DraLCD:一种新的数据中心流量工程方法

杨 洋[1,2],杨家海[1],秦董洪[3],王于丁[1],凌 晓[1]
(1.清华大学网络科学与网络空间研究院,北京 100084;2.西安通信学院,陕西西安 710106;
3.广西民族大学信息科学与工程学院,广西南宁 530006)

摘 要: 流量均衡是为了避免网络拥塞而作为流量工程中的路由优化目标提出来的,由于数据中心网络的流量特性,使得传统IP网络的流量工程方法不一定适合.为此,本文在SDN(Software Defined Network)的框架下,提出了一种基于链路关键度的自适应负载均衡流量工程方法:DraLCD(Dynamic Routing Algorithm based on Link Critical Degree).该方法通过对全局视图的网络管控,并充分利用了网络中存在的冗余路径,在完成细粒度流量均衡的同时,能够降低控制器的计算开销以及与交换机之间的通信开销,最终完成路由优化的目标.最后,基于DraLCD设计的原型系统,通过在Mininet仿真平台中部署并进行仿真实验,与现有的等开销多路径路由算法ECMP(Equal-Cost Multi-Path)以及GFF(Global First Fit)路由算法相比较,能够明显地提升网络性能.

关键词: 流量均衡;软件定义网络(SDN);关键链路;多路径路由
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2017)05-1261-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.05.032

DraLCD:Another Traffic Engineering Method for Data Center Networks

YANG Yang[1,2], YANG Jia-hai[1], QIN Dong-hong[3], WANG Yu-ding[1], LING Xiao[1]
(1. *Institute for the Network Sciences and Cyberspace, Tsinghua University, Beijing* 100084, *China*;
2. *Xi'an Communication Institute, Xi'an, Shaanxi* 710106, *China*;
3. *School of Information Science and Engineering, Guangxi University for Nationalities, Nanning, Guangxi* 530006, *China*)

Abstract: Traffic balancing is proposed as a routing optimal goal in traffic engineering in order to avoid the network congestion. Because of data center traffic characteristics, traffic engineering of traditional IP networks may be not suitable to the data centers. Thus, we present a dynamic routing algorithm: DralCD (Dynamic Routing Algorithm based on Link Critical Degree), which is based on the link critical degree within SDN (Software Defined Network) framework. By the global visibility and making full use of the redundant paths in the network, DraLCD can realize the fine-grained traffic balancing while reducing the computing cost of controller and communication overhead between controller and switches, and finally achieves the goal of routing optimal. Furthermore, we design and perform the prototype of DraLCD on Mininet platform, and compare it with two other popular algorithms: ECMP (Equal-Cost Multi-Path) and GFF (Global First Fit). In our evaluations, the results show that DraLCD can significantly improve the network performance compared with the other two algorithms.

Key words: traffic balancing; software defined network (SDN); critical link; multipath routing

1 引言

数据中心网络的流量工程大致经历了四个发展阶段,第一阶段采用传统网络的流量工程方法[1],由于数据中心本身的拓扑结构以及流量特性[2],实施效果并不理想.例如,生成树协议STP(Spanning Tree Protocol)及其扩展协议MSTP[3]为了避免路由环路的产生,没有利用数据中心的冗余链路.在网络处于高负载的状态下,将会增加链路拥塞和数据包丢失的概率.其次,数据中心流量具有高动态、突发的特性,流量矩阵难以建立.因此,传统网络基于MPLS的流量工程方法也并不适用.第二阶段,为了充分利用数据中心存在的冗余链路

收稿日期:2015-03-19;修回日期:2016-08-25;责任编辑:蓝红杰
基金项目:国家973重点基础研究规划(No.2012CB315806);国家863高技术研究发展计划(No.2015AA015601);国家自然科学基金重点项目(No.61432009,No.61462009);教育部博士学科点专项基金(No.20130002110058);广西自然科学基金(No.2014GXNSFAA118358)

而普遍采用 ECMP 静态路由机制. 例如,VL2[4] 利用 ECMP 选取核心交换机,将流量均匀分配到所有可用路径上来应对流量的动态特性. 但是当链路出现大流而长时间占用链路时,容易造成链路的拥塞. 第三阶段引入了链路状态监测机制,属于动态的路由策略. 例如,DARD[5] 部署 MPTCP[6] 协议进行多路径的流量均衡,并通过主动探测方式来检测链路的拥塞情况; CONGA[7] 则在数据包头增加了传输链路拥塞信息位,但是这些方案都增大了相应的开销.

由于以上 3 个阶段都属于分布式机制缺乏对全局信息的管控,只能做到局部最优. 基于 SDN 的集中控制方式[8~13] 则通过对全局视图的网络管控,能够将网络性能提升 15% 至 20%,接近最优的解决方案[8],使得 SDN 成为未来数据中心网络流量工程发展的一种趋势[9]. 但是当前面临的主要挑战:一是如何平衡降低控制平面开销与细粒度流量均衡之间的关系,例如,Hedera[10] 和 Mahout[11] 为了降低控制器的计算开销,只针对大流事件,但是对于数据中心小流占据 80%[2] 的数量来说,流量均衡并不精确. DevoFlow[12] 则通过下发控制权到交换机上来降低控制器开销,但是这样的方案又违背了 SDN 转发与控制分离的设计原则;二是如何简化部署、增强可扩展性,现阶段的部分研究工作为了进行细粒度的流量均衡,需要对现有的通信协议进行改动,例如,Fastpass[13] 除了设计新的通信协议外,还对数据源端的传输控制协议进行了修改,同时还需要增大接收端的缓存来解决数据包乱序问题,这些都增加了部署的复杂性,可扩展性不高.

综上所述,本文将基于 SDN 的设计理念,通过重新定义关键链路,并基于链路关键度提出了一种新的动态路由算法(DraLCD, Dynamic Routing Algorithm based on Link Critical Degree). 文章的主要贡献点在于:

(1) DraLCD 由离线预计算和在线计算两个阶段完成,目的是减轻控制器计算开销以及控制器与交换机之间的通信开销. 同时,算法并不需要对链路进行主动探测从而降低链路的额外开销.

(2) DraLCD 集中控制的方式能最大限度地减低拥塞链路产生的风险,能够针对网络瞬时的性能下降及时作出调整,并且不会对目前普遍使用的通信协议做任何改动.

(3) 基于 DraLCD 设计了原型系统. 通过在 Mininet 仿真平台中部署,并与 ECMP 以及 GFF(Global First Fit)启发式路由算法相比较,实验结果展示了 DraLCD 的优越性能.

2 路由算法设计

2.1 关键链路描述

在图 1 所示的 FatTree 简单网络拓扑中,可以看到两种情况可以产生拥塞,一种是从汇聚层到核心层的上行链路(0,0)产生拥塞,另一种是从核心层到汇聚层的下行链路(2,3)产生拥塞. 那么产生拥塞的上行链路(0,0)和下行链路(2,3)就成为关键链路,即容易发生拥塞的链路. 从图中可以看到在产生拥塞链路的同时,有同样能到达目的端的空闲链路存在,例如,对于拥塞的上行链路(0,0),可以将流量分流到空闲的链路(0,1)上. 为了实现该目标,需要对关键链路进行量化,即通过定义链路关键度来实施.

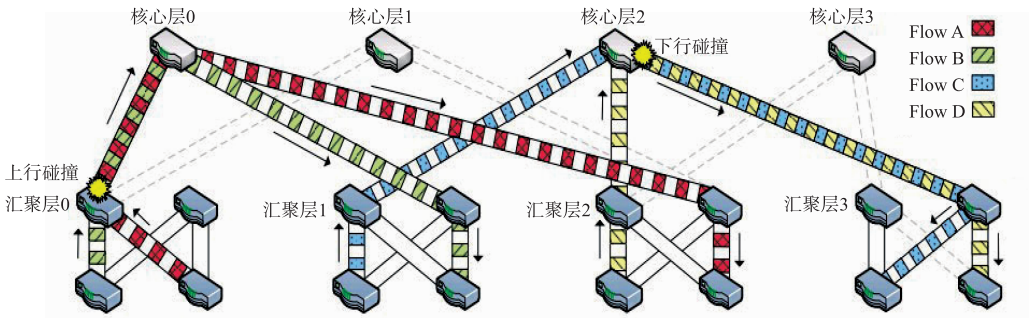

图 1 FatTree 简单网络拓扑

2.2 链路关键度定义

对于图 1 所示的数据中心网络,可以用无向连通图 $G=(V,E)$ 进行描述,并据此定义链路关键度,如图 2 所示. 其中 V 代表网络中所有节点的集合,这些节点可以代表 ToR(Top of Rack)架顶交换机或者网络中间节点交换机,E 代表所有节点之间的链路集合,P 表示节点对的集合,图中每个节点都有唯一的标识. s 和 d 表示图 G 中的任意两点,其中 $s,d\in V,(s,d)\in P$.

对于给定的网络拓扑,定义链路 l 的平均期望负载 $AVE(l)$. 期望值越高的链路,链路质量越好,数据源端选择这条链路的概率值就大. 将 $AVE(l)$ 定义为所有节点对 (s,d) 之间通过链路 l 的流量之和与通过链路 l 的最大流路径数目 m 的比值. 用公式表示如下:

$$AVE(l)=\sum_{(s,d\in p)} f_l(s,d)/m \tag{1}$$

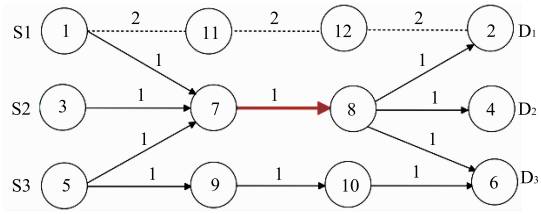

图2　简单网络拓扑模型

式中 $f_l(s,d)$ 表示节点对 (s,d) 之间通过链路 l 的流量. 根据链路的平均期望负载,确定链路的关键度. 链路的关键度定义为链路的平均期望负载与链路的容量比值. 公式如下:

$$\rho(l)=AVE(l)/C_l \tag{2}$$

式中 $\rho(l)$ 表示链路 l 的关键度,C_l 表示链路 l 的容量. 可以看出,链路 l 的关键度的值越大,表示该链路越关键,相应地链路就容易造成堵塞. 以图 2 为例,假设拓扑支持全等分带宽,且每条链路带宽均为 100 Mb,传输路径的链路代价均为 1,数据源端 S1、S2、S3 都以 30 Mb/s 速率向目的端发送数据包,S3 存在等价链路,在 S3 节点设备采用 ECMP 路由机制,通过计算,节点对(S3,D3)之间所有链路中,链路 7-8 的关键度值最高 $\rho(l_{7-8})\approx 0.25$,其余链路关键度值为 0.15,那么链路 7-8 即为关键链路,也是最容易产生拥塞的链路.

2.3　核心优化问题

为了降低控制器计算开销,算法设计了离线预计算和在线计算两个阶段. 离线阶段主要完成链路关键度的计算. 目前的路由协议中,通常定义链路容量或可用带宽的倒数作为链路的边权值. 例如,某条链路当前剩余带宽较大,那么该链路权值相对较小,容易吸引流量. 如果这是一条关键度值很高的链路,根据 2.2 定义的 AVE 物理意义,那么在下一刻,这条链路有可能产生拥塞,给它分配过多的流量并不一定合适. 但是如果将这条链路的关键度值作为剩余带宽倒数的修正因子,权值就不一定小了. 所以,为了更好的刻画链路的实时状态,本文将离线阶段得到的链路关键度值与该链路可用带宽相比,其值能很好的反映出链路的动态特性,提高算法的准确性. 在线计算阶段,可以获取链路的可用带宽 $C_l^{'}$,具体方法将在第 3 章介绍. 根据等式(1),(2)以及链路的可用带宽,定义链路的权值为链路的关键度与链路的可用带宽的比值. 公式如下:

$$\mathrm{cost}(t)=\frac{\rho(l)}{C_l^{'}} \tag{3}$$

式中 $\mathrm{cost}(t)$ 就表示链路 l 的权值. 公式(3)说明那些高关键度、低可用带宽的链路就是要进行拥塞避免的链路. 因此,本文提出优化的目标函数为:

$$f(r)=\sum_i\sum_j r_j^i\sum_l A_{lj}^i\cdot\frac{\rho(l)}{C_l^{'}} \tag{4}$$

式中 r_j^i 就是流量工程需要求解优化问题的目标值. A_{lj}^i 代表路由矩阵,其中 i 代表节点对,表示节点对 i 选择的路径 j 中某条链路 l. 对于优化问题的约束条件,首先是链路的负载不能超过链路的承载能力,即 $\sum_i A_{lj}^i\cdot r_j^i\leqslant C_l$;其次确保分配在节点对 i 之间有效路径上的流量总和等于该节点对源端节点的输出流量,即 $\sum_j r_j^i=R_i$;最后是链路流量分配的非负取值约束,即 $r_j^i\geqslant 0$. 最终的优化问题为:

$$\begin{aligned}&\text{Minimize}\sum_i\sum_j r_j^i\sum_l A_{lj}^i\cdot\frac{\rho(l)}{C_l^{'}},\\&\text{s.t.}\sum_i A_{lj}^i\cdot r_j^i\leqslant C_l,\forall l,\\&\sum_j r_j^i=R_i,\forall i,\\&r_j^i\geqslant 0\end{aligned} \tag{5}$$

通过求解该优化问题,得到数据源端在有效多路径上的最佳流量分配值.

3　原型系统设计

3.1　路由模块

本文基于 DraLCD 实现的原型系统主要由 3 个部分构成:路由模块、测量模块以及控制器模块,执行数据转发任务的底层交换机网络均支持 OpenFlow 协议,原型系统整体设计如图 3 所示. 其中,路由模块属于 SDN 架构中应用层的范畴. 文章 2.3 提出了路由算法的核心优化问题,通过求解目标函数可以得到节点对之间有效路径上的最佳流量分配值. 同时根据网络实际运行情况对链路负载进行合理地替换,最终转换成一个实际的算法,即基于链路关键度的动态路由算法 DraLCD,如算法 1 所示. 路由模块运行 DraLCD,并作为 SDN 架构中应用层的一个可执行程序,通过控制器调用计算并最终形成路由转发策略下发到网络节点设备.

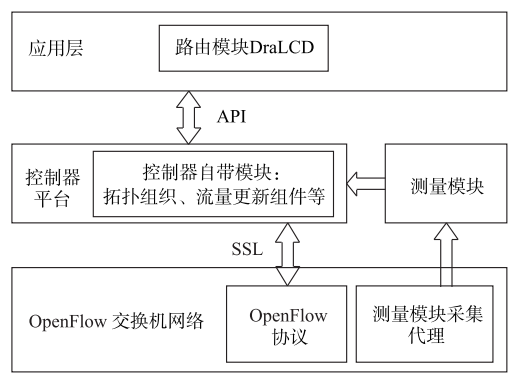

图3　原型系统

算法 1 中节点对之间传输路径数量 n 可以预先设定,算法的时间复杂度是 $O(n)$,因此算法执行效率较高. 算法 1 由在线计算和离线预计算两个阶段完成,其中在线阶段是指通过测量模块获取链路状态信息从而对相应的参数进行动态的调整(第 11 – 13 行),而对链路关键度的定义是在离线预计算阶段完成(第 16 行),目的是减轻控制器的计算开销. ε_β、ε_δ、ε_r 是求解优化问题中设定的步长参数,分别代表链路代价步长、节点对流量需求代价步长以及路径 j 上分配速率迭代步长,参数之间相互独立,在仿真实验中将对步长值的选取作进一步的讨论.

算法 1 DraLCD

1. #initialization
2. input:ε_β、ε_δ、ε_r//设定步长参数
3. input: r_j^i//获取数据流初始速率
4. input:n //获取节点对传输路径数量
5. input:C_l //获取传输链路容量
6. #begin
7. T //设定抽样周期值
8. B_T //调用链路传输字节值
9. dralcd_flag //算法执行标志位
10. while dralcd_flag = = true:
11. for $j=1$ to n:
12. $\sum_i \boldsymbol{A}_{lj}^i \cdot r_j^i = B_T/T$
13. end for
14. $\beta_l(t+T)=[\beta_l(t)-\varepsilon_\beta(C_l-\sum_i \boldsymbol{A}_{lj}^i \cdot r_j^i)]^+$//更新链路代价
15. $\delta_i(t+T)=[\delta_i(t)-\varepsilon_\beta(\sum_j r_j^i - R_i)]^+$//计算流量需求代价
16. $r_j^i(t+T)=r_j^i(t)+\varepsilon_r\{\delta_i-\sum_i \boldsymbol{A}_{lj}^i(\rho_l/C_l'+\beta_l)\}$//获取传输路径流量分配值
17. end while
18. #end

3.2 测量模块

根据 DraLCD 路由算法,控制器需要获取节点对之间流量传输的信息以及链路状态信息. SDN 架构中,采用 OpenFlow 协议可以支持两种方式对数据流信息进行统计:一种是控制器对交换机抽样的方式,即控制器向交换机发出 ReadState 消息获取数据流的计数器,目前很多采用 OpenFlow 协议的交换机都支持控制器对其进行抽样统计;另一种是交换机对控制器进行数据流信息的推送,即当数据流生存时间到期将向控制器推送 FlowRemoved 消息,该消息包含数据流大小信息. 由于算法中链路负载 $\sum_i \boldsymbol{A}_{lj}^i \cdot r_j^i$ 可以用 B_T/T 替换,本文将以上两种方式相结合,通过对交换机两次抽样,可以获取抽样间隔时间 T 内,流经链路的字节数 B_T,那么 B_T/T 可以代表该抽样时刻的链路负载,与链路容量求差后,即可获取路由模块计算所需要的链路剩余带宽. 另外,测量模块对节点对之间传输流量信息的收集,采用文献[8]的方法,即通过在每个机柜内部署一台服务器,专门负责向控制器推送数据源端流量传输信息.

3.3 控制器模块

控制器是 SDN 架构中的核心组件,主要负责收集从测量模块获取的信息并调用路由模块进行计算,最终形成路由转发策略下发到网络节点交换机,交换机则根据流表进行转发. 架构中控制器主要完成以下任务:(1)收集测量信息并形成全局视图,提供给路由模块进行计算;(2)由控制器自带的流表更新组件驱动 OpenFlow 交换机依据控制器下发的路由转发规则进行流表的安装、更新以及删除操作.

4 仿真与评估

4.1 仿真平台部署

4.1.1 仿真平台选择

随着云计算业务的兴起,数据中心网络中横向流量已经占到总流量的 80% 以上[14],并且横向流量更容易产生大流. 因此,本次实验模拟横向流量的场景,通过搭建 Mininet + Floodlight 仿真实验平台,部署基于 DraLCD 的原型系统. 仿真平台运行的宿主机是戴尔 OptiPlex9020,配置有一个 8 核、3.4GHz 主频的 64 位处理器,10G 内存,Ubuntu12.04 版本操作系统. 主机运行两台 VMware 虚拟主机:一台运行 Mininet V2.0 仿真平台,并基于 OVS(Open vSwitch)v1.11 搭建 OpenFlow 交换机网络,其中 Mininet V2.0 版本已经支持自定义网络拓扑;另一台运行 Floodlight V0.9 控制器以及支持 OpenFlow 协议的 wireshark 分析软件. 虚拟主机之间通过 eth1 网络接口连接.

4.1.2 拓扑设计

数据中心以服务器为核心的拓扑方案多采用 Fat-Tree 拓扑,依据拓扑构造规则[15,16],本文将构建一个规模为 8 个 POD(performance optimization datacenter)的简单拓扑,抽象成图 4 所示的实验拓扑图. 图中 S1、S2 代表两台接入层交换机,相当于 ToR 架构中的架顶交换机,分别属于两个不同的 POD. S1 接入 H1 到 H4 共 4 台主机,同样 S2 接入 H5 到 H8 共 4 台主机;C1 到 C16 代表拓扑中 16 台核心层交换机. 由于 FatTree 拓扑结构的特性以及链路带宽的统一规划,接入层交换机 S1 到 S2 之间总共有 16 条等价路径(为了拓扑的简洁,忽略掉 POD 中汇聚层交换机的图示),本次试验中将选取其中 3 条不相交路径用作流量均衡,并真实地模拟数据流竞争带宽的场景.

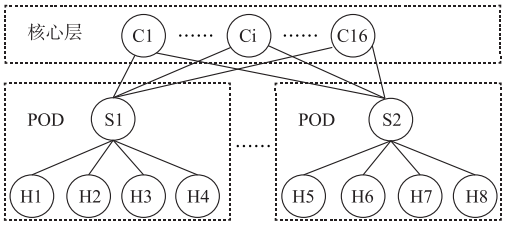

图4　实验拓扑

4.1.3　实验参数设置

实验拓扑中,每条链路的带宽被设定为 100Mbps,链路时延为 1ms,试验中节点对之间的多路径数量上限设置为 3 条,并且为不相交路径. 实验将模拟两个不同 POD 之间的数据传输,其中 H1 到 H4 为数据发送端,H5 到 H8 为数据接收端. 这 4 组节点主机对以第 1 组为目标测试组,后 3 组之间的数据传输作为背景流量,以此类推,并将 4 个节点对流量需求以 5Mb/s 的迭代步长从 10Mb/s 增加到 75Mb/s 来进行网络性能方面的测试. 为了更真实模拟实际场景,通过配置随机数产生器,使每个数据源端在 0 到 1 秒内由产生的随机数决定数据流开始传送的时刻,同时,加入 UDP 流量作为背景流量,模拟小流的场景. 控制器模块每隔 5s 进行一次路由策略计算. 针对每种路由策略,每个目标组进行 4 次实验,每个 POD 共进行 16 次实验以保证实验结果的准确性.

4.2　步长参数选取

DraLCD 涉及到 3 个步长参数的选择,如果步长值选取太大,最终解可能离最优值相差较多,而如果选取太小,则路由算法的收敛速度会变得非常缓慢. 以 ε_β 为例,假设一条链路带宽分别为 10Mb、100Mb 和 1000Mb,利用 MATLAB 仿真工具对步长值进行选取,其好处是能快速的遍历所设置的参数空间,并进行结果的比对. 结果发现链路带宽越小,步长值越大,可以得到 $\varepsilon_\beta \approx 1/C_1$,用同样的方法可以确定 $\varepsilon_\delta \approx 0.5/R_i$. 最后确定源端在路径 j 上分配速率迭代步长 ε_r,假设速率的初始值在[0,10]Mb 之间,并随机选取 10 个,选取的每一个迭代步长值的迭代次数都是 10 次计算平均的结果,最终得到 $\varepsilon_r \approx 10^{-5}$.

4.3　性能比较

4.3.1　实验对象选择

DraLCD 实验比较对象将选择实际网络中普遍支持部署的 ECMP 路由机制,以及 GFF 启发式路由算法. ECMP 路由机制相对于单路径路由,其优点是能极大提高网络吞吐量以及传输的可靠性,缺点是并不会因为链路状态的变化而改变流量分配的比例. GFF 路由算法是当算法触发时,为当前数据流线性的搜寻所有可能的有效路径,一旦发现合适的带宽链路即进行转发,并更新该链路状态,为下一次搜寻做准备.

4.3.2　时延比较

首先观察 DraLCD 平均传输时延的表现,实验结果如图 5 所示,其中横坐标表示数据源端的传输速率. 实验中,不同策略的应用导致各路径分配的流量各不相同,也就导致了时延上的差异. 实验一开始时延都有所上升,这与集中控制的架构有关,从图中观察到发送速率在 11Mb/s 时,时延迅速回落. 当发送速率在 11Mb/s 至 36Mb/s 之间时,三种路由算法时延表现相当,说明这一阶段网络链路状态良好,无拥塞链路出现. 当发送速率超过 37Mb/s 时,ECMP 时延开始增大;当发送速率大于 40Mb/s 时,ECMP 时延产生较大抖动,网络性能开始下降. 从图中可以看到在 ECMP 产生时延抖动阶段,DraLCD 最好,GFF 次之. 由于 ECMP 路由机制本身的设计,当传输路径上出现拥塞链路,算法并不能及时切换数据流到负载较轻的链路,从而使节点交换机缓存队列增加,数据包交付时延增大,出现丢包,网络性能下降. GFF 算法则是在输入路径集合中发现合适带宽链路立即引导数据流进行转发,相当于只是选取局部最优路径,相比之下,DraLCD 则能从整体上均衡多条链路之间的流量并将时延控制在有效范围内,总体平稳无抖动产生.

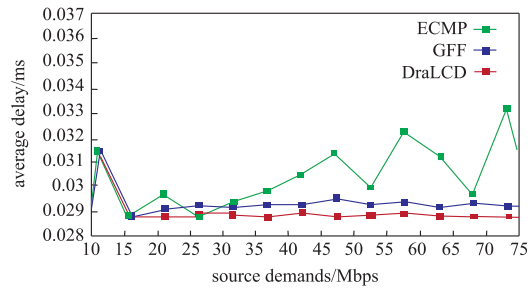

图5　时延比较

4.3.3　丢包率比较

DraLCD 丢包率的测试是在同一实验环境下进行的,如图 6 所示. 当数据源端发送速率低于 10Mb/s 时,三种策略均无丢包产生;当发送速率在 10Mb/s 与 41Mb/s 之间时,三种策略表现相当,但是当发送速率大于 41Mb/s 时,DraLCD 策略下的丢包率表现平稳,而 ECMP 和 GFF 部署下的丢包率均开始增加,在 61Mb/s 时刻,ECMP 的丢包率达到峰值. GFF 则在 47Mb/s 时,丢包率有所下降并保持平稳. 在上一节对时延的测试中,当发送速率大于 40Mb/s 时,ECMP 部署下的时延产生较大抖动,网络性能开始下降,其原因是随着负载增大,链路出现拥塞时,丢包率开始增加,当数据源端进入 TCP 拥塞避免时,丢包率又开始下降,如此反复是造成 ECMP 实验结果波动的原因,这也是 ECMP 机制所无法避免的. 这一现象与本次丢包率测试结果相一致,即

在网络时延增大的情况下,丢包率也相应增大.

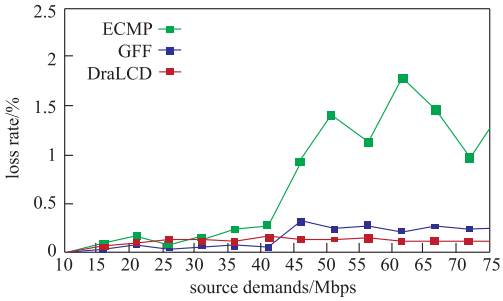

图6 丢包率比较

4.3.4 吞吐量比较

对网络吞吐量进行了比较测试,实验结果如图7所示. 实验一开始,三种策略部署下的吞吐量都增长迅速,从11Mb/s开始,ECMP吞吐量开始下降,整体波动较大;发送速率在11Mb/s与16Mb/s之间时,GFF与DraLCD表现相当,当源端速率大于17Mb/s时,GFF吞吐量开始下降,大于27Mb/s时,GFF吞吐量回升并趋于稳定,而DraLCD则整体表现较平稳. 实验结果表明,当节点交换机采用DraLCD路由策略进行部署时,与使用ECMP以及GFF部署时相比,网络吞吐量明显增加,进一步验证了DraLCD的优越性.

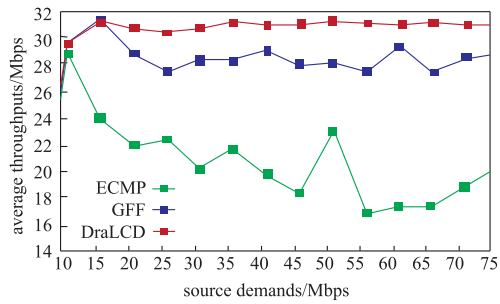

图7 吞吐量比较

5 结论

本文在SDN的框架下,提出了一种基于链路关键度的自适应负载均衡的流量工程方法DraLCD,并验证了DraLCD在提高网络性能方面的优越表现. 但同时也要注意到,在链路负载加重即将出现拥塞的情况下,DraLCD针对数据流的迁移尤其是大流,存在数据流分割的情况,从而有可能出现接收端数据包乱序的问题,这也是下一步工作中需要关注并改进的地方.

参考文献

[1] Awduche D,Chiu A,Elwalid A,et al. Overview and Principles of Internet Traffic Engineering [EB/OL]. https://www.rfc-editor.org/rfc/rfc3272.txt,2002-05/2015-03.

[2] Theophilus B,Aditya A,David A M. Network traffic characteristics of datacenter in the wild [A]. IMC 10:Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement [C]. New York:ACM,2010. 267 – 280.

[3] Ho T V,Yves D,Olivier B,et al. Traffic engineering for multiple spanning tree protocol in large data centers [A]. ITC '11:Proceedings of the 23rd International Teletraffic Congress [C]. San Francisco:IEEE,2011. 23 – 30.

[4] Albert G,James R H,Navendu J,et al. VL2:A scalable and flexible data center network [J]. Communications of the ACM,2011,54(3):95 – 104.

[5] Wu X,Yang X W. DARD:Distributed adaptive routing for datacenter networks [A]. ICDCS'12:Proceedings of 32nd IEEE International Conference on Distributed Computing Systems [C]. Piscataway:IEEE,2012. 32 – 41.

[6] Ford A,Raiciu C,Handley M,et al. Architectural Guidelines for Multipath TCP Development [EB/OL]. https://www.rfc-editor.org/rfc/rfc6182.txt,2011-03/2015-03.

[7] Mohammad A,Tom E,Sarang D,et al. CONGA:Distributed congestion aware load balancing for datacenters [A]. SIGCOMM '14:Proceedings of the 2014 ACM conference on SIGCOMM [C]. New York:ACM,2014. 503 – 514.

[8] Theophilus B,Ashok A,Aditya A,et al. MicroTE:Fine grained traffic engineering for data centers [A]. CoNEXT '11:Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies [C]. New York:ACM,2011. 8 – 8.

[9] Ian F A,Lee A,Wang P,et al. A roadmap for traffic engineering in SDN-OpenFlow networks [J]. Computer Networks,2014,71(1):1 – 30.

[10] Mohammad Al,Sivasankar R,Barath R,et al. Hedera:Dynamic flow scheduling for data-center networks [A]. NSDI10:Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation [C]. Berkeley:ACM,2010. 19 – 19.

[11] Andrew R,Wonho K,Praveen Y. Mahout:Low-overhead datacenter traffic management using end-host-based elephant detection [A]. INFOCOM'11 Proceedings IEEE INFOCOM 2011 [C]. Piscataway:IEEE,2011. 1629 – 1637.

[12] Andrew R,Jeffrey C,Jean T,et al. DevoFlow:Scaling flow management for high-performance networks [A]. SIGCOMM '11:Proceedings of the 2011 ACM Conference on SIGCOMM [C]. New York:ACM,2011. 254 – 265.

[13] Jonathan P,Amy O,Hari B,et al. Fastpass:A centralized zero-queue datacenter network [A]. SIGCOMM'14:Proceedings of the 2014 ACM Conference on SIGCOMM [C]. Chicago:ACM,2014. 307 – 318.

[14] Yukihiro N,Kazuki H,Chunghan Lee,et al. DomainFlow:

Practical flow management method using multiple flow tables in commodity switches [A]. CoNEXT '13: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments And Technologies [C]. New York: ACM, 2013. 399 – 404.

[15] Siddhartha S, David S, Sunghwan I, et al. Scalable optimal flow routing in datacenter via local link balancing [A]. CoNEXT '13: Proceedings of the ninth ACM Conference on Emerging Networking Experiments and Technologies [C]. New York: ACM, 2014. 151 – 162.

[16] Keqiang H, Eric R, Kanak A, et al. Presto: Edge-based load balancing for fast datacenter networks [A]. SIGCOMM'15: Proceedings of the 2015 ACM Conference on SIGCOMM [C]. New York: ACM, 2015. 465 – 478.

作者简介

杨　洋　男,1980 年出生,江苏无锡人,清华大学博士生,主要研究方向为计算机网络、路由协议、流量工程、SDN 等.
E-mail: y-yang-12@ mails. tsinghua. edu. cn

杨家海(通讯作者)　男,1966 年生于浙江云和,研究员,博士导师,清华大学网络科学与网络空间研究院党委副书记、网络运行与管理技术研究室主任、CCF 高级会员,主要研究方向为计算机网络、网络管理与测量、网络安全、SDN、云计算与大数据等.
E-mail: yang@ cernet. edu. cn