软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(8):2485−2500 [doi: 10.13328/j.cnki.jos.005543]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于时隙传输的数据中心路由算法设计*

杨 洋[1,2], 杨家海[1,3], 温皓森[4]

[1](清华大学 网络科学与网络空间研究院,北京 100084)

[2](国防科技大学 信息通信学院,陕西 西安 710106)

[3](清华信息科学与技术国家实验室(筹),北京 100084)

[4](Department of Computer Science, University of Rochester, New York 14627, USA)

通讯作者: 杨家海, E-mail: yang@cernet.edu.cn

摘 要: 基于软件定义网络(software defined network,简称 SDN)的数据中心流量工程,能够通过对全局视图的网络管控,动态选择路由路径,规避拥塞发生的风险.但是在制定路由策略时,经常会对数据流进行迁移,尤其是针对大流的迁移容易造成数据流丢包以及接收端数据包乱序的问题.提出了基于时隙的流片装箱算法(flowlet-binned algorithm based on timeslot,简称 FLAT),通过集中控制的方式获取链路状态信息并计算出合理的数据流传输时隙值,能够避免在数据流迁移过程中的丢包以及接收端数据包乱序问题;同时,在充分利用数据中心冗余链路的前提下,实现高效和细粒度的流量均衡.通过在 Mininet 仿真平台中部署并与 ECMP 以及 GFF 路由机制相比较,在链路高负载情况下,丢包率分别下降了 90%和 80%,而吞吐量分别能够提升 44%和 11%,实验结果展示了 FLAT 的优越性能.

关键词: 数据中心;软件定义网络;多路径路由;流量均衡;时隙

中图法分类号: TP393

中文引用格式: 杨洋,杨家海,温皓森.基于时隙传输的数据中心路由算法设计.软件学报,2018,29(8):2485−2500. http://www.jos.org.cn/1000-9825/5543.htm

英文引用格式: Yang Y, Yang JH, Wen HS. Routing algorithm design based on timeslot of transmission for data centers. Ruan Jian Xue Bao/Journal of Software, 2018,29(8):2485−2500 (in Chinese). http://www.jos.org.cn/1000-9825/5543.htm

Routing Algorithm Design Based on Timeslot of Transmission for Data Centers

YANG Yang[1,2], YANG Jia-Hai[1,3], WEN Hao-Sen[4]

[1](Institute for the Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China)

[2](College of Information and Communication, National University of Defense Technology, Xi'an 710106, China)

[3](Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing 100084, China)

[4](Department of Computer Science, University of Rochester, New York 14627, USA)

Abstract: Traffic engineering based on SDN (software defined network) can select routing paths dynamically in order to evade the risk of congestion through global view of network in data centers. However, the design of routing strategy often needs to change routing path during packet transmission, especially for elephant flows, which may commonly result in the problem of packet losses and out-of-order at receivers. To address the problem, an algorithm named "flowlet-binned algorithm based on timeslot (FLAT)" is proposed. FLAT is able to gather the information of link state and calculate the proper transmission timeslot under centralized control, which can solve the problem of packet losses and out-of-order. In the meantime, traffic balance with high efficiency and fine granularity can be achieved under

* 基金项目: 国家自然科学基金(61432009, 61462009); 国家重点研发计划(2016YFB0801302, 2017YFB0803004)

Foundation item: National Natural Science Foundation of China (61432009, 61462009); National Key Research and Development Program of China (2016YFB0801302, 2017YFB0803004)

收稿时间: 2015-10-09; 修改时间: 2017-01-10, 2017-04-11; 采用时间: 2017-10-26; jos 在线出版时间: 2018-02-08

CNKI 网络优先出版: 2018-02-08 11:55:32, http://kns.cnki.net/kcms/detail/11.2560.TP.20180208.1155.002.html

considerable use of the redundant links in data centers. Finally, simulation results show better performance of FLAT in Mininet platform compared with ECMP and GFF routing strategies with the packet loss rate respectively falling by 90% and 80%, and the throughput increasing by 44% and 11%, especially under the condition of high load of links.

Key words: data center; software defined network; multipath routing; traffic balance; timeslot

数据中心流量存在以用户访问为主的南北向流量和以数据备份、迁移为主的东西向流量.东西向流量主要在服务器之间产生,并且与南北向流量的比率达到了 4:1[1].服务器之间的数据备份、迁移容易产生大流,并造成链路之间流量分布的不均衡,从而产生拥塞,使网络性能急剧下降.由于数据中心通常具有对称的拓扑结构,使得节点对之间存在大量的冗余链路[2].充分利用这些冗余链路进行流量均衡,是目前数据中心流量工程的主要方法.当前,数据中心普遍采用 ECMP 这种静态的路由机制进行多路径传输,可改善单路径路由由于“选路”集中而造成的拥塞问题,同时能够做到故障链路的快速切换,增强可靠性,并能聚合链路带宽,充分使用网络资源.为了应对数据中心流量高突发的动态特性,VL2[3]在上行链路中增加了随机选择核心节点的步骤,但其选路本质上还是采用 ECMP 机制.这种静态路由策略存在的主要问题是:当链路中出现大流时,会造成链路资源分配不公平.

在采用多路径传输的基础上,动态的路由策略引入了链路状态监测机制,克服了 ECMP 路由的不足.例如,文献[4,5]就是通过多路径权重的优化来自适应路由.DARD[6]在数据源端通过传输层 MPTCP[7]协议进行多路径的流量均衡,源端通过主动探测的方式来检测链路的拥塞情况.CONGA[8]扩展了数据包头字段,增加了拥塞信息位来携带传输链路的拥塞信息,并沿着传输路径逐跳更新,数据包到达接收端后,通过反馈回路将拥塞信息反馈到数据源端.以上的研究工作都是基于传统网络的分布式转发,存在共性的问题是:由于缺乏对全局的信息掌握,使得流量均衡不能做到全局最优.例如,VL2 对核心交换机的随机选取都是在上行链路之间进行,并未关注下行链路的拥塞状况.

基于 SDN 的集中控制方式通过对全局视图的网络管控,能克服分布式转发的不足.基于 SDN 的流量工程具有以下优势:(1) 通过对全局信息的网络管控,针对网络资源的瓶颈处,通过下发策略到节点设备动态地调整网络状态;(2) 能够做到应用层信息全局的快速部署,例如 QoS、路由策略等,不需要对每个节点交换设备进行配置; (3) 具有可编程性,能够对处于数据平面支持 OpenFlow(OF)协议的交换机预编程以及根据控制器下发策略动态地重新编程;(4) OF 交换机支持的多流表流水线使得对数据流的管理更加灵活和有效.这些优势使得 SDN 成为流量工程未来的一种发展趋势[9,10],并且相对于传统的分布式转发网络,能够更精确地进行流量均衡.

数据中心归属单一运营商所有,便于流量工程的统一部署,天然符合 SDN 所需要的集中控制要求.文献[11]基于 SDN 的设计理念,首次将 NOX 集中控制器应用到数据中心.MicroTE[12]通过在终端部署流量监测服务器对传输的流量进行预测,当预测到的流量超过设定的阈值则触发控制器,通过路由策略的更新进行细粒度的流量均衡;否则,依然采用 ECMP 路由机制进行转发.Hedera[13]和 Mahout[14]提出了大流碰撞时的解决方案,两者的相同点在于都是针对大流进行流量均衡,而对小流的处理采用 ECMP 机制进行路由;二者的主要区别在于监测大流的位置不同,其中,Hedera 通过控制器每隔 5s 对边缘层交换机进行抽样来监测大流;Mahout 则是通过在终端操作系统中嵌入“夹层”来监测大流,并以带内信号的方式通知控制器.当监测到大流后,触发控制器数据流调度算法计算出合适的分流路径并下发到交换机.

Fastpass[15]是基于数据包粒度的流量均衡,在控制器与主机之间,通过设计新的通信协议“FCP”对数据包进行时隙以及路径的分配,做到交换机“零缓存”来降低拥塞的发生.Fastpass 在数据源端关闭了传统的 TCP 拥塞控制协议,即不再使用 TCP 拥塞控制窗口来对发送速率进行限制,通过这种方式提高网络吞吐量.Fastpass 扩大了接收端的缓存来克服由于采用数据包粒度的流量均衡方式而产生接收端数据包乱序的问题.

数据中心采用集中控制方式的流量均衡,通过对全局视图的网络管控,克服了传统网络分布式转发只能做到局部最优的缺点.当前面临的挑战:

- 一是如何设计数据流调度策略,既能简化部署、增强可扩展性,又能解决在数据流迁移的过程中造成的数据流丢包以及接收端可能产生的数据包乱序问题.目前的研究工作中,基于数据流粒度的均衡策略

不可避免地会在大流的迁移过程中产生丢包,并且可能造成接收端数据包乱序,例如 Hedera 的 GFF 路由策略.基于包粒度的均衡策略为了克服接收端数据包乱序问题,实现起来较为复杂,对传输协议或者体系结构改动较多.例如,Fastpass 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,这些都增加了部署的复杂性,可扩展性不高.

- 二是如何平衡控制平面开销与细粒度流量均衡之间的关系.例如,Hedera 和 Mahout 当监测到大流并触发算法时,需要为目标大流搜寻所有的有效传输路径,寻找具有合适转发带宽的路径,计算复杂度至少是 $O(n)$.虽然能够实施细粒度的负载均衡,但同时也增大了相应的开销[16],并且策略的时效性不能保证.

综上所述,本文将基于 SDN 的设计理念,通过重新定义数据流的传输时隙,提出一种新的动态路由算法(flowlet-binned algorithm based on timeslot,简称 FLAT).本文的主要贡献点在于:(1) 基于 FLAT 算法实现的流量工程方案,以流量均衡为目标,在数据流调度的过程中不会造成接收端数据包乱序的问题;(2) 针对数据中心普遍部署的商用交换机浅缓存的特性,提出新的算法机制防止缓存溢出;(3) 在进行细粒度流量均衡的同时,不会对目前普遍使用的 TCP 通信协议做任何改动,易于部署,具有可扩展性.最后,通过原型系统的设计并在 Mininet 仿真平台中部署,在基于 FatTree 拓扑上与 ECMP 以及 Hedera 中 GFF 路由机制相比较,实验结果展示了 FLAT 能够适应网络瞬时的性能下降,减少丢包率,能够最大限度地减低拥塞链路产生的风险,同时证明了在数据中心基于该算法的流量工程解决方案能够降低控制平面开销,易于部署.

本文第 1 节对算法设计的关键技术进行分析.第 2 节实现 FLAT 算法并基于算法设计原型系统.第 3 节对 FLAT 实施过程中所产生的开销进行评估.第 4 节对 FLAT 进行仿真实验,得出性能评价结果.第 5 节对本文进行总结并展望下一步研究工作.

1 算法关键技术分析

本节将提出基于 SDN/OF 流量工程方案的应用层路由策略,即基于时隙的流片装箱算法 FLAT,通过利用数据中心存在的冗余链路,在实现细粒度流量均衡的同时,能够保证数据流迁移过程中不产生丢包以及避免接收端数据包乱序.

1.1 传输时隙定义

1.1.1 数据流分割

相对于单路径,多路径路由能避免因为失效链路或者拥塞链路造成的数据包丢失,更好地保障传输的可靠性.在 SDN/OF 框架下,可以根据链路拥塞状态将数据流在节点对之间有效的多条路径上动态地进行分配,使得网络中各链路的资源利用率趋于均衡.但是在数据流迁移的过程中,不可避免地会产生丢包以及接收端数据包可能出现的乱序问题,例如 Hedera.如果只从流量均衡的角度来说,基于数据包粒度的传输是完美的流量工程解决方案,但由于传输链路之间不同的时延差,导致接收端不可避免地出现数据包乱序.例如,Fastpass 和 DRB[17]都是在数据中心采用基于数据包粒度的流量均衡,同时,为了解决接收端数据包乱序的问题,需要对 TCP 协议进行修改并提出新的控制协议,同时还需要增大接收端缓存,部署复杂,可扩展性不高.

从以上分析可以看到,无论是基于数据流粒度还是数据包粒度的流量均衡方式,造成接收端数据包乱序的根本原因就是传输路径之间的时延差.如果相邻的两个数据包之间发送间隔大于传输路径之间的时延差值,接收端就不会产生数据包乱序的问题.对此,提出定理 1 并加以证明.

定理 1. 如果节点对之间存在多条可达路径,只要相邻数据包发送间隔大于或者等于这些路径的时延差,属于同一会话的数据包就可以任意选择不同的路径分别进行传输,并且不会出现接收端数据包乱序问题.

证明:假设数据源和目的节点对之间存在 n 条传输路径 L_i,其中,$i\in[1,n]$,并且传输链路具有相同的传输介质,属于同一会话的 2 个相邻数据包分别为 P_{j-1},P_j,且 P_{j-1} 序号小于 P_j 并分别沿路径 L_{i-1} 和 L_i 传输,数据包到达目的节点的传输时间分别 T_{j-1} 和 T_j.如果数据包到达目的节点的传输时间满足 $T_{j-1}\geqslant T_j$,则定理得证.实际测得路径 L_{i-1},L_i 当前传输时延分别为 D_{i-1},D_i(可以假设短时间内链路负载状态变化不大),且连接数据源端的节点交换设备转发 P_j 和 P_{j+1} 数据包的发送间隔时间为 δ,并且设 $\delta=|D_i-D_{i-1}|$,即数据包发送间隔时间用所选择路径的传输时

延差值来表示,则有 $T_{j-1}=D_{i-1}$,$T_j=\delta+D_i$.将多路径传输时间求差得到 $T_j-T_{j-1}=\delta+D_i-D_{i-1}$,此时存在两种情况:如果 $D_i-D_{i-1}\leqslant 0$,则 $T_j-T_{j-1}=0$,说明 2 个数据包同时到达;如果 $D_i-D_{i-1}>0$,则 $T_j-T_{j-1}>0$.因此当$\delta=|D_i-D_{i-1}|$时,必然存在 $T_{j-1}\geqslant T_j$,故定理得证. □

1.1.2 时隙定义

数据中心 90%以上的数据流是 TCP 流,根据 TCP 的流量与拥塞控制机制,发送端要受到发送窗口的制约在单位时间内发送 1 个或多个报文段,然后停止发送并等待接收端对已经发送的报文段进行确认.这段空闲时间段用 T_w 来表示.根据定理 1,当与δ值满足大于或者等于的关系时,就可以对属于同一数据流的报文段在不同的路径上进行迁移而不会产生接收端数据包乱序问题.因此,定义 T_w 作为数据流传输的一个单位时隙,如果对链路中不同的会话的数据流进行染色,如图 1 所示(图中各数据流由水平方向的黑色虚线隔开,表示沿着不同的路径传输),同时对产生会话的时间轴以 T_w 为一个时隙单位划分为 n 个传输时隙,如果能使相邻时隙内不出现属于同一色块的数据包,就能保证属于同一会话的数据包在不同的路径上传输而不会出现接收端数据包乱序的问题.

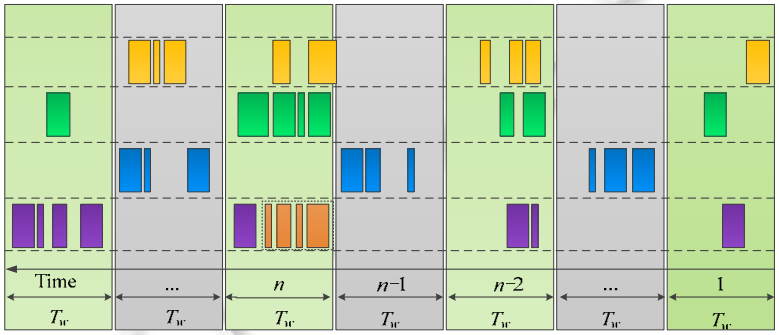

Fig.1 Definition of transmission timeslot

图 1 传输时隙定义

1.2 数学模型

根据定理 1,重新定义数据流的传输时隙,并且可以保证当数据流按照时隙分配在多条路径上传输时,不会引起接收端数据包出现乱序的问题.然而,数据流在实际传输过程中不一定按照定义的时隙到达,例如图 1 中,第 $n-1$ 个时隙内也会有属于同一会话数据流的黄色数据包出现,就需要对这一个时隙内的数据包进行缓存并在下一个时隙进行转发.图 1 中,第 n 个时隙底部的黄色数据包就是第 $n-1$ 个时隙内需要被缓存转发的数据包.为了实现 FLAT,需要对数据流的到达行为进行分析并建模.由于数据中心符合因为资源共享而导致大量信源叠加的事实,所以采用排队论是对这种网络流量特性进行建模分析的主要方法之一.根据排队论相关理论,通过测量与仿真,文献[18]推测,在链路带宽足够的情况下,TCP 流的到达行为服从泊松分布.此外,可以认为网络节点设备对数据流的转发,即对数据包进入链路的服务时间,服从指数分布,因而以往的研究工作中,比较常见的是采用 $M/M/1/\infty$排队模型进行建模[19].

当前,数据中心普遍使用浅缓存的商用以太网交换机,并且交换机通常采用共享缓存这种交换结构.在这种结构中,所有的输入和输出端口都共享一个缓存模块,所有需要经过交换机的数据都在缓存中存储转发,一台交换机就可以抽象成一个服务窗口.此外,可以认为交换机对数据流的转发,即对数据包进入链路的服务时间,服从指数分布.由于 FLAT 采用自定义的时隙传输数据,为了满足定理 1 可能需要对某个时隙内的数据包进行缓存转发,如果不对缓存队列进行优化处理,就容易造成交换机缓存溢出而产生丢包.因此,假设 t 时刻数据源端发现交换机缓存队长变小,则增大数据包进入系统的概率;反之则减小.数据流的到达率就不再是一个稳定值,而是依赖 t 时刻缓存队列长度 k 的一个函数,采用可变到达率的 $G/M/1/\infty$排队模型进行建模分析[20].用图 2 描述一个具有可变到达率的数据流生灭过程.

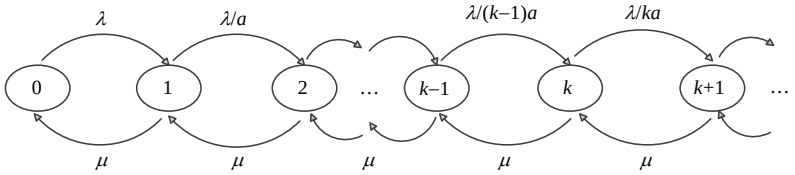

Fig.2 Variable arrival ratio of birth-death process

图 2 可变到达率生灭过程

以图 2 为例,其中假设缓存队列长度为 $k(k\geqslant 1)$,并且数据流是以 $a_k=1/a\times k(a\geqslant 1)$的概率进入排队系统,最终可以得到稳态下的数据流到达概率分布函数:

$$P_k=\frac{\rho^k}{(k-1)!\times a^{k-1}\times(1+\rho\times e^{\rho/a})} \tag{1}$$

其中,P_k 即为 t 时刻队列长度处于 K 状态的概率分布,ρ为数据流的到达强度.

1.3 优化问题

上一节提到数据中心普遍采用浅缓存的商用交换机,因此 FLAT 设计中需要有防止缓存溢出的算法机制.通常,交换机转发数据包的能力是固定不变的,即排队系统服务速率μ不变,通过上一节对数据流到达行为进行分析,采用可变到达率的排队模型,优化问题的目标就是最小化缓存队列的长度.根据 FLAT 算法的设计,数据流 i 在 t 时刻一个 T_w 时隙内的队列应该由 3 部分构成:上一时隙未处理的队列、本时隙内新到达的队列以及本时隙能处理的队列.用 $L_i(t-1)$表示上一时隙未处理的队列,$L_{in}(t)$表示当前时隙内新到达队列,$L_{io}(t)$表示当前时隙内能处理的最大队列.t 时刻,数据流 i 在一个时隙内需要缓存的队列长度为

$$L_i(t)=[L_i(t-1)+L_{in}(t)-L_{io}(t)]^+ \tag{2}$$

其中,当前时隙能处理的最大队列长度是由交换机转发链路带宽 C_l 以及传输时隙值 T_w 共同决定,即 $L_{io}(t)=C_l\times T_w$;表达式$[]^+$代表正值优化问题才有意义,优化问题的目标函数即为

$$\text{Minimize}\sum\nolimits_i L_i(t) \tag{3}$$

对公式(2)进行推导:t=1,即排队系统的起始时刻初始值 $L_i(t-1)=L_i(0)=0$,表示初始时刻时隙内并没有上一时隙缓存的队列,因此 $L_i(1)=L_i(0)+L_{in}(1)-L_{io}(1)=L_{in}(1)-L_{io}(1)$;当 t=2,$L_i(2)=L_i(1)+L_{in}(2)-L_{io}(2)=L_{in}(1)+L_{in}(2)-L_{io}(1)-L_{io}(2)$;以此类推,当 t 趋于无穷时可以得到:

$$\lim_{t\to\infty}L_i(t)=\sum\nolimits_1^{\infty}[L_{in}(t)-L_{io}(t)] \tag{4}$$

当一个排队系统运转一段时间后,系统的状态将独立于初始状态及经历的时间,这时称系统处于稳定状态.排队论中主要研究系统处于稳定状态下的工作情况,优化的目标函数即公式(4)可以变形为

$$F(K_i)=\sum_{K_i=L_{io}}^{\infty}(K_i-L_{io})\times P_{Ki} \tag{5}$$

其中,K_i 为当前队列长度,P_{ki} 即为公式(1),将公式(1)代入并对公式(5)进行级数求和整理后得到:

$$F(K_i)=\sum_{K_i=L_{io}}^{\infty}(K_i-L_{io})\times P_{Ki}=\sum_{K_i=L_{io}}^{\infty}(K_i-L_{io})\times\frac{\rho^k}{(k-1)!\times a^{k-1}\times(1+\rho\times e^{\rho/a})}=\frac{\rho\times e^{\rho/a}}{(1+\rho\times e^{\rho/a})}\times(\rho/a+1-L_{io}) \tag{6}$$

其中,$e^{\rho/a}$的取值范围在(0,2.7)之间,用常数符号 C_1 代替;表示排队强度的参数$\rho=\lambda/\mu$,其到达率λ可以用一个 $\overline{RTT}$ 内发送窗口值来替代,当发送端以速率 r_i 向链路注入数据流时,$\lambda=r_i\times\overline{RTT}$,由于交换机转发数据包的服务速率是固定的,因此设常数 $C_2=\overline{RTT}/\mu$;同时设 $C_3=\rho/a+1-L_{io}$,如果 $C_3<0$,则说明不需要对优化目标函数进行求解,即不需要对源端发送速率进行限速;对公式(6)进一步做变换后得到目标函数:

$$F(r_i)=\frac{C_1\times C_2^2\times r_i^2+a\times(1-C_l\times T_w)\times r_i}{a\times C_1\times C_2\times r_i+a} \tag{7}$$

优化问题总是伴随着约束条件,首先是链路实际负载不能超过链路自身承载能力,链路负载能力用 C_l 表示,即 $r_i^l \leqslant C_l$;其次是优化问题变量的非负取值约束,即 $r_i>0$.最终的优化问题为

$$\left.\begin{aligned}&\text{Minimize}\sum_i \frac{C_1\times C_2^2\times r_i^2+a\times(1-C_l\times T_w)\times r_i}{a\times C_1\times C_2\times r_i+a}\\&\text{s.t. } r_i^l \leqslant C_l\\&\quad r_i \geqslant 0\end{aligned}\right\} \tag{8}$$

公式(8)属于非线性比式和的分式规划,是一类全局优化问题,求解该类优化问题已被证明属于 NP-难问题[21],这一类优化问题可以采用分支定界法进行求解.最终提出数据源端控制算法 SRC(source rate control)对求解过程进行描述,见算法 1.

算法 1. SRC.

1. T /*设定控制器轮询周期值*/
2. B_T /*轮询周期内数据流传输字节值*/
3. $\overline{RTT}$ =*sampled_rtt*/*sampled_num*; /*计算平均时延 $\overline{RTT}$ */
4. $r_i^0=B_T/T$ /*数据流 *i* 初始速率*/
5. $C_3=\rho/a+1-L_{io}$
6. **if** $C_3<0$:
7. return null
8. **else**:
9. C_1=*random*(0,2.7) /*取值范围在(0,2.7)之间*/
10. $C_2=\overline{RTT}/\mu$
11. 求解优化问题(8)获取更新后的源端速率 r_i
12. **end if**

算法 1 中,*sampled_rtt* 表示时延抽样值,*sampled_num* 表示抽样次数.

2 算法实现与原型系统设计

本节将基于 SDN/OF 架构实现基于时隙的流片装箱算法 FLAT,并完成原型系统的设计.

2.1 FLAT算法实现

前文中分析了 FLAT 算法设计的关键问题,以定理 1 为设计原则定义数据流的传输时隙,并提出 FLAT 的核心优化问题,通过求解优化问题得到数据源端最佳的数据流发送速率,来保证交换机不会因为缓存溢出而丢弃数据包.同时,为了保证 FLAT 算法设计的可靠性,将算法 1 定义为函数 *src*(),并作为 FLAT 算法实现的关键函数(算法 2 中第 2 行).

算法 2. FLAT.

1. #function define
2. **def** *src*() /*源端速率控制函数*/
3. **def** *checkFlat*(*flag*) /*判断是否继续执行 FLAT*/
4. #begin
5. *flat_flag* /*算法执行标志位*/
6. *buffer_flag* /*判断到达数据包是否需要缓存:1(true)表示缓存;0(false)表示转发*/
7. *buffer_dumped_flag* /*判断缓存中是否存在待处理的数据包:1(true)表示不存在,即已经处理完毕;0(false)表示存在*/
8. *flat_flag*=true

```
9.  buffer_flag=false
10. buffer_dumped_flag=false
11. while flat_flag==true:
12.     T_w   /*获取传输时隙值*/
13.     src()
14.     buffer_flat=true
15.     sleep(T_w)   /*缓存 T_w 时间长度,即执行算法 3 中缓存函数*/
16.     buffer_dumped_flag=false   /*算法 3(第 13 行)中被重新置为 1(true)*/
17.     buffer_flat=false
18.      sleep(T_w)   /*转发 T_w 时间长度,即执行算法 3 中转发函数*/
19.      checkFlat(flat_flag)
20. end while
21. #end
```

算法 2 中,第 11 行~第 20 行描述了 FLAT 工作过程:当算法标志位 *flat_flag* 为真,则表示 FLAT 被触发,紧接着判断是否需要执行源端速率控制函数;当缓存标志位 *buffer_flat* 为真,则执行缓存函数并对到达的数据包进行缓存,函数执行周期为一个传输时隙 T_w(第 15 行),并将 *buffer_dumped_flag* 标志位置为 0,即表示缓存中存在待处理的数据包并将在下一个转发时隙优先处理(第 16 行);当缓存标志位 *buffer_flat* 为假,则执行转发函数并优先处理上一个时隙缓存的数据包,函数执行周期为一个传输时隙 T_w(第 18 行);算法第 19 行将判断一个循环周期(缓存和转发周期)结束后,是否继续执行算法.FLAT 算法的复杂度将在本文第 3 节具体分析.

由于当前的控制器只能提供 s 级的时钟粒度,而 FLAT 算法设计中的转发与缓存函数,需要到达 μs 级的时钟粒度,为了保证 FLAT 算法执行的效率,算法执行过程中的转发与缓存函数将通过扩展交换机中的转发模块来实现.算法 3 则具体实现了交换机处理 FLAT 数据包的过程,包括缓存和转发两个动作.

算法 3. FPP(FLAT packet processing).

```
1.  #function definition
2.  def output(pkt)   /*数据包转发函数*/
3.  def buffer(pkt,array)   /*数据包缓存函数*/
4.  #begin
5.  array   /*数据包缓存容器*/
6.  while new_pkt:
7.     if buffer_flat==true:
8.        buffer(new_pkt,array)
9.     else:
10.       if buffer_dumped_flag==false:   /*缓存中存在待处理的数据包*/
11.          foreach pkt in array:
12.             output(pkt)
13.             buffer_dumped_flag=true
14.             output(new_pkt)
15.       end if
16.    end if
17. end while
18. #end
```

其中,算法第 6 行~第 17 行描述了算法工作过程:当数据包到达交换机时,此时的缓存标志位如果为真,则需

要将该数据包缓存至 *array* 容器(第 8 行);如果缓存标志位为假,则先判断上一个时隙内是否有缓存的数据包并优先转发缓存的数据包(第 10 行~第 12 行),同时将 *buffer_dumped_flag* 标志位重新置为 1(第 13 行),然后再转发新到达的数据包(第 14 行).

2.2 原型系统设计

基于 SDN/OpenFlow 架构以实现 FLAT 算法为目标进行原型系统的设计,系统由 3 个层面构成:应用层、中间层的控制器平台以及底层的交换机网络.中间层为应用层提供 API 接口用于应用程序检测网络状态、下发控制策略,同时通过 SSH 安全通信协议与底层交换机网络建立连接.原型系统整体设计如图 3 所示,其中具有逻辑关系的功能模块用虚线连接表示.

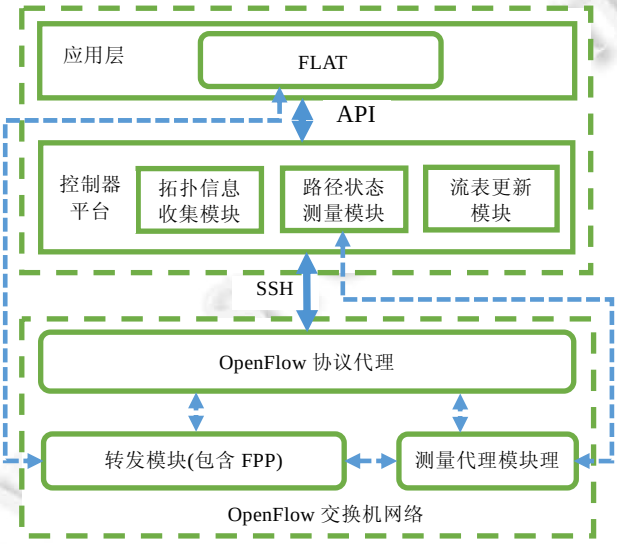

Fig.3 FLAT-Based porotype system design

图 3 基于 FLAT 原型系统设计

2.2.1 应用层

属于原型系统架构中的最高层,在应用层中用户可以自定义应用程序进而触发网络中的数据流定义事件.本文原型系统设计的应用层,主要实现 FLAT 数据流调度算法,并利用 API 接口与控制器实现通信,通过控制器将其编译后发送至底层交换网络,最终可以实现基于时隙传输的数据流调度算法,完成流量工程的目标.

2.2.2 中间层

属于系统架构中间层的控制器平台,是 SDN 架构的核心组件.控制器通过对拓扑信息以及相关测量信息的收集为上层的应用程序提供相关的数据支持,最终路由策略的实施通过流表,由南向接口的安全连接通道下发到网络节点交换机,交换机则根据流表进行转发实现应用层的目标.图 3 中,控制器平台通过 3 个功能模块完成以下任务.

(a) 拓扑信息收集模块通过向与其相连的交换机收集更新信息来发现网络拓扑.交换机可更新的信息包括发现邻居消息和每个邻居互联的链路状态消息,通过与交换机信息的交互,以形成全局的网络视图,并向应用层模块提供所需的拓扑信息.

(b) 路径状态测量模块与底层网络的测量代理模块,共同实现了 FLAT 基于时隙传输的主动测量架构.两模块通过虚线连接体现出功能模块的逻辑关系,即由控制器测量模块触发路径传输时延的探测任务,而底层交换机的测量代理模块负责封装探测包以及辨别探测包与普通数据包等任务.同时,测量模块还能够支持 OpenFlow 自带的原生测量功能,为控制器周期性轮询底层交换机以获取交换机相关的状

态统计信息进行汇总、分析,通过 FLAT 路由策略触发条件的判断(发现大流并且大流传输的链路负载超过设定的阈值),并为算法触发后的计算提供所需的测量信息.FLAT 策略的实现需要测量模块提供精准的路径传输时延信息,其主动测量开销将在第 3 节进行评估.

(c) 流表更新模块属于控制器自带组件,可驱动 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,FLAT 将利用该模块完成对底层交换机多传输路径流表的预安装.

2.2.3 底层网络

属于系统架构最底层的交换机网络,实现数据的转发任务并且执行转发任务的交换机均支持 Openflow 协议.OpenFlow 交换机运行协议代理软件,并通过 SSH(secure shell)或 TCP 连接与控制器相连,并进行流表的安装和分析.当前,支持 OpenFlow v1.3+的交换机都具有统计交换机状态信息的功能,实质是交换机能够统计各个转发端口处理数据流的信息,并可以对控制器周期发出的 Read_State 轮询消息进行响应并提交相关数据,属于 OpenFlow 原生测量范畴.另外,FLAT 策略的实施以及主动测量的实现还需要对底层网络的转发平面进行扩展,即需要在交换机转发模块中扩展并实现算法 3(FPP)以及测量代理模块.其中,测量代理模块完成以下功能.

- 探测包封装操作:主要负责根据控制器下达的测量任务(由测量模块触发),封装探测包能够区分出全局路径时延探测任务(全局探测)和 FLAT 路径时延探测任务(局部探测).限于篇幅,这部分内容可参阅文献[22].
- 时间戳操作:主要负责为探测包打上时间戳.首先发送端根据测量任务为初始探测包打上时间戳,探测包到达接收端后,再打上时间戳标识到达时间.最后,通过测量系统的探测包回收函数对到达时间和发送时间戳求差,最终得到测量传输路径的时延值.通过将传输时延和定义的探测包生存时间相比较,若比较结果发现传输时延小于定义的生存时间,则探测包有效,并对测量模块中的路径状态表进行相应的更新操作,为应用层模块提供数据支撑;否则,丢弃探测包.
- 探测包转发操作:主要负责根据探测包携带的传输路径信息进行转发端口匹配操作,并进行相关的端口转发.另外,在定义的测量传输路径上,交换机根据获取的探测包路径信息并与本地交换机信息匹配,若不是探测包的目的端交换机,则需要根据探测包携带的测量路径信息从相应的端口进行转发.

另外,交换机将预先安装控制器下发的静态流表.以 FatTree 拓扑为例,为每条数据流选择最多不超过 $k/2$ 条不相交的传输路径(k 为交换机级联端口数),并且只有一条路径处于激活状态,其余路径暂时休眠直到 FLAT 算法启动.如果在实际网络环境中部署,还可以对转发路径采用“Shadow MACS”方案[23],这样做的好处是能够降低控制器与交换机的通信开销.

3 开销评估

数据中心采用集中控制方式的流量均衡,面临的一项重要挑战就是如何在细粒度的均衡策略与策略实施所产生的相关开销之间进行取舍.首先,控制器与交换机之间过于频繁的通信虽然能更准确地掌握网络状况,更好地进行细粒度的负载均衡,但是会产生巨大的开销,甚至会造成两者之间通信链路的拥塞而导致控制器流表下发失败,严重的状况下会使整个网络瘫痪;其次,下发过于庞大的转发流表也会造成数据平面开销增大,尤其考虑到 TCAM 的容量和成本;最后,由于 FLAT 路由策略的实施还需要主动测量架构获取的精确测量信息作为支持,然而主动测量必然会带来额外的测量开销.另外,控制器的计算开销主要是由 FLAT 算法的复杂度来决定的,由于算法中 T_w 的计算是基于对路径状态表的遍历,而每条数据流选择最多不超过 $k/2$ 条不相交的传输路径,极限情况下的算法复杂度为 $O(n^2)$,属于多项式级的复杂度,因此,对 FLAT 实施过程中所产生的开销主要从控制平面、数据平面以及主动测量 3 个方面进行评估.

3.1 控制平面开销

文献[13](Hedera)指出,拥塞链路中的大小流碰撞是导致小流丢包以及网络性能下降的主要原因.对链路中产生的大流进行负载均衡,能够有效提高网络性能以及避免拥塞发生,同时,对于数据中心来说,能够降低控制平面与数据平面之间的通信开销.FLAT 路由策略的实施同样针对网络中出现的大流并且只需要在边缘层交换

机部署策略,相对于以往的研究工作需要在大流传输路径上所有相关的交换机进行策略部署来说,进一步降低了开销.

以 FatTree(K=8)拓扑为例,评估涉及到的参数设置如下:网络中主机服务器数量 H=128;核心交换机数量 N_c=16;汇聚层交换机数量 N_a=32;边缘层交换机数量 N_e=32;每台边缘层交换机连接服务器主机数量 H_e=4;每台主机每秒平均产生数据流 F=20 条;流表中每条数据流平均生存时间 T=60s;网络流量中大流出现的概率为 P=0.01[14].以 Hedera 大流调度策略所产生的控制开销作为参照,假设当前拓扑在 1s 内共产生大流 $H\times F\times P\approx$26 条,考虑极限情况即大流传输相关路径覆盖网络所有交换机,控制器针对目标大流需要同时下发流表 $26\times(N_c+N_a+N_e)$=2080,而 FLAT 只需要下发流表 $26\times N_e$=832,能节约 60%的控制开销.

3.2 数据平面开销

数据平面开销的主要考虑因素是下发流表的规模与 TCAM 容量的限制.TCAM 是交换机实现快速转发的关键部分,控制器下发的流表需要在 TCAM 安装,由于 TCAM 受到功耗大和高成本的限制使得存储容量有限,TCAM 内存溢出将使交换机转发能力降低,影响网络整体性能.假设流表在 TCAM 的生存时间同样是 60s,一台边缘层交换机则需要同时安装流表 $H_e\times F\times T\times P$=48,以 PICA8 交换机提供的 TCAM 最低存储流表容量 2K 为例,FLAT 策略实施只需要消耗 TCAM 总存储容量的 2.4%,极大地节省了 TCAM 存储空间,降低了数据平面的部署开销.

3.3 测量开销

由于 FLAT 策略的执行需要实施主动测量为其提供精确的测量数据支撑,然而主动测量必然会带来额外的测量开销,因此同样需要对测量的开销进行评估.测量开销的评估需要考虑以下要素:探测包大小、探测包发包频率、传输链路带宽以及探测拓扑规模[24].其中,针对本实验探测拓扑规模是由目的边缘层交换机数量以及测量的传输路径数量共同决定.首先提出测量开销评估公式,如下所示:

$$\frac{probeSize\times N_e\times\tau}{probeSize\times C_l}\times 100\% \tag{9}$$

其中,$probeSize$ 表示探测包大小,N_e 表示边缘层交换机数量,τ表示预设的多路径传输数量,$probeFre$ 表示探测包发送频率,C_l表示链路带宽.同样以 K=8 的胖树拓扑为例,探测包基于单向时延测量传输路径,各评估要素的大小计算如下.

$probeSize$:探测包大小构成包括以太网帧头+帧间隙=20bytes,IP 报文头 20bytes,UDP 报文头 8bytes,meas_data 数据元字段 44bytes,总共 92bytes;N_e=32;$\tau=k/2$=4;$probeFre$ 以局部探测的 1s 计数;链路带宽假设为 100Mbps.最终,测量总开销为 0.094%.

从测量评估结果来看,基于 FLAT 实施的主动测量系统所产生的开销并不会对正常的数据通信产生影响.如果将探测的目标交换机数量提高 1 个数量级,测量开销依然能够控制在 1%以内,因此在网络设备硬件条件允许下,还可以进一步优化测量精度,例如缩短探测包测量频率等.

4 仿真与评估

仿真实验的目标是检验基于 FLAT 算法的原型系统是否能很好地解决数据中心的流量工程问题.前文中提到,数据中心流量存在以用户访问、查询为主的南北流量和以备份、迁移为主的东西流量,本次实验模拟以东西流量为主的场景,因为数据的备份、迁移很容易产生大流,而链路中大、小流的碰撞是造成网络拥塞的主要原因.本节将搭建 Mininet+Floodlight 仿真实验平台,并在平台基础上测试 FLAT,通过与 ECMP 路由机制以及模拟 Hedera 中 GFF 的路由算法进行性能的比对,体现 FLAT 的优越性.

4.1 仿真平台部署

4.1.1 仿真平台选择

Mininet 是基于 Linux Container 架构开发的进程虚拟化平台,可以对基于 OpenFlow,Open vSwitch(OVS)等

各种协议进行开发验证,支持 SDN 任意拓扑的网络结构,并可以灵活地进行相关测试.在验证了设计的正确性后,所有代码几乎可以无缝迁移到真实的硬件环境中.Mininet v2.0 版本已经支持自定义网络拓扑,通常采用的方法是使用 Python 类进行定义,通过修改 topo-2sw-2host.py 文件中定义的 MyTopo 类,可以使 Mininet 在启动时就能够通过参数选择新定义的拓扑.

Floodlight 是一款基于 java 编写的开源的 SDN 控制器,其中的 Forwarding 模块主要实现路径发现和在设备间的数据转发功能,并且预置的 Forwarding 模块只支持单路径路由.由于本次实验中最关键的是要控制器实现多路径转发策略,需要对 Forwarding 类进行扩展,使其能够通过某种算法在现有的网络拓扑中寻找多条可用于数据流转发的路径,从而能够将数据流按照某种路由策略分配到其中某些路径上进行转发,避免拥塞链路的出现,同时避免出现接收端数据包乱序的问题.在实际扩展过程中,通过修改 doForwarding 方法,将 FLAT 算法扩展到系统中实现上述功能.

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,并安装 Ubuntu12.04 版本操作系统.为了调试方便,在这台宿主机上运行 Mininet V2.0 仿真平台,并基于 OVS v1.11 搭建 OpenFlow 交换机网络;同时运行 Floodlight V0.9 控制器以及支持 Openflow 协议的 wireshark 分析软件.当 Mininet 启动后,通过调用 Mininet.net 包中的 *net* 函数 *net.addController*(‘*controller*’,*controller=RemoteController*, *ip*=“127.0.0.1”,*port*=“6653”)建立与控制器之间的通信连接.

4.1.2 拓扑设计

目前数据中心以服务器为核心的拓扑方案多采用 FatTree 拓扑[25],FatTree 将交换机分为 3 层结构:边缘层、汇聚层和核心层.构建一个 K 叉树,则具有 K 个 POD(performance optimization datacenter),每个 POD 包含 K 个交换机,其中,$K/2$ 个属于边缘层交换机,另外 $K/2$ 属于汇聚层交换机;边缘层、汇聚层每个交换机都具有 K 个端口,其中,$K/2$ 个向下级联,$K/2$ 个向上级联;核心层总共有 $(K/2)^2$ 个核心交换机,每个交换机的 K 个端口分别连接到分属于 K 个 POD 的汇聚层交换机向上的级联口,整个网络可以容纳 $K^3/4$ 个端主机.实验构建一个 K=4 的 FatTree 拓扑,如图 4 所示.

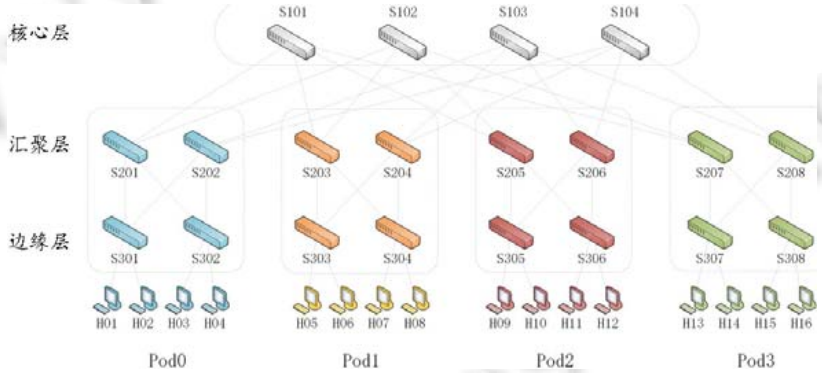

Fig.4 Test topology

图 4 实验拓扑

在实验拓扑中,交换机核心层编号为 S101~S104,汇聚层编号为 S201~S208,接入层编号为 S301~S308;终端主机编号为 H01~H16.由于不同 POD 主机之间的通信都需要经过核心层交换机进行转发,根据 FatTree 拓扑的对称特性,在 K=4 叉树下很容易算出属于不同 POD 主机之间存在 4 条等价路径,本次实验随机选取其中 2 条路径用作流量均衡,真实地模拟数据流竞争带宽的场景.

4.2 实验参数设置

4.2.1 网络运行参数设置

在实际部署 FatTree 这种多层拓扑的数据中心网络中,汇聚层到核心层采用 10G 链路带宽,其余采用 1G 链

路带宽.但是对于 Mininet 仿真平台来说,链路带宽按照以上设置将无法模拟实际的网络拥塞状况.因此,在实验中所有链路带宽被设定为 100Mbps.任意选取不同 POD 之间主机组成测试的源目节点对,例如,设定 POD1 和 POD4 之间 H01~H13,H02~H14,H03~H15,H14~H16 为 4 组源目主机对,第 1 组为测试组,后 3 组之间的数据传输作为背景流量,以此类推.实验中,将测试组主机注入流量逐步增大,使瓶颈链路负载以固定步长从 0 增加到接近满载来进行网络性能方面的测试.为了更真实地模拟实际场景,通过配置随机数产生器,使每个数据源端在 0~1s 内由产生的随机数决定数据流开始传送的时刻,同时,也会加入 UDP 流量来模拟小流作为背景流量的场景.控制器每隔 5s 进行一次路由策略计算.

4.2.2 传输时隙值设置

FLAT 算法重新定义了数据流传输时隙,时隙值是以不小于节点对之间有效的多条链路时延差为起点值,如果时隙值与数据流传输时间间隔天然吻合,就属于最佳情况;否则,无论偏差过大或者过小都将影响流量工程的实施效果.数据中心流量具有高突发性,并且物理链路的高带宽低时延决定了数据流在传输过程中存在大量的因为等待接收端的确认 ACK 消息而暂停发送的空闲时间间隔,这些空闲的时间间隔之间传输的数据包类似流片[26].文献[8]对一个包含 30 个机架的数据中心进行数据流特性分析,其中每个机架能够提供 4 500 台虚拟主机并支持 2 000 个不同的企业级应用,通过对 150GB 的数据进行抓包分析后得出结论,即传输 10MB 的数据中 500μs 的发送间隔占整个传输过程的 80%,100μs 占 90%以上.因此,本文实验当测得δ值小于 100μs 时,FLAT 的传输时隙值 T_w 就设置为 100μs;当δ值大于 100μs 时,T_w 就设置为 500μs;当δ值大于 500μs 时,则以实际测量的路径时延差值作为传输时隙值.另外,通过设置 10μs、500μs 这两个阈值,也避免了由于传输路径时延差值的变化使得控制器过于频繁地更新路由转发策略,并降低控制器与交换机之间的通信开销,提高了路由转发策略执行的效率.

4.3 性能比较

4.3.1 实验对象的选择

FLAT 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 Hedera 中的 GFF 路由算法.ECMP 路由机制属于静态的路由算法,是在计算出两节点间的多条可用路径后,在可供选择的多条等价链路上平均的分配流量.相对于单路径路由,该算法的优点是能够大幅度提高网络吞吐量,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.Hedera 主要针对网络中出现大流的场景,即当边缘层交换机监测到链路产生大流时,触发大流调度算法,其中的 GFF 路由算法就是当网络监测到大流产生的时候,为该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,为一下次搜寻做准备.相对于 ECMP,GFF 能根据链路状态进行路由转发.针对每一种路由策略,将根据不同阶段的链路负载值对每个 POD 进行 4 次实验采集数据,获取 16 组有效数据来进行最后的均值计算,以保证实验的准确性.

4.3.2 丢包率比较

首先观察 FLAT 丢包率的表现,实验结果如图 5 所示.其中,纵坐标表示丢包率,横坐标表示链路负载.当链路负载超过某个阈值时,网络性能将到达一个临界状态,要么趋于稳定,要么开始急剧恶化.为了更准确地描述网络性能随链路负载增大的变化,后面的实验将以链路负载率达到 90%为阈值,同时将实验结果分为两个部分:一是链路负载低于 90%并以 15%步长增长的情况下实验目标值的变化,二是负载高于 90%并分别以 5%和 1%步长增长的情况下实验目标值的变化.实验观察到:图 5(a)随着链路负载逐渐增大,采用多路径转发方式的丢包率明显比单路径转发要低;当链路负载达到 90%成为网络性能的临界点时,观察图 5(b)可以看到,此时丢包率稳定,而单路径转发丢包率达到 50%,部署 FLAT 路由策略丢包率最低.实验结果表明:当链路负载逐渐增大并使瓶颈链路出现拥塞时,单路径路由转发策略由于无法及时调整转发路径因而丢包率急剧增长;多路径转发克服了单路径转发的不足,由于 ECMP 属于静态路由策略,相对于 GFF 和 FLAT 动态路由策略,丢包率差距较明显;GFF 在对数据流进行路径切换的过程中,尤其是针对大流依然容易出现丢包的现象,而 FLAT 克服了 GFF 的不足,在实验过程中,丢包率始终处于较低的水平并且保持平稳,体现出 FLAT 路由策略的优越性.

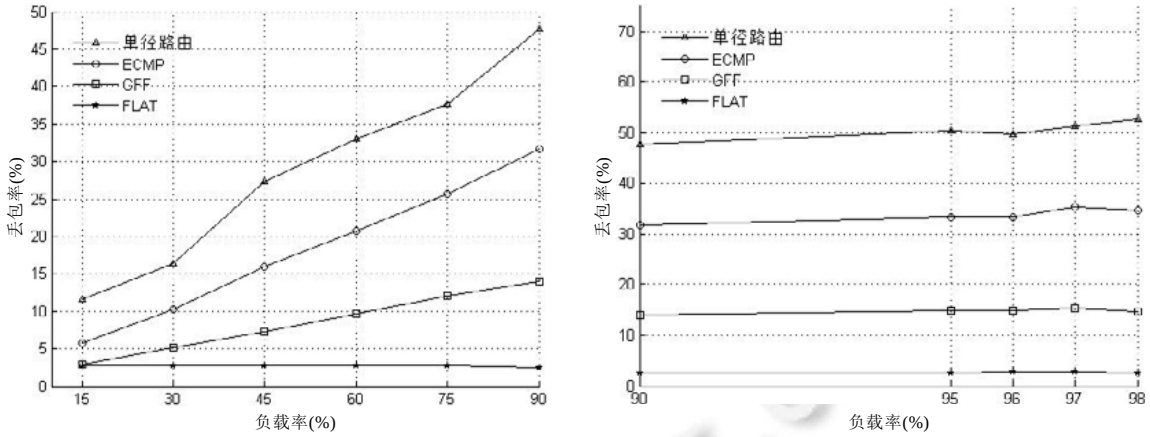

Fig.5 Comparison of packet loss rate

图 5 丢包率比较

4.3.3 吞吐量比较

网络吞吐量的比较测试结果如图 6 所示.从图 6(a)观察到,链路负载从 15%开始,4 种路由转发策略除了 FLAT 以外,吞吐量都开始下降,整体波动较大;从图 6(b)观察到,链路负载达到 90%以后,吞吐量趋于稳定,此时单路径路由转发吞吐量最低,FLAT 路由策略下,吞吐量最高.实验结果表明:当节点交换机采用 FLAT 路由策略进行部署时,与使用 ECMP 以及 GFF 部署时相比,网络吞吐量明显高于后两者.这种现象与丢包率测试结果相一致,进一步验证了 FLAT 的优越性.

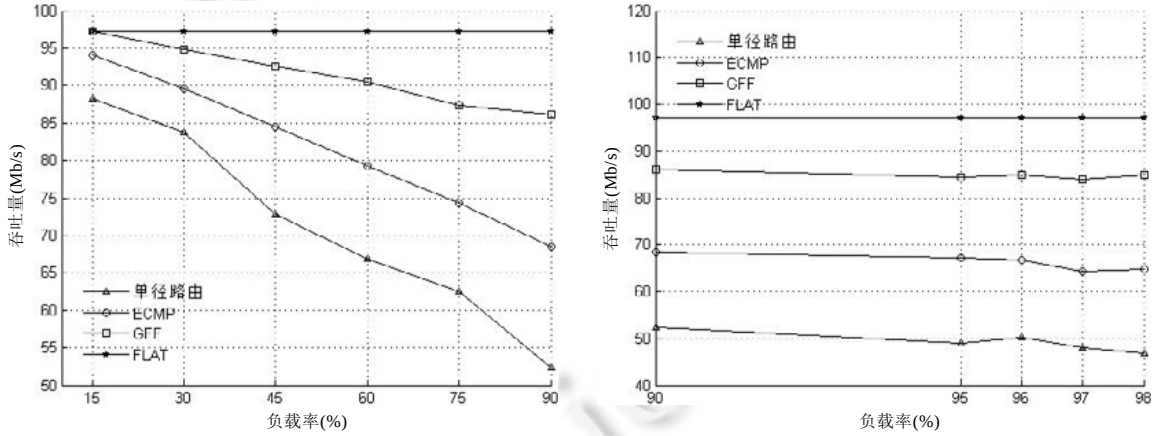

Fig.6 Comparison of throughput

图 6 吞吐量比较

4.3.4 时延抖动比较

时延抖动也是衡量网络性能的重要指标,实验结果如图 7 所示,4 种路由策略均存在不同程度的时延抖动.出现这种现象的原因与目前数据中心主要采用 TCP 传输协议相关,随着负载的增大,链路出现拥塞时,网络性能开始下降,丢包率开始上升;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,是造成实验结果波

动的原因,这也是数据传输采用 TCP 拥塞控制机制所无法避免的.实验中,单路径路由时延抖动振幅最大,基于多路径传输的路由策略时延抖动振幅相对较小,其中,从图 7(a)观察到,GFF 时延抖动振幅要大于 ECMP 和 FLAT;而图 7(b)中,FLAT 时延抖动振幅却高于 ECMP 和 GFF.实验结果表明:基于 SDN 架构的路由策略,当到达交换机的数据包未匹配当前转发流表时,将被发送到 Floodlight 控制器,再由 Forwarding 模块进行路由策略的计算,并向该路径所有 OF 交换机下发流表项,后继属于同一条流的数据包就可以直接进行转发,这个过程会对时延抖动产生影响.图 7(a)中,FLAT 时延抖动振幅最小,ECMP 次之.由于 GFF 是在出现大流之后触发流调度算法,对小流依然采用 ECMP 路由方式,因而 GFF 时延抖动振幅要大于 ECMP.随着链路负载超过阈值,如图 7(b)所示,此时,FLAT 时延抖动振幅较大.这是由于 FLAT 算法机制需要对某个时隙的数据包进行缓存转发,当链路负载过大,缓存的数据包增多,同时,向控制器请求流表转发的信号也将增多,从而导致时延增大.当时延增大到某一值后,交换机缓存队列长度也将超过阈值,此时,FLAT 将触发算法 1,通过控制源端发送速率,时延将迅速回落,体现出 FLAT 算法的优越性.

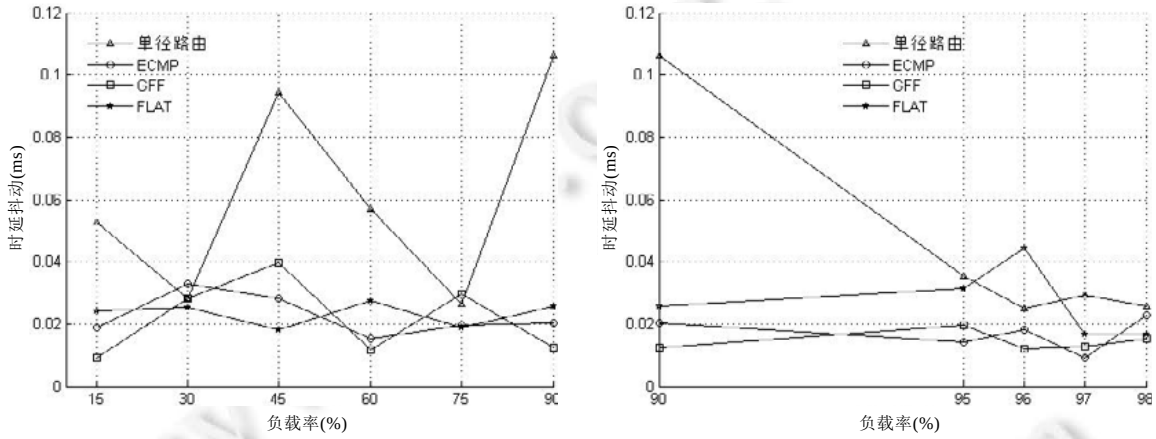

Fig.7 Comparison of delay jitter

图 7 时延抖动比较

从以上的实验结果中可以得出结论:数据中心采用多路径路由,能够明显改善单路径路由由于“选路”集中而造成的拥塞问题,同时能够做到故障链路的快速切换,并能聚合链路带宽,充分使用网络资源.但是多路径路由在对数据流尤其是大流由于链路拥塞而进行迁移的过程中容易出现的丢包以及接收端数据包乱序的问题,FLAT 算法机制很好地克服了数据流迁移过程中的丢包现象并保证接收端不会出现数据包乱序问题.通过与目前普遍采用的 ECMP 多路径路由机制以及 GFF 动态的多路径路由机制相比较,以上实验结果能够体现出 FLAT 算法的优越性.

5 结束语

本文针对数据中心采用多路径传输的路由机制进行流量均衡时,在面对数据流尤其是大流的迁移过程中容易造成数据流丢包以及接收端乱序的问题,在 SDN 的架构下,提出了一种基于时隙传输的路由算法:FLAT.该算法通过集中控制的方式获取链路状态信息,计算出合理的数据流传输时隙值,在完成细粒度的流量均衡的同时,能够避免在数据流迁移过程中的丢包以及接收端数据包乱序问题.通过 Mininet 仿真平台,在基于 FatTree 网络拓扑上进行 FLAT 的仿真实验,验证了 FLAT 在提高网络性能方面的优越表现.下一步工作将在真实的数据中心环境中对 FLAT 进行实际的部署以检验其有效性;基于 Mininet 仿真平台进行的仿真实验,其所有代码几乎可以无缝迁移到真实的硬件环境中,这也为下一步工作奠定了坚实的基础.

References:

[1] Yukihiro N, Kazuki H, Lee CH, Shinji K, Osamu S, Takeshi S. DomainFlow: Practical flow management method using multiple flow tables in commodity switches. In: Proc. of the 9th ACM Conf. on Emerging Networking Experiments and Technologies (CoNEXT 2013). New York: ACM Press, 2013. 399−404.

[2] Theophilus B, Aditya A, David AM. Network traffic characteristics of datacenter in the wild. In: Proc. of the 10th ACM SIGCOMM Conf. on Internet measurement (IMC 2010). New York: ACM Press, 2010. 267−280.

[3] Albert G, James RH, Navendu J, Srikanth K, Kim C, Parantap L, Maltz DA, Patel P, Sengupta S. VL2: A scalable and flexible data center network. Communications of the ACM, 2011,54(3):95−104.

[4] Zhou J, Malveeka T, Zhu M, Kabbani A, Poutievski L, Singh A, Vahdat A. WCMP: Weighted cost multipathing for improved fairness in data centers. In: Proc. of the 9th European Conf. on Computer Systems (EuroSys 2014). New York: ACM Press, 2014.

[5] Bharti S, Pattanaik KK. Dynamic distributed flow scheduling for effective link utilization in data center networks. Journal of High Speed Networks, 2014,20(1):1−10.

[6] Wu X, Yang XW. DARD: Distributed adaptive routing for datacenter networks. In: Proc. of the 32nd IEEE Int'l Conf. on Distributed Computing Systems (ICDCS 2012). Piscataway: IEEE Press, 2012. 32−41.

[7] Ford A, Raiciu C, Handley M, Barre S, Iyengar J. Architectural guidelines for multipath TCP development. 2011. http://tools.ietf.org/html/rfc6182

[8] Mohammad A, Tom E, Sarang D, Vaidyanathan R, Chu K, Fingerhut A, The Lam V, Matus F, Pan R, Yadav N. CONGA: Distributed congestion aware load balancing for datacenters. In: Proc. of the 2014 ACM Conf. on SIGCOMM (SIGCOMM 2014). New York: ACM Press, 2014. 503−514.

[9] Ian FA, Lee A, Wang P, Luo M, Chou W. A roadmap for traffic engineering in SDN-OpenFlow networks. Computer Networks, 2014,71(1):1−30.

[10] Chen K, Hu CC, Zhang X, Zheng K, Chen Y, Vasilakos AV. Survey on routing in data centers: Insights and future directions. IEEE Network, 2011,25(4):6−10.

[11] Arsalan T, Martin C, Teemu K, Shenker S. Applying NoX to the datacenter. In: Proc. of the 8th ACM Workshop on Hot Topics in Networks (HotNets-VIII). New York: ACM Press, 2009. 1−6.

[12] Theophilus B, Ashok A, Aditya A, Zhang M. MicroTE: Fine grained traffic engineering for data centers. In: Proc. of the 7th Conf. on Emerging Networking Experiments and Technologies (CoNEXT 2011). New York: ACM Press, 2011.

[13] Mohammad A, Sivasankar R, Barath R, Huang N, Vahdat A. Hedera: Dynamic flow scheduling for data-center networks. In: Proc. of the 7th USENIX Conf. on Networked Systems Design and Implementation (NSDI 2010). Berkeley: ACM Press, 2010.

[14] Andrew R, Wonho K, Praveen Y. Mahout: Low-Overhead datacenter traffic management using end-host-based elephant detection. In: Proc. of the IEEE INFOCOM 2011 (INFOCOM 2011). Piscataway: IEEE Press, 2011. 1629−1637.

[15] Jonathan P, Amy O, Hari B, Shah D, Fugal H. Fastpass: A centralized zero-queue datacenter network. In: Proc. of the 2014 ACM Conf. on SIGCOMM (SIGCOMM 2014). Chicago: ACM Press, 2014. 307−318.

[16] Andrew R, Jeffrey C, Jean T, Yalagandula P. DevoFlow: Scaling flow management for high-performance networks. In: Proc. of the 2011 ACM Conf. on SIGCOMM (SIGCOMM 2011). New York: ACM Press, 2011. 254−265.

[17] Cao J, Xia R, Yang P, Guo C, Lu G, Yuan L, Zheng Y, Wu H, Xiong Y, Maltz D. Per-Packet load-balanced, low-latency routing for clos-based data center networks. In: Proc. of the 7th Conf. on Emerging Networking Experiments and Technologies (CoNEXT 2013). New York: ACM Press, 2013. 49−60.

[18] Michele G, Renato L, Michela M, Marsan MA. Closed queueing network models of interacting long-lived TCP flows. IEEE/ACM Trans. on Networking, 2004,12(2):300−311.

[19] Mishra B, Singh S, Joshi C. Modeling arrival rate and service rate for next generation network as an open queue using M/M/1 queue model. In: Proc. of the Int'l Conf. and Workshop on Emerging Trends in Technology (ICWET 2010). New York: Association for Computing Machiner Press, 2010. 401−405.

[20] Lu ZL. Queuing Theory. 2nd ed., Beijing: Beijing University of Posts and Telecommunications Press, 2009 (in Chinese).

[21] Siegfried S, Jianming S. Fractional programming: The sum-of-ratios case. Optimization Methods and Software, 2003,18(2): 219−229.

[22] Yang Y, Yang JH, Wen HS. Another SDDC(software defined data center)-oriented method and device for traffic balancing: China. 2016103261765, 2016 (in Chinese).

[23] Kanak A, Colin D, Eric R, Carter J. Shadow MACs: Scalable label-switching for commodity ethernet. In: Proc. of the 3rd Workshop on Hot Topics in Software Defined Networking (HotSDN 2014). New York: ACM Press, 2014. 157−162.

[24] Naga K, Mukesh H, Changhoon K, Sivaraman A, Rexford J. HULA: Scalable load balancing using programmable data planes. In: Proc. of the Symp. on SDN Research (SOSR 2016). New York: ACM Press, 2016.

[25] Eric J, Deng P, Liu J, Liu J. A simulation and emulation study of SDN-based multipath routing for FAT-TREE data center networks. In: Proc. of the 2014 Winter Simulation Conf. (WSC 2014). Piscataway: IEEE Press, 2014. 3072−3083.

[26] Srikanth K, Dina K, Shantanu S, Berger A. Dynamic load balancing without packet reordering. ACM SIGCOMM Computer Communication Review, 2007,37(2):51−62.

附中文参考文献:

[20] 陆传赉.排队论.第 2 版,北京:北京邮电大学出版社,2009.

[22] 杨洋,杨家海,温皓森.一种面向软件定义的数据中心流量均衡方法及装置:中国.2016103261765,2016.

杨洋(1980－),男,江苏无锡人,博士,讲师,主要研究领域为计算机网络,路由协议,流量工程,SDN.

温皓森(1994－),男,博士生,主要研究领域为 SDN,并行计算.

杨家海(1966－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机网络,网络管理与测量,网络空间安全,云计算.