**文献名称**：Dynamic Switch Migration in Distributed

Software-Defined Networks to Achieve

Controller Load Balance

**中文名称**：在分布式软件定义网络中利用动态交换机

迁移实现控制器负载均衡

**作者：**Yang Xu,Marco Cello,I-Chih Wang,Anwar

Walid,Gordon Wilfong,Charles H.-P Wen

Mario Marchese, H.Jonathan Chao

摘要

为了提高可扩展性和可靠性，软件定义网络(SDNs)中使用了多个分布式控制器，其中每个控制器管理网络的一个静态分区。在本文中，我们证明了交换机和控制器之间的动态映射可以提高管理流量变化的效率。特别地，我们提出了平衡控制器(BalCon)和BalConPlus两种SDN交换机迁移方案，以实现SDN控制器之间的负载均衡，迁移成本小。Balcon适用于网络不需要对交换机请求进行串行处理的情况。对于其他场景，BalConPlus是更合适的，因为它不受交换机迁移中断的影响，并且不会导致任何服务中断。仿真结果表明，BalCon和BalConPlus只迁移少量计算开销较低的交换机，可以显著降低SDN控制器之间的负载不均衡。我们还构建了一个基于开源SDN框架RYU的原型测试平台，验证了Balcon和Balconplus的实用性和有效性。实验证实了模拟结果，它还表明，Balconplus对交换机迁移中断免疫，这在Balcon中是一个不利影响。

索引术语：软件定义网络、分布式控制器、负载均衡、交换机迁移。

目录

[第一章 绪论 1](#_Toc8601769)

[第二章 动机 3](#_Toc8601770)

[第三章 控制器负载均衡问题建模 7](#_Toc8601771)

[3.1 系统模型 7](#_Toc8601772)

[3.2 OCLB作为图分区问题 8](#_Toc8601773)

[3.3 NP-Completeness证明 9](#_Toc8601774)

[第四章 BALCON算法 11](#_Toc8601775)

[第五章 迁移中断和BalConPlus 15](#_Toc8601776)

[5.1 迁移中断 15](#_Toc8601777)

[5.2 BalConplus 16](#_Toc8601778)

[5.3 BalCon Vs. BalConPlus 17](#_Toc8601779)

[第六章 模拟性能评估 19](#_Toc8601780)

[6.1 BalCon的动态场景有效性 19](#_Toc8601781)

[6.2 静态场景 20](#_Toc8601782)

[第七章 BALCON和BALCONPLUS的原型和实验结果 23](#_Toc8601783)

[7.1设计 23](#_Toc8601784)

[7.2实验装置 24](#_Toc8601785)

[7.3 自动参数测量 25](#_Toc8601786)

[7.4控制器负载均衡 26](#_Toc8601787)

[7.5 BalCon Vs BalConPlus分组响应时间 27](#_Toc8601788)

[第八章 相关工作 29](#_Toc8601789)

[第九章 总结 31](#_Toc8601790)

[参考文献 33](#_Toc8601791)

第一章 绪论

软件定义网络（SDN）是一种很有前途的网络技术，它可以实现网络创新，并为网络运营商提供对网络基础设施的更多控制。它通过将网络控制功能从转发设备（如交换机/路由器）移动到逻辑集中控制器，将控制平面逻辑与数据平面分离，从而通过软件实现网络功能。但是，随着SDN中交换机的数量增加，集中控制器可能无法处理来自交换机的所有请求。此外，由于单点故障，SDN控制器的故障会导致整个网络瘫痪。最近的工作已经提出使用多个物理分布的SDN控制器来提高系统的可扩展性和可靠性，同时保持逻辑集中系统的简单性[2][4]。

现有的多控制器体系结构存在的问题之一是它们在SDN交换机和控制器之间的静态映射，使得控制平面无法适应流量变化。如[5]所述，真实网络在时间维度（一天中不同时间的流量变化，甚至在较短的时间尺度内）和空间维度（网络不同位置的流量变化）上都可能表现出巨大的变化[6]。如果SDN交换机控制器映射是静态的，那么这些巨大的变化可能会导致控制器之间的不平衡，即一些过载和一些未充分利用。过载的控制器将以更大的延迟响应交换机请求，从而降低用户体验的质量。因此，交换机和控制器之间的动态映射可以通过将某些交换机从过载的控制器迁移到负载较轻的其他控制器来克服不平衡并减少连接设置延迟。但是，由于四阶段交换机迁移协议[7]会导致服务中断（详见第五节），动态交换机迁移确实会产生一些开销。

虽然已经提出了一些工作来解决多个控制器之间的交换机迁移问题，但是缺乏系统的方法来定量识别要迁移哪些交换机以获得更好的控制器负载均衡。

本文在理论，算法和实现方面的贡献总结如下：

* 我们展示了SDN交换机和控制器之间的动态映射在不同的流量负载下提供了系统的灵活性和效率。控制器之间的交换机迁移来实现控制器负载均衡已经被建模为优化问题并且表明它是NP-Complete问题。
* 由于模型最优解的计算复杂度非常高，我们提出BalCon，一种能够通过交换机迁移实现控制器之间的负载均衡的启发式解决方案。
* 我们分析了交换机迁移过程中产生的开销，并讨论了迁移过程中的服务中断问题，称为交换机迁移中断。 BalConPlus是BalCon的改进版本，旨在通过引导新到达的流量远离正在迁移的交换机来消除中断。 BalConPlus仅需要从BalCon基础上进行微小更改，而不会产生额外的实施复杂性;
* 我们在Matlab中实现BalCon和BalConPlus。仿真结果表明，控制器之间的负载不平衡（表示为负载的方差）减少了40％，拥塞控制器的负载减少了19％，同时迁移的SDN交换机数量相对较少;
* 我们基于RYU（一种用Python编写的流行SDN控制器[8]）对测试平台进行原型设计，以验证BalCon和BalConPlus的实用性和有效性。通过实验结果，我们证明BalConPlus不受中断影响;
* BalCon和BalConPlus的操作需要事先知道一些系统参数（例如，计算路径的成本和安装到交换机的流条目的成本）以预测迁移结果。我们提出了一个自动参数测量和计算框架，在我们的原型测试平台中运行BalCon和BalConPlus。

BalCon和BalConPlus可用于大型数据中心网络，运营商网络或企业网络，这些网络具有大量由多个控制器控制的网络设备(如交换机、防火墙和入侵检测系统(IDS))，以实现控制器负载均衡。尤其是，当网络不需要对交换机请求进行串行处理时，可以使用balcon，因为这样的网络不会出现交换机迁移中断问题（详情在第V节中讨论）。对于需要对交换机请求进行串行处理的网络，BalConPlus更适合，因为它可以引导新流量远离正在迁移的交换机，以避免服务中断。

本文的其余部分安排如下：第二节介绍了我们工作的动机。第三节介绍了系统模型。第四节介绍了BalCon的设计和细节。 第五节讨论了交换机迁移中断问题并提出了BalConPlus。在第六节中，我们使用Matlab仿真评估BalCon和BalConPlus的性能。第七节介绍了基于RYU控制器的原型试验台，并给出了详细的实验结果。第八节回顾了以前的相关工作。结论见第九节。

第二章 动机

SDN网络由SDN交换机和逻辑集中式SDN控制器组成。 每个SDN交换机根据存储在其流表（转发状态）中的规则处理和传送分组，而SDN控制器使用标准协议（例如，OpenFlow [9]）配置每个交换机的转发状态。当新的流量到达时，代表转发状态的流量规则被安装在SDN交换机中。

为了克服单个集中控制器的可扩展性问题，在文献中已经提出了几种方法。 最有效的方法之一是使用分布式控制器。现有的分布式控制器解决方案仍然存在SDN交换机与控制器之间的静态映射问题，限制了动态负载自适应能力。

让我们使用图1中的示例简要解释SDN中的响应模式行为，其中网络被分成两个域，每个域由控制器控制。假设主机H1产生的新流f1到达交换机S1。S1不具有与流相关联的任何规则，并且生成一个packet-in包到控制器C1（即步骤a中的第一个红色箭头）。然后C1计算路线（即，蓝色的步骤b），并将流规则安装到由其本身控制的SDN交换机上（即步骤c中指向S1和S2的绿色箭头，假设流的转发路径是S1→S2→第二个域）。当流到达S5时，交换机不具有与流相关联的任何规则，因此发送一个packet-in包至C2请求其计算流路径并在S5和S6上安装流规则（假设流的转发路径在第二个域是S5→S6）。

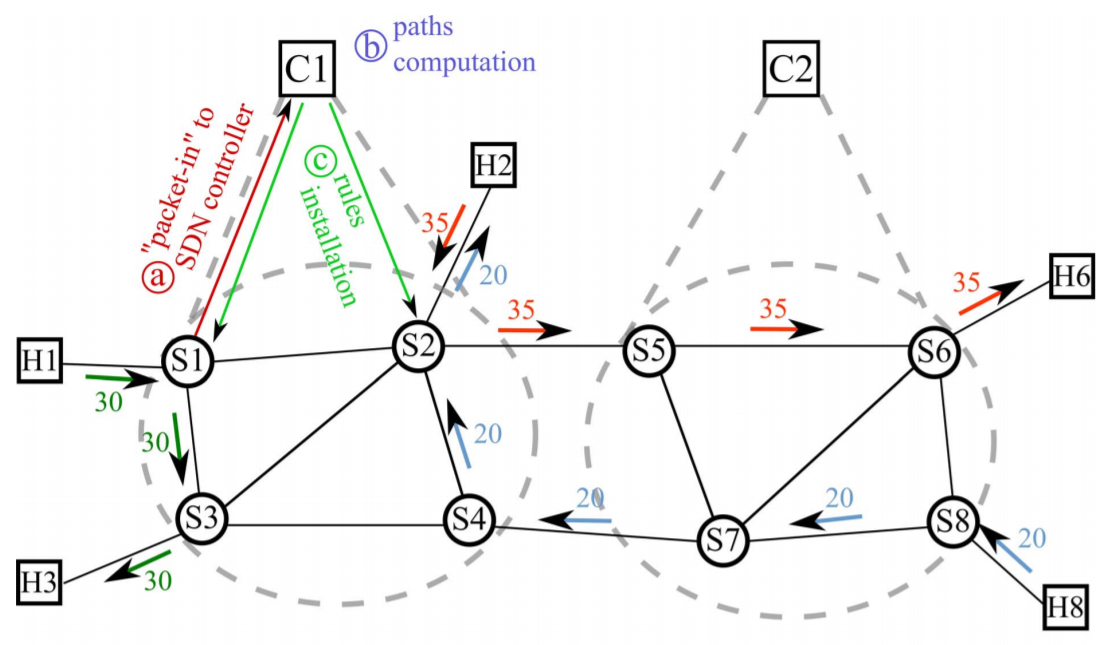


图1 SDN控制器负载不均衡场景

假设由于流量变化，大量的新流量到达网络，图1中描述了当前的流量模式。特别是：

* 主机H1每秒生成30个新流到H3，这些流通过S1→S3→H3(绿色箭头)；
* 主机H2每秒生成35个新流到H6，这些流通过S2→S5→S6→H6(红色箭头);
* 主机H8每秒生成20个新流到H2，这些流通过S8→S7→S4→S2→H2(蓝色箭头)。

此时，我们想知道由于新流的实例化，控制器C1和C2的计算负担是什么。假设单个流的路径计算需要α单位的负载,而在单个交换机中安装单个流规则需要β单位的负载。

在控制器C1中：

* 绿流产生30α单位用于路径计算和（30+30）β单位用于S1和S3的规则安装。
* 红流产生35α单位用于路径计算和35β单位用于S2的规则安装。
* 蓝流产生20α单位用于路径计算和（20+20）β单位用于S2和S4的规则安装。

在控制器C2中：

* 红流产生35α单位用于路径计算和（35+35）β单位用于S5和S6的规则安装。
* 蓝流产生20α单位用于路径计算和（20+20）β单位用于S7和S8的规则安装。

如果我们假设α=1和β=0.1,我们得到:

LC1=(30+35+20)α+β(30+55+30+20)=98.5 units/s.

LC2=(35+20)α+(35+35+20+20)β=66 units/s.

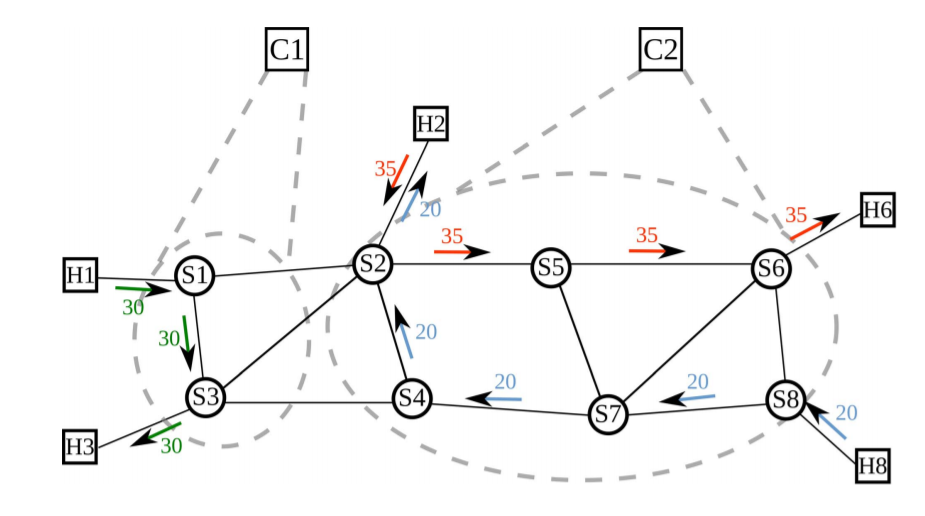


图2 交换机迁移后控制器负载均衡提升

在上述示例中，控制器C1和C2之间的负载高度不平衡。如果我们能够通

过适当的交换机迁移动态缩小或扩大SDN域或分区，我们就可以获得图2中控制器和交换机之间的新映射。S2和S4现在是第二域的一部分，由C2控制。现在新的控制器负载为：

LC1=(30)α+β(30+30)=36 units/s.

LC2=(35+20)α+(55+20+35+35+20+20)β=73.5 units/s.

因此，我们在C1处的控制器负载显著减少(63%)，而在C2处的控制器负载增加相对较小(11%)。

如[5]所述，使用生产数据中心的实际测量，Benson等人[6]发现交换机处的峰值和中间流量到达率有1-2个数量级的差异：峰值流量到达率可达300M/s，中间流量介于1.5M/s和10M/s之间。假设每个控制器都能管理高达2M/s的流量到达率，则只需要1-5个控制器来处理中间负载，而峰值负载需要150个。如果我们使用静态映射，每个控制器都需要有能力处理峰值流量到达（最坏情况）。如果我们有一个动态映射，每个控制器的容量可以降低，因为不同分区（域）的峰值通常不会由于多路复用和共享效应同时出现。

受上述观察的启发，我们试图回答我们的关键问题：如何从一个控制器的域动态选择和迁移交换机到另一个控制器域以均衡控制器负载？答案将在很大程度上取决于交换机迁移过程的复杂性和成本。

我们首先在SDN多控制器场景中提出最优控制器负载均衡（OCLB）问题，并证明它是NP-Complete问题。然后，我们将OCLB问题建模为一个图划分问题，并开发了BalCon和BalConPlus：SDN控制器之间通过SDN交换机迁移进行负载均衡的两种有效算法。

第三章 控制器负载均衡问题建模

3.1 系统模型

本节的目标是找到一个合适的模型，该模型考虑到每个SDN交换机的流量的到达动态，并将它们与每个SDN控制器的计算负载联系起来。然后我们将控制器负载均衡（CLB）问题正式化为优化问题。

SDN场景由一组SDN交换机Si∈S组成，由一组SDN控制器Cm∈C管理。根据现有技术，我们不能假设SDN交换机之间可预测的流量或众所周知的流量模式，但我们可以在运行时监控流量负载。因此，我们用fo,Si表示从SDN网络外部到SDN交换机Si的新流的当前到达率，用fSi,o从交换机Si离开SDN网络的新流的当前到达率，而用fSi,Sj表示通过两个连接的SDN交换机Si和Sj之间的链路的新流的当前到达速率。换句话说，fSi,Sj表示来自SDN交换机Si的新流到达SDN交换机Sj处的当前到达率。参考图1，我们得到：fo,S1=30，fo,S2=35, fo,S8=20,fs3,o=30,fs2,o=20,fs6,o=35,fs1,s3=30,fs2,s5=35,fs4,s2=20,fs5,s6=35, fs7,s4=20,fs8,s7=20.

如前所示，控制器Cm处的负载LCm由三个主要组件组成:来自SDN网络外部的新流的路径计算负载(如图1中绿色箭头H1→S1和红色箭头H2→S2);来自另一个SDN域的流的路径计算负载(如图1中蓝色箭头S7→S4);对于通过Cm控制的域的所有流，Cm控制的每个交换机处的规则安装负载。

定义1：外部流的路径计算负载。当一批流量以fo,Si的速率从网络外部到达Si时，Si的SDN控制器的路径计算产生的计算负载等于：

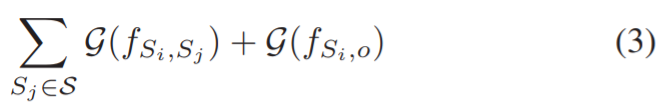


定义2：来自其他SDN域的流的路径计算负载。当一批流由另一个SDN控制器控制的交换机Sj到达Si时，速率为fSj,Si，Si的SDN控制器的路径计算产生的计算负载等于：



SDN控制器执行路径计算所需的计算负载依赖于流经函数K的流的到达率。函数K的定义不是本工作的目标。

定义3：规则安装负载。交换机Si中规则安装导致控制器计算负载等于：

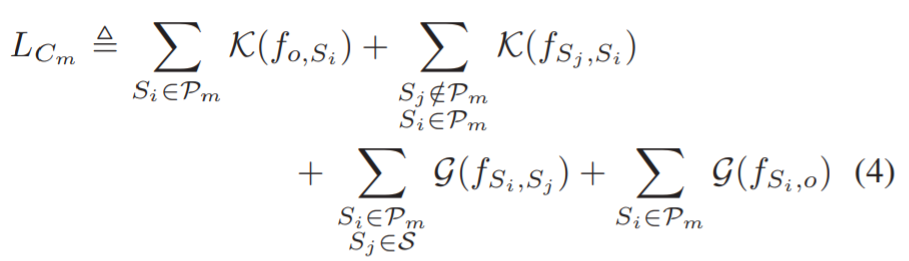


方程3表示通过Si到达其他交换机或离开SDN网络的流量。函数G将Si处的流到达率映射到规则安装所需的SDN控制器的计算负载。

定义4：由SDN控制器Cm控制的SDN交换机组用Pm表示

对S进行分区,Pm ⊂ S,Pm ∩ Pn = ∅, n = m.

定义5：SDN控制器Cm(LCm)的总计算负载为：



过载SDN控制器会降低其响应能力并导致性能下降，因为流将经历一个意外的延迟。

定义6：当SDN控制器的总体计算负载为:

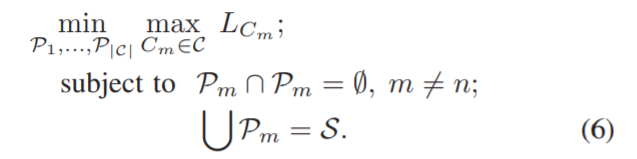


其中L表示每个SDN控制器所能承受的最大计算负载。

当拥塞发生时，需要迁移过程来减少过载。特别是，从分区（P1，...，P|C|）开始，其中至少有一个控制器Cm（条件LCm>L），我们需要找到一个新的分区（P1'，...，P|C|'）使得SDN控制器加载LCm≦L，Cm∈C。

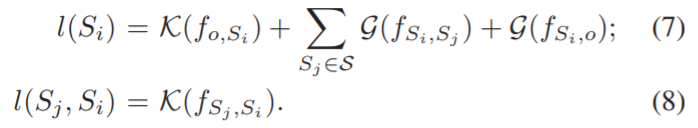
CLB问题可以表示为一个数学优化问题，我们称之为最优CLB(OCLB)问题，其定义如下：

定义7：OCLB问题。

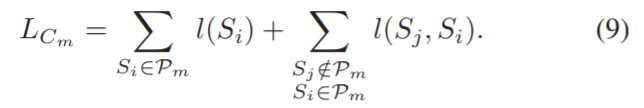


3.2 OCLB作为图分区问题

OCLB问题可以表示为一个图上的分区问题，LCm的计算可以直接在图上归纳出来。特别地，我们将SDN网络表示为有向边缘加权和顶点加权图G（S，E），其中SDN交换机是具有权重l（Si），Si∈S和边E={（Si，Sj）：Si，Sj∈S，l（Si，Sj）>0}，边是SDN交换机之间的连接。l（Si，Sj）是（Si，Sj）的边缘权重。 就是：



Cm处的总负载(用LCm表示)是属于其分区的顶点的权值之和加上指向Cm分区的边的权值之和。具体地说：



注意，方程9是方程4的另一种表达式。

图3将图1表示为一个图分区问题。例如，S1的顶点权值表示S1给C1带来的计算负载。特别地，l(S1)=33，它是K(fo,S1)= 30(30流/s)的总和以及流向S3G(fS1,S3)=3的流的规则安装。

参照同样的图，我们得到:

LC1=l(S1)+l(S2)+l(S3)+l(S4)+l(S7,S4)=33+40.5+3+2+20=98.5units/s. LC2=l(S5)+l(S6)+l(S7)+l(S8)+l(S2,S5)=3.5+3.5+2+22+35=66units/s.

3.3 NP-Completeness证明

证明了OCLB问题是一个NP完全问题。由于篇幅有限，省略了证明的细节。完整的证明可以在这里找到: NP-Completeness证明。

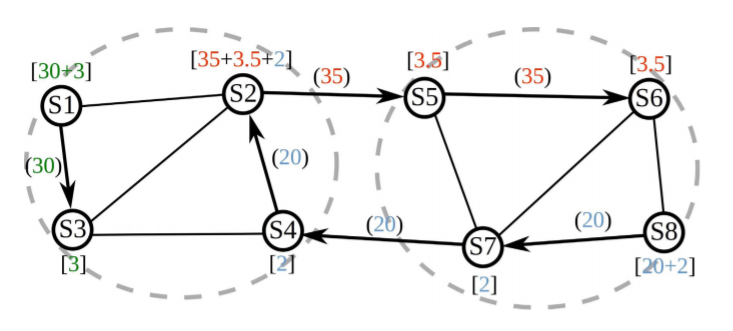


图3 将图1中的SDN网络场景化为图分区问题

第四章 BALCON算法

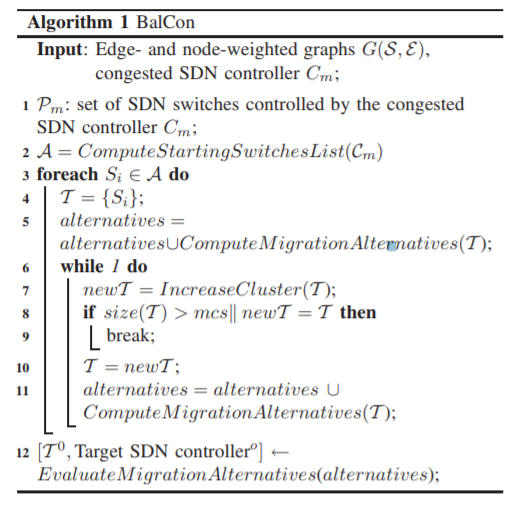
一个最优的SDN交换机迁移是不切实际的，因为它的计算复杂性(即OCLB问题是NP-complete)，并可能导致不必要的过度交换机迁移。更实际的方法应该包括对交换机分区进行增量调整，即只有少数SDN交换机被迁移。

在本节中，我们提出了平衡控制器（balcon），这是一种算法解决方案，旨在通过适当的SDN交换机迁移来解决和减少SDN控制器之间的负载不均衡。Balcon背后的关键观察是，通过分析SDN交换机的通信模式，可以实现有效的交换机迁移。交换机迁移应在集群的粒度上：具有强连接的交换机7应始终分配给同一个控制器。

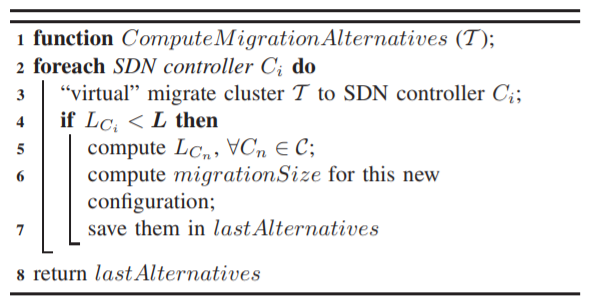
BalCon是一种启发式算法，它在网络运行时运行，能够通过适当的SDN交换机迁移来检测和解决SDN控制器的拥塞问题。BalCon可以实现为SDN控制器的北行应用程序(更多细节见第7节)。BalCon由三个阶段组成，如下所述：

1. 监控和拥塞检测:在网络运行过程中，BalCon不断监控每个SDN控制器的拥塞水平。当LCm达到预定阈值时，SDN控制器Cm被认为是拥塞的。然后BalCon计算可迁移的SDN交换机的列表。列表按使用预先确定的度量计算的优先级排序。例如，观察到新流量迅速增加的SDN交换机可以获得高优先级，因为它们可以用packet-ins快速地使SDN控制器过载。
2. 集群和迁移评估:BalCon从优先级列表中的SDN交换机开始，分析SDN交换机之间的流量模式，找到大量连接交换机的集群(下面讨论)。
3. 集群迁移:当找到最佳集群并对迁移进行评估，将属于集群的SDN交换机迁移到新的SDN控制器。

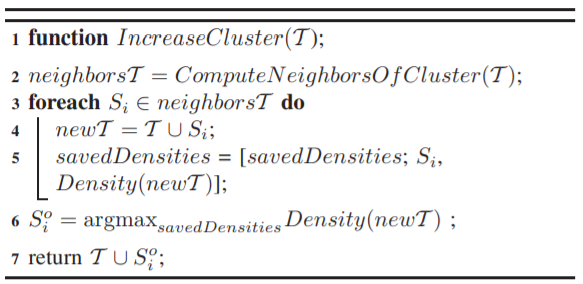
我们提出的算法实质上是基于三个函数的迭代：扩展集群的增量集群；计算迁移三个变量，其中对所选集群的不同目标SDN控制器的迁移进行评估（生成迁移变量或在后继中简单地称为替代变量）；Evaluate-Best Migration Alternative备选方案中给出了一个备选方案列表，计算出最佳替代方案（基于以下描述的一些标准）。算法如算法1所示。



从集合Pm（由拥塞的SDN控制器Cm控制的SDN交换机），算法提取包含用于集群构造的起始节点的子集列表A（StartingSwitch List）（行2）。例如，可以通过寻找具有流量到达率显著增加的SDN交换机来计算A。选择属于A的第一个SDN交换机并将其插入空集群T（行4）。属于T的SDN交换机的迁移备选方案是通过ComputeM迁移备选方案计算的。随后，该算法执行while循环，其中使用IncreaseCluster函数连续扩展集群，并使用ComputeMigrationAlternatives函数进行评估。当满足两个停止条件之一时，算法停止：集群达到预定大小mcs（最大群集大小），即size（T）>mcs，或者增加的群集等于旧群集（newT=T）。然后选择A中的下一个交换机并将其插入空集群T中。当达到mssls（最大启动交换机列表大小）时，将使用AlternativeEvaluation函数评估所有迁移备选方案。选择由T0（集群）和目标SDN控制器（将接收T0的控制器）组成的最佳替代方案，并且可以进行迁移。在下文中，我们将详细说明上述功能。

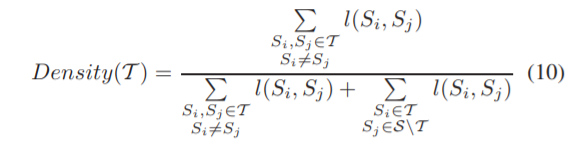


ComputeMigrationAlternatives虚拟将群集T迁移到不同的SDN控制器目标。 对于每个控制器，它计算控制器负载和迁移大小。 表I显示了当T = {S1，S2，S56}时，具有60个开关和5个控制器的方案中ComputeM igrationAlternatives例程的可能输出。 对于SDN控制器Ci，该功能将T迁移到SDN控制器Ci（第3行），计算每个SDN控制器（第5行）的新计算负载和迁移成本migrationSize（第6行），定义为需要的交换机数量迁移。



从集群T开始,该函数构造由所有与T相邻的SDN交换机组成的setneighborsT。SDN交换机Si是T的邻居，如果∃Sj∈T:：l（Si，Sj）≠0，l（Sj，Si）≠0。然后函数选择使新创建集群的相对密度[11]最大化的邻居。这种相对密度最大化背后的原理是，只有具有强连接的SDN交换机才应该分组到相同的集群中。然后将集群作为一个整体在控制器之间迁移，以降低控制器的总体计算复杂度。

定义8：相对密度是内度与入射边数之比，即



在给定备选向量的情况下，评估移民备选方案选择其中最优的备选方案([To,Target SDN controllero])，从而优化以下评估方法之一：

最小化最大控制器负载：



最小化控制器负载的总和：



积分-在拥塞的情况下，最大限度地增加与控制器负载配置的距离：



用[LC1，...，LC|C|]表示在BalCon之前出现拥塞时，控制器的矢量加载，并且函数D（u，v）定义如下：

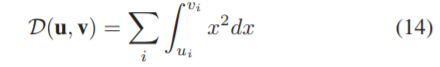
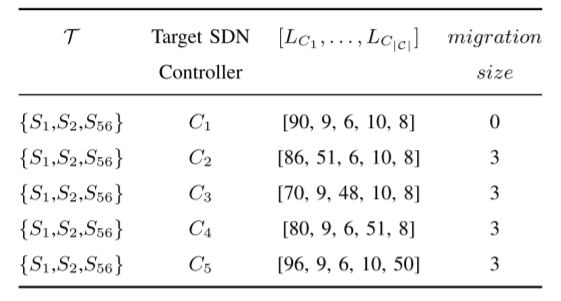


表1

BALCON算法在具有60个交换机、5个控制器和一个集群T={S1,S2,S56}的拓扑结构中执行的备选方案示例



第五章 迁移中断和BalConPlus

5.1 迁移中断

基于控制器负载动态地在控制器之间迁移交换机，可以平衡控制器负载，从而缓解拥塞。但是，在迁移过程中，一些交换机可能无法及时处理新的连接，这称为迁移中断[5]、[7]。

Dixit等人[5]、[7]提出了一种交换机迁移协议，该协议可以在不破坏活性、安全性和串行性的前提下，在两个控制器之间安全地迁移交换机。

迁移协议如图4所示，其中一个交换机从控制器1迁移到控制器2需要四个阶段。

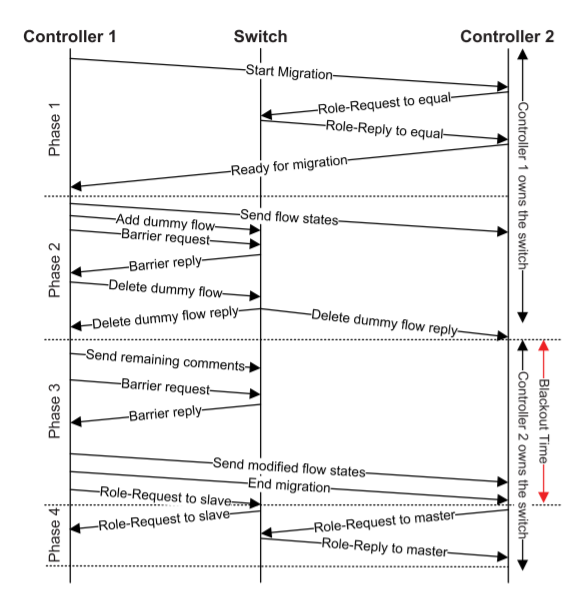


图4 将交换机从控制器迁移到另一个控制器的协议

在阶段1中，控制器1向控制器2发送开始迁移消息，控制器2在接收到消息时将其角色改变为相等，这意味着它现在可以从交换机接收消息，但是不能处理它们。然后，控制器2将立即向控制器1发送Ready for migration，完成阶段1。

在阶段2中，控制器1首先将交换机的状态信息发送给控制器2，使其能够从迁移后控制器1离开的地方接管。同时(与状态信息传输并行)，控制器1将向交换机安装一个虚拟流条目，然后删除它。这样做的目的是从交换机触发一个虚拟流删除响应，该响应被发送给两个控制器，以向它们发出迁移事件的信号。在此事件之后，所有的处理和决策将由控制器2负责，而控制器1将忽略来自交换机的任何消息。这就结束了第2阶段。

在阶段3中，尽管控制器2拥有交换机的控制权，但它还不能将任何流条目安装到交换机。这是因为控制器1可能正在处理未完成的任务。控制器2需要等到完成这些未完成的任务并安装相应的流条目才能将流条目安装到交换机。同时，控制器2接收的所有消息都将被缓冲。完成所有未完成的任务后，控制器1将通过向交换机发送屏障请求以清除所有未完成的flow-mod消息，确保相应的流条目成功安装在交换机中。在从交换机接收到屏障答复之后，控制器1需要将自阶段2开始以来的所有已修改状态信息发送到控制器2，并通过发送结束迁移消息来结束整个迁移过程。

在阶段4中，控制器1将其角色更改为从属，而控制器2将其角色更改为主。 现在可以处理在阶段3中在控制器2处缓冲的所有消息。

根据上面的迁移过程，我们可以看到有一个与阶段3长度相等的迁移中断期。在此期间，控制器2不能立即处理来自交换机的包入消息，这可能会延迟新流的连接设置。我们的原型显示，停电周期可以高达370ms(详细信息见第7节)，大于[5]和[7]中报告的值50—100ms。这种较大的停电周期是由于释放开关控制的控制器严重过载造成的。因此，与空闲控制器相比，处理速度更慢。

370ms的停电时间显然太大，特别是对于数据中心的应用，通常需要数十μs的延迟。但是，正如[5]和[7]中所指出的，如果不需要可串行化属性（即，来自交换机的消息可以无序处理），则可以删除此中断期并建立新连接 没有经历额外的延迟。BalCon很适合这种情况。

如果网络需要序列化属性，基于BalCon的切换迁移可能会导致临时服务中断。为了解决这个问题，我们提出了BalCon的一个改进版本，名为BalConplus，以避免在迁移期间服务中断。

5.2 BalConplus

BalConPlus的主要思想是暂时引导新到达的流量远离要迁移的交换机，因此它们的流设置不会受到迁移的影响。为了实现这一目标，BalConPlus对基线BalCon进行了两次更改。

（1）当BalCon选择要迁移的交换机时，第一个更改适用。为了确保始终存在绕过要迁移的交换机的备用路径，BalConPlus为迁移交换机的选择添加了新约束：所选交换机的（假设的）移除不应破坏网络连接。此更改仅需要对算法1中的ComputeStartingSwitchesList和IncreaseCluster稍作修改。在准备可以在ComputeStartingSwitchesList中迁移的单个SDN交换机列表时，我们会排除那些可能导致网络从网络中删除的网络脱节的列表。考虑图1中的网络，可以在ComputeStartingSwitchesList中考虑所有交换机，因为删除它们中的任何一个都不会破坏网络连接。当我们逐渐扩充IncreaseCluster中的候选迁移交换机集时，我们会排除那些在删除后可能会破坏网络的交换机。考虑图1中的相同示例，不能同时选择S5和S7进行迁移，因为从网络中删除它们会将网络分为两部分。但是，S5或S7可以单独迁移，因为删除其中任何一个都不会破坏网络。

（2）当BalCon对迁移过程中到达的新流进行路由计算时，第二个更改适用。 BalConPlus将引导新流程绕过迁移交换机。这可能会略微增加某些流的转发路径的跳数;但与数百ms的迁移中断期相比，由于路径较长而导致的稍大的转发延迟变得微不足道。迁移完成后，新流将在其最佳路径上路由。考虑图1中的示例，如果我们想要将S5从控制器C2迁移到C1，则在交换机迁移期间从H2到H6的新流应该路由到不同的路径（例如H2→S2→S4→S7→S6→H6）以绕过S5。

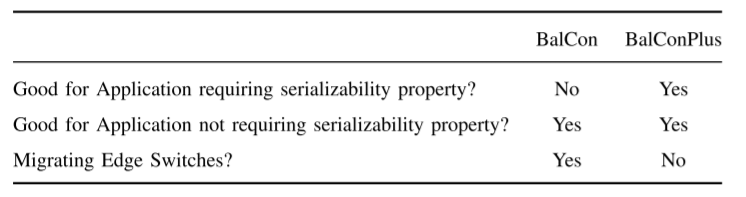
注意，基于上述两个变化，BalConPlus将不选择用于迁移的边缘交换机（即，直接连接到主机的那些）。因为如果选择边缘交换机进行迁移，则其连接的主机将不可避免地与网络断开连接并导致临时服务中断。使用图1中的示例，不应选择交换机S2进行迁移，因为在迁移中断期间它会暂时断开主机H2与网络的连接。

这里我们称之为未直接连接到主机核心交换机的交换机。在许多网络拓扑中，例如数据中心网络中的FatTree拓扑，我们通常在横向扩展结构中拥有比边缘交换机更多的核心交换机。因此，迁移核心交换机提供足够的灵活性来调整控制器之间的工作负载。

5.3 BalCon Vs. BalConPlus

表2给出了BalCon与BalConPlus的比较

表2 BalCon与BalConPlus的比较



第六章 模拟性能评估

BalCon和BalConPlus已经使用Matlab R2015a 64bit for Linux实现。 模拟使用配备英特尔酷睿i5-3340@3.10 GHz和8 GB 1600 MHz DDR3 RAM以及OS Linux Mint 17的PC进行。两种方案都有类似的模拟结果，由于空间不足我们只提供 BalCon的模拟结果。 BalConPlus的性能以及BalCon和BalConPlus之间的比较将在第七节中基于我们构建的原型测试平台进行介绍。

6.1 BalCon的动态场景有效性

在这里，我们修复BalCon参数（mcs，mssls和EvaluationMethod）并随时间演变网络，以便在（模拟）运行时网络操作期间显示BalCon的有效性。我们模拟了表III中所示的4种不同的网络拓扑，改变了边缘核心（dEC）和核心核心（dCC）节点的连接程度。特别地，dEC表示每个边缘节点对核心节点的连接数，而每个dCC表示每个核心节点对其他核心节点的连接数。为了执行动态场景模拟，我们实现了一个例程，该例程在泊松过程之后生成边缘节点处的流到达和离开。对于每个呈现的拓扑，我们使用泊松过程生成器的不同种子运行200个不同的模拟。每次运行都模拟2000s的网络运行时操作。使用公式13（积分）作为EvaluateM迁移备选方案中的EvaluationMethod，使用起始开关列表大小mssls=20和最大簇大小mcs=20来设置BalCon。

图5显示了由9个边缘节点（蓝色），5个核心节点（灰色）和3个控制器组成的拓扑。dEC=1表示每个边缘节点连接到单个核心节点，而dCC=全网状，因为核心节点形成全网状网络。

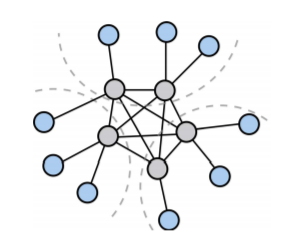


图5 网络拓扑的例子，有9个边缘节点(蓝色)，5个核心节点(灰色)，3个控制器，dEC=1,dCC=全网格

图6显示了在Topology1模拟过程中5个控制器的计算负载（0表示根本没有拥塞，100表示过载）。绿线表示控制器C5的拥塞程度。一旦达到阈值L=90，就使用起始开关列表大小swlsm=20和最大簇大小msc=20来触发BalCon.BalCon的不同例程用黑色虚线椭圆表示。

BalCon表现良好：4个BalCon实例期间的最大计算负荷平均减少15％，每个例程平均迁移2.4个交换机。计算时间为0.69秒。计算负荷的方差在每个例程平均减少66％。在这种情况下，BalCon可以有效地平衡计算负荷并解决控制器上的过载问题，几乎没有交换机迁移。

图7清楚地显示了BalCon算法与使用相同流量模式的交换机到控制器的静态分配相比的性能优势。图7（a）显示了没有负载平衡的5个控制器的计算负载，即静态分配，而图7（b）是实现BalCon的情况。正如我们所观察到的，BalCon在运行期间维持控制器负载低于阈值，而在静态分配情况下，拥塞负载超过阈值（90）50％。表3中具有不同拓扑的其他设置显示与图7类似的结果。

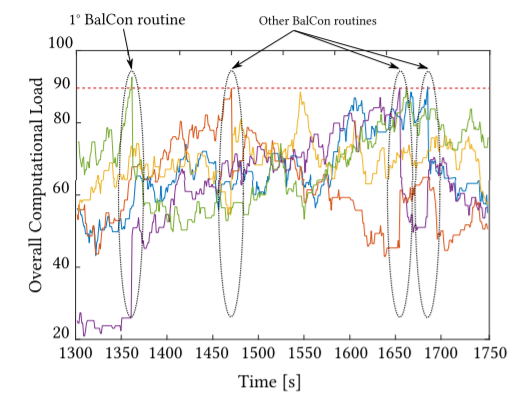


图6 在Topology1和seed=1的情况下，动态场景中5个控制器的计算量以及BalCon算法在仿真中的效果。蓝线是LC1，红线是LC2，黄线是LC3，紫线是LC4，绿线是LC5

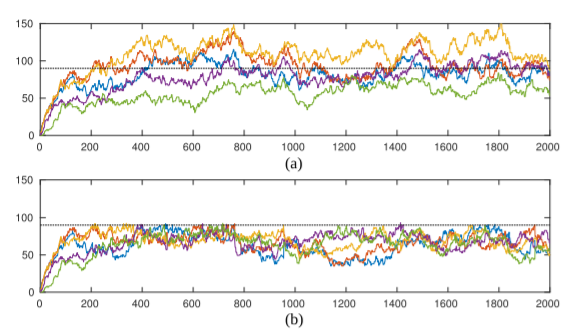


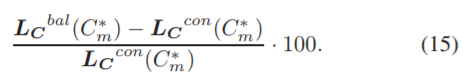
图7 用Topology3比较静态分配(a)和动态分配BalCon(b)的计算负载

6.2 静态场景

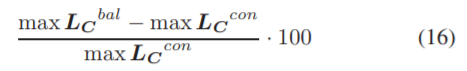
在Static Scenario模拟集中，我们修复了时间瞬间（发生拥塞时），并且我们改变BalCon参数以显示参数如何影响BalCon的性能。我们改变了mssls，mcs和EvaluateM迁移替代方法的方法。我们模拟了表III中所示的4种不同的网络拓扑。对于每个拓扑，我们综合生成500个不同的拥塞流量配置，其中一个控制器拥塞。对于每个拥塞流量配置，我们运行多个BalCon算法实例，其中mssls={3,5,10,20}和mcs={3,5,10,20}。

对于每次模拟，我们评估了不同的性能指标。设LC=[LC1，...]向量表示控制器负载，LCcon控制器负载在BalCon应用之前出现拥塞，LCbal负载BalCon例程之后。

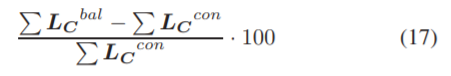
定义9：让拥塞的控制器Cm=argmaxLCcon和拥塞的控制器负载LCcon（Cm）。我们将减少拥塞控制器负载（％）定义为：



定义10:减少最大控制器负载(%)：



定义11:控制器负载(%)：



定义12:减少方差负载(%)：

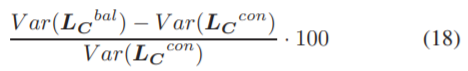


图8显示了使用T opology1和minMax作为EvaluationMethod改变mssls和mcs的不同版本BalCon的性能。首先，我们考虑黑条，代表参数的选择[mssls，mcs]=[3,3]。我们观察到拥堵控制器负载减少了12.55％（图8（a）），最大控制器负载减少了11.32％（图8（b）），控制器负载总和几乎可以忽略不计（图8（c）），方差减少的47.10％（图8（d））。我们还观察到我们获得1.37开关的平均迁移尺寸（图8（e））和平均BalCon计算时间0.13s（图8（f））。现在考虑其他条形，我们注意到性能高度依赖于参数。如果我们有更大的mssls和mcs，我们可以增加BalCon可能解决方案的搜索空间。这意味着更好的性能。事实上，如果我们考虑案例[mssls，mcs]=[20,20]，我们观察到之前描述的性能指标的显着增加。

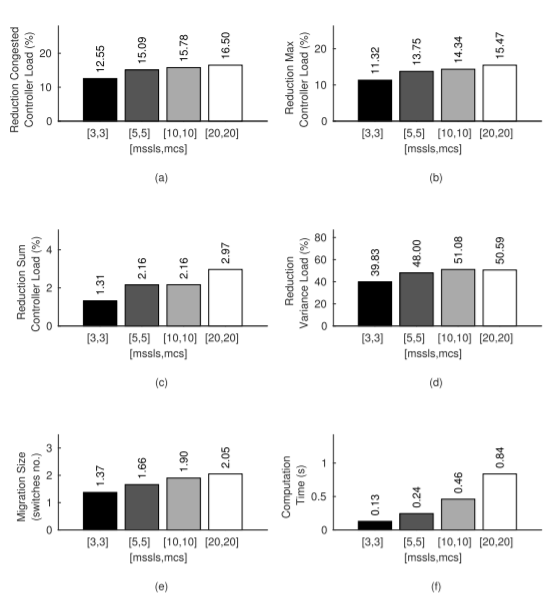


图8 不同版本的BalCon使用Topology1改变mssl和mcs的性能

使用大的mssls和mcs值，我们可以观察到迁移大小的小幅增加（从1.37到2.05）。BalCon非常快，实际上计算时间低于1s（0.84s），mssls值和mcs值更高。正如我们所观察到的，BalCon非常高效，计算时间短，所需的交换机迁移很少。

第七章 BALCON和BALCONPLUS的原型和实验结果

在本节中，我们将进一步详细介绍如何通过修改和添加组件到RYU控制器[8]来设计和实现BalCon控制器。

7.1设计

BalConController架构可以通过SDN控制器的NorthBound应用程序实现，并以分布式方式运行：只有拥塞的控制器才会根据更新的网络映射激活BalCon例程。特别是图9显示了BalConController中涉及的模块及其与SDN控制器中现有模块的关系。

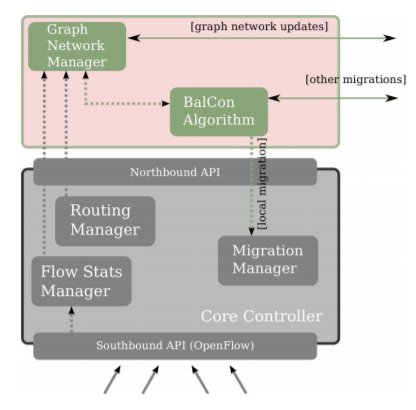


图9 BalConController架构

图形网络管理器是从Flow Stats Manager实体收集流到达统计数据和从路由管理器实体收集路由决策的实体，以便构建和更新图形表示G（S，E）的本地版本。然后，G（S，E）与其他SDN控制器不断更新（[图形网络更新]）。BalCon使用本地图中的更新信息，通过BalCon算法实体计算拥塞时的计算负载和迁移集群。在迁移的情况下，BalCon算法通知迁移管理器实体本地迁移以及其他迁移的其他控制器。

BalConController通过支持多控制器功能扩展了RYU功能：它可以在不同主机/网络上的多个实例上运行（每个控制器都有一个IP地址），每个实例管理整个网络的一部分。它还通过UDP套接字和Python中的自定义应用程序协议实现自制的控制器间消息传递。控制器间消息传递允许控制器自己交换不同类型的信息，例如图形网络管理器实体（例如，流量更新）和迁移管理器实体（例如，要迁移的交换机）。更可靠的解决方案可能是使用像Zookeeper或Hazelcast这样的分布式数据存储[7]。迁移管理器模块实现了[5]中提出的交换机迁移过程，该过程保证了每个交换机迁移的活跃性和安全性。最后，BalConController完全实现了BalCon算法，该算法可以基于不断更新的整个网络的统一视图，在每个SDN控制器中独立运行。

7.2实验装置

我们使用两个工作站来设置我们的实验测试平台：Workstation 1和Workstation 2，如图10所示。带有Intel Xeon Processor X5650和CentOS 7的工作站1主要用于测量Controller 4的硬件性能。开发为多线程Ryu应用程序，控制器4直接在Workstation 1上执行性能测量。 通过在独立的机器上分离，可以消除虚拟化的开销[12]和操作系统的调度。因此，可以在硬件上准确且直接地测量包括CPU负载和路径计算时间在内的性能。

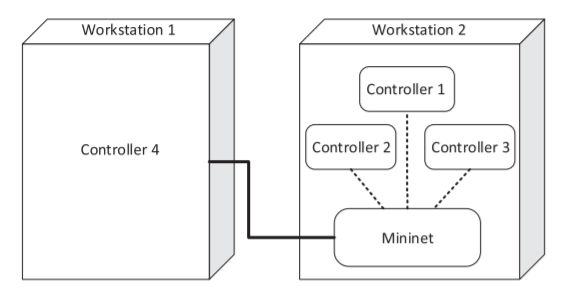


图10 实验环境的体系结构

另一方面，带有Intel Xeon CPU E5-2620 v4和Windows Server 2016 Datacenter的Workstation 2主要用于模拟在Mininet中运行的其他三个控制器和数据平面。三个控制器和Mininet在四个独立的虚拟机（VM）上运行。mininet的VM设置为16GB内存和12个处理器，而三个控制器的VM各有8GB内存和4个处理器。这些VM运行Ubuntu 16.04，并与物理网络接口桥接以连接到Controller 4。

我们的实验中使用的逻辑网络拓扑如图11所示，其中有11个主机和16个在Mininet中模拟的交换机。每个控制器最初控制四个开关。例如，控制器2在开始时控制S5，S6，S7和S8。我们将观察控制器之间的这些开关的迁移，以评估BalCon和BalConPlus的功能正确性。在以下实验中，每个控制器用于触发交换机迁移的阈值设置为CPU负载的60％。

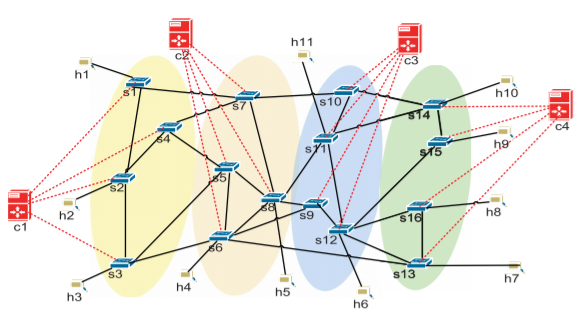
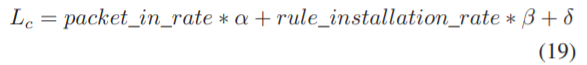


图11 实验中使用的逻辑网络拓扑

7.3 自动参数测量

BalCon和BalConPlus的操作需要知道α和β的值，基于此可以计算每个迁移候选者的预期控制器负载。 在这里，我们提出了一种测量α和β的自动方法。

控制器的负载可由下式描述：



其中δ是控制器中运行的基础（后台）工作负载。在等式中，Lc，packet\_in\_rate和rule\_installation\_rate可以由控制器实时测量。如果我们可以得到该等式的三个实例，则可以计算三个变量α，β和δ。在我们的实验中，我们在三个随机选择的时刻收集Lc，packet\_in\_rate和rule\_installation\_rate的三种组合并求解α，β和δ。这些计算出的参数应用于BalCon和BalConPlus，用于控制器负载建模和预测。

图12示出了测量的控制器负载（即，直接从CPU拉出负载）与计算的负载（即，基于求解的α，β，δ和使用等式19测量的packet\_in\_rate和rule\_installation计算的负载）。在图中，灰色曲线表示测量的瞬时控制器负载，这极大地受挫。为了获得稳定的α，β和δ值，我们使用16样本移动平均负载来解决它们。测得的移动平均载荷以黑色虚线表示。计算的载荷以实心黑色曲线显示。我们可以看到，即使由于[13]中提到的可能效果，实际CPU负载与我们的模型不完全匹配，BalCon和BalConPlus的趋势足够接近以预测控制器负载。在我们的实验中，α，β和δ的值会定期重新计算，因为它们可能会随时间变化。

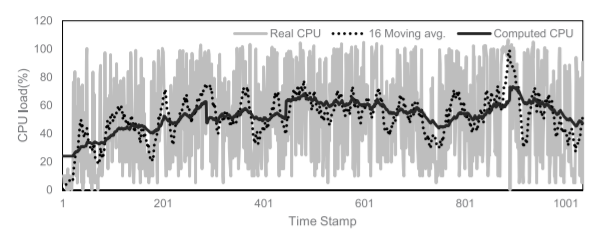


图12 控制器负载模型精度评估

7.4控制器负载均衡

首先，我们观察BalConPlus实现的CPU负载平衡（BalCon实现了类似的平衡结果，由于空间不足，我们只提供BalConPlus的实验结果）。我们使用合成流量和实际数据中心流量跟踪进行实验。

1）合成流量：我们首先在路径上从主机h1到h10以恒定速率生成新流量（h1→s1→s7→s10→s14→h10）。很容易看出四个控制器上的负载大致相似。然后，我们逐渐将从主机h4到h5的流量产生率从0增加到控制器2将开始拥塞的点。然后，交换机7将由BalConPlus（BalCon也）迁移到控制器1或控制器3。

实验结果如图13所示。从时间戳273开始，我们开始增加h4和h5之间的流量产生率。在时间戳536处，触发BalConPlus，并且将开关s7从控制器2迁移到控制器1.控制器2处的负载显着下降，而控制器1的负载仅略微增加。这表明由于基于我们模型的切换聚类效应，可以降低总体控制器负载。显示四个控制器的负载是平衡的。

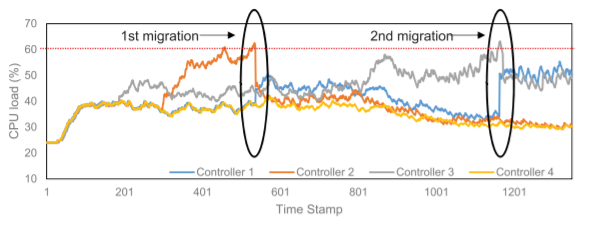


图13 通过BalConPlus在合成流量下实现控制负载平衡

然后我们逐渐将从主机h6到h11的流量产生率从0增加到控制器3开始拥塞的点。在实验中，交换机10将由BalConPlus（BalCon也）迁移到控制器1。在图13中的时间戳1168处，触发BalConPlus，并且控制器3的CPU负载降低到阈值以下。CPU负载在四个控制器之间没有完全平衡的原因是，如果这样做，可能会增加整体CPU负载。在这种情况下，BalConPlus选择迁移候选者，它不会增加太多的CPU负载，同时减少控制器3的负载。

2）真实数据中心流量跟踪：我们从纽约市教育部运营的数据中心收集了骨干链路上的实际流量跟踪。我们从跟踪中随机选择三个中继线，每个中继线持续10分钟，然后将它们馈送到图11中拓扑结构中的三个主机对（h1→h10，h4→h5和h6→h11）。

实验结果如图14所示。在时间戳466（第一次迁移），触发BalConPlus，并且开关11从控制器3迁移到控制器2.控制器3的负载下降，控制器2的负载增加。在时间戳506（第二次迁移），再次触发BalConPlus，并且开关6从控制器2迁移到控制器1.控制器2处的负载显着下降并且控制器1处的负载略微增加。这表明由于基于我们模型的切换聚类效应，可以降低总体控制器负载。四个控制器处的负载在时间戳607处开始增长。在时间戳743（第三次迁移），触发BalConPlus，并且交换机8从控制器2迁移到控制器4.控制器2处的负载下降并且控制器4处的负载增加。显示四个控制器的负载是平衡的。

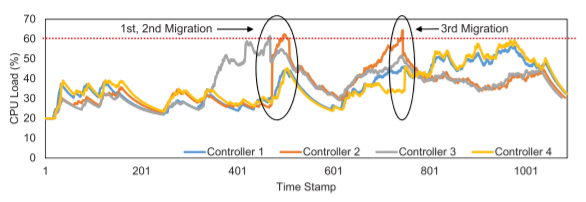


图14 在真实的数据中心流量下，通过BalConPlus控制负载均衡

7.5 BalCon Vs BalConPlus分组响应时间

与基线BalCon相比，BalConPlus消除了由交换机迁移中断引起的响应延迟，BalConPlus将在不通过迁移交换机的情况下找到替代路径。为了比较BalCon和BalConPlus中的数据包响应时间，我们测量从控制器接收数据包的时刻到相应的flow-mod发送到交换机的时刻的延迟。

我们首先测量当新流路径通过迁移交换机时迁移期间控制器4的响应时间。鉴于控制器4在专用服务器上运行，我们能够测量非常准确的延迟，大约370毫秒。然而，控制器13上的措施是困难的，因为这些控制器在与运行Mininet的VM共享相同物理工作站的VM中运行。虚拟机和Mininet的虚拟化以及操作系统的调度会对测量延迟造成巨大的干扰3秒，这太大并且会淹没迁移中断的延迟。在[14]中也观察到并报告了Mininet的类似延迟扰动。

为了通过虚拟化和OS调度引入干扰来显示迁移中断对控制器1—3的影响，我们故意将交换机迁移中断的长度（即图4中的阶段3）扩大到10秒。结果如图15所示，其中x轴是每个数据包输入的索引，y轴是数据包输入请求的响应时间。图15（a）显示BalCon中的许多分组输入请求由于在控制器1和控制器4处发生两次迁移而遭受大的响应延迟，并且在控制器3处发生一次迁移。在图15（a）中，控制器2具有较低的数据包到达速率，因此与控制器1或3相比，较少的数据包输入请求受到额外延迟的影响。另一方面，图15（b）显示了BalConPlus的数据包响应时间。我们可以看到，即使我们在控制器1上发生了两次迁移，并且在实验期间在控制器4上发生了三次迁移，但没有任何数据包输入请求会受到响应延迟的影响。

第八章 相关工作

参考文献[15][18]提出了SDN控制器中OS过程的多线程设计和并行化技术。Mallon等。[19]建议重新思考SDN控制器的设计，使其成为一个利用操作系统优化和现代硬件功能的低级软件。Renart等人。[20]通过将所有数据包检查和创建卸载到GPU来缓解SDN控制器的可扩展性问题。参考文献[21][27]研究了5G和运营商网络中SDN的控制器布局和QoS实施。

其他工作也通过使用多个主机来探索分布式控制器的实现：具有不同的角色[28][30]或具有相同的角色[2][4]。这些论文的主要焦点是解决跨分布式控制器实例的状态一致性问题，同时保持良好的性能。而[31][33]侧重于控制器放置问题，最小化控制器和开关之间的通信延迟。当前现有的分布式控制器解决方案仍然受到SDN交换机和控制器之间的静态映射的影响，限制了动态负载适配的能力。Dixit等人。[5]，[7]提出了一种弹性分布式控制器架构，能够使用现有的OpenFlow标准强制SDN交换机迁移到不同的控制器，而Bari等人。[34]尝试模拟开关控制器分配的问题，最小化控制器和交换机之间的通信成本（就跳数而言）。

Shah等人。[35]提出了一个名为Cuttlefish的SDN控制器框架，它可以自适应地将一部分应用程序状态卸载到本地控制器，以在控制平面上实现更高的吞吐量和更低的延迟。王等人。[36]提出了一种新的路由方案，以实现SDN中的控制器负载平衡和链路负载平衡。王等人。[37]，[38]提出了数据中心网络中的动态SDN控制器分配方案，其目标是在保持控制流量开销较低的同时平衡控制器负载。但是，它们的模型仅考虑由流请求处理引起的控制器负载，但忽略处理规则安装的负载。他们也没有考虑交换机迁移带来的开销。为了克服交换机迁移开销，Huang等人。[39]提出了交换机和控制器之间的调度层BLAC。BLAC拦截来自交换机的流请求并将它们分派到不同的控制器以实现控制器平衡。遗憾的是，该方案没有考虑交换机/控制器位置对性能的影响，并且引入的新调度层将增加交换机和控制器之间的通信延迟。Muthanna等人。[40]提出了一种动态聚类算法，用于平衡SDN网络中分布式控制器之间的负载。但是，该方案不考虑交换机迁移中涉及的开销，并且评估完全基于Matlab仿真。

第九章 总结

在本文中，我们介绍了两种SDN交换机迁移方案BalCon和BalConPlus，以实现SDN控制器之间的负载平衡，迁移成本较低。BalCon适用于网络不需要切换请求的串行处理的场景。对于其他情况，BalConPlus更适合，因为它不受交换机迁移中断的影响，并且不会导致任何服务中断。两种方案都经过模拟和实验的彻底评估。结果证明了两种方案实现SDN控制器负载均衡的实用性和有效性。在我们未来的工作中，我们计划将BalCon和BalConPlus的实施扩展到其他SDN控制器平台，如OpenDayLight，以研究控制器平台对BalCon和BalConPlus性能的影响。

参考文献

[1] M. Cello, Y. Xu, A. Walid, G. Wilfong, H. J. Chao, and M. Marchese, “Balcon: A distributed elastic SDN control via efficient switch migration,” in Proc. IEEE Int. Conf. Cloud Eng. (IC2E), Apr. 2017, pp. 40–50.

[2] T. Koponen et al., “Onix: A distributed control platform for large-scale production networks,” in Proc. 9th USENIX Conf. Operating Syst. Design Implement. (OSDI), Berkeley, CA, USA, Oct. 2010, pp. 1–6.

[3] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, “Logically centralized?: State distribution trade-offs in software defined networks,” in Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN), New York, NY, USA, 2012, pp. 1–6.

[4] A. Tootoonchian and Y. Ganjali, “Hyperflow: A distributed control plane for openflow,” in Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw. (INM/WREN), Berkeley, CA, USA, 2010, p. 3.

[5] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, “Towards an elastic distributed SDN controller,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN), New York, NY, USA, 2013, pp. 7–12.

[6] T. Benson, A. Akella, and D. A. Maltz, “Network traffic characteristics of data centers in the wild,” in Proc. 10th ACM SIGCOMM Conf. Internet Meas. (IMC), New York, NY, USA, 2010, pp. 267–280.

[7] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, “ElastiCon; An elastic distributed SDN controller,” in Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS), Oct. 2014, pp. 17–27.

[8] RYU Controller. [Online]. Available: <https://osrg.github.io/ryu/>

[9] N. McKeown et al., “OpenFlow: Enabling innovation in campus networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Apr. 2008.

[10] OpenFlow Switch Specification, Open Netw. Found., Menlo Park, CA, USA, Mar. 2014. [Online]. Available: <https://www.opennetworking.org>

[11] S. E. Schaeffer, “Survey: Graph clustering,” Comput. Sci. Rev., vol. 1, no. 1, pp. 27–64, Aug. 2007.

[12] S. Sudevalayam and P. Kulkarni, “Affinity-aware modeling of CPU usage for provisioning virtualized applications,” in Proc. IEEE 4th Int. Conf. Cloud Comput. (CLOUD), Jul. 2011, pp. 139–146.

[13] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, “Advanced study of SDN/openflow controllers,” in Proc. 9th Central Eastern Eur. Softw. Eng. Conf. Russia, Oct. 2013, p. 1.

[14] S.-Y. Wang, C.-L. Chou, and C.-M. Yang, “EstiNet openflow network simulator and emulator,” IEEE Commun. Mag., vol. 51, no. 9, pp. 110–117, Sep. 2013.

[15] Z. Cai, A. L. Cox, and T. S. E. Ng, “Maestro: A system for scalable openflow control,” CS Dept., Rice Univ., Houston, TX, USA, Tech. Rep. TR10-11, Dec. 2010.

[16] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, “On controller performance in software-defined networks,” in Proc. 2nd USENIX Conf. Hot Topics Manage. Internet, Cloud, Enterprise Netw. Services (Hot-ICE), Berkeley, CA, USA, Apr. 2012, pp. 1–6. [17] D. Erickson, “The beacon openflow controller,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN), New York, NY, USA, Aug. 2013, pp. 13–18.

[18] Floodlight OpenFlow Controller. Accessed: Apr. 24, 2014. [Online]. Available: <http://www.projectfloodlight.org/floodlight/>

[19] S. Mallon, V. Gramoli, and G. Jourjon, “Are today’s SDN controllers ready for primetime?” in Proc. IEEE 41st Conf. Local Comput. Netw. (LCN), Nov. 2016, pp. 325–332.

[20] E. G. Renart, E. Z. Zhang, and B. Nath, “Towards a GPU SDN controller,” in Proc. Int. Conf. Workshops Netw. Syst. (NetSys), Mar. 2015, pp. 1–5.

[21] A. Ksentini, M. Bagaa, and T. Taleb, “On using SDN in 5G: The controller placement problem,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2016, pp. 1–6.

[22] D. L. C. Dutra, M. Bagaa, T. Taleb, and K. Samdanis, “Ensuring end-toend QoS based on multi-paths routing using SDN technology,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2017, pp. 1–6.

[23] T. Taleb, M. Bagaa, and A. Ksentini, “User mobility-aware virtual network function placement for virtual 5g network infrastructure,” in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2015, pp. 3879–3884.

[24] M. Bagaa, T. Taleb, and A. Ksentini, “Service-aware network function placement for efficient traffic handling in carrier cloud,” in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), Apr. 2014, pp. 2402–2407.

[25] M. Bagaa, T. Taleb, A. Laghrissi, and A. Ksentini, “Efficient virtual evolved packet core deployment across multiple cloud domains,” in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), Apr. 2018, pp. 1–6.

[26] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, H. Flinck, and M. Namane, “Benchmarking the ONOS intent interfaces to ease 5G service management,” in Proc. IEEE GLOBECOM, 2018. [27] R. A. Addad, T. Taleb, and H. Flinck, “Towards modeling cross-domain network slices for 5G,” in Proc. IEEE GLOBECOM, 2018.

[28] S. H. Yeganeh and Y. Ganjali, “Kandoo: A framework for efficient and scalable offloading of control applications,” in Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN), New York, NY, USA, Aug. 2012, pp. 19–24.

[29] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, “Scalable flow-based networking with DIFANE,” in Proc. ACM SIGCOMM Conf., New York, NY, USA, Oct. 2010, pp. 351–362.

[30] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: Scaling flow management for high-performance networks,” in Proc. ACM SIGCOMM Conf., New York, NY, USA, 2011, pp. 254–265.

[31] B. Heller, R. Sherwood, and N. McKeown, “The controller placement problem,” ACM SIGCOMM Comput. Commun. Rev., vol. 42, no. 4, pp. 473–478, Sep. 2012.

[32] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, “Pareto-optimal resilient controller placement in SDN-based core networks,” in Proc. 25th Int. Teletraffic Congr. (ITC), Sep. 2013, pp. 1–9.

[33] Y. Jiménez, C. Cervelló-Pastor, and A. J. García, “Defining a network management architecture,” in Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP), Oct. 2013, pp. 1–3.

[34] M. F. Bari et al., “Dynamic controller provisioning in software defined networks,” in Proc. 9th Int. Conf. Netw. Service Manage. (CNSM), Oct. 2013, pp. 18–25.

[35] R. Shah, M. Vutukuru, and P. Kulkarni, “Cuttlefish: Hierarchical SDN controllers with adaptive offload,” in Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP), Sep. 2018, pp. 198–208. [36] H. Wang, H. Xu, L. Huang, J. Wang, and X. Yang, “Load-balancing routing in software defined networks with multiple controllers,” Comput. Netw., vol. 141, pp. 82–91, Aug. 2018. [37] T. Wang, F. Liu, J. Guo, and H. Xu, “Dynamic SDN controller assignment in data center networks: Stable matching with transfers,” in Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM), Apr. 2016, pp. 1–9.

[38] T. Wang, F. Liu, and H. Xu, “An Efficient online algorithm for dynamic SDN controller assignment in data center networks,” IEEE/ACM Trans. Netw., vol. 25, no. 5, pp. 2788–2801, Oct. 2017.

[39] V. Huang, Q. Fu, G. Chen, E. Wen, and J. Hart, “BLAC: A bindingless architecture for distributed SDN controllers,” in Proc. IEEE 42nd Conf. Local Comput. Netw. (LCN), Oct. 2017, pp. 146–154.

[40] A. Muthanna et al., “SDN multi-controller networks with load balanced,” in Proc. 2nd Int. Conf. Future Netw. Distrib. Syst., Jun. 2018, p. 57.