doi:10.11887/j.cn.201806013 http://journal.nudt.edu.cn

集中控制混合网络中基于流的资源分配算法*

宛 考,江 勇,徐 恪
(清华大学 计算机科学与技术系,北京 100084)

摘 要:集中控制混合网络中,异构化网络内部的流量具有一定的规律和特性,如果使用原有的单一的离散式最大极值和无状态的网络资源调度算法,忽略了异构网络规律,会造成网络利用率较低、易震荡、部分网络流延迟等问题。通过分析由集中控制网络和普通网络组成的混合网络的拓扑结构,对混合网络结构中常见问题如流闪现、不能估计的流、路径堵塞或连接震荡场景进行分析,并提出基于期望和状态的流量评价资源规划算法 POS 和 POS-FME。算法考虑混合网络的运行状态,对系统可用资源进行评估,为系统中各种流匹配对应可用资源,并具有一定的预测作用,从而避免混合网络出现运行效率低下的场景。通过实验,POS 算法和POS-FME算法相对传统算法,利用率提高了 10% ~30%,并降低了震荡和平均延迟。

关键词:混合网络;资源分配;流分析;流调度;优化算法
中图分类号:TN95 **文献标志码**:A **文章编号**:1001-2486(2018)06-089-16

Towards a flow based resource scheduling algorithm in center control hybrid network

WAN Kao, *JIANG Yong*, *XU Ke*
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: In the hybrid center control networks, the flows of isomerization networks have its own regularity and characteristic. While ignoring the regularity of the flows in isomerization networks, the original single and stateless network resource scheduling algorithms which discretely maximize value can cause the problems such as the low utilization, easier to concuss, some delayed network traffic. The topological architecture of the hybrid network which combines normal network with centered intelligent controllable network was analyzed, specially aimed to the common problems such as the flow burst, unpredictable flow, path jam/connection shocking scene. Then the resource scheduling algorithm named POS and POS-FME based on expectation and status were proposed. The algorithms consider the running status of hybrid network, evaluate the available resources of the system, and match available resources for various flows in the system, and have a certain prediction effect. The algorithms avoid the scenes of low running efficiency in the hybrid network, improve the utilization rate by about 10% ~30% im comparison with the traditional algorithm, and also reduce the concussion and average delay through experiments.

Key words: hybrid network; resource scheduling; flow analyses; flow scheduling; optimization algorithm

在网络研究中,网络的资源调度问题一直是研究的热点。资源调度一般分为两类:一类是通过对传输节点的调度,如通过在网络热点中设置额外服务节点,来分担热点区域的工作效率;另外一类是对流的调度,如通过链路调度算法进行最短路径匹配以运行流传输功能,或者设置防火墙、入侵检测系统(Intrusion Detection Systems, IDS)之类来主动进行流切换,控制系统安全或性能。网络中工作流大部分是基于 IP 协议点对点定向的,IP 层流调度不识别上层流的内容和应用类别,同时在网络中承载其他不同内容和应用类别的流;路由器不识别流的具体情况,仅仅根据流量来规划整体网络系统的运行状态和效率。随着网络技术发展,一些 IP 层的补丁协议不能够满足实际的网络需求,且各方提出的补丁是基于特定需求和场景实现的,不能进行通用的实践。在资源调度场景中,大部分网络中的流调度的传统方式是遵循改良性网络演化方式。Fortz 等[1-2]为了描述网络节点和链路运行状态,采用新的标准对开放式最短路径优先(Open Shortest Path First,

* **收稿日期**:2017-09-27
基金项目:国家自然科学基金资助项目(61170292, 61140454);国家科技重大专项基金资助项目(2012ZX03005001, 2014ZX03002004);国家重点基础研究发展计划资助项目(2012CB315803);国家863计划资助项目(2013AA013302)
作者简介:宛考(1982—),男,湖北武汉人,博士研究生,E-mail: wank12@mails.tsinghua.edu.cn;
江勇(通信作者),男,教授,博士,博士生导师,E-mail: jiangy@mail.sz.tsinghua.edu.cn

OSPF)和 IS-IS 协议设置权值,但该方法并不是全局性的,准确性与采取该算法的节点远近有关系,当需要全局信息时,路由器需要更复杂的扩展模块、命令行接口及配置,因此部署复杂、成本昂贵,难以实际推广。Applegate 等[3]通过建立新的权值设计方法来保证局部网络传输鲁棒性,Elwalid 等[4]通过多协议标签交换(Multi-Protocol Label Switching, MPLS)保持网络流量稳定,Mitra 等[5]收集服务质量(Quality of Service, QoS)网络事件信息来学习并预测未来网络事件,这些都是扩展路由器模块,对网络运行状态进行收集和分析。Handigol、Wang、Macapuna 等[6-8]通过设计静态算法来进行网络资源调度,其基本步骤是:收集器收集信息,根据算法限定可为全局或某一区域,通过最优化算法得出极值解,再根据极值解重新部署资源。以上算法收集系统状态仅限于收集时刻,最优解并不能对系统运行一段时间后的状态有效。

针对普通网络的问题,美国国家科学基金会(National Science Foundation, NSF)的 GENI[9](global environment for network innovations)计划提出了"革命性"型网络模型,McKeown 等[10]在校园网基础上设计了 OpenFlow 构架,提出了可编程网络雏形。在可编程网络中,如软件定义网络(Software Defined Networking, SDN)、云控制系统,控制节点可以对网络流进行分析,通过控制器系统扩展接口对流进行分析、检测、优化。其优势是全局可同步优化,劣势是在大流量和大数量不同的流的情况下,需要分布式控制节点对流进行同步,而同步需要时间消耗。在时间敏感网络如直播系统,需要大量资源进行同步来确保实时性要求。以日本电气股份有限公司 NEC 的 SDN 系统为例,Schneider 等[11]介绍了 NEC 的虚拟租户网络(Virtual Tenant Network, VTN)构架,来容纳多个租户的计费等管理工作;Hammad 等[12]建立了 XEN 服务器与 NEC 边界交换机上具有通过光网络发送/接收业务的密集型光波复用(Dense Wavelength Division Multiplexing, DWDM)光收发器的连接,使用该构架实现光网络大数据量传输。可编程网络的网络流的调度不是传输节点自发的,而是通过控制节点进行收集网络全局信息,依据某个特定的需求策略制定策略来进行整个网络的网络流调度。定制策略可通过各种方式,如 App、配置文件等,因为灵活方便,部署简单,可将复杂的网络命令行和调试过程转换为集中式控制机制下的编程过程。集中控制网络是可编程网络的一种控制方式,在本文中,默认集中控制网络具有可编程功能。

利用集中式控制机制最优化调度资源是网络资源调度的另一种模式。Handigol 等[6]在数据中心收集部分网络负载信息,通过调整交换机数量实现负载均衡;Wang 等[7]通过对 SDN 控制器建立规则组,设定控制器过载和延迟阈值,控制流量分布以及控制表规模。这两种策略更改了 SDN 的南向传输机制,对 OpenFlow 协议进行部分修改,缺少兼容性。Macapuna 等[8]额外增加 RackManager 模块处理源路由匹配,分离控制器功能,但其是静态收集信息且同步控制器需要额外机制。Handigol 等[13]设计了新的系统调度模块,提出新的范式设计路由策略;Koerner 等[14]独立出负载均衡模块,负载算法可作为插件被 NOX 调用。以上研究通过对控制器进行功能扩展实现优化调度目的,但这些扩展的协议或模块受限于路由器平台,其复杂数据处理功能模块成本昂贵,且不同的产品互不兼容。

目前,纯可控网络部署需要专业的控制器和交换机,可控网络目的是通过简化交换机的功能,解耦控制与转发功能,提高运行效率和降低成本。然而,一些网络大公司的可控网络交换机和控制节点服务器昂贵,违背了可控网络的设计初衷。为了提高运行效率,一些公司已经着手开始进行混合网络的部署和研究,并向普通网络推广。

混合网络中,流的类别通过控制节点的分层可分为可控流和非可控流。混合网络的基于流的调度,是通过控制可控流的调度来调整个混合系统的性能。Vissicchio 等[15]总结了四种混合 SDN 网络,分别为基于拓扑、基于服务、基于类别以及完整类型。Guo 等[16]在混合网络中提出基于 OSPF 的 SOTE 算法,动态分配权值减轻路径局部拥堵,提高混合网络的效率,该算法抓取即时信息,没有预测性。

1 混合网络背景

本文将日常运行的网络称为普通网络,集中控制网络是在网络技术发展过程中提出的新的网络模型,集中控制网络具有新的特性,如高聚合、可编程,弥补了普通网络的缺点。本文提到的混合网络,是集中控制网络和普通网络的混合。由于两种网络的不同特性,用原有的资源调度处理方式会带来问题。

1.1 集中控制网络

网络服务内容如视频、语音、数据服务等业务

内容越来越多样化,下一代网络需要应对大流量、富内容的承载和可控需求,需要有可编程、易调度等特点。根据使用目的不同,一个网络可能会覆盖多个类型范畴,各个类型中技术交叉混合使用。根据应用业务和层级的不同,智能可控网络可分为以下类型。

1)云:云系统中,将服务器资源作为节点资源的一种,运行在服务器上的虚拟机资源属于网络节点资源,并要满足服务器承载能力大于所有虚拟机资源。对云系统的资源调用涉及服务器、虚拟机、服务器网络、虚拟机网络等资源形式,同时也要考虑服务器开关待机方式来减少电能消耗。Jiang 等[17]通过 M/M/N 排队模型来调控虚拟机放置位置,达到系统电量消耗和网络流量的性能优化。Zhang 等[18]提出,在冷却系统、能源消耗、地理分布等因素上,小型数据中心比大数据中心更具有优势,因此,不同的数据中心混合需要考虑异构网络下资源调度等问题。Al-Fares 等[19]提出一种可扩展的云构架模型,但是面对已有的投资的云系统,无法在底层构架和传输协议上对其更改,只能通过控制器顶层进行流调度。

2) SDN:SDN 最显著特点是控制和数据平面分离,该结构能够使得信息处理高度可控、易于实现和部署。目前 SDN 系统基本以 OpenFlow[10]为主要协议进行开发和实现,在企业网、数据中心网络、接入网、网络虚拟智能等场景中表现突出。SDN 能够显著提高数据中心之间的链路利用率,如 Google 的 B4 网络中[20],使用 SDN 能够使得链路利用率达到 95% 以上;Microsoft 的 SWAN[21]能使用非专用商业交换机,维持高链路利用率的同时能够自动解决拥堵更新问题。

3)带 middlebox 策略网络:当今网络非常依赖 middlebox 提供的关键的性能、安全、策略执行能力等功能。为了实现这些功能优势,同时确保流量能够被一系列设计好的 middlebox 序列检测到,需要大量的人工努力和操作员的专业知识。通过对 middlebox 网络的混合改进,将复杂的操作成本转化到可编程网络上,用易于管理的方式进行替代,一直是 middlebox 网络的研究热点。Sherry 等[22]将开源路由器虚拟机 vyatta 放置在云端,模拟 middlebox 各种功能,将难部署管理的 middlebox 工作通过云来实现,大大简化了管理成本。Qazi 等[23]提出一个基于 SDN 的增强策略,该策略在 middlebox 网络上新建一层 overlay 来进行流量探查,简化 middlebox 处理流的过程。

4) CDN:商业视频流目前逐渐占用了大部分网络流量,为了提高服务质量,很多视频网站通过内容分发网络(Content Delivery Network,CDN)技术来加快视频浏览速度。随着业务扩展和用户增加,CDN 面临缺乏 IT 基础设施和储存空间的问题。Ling 等[24]设计出一套综合 CDN 和云技术的系统,通过云的可扩展、负载均衡机制和便于维护的优点来提高 CDN 的可发展性。同样不仅仅对于视频网站厂商,互联网服务提供者(Internet Service Provider, ISP)也需要对 CDN 进行资源分配优化。Rückert 等[25]设计出一个构架,通过上层 SDN、下层 P2P 结构来提高 ISP 厂商部署 CDN 系统的资源调度效率和平衡分配;Wichtlhuber 等[26]通过在 ISP 普通网络里设置少量的 SDN 交换机,通过域名系统(Domain Name System, DNS)在后台代理之间转向分配移动高容量长寿命流,来减少拥塞和提高传输效率。

1.2　使用混合网络原因

随着网络的扩展和业务加大,网络资源实体越来越多,组成越来越复杂,原有的功能平铺和叠加使得一些边缘和权重节点负载越来越大。可编程节点就是将控制功能独立出来,其独立的控制、全局调控优势能够做到:

1)处理各个链路浪费的带宽,充分提高网络利用率。

2)增加判断逻辑如 flow 信息,使得系统信息更加丰富,可根据控制信息做更精准决策。

3)可外接数据库,扩展数据储存和计算能力。拥有异步计算和判断信息储存能力,便于通过统计来预测大时间段内流量规律。

4)不太强烈依赖全局即时数据。

可控制编程网络引入后,也有一些需要注意的问题:

1)集中控制节点工作量大,分布式控制节点集需信息同步、需增加额外时间;

2)技术无统一标准,各家方案不兼容;

3)推广成本过大,业界每家各套方案里面交换机比一般的路由器更昂贵;

4)与一些控制器相配套的普通节点也需要更新特定接口支持。

一些 ISP 需要可控节点带来的好处对普通节点进行流控。网络从 1969 年 ARPANET 发展到现在,普通网络节点,如交换机或路由器技术发展成熟,价格便宜,由于互联网厂商和 ISP 前期普通节点投入很大,短期内不会有完全或大量转向可控制编程网络方案。混合网络是目前可行性最高的一种方案。

1.3 混合网络架构

混合网络的构架从拓扑上分类,有关键节点模式、平行网络模式以及相同节点模式。

1)关键节点模式:可控网络节点在上层组成控制网络,上层独立、不关心子网或者单独网络的具体细节,无论是采取1.1小节中何种网络类型,都是对下层普通网络资源的管理,如各种不同虚拟网、策略网络的整合。使用上层控制网络提高了网络利用率,增加了管理智能和降低了管理成本。Vissicchio 等[15]提出的四个混合 SDN 分类拓扑形式上都是该模式。关键节点模式类似骨干网,可控节点作为主要传输节点,适合大型网络公司或者 ISP 使用。此种情况中,可控节点层在普通路由器节点层之上,可控节点为普通路由器节点构成的子网络的邻接点,不同普通路由器节点网之间的流都经过可控节点。该结构如图1所示。

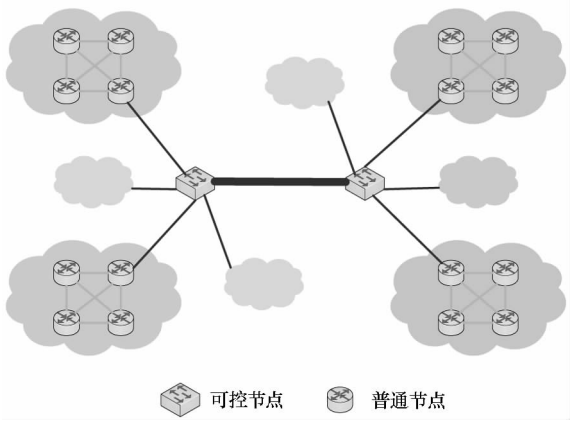


图1 关键节点模式:可控节点直接相连

Fig. 1 Key-node mode: controllable nodes are directly connected

2)平行网络模式:可控网络节点和普通网络节点各自成岛群,各个岛群通过边缘节点连接,普通路由器网络作为可控节点的虚拟传输路径,可控节点群域之间可相通。此模式适合小型网络公司、大型公司部门的同质网络使用者、利用不同公共网络传输、整合不同网络来进行备份、资源调配等。此种情况中,可控节点与普通路由器节点在同一层次。网络中可控节点作为探针或者关键拓扑位置节点,对普通路由器网络产生以点带面的作用。该结构如图2所示。

3)相同节点模式:可控节点与普通路由器节点为同一节点。此种情况中,厂商将可控节点和普通路由器节点功能结合在一起,可控节点和普通路由器节点在同一实体硬件上,作为一个更加昂贵的实体硬件代替原有的交换机或者路由器。用户采用此种方案,固然不用改变拓扑位置,可沿

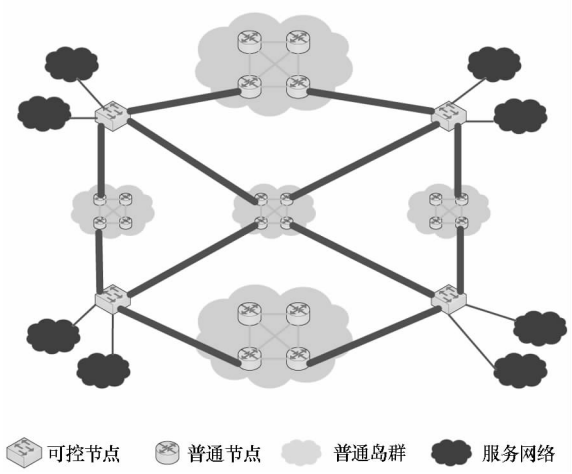


图2 平行网络模式:可控节点通过普通网络连接

Fig. 2 Parallel network mode: controllable nodes are connected through normal network

用之前的大部分配置,但是更加昂贵的硬件和更加复杂的集成配置指令以及带来的对网络配置人员的更加专业的要求,与可控网络的更加有效简捷便宜的指导思想相左,因此本节资源配置不讨论此结构。

1.4 混合网络中的流

在混合网络体系结构中,如果可控层在正常的路由器网络层以上,如图1所示,则可控交换机之间的流都是从普通路由器流中获得,其特征为:

1)不容易改变路径,当可控节点路由被改变后,须经过复杂的调配过程才能让普通路由器节点网络能够准确地将流传输到指定可控节点上。

2)一般情况,流频率不是一成不变的,如果流频率值是在合理的范围内,可以接受。

3)在系统操作员角度,流频率不是预设的,其不准确性也是不可以预测的,所以基于精确的流测量算法将无法实现。

4)普通路由器节点中有些流对可控交换机中的流没有很大的影响。

1.5 混合网络中的链路

1)如果可控交换机节点是直接线路连接的,带宽虽可能较大但数量有限,如果链路损坏,那么两个可控节点就无法连接。这在图1中有涉及。

2)如果可控交换机通过普通路由器节点网络进行连接,链路可能没有确定带宽,带宽会在一个范围内波动。如果普通路由器网络中其中一条路径损坏,可以用其他的替代,两个可控节点仍然可以连接,这在图2中有涉及。

2 混合网络的资源分配问题

混合网络中有多种不同的网络,拓扑类型和

资源调度追求目标各不相同,各网络内部情况基本自治。目前,普通网络功能成熟,普遍承担传输功能,而控制功能则由可控网络实现。根据 1.3 小节中控制网络的位置不同,分析到混合网络中出现的性能问题有:

1)两个混合模式都可能有流问题:流闪现,未预先设定的流。该问题在同质网络中同样出现,但是由于同质网络数据兼容、网络结构单一、调度便捷,通过更改底层协议或者控制器设置边界问题即可解决。而在混合网络场景中,无法简单设置底层或者高层参数,只能依据 IP 层原始数据解决。

2)在平行网络模式中,可能会出现路径问题:路径暂时改变。该问题在同质网络中不出现,只要路径出现问题,同质网络仅标记链路损坏,不出现在路径分配集合中。而混合网络中该情况经常出现。

混合网络目前都是作为控制网络出现的,面临的基本都是大型网络、网络流量集合情况,如忽略以上两个问题,控制网络采取原有资源调度算法,则会出现频繁调度网络资源、网络震荡等情况。

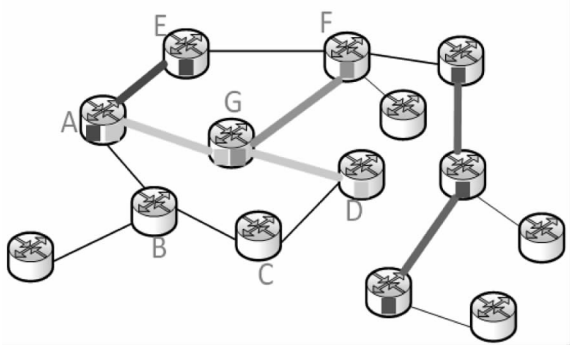
2.1　流闪现

流闪现问题出现在关键节点模式中,系统快处于饱和运行情况下,一些寿命短、容量小的流被无状态资源调度算法分配到关键路径上,造成整个路径上的流重新分配。解决方式是将这些小流分配到其他位置,或者将一些可以被移出来的大流分配到其他次优路径。

在这种情况下,存在的一个问题是:一些持续时间不长的小流或者具有一定时间间隔的短流出现,被流信息收集模块所捕获或检测到,这些流对系统本身负载影响有限。但是在混合系统中,特别是图 1 所示情况,很多流是合并在一条主干路径中,这样当快接近阈值时,就会引起大规模重新调路问题。算法根据这些流信息计算出整个系统的最大性能,来调整各节点和路径的运行状态和路径。大多数时候,算法重新分配资源时就会对系统产生影响,整个系统与新路径同步可能就会导致新运行状态和数据包的丢失,延长调整时间,提高传输成本。特别是,当系统状态进行优化转换时,改变路径和长期稳定的流都会使得成本更高。

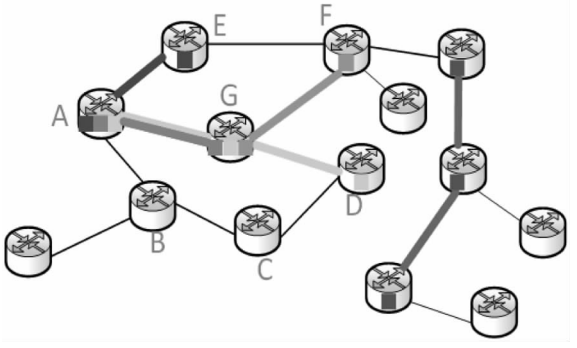
另一个问题是:如果优化算法在一个高频率工作抽样中工作的话,每个节点或路径的运行状态将会受到影响。假设在一个场景中,管理员可以根据时间周期启动优化算法,如果混合网络系统在实时模式下工作,相互作用的时间可能会很短,例如,约 10 min。在另一个场景中,优化算法只有在任务(工作)开始或接受时才有可能启动,如果任务(工作)小,优化算法也将在高频率环境下工作。当频率较高时,系统振动的概率也就增大。

在特定场景如数据中心中,可编程网络拓扑具有特定结构(如 B + 树),而混合网络相对可编程网络拓扑,具有一个更普通的形式,如图 3 所示。这些节点的拓扑结构随机构成,可认为是可编程网络交换机,连接节点的边就可认为是可编程网络交换机之间的链路,链路可以是真正的光



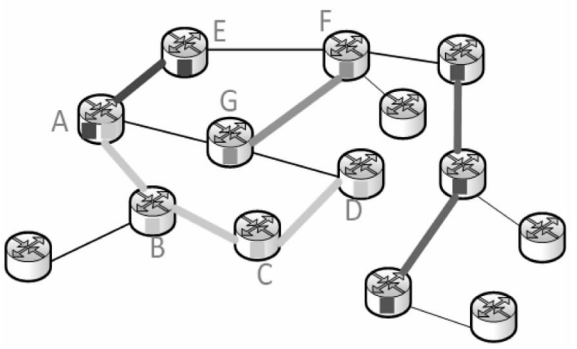
(a) 运行的系统

(a) Running system



(b) 闪现的 AG 流

(b) Bursting flow AG



(c) 使用传统资源调度算法后的系统状态

(c) The sate that after using traditional resource scheduling algorithm

图 3　流闪现情况

Fig. 3　Flow burst situation

纤直连或由下层普通路由器或交换机网络组合而成。通过节点沿着链路较粗的线可以作为流,一个或多个的流组成一个工作或任务。状态(a)(为方便描述,后文将子图(a)所示状态简称为状态(a),子图(b)、(c)类同)为 AE、GF、AGD 流在该系统中运行。如果不经常发生的流 AG 闪现只在很短的时间发生(状态(b)),系统不需要改变流 AD 的路径 AGD。如果信息收集器捕捉到状态(b),传统资源调度算法会将流 AD 的路径 AGD 改为 ABCD(状态(c)),并会持续到下一个状态变化点。系统并没有在最佳情况下运行,并且算法这次也未能发挥作用。这种情况的产生基本上是由小流造成的,在流预定义的场景中,小流不太经常出现在可编程网络系统。在数据中心内或数据中心之间,工作(任务)周期就可以定义足够长,这种情况更多出现在混合网络系统中。此种情况下,系统资源调度算法容易受到流闪现的影响,原因如下:①算法的收集器只收集有链路状态,不考虑流运行状态,如频率或完成时间;②资源调配算法是受时间周期间隔或任务的开始或结束控制,而不受流表现性能控制;③在大多数的场景下,预测都比较困难,该算法无法检测哪个流在闪现、在什么时间、在哪个链路爆发,也无法预测流的持续时间。

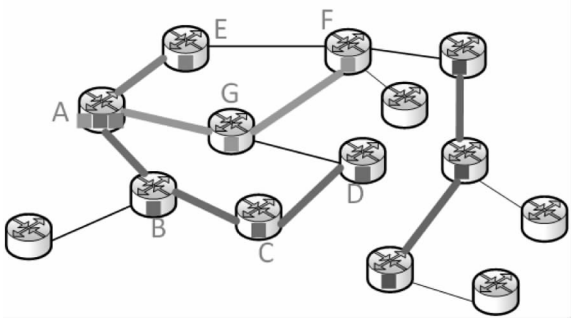
为了解决这个问题,需要有一个设计思路,系统资源配置算法的输入输出需要降低流闪现的影响。可以有两个选择:①不考虑;②标记流。在一段时间内如果流时常出现,会考虑它的影响;如果出现次数较少,就对之忽略。

在引进了流信息收集器和结果集优化选择器后,资源调度可以不受实时状态算法限制,因为选择器向算法提供的数据并不是实时的运行状态,而是需要很长时间来记录,提供的数据是具有统计性的,一次或两次的流性能并不能对选择器向算法提供的数据集有太大的影响。在选择完数据后,会对流进行评估判断并避免发生流闪现引起的后续现象。

2.2 非预测的流

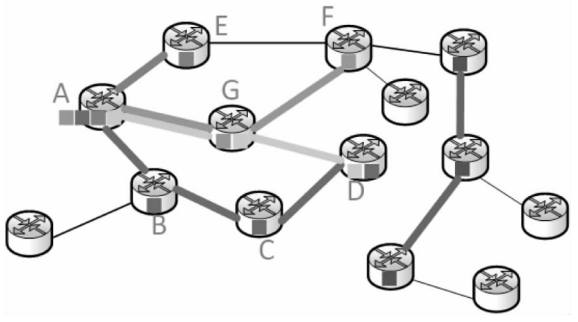
非预测的流问题是指在关键节点模式中,系统快处于饱和运行情况下,多条寿命长、容量大的流被无状态资源调度算法分配到关键路径上,造成整个路径上的流重新分配。解决方式是将这些大流进行评估,根据系统的运行状态和流的分布规律,部分分配次优路径。

在一些预设的场景中,如图 4 所示,数据中心的数据冗余备份,所有流都是预设的,所以可编程网络系统可以进行预先优化资源分配来最大限度地利用链路。对于状态(a),非预测资源调度算法将计算最好的方式:流 AD 走路径 AGD,而不是路径 ABCD 也不是路径 AEFGD,但如果流 AD 速率变大,超出路径 AG 的限制,流会导致路径 AG 堵塞。对于运行状况(a),状态(c)是第二个或第三个最佳途径:流 AD 走路径 ABCD,路径 ABCD 的承载能力可以应对流 AD 未来速率。



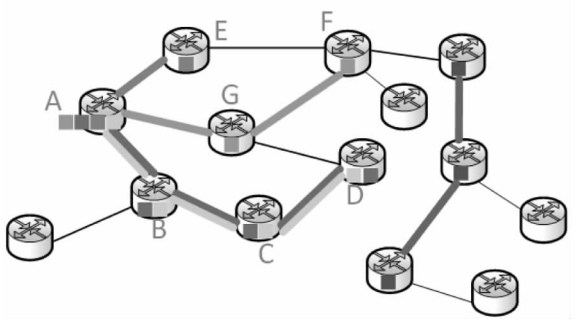
(a) 流 AD 产生前状态

(a) The sate before flow AD generating



(b) 非预测资源调度算法计算流 AD 最好路由

(b) The non-predictive resource scheduling algorithm will calculate the best route of flow AD



(c)流 AD 第二个或第三个最佳途径

(c) The second or third best route calculation result of flow AD

图 4 非预测估计流情况

Fig. 4 Non-predicted estimated flow situation

当在非预设流量矩阵情况下,这种资源调度方式就不太合适,因为纯的可编程网络系统中面对的实时流量矩阵没有规律性。在独立的纯普通网络和可编程网络中,这些非预测的流可以通过

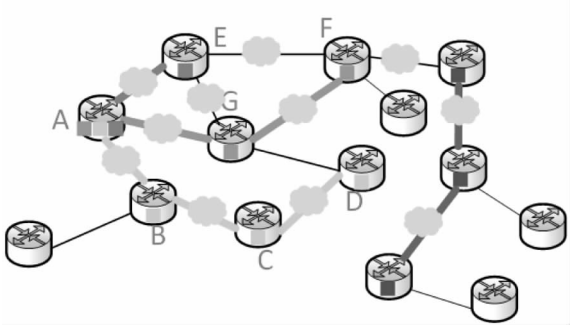
独立的流控措施进行管制,但是混合网络中各种管制措施由于软硬件环境不同,互相不能兼容和统一数据,通过确定性规划难度非常大,这个时候可通过非确定性方法进行异构网络的预测配合,通过混合网络结构可以处理一些场景中没有预估计的流。在特定的场景,如数据中心和 CDN,服务提供商没有将所有系统改造成纯 SDN,但会使用混合 SDN 以提升现有系统使其更有效率。在这些场景中,尽管具体的某个网段或者路径上的实时的流量矩阵是不能够预测的,但是整体的流量有一定的统计特性,这种场景运用在图 1 结构中,可编程网络系统通过收集器来统计或者由人工输入流量统计特性,则混合网络系统能够更好地进行资源检测和分析,从而进行优化调度。

因此,可以提供最佳子集的流分析模块来帮助算法为某个流选择最适合的通路。为了解决这个问题,应该减少流分析模块,即资源配置管理员在高水平上分析流设置,不分析个体流。这个场景非常适合基于服务产生的结构:在上一层的可编程网络节点和在下一层的普通路由器节点。该方法结合了所有未预先设定的流,能将流闪现的影响降到最低。在最坏的情况下,所有的小流都向一个方向聚集,那就是,所有的小流速同时增加或减少。减少流分析模块并不能完全解决这一问题,但可以在很大程度上解决它。原因是可编程网络节点具有更宽的带宽,相应地,小流所造成的影响就会更小。

2.3　路径堵塞或连接震荡场景

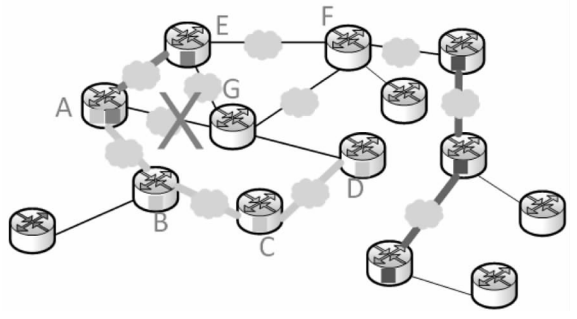
路径堵塞或连接震荡场景问题是指在平行网络模式中,传输网络不稳定,造成链路权值发生抖动变化,流分配路径随之抖动变化。解决方式是将这些边缘路径进行评估,同时评估流的可承受转移代价,根据系统的运行状态和流的分布规律,部分流分配次优路径。

1)路径未能连接或出现堵塞:在混合网络系统下,路径可以是由普通路由器节点连接的子网。如图 5 所示,如果一些子网路径拥塞,路径 EA 或 AG 同时不能连接,在一段时间后子网会通过自我修复重新连接,连接/堵塞时常发生。流 AF 非常重要,是长流,时序要求高,不能被频繁的改变造成不稳定。如果按照最优路径算法,应走路径 AGF。因为故障恢复或操作策略的变化,路径被改变到另一条线路,路径在状态(b)、状态(c)间切换。如果路径 EG、AG 经常变化,流 AF 这样权值高的流最好的办法是不选择 AGF,而是选择次优路径 AEF。



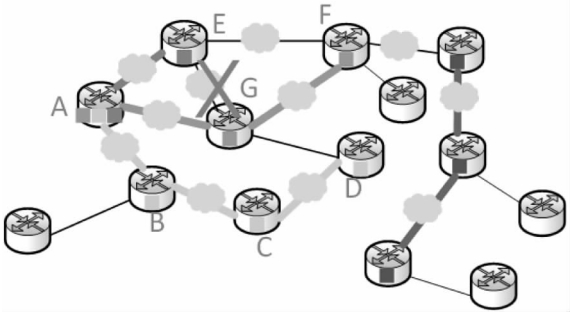
(a) 系统正常运行状态

(a) System running normally



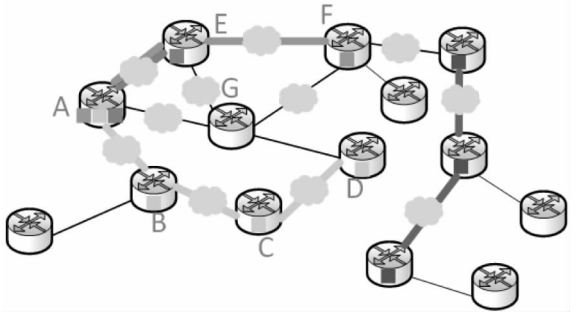
(b) 链路 AG 堵塞

(b) Path AG is blocked



(c) 链路 EG 堵塞

(c) Path EG is blocked



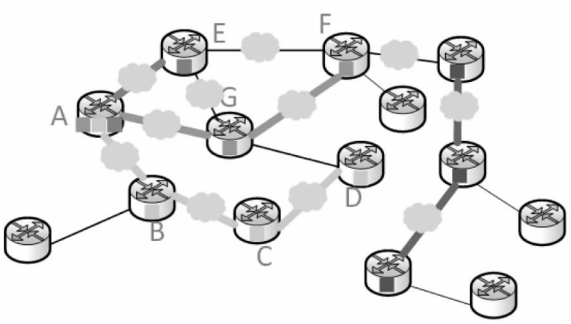
(d) 流 AF 选择 AEF 而不选择 AGF

(d) Flow AF do not select route AEF instead of AGF

图 5　路径堵塞场景

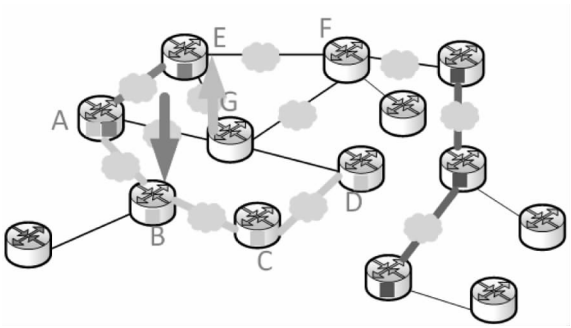
Fig. 5　Path blockage situation

2)路径容量过载震荡:更普遍的是,路径带宽性能因为较低的网络工作状态而发生改变。如图 6 所示,A 和 G、G 和 F 之间的子网工作状态一直都在变化。因此 AG 或 GF 路径性能可能会发



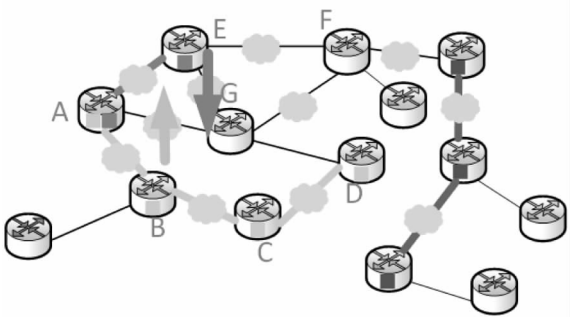
(a) 系统正常运行状态

(a) System running normally



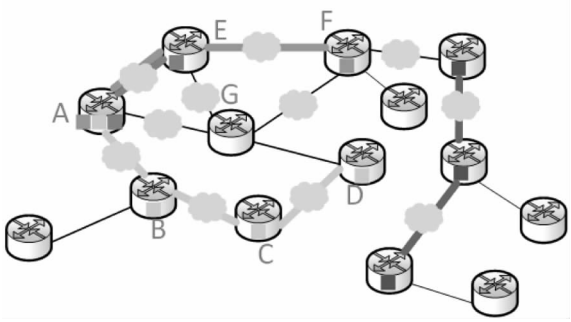
(b) AG 性能降低,EG 性能提高

(b) The performance of path AG decreases, the performance of path EG improves



(c) EG 性能降低,AG 性能提高

(c) The performance of path EG decreases, the performance of path AG improves



(d) 流 AF 最好的办法是不选择 AGF

(d) The best way of flow AF is to not select AGF

图 6 路径连接震荡情况

Fig. 6 Path shock situation

生变化。如果流 AGF 的带宽让 AGF 路径带宽达到上限,那么即使 AG 或 GF 路径发生很小的变化,也会影响到 AGF 流。同时,AEF 路径稳定但不是最好的选择路径。非随机资源调度算法为 AF 流找到的最好的路径为状态(a),但实际上系统不能通过 AGF 路径服务 AF 流,所以就选择了次优的 AEF 路径。

如果直接用原有可控制网络的调度算法,未考虑混合网络路径情况,得到结果是可选路径集减少,并使得选择不稳定节点的路径发生震荡。发生此情况的原因是:原有可控制网络的调度算法是基于一个前提——节点之间的连接是稳定的,但在混合网络系统,可编程网络节点之间的链路不是通过一条线路或一个设计良好的稳定网络直接连接的,系统运行状态受每个可编程网络节点的邻接子网影响。

3 模型和评价标准

3.1 字母标记

本文模型和定义分为三类:第一类是图的模型定义;第二类是在某个时刻下,网络的资源运行快照,适用于无状态模型;第三类是在某个时间段内,网络的资源运行情况,适用于预测模型。涉及的字母标记如表 1 所示,表 1 仅定义网络资源,资源规划算法涉及的字母表在算法章节中定义。

一个网络系统中资源集合可定义为 $Resource = \{Net, Flow\}$,其中 $Net = (V, E)$ 表示网络 Net 由节点集合 V 和链路集合 E 组成。N_V 为网络节点数量,N_E 为网络链路数量,i,j 分别为节点的标号,且有 $\boldsymbol{v}_i, \boldsymbol{v}_j \in V, e_{v_i,v_j} \in E$,为了简化标记,涉及的链路的节点标记用节点号代替,即有 $e_{i,j} \in E$。定义节点 $\boldsymbol{v}_i$ 的相邻节点集合为 $VJ_i = \{\boldsymbol{v}_m \mid m \in V, e_{m,i} \in E, e_{i,m} \in E\}$,节点 $\boldsymbol{v}_i$ 进出边集合分别为 $EJ_i^{\text{in}} = \{e_{m,i} \mid e_{m,i} \in E\}$, $EJ_i^{\text{out}} = \{e_{i,m} \mid e_{i,m} \in E\}$。$Flow$ 表示流集合,定义 $Flow = \{f_{s,d}, \boldsymbol{v}_s, \boldsymbol{v}_d \in V\}$,其中 $f_{s,d}$ 表示起点标号为 s、终点标号为 d 的流。定义流 $f_{s,d} = \{f_{s,v^1}, f_{v^1,v^2}, \cdots, f_{v^{k-1},v^k}, f_{v^k,v^d} \mid k \in K_{f_{s,d}}, \boldsymbol{v}_s, \boldsymbol{v}_d, \boldsymbol{v}^k \in V\}$,其中 $K_{f_{s,d}}$ 是流 $f_{s,d}$ 被规划算法分配到的路径节点集。同样方式,可以定义链路容量集合 $CapE = \{c_{i,j}, \boldsymbol{v}_i, \boldsymbol{v}_j \in V\}$,$c_{i,j}$ 表示链路 $e_{i,j}$ 能够接受流量的最大带宽;定义链路上所有流集合 $Flow_{e_{i,j}} = \{f \mid f \in Flow, e_{i,j} \in f\}$;定义节点的处理能力集合 $CapV = \{TH\boldsymbol{v}_i, \boldsymbol{v}_j \in V\}$,$TH\boldsymbol{v}_i$ 表示节点 $\boldsymbol{v}_j$ 处理的最大能力阈值。

系统运行中,每个时刻 t 系统中各个资源运行状态是不同的,因此引入时间序列 $T = \{t\}$ 来标记每个时刻下的资源运行状态。对于每个节点,

表 1　网络资源字母标记和意义

Tab. 1　Letter mark and meaning of network resources

字母缩写	表示意义
$Resource$	网络资源集合
Net	整个网络
V	网络节点集合
N_V	网络节点数量
E	网络链路集合
N_E	网络链路数量
$\boldsymbol{\upsilon}$	单个节点
e	单个链路
i	节点标号 i
j	节点标号 j
VJ_i	节点 i 邻接节点集合
EJ_i^{in}	节点 i 入边集合
EJ_i^{out}	节点 i 出边集合
$Flow$	流集合
f	流
K	流分配路径节点集合
k	流分配路径节点标号
s	流起始节点标号
d	流目的节点标号
$Flow_{e_{i,j}}$	链路 $e_{i,j}$ 上所有流集合
$CapE$	链路容量集合
c	链路容量
$CapV$	节点阈值集合
THv_i	节点 υ_i 的处理能力阈值
T	时间序列
t	时间序号
$\boldsymbol{\upsilon}_{\text{sta}}$	t 时刻节点 i 状态集合
υr_i	t 时刻节点 i 运行负载
VS_i	节点 i 状态集合
V_{state}	节点状态集合
e_{sta}	t 时刻链路 $e_{i,j}$ 状态
$er_{i,j}$	t 时刻链路 $e_{i,j}$ 运行负载
$ES_{i,j}$	链路 $e_{i,j}$ 状态集合
E_{state}	链路状态集合
f_{sta}	t 时刻流 $f_{s,d}$ 状态
$FS_{s,d}$	流 $f_{s,d}$ 状态集合
$b_{s,d}$	流 $f_{s,d}$ 流量带宽
F_{state}	流状态集合
M_t	t 时刻系统流量矩阵
MV_t	t 时刻系统流量矩阵具有流量的节点集合
MF_t	t 时刻系统流量矩阵具有流量的节点对集合
$Metic$	系统流量矩阵集合
$state$	t 时刻系统资源状态
$ResourceState$	多个时刻序号系统资源状态集合
$t_{b,e}$	时间段,从时刻 t_b 到时刻 t_e
$\Delta t_{b,e}$	从时刻 t_b 到时刻 t_e 的时间
$VS_i^{b,e}$	时间段 $t_{b,e}$ 下节点 i 状态集
$\overline{\upsilon r_i}$	时间段 $t_{b,e}$ 下节点 i 平均状态
$THv_i^{b,e}$	时间段 $t_{b,e}$ 下节点 i 平均负载阈值上限
$ES_{i,j}^{b,e}$	时间段 $t_{b,e}$ 下链路 $e_{i,j}$ 状态
$e_{\text{sta}}^{b,e}$	时间段 $t_{b,e}$ 下链路 $e_{i,j}$ 平均状态
$\overline{er_{i,j}}$	时间段 $t_{b,e}$ 下链路 $e_{i,j}$ 平均负载
$c_{i,j}^{b,e}$	时间段 $t_{b,e}$ 下链路 $e_{i,j}$ 容量取值
$FS_{s,d}^{b,e}$	时间段 $t_{b,e}$ 下流 $f_{s,d}$ 状态集合
$f_{\text{sta}}^{b,e}$	时间段 $t_{b,e}$ 下流 $f_{s,d}$ 平均状态
$b_{s,d}^{b,e}$	时间段 $t_{b,e}$ 下流 $f_{s,d}$ 带宽取值
$\boldsymbol{\lambda}_{s,d}^{b,e}$	时间段 $t_{b,e}$ 下流 $f_{s,d}$ 出现概率
$State^{\Delta t_{b,e}}$	时间段 $t_{b,e}$ 下系统的资源平均状态
$ResourceState^{\Delta t_{b,e}}$	时间段 $t_{b,e}$ 下系统的每个时刻下资源状态的集合

定义节点状态 $\boldsymbol{\upsilon}_{\text{sta}} = \{t, \boldsymbol{\upsilon}_i, \upsilon r_i, THv_i\}$,$\upsilon r_i$ 表示节点 $\boldsymbol{\upsilon}_i$ 运行负载,每个节点根据时序组成节点状态列表 $VS_i = \{\boldsymbol{\upsilon}_{\text{sta}}, t \in T\}$,对于网络中所有节点,定义节点状态集 $V_{\text{state}} = \{VS_i, \boldsymbol{\upsilon}_i \in V\}$。对于每个链路,定义链路状态 $e_{\text{sta}} = \{t, e_{i,j}, er_{i,j}, c_{i,j}\}$,$er_{i,j}$表示链路 $e_{i,j}$的运行负载,每个链路根据时序组成链路状态列表 $ES_{i,j} = \{e_{\text{sta}}, t \in T\}$,对于网络中所有链路,定义链路状态集 $E_{\text{state}} = \{ES_{i,j}, e_{i,j} \in E\}$。对于每个流 $f_{s,d}$,定义流状态 $f_{\text{sta}} = \{t, f_{s,d}, b_{s,d}\}$,其中 $b_{s,d}$表示 $f_{s,d}$流量大小,每个流可根据时序组成流状态列表 $FS_{s,d} = \{f_{\text{sta}}, t \in T\}$,对于网络中所有流,定义流状态集合 $F_{\text{state}} = \{FS_{s,d}, \boldsymbol{\upsilon}_s, \boldsymbol{\upsilon}_d \in V\}$。网络状态可通过不同角度描述,描述在每个时刻系统整个流量状态可通过流量矩阵表示,每个流量矩阵是快照,不同时刻快照下的流量矩阵中流量是

不一定相同的,为了表述准确和简洁,定义时刻 t 流量矩阵为 $M_t = \{t, b_{s,d}, \boldsymbol{v}_s, \boldsymbol{v}_d \in V\}$,流量矩阵集为 $Metic = \{M_t, t \in T\}$。根据以上的各个资源定义,可以描述资源的运行状态为 $State = \{t, Net, Flow, CapE, CapV, M_t\}$,资源运行状态集合为 $ResourceState = \{State_t, t \in T\}$。其中 $Flow$ 和 M_t 有流量的表述冗余,但大大简化了以后模型不同角度分析,故保留。

在预测场景中,面对的是多个时刻形成的时间段内的资源分配情况,定义时间段 $t_{b,e} = \{t_b, t_{b+1}, \cdots, t_{e-1}, t_e, e, b \in T\}$,时间间隔为 $\Delta t_{b,e} = t_e - t_b$, $e, b \in T$,则在时间段 $t_{b,e}$ 内,各个资源定义如下:对于节点 $\boldsymbol{v}_i$,定义时间序列内状态集合 $VS_i^{b,e} = \{\boldsymbol{v}_{\text{sta}}, t \in t_{b,e}\}$,时间序列内平均状态为 $\boldsymbol{v}_{\text{sta}}^{b,e} = \{t_{b,e}, \boldsymbol{v}_i, \overline{vr_i}, THv_i^{b,e}, e, b \in T\}$,其中 $\overline{vr_i}$ 为在该时间段内资源对应均值。对于链路 $e_{i,j}$,定义时间序列内状态集合 $ES_{i,j}^{b,e} = \{e_{\text{sta}}, t \in t_{b,e}\}$,时间序列内平均状态为 $e_{\text{sta}}^{b,e} = \{t_{b,e}, e_{i,j}, \overline{er_{i,j}}, c_{i,j}^{b,e}, e, b \in T\}$,其中 $\overline{er_{i,j}}$ 为在该时间段内资源对应均值,$c_{i,j}^{b,e}$ 为在该时间段选取值。对于流 $f_{s,d}$,定义时间序列内状态集合 $FS_{s,d}^{b,e} = \{f_{\text{sta}}, t \in t_{b,e}\}$,时间序列内平均状态为 $f_{\text{sta}}^{b,e} = \{t_{b,e}, f_{s,d}, b_{s,d}^{b,e}, \lambda_{s,d}^{b,e}, e, b \in T\}$,其中,$b_{s,d}^{b,e}$ 为在该时间段选取值,$\lambda_{s,d}^{b,e}$ 为在该时间段内流的出现概率。根据以上的各个资源定义,可以描述时间段内资源运行状态集合为 $ResourceState^{b,e} = \{State_t, t \in t_{b,e}\}$,平均状态为 $State^{\Delta t_{b,e}} = \{\Delta t_{b,e}, Net, Flow^{\Delta t_{b,e}}, CapE^{\Delta t_{b,e}}, CapV^{\Delta t_{b,e}}, Metic_{t_{b,e}}\}$,其中各项取值根据规划算法不同而不同。

无论在什么情况,一个 Net 系统正常运行,必须满足以下约束:

$$er_{i,j} = \sum_{f_{i,j}} b_{i,j} \leqslant c_{i,j} \tag{1}$$

$$\boldsymbol{v}r_i = Cost(\boldsymbol{v}_i) \leqslant THv_i \tag{2}$$

$$\sum_{e_{m,i} \in EJ_i^{\text{in}}} f_{m,i} - \sum_{e_{i,m} \in EJ_i^{\text{out}}} f_{i,m} = \delta_i, \quad \forall \boldsymbol{v}_i, \boldsymbol{v}_j \in V \tag{3}$$

式中,

$$\delta_i \begin{cases} 1 & i = s \\ 0 & i = d \\ 0 & i \neq s, i \neq d \end{cases}$$

$$f_{i,j} \geqslant 0, \quad \forall \boldsymbol{v}_i, \boldsymbol{v}_j \in V \tag{4}$$

式(3)和式(4)为流量守恒和链路守恒约束,其中节点 $\boldsymbol{v}_i$ 的负载消耗 $Cost(\boldsymbol{v}_i)$ 满足

$$Cost(\boldsymbol{v}_i) = \alpha \cdot \sum_{f_{i,j} \in EJ_i^{\text{in}}} b_{i,j} + \beta \cdot \sum_{f_{i,j} \in EJ_i^{\text{out}}} b_{i,j} + \theta(\boldsymbol{v}_i) \tag{5}$$

式中,$\theta(\boldsymbol{v}_i)$ 表示非流引起的节点负载,参数 α 和 β 调节节点 $\boldsymbol{v}_i$ 输入流和输出流的负载权重,本文中不着重表述节点的负载情况,输入输出流负载权重相同,故设置式(5)为简单情况,约束条件式(2)可变为:

$$\boldsymbol{v}r_i = \sum_{f_{i,j} \in EJ_i^{\text{in}}} b_{i,j} + \sum_{f_{i,j} \in EJ_i^{\text{out}}} b_{i,j} \leqslant THv_i \tag{6}$$

3.2 系统评价模型

本小节工作是:提出五个系统目标来评价系统资源分配性能,通过不同目标进行数学表述来量化该目标,分析混合网络中原有资源规划方案问题,提出针对性资源规划方案。

3.2.1 网络利用率

网络利用率越大,相同网络流量矩阵情况下,系统完成任务时间越少;在相同网络任务队列情况下,能够完成更多的任务数。

对于每个链路 $e_{i,j}$,定义网络利用率(Edge Usage, EU)为:

$$EU_{i,j} = \sum_{f_{m,n} \in Flow_{e_{i,j}}} b_{m,n} / c_{i,j}$$

整个网络链路利用率为其所有链路加权平均:

$$U_{Net} = \frac{1}{N_E} \sum \sum_{f_{m,n} \in Flow_{e_{i,j}}} b_{m,n} / c_{i,j} \tag{7}$$

算法利用率为 $O(N_E)$,可以通过 U_{Net} 来评价网络效率,但是期望值并不能说明网络的热点链路分布情况,需要提出链路利用率方差来评价其离散程度:

$$VAR(EU) = \frac{1}{N_E} \sum \Big(\sum_{f_{m,n} \in Flow_{e_{i,j}}} b_{m,n} / c_{i,j} \Big)^2 - U_{Net}^2 \tag{8}$$

传统无状态算法中,链路 $e_{i,j}$ 权值越高,路径分配算法就会将流全都分配在该链路上。因此,如果一个流集 $\{f\}$,最短路径的部分路段聚集在节点 $\boldsymbol{v}_i, \boldsymbol{v}_j$ 周围,则容易造成链路 $e_{i,j}$ 堵塞。

3.2.2 节点阻塞数目

节点阻塞数目直接影响到流的重新分配,无论是对新流的分配,还是对老流的重新分配,都要考虑节点的阻塞情况,资源调度目标是尽量减少阻塞数目。节点阻塞数目与网络利用率和网络任务完成时间间接相关。

定义在时间段 $\Delta t_{u,v}$ 内,节点阻塞数目

$$Bl_{v_i} = \sum_{t \in \Delta t_{u,v}} \sum_{v_i \in MV_t} \oslash \boldsymbol{v}_i$$

$$\oslash \boldsymbol{v}_i = \begin{cases} 0 & f \in MV_t, \ \boldsymbol{v}r_i < THv_i \\ 1 & f \in MV_t, \ \boldsymbol{v}r_i \geqslant THv_i \end{cases} \tag{9}$$

3.2.3 网络任务完成时间

在时间段 $\Delta t_{u,v}$ 内,对于一组流集 $Flow^{u,v}$,集

合内所有流完成的时间是该网络系统资源分配算法的最直观评价。定义 $T_{Flow^{u,v}}$ 为在时间段 $\Delta t_{u,v}$ 内任务完成时间:

$$T_{Flow^{u,v}} = \sum_{t \in \Delta t_{u,v}} \sum_{f \in Flow^{u,v}} t_f \tag{10}$$

式中,t_f 为流 f 在时间序列单位 t 时流量矩阵 M_t 对应的流量。

值得注意的是,流量矩阵时间序列单位 t 同算法的激发时间序列单位 τ 是不同的。在流量矩阵更新时间 Δt 内,可以激发多次流量规划算法,有多个 $\Delta\tau$;同样,也可以多个流量矩阵后才激发流量规划算法,即在 $\Delta\tau$ 内有多个 Δt。

1) $\Delta\tau \leqslant \Delta t$ 时,

$$t_f = \Delta t \cdot \varepsilon + \Delta\tau \cdot rd_f$$

$$\varepsilon = \begin{cases} 1 & f \in M_t \\ 0 & f \notin M_t \end{cases}$$

其中,rd_f 为流 f 在算法调度时不能被分配的次数,原因是链路被阻塞或者超过节点负载阈值,流 f 得不到链路或者节点。在调度算法被激活次数期间,可能会发生多次流得不到资源情况,$rd_f = \varnothing_{v_i},\ v_i \in K_f$。

2) $\Delta\tau > \Delta t$ 时,假设 $\Delta\tau = n\Delta t + t_{n+1}$,则有

$$t_f = \begin{cases} \Delta t \cdot \varepsilon, k \in [0, n] \\ \Delta t \cdot \varepsilon + \Delta\tau \cdot rd_f, k \in [n, n+1] \end{cases}$$

$$\varepsilon = \begin{cases} 1 & f \in M_t \\ 0 & f \notin M_t \end{cases}$$

其中,n 为 $\Delta\tau$ 与 Δt 的倍数,当时刻 t 序号为 $\Delta\tau$ 余数时,激发资源规划算法,此时如果流 f 分配不到资源则需等待。由于规划算法只发生一次,故 $rd_f = 1$。

3.2.4　策略变化代价

网络发生故障或者主动发起更改策略时,系统会更改路径,策略变化代价与更改路径数目成正比。在 t 时刻移动的流会影响 $t+1$ 时刻的系统状态,移动的流不一定会影响后续状态。要减少策略变化代价,可以减少策略变化次数,也减少每次算法规划下移动的流路径数目,但是同时需要保持系统利用率,不然失去系统优化意义。

策略变化由以下情况触发:

情况 1:新流 $f_{\text{new}}(s,d)$ 生成时,系统创建一条路径 $K_{f_{\text{new}}}$,$K_{f_{\text{new}}}$ 所有节点均能负载该流。

情况 2:新流 $f_{\text{new}}(s,d)$ 生成时,系统创建一条路径 $K_{f_{\text{new}}}$,$K_{f_{\text{new}}}$ 至少有一个节点负载该流超过阈值。

情况 3:算法对流 $f_t(s,d)$ 进行移动,在限定移动次数内,找到可选路径 P_a,P_a 所有节点均能负载该流。

情况 4:算法对流 $f_t(s,d)$ 进行移动,在限定移动次数内,找到可选路径 P_a,P_a 至少有一个节点负载该流超过阈值。

传统方法中,情况 2 和情况 4 如果没有可选路径,即 $K_{f_{\text{new}}}$ 或者 P_a 为空,系统就认为无法服务。设 v_k 为 $K_{f_{\text{new}}}$ 路径上的某一个超过阈值的节点,流 $f_t(s,d)$ 当前所用路径为 K_{f_t},v_a 为 P_a 路径上的某一个超过阈值的节点。在集中控制情况下,系统可尝试将超过阈值的节点 v_k 或者 v_a 上的流转移,释放节点承载能力,从而承载流 $f_{\text{new}}(s,d)$ 或者 $f_t(s,d)$ 服务。流的可移动性可通过对更改流表的节点计数来参考。

情况 2、情况 4 可规约到以下命题:设流 $f_t(s,d)$ 当前所用路径为 K_{f_t},通过算法找到可选路径 P_a,P_a 有非空集合 $S_{ov}(P_a)$,$\forall v_a,\ vr_a \geqslant Thv_a$。

该命题解决算法见算法 2 步骤 8 ~ 17。

对流 $f_t(s,d)$ 进行移动评估,就是求得确定 $S_{ov}(P_a)$ 节点移动最小值,而每个节点上的移动最小值取决于节点上流的移动评估值。这种方式定义为流可移动性评估(Flow Movable Evaluation, FME)。FME 公式定义如下:

$$Me_{f_t} = \begin{cases} \sum_{v_a \in S_{ov}(P_a)} Me_{v_a}, \mid \forall v_a,\ vr_a \geqslant Thv_a \\ C(S_{\text{add}}) + C(S_{\text{del}}), \mid \forall v_a,\ vr_a < Thv_a \end{cases}$$

$$Me_{v_a} = \sum_{f \in Flow_{e_{i,a}} \cup Flow_{e_{a,j}}} Me_f \mid e_{i,a} \in EJ_a^{\text{in}}, e_{a,j} \in EJ_a^{\text{out}}$$

其中,$C(S)$ 表示集合 S 中所有节点的负载函数,为了表述简便,往往用节点数目表示。由于两式是迭代的,当进入情况 1、情况 3,表明找到合适解决方案,停止迭代。如果一直没有找到合适解决方案,会一直迭代寻找下去,因此,需要引入迭代数 M^{up} 控制,否则会引起计数爆炸。

对每个过载节点 $v_a \in S_{ov}(P_a)$ 上承载的所有流集合 $X(v_a)$ 中每条流 X_i,原有路径为 $P_i(X_i)$,得到可选路径 $P_a(X_i)$ 后,有相同路径节点集合为 $P_i(X_i) \cap P_a(X_i)$。这对于原路径 $P_i(X_i)$,需要在流表中减少 X_i 流表项的节点集合为 $S_{\text{del}} = P_i(X_i) - P_i(X_i) \cap P_a(X_i)$,需要在流表中增加 X_i 流表项的节点集合为 $S_{\text{add}} = P_a(X_i) - P_i(X_i) \cap P_a(X_i)$。

流 X_i 转移代价 $Me(X_i)$ 为:

$$Me(X_i) = C(S_{\text{add}}) + C(S_{\text{del}}) \tag{11}$$

各节点调整流转移代价有着细微差别,与各节点流表以及整个系统的运行状态有关,本算法认为每个更改节点消耗相等。

3.2.5　发生分流次数和分流节点数

当新流产生时,如果通过分配路径算法不能

找到可承载的路径,就可能触发分流事件。该事件发生有两种情况:一是节点超过阈值,本节点触发分流;二是在 FME 算法引入时,流经过策略变化后,需要改变路径的流所经过路径的节点触发分流。t_a 时刻和 t_b 时刻之间,分流次数 Rr_{e_i} 根据以上两种情况可定义为:

$$Rr_{e_i} = \sum_{t\in\Delta t_{u,v}}\sum_{f_i\in MF_t}\varnothing f_i + \sum_{t\in\Delta t_{u,v}}\sum_{f_j\in Flow_{e_{m,n}}}\varnothing f_j$$

$$\varnothing f_i = \begin{cases}0 & f\in MF_t,\ vr_i < THv_i^{a,b}\\ 1 & f\in MF_t,\ vr_i \geqslant THv_i^{a,b}\end{cases}$$

$$\varnothing f_j = \begin{cases}0 & f\in MF_t,\ e_{m,n}\notin P_a\\ 1 & f\in MF_t,\ e_{m,n}\in P_a\end{cases}$$

分流后,流的路径部分可能会改变,引起的节点需要改变状态数量为:

$$Rr_{v_i} = \sum_{t\in\Delta t_{u,v}}\sum_{v_i\in K_{fi}}\sum_{f_i\in MF_t}\varnothing v_i + \sum_{t\in\Delta t_{u,v}}\sum_{v_i\in K_{fj}}\sum_{f_j\in Flow_{e_{m,n}}}\varnothing v_j$$

$$\varnothing v_i = \begin{cases}1 & f\in M_t,\ vr_i < THv_i^{a,b}\\ 0 & f\notin M_t,\ vr_i \geqslant THv_i^{a,b}\end{cases}$$

$$\varnothing v_j = \begin{cases}0 & f\in MF_t,\ e_{m,n}\notin P_a\\ 1 & f\in MF_t,\ e_{m,n}\in P_a\end{cases}$$

可以看到,流 f_i 分流发生后,其引起的迁移节点消耗分为改变流 f_i 上节点 v_i 的消耗和在可选路径 P_a 上节点 v_j 的消耗。流 f_i 发生分流后,对原有的权值大的流的干扰,如阻塞节点等情况,有可能引起在大权值流路径上的权值小的流的分流。为了避免这种迭代情况发生,在 4.2 节中设置了阈值 M^{up} 来控制迭代次数。

4 资源规划算法

根据第 3 节分析,需要在资源调度算法中满足以下目标:①系统处于饱和状态时,尽量减少流调度的次数;②在一段时间内,满足流传输流量最大化情况下,流变换尽量少。资源调度算法工作在可控制网络,调控的是可控制网络的流,间接调控普通网络的流,普通网络内部运行独立路径算法。流信息可被边界服务节点读取,可提供连接网络,异构网络无须知道其他层或者邻接的网络资源。

4.1 关键节点模式下的算法

关键节点模式下,保持链路利用率最大,同时网络分布更平均。根据式(7)和式(8),可设置防止过饱和算法(Preventing OverSaturation algorithm, POS),可设置目标函数最大网络利用率:

$$\max\ U_{Net} \tag{12}$$

定义 $\gamma_i = 1 - vr_i/THv_i$ 为节点空闲率,设置优化约束为:

$$\gamma_i \geqslant \sigma \tag{13}$$

$$\frac{1}{\eta} \leqslant \left|\frac{\gamma_i - \gamma_{i-1}}{\gamma_i - \gamma_{i+1}}\right| \leqslant \eta \tag{14}$$

同时满足系统运行的基本约束。

约束式(12)保证节点能有负载应对小流,使得小流不影响重新分配已运行流,可解决流闪现问题。σ 为调节参数,依照系统拓扑和流运行情况,设置经验值。约束式(13)保证相邻节点空闲率处于相同水平,这样使得大流能够有位置安置,可解决未预测大流问题。η 为调节参数,依照系统拓扑和流运行情况,设置经验值。为了计算方便,η 值同时可作为约束式(8)算法前后次数方差使用,具体见算法过程。约束式(13)算法效率为 $O(N_v)$,结合式(7)和式(8),POS 算法效率为 $O(N_E N_v)$。

POS 算法过程如算法 1 所示。

算法 1 POS 算法过程

Alg. 1 POS algorithm

Input: $Flow_{e_{i,j}}$, t, M_t, $CapE$, $CapV$, Net, σ, η, $state_{t-1}$

Output: $state_t$, 系统资源分配方案

```
1. 初始化各值,记 VAR(EU)_0 = 0, n = 0
2. if 触发开关 = = true then
3. 记录本次为第 n = n + 1 次,取得每个节点空闲率 γ_i
4. 根据约束式(13)、式(14)筛选可分配路径节点集 V_filt
5.   if V_filt = = null then
6.     return result = null
7.   else
8.   根据目标函数式(12),求出极值,得出分配节点集合 result(V_n),记录此次式(8)结果 VAR(EU)_n
9.     if VAR(EU)_n ≥ ηVAR(EU)_{n-1} then
10.        return result(V_n)
11.    else
12.        return result = null
13.    end if
14.  end if
15. end if
```

4.2 平行网络模式下的算法

通过 3.2.4 小节策略变化代价模型可分析,在当前节点 v_a 上所有流集合 $X(v_a)$ 中,对流权值排序,从小到大选出 r 个流移出,空出流量为:

$$b(X) = \sum_{i=1}^{r} b_{X_t}, X = (X_{\text{new}}, X_t)$$

其满足条件为:$b_{f_t} \geqslant M(X)$。

FME 公式每迭代一次,M^{up} 减 1,当 $M^{\text{up}}<0$ 时,停止迭代。$M^{\text{up}}<0$ 时,P_a 无解表示没有可选路径,则判定该流无法移动,只能选择不服务。其算法复杂度为 $O(n^{M^{\text{up}}-1})$。

该模式下,保持链路利用率最大,同时网络分布更平均。根据式(7),可使用有流可移动性评估的防止过饱和算法(Preventing Over-Saturation algorithm with Flow Movable Evaluation, POS-FME),可设置极值目标函数:

$$\max \sum U_{Net}^{n,n+1}, e<n<d-1 \tag{15}$$

设置优化约束除了式(12)和式(13)外,增加 FME 流评价约束:

$$b_{f_t} \geqslant M(X) \tag{16}$$

同时满足系统运行的基本约束。其中约束式(16)的算法效率为 $O(N_E^{M^{\text{up}}-1})$,结合 POS-FME 目标函数,可得其算法效率为 $O(N_E^{M^{\text{up}}-1}N_E N_v)$,即为 $O(N_E^{M^{\text{up}}-1}N_v)$。为了控制算法效率在一定限度内,经验值 M^{up} 一般取 2 或 3,本文实验取值 $M^{\text{up}}=3$。

POS-FME 算法过程如算法 2 所示。

算法 2　POS-FME 算法过程

Alg. 2　POS-FME algorithm

Input: $Flow_{e_{i,j}}, t, M_t, CapE, CapV, Net, \sigma, \eta, state_{t-1}, M^{\text{up}}$

Output: $state_t$, 系统资源分配方案

初始化各值,记 $VAR(EU)_0=0$, $n=0$
if 触发开关 = = **true then**
　记录本次为第 $n=n+1$ 次,取得每个节点空闲率 γ_i
　根据约束式(13)、式(14)筛选可分配路径节点集 V_{filt}
　if V_{filt} = = *null* **then**
　　return *result* = *null*
　else
　　对约束式(16)进行计算
　　初始化循环次数 $j=0$
　　while $M^{\text{up}} \geqslant 0$ **do**
　　M^{up} − −
　　$j=j+1$
　　对每个节点,根据运行流权值排序,选择 r 个流,记录此次 $b(X)_j$
　　if $b_{f_i} = \sum_j b(X)_j < M(X)$ **then**
　　return *result* = *null*
　　end if
　　end while
　　根据目标函数式(12),求出极值,得出分配节点集合 $result(V_n)$,记录此次式(8)结果 $VAR(EU)_n$
　　if $VAR(EU)_n \geqslant \eta VAR(EU)_{n-1}$ **then**
　　return $result(V_n)$
　　else
　　return *result* = *null*
　　end if
　end if
end if

4)路径选择事件使用最短路径 *POS* 算法。

5)如果分流事件增加流评价功能,则对流使用 FME 算法。

5　实验与分析

5.1　实验环境

目前实际的硬件环境还没有适合于混合网络的测试,网络运营商和互联网公司之间因接口和流量认证以及安全等情况下,较多数据节点会导致混合架构模型实际部署不确定性太大,只好通过 MATLAB 进行仿真模拟。本文拓扑和流量矩阵参照使用 Abilene 的数据集[27]。当流不能得到服务时,重发直到传输成功。除去数据集的基本流外,为了使实验更加接近突发状态,可以添加探针流。探针流分为大小流,数量各占 50%,时间段 t 内服从 M/M/1 分布。具体为大流带宽在[1 Mbit/s, 10 Mbit/s]区间指数分布,小流带宽在[1 Kbit/s, 10 Kbit/s]区间指数分布。探针流数量并发累加,累加公式为:

$$f_n=\begin{cases}2f_{n-1} & n>0\\ N_V & n=0\end{cases}$$

经验参数取值为:$\sigma=0.3$,$\eta=1.3$,$M^{\text{up}}=3$。

实验过程中,系统运行过程如下:

1)节点正常工作,按照控制器分配好的流表正常进行转发包工作。

2)判断节点 v_i 是否超过阈值。

3)节点状态超过阈值,触发该节点重新选择路径事件,并对 Bl_{v_i} 计数。

5.2　实验评价

5.2.1　网络任务完成时间

在相同负载和相同网络情况下,比较最短路径算法(ShortP)、POS 和 POS-FME 的任务完成时间,规定当 Abilene 的数据集中某个任务完成不了时,整个系统等待完成。实验结果如图 7 所示。

通过图可以看到,三个算法在任务数 10^7 以内性能相近。任务数达 10^7 附近以后三个算法开

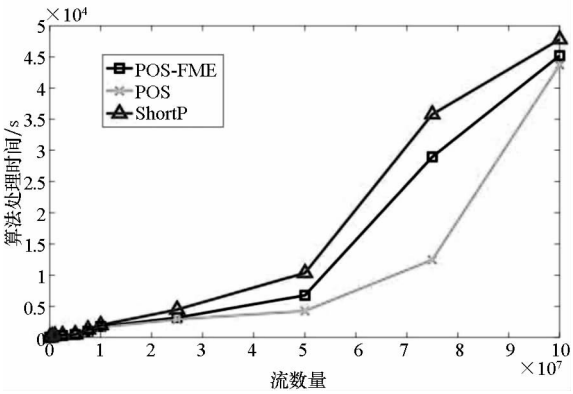


图 7 各算法的流处理时间比较

Fig. 7 Comparison of algorithms on flow processing time

始有显著区别,此刻系统大量节点开始运行在阈值附近,需要各个算法进行流控,来避免超过阈值。

在任务数为 7.5×10^7 处,POS-FME 相对于 ShortP 和 POS 能够达到最好性能,此时任务数较多,达到大部分节点处理阈值,POS-FME 加入流的判定,预测了一些流的行为,为大流预提供了带宽,不需要大量进行流路径变换;POS 保持大量节点和附近节点处理能力绝对差在一个范围,因此也具有保证流得到尽可能服务的预置带宽能力;而 ShortP 没有预测和预置带宽,所以性能较差。

任务数 1×10^8 处,系统达到处理阈值状态,大量流不能得到服务,流量矩阵中对应的各服务网络节点均在阈值附近。三个算法迅速收拢,性能相近。

5.2.2 任务完成时间比率

在相同负载和相同网络情况下,比较 ShortP、POS、POS-FME 算法的时间比率,可以看出各算法时间累加的效率。实验结果如图 8 所示。

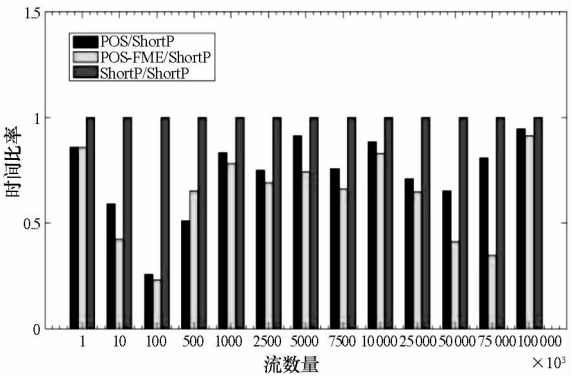


图 8 任务完成时间比率

Fig. 8 Task completion time ratio

设 ShortP 算法完成时间为 1,在算法开始阶段,流数目较少。除了在 5×10^5 流数目处,POS 算法比 POS-FME 算法计算量小外,在流数目 1×10^5 到 5×10^6 之间,POS-FME 算法优于 POS;特别在 7.5×10^7 处,因为预先对流进行预留服务带宽,POS-FME 性能更好。之后到达系统阈值,整体性能逐渐接近。

5.2.3 发生的分流事件次数

在相同负载和相同网络情况下,比较 ShortP、POS、POS-FME 算法的累加的分流事件触发次数。可选路径趋于饱和时,如果继续分配流,超过阈值就触发分流事件。触发事件后,ShortP 仅记录,不采取任何操作,POS、POS-FME 会触发算法重新分配路径。结果如图 9 所示。

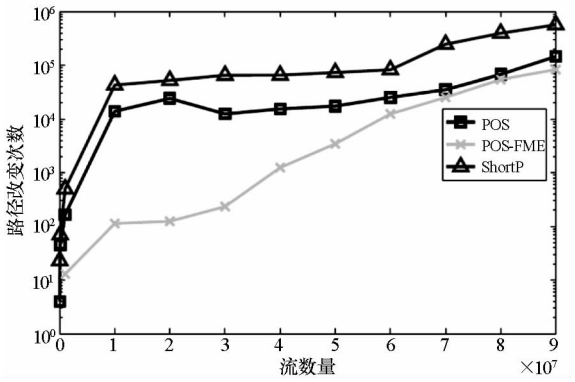


图 9 发生分流事件数目

Fig. 9 Number of flow diversion incidents

从实验结果可以看出,一般情况下,ShortP 分流事件最多,因为节点饱和时并不处理,以致超过阈值后每分配一次流都会触发分流事件。POS 具有一定处理能力,在流数量 3×10^7 处附近能够降低分流次数,那是因为采取次优路径重新规划了大流,增加了系统部分容量。POS-FME 每次算法都预估流量,使得在大部分时间段内分流事件很少,并同流数量具有一定的线性关系。当到达系统整体阈值后,三算法触发事件数目逐渐靠拢。

5.2.4 节点过载次数

在相同负载和相同网络情况下,比较 ShortP、POS、POS-FME 算法的累加的节点过载次数。当超过阈值时就触发节点过载数增加事件,流所经过节点可能会触发多个节点增加事件。触发节点增加事件后,ShortP 仅记录,不采取任何操作,POS、POS-FME 会触发算法重新分配路径。节点过载引起新流停止服务,直到节点上流被服务完毕,再轮到新流,这样增加流服务时间。流经过节点过载数目越多,服务质量越不稳定,累计节点过载数目越多,该流被拒绝服务造成重传的可能性越高。实验结果如图 10 所示。

从实验结果可以看出,一般情况下,ShortP 节

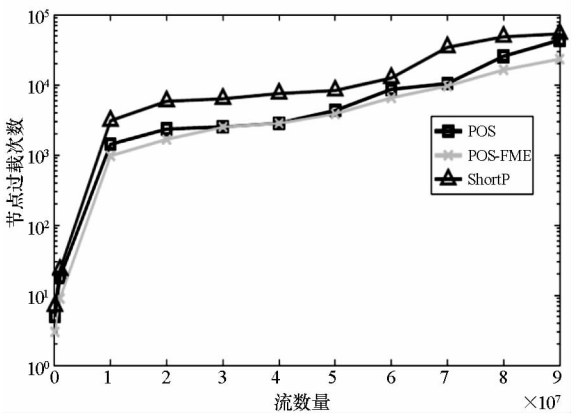


图 10　节点过载情况

Fig. 10　Number of overloaded nodes

点过载数目最多,因为节点饱和时并不处理,以致超过阈值后每分配一次流都会触发节点过载事件。POS 和 POS-FME 性能相近,部分情况 POS-FME 优于 POS,在流数量 3×10^7 附近几乎相同,说明尽管 POS 和 POS-FME 处理流的方式不同,但是已经到达系统能够处理的最大能力。当到达系统整体阈值后,POS 和 ShortP 迅速接近,此时各个节点都处于饱和状态,每次新流都会触发过载事件。

6　结论

混合网络既能够获得可编程网络的便捷处理、高扩展性、低维护成本等优势,也可以继续保持之前的普通网络投资和技术积累。目前混合网络都是作为控制网络出现的,面临的基本都是大型网络、网络流量集合情况,网络延迟往往较小,需要尽量少地更改流的位置。混合网络是多个异构网络组成,有关键节点模式和平行网络模式两种拓扑结构。在关键节点模式中,系统快处于饱和运行情况下,一些寿命短、容量小的流被无状态资源调度算法分配到关键路径上,造成整个路径上的流重新分配,出现流闪现问题。解决方式是将这些小流分配到其他位置,或者将一些可以被移出来的大流分配到其他次优路径。非预测的流问题是指在关键节点模式中,系统快处于饱和运行情况下,多条寿命长、容量大的流被无状态资源调度算法分配到关键路径上,造成整个路径上的流重新分配。解决方式是将这些大流进行评估,根据系统的运行状态和流的分布规律,部分分配次优路径。路径堵塞或连接震荡场景问题是指在平行网络模式中,传输网络不稳定,造成链路权值发生抖动变化,流分配路径随之抖动变化。解决方式是将这些边缘路径进行评估,同时评估流的可承受转移代价,根据系统的运行状态和流的分布规律,部分流分配次优路径。如果忽略以上三个问题,控制网络采取原有资源调度算法,则会出现频繁调度网络资源、网络震荡等情况。本文提出 POS 资源规划算法和基于流量评价的资源规划算法 POS-FME,在混合网络环境中,不降低延迟情况下,减少了流重新分配数目、提高了网络利用率,避免了热点链路拥塞问题。在仿真实验中,验证了 POS 和 POS-FME 算法的性能,在大部分情况下 POS 和 POS-FME 算法表现都比传统算法好,在负载快接近系统极限情况下,POS 和 POS-FME 算法相比传统算法性能有显著提高。

参考文献(References)

[1] Fortz B, Thorup M. Optimizing OSPF/IS-IS weights in a changing world [J]. IEEE Journal on Selected Areas in Communications, 2002, 20(4): 756 – 767.

[2] Fortz B, Thorup M. Robust optimization of OSPF/IS-IS weights[C]//Proceedings of INOC, 2003: 225 – 230.

[3] Applegate D, Cohen E. Making intra-domain routing robust to changing and uncertain traffic demands[C]/Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2003: 313 – 324.

[4] Elwalid A, Jin C, Low S H, et al. MATE: MPLS adaptive traffic engineering [C]//Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, 200: 1300 – 1309.

[5] Mitra D, Ramakrishnan K G. A case study of multiservice, multipriority traffic engineering design for data networks[C]// Proceedings of Global Telecommunications Conference, 1999: 1077 – 1083.

[6] Handigol N, Seetharaman S, Flajslik M, et al. Plug-n-serve: load-balancing web traffic using OpenFlow [J]. ACM SIGCOMM Demo, 2009, 4(5): 6.

[7] Wang R, Butnariu D, Rexford J. OpenFlow-based server load balancing gone wild[C]//Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, 2011: 12.

[8] Macapuna C A B, Rothenberg C E, Maurício F M. In-packet Bloom filter based data center networking with distributed OpenFlow controllers [C]//Proceedings of Globecom Workshops, 2010: 584 – 588.

[9] Mark B, Chase J S, Landweber L, et al. GENI: a federated testbed for innovative network experiments [J]. Computer Networks, 2014: 5 – 23.

[10] 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.

[11] Schneider F, Egawa T, Schaller S, et al. Standardizations of SDN and its practical implementation [J]. NEC Technical Journal, 2014, 8(2): 16 – 20.

[12] Hammad A, Aguado A, Peng S, et al. On-demand virtual infrastructure composition over multi-domain and multi-technology networks [C]//Proceedings of Optical Fiber Communication Conference, 2016.

[13] Handigol N, Seetharaman S, Flajslik M, et al. Aster* x: load-balancing web traffic over wide-area networks[C]// Proceedings of GENI Engineering Conference, 2011.

[14] Koerner M, Kao O. Multiple service load-balancing with OpenFlow[C]//Proceedings of 13th International Conference on High Performance Switching and Routing, 2012: 210 - 214.

[15] Vissicchio S, Vanbever L, Bonaventure O. Opportunities and research challenges of hybrid software defined networks[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(2): 70 - 75.

[16] Guo Y Y, Wang Z L, Yin X, et al. Traffic engineering in SDN/OSPF hybrid network [C]//Proceedings of 22nd International Conference on Network Protocols, 2014: 563 - 568.

[17] Jiang J W, Lan T, Ha S, et al. Joint VM placement and routing for data center traffic engineering[C]//Proceedings of IEEE INFOCOM, 2012: 2876 - 2880.

[18] Zhang Q, Cheng L, Boutaba R. Cloud computing: state-of-the-art and research challenges [J]. Journal of Internet Services and Applications, 2010, 1(1): 7 - 18.

[19] 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.

[20] Jain S, Kumar A, Mandal S, et al. B4: experience with a globally-deployed software defined wan[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 3 - 14.

[21] Hong C Y, Kandula S, Mahajan R, et al. Achieving high utilization with software-driven wan [J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 15 - 26.

[22] Sherry J, Hasan S, Scott C, et al. Making middleboxes someone else's problem: network processing as a cloud service[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 13 - 24.

[23] Qazi Z A, Tu C C, Chiang L, et al. SIMPLE-fying middlebox policy enforcement using SDN[J]. Computer Communication Review, 2013, 43(4): 27 - 38.

[24] Ling L, Xiaozhen M, Yulan H. CDN cloud: a novel scheme for combining CDN and cloud computing[C]//Proceedings of 2nd International Conference on Measurement, Information and Control, 2013: 687 - 690.

[25] Rückert J, Blendin J, Hausheer D. Software defined multicast for over-the-top and overlay-based live streaming in ISP networks[J]. Journal of Network & Systems Management, 2015, 23(2): 280 - 308.

[26] Wichtlhuber M, Reinecke R, Hausheer D. An SDN based CDN/ISP collaboration architecture for managing high-volume flows[J]. IEEE Transactions on Network and Service Management, 2015, 12(1): 48 - 60.

[27] Zhang Y. Yin Zhang's Abilene TM[EB/OL]. [2017 - 08 - 27]. http://www.cs.utexas.edu/~yzhang/research/AbileneTM/.