计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(11):82−88 [doi: 10.15888/j.cnki.csa.006140]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于自优化的 SDN 交换机动态迁移机制①

童俊峰, 闫连山, 邢焕来, 崔允贺

(西南交通大学 信息科学与技术学院, 成都 611730)

摘　要: 为提高 SDN 控制器的使用效率以及多控制器之间的负载均衡度, 对多控制器的部署问题进行了研究, 并提出了一种交换机动态迁移机制. 该动态迁移机制基于周期性运行的自优化的算法实现, 按照控制器的部署情况, 将网络划分成多个域, 通过分析各域内相关参数, 分别找出负载最高和最低的控制器节点, 并根据控制器负载和交换机请求率快速选择出最佳的迁移交换机和迁移目的地. 控制器的负载均衡度、交换机请求的处理时延和算法的复杂度是算法设计中所考虑的主要因素. 该算法的优点在于通过局部的动态调整实现了对 SDN 控制层的灵活管理. 仿真结果表明, 基于自优化的交换机动态迁移方案能够有效提高多控制器间的负载均衡度, 减小流请求的处理时延, 同时将运算复杂度保持在一个相对合理的水平.

关键词: 多控制器; 负载均衡; 交换机; 动态迁移; 自优化

引用格式: 童俊峰,闫连山,邢焕来,崔允贺.基于自优化的 SDN 交换机动态迁移机制.计算机系统应用,2017,26(11):82–88. http://www.c-s-a.org.cn/1003-3254/6140.html

Self-Optimizing Mechanism for Dynamic Switch Migration in SDN

TONG Jun-Feng, YAN Lian-Shan, XING Huan-Lai, CUI Yun-He

(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611730, China)

Abstract: In order to make full use of the resource of SDN controllers, as well as to improve the load balance degree among those controllers, a dynamic switch migration mechanism is proposed in this paper. The proposed dynamic migration solution is designed based on the Self-Optimizing Mechanism (SOM). It divides the network into several domains according to the deployment of SDN controllers. By comparing the relevant parameters of each domain, the proposed mechanism can quickly select appropriate target switches and migrating destinations. The load balance of multi-controller, flow latency and algorithm complexity are the main factors of the algorithm. The advantage of the algorithm is that it can flexibly manage the SDN control plane by local dynamic adjustment. Simulation results verify that the proposed mechanism can enhance the balance among controllers, and reduce the flow setup latency, while the computation complexity during the migrating process is kept at a reasonable level.

Key words: multi-controller; load balance; switch; dynamic migration; self-optimizing

软件定义网络 (Software Defined Network, SDN) 是一种新型网络架构. 该架构实现了网络中控制层和数据层的分离, 极大地提高了网络的可编程性和可管理性[1,2]. SDN 架构中控制器负责控制其域内的所有交换机, 处理来自交换机的路径请求并且下发流表进行管理, 是 SDN 的核心设备. 由于控制器承担大量的处理任务, 而受限于单个控制器的处理能力, 在大规模网络中, 往往需要部署多个控制器来满足管理需求. 多控制器部署的一般方法是将网络划分成多个域, 每个域由一个控制器管理, 不同域之间的控制器通过通

① 基金项目: 国家自然科学基金 (61401374); 国家铁总重大项目 (2016X008-D)
收稿时间: 2017-03-26; 修改时间: 2017-04-26; 采用时间: 2017-04-27

信协议完成信息交互. 这种部署方式在一定程度上能够降低控制器的负载, 但由于缺乏调整能力, 当网络中出现流量变动时, 传输延迟就会变大, 导致控制器效率降低. 具体表现为不同控制器之间的负载可能会出现较大的差异, 从而使高负载的控制器的对数据流请求的处理能力降低, 而低负载的控制器资源又不能得到充分使用.

在多控制器部署这一问题上, 国内外已经提出多种解决方案. 文献[3]对网络中需要的控制器数量及部署的位置进行了研究. 文献[4]以优化多控制器负载为出发点, 提出了基于 Capacitated K-Center 的解决方案. 由于这些方案是基于静态的部署, 所以仍然无法满足网络中的动态流需求. 文献[5]提出一种更为灵活的控制层方案, 并且改进了 Openflow 协议[6], 使交换机可以在不同控制器之间迁移. 该方案只研究了迁移的技术实现, 并未给出详细的迁移策略. 文献[7]建立了网络的重新分配机制, 实现了控制器动态部署. 文献[8]通过竞争匹配, 实现了控制器和交换机的最优分配. 但对于网络进行整体规划和重新分配, 算法的运算量和数据的存储量都很大, 对网络本身的稳定性也有一定的负面影响, 难以在大规模网络中应用.

为了更好地解决多控制器部署的问题, 本文提出了一种局部自优化的动态迁移方案. 该方案在原有的静态部署方案之上, 通过对比各控制器之间的负载情况, 迁出部分高负载控制器所管理的交换机, 从而达到改善控制器负载均衡度的效果. 该方案所设计的交换机的迁移方式具有周期性和自发性, 且每个控制器只负责收集和计算本域内的相关数据, 从而减小了迁移过程中算法的复杂度. 同时, 由于负载均衡度得到提升, 使相对空闲的控制器资源得到合理的利用, 从而降低高负载控制器响应交换机请求的时间, 提高了网络的整体性能.

1 多控制器部署模型

1.1 SDN 模型

本文以 F 来表示 SDN 的网络配置. 假定网络中共有 N 个交换机和 M 个控制器, 其集合分别表示为 $\Phi=\{s_1,s_2,\cdots,s_N\}$和$\Psi=\{c_1,c_2,\cdots,c_M\}$. A 表示控制器的容量, 其大小由 CPU、带宽及内存等因素决定的[9], 本文将其定义为单位时间内可以处理的交换机请求的数目. 控制器通常不能满负载运行[10], 因此需要给每个控制器设定一个负载上限因子, 负载比例超过上限因子值表示该控制器处于过载运行状态. 本文中用 L 表示上限因子, 其取值范围在 0 到 1 之间, 各参数的定义如表 1 所示.

表 1 基本定义

符号	定义
s_i	第i个交换机
c_m	第m个控制器
λ_{im}	单位时间内s_i向c_m发送的请求数, 即请求率
A_m	c_m的容量, 单位时间内可以处理的请求总数
Θ_m	c_m的负载, 单位时间内实际处理的请求数
L_m	c_m的负载上限因子
$R(i,j)$	从s_i到s_j的请求率
S_{c_m}	c_m所管理的所有交换机的集合

SDN 中数据包的转发过程可以简化成三部分, 即交换机请求、请求处理和流表下发. 其中, 交换机请求和流表下发的过程与控制器和交换机的连接模式有关. 多控制器部署一般采用带内模式[11], 如图 1 所示, 控制器仅与部分交换机相连. 当 s_i 收到一个发往相邻域的交换机 s_k 的数据包时, 首先会在其缓存区进行流表匹配. 若匹配不成功, s_i 会向控制器 c_m 发送一个 *packet-in* 报文. c_m 将根据报文内的信息计算出合适的转发路径, 并将其封装在流表中. 假设该数据包是沿着 $s_i \rightarrow s_j \rightarrow s_k$的路径传输的, 由于 s_i 和 s_j 在同一个域内, c_m 将封装好的流表下发给 s_i 和 s_j. 通过两次转发后, 该数据包最终到达 s_k. s_k 与 s_i 在不同域内, 且属于控制器 c_n 管理, 于是继续执行上述的请求步骤, 由 c_n 将封装好的流表下发给 s_k, 最终 s_k 根据流表中的指令将数据包发送至目的主机. 如图 1 所示, 在上述转发报文过程中产生了两次路径请求 (如图中①、④) 操作及三次流表 (如图中②、③、⑤) 下发操作.

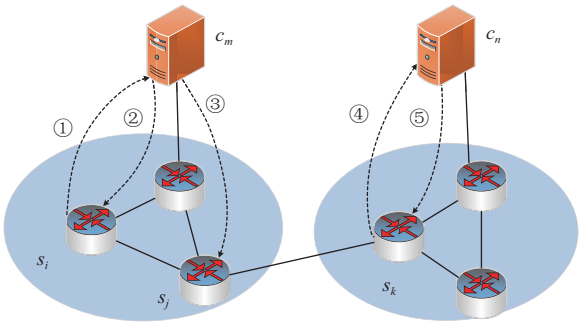

图 1 数据包跨域转发示意图

基于上述分析, 假设单个交换机 s_i 所发送的路径请求率为 λ_{im}, 则 λ_{im} 可表示为:

$$\lambda_{im}=\sum_{s_j\in\Phi}R(i,j)+\sum_{s_{j'}\in\Phi,s_{j'}\notin S_{cm}}R(j',i) \tag{1}$$

控制器 c_m 的负载可以表示为:

$$\Theta_m=\sum_{s_i\in S_{cm}}\lambda_{im} \tag{2}$$

其中, S_{cm}表示由控制器 c_m 所管理的所有交换机的集合, $R(i,j)$ 表示单位时间内交换机 s_i 向 s_j 发送数据包所生成的新路径请求数. 根据数据包的跨域转发原理, 当交换机接收来自同域内其它交换机转发的数据包时, 只接收下发的流表而不会生成新的路径请求 (如图中③), 而接收来自邻域的数据包时, 则会生成新的请求 (如图中④、⑤). 所以, 利用公式 (1) 计算请求率时要区分 $s_{j'}$与 s_i 不在同域内的情况.

在数据包的转发过程中, 交换机产生的路径延迟一般为微秒 (μs) 级别, 而控制器处理时延通常为毫秒 (ms) 级别[8]. 因此在计算总时延时, 路径延迟可以忽略不计. SDN 控制器对 *packet-in* 报文的处理, 是一个反馈型 $M/M/1$ 排队过程[8]. 假设网络中交换机的路径请求服从 *Poisson* 分布[9], 利用 *Little* 法则, 可以得到单个控制器 c_m 对路径请求的平均处理时间:

$$T_m=\frac{K^2}{2(A_m-\Theta_m)} \tag{3}$$

其中, K 表示网络中的节点总数, 表示处理时间受到网络规模的影响.

1.2 控制器资源效用

为了作量化处理, 本文以控制器效用作为衡量控制器资源利用状况的指标. 控制器的效用定义为交换机的请求率与控制器容量之间的数值关系. 假设交换机 s_i 位于控制器 c_m 管辖域内, 则 s_i 对控制器 c_m 容量资源的利用率可以表示为:

$$u_{im}=\frac{\lambda_{im}}{A_m},\lambda_{im}<A_m \tag{4}$$

显然 λ_{im} 越大, 则 s_i 对 c_m 的容量资源利用率越高. 将同域内所有交换机的利用率相加并取对数, 能够得到控制器 c_m 的效用值 U_m 以及网络中所有控制器效用的总和 U_{total}:

$$U_m=\ln\left(\frac{\Theta_m}{A_m}\right),\Theta_m<A_m \tag{5}$$

$$U_{total}=\ln\frac{\prod_{x\in\Psi}\Theta_x}{\prod_{y\in\Psi}A_y} \tag{6}$$

U_m 值为负数, 当控制器负载越高时, 其效用值也越大. 假设一定时间内网络中交换机的请求率之和 (即控制器负载之和) 保持不变, 通过调整交换机和控制器的映射关系 (即交换机的动态迁移), 能够改变控制器效用的总和.

2 算法设计

为提高 SDN 多控制器的总体效用以及控制器之间的负载均衡度, 本文提出一种交换机动态迁移机制, 即 Self-Optimizing Mechanism (SOM). 首先, 将迁移过程分成准备和执行两个阶段. 在准备阶段, 控制器通过记录单位时间内收到的请求数及其来源, 计算并存储各个交换机的请求率, 生成各项参数. 然后利用这些数据建立优先级列表. 建立优先级列表的目的是对交换机进行排序, 以挑选出最佳的待迁移交换机. 在执行阶段, 根据交换机的优先级列表确定迁移对象, 同时匹配最佳接收控制器, 并将交换机迁移到该控制器管理域内.

2.1 准备阶段

在该阶段, 控制器会定期获取网络的相关参数, 包括链接、实时请求率、控制器容量及效用值等. 通过这些数据设置控制器和交换机的优先级列表.

定义. 最高负荷控制器. 指负载最高的控制器, 用 c_{max} 表示. 单个控制器的负载和其效用值相关. 即效用值越大, 表明控制器负载越高. 所以最高负荷控制器 c_{max} 满足以下条件:

$$U_{\max}\geq\max_{c_m\in\Psi}U_m \tag{7}$$

同一时间内, 最高负荷控制器 c_{max} 只有一个. 准备阶段分两步执行, 具体步骤如下:

步骤 1. 选择迁出控制器. 假设网络中的控制器容量都相同, 并且单位时间段内, 各交换机的请求率保持不变. 根据均值定理和公式 (5)、公式 (6) 可以推导出: 当控制器之间负载完全相等时, 网络的总效用值可以达到最大值. 虽然真实环境中效用不可能完全相等, 但可以通过缩小高效用和低效用控制器之间的差值而起到优化作用. 这就需要把高负载控制器所管理的交换机部分迁移至低负载控制器的域内. 因此, 这一步骤将

挑选出最高负荷控制器作为迁出控制器.

步骤 2. 根据选出的控制器, 对其域内的交换机进行优先级排序. 考虑到交换机的迁移对网络自身的数据传输等业务会产生一定影响, 因此被迁移的交换机请求率不宜过大. 所以当迁移的交换机对其控制器上的资源利用率较小, 且与控制器之间的跳数较大时, 可以将迁移对整个网络的造成的扰动影响降至最低. 所以交换机的优先级排序将按照以下原则进行:

$$p_{im}=\exp\left(\frac{u_{im}}{d_{im}(1-u_{im})}\right) \tag{8}$$

公式 (8) 综合考虑了交换机 s_i 的请求率和与控制器 c_m 之间的跳数 d_{im}, 计算得出的 p_{im} 值越大, 表示交换机的优先级越高且越容易被迁移出去.

2.2 执行阶段

目标交换机被挑选出后, 紧接着执行迁移操作. 首先要选取最佳接收控制器. 接收控制器的选取需要符合效用值最大化的原则. 根据之前的分析, 效用值高的控制器应当迁移部分交换机到效用值低的控制器域内. 由于同时间段内, 网络中数据包的转发路径和交换机请求率是保持不变的, 因此能够对交换机迁移后控制器的负载进行估算. 假设在控制器 c_m 管理域内的交换机 s_i 需要被迁出, 则选取出的接收控制器 c_n 应满足以下要求:

$$\max_{c_n\in\Psi}(U_m-U_n) \tag{9}$$

$$U_m<U_n+u_{in} \tag{10}$$

$$A_nL_n>\Theta_n+\lambda_{in} \tag{11}$$

其中 u_{in} 和 λ_{in} 是通过计算 s_i 发送的路径请求而得出的迁移后 s_i 对控制器 c_n 的请求率和效用值. 在网络中寻找资源利用率最低的控制器, 以使公式 (9) 中目标值达到最大. 公式 (10) 确保迁移后控制器 c_m 和 c_n 的总效用值增加. 公式 (11) 确保迁移后 c_n 负载的增加值仍在允许情况范围之内.

2.3 解决方案

自优化机制及其算法会更改网络的原配置 F, 并最终产生一个新配置. 最高负荷控制器则需要与其它所有控制器对比之后才会被列出, 剩余的控制器作为潜在接收端. 代码中使用 C_X 来表示接收控制器集合.

算法按照一定的周期自动运行和终止. 在运行过程中, 首先确定最高负荷控制器 c_{max} 和最佳接收控制器 c_x, 并分别列出二者所管理的交换机集合 S_{max} 和 S_x, 令 S_{mig}=S_{max} 表示可迁移的交换机集合 (第 5-10 行).

执行 repeat 迭代, 从集合 S_{mig} 中按规定找出待迁移交换机 s(第 13 行). 检查迁移条件, 若满足公式 (10) 和公式 (11), 则执行迁移, 更新 S_x 和 c_m 并终止当前迭代 (第 14-17 行). 若 while 循环条件未发生改变, 则继续执行下一轮 repeat 迭代, 直到 $c_{max}\neq c_m$ 时, 表明最高负荷控制器的对象已经发生改变, 优化完成, 则当前周期内的 while 循环将会终止; 若公式 (10) 和公式 (11) 的条件不能满足, 表明 s 无法对外迁移, 需要从 S_{mig} 中移除 (第 19 行). 当检测到 S_{mig} 为空时, 表明 S_{mig} 集合中的交换机被全部移除, 控制器 c_{max} 管理域内的交换机已经无法迁移, 此时 while 循环将会终止 (第 9 行). 这时会出现控制器负载过高而交换机又无法迁移的不正常情况, 表明网络的整体负载过高, 现有的控制器已经无法满足服务需求, 需要通过部署新的控制器来解决.

算法. Self-Optimizing Algorithm

输入: 网络原配置 ***F***

输出: 新的网络配置 ***F***

```
1. c_max←最高负荷控制器
3. C_X←接收控制器集合
4. Begin
5.    c_m←c_max
6.    S_max←set of switches in c_max
7.    S_mig←set of switches to be migrated
8.    S_mig←S_max
9.    while c_max=c_m and S_mig≠Ø do
10.   select c_x as receiver from C_X by Eq.(9)
11.   S_x←set of switches managed by c_x
12.   repeat
13.   select switch s from S_mig by Eq.(8)
14.   if constraints in Eq.(10) and Eq.(11) are not violated then
15.      S_x←S_x ∪ {s}
16.      update (c_m)
17.      break
18.   end if
19.   S_mig←S_mig\{s}
20.   until S_mig=Ø
21. end while
22. update (F)
23. End
```

该算法的触发和终止都是按照预先设定的运行周期而自发进行的, 不需要设定其它条件, 即便迁移失败也不会陷入无限迭代过程. 每运行一次称为一个周期, 每个时间段内算法可自动运行若干次, 因此称为自优

化机制. 该算法有两个优点, 一是通过周期性的循环, 不断降低控制器的最高负载值, 从而使所有控制器的负载相对均衡; 二是局部调整的策略中交换机的信息只在本域内计算和保存, 领域之间只比较控制器的整体负载情况. 通过与文献[7,8]中将网络中控制器和交换机整体重新分配, 全体控制器和交换机均参与迭代的方案相比较, 本文的方案大大降低了数据的运算量和存储量, 减轻了控制器的决策负担.

3 仿真结果与分析

3.1 仿真参数设置

该部分通过仿真对文章提出的方案进行评估. 仿真采用的是数据中心中最常见的 Fat-Tree 拓扑, 利用仿真工具模拟数据包转发所产生的路径请求, 并根据文中的各项公式构建相应的数据收集、存储和决策制定模块. 为了接近商用数据中心的规模, 设置拓扑的 pod 数为 24. 为了便于测量和计算, 本文设定控制器的数量为 12 个, 且均被部署在不同 pod 内的主机上. 根据文献[12,13]的测量, 数据中心内的交换机请求率在高峰时期可达到 5000 K/s. 为了有效评估方案的可行性, 将仿真网络内的总请求率设定在 3000 K/s 至 4000 K/s 的范围内. 而单个交换机每秒产生的请求数及请求路径都是随机设定的, 这样能够更好地模拟出真实环境中交换机请求率波动的情况. 同时根据参考文献[14,15]的部分测量结果, 本文将控制器的容量定为 1000 K/s. 为了使不同控制器之间存在差异, 将每个控制器上限因子 L 设定为从 0.7 到 0.9 不等的随机值.

本文的仿真采用了两种不同的交换机请求率模型, 将其分别命名为 Model-I 和 Model-II. 如图 2 所示, 两种模型均被分成了 100 个运行时间段, 每段时间为一小时. 同一时间段内, 网络内交换机的总请求率保持不变. 其中, Model-I 中交换机请求率服从均匀分布, 变化区间为 3250 K/s-3500 K/s. 而 Model-II 中的总请求率在 3000 K/s 到 4000 K/s 之间规律性波动. 综合这两种模型, 可以比较全面地检验本文所提方案合理性和可行性.

3.2 结果分析

仿真中主要计算并记录三个指标值, 分别是交换机请求在控制器中的最大处理时延, 平均处理时延和控制器负载均衡指数. 时延是根据控制器的容量和交换机的请求率综合计算得来的. 其中, 最大处理时延是网络中控制器处理一条交换机请求所耗费的最长时间, 平均处理时延则是处理所有请求的平均时间, 这两个都是衡量网络性能的重要指标. 控制器的负载均衡指数是根据公式 (2) 得到所有控制器的负载值, 并计算其标准差得到的. 指数值越大, 表明多控制器之间的负载差异就越大. 仿真所执行的 SOM 动态迁移方案与静态部署 Capacitated K-Center (CKC) 方案[4]进行了结果的对比. 仿真过程中, 首先进行静态的控制器部署, 测定 CKC 方案中控制器的各项指标值. 再执行 SOM 动态迁移进行优化, 并记录优化后趋于稳定的各项指标值.

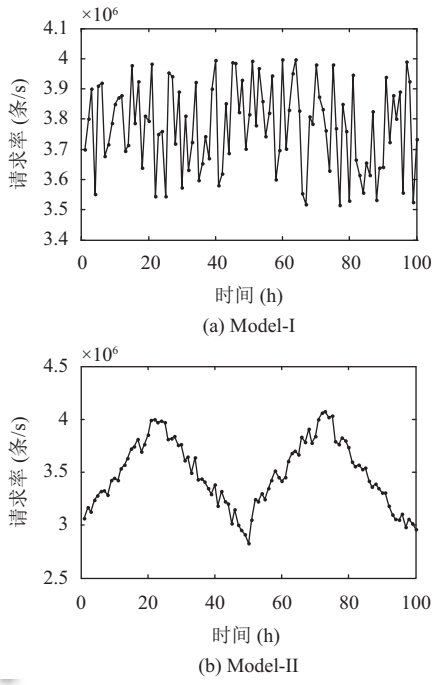

图 2 交换机请求率模型

图 3 和图 4 分别对应两种模型中的最长处理时延和平均处理时延. 结果显示, SOM 方案对最长处理时延的优化效果十分明显, 这是因为 SOM 在运行的过程中, 不断地降低控制器的最高负载值, 有效改善了部分控制器负载过高的情况, 使得最长处理时延大大降低. 平均处理时延也有所降低, 但因为控制器的数量是固定的, 可用的总资源有限, 所以当总的请求率大幅上升或下降时, 平均处理时延也会受到一定的影响. 需要指出的是, 当总体负载较高时, 平均处理时延的优化效果更为明显.

如图 5 所示, 通过比较 Model-I 和 Model-II 中的负载均衡指数值, 发现经过 SOM 方案优化后的负载均

衡指数在两种模型里都保持着较小的值, 而且振动的幅度明显小于请求率本身振动的幅度. 这表明基于 SOM 算法的动态迁移方案具有良好的稳定性, 能够有效调节控制器之间的负载, 使之最终达到相对平衡的状态.

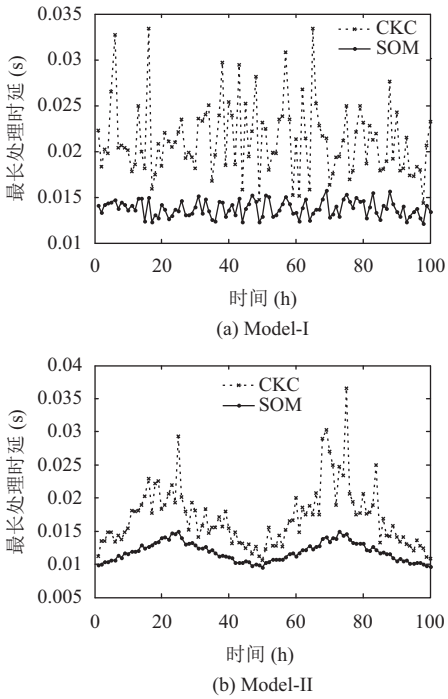

图 3 最大处理时延

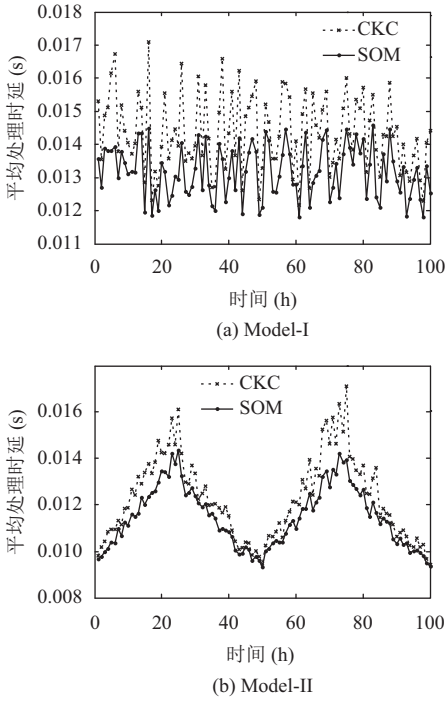

图 4 平均处理时延

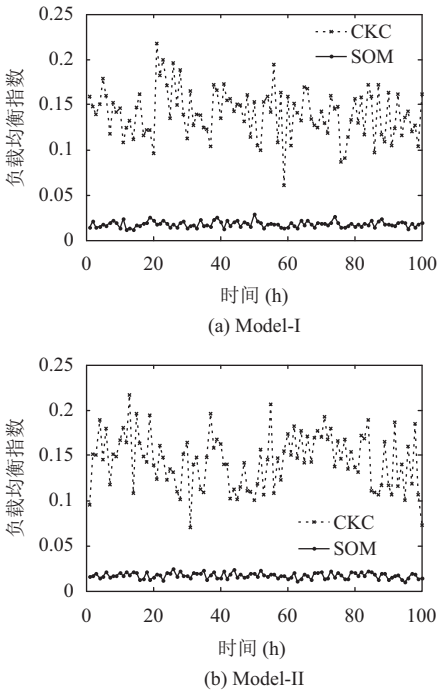

图 5 负载均衡指数

4 总结

针对 SDN 网络中多控制器负载不均衡的缺陷, 同时为了避免网络因重新新分配而产生的扰动和复杂计算, 改进了控制器和交换机的动态分配机制, 并且提出了基于自优化下的动态迁移方案. 该方案利用分布式的思想, 将整个网络划分成多个域, 以独立的域为单位制定局部的迁移策略, 这样仅用少量的计算量和数据存储量就可以实现对控制器和交换机之间映射关系的动态调整. 仿真结果显示, 引入自优化动态迁移机制后, 相对于静态部署, 网络的管理更加灵活, 控制器的负载更加均衡, 传输延迟得到有效改善.

参考文献

1 Xia WF, Wen YG, Foh CH, *et al*. A survey on software-defined networking. IEEE Communications Surveys & Tutorials, 2015, 17(1): 27–51.

2 Nunes BAA, Mendonca M, Nguyen XN, *et al*. A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications Surveys & Tutorials, 2014, 16(3): 1617–1634.

3 Heller B, Sherwood R, McKeown N. The controller placement problem. Proc. of the first Workshop on Hot Topics in Software Defined Networks. New York, USA. 2012. 7–12.

4 Yao G, Bi J, Li YL, *et al*. On the capacitated controller placement problem in software defined networks. IEEE Communications Letters, 2014, 18(8): 1339–1342. [doi: 10.1109/LCOMM.2014.2332341]

5 Dixit A, Hao F, Mukherjee S, *et al*. Towards an elastic distributed SDN controller. Proc. of the second ACM SIGCOMM workshop on Hot topics in software defined networking. New York, USA. 2013. 7–12.

6 Mckeown N, Anderson T, Balakrishnan H, *et al*. OpenFlow: Enabling innovation in campus networks. Acm Sigcomm Computer Communication Review, 2008, 38(2): 69–74. [doi: 10.1145/1355734]

7 Bari MF, Roy AR, Chowdhury SR, *et al*. Dynamic controller provisioning in software defined networks. Proc. of the 9th International Conference on Network and Service Management. Zurich, Switzerland. 2013. 18–25.

8 Wang T, Liu FM, Guo J, *et al*. Dynamic SDN controller assignment in data center networks: Stable matching with transfers. IEEE INFOCOM 2016-the 35th Annual IEEE International Conference on Computer Communications. San Francisco, CA, USA. 2016. 1–9.

9 Yao L, Hong P, Zhou W. Evaluating the controller capacity in software defined networking. Proc. of the 23th IEEE International Conference on Computer Communication and Networks. Shanghai, China. 2014. 1–6.

10 Krishnamurthy A, Chandrabose SP, Gember-Jacobson A. Pratyaastha: An efficient elastic distributed SDN control plane. Proc. of the third Workshop on Hot Topics in Software Defined Networking. New York, USA. 2014. 133–138.

11 Akyildiz IF, Wang P, Lin SC. SoftAir: A software defined networking architecture for 5G wireless systems. Computer Networks, 2015, (85): 1–18. [doi: 10.1016/j.comnet.2015.05.007]

12 Kandula S, Sengupta S, Greenberg A, *et al*. The nature of data center traffic: Measurements & analysis. Proc. of the 9th ACM SIGCOMM Conference on Internet Measurement. New York, USA. 2009. 202–208.

13 Benson T, Akella A, Maltz DA. Network traffic characteristics of data centers in the wild. Proc. of the 10th ACM SIGCOMM conference on Internet Measurement. New York, USA. 2010. 267–280.

14 Cheng GZ, Chen HC, Hu HC, *et al*. Dynamic switch migration towards a scalable SDN control plane. International Journal of Communication Systems, 2016, 29(9): 1482–1499. [doi: 10.1002/dac.v29.9]

15 Jarschel M, Lehrieder F, Magyari Z, *et al*. A flexible openflow-controller benchmark. Proc. of the 2012 European Workshop on Software Defined Networking. Washington DC, USA. 2012. 48–53.