenFlow-based SDN technologies. Ruanjian Xuebao/Journal of Software, 2013 (in Chinese). http://www.jos.org.cn/1000-9825/4390.htm.

[69] Bi J. SDN architecture and future network architecture innovation environment [J]. Telecommunications Science, 2013, 29(8). DOI:10.3969/j.issn.1000-0801.2013.08.002.

[70] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R.Johari, "Plug-n-Serve: Load-balancing web traffic using OpenFlow," Demo at ACM SIGCOMM, Aug. 2009.

[71] R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-Based Server Load Balancing Gone Wild," in Workshop on Hot-ICE, Mar. 2011.

[72] J. P. VASSEUR, M. PICKAVET, P. DEMEESTER, "Network Recovery: protection and restoration of optical, SONET-SDH, IP and MPLS", Morgan Kaufmann, 2004.

[73] Van Adrichem N L M, Van Asten B J, Kuipers F. Fast recovery in software-defined networks[C]//Software Defined Networks (EWSDN), 2014 Third European Workshop on. IEEE, 2014: 61-66.

[74] Capone A, Cascone C, Nguyen A Q T, et al. Detour Planning for Fast and Reliable Failure Recovery in SDN with OpenState[J]. arXiv preprint arXiv:1411.7711, 2014.

[75] Thorat P, Raza S M, Nguyen D T, et al. Optimized self-healing framework for software defined networks[C]//Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication. ACM, 2015: 7.

[76] Kuźniar M, Perešíni P, Vasić N, et al. Automatic failure recovery for software-defined networks[C]//Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013: 159-160.

[77] D. Syrivelis, G. Iosifidis, D. Delimbasis, K. Chounos, T. Korakis, and L. Tassiulas, "Bits and Coins: Supporting Collaborative Consumption of Mobile Internet'', IEEE Infocom, 2015.

[78] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in Signal & Information Processing Association Annual Summit and Conference(APSIPA ASC), 2012 Asia-Pacific. IEEE, Dec. 2012, pp. 1–8.

[79] Francesco Ongaro, Eduardo Cerqueira, Luca Foschini, Antonio Corradi, Mario Gerla. Enhancing the quality level support for real-time multimedia applications in software-defined networks[C]//Computing, Networking and Communications (ICNC), 2015 International Conference on. IEEE, 2015: 505-509.

[80] Francesco Ongaro, Antonio Corradi, Mario Gerla, Eduardo Cerqueira, Luca Foschini. Enhancing Quality of Service in Software-Defined Networks[D]. ALMA MATER STUDIORUM-UNIVERSITY OF BOLOGNA, 2014.

[81] Gomes R L, Bittencourt L F, Madeira E R M, et al. An architecture for dynamic resource adjustment in VSDNs based on traffic demand[C]//Global Communications Conference (GLOBECOM), 2014 IEEE. IEEE, 2014: 2005-2010.

[82] Long H, Shen Y, Guo M, et al. LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks[C]//Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on. IEEE, 2013: 290-297.

[83] Li J, Chang X, Ren Y, et al. An Effective Path Load Balancing Mechanism Based on SDN[C]//Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. IEEE, 2014: 527-533.

[84] Liu H, Chen Z, Tian X, et al. On content-centric wireless delivery networks[J]. Wireless Communications, IEEE, 2014, 21(6): 118-125.

[85] Zhao T, Li T, Han B, et al. Design of Software Defined hardware counters for SDN[C]//Local & Metropolitan Area Networks (LANMAN), 2014 IEEE 20th International Workshop on. IEEE, 2014: 1-6.

附中文参考文献:

[1] 张朝昆，崔勇，唐翯祎，吴建平．软件定义网络(SDN)研究进展．软件学报,2015,26(1):62−81. http://www.jos.org.cn/1000-9825/4701.htm.

[68] 左青云，陈鸣，赵广松，邢长友，张国敏，蒋培成．基于 OpenFlow 的 SDN 技术．软件学报,2013, http://www.jos.org.cn/1000-9825/4390.htm

[69] 毕军. SDN 体系结构与未来网络体系结构创新环境[J]. 电信科学, 2013, 29(8). DOI:10.3969/j.issn.1000-0801.2013.08.002.