一种基于 OpenFlow 的多路径传输机制

陈 鸣 胡 慧* 刘 波 邢长友 许 博
(解放军理工大学指挥信息系统学院 南京 210007)

摘 要：针对高连通度数据中心网络存在吞吐量低、网络负载均衡差等流量工程问题，该文提出一种基于 OpenFlow 的多路径传输(OpenFlow based Multipath Transmission, OFMT)机制。借助 OpenFlow 集中控制的优势，OFMT 为每条流精确计算传输路径，将网络中的流量优化分配到端到端的多条路径上，并通过周期性轮询和动态调度实现良好的负载均衡，进而提高数据中心网络吞吐量。实验结果表明，在相同的网络流量负载下，OFMT 与较典型数据中心传输协议相比，有效提高了网络的吞吐量，并降低了流完成时间。

关键词：数据中心网络；OpenFlow；多路径传输；负载均衡；流量工程
中图分类号：TP393 **文献标识码**：A **文章编号**：1009-5896(2016)05-1242-07
DOI: 10.11999/JEIT150928

An OpenFlow Based Multipath Transmission Mechanism

CHEN Ming HU Hui LIU Bo XING Changyou XU Bo
(*College of Command Information System, PLA University of Science and Technology, Nanjing* 210007, *China*)

Abstract: In order to solve the traffic engineering problems such as low throughput and poor load balancing in high connected data center networks, an OpenFlow based Multipath Transmission (OFMT) mechanism is proposed. By taking advantage of OpenFlow centralized control, OFMT calculates precisely the transmission path for each flow and assigns optimally the traffic flow in all transmission paths, it also executes periodic polling and dynamic scheduling mechanisms to achieve good load balancing and high throughput. The experimental results show that the network throughput is significantly improved in OFMT compared with typical data center transport protocols, and OFMT shortens the flow completion time under the same traffic workloads.

Key words: Data center network; OpenFlow; Multipath transmission; Load balancing; Traffic engineering

1 引言

随着云计算的蓬勃发展和广泛应用，作为信息服务基础设施的数据中心得到广泛的部署和使用。数据中心中的海量数据存储在分布式的服务器中，数据的迁移和一些带宽密集型应用(如 MapReduce)触发数据中心内部流量显著增加，而传统基于树状结构的数据中心网络存在连接带宽受限、网络鲁棒性差等问题，使得网络成为数据中心的性能瓶颈。Fat-Tree[1], VL2[2] Bcube[3]等高连通度网络拓扑目前已在数据中心中得到广泛使用，使得任何两台服务器间存在多条物理路径。然而，基于分布式网络资源分配方式的单路径 TCP(Single Path TCP, SPTCP)传输机制并无法有效利用多条端到端的传输路径，导致网络吞吐量低，负载均衡差等问题[4]。

针对上述问题，国内外研究人员提出了相应的解决方案。文献[5]提出等价多路径路由(Equal-Cost MultiPath routing, ECMP)的传输机制，通过对报文首部的五元组进行哈希运算，根据哈希值决定下一跳路径，以随机化方法使端到端的流量分布到不同传输路径上。文献[4]提出轮询多路径(Round Robin MultiPath, RRMP)，该算法从所有端到端路径中循环选择不同的传输路径。文献[6]提出虚拟局域网(Virtual Local Area Networks, VLANs)，将网络中不同的路径划分为不同的 VLAN，由一个离线控制系统计算并预先安装端到端的传输路径，实现流量随机负载均衡。文献[7]提出多路径 TCP (MultiPath TCP, MPTCP)，通过使用多网卡主机，为端到端的数据传输建立多条连接，将一条流分为多条子流进行传输，所有子流使用耦合的拥塞控制

收稿日期：2015-06-24；改回日期：2016-02-25；网络出版：2016-02-19
*通信作者：胡慧 huhui_email@126.com
基金项目：国家重点基础研究发展计划(2012CB315806)，国家自然科学基金(61379149)，江苏省科技计划(BY2013095-1-06)，江苏省自然科学基金(BK20140070)
Foundation Items: The State Key Development Program for Basic Research of China (2012CB315806), The National Natural Science Foundation of China (61379149), Jiangsu Province Project of Science and Technology (BY2013095-1-06), Natural Science Foundation of Jiangsu Province (BK20140070)

算法[8]。研究表明，以上方法没有实现较理想的负载均衡以及较高的网络吞吐量，主要存在以下的问题：首先，网络拓扑的高连通性未得到充分利用，导致网络带宽资源的浪费。其次，不合理的网络资源分配方式使得一些链路超额认购[7]成为瓶颈链路，而另一些链路负载却很轻。究其原因，这些算法仅掌握网络拓扑结构、网络流量分布等部分信息，缺乏全局网络状态的路由与调度，使网络的吞吐量与最优值存在较大的差异。例如，ECMP 与 VLANs 存在哈希碰撞而导致负载分配不均衡；RRMP 在计算路径时没有考虑当前链路负载；MPTCP 可以有效提高网络的吞吐量，但其多宿传输不仅使传输过程变得复杂，也增加了短流的完成时间[7]。如果能够获取有关网络拓扑、网络流量分布更全面的信息，就有可能得到更好的优化结果。

基于 OpenFlow[9] 的软件定义网络(Software Defined Networking, SDN)体系结构为数据中心流量工程问题提供了一种新的解决思路[10]。相比于传统的 TCP/IP 结构，SDN 采用集中控制方式，易于获取网络全局拓扑和流量分布信息，按优化决策向交换机下发流表以控制流量分发，为更好地利用多路径的可用带宽提供了可能。B4[11]和 SWAN[12]等相关研究表明 OpenFlow 在资源管控中具有不可替代的优势[13]。

本文提出了一种基于 OpenFlow 的多路径传输(OpenFlow based Multipath Transmission, OFMT)机制：当服务器之间经数据中心网络有多条路径可用时，OFMT 为每条流选择剩余带宽最大的路径作为其传输路径；对于时延敏感的短流，OFMT 优先为其分配带宽，并通过预安装默认流表进一步减小短流的完成时间。

本文其他部分结构如下：第 2 节分析并构建了 OFMT 优化模型，第 3 节给出了 OFMT 关键机制及算法设计，第 4 节通过实验对 OFMT 进行性能测试和分析，最后，第 5 节总结全文。

2　OFMT 优化模型

本节通过图论构建网络模型，并给出 OFMT 优化目标的数学模型。

若有向图 $G=(V,\ E)$ 表示一个网络，其中 V 为 OpenFlow 交换机集合，E 为网络中的链路集合，对于任意一条链路 e，$f(e)$ 表示链路 e 上的网络流量，$c(e)$ 表示链路 e 的带宽，(u,v) 表示链路的方向，$NH(u,v)$ 表示从交换机 u 到交换机 v 的第 1 跳交换机集合。

对于给定的任意两台服务器 s，d 与任何连通 s，d 的路径 $<s,u_0,\cdots,u_k,\ d>$，其中 $u_k\in V$，当 $j=1,\ 2,\cdots,k$ 时，存在 $(u_{j-1},u_j)\in E$，且 $u_j=NH(u_{j-1},d)$，则称路径 $<u_0,\cdots,u_k>$ 为 s 到 d 的可达路径，用 p 表示，所有可达路径的集合表示为 P_{sd}。W_{sd} 表示服务器 s 与服务器 d 间的一条路径的连接带宽。

优化目标 1：最优化负载均衡

OFMT 的第 1 个优化目标为实现最优化的负载均衡特性，即实现预优化流量 x_{sd} 在链路 $e(e\in p)$ 上的流量分量 $x_{sd}(p)$ 和该链路上已有流量 $f(e)$ 之和占据该链路带宽的比例 θ 尽可能地小，因此流量分解问题可以用式(1)~式(3)优化模型描述：

$$\begin{aligned}&\min\ \theta\\&\text{s.t.}\\&x_{sd}(p)+f(e)\le\theta c(e),\ \forall e\in E,\ p\in P_{sd},\ \ e\in p &(1)\\&\sum_{p\in P_{sd}}x_{sd}(p)\le W_{sd},\ \ \forall s\in V,\ \ d\in V &(2)\\&x_{sd}(p)\ge 0,\quad \forall p &(3)\end{aligned}$$

其中，$x_{sd}(p)$ 表示 s 到 d 的流在路径 p 上的流量分量，式(1)表明预优化流 x_{sd} 在链路 e 上的流量分量 $x_{sd}(p)$ 和该条链路上已有流量 $f(e)$ 总和不超过该条链路的容量 $c(e)$ 的 θ 倍，显然 $\theta\le 1$；式(2)表明预优化流 x_{sd} 的输入速率不超过 s 到 d 连接带宽 W_{sd}；式(3)表明任何可达路径流量不应为空。

优化目标 2：最大化带宽利用率

OFMT 的第 2 个目标是提高网络带宽的利用率，即在保持端到端带宽一定的条件下，对流而言，尽可能提高流对端到端连接带宽的利用率 λ，因此通过式(4)~式(6)优化问题来描述流量分解策略。

$$\begin{aligned}&\min\ \ \lambda\\&\text{s.t.}\\&x_{sd}(p)+f(e)\le c(e),\ \ \forall e\in E,\ \ p\in P_{sd},e\in p &(4)\\&\sum_{p\in P_{sd}}x_{sd}(p)\le\lambda W_{sd},\quad \forall s\in V,\ \ d\in V &(5)\\&x_{sd}(p)\ge 0,\quad \forall p &(6)\end{aligned}$$

其中 $x_{sd}(p)$ 表示 s 到 d 的流量在路径 p 上的流量分量，式(4)表明预优化流量 x_{sd} 在链路 e 上的流量分量 $x_{sd}(p)$ 和该条链路上已有流量 $f(e)$ 总和不超过该条链路的容量 $c(e)$；式(5)表明预优化流量 x_{sd} 的输入流量带宽不超过 s 到 d 连接带宽 W_{sd} 的 λ 倍，显然 $\lambda\le 1$；式(6)表明任何可达路径流量不应为空。

分析表明，提高网络中任何一条链路带宽的利用率，就是在当网络链路带宽一定的条件下，尽可能地提高网络注入的流量，因此对于优化目标 1 和目标 2，两者本质属同一问题，选取优化模型 2 作

为流量分解参量模型。显然优化模型 2 是 NP 完全问题，通过启发式算法进行求解。本文通过在OFMT中设计相关机制和算法实现最优的流量分解。

OFMT 实现的主要功能包括以下几个方面：

(1)维护网络视图。控制器首先需要获取网络拓扑的最新视图，然后计算服务器之间所有可用的端到端路径；同时，通过周期性轮询操作，控制器收集底层网络状态的数据，并对网络流量进行统计，获取单流的吞吐量和单链路的可用带宽。

(2)路由。对于进入网络中的流量，控制器基于全局网络视图，为每一条流计算最优的传输路径，最大化单流的吞吐量和最优化网络的负载均衡特性。

(3)调度。OFMT 的调度主要包含以下 3 个方面，第一，当多条流同时到达网络时，控制器优先传输短流，减小短流的完成时间；第二，控制器在轮询操作中发现网络存在较为严重的负载均衡问题时，对一些流进行重路由，提高网络的负载均衡特性；第三，在当前网络视图下，一条新流无法获得所需的带宽时，控制器通过重路由的方式对网络中的流进行调整，使得网络能够为新流提供更多的可用带宽。

为了提高所设计传输机制的扩展性和减小短流的完成时间，OFMT 仅对长流进行精细控制，而将短流作为背景流，通过预安装流表的方式进行传输。

3 OFMT 关键机制与算法

本节对 OFMT 的实现算法进行详细的设计，主要包括网络视图的更新，流的路由和调度。

3.1 网络视图的更新

3.1.1 统计信息的获取 OFMT 对网络视图的更新主要借助于 OpenFlow 完善的统计功能。OpenFlow 交换机中流表的计数器能够记录网络的状态信息，如单流的报文发送数目，端口发送的报文数目等，控制器通过发送单流和端口统计信息查询报文，便可获取相应的信息，继而对统计信息进行处理，可获取到单流的吞吐量和链路负载等网络状态。控制器轮询周期粒度越细，越能准确感知网络的实时信息，但会增加控制器的处理负担，同时过多的统计报文会增加控制器响应数据报文的时间，进而影响网络的性能。研究表明[9]，目前普通计算机中控制器每秒可以处理大约 100000 条流请求报文，且当 CPU 利用率不超过 75%时，控制器对新流请求能实现实时处理；而当 CPU 利用率低于 50%时，控制器资源未得到充分利用。因此在 OFMT 中，控制器的 CPU 利用率尽可能维持在 50%~75%的范围内，以平衡控制器的管理粒度和处理负载。为此，OFMT 在轮询操作中进行了如下的优化：

第一，选择合适的轮询周期。控制器轮询周期 T 初始化值为 5 s，控制器当前轮询周期为 T_{o}，CPU 的当前利用率为 p，更新后的轮询周期 T_{n} 根据式(7)的函数进行调整：

$$T_{\mathrm{n}}=\begin{cases}T_{\mathrm{o}}-(50\%-p)\times 0.5, & p\le 50\%\\ T_{\mathrm{o}}-(p-50\%)\times 0.1, & 50\%<p\le 75\%\\ T_{\mathrm{o}}+(p-50\%)\times 0.5, & p>75\%\end{cases} \tag{7}$$

通过使用式(7)的函数对轮询周期 T 进行适应当前网络状态的调整，OFMT 能够在保证对新流的实时处理下，尽可能提高对网络的管理粒度。

第二，OFMT 仅在服务器的接入交换机中进行信息查询，减少了统计报文的数量，进一步减轻了控制器的负载，同时也能较为准确地评估链路的负载情况。由于 TCP 流采用端到端的拥塞控制，因此，仅在接入交换机采集数据可以有效评估单 TCP 流的吞吐量；对于 UDP 流，在接入交换机进行信息的采集可能无法准确评估单条 UDP 流的实际吞吐量，但该 UDP 流的吞吐量不会高于接入交换机中的转发速率，因此仅在接入交换机处统计流的信息可以有效评估单流的吞吐量。

3.1.2 网络视图的更新 在 OFMT 框架中，控制器需要维护每一条链路的状态信息，其中，链路的状态信息包括链路的带宽、负载信息(单流信息，其中背景流作为一条流对待)和可用带宽。控制器通过周期性轮询操作，获取每一条流的当前吞吐量，更新流所在路径上每一条链路的状态信息。

描述网络视图更新过程的算法 1 如表 1 所示。

在数据中心网络中，每条传输路径具有相同数目的链路。若网络中共需要传输 m 条流，每条传输路径包含 n 条链路，其中链路数 n 相对于流数 m 来说是固定的，并且在数据中心网络中链路数 n 要远小于流数 m，所以算法 1 的时间复杂度为 $O(m)$。

3.2 路由和调度

对于任何一条流，OFMT 的路由机制旨在找到一条负载最轻的路径作为该流的传输路径。OFMT 路由机制的实现思想为：当一条流的报文达到控制器，控制器首先计算出所有可达路径，并计算每一条路径的可用带宽，然后将可用带宽最大的路径作为该流的传输路径；当网络中所有可达路径的可用带宽均相等时，控制器选择流数目最少的路径作为该流的传输路径。其中，路径的可用带宽为路径上每一条链路可用带宽的最小值，路径中流的数目为路径上每一条链路传输流数目的最大值。

表 1 网络视图更新

算法 1: 网络视图更新

输入: $F \leftarrow \{f_{uc}\}$ #网络中所有未完成的流
$Link\_state\_old$ /*旧的网络视图*/

输出: $Link\_state\_new$

```
Link_state_new ← φ
foreach f in F do
    calculate f current throughput B_c(f)
    foreach link in path(f) do/*path(f)为 f 的传输路径*/
      if f in Link_state_old (link) do
          B_a(link) ← B_a(link)+B_o(f) − B_c(f)
          /* B_a(link) 为 link 的可用带宽，B_o(f) 为 f 上一
          轮询周期的带宽 */
        end if
        else do
            /*新流，link 上没有流时，B_a(link) = c(e) */
            B_a(link) ← B_a(link)+B_c(f)
            N(link) ← N(link)+1
          end else
          Link_state_new ← [link, B_a, N]
      end for
  end for
  return Link_state_new
```

表 2　OFMT 路由

算法 2: OFMT 路由

输入: $F \leftarrow \{f_n\}$ /*所有新来的流*/
$Link\_state$ /*链路状态信息*/

输出: R/*路由信息*/

```
sort all the flows in F according to their flow size in ascend
order
for f in F do
  calculate all paths for f
  foreach path in paths do
    path_B_a = Min(B_a(link)), links ∈ path
    path_N_f = Max(N_f(link)), links ∈ path
    if ∃path_B_a>0 do
      path(f) ← path[Max(B_a) ⊕ Min(N)]
    end if
    else do
      path(f) ← path[Min(N)]
    end else
    updating network view
  end for
end for
```

为了实现有效的路由计算，OFMT 将路由与调度机制相结合：控制器在计算流的传输路径和带宽分配时，优先为短流分配所需的带宽，并优先传输短流。其次 OFMT 采用负载均衡优先的原则进行流传输路径的计算，即控制器优先为一条流计算一条负载最轻的路径，实现良好的负载均衡特性。当网络资源的分配不能达到最优时，需控制器对网络中的部分流进行重路由，为新进入网络的流分配更多的传输带宽，实现最大化单流传输带宽。描述 OFMT 的路由过程的算法 2 如下：

若网络中交换机数为 s，共需要传输 m 条流，每条传输路径包含 n 条链路，每一对服务期间有 r 条可用路径。在算法 2 中，为每条流计算所有可用传输路径所需时间为 $O(ns^2)$，继而总的时间复杂度为 $O(mns^2r)$。而在数据中心网络中，r 与 n 相对于流数 m 和交换机数 s 较为固定，且要远小于流数 m 和交换机数 s，此外，在一个数据中心网络内，交换机数 s 为一个固定值，所以算法 2 的时间复杂度为 $O(m)$。

3.3 流的重路由

当控制器检测到网络中存在较为严重的负载均衡问题，或所有可达路径均无法满足一条新流所需的传输带宽，此时需要控制器对网络中一些流量进行重路由。本文设定当两条等价多路径上负载带宽的差值超过链路带宽的 30%，就认为这两条等价多路径之间存在负载不均衡问题，此时需要将负载较重路径上的流迁移到负载较轻的路径上。

在 OpenFlow 网络中，流的重路由就是控制器为流计算新的传输路径并在交换机上安装新的路径流表。在流的重路由过程中，OFMT 需要保证流的路径切换无丢包，对服务器实现透明切换。本文提出一种无缝流迁移算法，算法的核心内容是：控制器首先为流计算新的端到端路径并安装对应的流表，此时新流表的优先级低于原路径流表的优先级，从而保证新流表的安装过程对流的传输无影响[14]。其次，找到新路径和旧路径的共享交换机，此时，两条路径的共享交换机可能不止一个，依次删除共享交换机上旧路径的流表，使流切换到新路径上进行传输。由于在流路径切换的过程中始终有一条连通的路径，能够保证该流的传输过程不丢包，实现对服务器的透明切换。

4　性能测试

本节通过仿真验证 OFMT 的性能，并与 SPT, ECMP, RRMP 的性能进行对比。

4.1 实验设计

实验选用 Mininet[15]作为实验平台。Mininet 作为 OpenFlow 网络的仿真器，具有很高的精确度，可以高质量再现文献[16]等的实验结果，目前已作为一个灵活的网络实验平台得到了大规模的使用。OpenFlow 交换机使用 Open vSwitch v2.3.1 版本，控制器使用 POX-carp。实验中选用 k=4 的 Fat-Tree 拓扑，网络中包含 20 台交换机和 16 台服务器，设定所有链路的带宽为 10 Mbps，链路时延为 3 ms，交换机端口缓存为 50 个报文，端口采用先来先服务调度模型和弃尾排队模型。为了提高实验的准确性，每组实验重复 10 次，并取平均值作为实验结果。每次实验的运行时间为 220 s, OFMT 中控制器的轮询周期 T 初始化为 5 s。

实验中使用文献[1]中给出的基于 Web 搜索集群产生的流量分布模型，该流量模型与文献[17,18]研究结论一致，数据中心边缘的流量呈 ON-OFF 模型，ON 和 OFF 的间隔时间以及报文到达的时间分别服从 3 个不同参数的正态分布，流量具有重尾分布和突发的特性。该 Web 搜索流量中，大于 1MB 的流占总流数目的 30%，并且网络中 95%的字节由这些流产生。在本实验中，使用 iperf 作为流量发生器，通过运行脚本使 iperf 产生服从 ON-OFF 模型的流量，并设定传输数据量小于 100 kB 的流为短流，大于 100 kB 的流为长流。实验中采用的通信模型如下：

(1)一对一通信(1-1)： 一台服务器同时仅与一台服务器通信，并且使用参数为 $n/2$ 的跳跃机制，即第 x 台服务器与第 $(x+n/2)\bmod n$ [15]台服务器进行通信，其中 n 为服务器的总数。

(2)一对多通信(1-n)： 一台服务器可同时至多和 n 台服务器进行通信，其中，通信目标服务器随机选择，实验中取 $n=2$ 。

4.2 仿真结果与分析

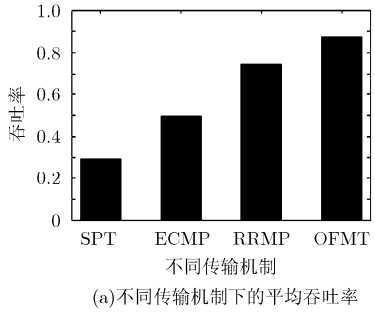
(1)**1-1 通信模式**：图 1(a)给出了不同传输机制下网络的平均吞吐率，图 1(b)给出了不同传输机制下网络中单流吞吐率的累积分布。OFMT 使得网络的平均吞吐率达到 0.87，较 SPT, ECMP, RRMP 的吞吐率都有显著提高。同时，OFMT 中超过 70%流的吞吐率超过 0.9，因此具有更好的公平性和网络负载均衡。图 2(a)和图 2(b)分别给出不同传输机制下短流与长流随网络负载变化的流完成时间曲线，总体来说，OFMT 较其他 3 种传输机制有效地减小了流的完成时间，尤其当网络负载大于 0.7 时，OFMT 取得更加明显的优势。短流完成时间的缩短是由于 OFMT 对短流采用预安装流表和优先为其分配所需带宽的策略；长流的完成时间被缩短主要得益于 OFMT 使网络具有良好的负载均衡特性，每条流分配到更多的传输带宽。

(2)**1-2 通信模式**： 图 3 给出了 1-2 通信模式下不同传输机制中网络的平均吞吐率和单流吞吐率的累积分布如图 3 所示。不同网络负载下短流和长流的完成时间曲线如图 4 所示。从图 3(a)可以看出，OFMT 中流平均吞吐率达到 0.86，在所有传输机制中仍取得了最高吞吐量，较 ECMP 提高了 70%；从图 3(b)中可以看出，OFMT 中单流的吞吐量均在 0.7 以上，因此有效地提高了网络的负载均衡特性。对于流的完成时间，OFMT 较其他 3 种传输机制具有较好的性能，尤其对于短流，在网络负载大于 0.7 时，短流的完成时间维持在较小的范围内(小于 50 ms)，其他 3 种传输机制在一定程度上均出现重尾现象；对于长流，OFMT 也将流的完成时间维持在较低水平。

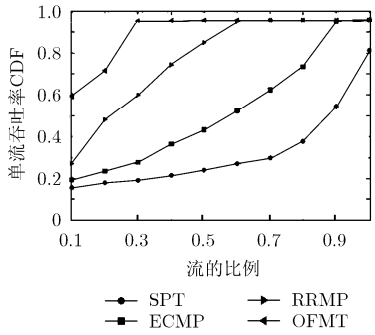
相比 1-1 通信模式，1-2 通信模式中网络中流的数目成倍增加，短流的数目增加尤为明显。较 1-1 通信模式，OFMT 中网络吞吐量有所下降，主要是由于单控制器的处理能力有限，当控制器负载较重时，增大轮询周期 T ，降低了感知网络实时状态的准确度，影响了网络中流的带宽分配和路径的最优化选择。同时为了保证短流的完成时间，OFMT 通过牺牲一部分长流的吞吐量来保证短流的完成时间，导致部分长流的完成时间增加，但整体而言，OFMT 依然保证了网络的高吞吐量以及短流的完成时间。

5 结束语

高连通度数据中心网络在现有传输机制下带宽利用率低、负载均衡较差，主要由于以下因素：首先未充分利用高连通数据中心网络提供的多路径，其次对流进行路径分配时未结合当前网络状态。在分析和总结现有方案缺陷的基础上，本文提出了 OFMT 机制。OFMT 借助于 OpenFlow 集中控制的优势，在全局网络上实现资源的最优化分配，对于进入网络的流，OFMT 为其选择当前网络中链路负载最轻的路径，并且通过周期性轮询和动态调度使网络保持较好的负载均衡特性并维持网络高吞吐量。在 Fat-Tree 数据中心网络拓扑中的实验测试表明，OFMT 将网络的吞吐量提高到 85%，较 ECMP 提高了 70%，同时缩短了流的完成时间，使短流的完成时间维持在较低的时间范围内。在下一步工作中，将进一步优化完善 OFMT 对网络流量的调度机制，实现在网络性能约束下负载均衡和短流完成时间的综合优化。此外，进一步在真实网络环境中对 OFMT 进行性能验证分析也是下一步工作的一部分。

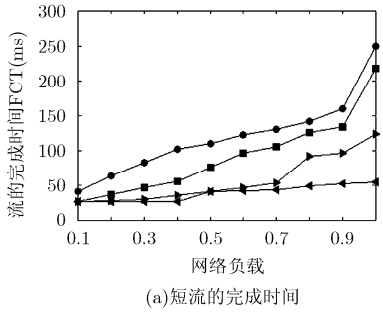


(a)不同传输机制下的平均吞吐率

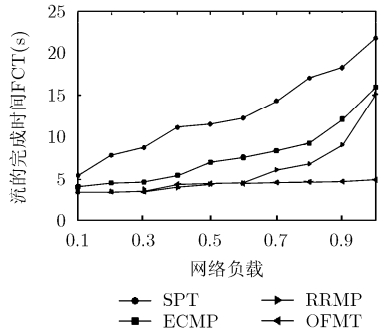


(b)单流吞吐率累积分布

图 1　1-1 通信模式下不同传输机制的网络吞吐率

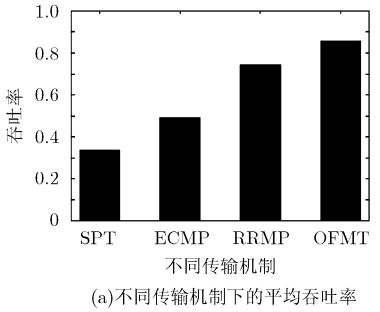


(a)短流的完成时间

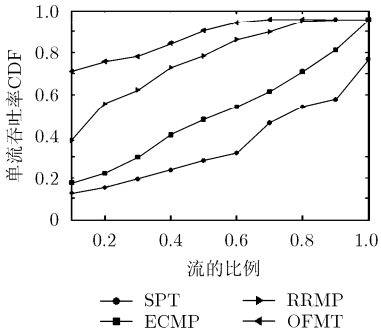


(b)长流的完成时间

图 2　1-1 通信模式下流的完成时间

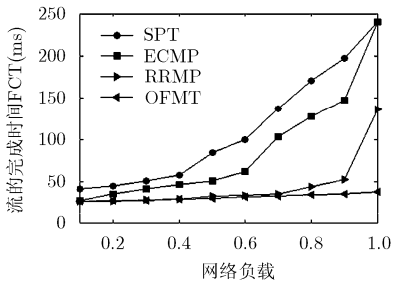


(a)不同传输机制下的平均吞吐率

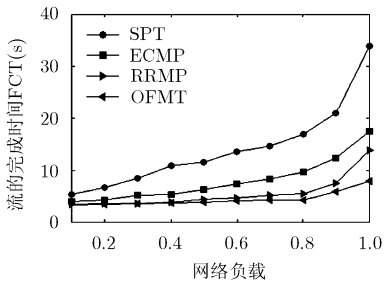


(b)单流吞吐率累积分布

图 3　1-n 通信模式下不同传输机制的网络吞吐率



(a)短流的完成时间



(b)长流的完成时间

图 4　1-n 通信模式下流的完成时间

参 考 文 献

[1] Al-FARES M, LOUKISSAS A, and VAHSAT A. A scalable, commodity data center network architecture[J]. *ACM SIGCOMM Computer Communication Review*, 2008, 38(4): 63–74. doi: 10.1145/1402946.1402967.

[2] GREENBERG A, HAMILTON J R, Jain N, *et al*. VL2: a scalable and flexible data center network[J]. *ACM SIGCOMM Computer Communication Review*, 2009, 39(4): 51–62. doi: 10.1145/1594977.1592576.

[3] GUO C, LU G, LI D, *et al*. BCube: a high performance, server-centric network architecture for modular data centers[J]. *ACM SIGCOMM Computer Communication Review*, 2009, 39(4): 63–74. doi: 10.1145/1592568.1592577.

[4] BREDEL M, BOZAKOV Z, BARCZYK A, *et al*. Flow-based load balancing in multipathed layer-2 networks using OpenFlow and multipath-TCP[C]. Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, Chicago, 2014: 213–214.

[5] CHIESA M, KINDLER G, and SCHAPIRA M. Traffic engineering with ECMP: an algorithmic perspective[C]. IEEE INFOCOM, Toronto, 2014: 1590–1598.

[6] MUDIGONDA J, YALAGANDULA P, AL-FARES M, *et al*. SPAIN: COTS data-center ethernet for multipathing over arbitrary topologies[C]. USENIX Symposium on Networked System Design and Implementation, San Jose, 2010: 265–280.

[7] RAICIU C, BARRE S, PLUNTKE C, *et al*. Improving datacenter performance and robustness with multipath tcp[J]. *ACM SIGCOMM Computer Communication Review*, 2011, 41(4): 266–277. doi: 10.1145/2018436.2018467.

[8] RAICIU C, HANDLEY M, and WISCHIK D. Coupled

congestion control for multipath transport protocols[R]. 2011.

[9] 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.

[10] KIRKPATRICH K. Software-defined networking[J]. *Communications of the ACM*, 2013, 56(9): 16–19. doi: 10.1145/2500468.2500473.

[11] JAIN S, KUMARA, MANDAL S, *et al*. B4: experience with a globally-deployed software defined WAN[J]. *ACM SIGCOMM Computer Communication Review*, 2013, 43(4): 3–14. doi: 10.1145/2486001.2486019.

[12] HONG C Y, KANDULA S, MAHAJAN R, *et al*. Achieving high utilization with software-driven WAN[J]. *ACM SIGCOMM Computer Communication Review*, 2013, 43(4): 15–26. doi: 10.1145/2486001.2486012.

[13] ALIZADEH M, EDSALL T, DHARMAPUEIKAR S, *et al*. CONGA: distributed congestion-aware load balancing for datacenters[C]. ACM SIGCOMM, Chicago, 2014: 503–514.

[14] OpenFlow Switch Specication, Version 1.0.0[OL]. http://www.openflowswitch.org/documents/openflow-spec-v1.0.0.pdf, 2015.

[15] HANDIGOL N, HELLER B, JEYAKUMAR V, *et al*. Reproducible network experiments using container-based emulation[C]. ACM CoNEXT, New York, 2012: 253–264.

[16] ALIZADEH M, GREENBERG A, Maltz D A, *et al*. Data center tcp (dctcp)[J]. *ACM SIGCOMM Computer Communication Review*, 2011, 41(4): 63–74. doi: 10.1145/1851182.1851192.

[17] PENG Y, CHEN K, WANG G, *et al*. Hadoopwatch: A first step towards comprehensive traffic forecasting in cloud computing[C]. IEEE INFOCOM, Toronto, 2014: 19–27.

[18] CHEN Y, JAIN S, ADHIKARI V K, *et al*. A first look at inter-data center traffic characteristics via yahoo! datasets[C]. IEEE INFOCOM, Shanghai, 2011: 1620–1628.

陈 鸣： 男，1956 年生，教授，研究方向为网络测量、网络性能分析与建模和软件定义网络.

胡 慧： 女，1993 年生，硕士生，研究方向为软件定义网络、网络测量和管理.

刘 波： 男，1988 年生，博士生，研究方向为软件定义网络、网络测量和管理.