基于负载通告的 SDN 多控制器负载均衡机制

王 颖 余金科* 裴科科 邱雪松
(北京邮电大学网络与交换技术国家重点实验室 北京 100876)

摘 要：多控制器负载均衡是 SDN 网络部署研究中关注的问题之一。该文针对多控制器间负载均衡的时间效率问题，提出一种基于负载通告策略的负载均衡机制(LILB)。该机制包括负载测量、负载通告、均衡决策和交换机迁移 4 个核心功能组件。借助于负载通告的能力，每个控制器可以在过载后无需收集其他控制器的负载信息而尽快完成均衡决策。为了减少负载通告带来的通信负荷和处理负荷，该文提出一个抑制算法来降低负载通告的频率。此外，该文还提出了最重过载控制器、迁移交换机和目标控制器的决策方法，以及目标控制器接受迁移请求的判定策略来避免控制器的负载震荡；并为支持交换机迁移过程中控制器角色的平滑切换设计了一种交换机迁移的消息交互机制。最后，在基于 Floodlight 和 Mininet 的实验环境中验证了所提出方法的有效性。

关键词：软件定义网络；控制器负载均衡；负载通告；均衡决策
中图分类号： TP393 **文献标识码**： A **文章编号**：1009-5896(2017)11-2733-08
DOI: 10.11999/JEIT161054

A Load Informing Based Load Balancing Mechanism for Multiple Controllers in SDN

WANG Ying YU Jinke PEI Keke QIU Xuesong
(*State Key Laboratory of Networking and Switching*, *Beijing University of Posts and Telecommunications*, *Beijing* 100876, *China*)

Abstract: Load balancing of multiple controllers is currently a focused issue in the research area of Software Defined Networking (SDN) deployment. Considering the issue of time efficiency of load balancing, this paper proposes a Load Balancing mechanism based on a Load Informing strategy (LILB). The mechanism involves four components: load measurement, load informing, balance decision and switch migration. Due to the function of load informing component, when a controller becomes overloaded, it can make load balance decisions without collecting other controllers' load information. To reduce the communication overload and processing overhead caused by load informing component, this paper also proposes an inhibition algorithm to lower the frequency of informing load information. Moreover, this paper designs some decision methods of judging overloaded controllers, migrated switches, target controllers, and a judge about accepting a migration request for target controllers to avoid the load oscillation among controllers. Meanwhile, to achieve the smooth switching of controllers' roles during migrating switches, an information interaction procedure is also designed. Finally, experiments are carried out based on Floodlight and Mininet to verify the feasibility and efficiency of the proposed mechanism.

Key words: Software Defined Networking (SDN); Controller load balancing; Load informing; Balance decision

1 引言

在从传统网络到未来互联网的演进过程中，软件定义网络(Software Defined Networking, SDN)是一种新型且受欢迎的模式。通过动态地自定义网络行为，它给网络提供了可编程性、易管理性和快速的创新性[1]。SDN的本质思想是将控制平面和数据平面分离。这种分离带来了许多优势，如通过一个集中化的控制器管理整个网络和为上层的应用抽象底层网络的基础设施。然而单一且集中的控制器存在可靠性[2]和扩展性问题，为此一些研究工作部署多控制器来避免单一集中控制器的这些瓶颈。

已有许多文献给出了多 SDN 控制器的部署解决方案，如 Kandoo[3], Onix[4]和 HyperFlow[5]。虽然这些解决方案可以解决单一集中式控制器的可靠性和扩展性问题，但是带来另一个不可避免的问题：

收稿日期：2016-10-12；改回日期：2017-08-21；网络出版：2017-09-13
*通信作者：余金科 zjjyjk@bupt.edu.cn
基金项目：国家自然科学基金(61501044)，国家 863 计划项目(2013AA013502)
Foundation Items: The National Natural Science Foundation of China (61501044), The National Program 863 of China (2013AA013502)

多控制器节点间负载分布不均的问题。因为相较于其他交换机，某些交换机可能接收到大量新数据流而产生众多 Packet-In 消息。这样，控制具有众多 Packet-In 消息的交换机的控制器因为处理信息而负载过重，其他控制器则处于相对较低的负载状态下。此外，这些部署方案中，控制器与交换机间的映射关系是静态的，不均匀负载分布得不到解决而导致整个网络性能的下降，因此解决多控制器间负载分布不均问题非常重要。

当前，关于多控制器负载均衡决策的研究工作可以分为集中式决策和分布式决策两类。集中式决策[6–8]的特点是在整个网络中存在一个逻辑集中的模块或全局控制器负责局部控制器间的负载均衡。例如，文献[6, 8]提出的ElastiCon方案设计一个负载调节模块负责收集控制器池中各控制器负载信息，并根据收集的信息做出相应的决策来均衡控制器间负载。文献[8]的方案则是从网络中所有控制器中选择一个作为协调控制节点，此节点负责收集每个控制器的负载状态并维持一个全部控制器负载信息表。根据这个表，协调控制节点决定是否均衡一定区域内控制器间的负载。文献[9]阐述了集中式决策通常具有两个不足：(1)逻辑集中的全局控制器周期地收集负载信息需要与局部控制器频繁的交换信息，这样影响整个系统的性能。同时，如果全局控制器故障，网络中控制器间负载均衡功能将失效。(2)集中式决策的一个负载均衡周期中，需要两个基本的过程：一个是全局控制器周期地收集局部控制器的负载信息，另一个是给需要均衡的过载局部控制器发送负载均衡命令。在这两个过程完成的时候，局部控制器的负载状况可能发生了变化，致使均衡命令滞后于实时的负载状况。

针对上述集中式决策的不足，分布式决策则保证SDN网络中每个控制器有一个负载均衡器来负责此控制器的负载均衡，没有全局和局部控制器之分。这样可以避免全局控制器的存在带来的系统性能和故障问题。同时，每个控制器可以做出负载均衡决策，所以相较于集中式决策，发送均衡命令的过程可以省略。因此，本文研究分布式负载均衡决策。然而，现有的分布式决策[9–11]在控制器检测到自己过载后，首先收集其他所有控制器的负载信息，再做出负载均衡策略。DALB方案[9]首先周期地测量控制器的负载状况，然后判断是否超过门限值。如果超过，此控制器则收集网络中其他控制器的负载信息，再判断自己是否符合负载迁移的条件。如果符合，则做出负载均衡策略，如选择待迁移交换机并迁移。文献[10]是将网络分为多个域，每个域含有一个控制器及其控制的多个交换机。当某个域的控制器过载后，它收集邻域中控制器的负载信息，然后选择边界交换机作为待迁移交换机迁移至邻域。文献[11]首先将控制器负载均衡问题转化为交换机迁移问题，然后提出的DHA算法程序根据收集的控制器负载信息求解交换机迁移问题从而决定是否需要负载均衡。这些方案中，控制器过载后收集其他控制器信息的过程延长了负载均衡的完成时间。

基于上述分析，控制器过载后收集其他控制器负载信息的过程影响着负载均衡的时间效率。为了使过载的控制器尽可能快地被均衡，本文提出了LILB 机制(Load Informing based Load Balancing mechanism)。这个机制基于一个负载通告策略，即控制器主动地将其负载状态通告给其他控制器，同时也处理和存储其他控制器通告的负载状态。这样，控制器过载后不再需要收集其他控制器的负载信息就可以及时地做出决策。本文提出的机制作为控制器的一个模块，周期性地收集控制器的负载状态、决定是否通告负载状态给其他控制器，做出均衡决策以及迁移交换机。其中，均衡决策中含有最重过载控制器的判断以及在选择迁移交换机和目标控制器时添加相应的约束条件。交换机迁移则通过设计一种有效且简化的迁移时消息交互机制来实现。

具体来说，本文的贡献主要如下：

(1)提出一种负载通告策略，使控制器可以尽快地做决策从而缩短负载均衡时间；为了降低负载通告给控制平面网络带来的通信负荷和控制器处理通告的负荷，本文提出一个抑制算法来减少负载通告的频率；

(2)设计了最重过载控制器、待迁移交换机和目标控制器的抉择方法，同时规定在同一时间，目标控制器只响应一个负载迁移请求，在此请求完成后才能再次接受请求。这些策略可以在一定程度上避免控制器间负载震荡；

(3)设计了一种交换机迁移的消息交互机制，可以实现交换机迁移过程中控制器角色的平滑切换；

(4)实现了本文提出的负载均衡机制，并且将实现的机制嵌入 SDN 实验环境中加以验证。

本文第 2 节说明此机制的设计细节，第 3 节验证该机制的有效性和快速性，第 4 节总结全文。

2 机制描述

本文提出的 LILB 作为一个模块运行在 SDN 控制器中。此模块含有 4 个组件：(1)负载测量组件，用于测量负载度量参数和判断控制器的负载状况是否超过负载门限；(2)负载通告组件，负责发送控制器的负载状态给其他控制器；(3)负载决策组件，负

责做出均衡决策；(4)交换机迁移组件，用于迁移所选择的交换机。每个控制器的负责均衡负载的模块之间相互协调和合作，共同实现控制器间负载均衡。图 1 是负载均衡模块的流程图。

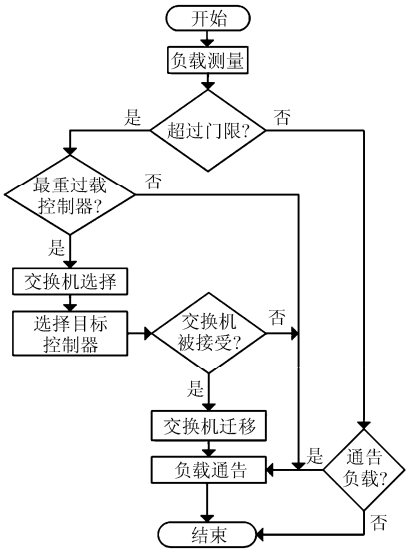

图 1 负载均衡模块的流程图

2.1 负载测量

负载测量组件周期性地测量控制器的负载参数。测量的数据包括两类：来自每个交换机的平均信息到达率(I)和交换机与控制器间的往返时间(RTT)。我们知道 CPU 是控制器的重要性能瓶颈，而 CPU 负载与信息到达率成正比。在 OpenFlow 协议中，来自交换机的信息主要有 Packet-In, Flow-Removed 和 Echo 消息。其中 Packet-In 又占着很大的比重[12]。因此本文计算 Packet-In 的总到达率来衡量控制器的负载程度。此外，RTT 是评估控制路径性能的重要参数，当选择目标控制器时，本文除了考虑目标控制器的负载程度也需要考虑控制路径的性能。本文假定控制器与交换机的连接是带内模式，即控制信息和数据信息共用通道。这样，RTT 参数可以用交换机到控制器的跳数来度量[12]。表 1 列出了本文关注的性能。负载测量组件除了测量这些参数外，还负责判断控制器负载是否超过门限。

2.2 负载通告

负载通告组件负责将控制器的负载状态通告给其他控制器，同时也处理和存储来自其他控制器的负载信息。主动的负载通告省去了控制器过载后收集其他控制器的负载状态的过程，避免了其导致的决策时延。然而，控制器间互相通告负载信息需要消耗控制平面网络的带宽。同时控制器处理来自其他控制器通告的负载信息需要存储和计算资源，这样增加控制器的处理负荷。另外，如果当前的负载状态相比上一次测量的状态并未发生大的变化时，将当前的负载状态通告给其他控制器会是一个冗余的操作。

表 1 负载测量参数

参数名称	采集目的
平均信息到达率(I)	负载评估和交换机选择
往返时间(RTT)	目标控制器选择

为了减少控制平面通信负荷、控制器处理负荷以及冗余的操作，本文提出了一个通告抑制算法来降低负载通告的频率。抑制算法如表 2 所示。L_{cu} 表示控制器当前的负载值，L_{fo} 表示上个周期测量的负载值。本文将负载值从零开始到负载门限分为若干段，例如算法中 V_1-V_2, V_4-V_5 段。当 L_{cu} 和 L_{fo} 在同一段内时，即 $V_i \le L_{\mathrm{fo}}, L_{\mathrm{cu}} \le V_{i+1}$，控制器不通告它的负载状态给其他控制器。然而，提出的抑制算法存在一个问题：当一个控制器的 L_{cu} 和 L_{fo} 处于同一个段内，并且这两个值存在误差时，由于 L_{cu} 没有被通告，其他控制器还存储着此控制器的负载信息为 L_{fo}。这时其他控制器存储的负载信息与此控制器的实时负载是存在误差的。如果一个控制器处于低负载时，当它被选为目标控制器来接受交换机时，上述的误差对它可能没有太大的影响，因为它有足够的能力来接受迁移的负载。然而，如果一个控制器处于高负载时，上述的误差可能使得它在接受迁移的负载后变为一个新的过载控制器。为了缓解这一问题，本文在分段时采取逐渐递减的方式，即 $V_m - V_{m-1} < V_{m-1} - V_{m-2}$, $m \ge 2$。这样，接近于负载门限的控制器将较频繁地通告负载状况而减少上述误差。抑制算法的效果证明如下：

本文假定控制平面有 N 个控制器。为了简化，假定每个控制器的负载门限相同且为 Thr，负载值从 0 到 Thr 分为 n 段。第 1 段的长度为 nFrg，第 2 段的长度为 $(n-1)$Frg，以此类推，第 n 段的长度为 Frg。作者保证 $(n\mathrm{Frg}+(n-1)\mathrm{Frg}+\cdots+\mathrm{Frg})+r=\mathrm{Thr}$，其中 $0 \le r < \min(n, \mathrm{Frg})$。如果 r 是 0，则段数为 n，若不是，则段数为 n 加 1，添加的段的长度为 r。

(1)如果没有抑制算法，一个周期内，负载通告的次数为 $N(N-1)$。

(2)当有了抑制算法，一个控制器不通告负载的概率为 $\{(n\mathrm{Frg})^2+((n-1)\mathrm{Frg})^2+\cdots+(2\mathrm{Frg})^2+\mathrm{Frg}^2+r^2\}\big/\mathrm{Thr}^2$。计算如下：

表 2 通告抑制算法

算法 1 通告抑制算法

输入：

L_{cu} //当前负载值

L_{fo} //上个测量周期的负载值

$\{V_0=0,V_1,V_2,\cdots,V_7,\cdots,V_n=\text{Thr}\}$

$V_m-V_{m-1}<V_{m-1}-V_{m-2},m\ge 2,n\ge 7$ //n 为负载区段数目

输出：

True or False//是否通告负载信息

(1) Value = False
(2) if($L_{\text{cu}}<$ Thr) then
(3) for ($i:1\to n-1$)
(4) if($L_{\text{cu}}\ge V_i$ && $L_{\text{fo}}<V_i$ || $(L_{\text{cu}}<V_i$ && $L_{\text{fo}}\ge V_i)$)
(5) Value = True
(6) break
(7) end if
(8) end for
(9) $L_{\text{fo}}=L_{\text{cu}}$
(10) end if
(11) return Value

假定 L_{fo} 在长度为 nFrg 的段内，L_{cu} 不通告的概率为 $\dfrac{(n\text{Frg})^2}{\text{Thr}^2}$，在长度为 $(n-1)$Frg 的段内，不通告的概率为 $\dfrac{((n-1)\text{Frg})^2}{\text{Thr}^2}$，以此类推，在长度为 r 的段内，不通告的概率为 $\dfrac{r^2}{\text{Thr}^2}$。这些概率之和即为一个控制器不通告负载的概率。$N$ 个控制器时，通告减少的次数期望则为 $\left\{\left[(n\text{Frg})^2+((n-1)\text{Frg})^2+\cdots+(2\text{Frg})^2+\text{Frg}^2+r^2\right]\Big/\text{Thr}^2\right\}N(N-1)$。本文假定控制器的门限 Thr 为 11,000，通过给出几组段数 n 的值，计算通告次数减少的百分比，如表 3 所示。

2.3 负载决策

负载决策组件主要包括 3 个功能：判断过载控制器是否是负载最重的过载控制器；是则选择需要迁移的交换机；并选择接受该交换机的目标控制器。否则，进入负载通告组件，判断是否需要将控制器的负载状态通告给其他控制器。

表 3 通告次数减少的百分比(%)

段数 n 的值	减少的百分比
7	17.78
11	11.53
15	8.49
19	6.66

2.3.1 最重过载控制器判断 在负载均衡的过程中，一个问题是不可避免的，即可能有两个或两个以上控制器超过负载门限，这时它们均需要迁移交换机来减轻负载程度。如果它们选择同一控制器作为目标控制器来接受选择的交换机，这样可能导致过载控制器被均衡后致使目标控制器成为新的过载控制器，然后均衡此目标控制器的负载时又致使它所选择的目标控制器过载。为了解决上述问题，本文规定一个过载控制器根据记录的其他控制器负载信息，判断自身为最重的过载控制器后才做出均衡决策并提出负载迁移请求。否则通告其负载信息给其他控制器，结束负载均衡周期。为此，本文在一个过载控制器判断自身是否为最重过载控制器时，提出一个过载比重公式(1)。过载控制器根据其记录的其他控制器负载信息计算所有过载控制器的过载比重，如果自身过载比重最大，则判断自己为最重过载控制器。如果此控制器与其他多个过载控制器的过载比重相同，再判定自身负载是否为最大，是则为最重过载控制器。

$$P_{\text{oL}}=\frac{L_{\text{cu}}-\text{Thr}}{\text{Thr}} \tag{1}$$

式中，L_{cu} 表示控制器当前的负载值，Thr 是控制器负载门限值，P_{oL} 是过载控制器的过载比重。

2.3.2 交换机选择 在过载控制器判定自身为最重过载控制器后，此控制器随后选择待迁移的交换机。一个交换机的平均信息到达率越大，表明这个交换机给控制器带来越多负载。为了大幅度地降低过载控制器的负载，本文优先选择高信息到达率的交换机迁移。当被选择的一个高到达率交换机有目标控制器接收并使过载控制器负载降至门限以下时，就结束交换机选择。若不能降至门限以下，则组合另一个高到达率而且可被接收的交换机为待迁移交换机，以此类推。为此，本文根据信息到达率对被过载控制器控制的交换机进行降序排列。

$$G_{\text{sw}}=\text{sort}\{I_{\text{ID}}\} \tag{2}$$

式中，I_{ID} 表示编号为 ID 的交换机的平均信息到达率。sort 代表对集合中元素降序排序。G_{sw} 为被降序排序后的交换机集合。因为选择高信息到达率的交换机迁移可能使得目标控制器变得过载，而导致控制器间的负载震荡，所以当选择交换机时本文也定义一个约束公式(3)。确保迁移的负载不超过目标控制器负载门限与当前负载差值的 $1/\alpha$。在实验中评估本文负载均衡机制时，α 被设置为 3。

$$L_{\text{mig}}\le\frac{\text{Thr}_{\text{tar}}-L_{\text{tar}}}{\alpha} \tag{3}$$

式中，Thr_{tar} 和 L_{tar} 分别表示目标控制器的门限值和负载值，L_{mig} 表示迁移给目标控制器的负载。本文

搜索集合 G_{sw}，选择满足约束公式(3)又可将控制器负载降低到门限以下的交换机组合。

2.3.3 目标控制器选择　对于所选择的交换机，可能存在多个从控制器可以作为目标控制器。通常，可以在众多控制器中选择最轻负载的控制器来接受所选择的交换机。然而，如果当最轻负载的从控制器与选择的交换机间的往返时间较大，就会影响控制路径的性能。为此，本文在选择目标控制器时既考虑控制器的负载状态又考虑其到所选择交换机的往返时间。本文设计的目标控制器选择公式为

$$C_{\text{tar}} = w_1\left(\frac{\text{Thr}_{\text{tar}} - L_{\text{tar}}}{\text{Thr}_{\text{tar}}}\right) + w_2 \text{e}^{-\text{RTT}} \tag{4}$$

式中，RTT 表示往返时间。w_1 和 w_2 分别是 $\dfrac{\text{Thr}_{\text{tar}} - L_{\text{tar}}}{\text{Thr}_{\text{tar}}}$ 与 $\text{e}^{-\text{RTT}}$ 的权重系数，并且 $w_1+w_2=1$。C_{tar} 是选择目标控制器的指标参数。C_{tar} 最大的控制器将作为目标控制器接受选择的交换机。

2.4 交换机迁移

当两个或两个以上控制器在同一时间检测到自身过载，有可能由于负载通告的不及时，导致这些过载控制器都判定自身为最重过载控制器。假如它们选择了同一目标控制器，这样可能带来控制器的负载震荡问题，为避免上述问题，本文规定同一时间目标控制器仅接受来自一个过载控制器的交换机迁移请求，在迁移过程完成后才能再次接受迁移请求。

OpenFlow1.3[13]或以上版本为交换机所连的 SDN 控制器定义了 3 种角色：master, equal 和 slave。其中，slave 控制器收不到交换机异步信息(如 Packet-In)。每个控制器可以发送角色请求信息给交换机，请求变为交换机的 master, equal 或 slave 控制器。当交换机收到一个从 slave 控制器发来的变为 master 角色的请求，此交换机改变 slave 控制器为 master 并设置先前的 master 控制器为 slave。为了完成交换机迁移过程中源控制器、目标控制器及待迁移交换机间的消息交互和角色正常切换，本文基于 OpenFlow 协议设计了交换机迁移过程的信息交互机制，如图 2 所示。

首先高负载的控制器 A 给目标控制器 B 发送带有两个参数的交换机迁移请求信息(<1>)。然后，控制器 B 在变为交换机 T 的 equal 控制器(<2>)后，会通告 master 控制器 A，它的角色改变完成(<4>)。完成角色改变后，控制器 B 就可以收到来自交换机 T 的异步信息 (<6>)。在收到控制器 B 回复迁移请求前，控制器 A 可能尚有未完成的来自交换机 T 的请求，所以控制器 A 继续与交换机 T 交互完成未完成的工作直到给控制器 B 发送“结束迁移”消息(<5>,<7>,<8>)。控制器 B 在收到结束迁移信息后，再次发送角色请求信息给交换机 T 请求从 equal 变为 master(<9>)。交换机也设置控制器 A 为 slave (<10>)。最终两个控制器都会同步更新控制器与交换机映射的信息。此时，整个交换机迁移过程结束。

2.5 负载均衡震荡规避

在控制器间进行负载迁移时，需要考虑的一个问题就是在交换机迁移后，目标控制器可能过载，致使目标控制器需要再次迁移负载，形成一种震荡迁移的现象。为此，本文在负载决策和交换机迁移组件中采取了相应的规避策略。这些策略的结合可以有效约束迁移行为，规避负载均衡震荡。具体的策略如下：

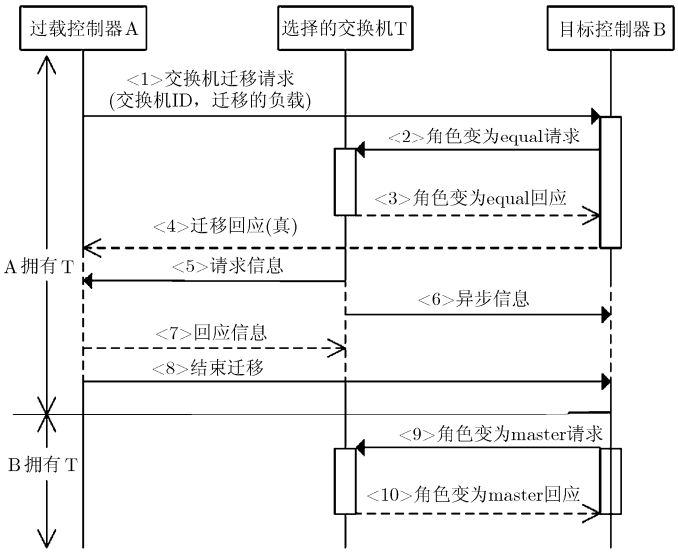

图 2 交换机迁移信息流程

(1)最重过载控制器判断：以图 3 为参考，假设控制器 A 和 B 过载，控制器 C 不过载，并且各控制器存储着其他控制器的实时负载信息。由于控制器 A 和 B 过载，它们都需要均衡负载，最重过载控制器判断策略的存在，可以保证仅判定为最重过载的过载控制器才做均衡决策并发起负载迁移请求。通常而言，控制器 A 和 B 中只有一个控制器判定自身为最重的过载控制器，然后做出均衡决策，完成负载迁移。另外一个则通告自身负载信息给其他所有控制器，然后结束此次的负载均衡周期。然而如果网络中负载通告信息不及时，也可能出现多个过载控制器同时判断自身为最重过载控制器。如果其目标控制器不同，则各自迁移交换机。如果目标控制器相同，则本文设计的迁移请求接受策略可以进一步规避目标控制器的负载震荡。

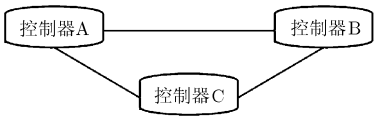

图 3 震荡规避参考图

(2)待迁移交换机选择：在过载控制器判定自身为最重过载控制器后，在此控制器选择待迁移交换机时，本文给出了一个约束公式(3)。式(3)表示待迁移交换机给目标控制器所带来的负载应该小于或等于目标控制器的门限与当前负载差值的 $1/\alpha$ 。即保证待迁移的交换机迁移后不会致使目标控制器过载，从而规避负载均衡震荡。

(3)迁移请求接受：仍以图 3 为参考，假设控制器 A 和 B 同一时间检测自身过载，它们之前不过载。控制器 C 一直未过载，由于没有同步机制，控制器 A 和 B 在没有收到对方通告的负载信息之前，可能都会判定自身是最重过载控制器。从而它们都会选择待迁移交换机和目标控制器。假定目标控制器同为控制器 C，控制器 A 和 B 都会给控制器 C 发送迁移请求，然而控制器 C 在接受控制 A 待迁移的交换机和控制器 B 待迁移的交换机后可能过载。为此，本文规定每个目标控制器在同一时间仅接受一个迁移请求，在此请求完成后才能再次接受请求，从而规避负载均衡震荡。

3 实验评估

本文通过在数据层面模拟大量且不同的数据包分布，来模拟控制层面上多节点控制器负载分布不均匀。仿真实验的目的有 2 个：(1)验证本文 LILB 机制的有效性。在控制器负载分布不均匀的情况下，比较控制器与交换机映射关系为静态(STATIC)时、控制层面的控制器含有 LILB 模块时或 DALB 方案[9]时，各自控制层面的吞吐量；(2)统计和记录本文 LILB 机制的完成时间，证明其高效性。在控制器负载分布不均匀的模型下，比较方案 DALB 和本文 LILB 负载均衡机制完成负载均衡的时间。

3.1 实验环境设置

为验证本文提出的负载均衡机制 LILB，我们以 Floodlight[14]作为分布式 OpenFlow 控制器，本文提出的负载均衡机制作为控制器的一个模块。本文运用 Cbench[15]测量控制器的最大 Packet-In 包处理能力。另外，本文选择 Minient[16]模拟 SDN 实验环境的数据网络，网络中的交换机为基于软件的虚拟 OpenFlow 交换机 OpenVswitch[17]。实验中，本文部署两个 SDN 控制器节点，同时配置 4 个交换机以 master 角色连接控制器 A，以 slave 角色连接控制器 B，另外 4 个交换机以 master 角色连接控制器 B，以 slave 角色连接控制器 A。为模拟出大量且分布不均的 Packet-In 消息，本文通过 Hping[18]调整主机到主机的流发送速率。

3.2 吞吐量

在仿真之前，本文使用 Cbench 工具测量了 Floodlight 控制器能处理 Packet-In 数据包的最大速率。测量结果显示控制器每秒能处理约 12758 个 Packet-In 包(12758 pps)。为模拟出负载分布不均匀情况，在某个时刻，我们给控制器 A 注入每秒 5000 个 Packet-In 包(5000 pps)，给控制器 B 注入 16000 个 Packet-In 包(16000 pps)。作为比较，本文同时测量在上述工作量(WORKLOAD)下交换机与控制器间映射关系为静态时的吞吐量以及 DALB 方案[9]的吞吐量。

如图 4 所示，在相同工作量下，静态映射方式下两控制器的总吞吐量平均低于文献[9]的 DALB 方案和本文的 LILB 机制。这是因为 Packet-In 请求注入的差异导致控制器 A 和控制器 B 间负载的不均衡，控制器 B 过载，缓冲区溢出而导致 Packet-In 包丢失。而在 DALB 和本文的 LILB 下，当控制器 B 过载时，负载均衡功能会被触发。控制器 B 迁移部分负载到控制器 A，而控制器 A 有能力处理迁移过来的 Packet-In 包，从而两控制器的总吞吐量得到增加。可见 DALB 和本文的 LILB 都可以有效地均衡控制器间的负载。从图 4 中也可以看出 DALB 和 LILB 在负载均衡后，吞吐量的变化趋势近似。

3.3 完成时间

在评估本文机制 LILB 的完成时间时，我们设置控制器 A 和控制器 B 的负载门限值都为 11000

pps。一旦控制器 A 或控制器 B 超过各自的门限，它将迁移部分负载到另一个控制器。

图 5 中，从 0~45 s，两控制器的负载都低于各自门限值。在 50 s 时，我们为控制器 B 所控交换机的主机传输指令使其发送更多数据流。这样负载均衡模块检测到控制器 B 过载并做出负载均衡决策，迁移选择的交换机到控制器 A。在 55 s，控制器 B 的负载降到门限以下，控制器 A 的负载上升。作为比较，我们在 DALB[9]方案下实施了相同的测试过程并记录完成时间。从图 5 中可以看出本文机制在 5 s 内完成了负载均衡，少于 DALB 方案的 10 s。

4　结束语

交换机与控制器间静态的映射关系很难适应流量负载分布不均而导致网络性能下降。为了解决这个问题，本文提出一个基于负载通告策略的负载均衡机制来均衡多个 SDN 控制器间的负载。本文围绕负载均衡机制的 4 个组件：负载测量、负载通告、负载决策和交换机迁移设计并实现了多控制器的负载均衡机制。首先明确了负载测量参数，其次提出负载通告策略，为了减少负载通告带来的通信负荷和处理负荷，本文还提出一个通告抑制算法来降低负载通告的频率。然后，本文还提出了最重过载控制器判断、待迁移交换机和目标控制器的决策方法、以及目标控制器接受迁移请求的策略，可以有效的避免控制器的负载震荡。再次，本文设计了一种交换机迁移的消息交互机制实现交换机迁移。最后，本文从吞吐量和完成时间评估了所提出的负载均衡机制。评估的结果表明本文提出的机制可以确保吞吐量并缩短负载均衡的完成时间。

未来工作中，作者将继续优化负载均衡机制的 4 个组件，特别是负载通告组件和负载决策组件。另外，也将本文的负载均衡机制应用于多个异构的 SDN 控制器环境中。

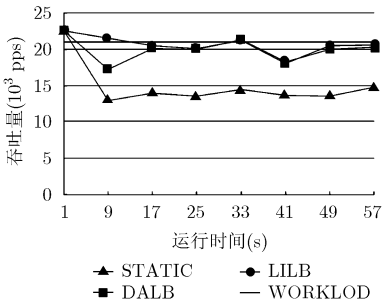

图 4 控制层面的总吞吐量

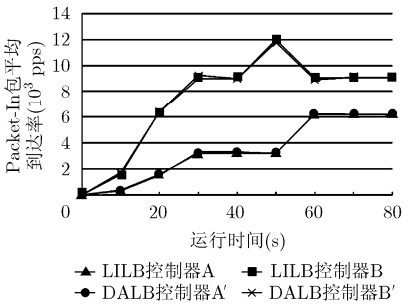

图 5 负载均衡的完成时间

参 考 文 献

[1] ONF. The open networking foundation[OL]. https://www.opennetworking.org/, 2015.5.

[2] 张朝昆, 崔勇, 唐澍澍, 等. 软件定义网络(SDN)研究进展[J]. 软件学报, 2015, 26(1): 62–81. doi: 10.13328/j.cnki.jos.004701. ZHANG Chaokun, CUI Yong, TANG Hehe, *et al*. State-of-the-art survey on software-defined networking (SDN)[J]. *Journal of Software*, 2015, 26(1): 62–81. doi: 10.13328/j.cnki.jos.004701.

[3] YEGANEH S H and GANJALI Y: A framework for efficient and scalable offloading of control applications[C]. Proceedings of the First Workshop on Hot Topics in Software Defined Networks ACM, Helsinki, 2012: 19–24.

[4] KOPONEN T, CASADO M, GUDE N, *et al*. Onix: A distributed control platform for large-scale production networks[C]. 9th USENIX Symposium on Operating Systems Design and Implementation, Vancouver, 2010: 351–364.

[5] YEGANEH S H, TOOTOONCHIAN A, and GANJALI Y. On scalability of software-defined networking[J]. *IEEE Communications Magazine*, 2013, 51(2): 136–141. doi: 10.1109/MCOM.2013.6461198.

[6] DIXIT A A, HAO F, MUKHERJEE S, *et al*. Towards an elastic distributed SDN controller[C]. ACM Sigcomm Computer Communication Review, Hong Kong, 2013: 7–12.

[7] DIXIT A A, HAO Fang, MUKHERJEE S, *et al*. ElastiCon: An elastic distributed sdn controller[C]. Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems ACM, California, 2014: 17–28.

[8] LIANG C, KAWASHIMA R, and MATSUO H. Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers[C]. Second International Symposium on Computing and Networking (CANDAR'14), Shizuoka, 2014: 171–177.

[9] ZHOU Yuanhao, ZHU Mingfa, XIAO Limin, *et al*. A load balancing strategy of SDN controller based on distributed decision[C]. Trust, Security and Privacy in Computing and

Communications (TrustCom), Beijing, 2014: 851–856.

[10] YAO L, HONG P, ZHANG W, *et al*. Controller placement and flow based dynamic management problem towards SDN[C]. 2015 IEEE International Conference on Communication Workshop (ICCW), London, 2015: 363–368.

[11] CHENG Guozhen, CHEN Hongchang, HU Hongchao, *et al*. Toward a scalable SDN control mechanism via switch migration[J]. *China Communications*, 2017, 14(1): 111–123. doi: 10.1109/CC.2017.7839762.

[12] YAO Guang, BI Jun, and LI Yuliang, *et al*. On the capacitated controller placement problem in software defined networks[J]. *IEEE Communications Letters*, 2014, 18(8): 1339–1342. doi: 10.1109/LCOMM.2014.2332341.

[13] Open Networking Foundation. OpenFlow Switch Specification Version 1.3.3(Protocol version Ox04)[OL]. https://www.opennetworking.org/, 2013.7.

[14] Big Switch. Floodlight[OL]. http://www.projectfloodlight.org/floodlight/, 2015.6.

[15] Slashdot Media. Cbench[OL]. http://sourceforge.net/projects/cbench/, 2015.1.

[16] Mininet Team. Mininet[OL]. http://mininet.org/, 2015.6.

[17] The Linux Foundation. Open vSwitch[OL]. http://openvswitch. org, 2015.9.

[18] Salvatore Sanfilippo. Hping[OL]. http://hping.org/, 2015.12.

王 颖： 女，1976 年生，博士，副教授，研究方向为 IT 服务管理、云计算运维管理、未来网络管理等.

余金科： 男，1991 年生，硕士生，研究方向为软件定义网络的负载均衡技术.

裴科科： 男，1992 年生，硕士生，研究方向为网络虚拟化管理平台故障恢复.

邱雪松： 男，1973 年生，博士，教授，研究方向为网络管理与通信软件.