SDN 数据中心网络基于流表项转换的流表调度优化

唐 莞[1],王敢甫[2],吴京京[1],王文涛[1]

(1 中南民族大学 计算机科学学院,武汉 430074;2 武汉烽火信息集成技术有限公司 IT 事业部,武汉 430074)

摘 要 针对基于软件定义网络(SDN)架构的数据中心网络中,SDN 交换机流表资源的有限性导致的流表溢出或控制器拥塞等问题,引入空闲流表资源代价的概念描述了网络资源的利用率,并分析了空闲流表资源量与重复下发的流表项数量之间的关系,提出了一个基于流表超时机制的流表调度策略,依据流表项生存时间和匹配计数来进行静态流表项和动态流表项之间的实时转换. 在 Fat-tree 拓扑 SDN 数据中心网络仿真实验中,对该机制对流表资源优化的有效性进行了验证.

关键词 软件定义网络;数据中心网络;资源代价;流表调度

中图分类号 TP393 **文献标识码** A **文章编号** 1672-4321(2017)03-0111-07

Flowtable Scheduling Optimization based on Flowentry Conversion in SDN-based Datacenter Networks

Tang Wan, *Wang Ganfu*, *Wu Jingjing*, *Wang Wentao*

(1 School of Computer Science, South-Central University for Nationalities, Wuhan 430074, China;
2 IT Business Unit, Wuhan FiberHome Informtion Integration Technologies Co., Ltd., Wuhan 430074, China)

Abstract The resource limitation of the flowtable in OpenFlow switches causes the problem of flowtable overflow or controller congestion in the SDN-based datacenter network. To address the issue, in this paper, the concept of Free Entries Cost is introduced for describing the utilization efficiency of network resources, and the relationship between the numbers of free flowtable resources and that of the flowentries being forwarded repeatedly is analyzed. Then, a flowtable scheduling strategy based on the flowtable timeout mechanism is proposed, which can do real-time conversion between static and dynamic flowentries according to the survival time and the count of the flowentry matching. Via the simulation platform of SDN-based datacenter network with the Fat-tree topology, the availability of the proposed mechanism for optimizing the usage of flowtables is verified.

Keywords software-defined networking; datacenter network; resource cost; flowtable scheduling

近年来,软件定义网络(SDN)的发展与数据中心的联系越来越紧密[1]. SDN 通过分离控制平面与数据平面及提供控制层北向编程接口(API),给云数据中心网络的管理提供了新的手段和途径. 数据中心网络每秒会产生几十万条流甚至更多,数据中心内的流量占数据中心总流量的 80% 左右[2],数据中心流量已经由“南北”为主转变为“东西”为主. 对于 OpenFlow 交换机,SDN 网络提供基于流的细粒度控制,网络中的每一条数据流可能对应不止一条流表项,加上不同服务器之间丰富且频繁的交互,需要控制器下发大量的流表项来维持通信. 当突发流很多时,容易导致控制器过载. 并且流表大小受 TCAM(Ternary Content Addressable Memory)限制,过多的流表项可能会由于流表资源不足引起流表溢出. 此外,在新的流到达时必然会产生新旧流表项的替换,延长了流表项的下发所需时间,导致更长的网络延时.

为了解决流表容量有限而导致的控制器拥塞和流表资源溢出问题,OpenFlow 协议引入了多流表形

收稿日期 2017-06-08

作者简介 唐 莞(1974-),女,教授,博士,研究方向:光/无线网络协议、软件定义网络、网络安全,E-mail: tangwan@scuec.edu.cn

基金项目 国家自然科学基金资助项目(61103248)

式,通过基于流水线的流表查找来解决指数级增加的流表项数. 学术界则主要从几个方面来解决:(1)针对SDN架构存在的缺陷,扩展SDN架构的软件和硬件功能[3];(2)认为流表项的数据结构过于繁杂是造成流表资源不足主要原因,从而通过改进流表项数据结构来解决[4];(3)使用统计或数学模型来预测网络资源使用[5];(4)根据网络资源使用情况,改进流表超时机制,动态设置流表项的超时时间[6].

在当前通过停滞等待超时时间idle-timeout(简称停滞超时时间或超时时间)调控流表资源的工作中,还没有对控制器资源和流表资源与idle-timeout关系的深入研究与探讨,且大部分侧重于如何减少流表资源的使用来保证其不溢出. 其实,要保证流表不溢出,只需要设置很小的超时时间,但考虑在减少流表资源使用时对控制器的负载增加的研究也不多见. 因此,本文综合这两个方面提出"空闲流表资源代价"的概念,用来评价相关算法或策略对idle-timeout的调控是否有效. 此外,在不同的数据中心网络流量特性下,研究流表项的超时设置对流表资源和控制器资源的影响,提出流表项的动静态转换机制,从而改善流表溢出时控制器资源与流表资源之间的调度问题.

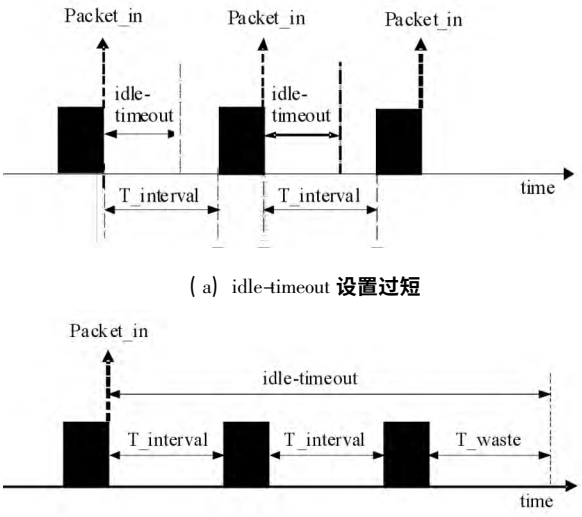
1 停滞超时时间对流表资源的影响

1.1 流表资源问题描述

当前流表项在流表中存活的时间一般为固定值或永久存在,东西向流量中的业务种类丰富,每一个流产生周期并不一致,如图1,流表项的idle-timeout即为流匹配完毕后该流表项能继续存在于流表中的时间. 如果idle-timeout设置过短,对于数据间隔时间(T_interval)比较长的流,就会导致交换机频繁向控制器发送Packet_in消息,控制器会频繁重复下发此流表项,浪费控制器资源;如果idle-timeout设置过长,对于数据间隔时间比较短的流,流表项可能在流匹配结束后在流表中继续存在,成为无效流表项,占用流表资源,并且由于流表的匹配机制,无效流表项会延长流匹配的过程. 所以,为了减小此类问题所产生的影响,平衡控制器资源与流表资源的使用,必须要对SDN网络的流表资源进行优化.

1.2 空闲流表资源代价

SDN网络流表项的idle-timeout设置是否合理,取决于是否能最大限度地节省流表资源,同时没有



(a) idle-timeout设置过短

(b) idle-timeout设置过长

图1 停滞等待超时时间设置示例

Fig. 1 Examples for idle-timeout setting

带来更多的控制器资源消耗,这里所说的控制器资源的消耗指重复流表项的下发. 如果流表资源的不足,加上控制器需要对同一条流表项不断地重新下发,必然大量消耗控制器的资源.

空闲流表资源是指交换机中节约的流表资源(一般以流表项的条目数多少来衡量). 本文提出"空闲流表资源代价",表示一条空闲的流表项导致被重复下发流表项的次数,用于评估流表资源和控制器资源的使用效率,进而判断idle-timeout的设置是否合理. 为了更简单讨论问题,下面对问题进行抽象,建立一个简单数学模型来描述空闲流表项资源与重复流表项数量之间的联系.

对一个SDN网络中的流量作如下假设(图示见图2):

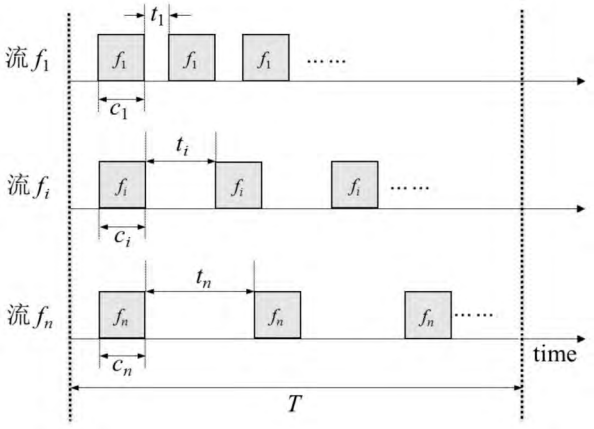


图2 流的特征参数示意

Fig. 2 Characteristic parameters of flows

① 取某一段通信时长为T,设这段时间内有n

条不同的流$\{f_1, f_2, \cdots, f_n\}$传输,对应着 n 条流表项;

② 网络中的每一条流具有这样的特征参数,如流f_i:$\{t_i, c_i\}$,$i \in \{1, \cdots, n\}$,其中t_i为流f_i的时间间隔特征,表示流f_i每次消失后到再次出现所经历的时间段,c_i表示f_i每次出现后所持续的时间;

③ 所有流的时间间隔特征为$\{t_1, t_2, \cdots, t_n\}$(其中$t_1 \leqslant t_2 \leqslant \cdots \leqslant t_n$),与之对应的流持续时间为$\{c_1, c_2, \cdots, c_n\}$;

④ 控制器默认设置的 idle-timeout 为T_{out},$t_{i-1} \leqslant T_{out} \leqslant t_i$,$i \in \{1, \cdots, n\}$.

可得,在时间 T 内重复下发的流表项数量为:

$$Count_{re\_flowenrice} = \sum_{j=i}^{n} \left[\frac{T}{t_j + c_j} \right]. \quad (1)$$

所有下发的流表项在交换机上存在的总时间为:

$$TotalTime = (i-1)T + \sum_{j=i}^{n} \frac{T(T_{out} + c_j)}{t_j + c_j}.$$

把 $TotalTime$ 归一化到流表资源的占用数量上,并计算出空闲的流表项资源数量:

$$FreeEntries = n - \frac{TotalTime}{T},$$

化简可得:

$$FreeEntries = \sum_{j=i}^{n} \left(1 + \frac{T_{out} + c_j}{t_j + c_j}\right). \quad (2)$$

根据定义,空闲流表资源代价可表示为:

$$FreeEntriesCost = \frac{Count_{re\_flowenries}}{FreeEntries}. \quad (3)$$

由式(1) 和式(2) 可以看出,重复下发的流表项数量 $Countre\_flowentries$ 与空闲的流表项资源数量 $FreeEntries$ 都与超时时间 $Tout$ 密切相关.

1.3　停滞超时时间的合理设置

本节将通过实验来分析不同流量场景下,空闲流表资源代价与超时时间 idle-timeout 的关系.

1.3.1　仿真网络及相关设置

本文采用仿真软件搭建网络实验环境,在一台 PC 机上的 VMware 工作站中安装 Ubantu14.04LTS 虚拟机,并在其中安装 OpenDaylight 和 Mininet. SDN 控制器使用 OpenDayLight 的 Lithium-SR2 版本[7],OpenFlow 交换机以及服务器结点(本文也称之为主机结点)的模拟用 Mininet 实现. 使用 Mininet 模拟的交换机是 OpenvSwitch2.3.1 版本,北向接口采用 OpenFlow1.3.0 协议. 详细软硬件环境分别见表 1 和表 2.

表 1　硬件环境

Tab.1　Hardware environment

设备	相关配置信息
PC 机	CPU: Intel(R) Core(TM) i7-6700HQ 2.60GHz 八核　内存: 16.0GB　硬盘容量: 1128GB　操作系统: Win7 64 位
VMware Workstation 版本	ProVMware® Workstation 12 Pro 12.1.0 build-3272444
Ubantu14.04LTS 虚拟机资源	处理器数量: 1　处理器内核数量: 8　内存: 14264MB 硬盘容量: 128GB

表 2　软件仿真工具

Tab.2　Simulation software

对象	软件名	版本	功能描述
SDN 控制器	OpenDayLight	Lithium-SR2	开源 SDN 控制器,与网络中各交换机相连,对网络进行资源调度和集中控制
虚拟交换机	OpenvSwitch	2.3.1	虚拟交换机软件,支持 OpenFlow 协议,根据流表控制数据包的转发
仿真拓扑	Mininet	2.2.1	模拟虚拟的交换机以及主机结点,构造网络拓扑结构

实验网络拓扑采用数据中心网络研究中常用、目前工业界应用最广泛的 Fat-tree[8-10] 结构. 如图 3,每个交换机都连接到 OpenDayLight 控制器,网络中交换机分为核心层、汇聚层和边缘层. 虚线框内由汇聚层和边缘层的交换机组成一个 Pod,共有 4 个 Pod,每个边缘交换机连接 2 台服务器. 在本文实验中进行流表资源调控的交换机是核心交换机 S4,流表容量为 160 条流表项.

为了流量模拟的准确性和全面性,本文参考文献[11]模拟了数据中心两种常见的流量模型: 平均概率模型和选择概率模型. 平均概率模型是指主机向网络中的另一任意主机以等概率发送数据包,不同主机之间为随机通信,从而使流量在网络间均匀分布,这种流量模型在实际中并不多,主要是为了理论实验; 选择概率模型是指在网络中少数的主机间会有比较大的概率互相发送数据包,而其他主机之

间则是随机通信,使网络流具有明显的局部性,这种流量模型则是为了模拟实际的数据中心流量模型.

本文实验只关注流的数据间隔时间以及在不同的流量模型和不同 idle-timeout 的设置下,虚拟交换机 OpenvSwitch 的流表项数量变化. 因此,本文中虚拟主机之间通过 Iperf[12] 产生带宽 1 Mbps 的流量,每 0.2 s 会根据指定的流量模型产生并发送一条流数据,每次流持续发送时间为 6 s.

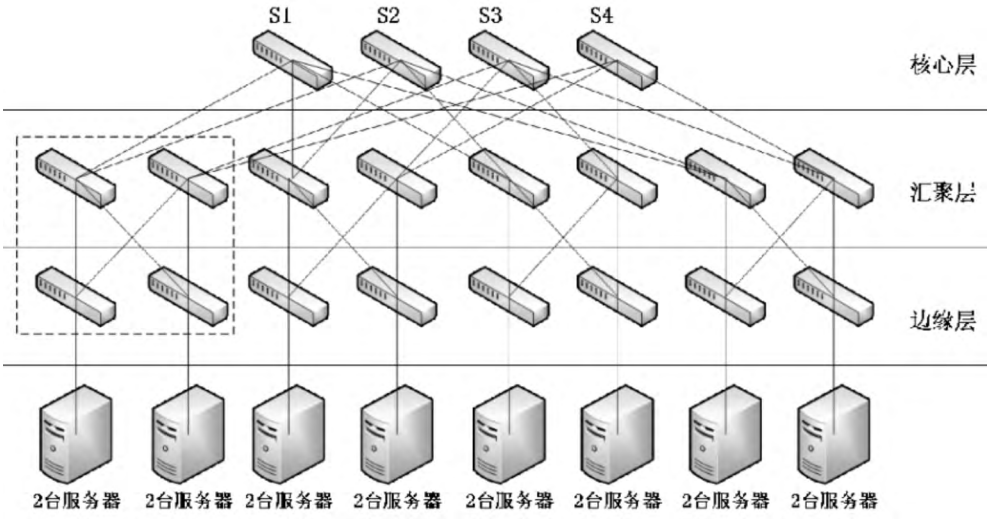


图3 仿真实验网络的 Fat-tree 架构

Fig. 3 Fat-tree Architecture of the simulation network

1.3.2 实验结果与分析

首先基于实验来分析停滞等待超时时间 idle-timeout 对流表资源的影响. 控制器下发具有默认固定 idle-timeout 值的流表项,通信时长 T 为 360 s,每隔 1 s 记录交换机上流表项的数量,最后取流表项数量的平均值来表示这段时间内流表资源的使用情况,称为平均流表项数量,实验结果在图 4 ~ 6 中给出.

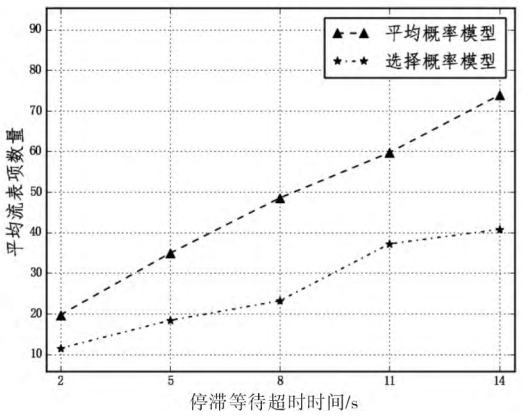


图4 不同流量模型下流表资源占用统计

Fig. 4 Flowtable resource overhead under different traffic models

图 4 中横坐标表示交换机流表的 idle-timeout 值(说明: 后面其他结果图中的横坐标均同样含义),纵坐标表示在没有任何其他资源调度程序调控时,交换机上在这段时间内的平均流表项数量,即流表资源的使用情况. 可以看到: (1) 两种流量模型下流表资源的使用量随 idle-timeout 的增加而增加; (2) 不同特性的流量占用流表情况不同,其中主机之间通信完全随机的平均概率模型下,对流表资源的占用会更高.

因为控制器最主要的功能就是根据实时网络信息进行计算,产生并下发针对某一条流的流表项,所以本文用控制器重复下发的流表项数量来表示对控制器资源的影响程度. 从 idle-timeout 的设置上来看,只要某一条流的数据间隔时间大于 idle-timeout 的设置,就必然会带来针对此流表项的重复下发.

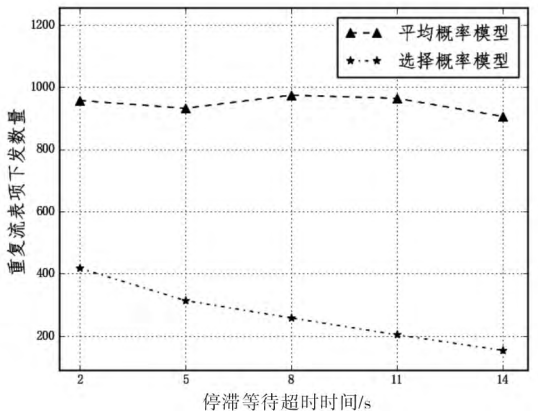


图5 不同流量模型下流表项重复下发数量统计

Fig. 5 Numbers of repeated-forwarding flowentries under different traffic models

图 5 通过流表项重复下发的情况,反映 idle-timeout 对控制器资源的影响. 在选择概率模型下,

重复下发的流表项数量是随着交换机流表 idle-timeout 的增加而逐渐减少; 而在平均概率模型下,在一定 idle-timeout 设置的时间范围内,重复下发的流表项数量随着 idle-timeout 的增加变化不大. 但是在实际的数据中心网络中,由于流量很大,并且时有不同,所有流量全部产生的最短时间很难统计. 因此,idle-timeout 的时间设置也不可能那么长,所以本实验的设计仍然符合数据中心网络的特点.

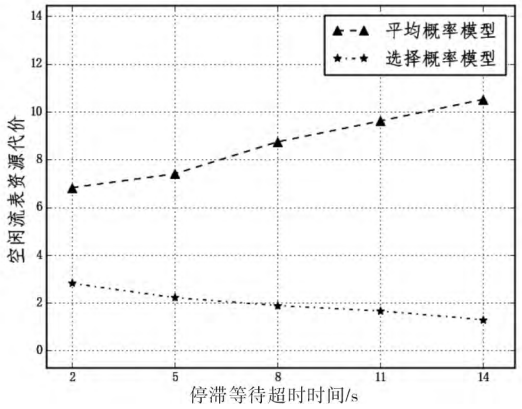


图 6　不同流量模型下的空闲流表资源利用情况

Fig. 6　Usage of free flowtable resource under different traffic models

图 6 展示的是基于空闲流表资源代价来评价交换机流表的 idle-timeout 对空闲流表利用的影响. 在平均概率模型下,空闲流表资源代价基本分布在 6 到 11 之间,呈现递增趋势. 在选择概率模型下,空闲流表资源代价总体上比平均概率模型的要小,且随着 idle-timeout 的增加而呈现递减的趋势. 实验结果表明,为了充分利用空闲流表资源,不同流量模型下对 idle-timeout 的设置应该不同: 在网络流量具有平均概率特性时,idle-timeout 设置一个低值; 而当流量特性是选择概率时,则应该设置较高的 idle-timeout.

在上述实验中的平均概率模型下突发流全部是随机的,而选择概率模型下网络中的流大部分为频繁流,本文注意到: 对随机性的流设置较小的 idle-timeout 值,对持续性的流设置较长的 idle-timeout 值,都能降低空闲流表资源代价,这也是本文后面流表项的动静态转换机制提出的基础.

2　动态流表项与静态流表项的相互转换机制

实际的 SDN 交换机上的流表项可以永久存在,也可以设置其超时时间. 基于前面实验数据的分析结果,本节提出一个动态和静态流表项相互转换机制(Exchange of Static and Dynamic flowentries, ESD) ,对流表资源的使用进行优化调度.

2.1　动静态流表项转换机制

OpenFlow 协议采用两种方式设定流表项的存活时间,一种是停滞等待超时时间 idle-timeout 方式,表示在此超时时间内如果没有任何数据包匹配,则在超时时间到达时自动删除流表项,若流表项的 idle-timeout 值设置为零,则表示此值域对流表项的存活时间不起作用,实际上对应的是同一条流的数据间隔时间. 另一种是硬超时时间 hard-timeout 方式,表示在此超时时间内不管有没有数据包匹配,只要硬超时时间到达,则自动删除流表项,若流表项的 hard-timeout 值设置为零,则表示此值域对流表项的存活时间不起作用,实际上对应的是同一条流的数据持续时间. 因为极易产生流匹配过程中流表项丢失的问题,hard-timeout 方式并不适合进行调控. 因此,本文采用设置 idle-timeout 方式,根据 SDN 数据中心的流量特性来动态设定合理的 idle-timeout 的值.

本文中所指的动态流表项是 idle-timeout 值不为零而 hard-timeout 为零的流表项,此情况下流表项存活时间(Duration) 只受 idle-timeout 值影响; 静态流表项指的是 idle-timeout 和 hard-timeout 均为零的流表项,此种情况下流表项的存活时间为无穷大. 在 ESD 机制中,对于持续性的流,设置这些流对应的流表项为永久存在的静态流表项,以减少流表项的重复下发; 对于间歇或偶尔出现的流的流表项,则设置为有超时时间的动态流表项,从而减少不必要的流表占用. 最终通过这种动态流表项与静态流表项的互相转化,来降低空闲流表资源代价.

在 SDN 应用层面,ESD 机制使用控制器 OpenDayLight 的北向 REST API,实时获取流表项的匹配包数量、生存时间等统计信息,动态调整流表项的 idle-timeout 值,把流表项区分为动静态流表项. 静态流表项是表示在交换机上一直存在,但在触发某一条件时可被转变为动态的流表项. 动态流表项的 idle-timeout 值会设置为用户允许的值,并在超时时间达到后,如果没有被匹配则会被删除. ESD 具体流程如图 7,其中包匹配状况的监测时间周期 idle-timeout′等于 idle-timeout 和流持续时间之和. ESD 只专注于 idle-timeout 的值,对 hard-timeout 一直设为 0(表示此域值无效) .

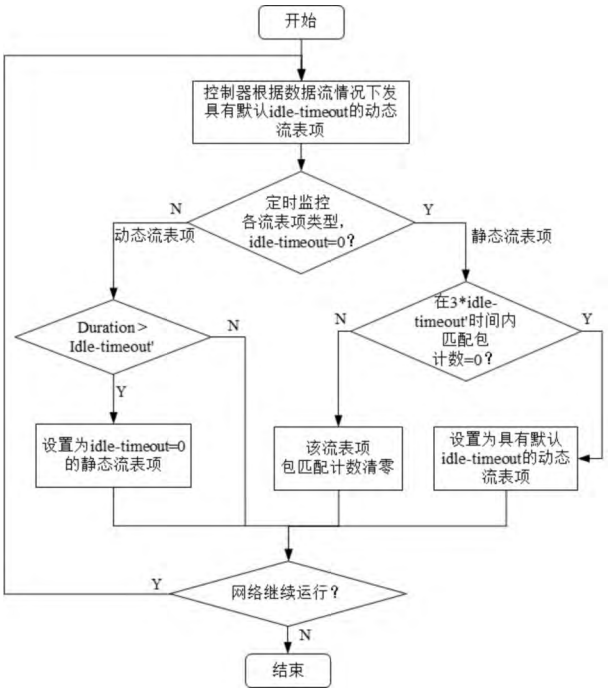


图7 ESD机制工作流程

Fig. 7 Working process of ESD mechanism

当一条静态流表项在允许的时间内没有匹配到任何数据包，那么就会被转化为动态流表项. 实际上静态流表项对应的就是网络中长时间持续的流，比如频繁通信的某两台主机，而动态流表项对应的是网络中的随机流量. ESD会对误转为静态的流表项重新转换为动态流表项，实际上是延长了可能是频繁流的流表项在流表上的存在时间，以此换来流表项重复下发数量的下降.

2.2 ESD有效性验证

为了验证ESD的有效性，是否能更好地协调流表资源和控制器资源即具有更低的流表空闲资源代价，本文基于1.3.1节的实验环境进行ESD与没有调控情况的对比试验，分别在平均概率模型和选择概率模型下比较了平均流表数量、流表项重复下发数量、空闲流表资源代价的变化，各项结果在图8～11中分别给出.

图8和图9中，当流量为平均概率特性时，ESD的重复流表项下发的数量和空闲流表资源代价比没有ESD调控时要低很多，表示ESD减少了多余流表项的重发，从而降低对流表资源和控制器资源的消耗. 图10和图11中，选择概率模型的情况与平均概率模型的相似. 在采用ESD机制后，重复流表项下发的数量和空闲流表资源代价降低很多而且相对平稳. 对于流量具有平均概率特性的数据中心网络，采用ESD机制时设置较小的idle-timeout，而对于选择概率情况则设置较大的idle-timeout，将更好保证资源的使用.

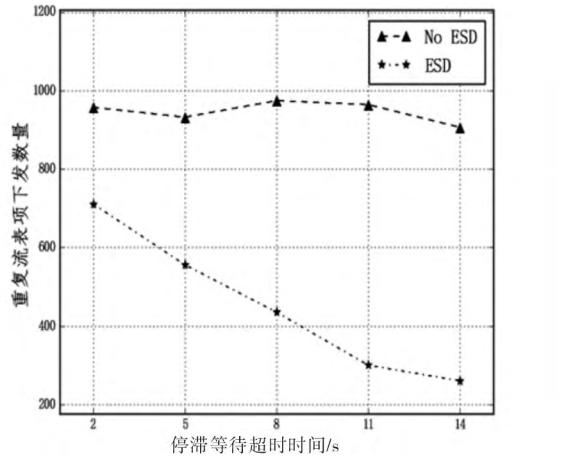


图8 流表项重复下发数量比较(平均概率流量)

Fig. 8 Comparison of the numbers of repeated-forwarding flowentries (average-probability traffic)

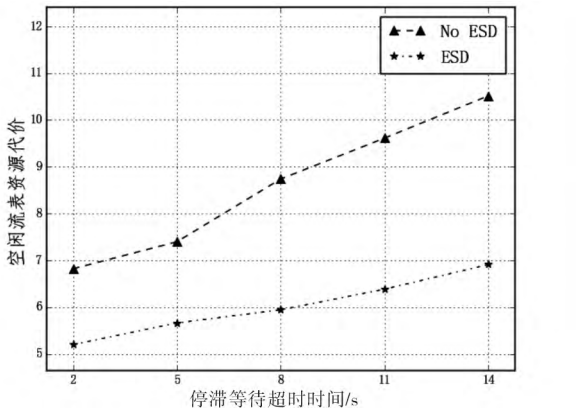


图9 空闲流表资源代价的比较(平均概率模型)

Fig. 9 Comparison of Free Entries Cost (average-probability traffic)

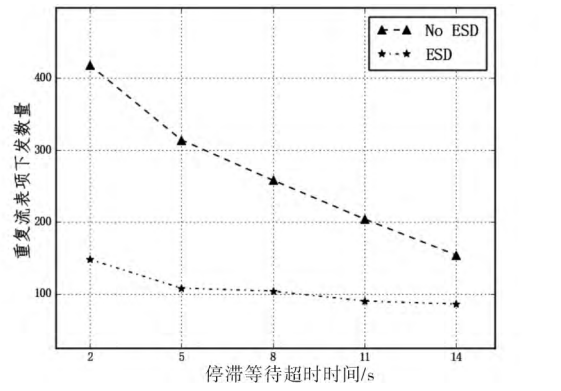


图10 流表项重复下发数量比较(选择概率模型)

Fig. 10 Comparison of the numbers of repeated-forwarding flowentries (selective-probability traffic)

综合以上结果可知,不管网络的流量是具有平均概率特性还是选择概率特性,根据空闲流表资源代价的定义,ESD 机制都能有效地降低流表空闲资源代价,对协调流表资源和控制器资源是有效的.

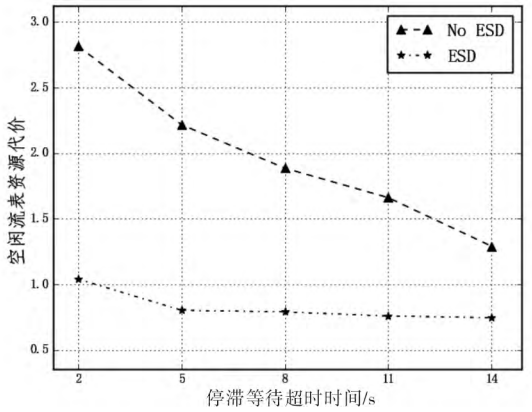


图 11　空闲流表资源代价的比较(选择概率模型)

Fig. 11　Comparison of Free Entries Cost (selective-probability traffic)

3　总结

本文研究如何在 SDN 数据中心网络中合理设置停滞等待超时时间 idle-timeout,以达到合理使用控制器资源和交换机的流表资源的问题. 具体做了以下工作: 基于不同的流量模型,分析并研究了 SDN 数据中心网络中 OpenFlow 交换机的流表项数和控制器资源使用与 idle-timeout 值的关系; 引出空闲流表资源代价的概念,基于仿真实验结果,发现一定程度增加持续性流的 idle-timeout 值和减少随机性流的 idle-timeout 值,会降低空闲流表资源代价; 最后提出了一个基于动静态流表项转换的流表调度机制 ESD,对流表资源和控制器资源的使用进行优化,并对其可行性与有效性进行了验证. 下一步工作将研究 ESD 中 idle-timeout 值的设置方案,并进行 ESD 对其他拓扑结构的 SDN 数据中心网络的有效性验证.

参　考　文　献

[1] 王蒙蒙, 刘建伟, 陈　杰,等. 软件定义网络: 安全模型、机制及研究进展 [J]. 软件学报, 2016, 27(4): 969-992.

[2] 邓　罡, 龚正虎, 王　宏. 现代数据中心网络特征研究 [J]. 计算机研究与发展, 2014, 51(2): 395-407.

[3] Yu M, Rexford J, Freedman M J, et al. Scalable flow-based networking with DIFANE [J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 351-362.

[4] Banerjee S, and Kannan K. Tag-In-Tag: Efficient flow table management in SDN switches [C] // IEEE. International Conference on Network and Service Management. Rio de Janeiro: IEEE, 2014: 109-117.

[5] Zhou B, Gao W, Wu C, et al. AdaFlow: adaptive control to improve availability of openflow forwarding for burst quantity of flows [C] // V. C. M. Leung et. al. (Eds.) TridentCom 2014, LNICST 137. Guangzhou: Springer, 2014: 406-415.

[6] Zhu H, Fan H, Luo X, et al. Intelligent timeout master: dynamic timeout for SDN-based data centers [C] // IEEE. IFIP/IEEE International Symposium on Integrated Network Management (IM). Ottawa: IEEE, 2015: 734-737

[7] Linux Foundation. OpenDayLight [EB/OL]. [2016-02-20]. http://www.opendaylight.org/.

[8] Al-Fares M, Loukissas A, and Vahdat A. A scalable, commodity data center network architecture [J]. ACM SIGCOMM Computer Communication Review, 2008, 38(4): 63-74.

[9] 丁泽柳. 数据中心网络拓扑探讨 [J]. 中兴通讯技术, 2012, 18(4): 7-10.

[10] 赵　哲, 胡莹莹. 如何在三种典型的数据中心网络拓扑架构中进行取舍 [J]. 数字通信世界, 2016, (5): 56-58.

[11] 张　歌. 两种常用流量模型运用 mininet 的实现 [EB/OL]. [2015-04-28]. http://www.sdnlab.com/11079.html.

[12] NLANR/DAST. Iperf [EB/OL]. [2016-02-20]. https://iperf.fr/.