SDN中基于全局拓扑感知的自适应流量均衡算法

王莅晟[1],伊 鹏[1],胡 涛[1],江逸茗[1],胡静萍[1],胡宗魁[2]
(1. 中国人民解放军战略支援部队信息工程大学信息技术研究所,河南郑州 450002;
2. 中国人民解放军第91445部队,辽宁大连 116043)

摘 要: 针对软件定义网络(Software-Defined Networking,SDN)中控制平面和数据平面之间的控制链路性能受限导致的Packet_In传输瓶颈问题,提出了一种自适应的流量均衡算法,利用SDN网络控制平面拥有全局拓扑和交换机实时状态的特点,运用阈值对流量均衡起止条件进行控制,通过将超载交换机中流量重定向到邻居交换机的一跳转发的自适应方法,解决上行控制链路瓶颈问题. 与现有方法相比,减小了33%的上行控制链路负载与50%的Packet-In消息丢包率,且部署开销小.

关键词: 软件定义网络; 控制链路; 全局拓扑; 流量优化; 自适应
中图分类号: TP393.0 **文献标识码:** A **文章编号:** 0372-2112 (2021)05-0964-11
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20190989

Adaptive Flow Balancing Algorithm Based on Global Topology Awareness in SDN

WANG Li-sheng[1], YI Peng[1], HU Tao[1], JIANG Yi-ming[1], HU Jing-ping[1], HU Zong-kui[2]
(1. *Institute of Information Technology, PLA Strategic Support Force Information Engineering University, Zhengzhou, Henan* 450002, *China*;
2. *Unit* 91445 *of PLA, Dalian, Liaoning* 116043, *China*)

Abstract: An adaptive flow balancing algorithm is proposed for the Packet_In transmission bottleneck caused by the limited control link performance between the control plane and the data plane in software-defined networking (SDN). The control plane of the SDN network has the characteristics of having a global topology and real-time status of switches, therefore we solve the bottleneck problem of the uplink control link by using thresholds to control the starting and ending conditions of flow balancing and redirecting flow in overloaded switch to neighbor switch. Compared with the existing methods, the upload control link load is reduced by 33% and packet loss rate of Packet-In massage is reduced by 50%, and the deployment overhead is little.

Key words: software defined network; control link; global topology; flow optimization; adaptive

1 引言

过去的几十年里,互联网的蓬勃发展深刻的影响了人们的生活,然而,传统TCP/IP网络设计初期未考虑到网络发展如此迅速,因此难以有效应对互联网发展中出现的越来越多的挑战,例如,高质量的网络服务、快速有效的网络部署、高效便捷的故障检测等[1].

针对传统网络中存在的缺陷,软件定义网络提出了数控分离的体系架构. 相对于传统网络,它具有控制逻辑集中,开放可编程,细粒度流管控的优势. 控制平面通过OpenFlow[2]协议掌握数据平面的全局拓扑,数据平面的交换机将新到达的流的报头分组封装成Packet-In消息通过OpenFlow代理(OpenFlow Agent,OFA)向上传递给控制平面,控制平面对路由进行计算后通过下放流表的方式在沿途交换机上安装相应转发规则,下放的流表均安装在TCAM(Ternary Content Addressable Memory,三态内容寻址存储器)[3]中. 本文将上行链路和下行链路统称为SDN的控制链路(Control Link).

收稿日期:2019-08-28;修回日期:2020-08-19;责任编辑:孙瑶
基金项目:国家重点研发计划"基于自适应网络平台的动态防御技术"(No. 2018YFB0804002);国家自然科学基金(No. 62002382)

虽然 SDN 相较于传统网络具有诸多的优势,但仍然存在着两大方面的问题亟待解决,负载均衡问题一直备受大家关注. 然而以往的负载均衡问题研究基本都集中在控制器的负载均衡问题上. 与控制器负载均衡问题相似,在上行控制链路中也存在负载均衡问题. 由于在 SDN 设计过程中将控制功能都交给了控制平面,这就势必会导致数据平面专注于转发功能,而不拥有很强的数据处理能力. 在上行链路方面,OFA 使用的是处理能力较低的 CPU,以此来节约成本,并降低结构复杂性. 但是由于 CPU 处理能力有限,导致 OFA 将新到达流的报头数据封装成 Packet-In 消息的速度有限. 这就造成了上行控制链路的传输能力十分有限[4]. 相对于早期控制平面单个 NOX[5] 控制器就达到了 30000 个/s 处理速度,在使用并行的 Maestro 控制器[6] 下甚至达到 600 000 个/s 的请求处理速度相比,上行控制链路基于 Pica8 交换机[7] 测出的 150 个/s 的 Packet-In 数据包上报速率就显得有些相形见绌了. 在数据中心中,当许多流量在短时间内涌入交换机时,由于 OFA 处理能力有限,新到达的流无法及时得到响应,导致用户 QoS(例如,响应时间和吞吐量)急剧下降. 更严重的,对于无法转发的数据包,OpenFlow 会选择将它们存储在缓存中,然而交换机缓存容量有限,当交换机缓存填满时,交换机会选择将整个数据包打包成 Packet-In 数据包上传,而不是只上传报头数据,这会进一步加大上行链路的负载,甚至导致上行链路的服务中断. 对于超时未发送的数据包,交换机会选择进行丢弃处理,进一步降低用户的 QoS.

因此,上行控制链路瓶颈问题与控制平面负载均衡问题一样对 SDN 系统性能有重要影响. 在以往的研究中,控制平面负载均衡问题已经进行了充分地研究[8~11],并有许多已经很成熟的应用. 然而相比较之下,业界对于控制链路问题仍然缺乏足够的研究. 文献[12,13]首次指出了 OpenFlow 交换机在现有硬件水平下存在明显的性能瓶颈的问题,即 Packet-In 数据包打包上报和流表下放安装速率过低. 并且文献[12]提出了只对路由请求和流统计数据进行上报,以此来缓解交换机压力. 但是这样就失去了 SDN 网络细粒度流管控的特色. 文献[6]提出一种运用虚拟交换机的方式,将大部分的网络流量请求迁移到虚拟交换机上进行操作,利用虚拟设备高性能的处理器进行 Packet-In 消息上报等操作,从而减少物理交换机上的链路负载. 然而这种方法需要消耗额外的资源,具有较高的部署开销. 文献[14]首次将上行和下行链路统一起来,提出了控制链路的概念,并且对控制链路可靠性进行了量化分析,提出了影响控制链路可靠性的因素. 文献[14]对控制链路可靠性问题进行了系统分析,对控制链路负载均衡问题进行了研究,提出了一种流量重定向的方式,将高负载交换机的部分流量用重定向方式通过其他交换机上传给控制器,可以很好地缓解突发流情况下的控制链路负载过高问题. 然而该算法将控制链路负载均衡问题建模为了 NP-hard 问题,因此只能得出近似解,其次该方法在流量重定向过程中以宏流为单位进行路由重定向,这就导致在重定向时受限于宏流大小,无法做到完全的负载均衡,性能上依然存在一定的瓶颈.

本文针对软件定义网路中长期存在的控制链路上行瓶颈问题,提出了一种新的自适应流量均衡算法(Adaptive Flow Balancing Algorithm,AFBA),本文的创新工作和主要贡献如下:

(1)设置重定向阈值控制系统负载,可以在部分交换机中数据流超出阈值后,将热点交换机中流量重定向到周围负载相对较低的交换机上,缓解单个交换机上行链路和缓存负载过大的问题.

(2)每个交换机只需要进行一跳转发,将流量传递给邻居交换机节点即可. 不必计算复杂的重定向路径,利用系统的自适应性进行负载均衡.

(3)部署简单,不消耗额外的系统资源,并且具有自适应特点,可以很好地适应交换机拓扑发生变化的情况.

2　问题分析与建模

2.1　问题分析

回顾 OpenFlow 交换机在新流到达时的操作,当一条新流到达 OpenFlow 交换机时,交换机首先会在已存储的流表项中查找是否有匹配条目,如果存在与该流匹配的流表项,则根据流表项对新流进行转发. 如果没有匹配条目,则由 OpenFlow 代理对新流进行处理,将元数据与新流前 128 字节的内容封装成 Packet-In 消息上报给控制器,其他部分则存储在交换机的缓存区中. 如果缓存区已满,则将整条新流打包成 Packet-In 消息上传给控制器,此时就会进一步增大上行链路的负载.

由 OpenFlow 交换机工作原理可以看到,对于单个交换机,在一段时间内,它会以完全随机的数量收到来自用户的流请求,在有突发流量时,设突发流请求的速率为 f_{in}(条/s). 同时,在有新流到达时,它就会启动查找转发以及打包机制,查找转发部分暂且不管,对于打包上传部分,交换机以 f_p(条/s)的速率将新流的报头打包为 Packet-In 消息,同时,未经处理的新流的剩余部分将会缓存在交换机中. 可以看到,当 $f_{in} < f_p$ 时,上行控制链路未满载,此时新到达流均可以得到及时处理. 然而当 $f_{in} > f_p$ 时,上行控制链路达到满载,导致新流无法得到及时处理,增大消息处理时延,降低用户 QoS. 同时新到达流无法得到转发就只能存储在缓存中,由于上下行链路以及控制平面存在处理延迟,会导致缓存区流数量越积越多. 当缓存区满之后,对于无法进入上行链

路的流请求就会采用丢弃的方式进行处理,导致系统可靠性显著下降,而当交换机上行链路和缓存区长期处于满载情况时,还会出现系统宕机等更加严重的可靠性问题,严重影响系统的性能.

因此,当 $f_{in} > f_p$ 时,为了防止大量的突发流对系统性能的影响,有四种思路解决该问题,第一种是减小 f_{in},然而突发流量是来自于用户,是一个完全不确定性的事件,目前并没有有效的办法来对 f_{in} 进行改变;第二种是选择性的将一部分流量上报,即只有一部分流量 f'_{in}(条/s)的流量需要处理,但是这违背了 SDN 细粒度流管控的要求,而 SDN 网络中诸如安全监测,流量工程等功能均是基于细粒度流管控实现的;第三种方法是增加 f_p,即增加控制链路上行带宽. 然而目前上行链路限制主要在于 OFA 较低的 CPU 处理性能,增加链路带宽即意味着给予交换机较强的处理性能,这与 SDN 发展初期的数控分离,将控制功能交给控制平面相违背,且会增加交换机成本,并不是一种有效的解决办法;第四种方法即重定向,以 f_t(条/s)的速率将流量重定向到其他设备,使得 $f_{in} \leqslant f_p + f_t$,以此来缓解单个交换机上行链路负载有限的问题. 如文献[7]的重定向到虚拟交换机和文献[14]的直接通过安装重定向路径重定向到控制器的方法,但是这两种重定向方法均会增加大量系统部署开销,且实现起来较为复杂.

2.2 问题建模

2.2.1 系统用网格间距元素

底层网络

本文中,考虑将软件定义网络结构用一个三元组来表示,即 $S(U,V,E)$. 一部分为 $U = \{u_1, u_2, \cdots, u_m\}$ 代表所有控制器,一部分为 $V = \{v_1, v_2, \cdots, v_m\}$ 代表所有交换机. 其中 $m = |U|$ 表示控制器数量,$n = |V|$ 表示交换机数量. 在多控制器 SDN 网络中,有 $m < n$,且对于每个交换机 $v_i \in V$,均存在控制器 $u_j \in U$ 与之相连. 对于交换机网络拓扑,我们将其抽象为 $G \subset S$ 且 $G = (V,E)$,其中 E 为所有数据链路的集合,并且对于每条链路 $e_k \in E$,我们用 $c(e_k)$ 表示其链路容量. 而对于每个交换机 v_i,均具有一个上行链路容量,用 $c_u(v_i)$ 表示,以及一个流表项容量,用 $c_d(v_i)$ 表示.

针对交换机上行链路瓶颈问题,本文提出一种新的流量重定向方法. 基本思路是利用高负载交换机附近负载较低的交换机来帮忙分担向控制器上报 Packet-In 消息. 具体执行过程如下:交换机周期性的对自己上行链路负载进行统计,控制器通过对全局网络中所有交换机的上行链路负载进行统计,向负载超过阈值的交换机下放重定向流表. 该流表具有以下特点:(1)只执行一跳转发;(2)优先级在精确匹配流表之后,在打包为 Packet-In 消息之前.

阈值设定

对于交换机 v_i,用 $F(v_i)$(条)表示当前待处理流量数,并定义其重定向阈值,如式(1)所示:

$$R_i = c_u(v_i) \cdot B \tag{1}$$

其中,B 为其重定向阈值因子. 用 V_i 表示与 v_i 距离只有一跳的所有交换机集合,$p_i = |V_i|$. 我们规定所有超出阈值的交换机集合为 V'. 当超过阈值时,将 $f_t(v_i, v_{ij})$ 条流量从 v_i 重定向到 $v_{ij}, j = 1,2,\cdots,p_i, v_{ij} \in V_i$. 为讨论简单,假设所有上行链路具有相同的链路容量,即所有交换机链路容量均为 $c_u(v)$,则阈值可以统一表示为式(2):

$$R = c_u(v) \cdot B \tag{2}$$

为增加系统弹性,设定重定向下限 T,其中 $T < R$. 即最终实现将超载交换机 v_i 负载降低到 T 或在集合 $\{v_i, V_i\}$ 中负载最低.

2.2.2 优化目标与约束条件

优化目标

对于本文提出的上行控制链路瓶颈问题,以最小化最大上行控制链路负载 μ(条)为控制目标,将本问题定义为一个最优化问题. 交换机最大上行控制链路负载如式(3)所示:

$$\mu = \max F(v_i) \tag{3}$$

则本问题的最优化目标如式(4)所示:

$$\min \mu, \forall v_i \in V \tag{4}$$

约束条件

用 0/1 变量 λ^{ij} 表示交换机 i 是否向相邻的交换机 j 重定向流量,用 C_t 表示交换机的流表项开销阈值. 且用 G_{ij} 表示由交换机 v_i 重定向到交换机 $v_{ij}, j = 1,2,\cdots,p_i$ 的数据流大小. 最后可以得到式(5)到(8)的约束条件.

表项开销约束

由于 SDN 系统中 TCAM 资源有限,且价格昂贵,因此在重定向时需要考虑系统总的表项开销,防止重定向占用过多系统表项资源导致系统出现拥塞,延时增加的情况. 因此规定系统总的流表项开销不得高于给定阈值 C_t,则交换机流表项开销约束如式(5)所示:

$$\sum_i \sum_j \lambda^{ij} < C_t, \ \forall_{i,j} \tag{5}$$

数据链路约束

重定向流量的同时也需要考虑会给数据链路带来的额外开销,如果重定向的流量数超过数据链路承载能力,可能会导致数据链路出现拥塞,造成数据传输延时,降低系统的性能,数据链路容量约束如式(6)所示:

$$f(v_i, v_{ij}) \cdot G_{ij} < c(e_{ij}), \ \forall_{i,j} \tag{6}$$

上行链路负载约束

重定向后,系统上行链路负载应该低于负载 μ,保证 μ 为系统最大上行控制链路负载,上行负载链路约束如式(7)所示:

$$F(v_i) - \sum_j \lambda^{ij} \cdot f(v_i, v_{ij}) \leqslant \mu, \forall_{i,j} \quad (7)$$

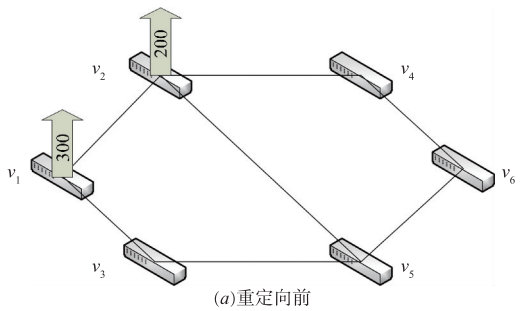
λ^{ij}取值约束

λ^{ij}决定了交换机 v_i 是否向交换机 v_{ij}进行重定向,λ^{ij}取值为 0 或 1,λ^{ij}为 1 时代表交换机 v_i 向邻居交换机 v_{ij}进行流量重定向,λ^{ij}为 0 表示不进行重定向操作,则 λ^{ij}取值约束如式(8)所示:

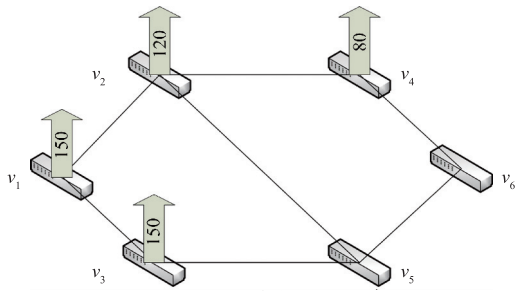
$$\lambda^{ij} \in \{0,1\}, \forall_{i,j} \quad (8)$$

3　算法设计

考虑到相比于上行控制链路每秒能处理的流量数,需要重定向的流量总大小远小于数据链路的传输容量 $c(e_{ij})$,因此可以认为 $c(e_{ij}) = \infty$. 针对以上的约束条件和和优化目标,首先用一个实例对工作流程进行说明,对于如图 1 所示的网络拓扑,参考文献[7]的数据,我们假设拓扑中交换机上报 Packet-In 消息的速度为 150 条/s. 假设某一时刻 v_1 到达了 300 条新流,同时 v_2 到达了 200 条新流,则按照默认的上传方式,v_1 将负载全部的 300 条流的上传任务,同时 v_2 也将负载全部的 200 条流的上传任务,如图 1(a)所示. 这方式显然超出了交换机的上报能力,会造成严重的链路拥塞甚至丢包现象. 本文针对这种情况提出的新方法,可以在出现如上问题时,将 v_1 和 v_2 中的流重定向到相邻的低负载的交换机. 对于如图所示的网络拓扑,将重定向阈值设为 100%,重定向下限设为 80%,即当交换机满载时向周围进行流重定向. 最后重定向结果如图 1(b)所示,可以看到,交换机 v_3 和交换机 v_4 分别负担了交换机 v_1 和 v_2 的上传负载,使交换机 v_1 和 v_2 不再过载运行,降低了传输延时和丢包可能,优化了交换机性能.



(a)重定向前



(b)重定向后

图1　算法流程显示图

3.1　基于全局拓扑感知的流重定向算法

针对 SDN 网络中存在的上行控制链路瓶颈问题,利用 SDN 网络拥有全局拓扑感知和信息收集能力的特点,对本文提出的重定向算法进行部署实施.

SDN 控制器通过实时收集网络中各交换机拓扑关系,获得每个交换机的邻居交换机信息,并存储在邻接矩阵 V 中用于实验使用. 周期性的收集系统中各个交换机的上行链路负载情况,并根据收集到的信息预测下一周期交换机中流量情况. 当系统中有交换机负载会在下个周期超过给定阈值后,触发模拟重定向过程,根据重定向规则,首先在控制平面进行模拟重定向,将超出阈值的流量重定向到邻居交换机,直到超载交换机负载低于重定向下限或者超载交换机负载在邻居节点中最低为止.

控制平面在进行一次模拟重定向后检测是否系统中依然会存在超载交换机,如果有则再次执行模拟重定向过程,直到在模拟的结果中没有交换机过载或者检测到模拟循环次数超出限制. 模拟重定向过程结束后再将每次重定向结果进行累加,并最终确定部署到交换机中需要进行重定向的流表情况,然后进行流表下发与部署,完成实际系统中的重定向过程. 该方法可以保证在每个周期内,每个交换机只会将自身流量转发至邻居交换机,而不会将其他交换机转发至自身的流量再转发出去,因此不会导致流量环路.

转发后的流量在到达目标交换机后与其他流量处理方式相同,被转发的流量必然是其他交换机未进行上报的流量,因此从其他交换机重定向至本交换机的流量也需要由上行控制链路上报给控制平面. 同时由于采用了周期性检测的方式,因此在本周期对流量进行转发后,如果经过了本周期的 packet-in 消息上报处理后,下周期该交换机中的流量数仍然超出了最大上行控制链路负载,且新到达的流量数少于下一周期进行重定向时需要转发给邻居交换机的流量数总和时,上周期转发至本交换机的流量会被二次转发至其他交换机,但这只会出现在极少数较极端情况下.

当控制器检测到系统中有交换机负载超过阈值后,启动流量负载均衡机制,算法整体流程设计如算法 1 所示.

算法 1　自适应流量均衡算法

1. **输入**:超载交换机负载情况
2. **初始化**:根据控制器全局拓扑信息获取超载交换机邻居交换机信息,并存储在邻接矩阵中
3. 执行流量重定向
4. while $V' \neq \varnothing$ and 循环次数不大于阈值 t do
5. for $v_i \in V'$ do

6. if $\exists F(v_{ij}) < F(v_i)$ then
7. 根据之前的重定向规则分情况对流进行重定向
8. end if
9. end for
10. end while
11. **输出**:根据每步重定向结果最终求得需要部署到每个交换机的重定向规则

3.2 重定向规则

前文所示实例只是系统中一种最为简单的情况,我们规定当 $F(v_i) > R$,即交换机负载大于重定向阈值 R 时,此时 v_i 有邻居交换机 $v_{i1}, v_{i2}, \cdots, v_{ip} \in V_i$,且 $F(v_{i1}) < F(v_{i2}) < \cdots < F(v_{ip})$,则针对各种不同交换机负载与阈值之间的关系,在具体几种情况下的重定向规则如式(9)~(14)所示.

规则 1 当 $F(v_i) < R$,即交换机负载小于重定向阈值 R 时,不进行重定向,此时:

$$f_t(v_i, v_{ij}) = 0 \tag{9}$$

规则 2 若对于 v_{i1},有 $F(v_{i1}) < R$,即负载最小的邻居节点负载低于重定向阈值,且 $F(v_i) - T < R - F(v_{i1})$,则

$$f_t(v_i, v_{i1}) = F(v_i) - T \tag{10}$$

规则 3 若对于 v_{i1},有 $F(v_{i1}) < R, F(v_i) - T > R - F(v_{i1})$,且对于 v_{ij}, $j = 1, 2, \cdots, m, m < p$,有 $F(v_{ij}) < R$,且 $(m-1)R - \sum_{j=1}^{m-1} F(v_{ij}) < F(v_i) - T < mR - \sum_{j=1}^{m} F(v_{ij})$,即 v_i 中流量总数减去 T 后,多出的流量可以重定向到 m 个邻居节点且这些邻居节点均不过载,则

$$\begin{cases} f_t(v_i, v_{ij}) = R - F(v_{ij}), j = 1, 2, \cdots, m-1 \\ f_t(v_i, v_{im}) = F(v_i) - (m-1)R + \sum_{j=1}^{m-1} F(v_{ij}) \end{cases} \tag{11}$$

规则 4 若对于 v_{i1},有 $F(v_{i1}) < R$,且 $F(v_i) - T > R - F(v_{i1})$,且对于 v_{ij}, $j = 1, 2, \cdots, m, m < p$,有 $F(v_{ij}) < R$ 以及 $F(v_{i(m+1)}) > R$ 且 $F(v_i) - T > mR - \sum_{i=1}^{m} F(v_{ij})$,即 v_i 中流量总数减去 T 后,多出的流量全部重定向到邻居节点后仍有多出流量,则

$$f_t(v_i, v_{ij}) = R - F(v_{ij}), j = 1, 2, \cdots, m \tag{12}$$

规则 5 若对于 v_{i1},有 $F(v_{i1}) > R$,且 $F(v_i) - R > F(v_{i1}) - R$,即邻居节点均超载但仍有负载低于 v_i 的节点,则

$$f_t(v_i, v_{i1}) = \frac{F(v_i) - F(v_{i1})}{2} \tag{13}$$

规则 6 若对于 v_{i1},有 $F(v_{i1}) > R$,且 $F(v_i) - R < F(v_{i1}) - R$,即则 v_i 为所有节点中负载最小的节点,则

$$f_t(v_i, v_{ij}) = 0 \tag{14}$$

重定向算法如算法 2 所示.

算法 2 流重定向算法

输入:重定向阈值 R,邻接矩阵 V,系统内节点负载情况
1. $\forall v_i \in V \ \& \ F(v_i) > R$
2. for $v_{ij} \in V_i, i = 1, 2, \cdots, n$
3. if $F(v_{ij}) < R$
4. if $F(v_i) - T < R - F(v_{ij})$
5. $f_t(v_i, v_{i1}) = F(v_i) - T$
6. else
7. $f_t(v_i, v_{ij}) = R - F(v_{ij})$
8. end if
9. else if $F(v_i) - R > F(v_{i1}) - R$
10. $f_t(v_i, v_{i1}) = \frac{F(v_i) - F(v_{i1})}{2}$
11. end if
12. end for
输出:重定向流量数和方向

3.3 算法分析

3.3.1 重定向阈值确定

由算法原理可知,当系统中仍有交换机存在空闲时,重定向阈值 $R = c_u(v) \cdot B$ 就决定了控制链路最大上行负载 μ,因此,在最理想的情况下,取重定向阈值如式(15)所示:

$$R' = c_u(v) \cdot B = \frac{\sum_{v_i \in V} F(v_i)}{n} \tag{15}$$

即系统中所有流量平均分配到了每个交换机中,此时可以得到式(16):

$$\min \mu = R \tag{16}$$

为系统理论上最小的 μ,但是在实际实现中,当大多交换机中流量都低于 R' 后,少数大于 R' 的交换机会需要经过很多次寻找和重定向才能够将流量实现完全平均,因此考虑引入参数 α,令

$$R = c_u(v) \cdot B = \alpha \cdot R' = \alpha \cdot \frac{\sum_{v_i \in V} F(v_i)}{n} \tag{17}$$

虽然实现的并不是完全的平均,但是可以在对指标 μ 和 ξ 形象很小的条件下减少系统循环次数,降低开销,参数 α 取值根据系统不同会有所变化,需在实验开始前通过仿真进行确认.

3.3.2 算法复杂度分析

对于每一个单位时间的流量请求,本算法计算过程包括:(1)检测交换机超载情况;(2)执行流量重定向过程. 其中当检测到交换机超载时,便会启动重定向过程,遍历超载交换机的所有邻居节点,进行一次重定向后,会再次查找是否有超载交换机并执行相同的重定向操作,直到没有交换机超载或循环次数超过某个阈值 t,因此可以得到本算法复杂度为 $O(n^2)$.

4　仿真及平台试验

4.1　指标确认与参数调整

针对以下的对比实验,提出两个衡量指标对其性能进行衡量:

(1)第一个指标为最大上行控制链路负载,即所有交换机中负载最大交换机的负载,也就是前文所提到的 μ. 以此来衡量最差情况下交换机的负载.

(2)第二个指标为定义的控制链路负载均衡因子 ξ,其表达式如式(18)所示:

$$\xi = \frac{\left(\sum_{v\in V} F(v)\right)^2}{n\sum_{v\in V}(F(v))^2} \tag{18}$$

$F(v)$表示交换机 v 的上行控制链路负载,该表达式的取值区间为[0,1],取值越接近于 1 表示整个系统负载越均衡. 用负载均衡因子来衡量整个系统的负载均衡情况.

进行实验前,首先需要确定一个重要的参数,即流量重定向的阈值 R. 由算法原理可以知道,阈值 R 越低,则会有越多的流量重定向到其他交换机中,也就是可以得到越低的最大上行控制链路负载 μ,越接近 1 的负载均衡因子 ξ,但是也会需要更大的部署开销 ε,如式(19)所示:

$$\varepsilon = \sum_i \sum_j \lambda^{ij}(\text{个}) \tag{19}$$

部署开销 ε 表示在部署重定向时占用了多少个 TCAM 流表项.

首先搭建了仿真实验平台,选取了一个拥有 100 个交换机节点和 50 个终端节点的校园网络拓扑[15],设定该网络数据链路容量为 100Mbps,表项开销阈值 $C_t = 0.01 \cdot c_u(v) \cdot n$,其中 n 为拓扑中交换机个数. 利用 mininet[16]仿真平台对以上拓扑结构中三种算法的情况进行仿真验证,运用随机算法模拟流量随机到达终端,为减少随机算法对算法结果造成影响,对每组仿真均重复进行 50 次,利用最终的平均值作为仿真的结果.

为了找到最合适的重定向阈值 R,使得最大上行控制链路负载 μ,负载均衡因子 ξ 和部署开销 ε 三者达到平衡,我们通过改变重定向阈值因子 B 从 0.1 到 1,即重定向阈值从 15 到 150 共十个不同阶段,让系统中流量总数从 2000 到 8000 随机分配到 50 台主机中,分别测试不同流量总数下,不同重定向阈值对于最大上行控制链路负载 μ,负载均衡因子 ξ 和部署开销 ε 的影响,为了消除随机性带来的误差,每组数据仿真 50 次并取平均数作为最终结果. 图 2 便是三种不同评价因素随着重定向阈值变化的仿真结果.

可以看到,随着重定向阈值因子增加,最大上行控制链路负载 μ 均会在保持一个值一段时间后成线性增长,直到增长到不重定向时的最大上行控制链路负载为止;而负载均衡因子则是在维持在 1 左右一段时间后急剧下降,到与不重定向的负载均衡因子相同时结束;部署开销则是会在保持一个很高的值之后突然急剧降低,到一个较小的值之后开始缓慢下降. 经过观察和对比分析,可以发现,这个在三种指标中造

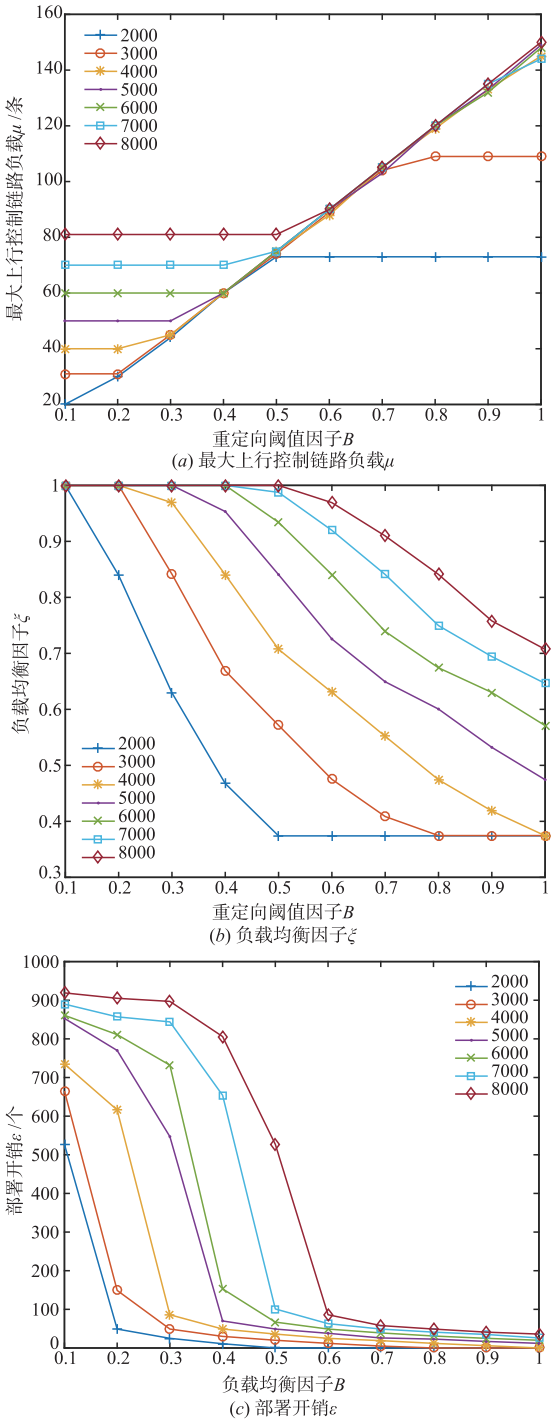


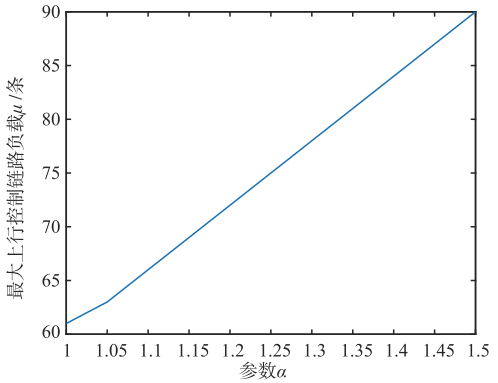
图2　重定向阈值因子B对评价指标影响

成显著拐点的重定向阈值 $R=F(v_i)\cdot B$ 均约等于理论值 R',即重定向之后所有流量平均分布在了每个交换机中时.

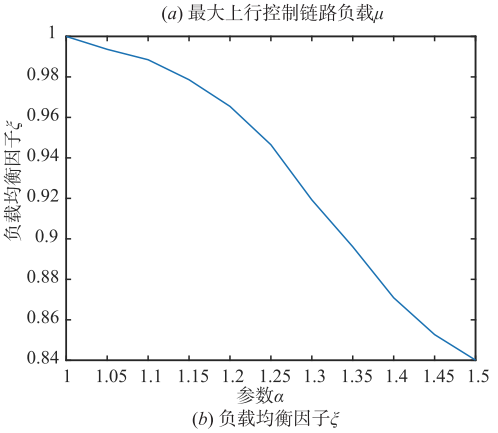
下面对三种指标的变化情况进行理论分析. 对于最大上行控制链路负载 μ,当重定向阈值小于等于 R' 时,由于重定向中规则 5 的原因,最终重定向结果均会使所有流量平均分布到每个交换机中,在图中就表现为重定向阈值因子 B 小于等于$\frac{R'}{c_u(v)}$时,μ 的值均等于 R',此时因为流量在交换机中均匀分布,因此负载均衡因子 ξ 也趋近于 1. 但是重定向阈值越小,每次重定向的流量数就越小,就会需要更多的循环次数,甚至到达循环上限限制才会跳出循环,因此过小的阈值不但会增加部署开销 ε,还可能取得比适当的阈值更差的重定向结果. 而当重定向阈值因子 B 大于$\frac{R'}{c_u(v)}$ 时,μ 的值会与 $F(v_i)\cdot B$ 相等. 直到重定向阈值大于最大上行控制链路负载. 此时相对而言流量分布没有之前均衡,因此最大上行控制链路负载 μ 和负载均衡因子 ξ 会差于 R' 小于等于 $\frac{\sum_{v_i\in V}F(v_i)}{n}$ 时,但此时因为需要进行重定向的流量较少,也就会需要更少的部署开销 ε.

综上所述,当重定向阈值设定为 $R'=\frac{\sum_{v_i\in V}F(v_i)}{n}$ 时,可以取得最好的最大控制链路上行负载 μ 和负载均衡因子 ξ,以及较小的部署开销 ε. 但是如果选择 $R=R'$,仍然可能会造成大量刚超出阈值的流量在交换机中出现震荡状况,因此考虑使重定向阈值 $R=\alpha\cdot\frac{\sum_{v_i\in V}F(v_i)}{n}$,增加参数 α,其中 $\alpha>1$,使重定向阈值 R 略大于 R',以在不显著降低系统性能的情况下进一步降低部署开销 ε. 首先对改变参数 α 的情况下三种指标的变化情况进行了仿真. 由于当系统中流总数从 2000 增长到 8000 时,三种指标的随参数 α 的变化趋势基本相同,因此为了便于分析,选取了网络中流总数为 6000 时的情况进行讨论. 如图 3 为三种指标随参数 α 的变化情况.

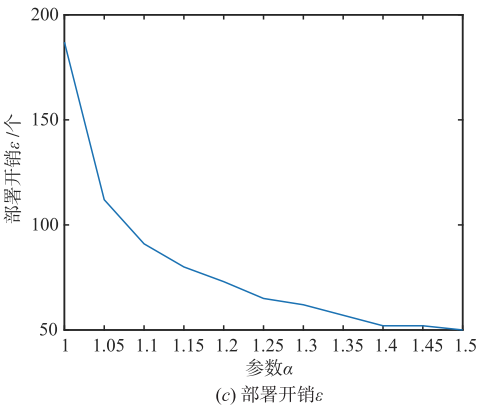
由图 3 可以看到,当参数 α 为 1 时,最大上行控制链路负载 μ 并未达到预期的 60,而是 61,这是因为限制循环次数后,在有限的循环次数下并未实现完全的均衡分配. 但增加循环次数上限会增加控制平面计算资源负载的增加,以及带来更大的部署开销. 当参数 α 取 1.05 后,最大上行控制链路负载 μ 达到了预期的 63,且负载均衡因子 ξ 仍保持在 0.99 以上,而部署开销 ε 则减少了近一半,因此增加参数 α 确实可以优化系统性能. 随着 α 的增加,μ 匀速上升,ξ 的下降速度越来越快,



(a) 最大上行控制链路负载μ



(b) 负载均衡因子ξ



(c) 部署开销ε

图3 参数α对评价指标影响

ε 的下降速度越来越慢. 这也就意味着 α 越大,相比 α 为 1.05 时,性能下降会越来越显著,而部署开销的节省力度会越来越小,因此 1.05 为最佳 α 取值.

经过仿真实验和理论分析,最终选取式(20)为系统重定向阈值.

$$R=1.05\cdot\frac{\sum_{v_i\in V}F(v_i)}{n} \tag{20}$$

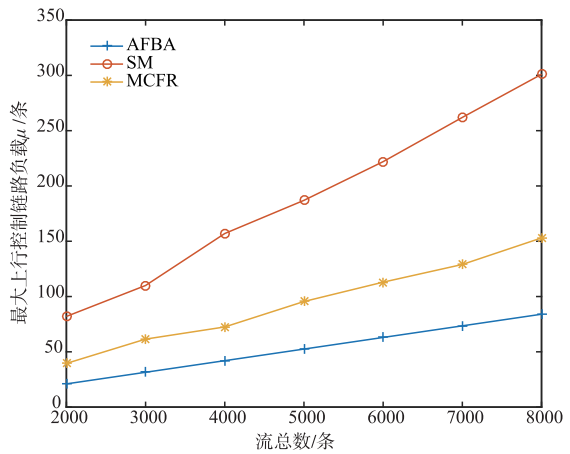
对于参数 α,由于在重定向阈值前增加 α 的目的为利用较小的系统负载换取较大的部署开销的减小,且 $R=\alpha\cdot\frac{\sum_{v_i\in V}F(v_i)}{n}$ 只需要略大于 $R'=\frac{\sum_{v_i\in V}F(v_i)}{n}$,因此 α 只需是一个略大于 1 的数即可,这也就导致在以 0.5 为

步进的实验中 1.05 具有最好的效果,进一步缩小步进长度或许可以取得更好的效果,但是效果提升极小,大多数情况下取 1.05 即可取得预期效果,后续实验中绝大多数情况可以直接使 α 等于 1.05 即可取得不错的效果.

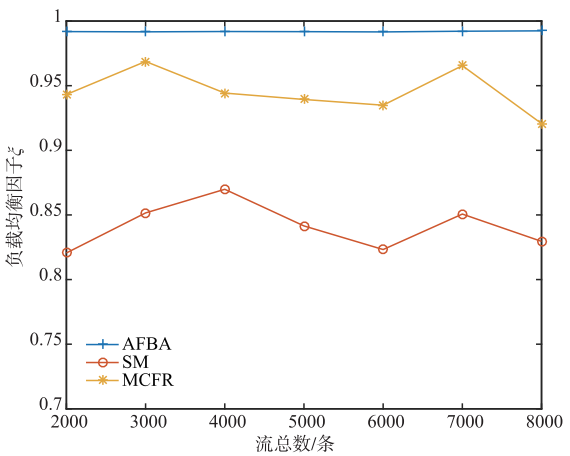
4.2　仿真实验

本文在确定参数时使用的仿真平台上进行了进一步的仿真实验,本次仿真实验依然选择最大控制链路上行负载 μ 和负载均衡因子两个衡量指标,用部署开销来衡量系统的额外负担,并加入考虑系统总的丢包率衡量系统整体的可靠性. 作为对比算法,第一个选取了最经典的 SM(the Static Method) 算法,即软件定义网络默认的流处理算法,不进行任何重路由等操作,由原交换机直接进行 Packet-In 消息的上报. 第二个选取了文献[14]中的 MCFR 算法(最小代价流重定向算法,Minimum-Cost Flow Redirecting),即以宏流(即一组源节点与目的节点相同的流)为单位用重定向的方式对流进行操作,以此来减轻交换机上行控制链路的压力.

在确定阈值后,首先进行了最大上行控制链路负载的对比,如图 4(*a*)所示,由于本文算法能够几乎将系统内所有流量均匀分布在每个交换机中,因此相较于 SM 算法以及 MCFR 算法具有更小的最大上行控制链路负载 μ. 而 MCFR 算法受限于宏流的大小,难以做到将流量完全分摊到所有交换机上,因此效果会稍差于本文的算法. 接下来进行了负载均衡因子的比较,如图 4(*b*)所示. 可以看出对于未进行重定向的流量,负载均衡因子在 0.85 左右,用 MCFR 算法进行重定向之后,负载均衡因子可以到达 0.95 左右,而相比之下,本文的算法可以使系统负载均衡因子保持在 0.99 及以上,具有很明显的优势. 而具有更高的负载均衡因子也就意味着系统中能够承载更多的突发流量,在解决上行控制链路瓶颈问题上具有更好的效果.



(*a*)不同流总数时最大上行控制链路负载μ变化



(*b*)不同流总数时负载均衡因子ξ变化

图4　三种算法最大上行控制链路负载与负载均衡因子对比

针对本文提出的算法,对其系统开销进行了仿真,结果如图 5 所示,可以看到,在流总数从 2000 到 8000 的变化过程中,系统的部署开销以很缓慢的速度在增长,且基本都保持在 100 到 110 之间,按文献[7]中一个交换机单位时间可以部署的流表数为 200 条,经过计算,规定的开销阈值 $C_t = 0.01 \cdot c_u(v) \cdot n = 200$,符合本文对于部署开销的要求,本算法平均来算一个交换机只需要额外部署一条流表即可完成整个系统的负载均衡.

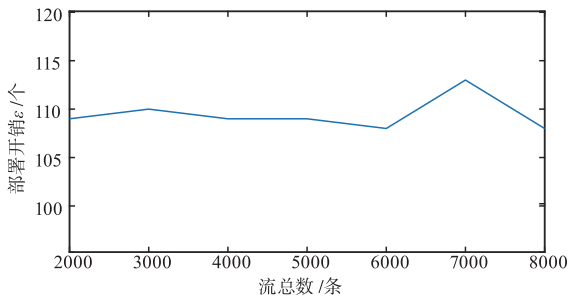


图5　流总数变化对部署开销ε影响

最后,对本文提出的算法以及 SM 算法和 MCFR 算法的 Packer-In 消息丢包率进行仿真对比分析,仿真结果如图 6 所示. 根据仿真结果可以看到,SM 算法在流总数大于 4000 后丢包率会急剧上升,相比之下 MCFR 算法丢包率在流总数 6000 以下时较小,高于 6000 后也会持续增加,但是总体增伤速度小于 SM 算法,相比于 SM 算法最多可减少 50% 丢包率. 而本文的算法可以在 MCFR 算法的基础上进一步降低丢包率. 可以看到在流总量达到 10000 时,本文的 Packet-In 消息丢包率依然几乎为 0,随着流总数增加,Packet-In 消息丢包率会有所增加,但是到 14000 流总数时,丢包率仍然低于 10%,相比 SM 算法最多可以降低超过 80% 的丢包率.

经过本章对真实网络拓扑进行仿真模拟,并从多个指标对本文的方法和 SM 算法以及 MCFR 算法进行比较,可以看到本文提出的算法在最大控制链路上行负载,负载均衡因子和 Packet-In 消息丢包率等指标中都取得

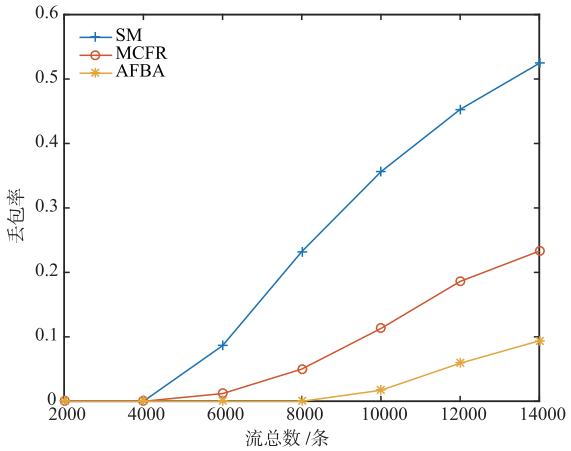


图6 三种算法不同流总数时Packet-In丢包率比较

了更优的效果,相比于 SM 算法,本文提出的算法平均可以降低 67% 的最大控制链路上行负载和超过 80% 的 Packet-In 消息丢包率,相比于 MCFR 算法,也能够降低近 33% 的最大上行控制链路上行负载和近 50% 的 Packet-In 消息丢包率,且拥有最接近于 1 的负载均衡因子,即在三种算法中,本文提出的算法负载最均衡,可以尽可能地利用每一个交换机,进而也可以增加系统的吞吐率. 同时在开销方面,本文提出的算法并没有花费很多额外开销,只需要占用每个交换机中几个额外的流表项,不会有其他额外的部署开销,具有极低的部署成本. 可以看出,本文的算法能够在不额外增加系统部署成本的条件下很好地解决上行控制链路瓶颈问题.

4.3 平台实验

最后将上述上行控制链路负载均衡问题的解决方法进行了平台部署与试验,为方便进行对比实验,选择了与文献[14]相同的实验结构,如图 7. 选用了一台处理器为 i7-3770K,内存为 16G 的主机,利用 Ryu 开源控制器软件作为系统的控制平面,用了六台支持 OpenFlow 的 H3C S5120-28SC-HI 商用交换机作为数据平面,并连接了四台终端,利用安装在终端上的流量模拟器模拟网络流量.

在实验时,我们让终端 h_1 分别向 h_3 和 h_4 发送 $2\times m$ 条流量,h_2 分别向 h_3 和 h_4 发送 m 条流量. 根据文献[7]中测量结果,我们规定上行控制链路容量为 150 条/s. 由于在系统满载之前可以及时处理新流,因此我们取重定向阈值等于上行控制链路容量,即 $R=150$ 条/s,这样可以降低系统部署开销,同时仍然可以保证系统性能. 依然是对本文提出的算法和 SM 算法以及 MCFR 算法进行了对比试验.

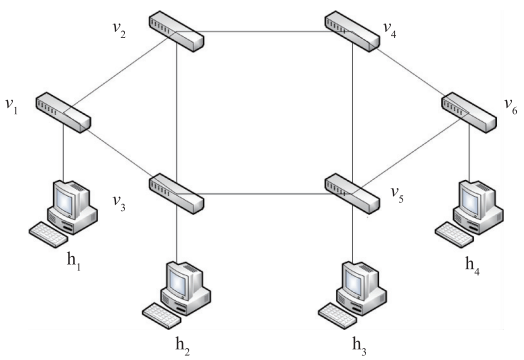
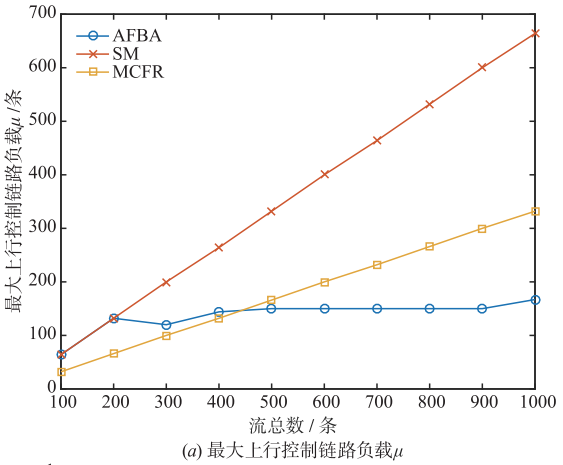
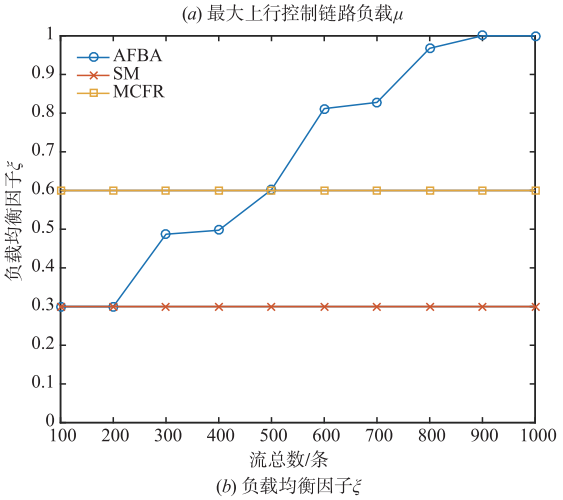


图7 实验平台拓扑图

首先衡量了几种方法的最大上行链路负载,结果如图 8 所示. 取了系统中总流量从 100 到 1000 的共 10 种情况. 当总流量超过 900 时,即使流量完全分摊到每个交换机,仍然会造成所有交换机超载,属于很极端的一种情况. 可以看到,对于 SM 算法,当系统中流量超过 300 后,就已经开始面临很严峻的负载过高问题,而对于 MCFR 算法,其选择将 v1 中的一条宏流重定向到 v2,将 v3 中一条宏流重定向到 v5,这样就可以将最大上行链路负载降低到 SM 算法的 50%,但是这种方式受限于宏流大小,因此若一个终端短时间内向另一个终端发送了大量数据的话,依然会造成严重的上行控制链路拥塞问题. 而本文提出的方法,由于并没有使用宏流的定义,而只是取决于流量的总数与设定的阈值,因此可



(*a*) 最大上行控制链路负载μ



(*b*) 负载均衡因子ξ

图8 三种算法性能比较

以看出,当系统中没有超出阈值的交换机时,重定向系统并不会启动,可以避免占用额外的计算资源与流表资源. 而当超过阈值后,本文的算法可以通过重定向将系统中所有交换机的负载均控制在阈值以下,直到每个交换机中负载均超出阈值后,系统中才会出现超出阈值的交换机,且超出阈值后,交换机中流量数仍然几乎为总流量数的平均分摊数量,远远小于其他两种方法的最大控制链路上行负载.

而对于控制链路负载均衡因子,如图 8(*b*)所示,可以看出对于这种固定的流量模型,SM 算法与 MCFR 算法负载均衡程度固定,而本文提出的算法负载均衡度会随着流总数增加而增加,即超出负载的流量越多,本文重定向后的负载均衡率越高.

图 9 给出了本文算法的流表项开销,可以看到,即使在流表总数到达 1000 时,本算法的流表项开销也只有 7,即只需要向 TCAM 中安装 7 条流表规则,平均下来一个 TCAM 中平均只需要安装一条流表项,相对于一个 TCAM 有 200 条流表项的容量来说,开销不到 1%,属于可以接受的范围内. 因此可以看出本算法在解决上行控制链路瓶颈问题时具有很好的效果.

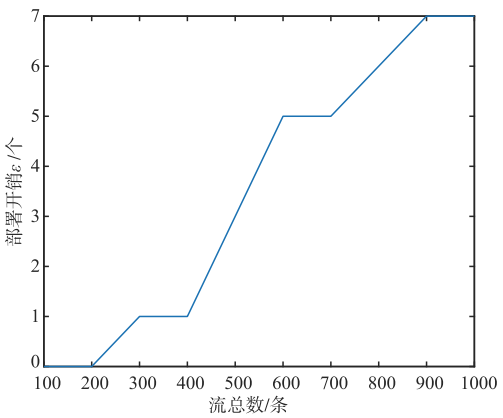


图9　部署开销

5　结束语

本文针对 SDN 系统中存在的上行控制链路瓶颈问题,同时考虑 Packet-In 消息上行链路负载,OpenFlow 交换机流表约束和数据链路约束,提出了一种改进算法,并在 SDN 实验平台和更广泛的仿真平台上进行了验证,与传统方法相比可以减少 80% 以上的上行控制链路负载和系统丢包率,相比 MCFR 算法能够减少近 50% 的上行控制链路负载和系统丢包率,且不需要部署额外的设备,只需要占用很有限的现有资源.

参考文献

[1] Wu J. Thoughts on the development of novel network technology[J]. Science China Information Sciences, 2018, 61(10):101301. DOI:10.1007/s11432-018-9456-x.

[2] 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. DOI:10.1145/1355734.1355746.

[3] Kannan K, Banerjee S. Compact TCAM:Flow entry compaction in TCAM for power aware SDN[A]. International Conference on Distributed Computing and Networking[C]. Berlin, Heidelberg, GER:Springer, 2013. 439 – 444. DOI:10.1007/978-3-642-35668-1_32.

[4] Luo S, Yu H. Fast incremental flow table aggregation in SDN[A]. 2014 23rd International Conference on Computer Communication and Networks (ICCCN)[C]. Shanghai, China:IEEE, 2014. 1 – 8. DOI:10.1109/ICCCN.2014.6911781.

[5] Gude N, KoponenT, Pettit J, et al. NOX:Towards an operating system for networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(3):105 – 110. DOI:10.1145/1384609.1384625.

[6] Cai Z, Cox A L, Ng T S. Maestro:Balancing Fairness, Latencyand Throughput in the Openflow Control Plane[R]. Rice University Technical Report TR11-07, Berlin, GER:Springer, 2011.

[7] Wang A, Guo Y, Hao F, et al. Scotch:Elastically scaling up sdn control-plane using vswitch based overlay[A]. Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies[C]. Sydney, AUS:ACM, 2014. 403 – 414. DOI:10.1145/2674005.2675002.

[8] 胡涛,张建辉,邬江,等. SDN 中基于分布式决策的控制器负载均衡机制[J]. 电子学报, 2018, 46(10):2316 – 2324.
Hu T, Zhang J H, Wu J, et al. Controller load balancing mechanism based on distributed policy in SDN[J]. Acta Electronica Sinica, 2018, 46(10):2316 – 2324. (in Chinese)

[9] 姚蓝,胡涛,伊鹏,等. SDN 中基于效能优化的交换机动态迁移策略[J]. 电子学报, 2019, 47(7):1482 – 1489.
Yao L, Hu T, Yi P, et al. Switch dynamic migration strategy based on efficiency optimization in SDN[J]. Acta Electronica Sinica, 2019, 47(7):1482 – 1489. (in Chinese)

[10] Song P, Liu Y, Liu T, et al. Flow Stealer:Lightweight load balancing by stealing, flows in distributed SDN controllers[J]. Science China Information Sciences, 2017, 60(3):032202.

[11] 张栋,郭俊杰,吴春明. 层次型多中心的 SDN 控制器部署[J]. 电子学报, 2017, 45(3):680 – 686.
Zhang D, Guo J J, Wu C M. Controller placement based on hierarchical multi-center SDN[J]. Acta Electronica Sinica, 2017, 45(3):680 – 686. (in Chinese)

[12] Curtis A R,Mogul J C,Tourrilhes J,et al. DevoFlow:Scaling flow management for high-performance networks[J]. ACM SIGCOMM Computer Communication Review,2011,41(4):254 – 265. DOI:10.1145/2018436.2018466.

[13] Huang D Y,Yocum K,Snoeren A C. High-fidelity switch models for software-defined network emulation[A]. Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking[C]. Hong Kong,China:ACM,2013. 43 – 48. DOI:10.1145/2491185.2491188.

[14] Wang P,Xu H,Huang L,et al. Control link load balancing and low delay route deployment for software defined networks[J]. IEEE Journal on Selected Areas in Communications,2017,35(11):2446 – 2456. DOI:10.1109/JSAC.2017.2760187.

[15] The Network Topology Fromthe Monash University[EB/OL]. http://www.eese.monash.edu.au/twiki/bin/view/InFocus/LargePacket-switchingNetworkTopologies.2018.

[16] Mininet:An Instant Virtual Networkon Your Laptop(or other PC)[EB/OL]. http://mininet.org/.2014.

作者简介

王莅晟 男,1995 年生,甘肃人. 中国人民解放军战略支援部队信息工程大学研究生,研究方向为新型网络体系结构.

伊 鹏 男,1977 年 11 月生. 博士,研究员,博士生导师,国家数字交换系统工程技术研究中心宽带网络研发部部长,河南省网络空间拟态防御重点实验室主任,长期从事新型网络体系结构、网络安全管控和主动防御技术研究.

胡 涛(通信作者) 男,1993 年 8 月生,陕西武功人. 现为国家数字交换系统工程技术研究中心博士研究生,主要研究方向为新型网络体系结构、网络安全.
E-mail:hutaondsc@163.com