基于多路广播树的SDN多路径路由算法

覃匡宇 黄传河 刘柯威 史姣丽 陈 希
(武汉大学计算机学院 武汉 430072)

摘 要 传统的网络使用基于最短路径的单一路径路由,无法有效地利用网络的全部带宽。软件定义网络(Software Defined Networking,SDN)采用中心化的控制平面能方便地实现对路由的精确控制。针对SDN网络下的多路径路由问题,提出了基于多路广播树的路由存储结构及相应的多路径选择算法。该算法根据各路径的可用带宽和时延进行概率分配,优先选择可用带宽大和时延小的路径。实验结果表明,该算法能快速地进行路由,并有效地减小传输时延和增大吞吐率。

关键词 SDN,多路广播树,多路径传输,流量工程

中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.01.037

Multipath Routing Algorithm in Software Defined Networking Based on Multipath Broadcast Tree

QIN Kuang-yu HUANG Chuan-he LIU Ke-wei SHI Jiao-li CHEN Xi
(School of Computer,Wuhan University,Wuhan 430072,China)

Abstract Shortest path based single path routing is used in traditional network. It cannot use all links' capacity effectively. Software defined networking (SDN) provides the centralized control plane to implement the precise control of the routing. To solve the multipath routing problem in SDN,a multipath broadcast tree structure and a multipath selection algorithm were proposed in this paper. The algorithm can allocate the probabilities to the paths according to their available bandwidths and latencies. The path which has bigger bandwidth and less latency will be given higher priority. The results of the simulation show that the algorithm can make routing decision fast while significantly reducing the transmission delay and increasing the throughput.

Keywords Software defined networking,Multipath broadcast tree,Multipath transmission,Traffic engineering

1 引言

传统网络通常使用基于最短路径的单一路径路由,单一路径路由具有结构简单、稳定性好、易于收敛等优点,但也具有易导致部分链路负载过重、链路负载不均衡、整体网络资源无法得到有效利用等问题。传统的骨干网络链路带宽利用率只有30%~40%,随着互联网数据信息的超线性增长,网络的不断扩容带来了巨额的成本。为了有效地利用全体的网络链路,将原有的单一路径路由方式变为多路径路由方式,可以有效利用一些原来较少使用的链路,让其更多地负担数据转发工作,从而使网络的整体利用率得到提高。

近年来,软件定义网络得到了大力的发展[1]。在SDN网络中,转发功能和控制功能分别位于不同的平面。路由机制也从传统网络的各交换节点逐跳计算变成由控制器进行整体计算。集中式的计算方式使得控制器掌握整体网络的全局视图,从而能够对数据流进行更精细的控制,提高链路的利用率。目前,代表性的SDN协议Openflow等已经成功应用于谷歌数据中心B4网络、德国电信网等。

尽管SDN由中央控制器进行统一处理,可以避免路由的收敛问题,为多路径路由创造了便利条件,但要实现多路径路由仍然面临着一些挑战,具体表现为:1)由单路径路由变为多路径路由后,计算工作量显著加大,在海量的数据请求下,需要合理的结构设计和高效的路由算法。2)在拥有多条路径可供选用的情况下时,需要选择合适的路径,使得数据传输既能保证较小的时延,又能使得空闲的带宽得到合理的利用。

本文在对SDN研究的基础上,提出了一种多路广播树MPBT的路由存储结构,并给出了时延和带宽优先的多路径选择算法。仿真结果表明,采用该算法能快速地进行路由,并能在保证较小传输时延的同时,有效地提高网络的吞吐率。

2 相关研究工作

目前,在多路径路由方面已有大量的研究。在传统的IP

到稿日期:2017-04-02 返修日期:2017-07-23 本文受国家自然科学基金(61373040,61572370)资助。

覃匡宇(1974-),男,博士生,CCF会员,主要研究方向为软件定义网络,E-mail:qky@whu.edu.cn;**黄传河**(1963-),男,教授,博士生导师,主要研究方向为计算机网络,E-mail:huangch@whu.edu.cn(通信作者);**刘柯威**(1980-),男,博士生,主要研究方向为移动网络安全,E-mail:kwliu@whu.edu.cn;**史姣丽**(1979-),女,博士生,主要研究方向为网络安全,E-mail:shijiaoli@whu.edu.cn;**陈 希**(1988-),男,博士生,主要研究方向为无线网络优化,E-mail:chenxi_cs@whu.edu.cn。

路由中，有逐跳路由和源路由两种方式。在源路由中，源节点通过自身的计算来获得到达目的节点的多条路径，选择其中一条路径后，将沿途要经过的节点的标识写入数据包的包头中，这些标识将帮助中间节点把数据包转发到目的节点。但源路由方案要求源节点拥有网络的整体拓扑信息，对源节点的要求较高。在逐跳路由中，包头的大小远远小于源路由。当路由器学习到多个路由时，将存储多个路由，并根据多条路径来平衡流量，该功能被称为负载均衡。在负载均衡的多路径选取中，若到目的地的多条路径具备相同的代价，则称之为等代价的多路径(ECMP)[2]，流量在这些等代价的路径中平均分配。而不等代价的负载均衡除了使用最小代价的路径外，还可以使用次优的路径。在逐跳路由方案中，当网络拓扑动态变化时，要实现各节点拓扑信息的快速收敛较为困难。

多拓扑路由(MTR)允许网络上的每个路由器对同一个目的地维持几个可用路由[3]。为了建立不同的拓扑，必须启用多个 OSPF 或 IS-IS 实例。每个实例维护一个单独的链路状态库，建立一张单独的路由表。一个包进入路由器后将被检查以决定使用哪一个拓扑。该类方案中，由于不同的路径依赖同一套拓扑，当路径较多时，该方案对拓扑的数量需求较大。

在传输层的多路径方案中，多路径 TCP(MPTCP)是 TCP 协议中支持并发传输的 IETF 版本[4]。一个 TCP 会话监督两个或更多的标准 TCP 会话，从而可以在同一个 TCP 连接下使用几条路径来进行数据传输。MPTCP 的前提条件是两个设备中至少有一个含有多个逻辑接口，从而有超过一个的 IP 地址，同时双方都支持 MPTCP 协议。

在 SDN 的多路径路由中，文献[5]提出了一种在 SDN 上集成了动态负载均衡多路径(DLBMP)的拥塞控制算法。在其提出的机制中，链路的负载信息被直接送到中央控制器中，实现了动态的路由决策，并进行端到端的拥塞控制。文献[6-7]在 openflow网络中使用了 MPTCP。其方案中的寻路机制是先通过 Dijkstra 找到最短路径，然后在拓扑中删除此路径，再继续用 Dijkstra 寻找下一条最短路径。文献[8]在 Layer-2 网络上通过静态的容量和延时指标来使用多路径路由，以适应链路故障。其寻路机制也是通过 Dijkstra 找到最短路径，然后减去其路径容量并继续寻找下一条。文献[9]提出了 HiQos 方案，在源和目的之间使用多条路径并通过排队机制来保证在不同的传输类型上的 Qos。该方案通过改造 Dijkstra 来获取多条路径。文献[10]使用 SDN 网络连接抗灾存储系统，此方案用于在灾害发生时根据可用的网络资源使用多路径路由提供的有效的数据传输。在寻找多路径时，该方案通过探测每个邻居节点以获得可用的路径。文献[11]以节省 TCAM 为目标，为多路径路由提出了一种三阶段的启发式算法，从而实现流表转发规则的放置。3 个阶段分别是路径生成、路径测试和路径流量合并，其中路径生成通过线性规划实现。文献[12]在两种案例下研究减少转发代价，使得在满足所需带宽的前提下经过的链路最少或经过的节点最少。文献[13]提出了一种 Dijkstra-Repeat 的路由算法，计算不相交的多路径路由，同时使用了懒路由更新(LRU)来减少控制器的路由计算量。在 Dijkstra-Repeat 算法中，控制器首先使用 Dijkstra 计算最短路径，然后重复使用 Dijkstra 计算不包括之前路径的最短路径，直到没有路径为止。

在上述文献的方案中，SDN 网络的多路径路由均是在源节点和目的节点间多次使用 Dijkstra 算法。若构造 k 条路径，则需要的时间复杂度为 $O(kn^2)$。

3 系统模型

3.1 网络模型

设网络用 $G(V,E,D,B)$ 表示，其中 V 表示节点集合，E 表示链路集合，D 表示链路的时延矩阵，B 表示链路的带宽矩阵。设找到从源节点 s 到目的节点 t 的一条路径 p，路径 p 由若干链路组成，则路径 p 的时延为组成路径的所有链路的时延之和。若用 d_p 表示路径 p 的时延，则有：

$$d_p=\sum_{e\in p}d_e \tag{1}$$

在路径 p 中，实际可用带宽为组成路径的所有链路的带宽中的最小值。若用 b_p 表示路径 p 的实际可用带宽，则有：

$$b_p=\min\{b_e\mid e\in p\} \tag{2}$$

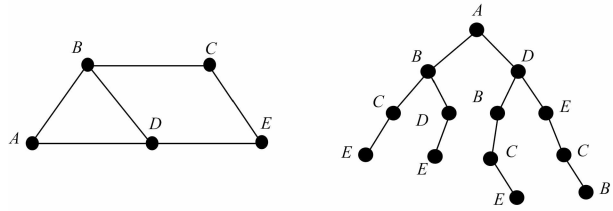
在模型中使用一个路径选择因子指标 $index_p$ 来表示在多路径路由中选用路径 p 的倾向度。路径选择因子定义为：

$$index_p=\frac{b_p}{d_p^2} \tag{3}$$

从式(3)可以看出，路径的可用带宽越大，延迟越小，路径选择因子的指标值越大。

3.2 多路广播树模型

传统网络通过对路由表进行查表来实现路由，多路广播树模型提供了一种完全不同的路由查找方式。多路广播树是一棵以目的节点为根节点，以各节点的邻居节点为子节点构造的树。与多路广播树不同，普通的树中每个节点是唯一的，在树中只出现一次。在多路广播树中，同一个节点可以出现多次。多路广播树如图 1 所示。



(a)拓扑图 (b)多路广播树

图 1 拓扑图与多路广播树的对应关系

Fig. 1 Relationship between topology and multipath broadcast tree

在控制器内为每个节点维持一棵以该节点为根节点的多路广播树，对于 n 个节点的网络，共存在 n 棵树。在拓扑图中，要查找给定源节点到目的节点的所有路径，只需在以该目的节点为根节点的多路广播树中，从每个源节点开始，依次寻找父节点，当到达根节点时，顺序经过的节点即为路径。如图 1 中，从源节点 C 到目的节点 A 可选的路径有：C-B-A，C-B-D-A 和 C-E-D-A。有了多路广播树后，新流到来时，控制器无需再次进行 *Dijkstra* 的计算，只需在以目的节点为根节点的多路广播树中找到所有源节点，依次查找父节点即可一路到达目的地。

系统中包含一些全局数据结构和局部数据结构。全局数据结构包括全局节点集和全局链路集。链路集中的数据包含每条链路的当前带宽、时延等信息。局部数据结构维护每棵多路广播树的指针结构和树节点链表集。

树指针结构用于多路广播树中连接父子节点的链接,它是一种带数据的双向指针结构,其数据部分表示该指针对应着全局链路集中的哪一条链路。

树节点链表集用于表示全局节点与多路广播树中树节点的映射关系。每个全局节点对应多路广播树中的若干节点,这若干节点按照树中该节点到根的跳数进行排序。

4　算法

4.1　多路广播树构造算法

多路广播树构造算法在控制器系统启动和拓扑发生变化时进行计算。算法依次计算拓扑中的节点,对每个节点都生成一棵树。当计算某个节点时,首先将该节点作为根节点,然后寻找其邻居节点,并将找到的邻居节点作为该点的孩子节点。然后对各孩子节点继续做寻找邻居节点的工作,并将新找到的邻居节点作为孩子节点的新孩子。在加入孩子节点时,算法将检查是否有环路,若新找到的孩子节点已经在从该节点到根节点的路径沿途出现过,则将该孩子节点丢弃。当再没有新的孩子加入时,算法即停止。以某节点为根节点的多路广播树建树算法 BMPBT(Building Multiple Path Broadcast Tree)的伪代码如下所示。

算法 1　多路广播树建树 BMPBT 算法

```
输入:旧多路广播树 Tree,当前节点 node,拓扑图 G
输出:新多路广播树 Tree
1. neighbors←getNeighbors(G,node)
2. parents←getParents(Tree,node)
3. neighbors←neighbors \ parents
4. if (neighbors==∅) return Tree
5. for each i ∈neighbors do
6.    addChild(Tree,node,i)
7.    Tree←BMPBT(Tree,i,G)
8. end for
9. return Tree
```

在计算以节点 t 为目的节点的多路广播树时,系统调用 BMPBT(*Tree*,*t*,*G*),此时的初始参数 *Tree* 中只包括一个根节点 t。在多次迭代计算后,返回的是整个以 t 为根节点的多路广播树。算法中 *node* 是树节点的数据结构,包含相关的指针及拓扑中对应的节点。函数 *getNeighbors* 用于求取拓扑图中某节点的邻居节点集。函数 *getParents* 用于求解树中某节点到根节点沿途经过的所有父节点集。为了防止出现环路,新发现的邻居节点若已经在父节点中出现过,则必须去除。函数 *addChild* 用于在树中的某节点下增加一个子节点。

4.2　树信息更新

网络运行期间,控制器需周期性地收集拓扑及链路状态。若期间发生了拓扑结构变化,控制器需要对多路广播树进行更新。若原拓扑中某链路断裂失效,则需对树中该链路对应的树枝进行剪枝。若增加了链路,则需要根据新增链路加入新树枝。对于链路的时延和当前剩余带宽等信息,更新全局链路集中的相关数据。

4.3　多路广播树路由算法

多路广播树路由 MPBTR(Multiple Path Broadcast Tree Routing)算法从源节点到目的节点的多条路径中选择一条作为本次传输的路径。当网络链路空闲时,该算法使用最短路径进行路由,当链路达到一定繁忙程度时,则启用多路径路由分担该链路的压力。算法以跳数最少的前 k 条路径作为备选,然后检查其中时延最少的路径,若该路径的可用带宽大于系统设置的阈值 *mb*,则选用该条路径;否则从 k 条备选路径中以各路径的路由倾向度作为概率比例随机分配传输路径。多路广播树路由算法的伪代码如下。

算法 2　多路广播树路由 MPBTR 算法

```
输入:多路多播树集 Trees,源节点 s,目的节点 t,最多考虑的路径数
      k,带宽阈值 mb
输出:路径 path
1. prob←{0,…,0},indexSum←0
2. Tree←getTree(Trees,t)
3. nodes←getNodes(Tree,s,k)
4. for i=1 to |nodes| do
5.    delays[i]←getDelay(Tree,nodes[i])
6.    bws[i]←getBandwidth(Tree,nodes[i])
7.    indexs[i]←calcIndex(delays[i],bws[i])
8.    indexSum←indexSum+indexs[i]
9.    prob[i]←indexSum
10. end for
11. i←argmin(delays)
12. if bws[i] > mb then
13.   return getPathToRoot(Tree,nodes[i])
14. else
15.   r←random()
16.   For j=1 to |nodes| do
17.      if r <(prob[j] / indexSum) then
18.      return getPathToRoot(Tree,nodes[j])
19.      end if
20.   end for
21. end if
```

在算法 2 中,函数 *getTree*(*Trees*,*t*)在所有的多路广播树中选出根节点为 t 的树。函数 *getNodes*(*Tree*,*s*,*k*)从树 *Tree* 的树节点链表集中选出对应于节点 s 的 k 个树节点,因为链表节点已经按照根的跳数进行了排序,所以从链表中选出前 k 个节点即可得到所要的树节点集。函数 *getDelay* 和 *getBandwidth* 分别计算树节点到根节点的延迟和可用带宽。函数 *calcIndex* 根据式(3)计算路径选择因子。变量 *prob* 用于存放放大 *indexSum* 倍后的概率累计值,经过第 8 行和第 9 行的计算后,*prob*[*i*]和 *prob*[*i*−1]之间的值除以 *indexSum* 即为采用第 i 条路径的概率。第 11 行选择延迟最短的一条路径,若该路径的可用带宽大于阈值,则执行第 13 行来选用该路径,否则执行第 15−20 行。第 15 行生成一个随机数 r,根据前面计算的 *prob*[*i*]和 *prob*[*i*−1]之间的值除以 *indexSum* 即为采用第 i 条路径的概率,第 16−19 行依次检查 *prob*[1]到 *prob*[*k*]的值,若 r 超过 *prob*[*i*−1]/*indexSum* 且小于 *prob*[*i*]/*indexSum*,则选用第 i 条路径。

定理 1 若网络共有 n 个节点，要生成 k 条路径的路由，多路广播树路由 MPBTR 算法的时间复杂度为 $O(kn)$。

证明：算法第 2 步的时间复杂度为 $O(1)$。算法第 3 步的时间复杂度为 $O(k)$。在算法的第 4 步到第 10 步，由于 *nodes* 集合的元素不超过 k 个，且生成树时没有环路，每个树节点到根节点的跳数不超过 n，则第 4 步到第 10 步的时间复杂度为 $O(kn)$。第 11 步的时间复杂度为 $O(k)$。第 12 步和第 13 步的时间复杂度为 $O(1)$。第 15－20 步的时间复杂度为 $O(k)$。因而全体相加，整体算法的时间复杂度为 $O(1+k+kn+k+1+k)$，可简略表示为 $O(kn)$。

5 实验仿真

为了验证前面提出的 SDN 多路径路由方案，本节对该方案进行了仿真实验。实验以 Mininet 软件为仿真环境，Floodlight 软件为控制器。Mininet 部署在一台处理器为 Intel i5 6500，主频 3.2G，16GB 内存，64 位 Linux 的机器上。Floodlight 控制器部署在另一台处理器为 Intel i5 3230m，主频 2.6G，16GB 内存的 64 位 Windows 7 机器上。Floodlight 的 JAVA 环境采用 64 位 JDK 8.0 版本。实验的拓扑环境采用美国 Internet 2 的拓扑结构，如图 2 所示。考虑到在 Mininet 中模拟的性能问题，实验中各链路的带宽均为 10Mbps，链路时延根据线路长度比例在 2～20ms 之间进行分配。实验中 MPBTR 算法将与最短路径优先的单路径路由 SPF 算法、基于哈希的多路径路由 HASH-MP 算法、轮询多路径路由 RRMP 算法进行比较。算法用 Python 编程实现，通过 Floodlight 控制器的 REST API 向交换机推送流表。实验使用 C 语言编制的 Client/Server 程序来模拟流量，Client 端按照指定的速率来发送数据，并根据往返数据来计算时延，服务端显示数据的接收速率。实验运行多对 Client/Server 副本，内容包含单点对单点的通信和多点对多点的通信。

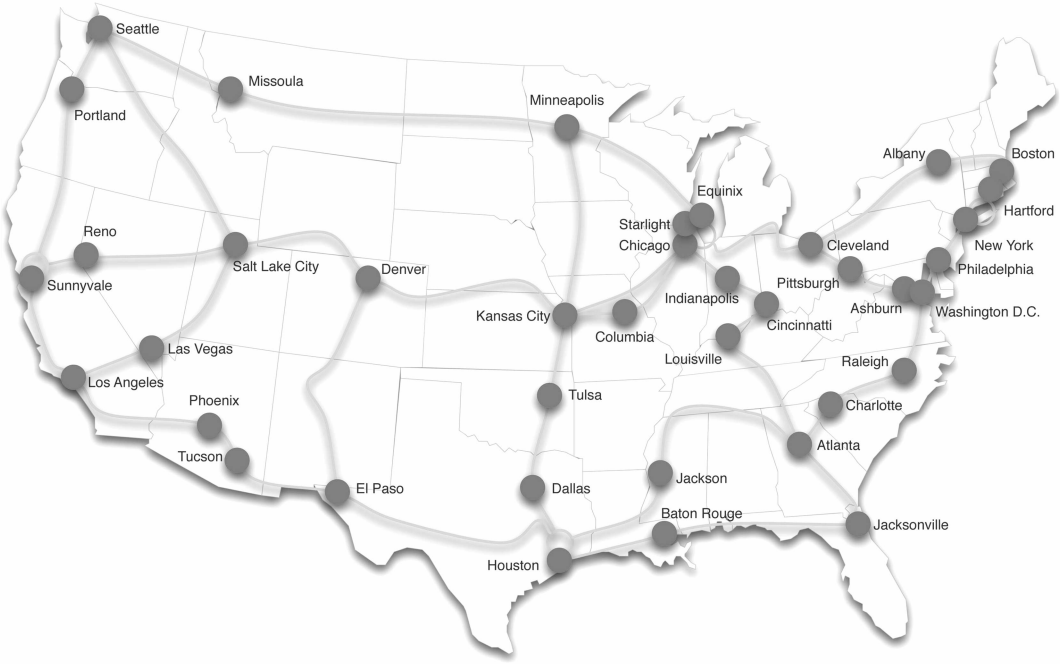
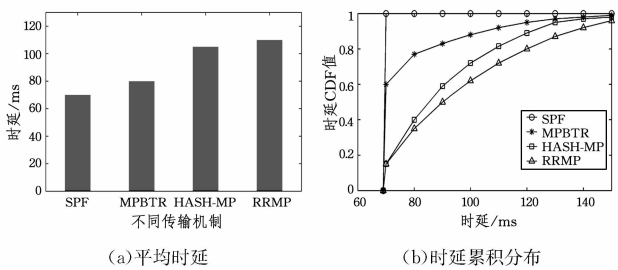


图 2 仿真实验的拓扑结构

Fig. 2 Topology of simulation experiment

5.1 网络延迟

在网络时延的测试中，使用单点对单点进行传输并依次加入数据流，每条流需要的带宽约为 1Mbps，流数从 1 条逐渐增加到 10 条，使得总带宽达到链路上限。将 MPBTR 算法中的可用带宽阈值设置为 4Mbps。数据采用多次实验的平均值，实验结果如图 3 所示。图 3(a)为不同传输机制下的平均时延，SPF 总是使用最短单一路径，得到的时延最短；MPBTR 算法在初始链路空闲时采用最短路径，在链路繁忙时才采用概率来选择多路径，在几种多路径算法的比较中时延最小；HASH-MP 算法根据传输流五元组的哈希值来确定路径，具有一定的不确定性，总体略优于 RRMP，但不如 MPBTR；RRMP 轮流使用各种路径，但因总有一部分流采用超长的路径，使得其时延总体最多。图 3(b)为各种算法的时延累计分布函数。在各个流的传输中，SPF 算法一直选用时延为 70ms 的最短路径；MPBTR 算法在链路可用带宽到达阈值之前都可以保持 70ms 的时延，在采用多路径之后，由于高带宽，低时延路径的路径选择因子大，选用该路径的概率就高，时延在 100ms 内的 CDF 值已接近 90%；HASH-MP 算法和 RRMP 算法在同等时延内的 CDF 值则明显偏低。



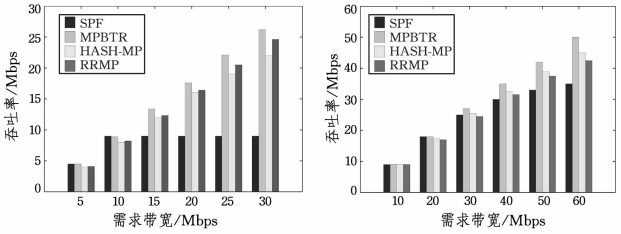
(a)平均时延 (b)时延累积分布

图 3 不同传输机制下的时延情况

Fig. 3 Latency of different algorithms

5.2　网络吞吐率

在网络吞吐率时延中，分别进行了单点对单点、多点对多点的数据传输。在单点对单点的实验中，在同一对节点的不同端口上运行多个 Client/Server 副本，每个副本的数据发送速率约为 128KB/s，根据多个 Server 端显示的接收速率的总和来计算吞吐率。在多点对多点的数据传输中，多个 Client/Server 副本分布在不同的节点上。实验结果如图 4 所示。图 4(a)为单点对单点的吞吐率。SPF 算法由于只使用一条路径，当带宽需求超过 10Mbps 后，吞吐率就已经无法增加；而 MPBTR 能根据链路剩余带宽调整路由，吞吐率基本接近发送速率；HASH-MP 算法通过哈希值来确定选用哪条路径，但因部分哈希值重合，其路由分布不够均匀，部分链路出现了拥塞，有些路径的时延也较大，吞吐率不如 MPBTR；RRMP 采用循环选择路由，链路分布较均匀，但长路径和短路径混杂，吞吐率仍然不及 MPBTR。图 4(b)为多点对多点的吞吐率。SPF 算法只考虑了最短路径的固定链路，在拓扑中央的部分链路较容易达到饱和；MPBTR 在进行路径选择时综合考虑了可用带宽和时延，在吞吐率比较中相对最好；HASH-MP 在多点对多点的传输中哈希冲突情况没有单点对单点明显，但其部分链路较长，跳数较多，同一份流量占用多条链路的带宽，带宽被浪费，吞吐率不如 MPBTR；RRMP 流量会轮换到有些很多跳数的链路，网络整体资源浪费比 HASH-MP 严重，总体吞吐率也不及 HASH-MP。



(a)单点到单点传输的吞吐率　　(b)多点到多点传输的吞吐率

图 4　吞吐率的比较

Fig. 4　Throughput of different algorithms

5.3　控制器负载

图 5 给出了 MPBTR 算法与基于 Dijkstra 的算法在控制器负载上的比较。仿真程序使用大量数据模拟流表请求来调用 Python 实现的路由算法，并测量控制器的负载。从结果中可以看到，由于 MPBTR 算法的时间复杂度只有 $O(kn)$，MPBTR 算法的控制器负载明显少于 Dijkstra 类的算法。

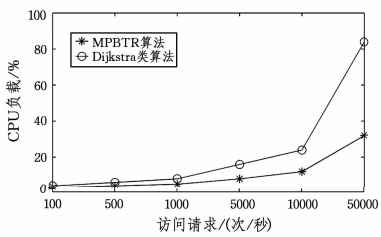


图 5　控制器负载

Fig. 5　Load of controller

结束语　本文提出了一种基于多路广播树的 SDN 多路径路由算法。该算法通过构建多路广播树，以空间换时间的方法来实现路径的快速查找。算法中设立了路径选择因子指标，在路径的选择上，该指标使得算法能根据路径的可用带宽和时延进行概率计算，优先选择可用带宽大和时延小的路径。实验结果显示，该算法路由速度快，并能有效地减小传输时延和增大吞吐量。当然，该算法也存在一些不足，如对于大型的网络，其构建的多路广播树消耗的空间较多。对于这种情况，可以考虑在拓扑上将相邻的若干节点收缩为一个节点，通过分块处理的方式来减小节点规模。在大型网络下支持分块处理的多路广播树路由算法，将是下一步的研究内容。

参 考 文 献

[1] KARAKUS M, DURRESI A. A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN) [J]. Computer Networks, 2017, 112: 279-293.

[2] JAROENRAT K, CHIMMANEE S, CHARNKEITKONG P. Algorithms for IP networks design with ECMP routing enable [C]// International Conference on Computing and Convergence Technology. IEEE, 2013: 420-425.

[3] SCHEFFEL M C, GRUBER C G, SCHWABE T, et al. Optimal multi-topology routing for IP resilience[J]. AEUE-International Journal of Electronics and Communications, 2006, 60(1): 35-39.

[4] FORD A, RAICIU C, HANDLEY M, et al. RFC 6182, Architectural Guidelines for Multipath TCP Development[EB/OL]. http://tools.ietf.org/html/rfc6182.

[5] FANG S, YU Y, FOH C H, et al. A Loss-Free Multipathing Solution for Data Center Network Using Software-Defined Networking Approach[J]. IEEE Transactions on Magnetics, 2013, 49(6): 2723-2730.

[6] POL R V D, BOELE S, DIJKSTRA F, et al. Multipathing with MPTCP and OpenFlow[C]// IEEE. 2013: 1617-1624.

[7] SANDRI M, SILVA A, ROCHA L A, et al. On the Benefits of Using Multipath TCP and Openflow in Shared Bottlenecks[C]// International Conference on Advanced Information Networking and Applications. IEEE, 2015: 9-16.

[8] SUBEDI T N, NGUYEN K K, CHERIET M. OpenFlow-based in-network Layer-2 adaptive multipath aggregation in data centers[J]. Computer Communications, 2015, 61: 58-69.

[9] YAN J Y, ZHANG H L, SHUAI Q J, et al. HiQoS: An SDN-based multipath QoS solution[J]. China Communications, 2015, 12(5): 123-133.

[10] IZUMI S, EDO A, ABE T, et al. An Adaptive Multipath Routing Scheme Based on SDN for Disaster-Resistant Storage Systems [C]// International Conference on Broadband and Wireless Computing, Communication and Applications. IEEE, 2015: 478-483.

[11] ZHANG J, ZENG D Z, GU L, et al. On Rule Placement for Multi-path Routing in Software-Defined Networks[C]// International Conference on Collaborative Computing: Networking, Applications and Worksharing. Springer International Publishing, 2015: 59-71.

[12] NAKIBLY G, COHEN R, KATZIR L. Optimizing Data Plane Resources for Multipath Flows[J]. IEEE/ACM Transactions on Networking, 2015, 23(1): 138-147.

[13] SUN X, JIA Z, ZHAO M, et al. Multipath Load Balancing in SDN/OSPF Hybrid Network[C]// IFIP International Conference on Network and Parallel Computing. Springer International Publishing, 2016: 93-100.