# 基于流量调度的 SDN 数据中心网络拥塞控制算法

樊自甫 李 书 张 丹
(重庆邮电大学通信与信息工程学院 重庆 400065)

**摘 要** 针对采用软件定义网络(SDN)的数据中心网络拥塞的问题,提出一种基于流量调度的数据中心网络拥塞控制算法。当链路发生拥塞时,该算法首先判别拥塞链路中链路上关键度最大的大流,然后对大流进行重路由计算,选择调度开销最小的流,并进行调度代价计算,最后对调度代价最小的流进行调度。实验结果表明,所提算法能够有效缓解网络拥塞,降低丢包率,提高链路利用率,使得网络性能更为稳定。

**关键词** 数据中心网络,软件定义网络,网络拥塞
**中图法分类号** TP393 **文献标识码** A

## Traffic Scheduling Based Congestion Control Algorithm for Data Center Network on Software Defined Network

FAN Zi-fu LI Shu ZHANG Dan
(School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

**Abstract** To alleviate the congestion problem using software defined network (SDN) in the modern data center network, a traffic scheduling based congestion control algorithm was proposed. When the link is congested, the proposed algorithm firstly discriminates the large flow of the maximum critical degree link in the congestion link, and then it reroutes the large flow, selects the minimum flow scheduling overhead, and calculates the scheduling cost. Finally, the minimum scheduling cost flow can be scheduled based on the above-mentioned process. Experimental results show that, the proposed algorithm can alleviate network congestion, improve the link utilization and enhance the network stability by reducing the dropout rates.

**Keywords** Data center network, SDN, Network congestion

## 1 引言

随着数据中心应用的不断增加及用户数量的迅速增长,其数据流量呈现大规模增长的趋势。思科预测未来 5 年全球数据中心服务流量将以平均每年 30%的速率增长,数据中心进入了发展高峰,数据中心单站点规模也在急剧增长。这一发展趋势使得数据中心的网络拥塞问题日益严重。软件定义网络(Software Defined Network, SDN)[1-2]以其控制平面和转发平面相分离、集中式控制及网络虚拟化[3]的特点,能够很好地满足数据中心网络的集中式管理、灵活低成本组网、网络虚拟化统一部署及多路径转发等需求[4],因此 SDN 在解决拥塞问题时有着先天的优势。目前关于拥塞问题的研究主要集中在终端侧和网络侧。终端侧只能暂时地缓解网络拥塞状况,不能有效地利用网络中其他处于空闲状态的路径,对网络资源是一种极大的浪费;网络侧拥塞解决方案通过调度整个网络的流量,使得流量尽可能占满整个网络所有可用路径,避免流量过度集中,从而减轻网络拥塞。相较于终端侧,网络侧更具研究价值。

目前针对基于软件定义网络架构的网络拥塞管理机制的研究还在起步阶段,而数据中心网络流量复杂、应用需求巨大,面临的问题也很严峻。Hui L 等人[5]提出一种拥塞避免机制(LABERIO 机制),该机制利用 OpenFlow 全网监控的特点实时监控网络的流量状态,如果监测到某一条或多条链路的负载相对集中,则立刻启动相应的拥塞避免措施,并对负载较大处的流量重路由。但是,这种算法准确性较差、灵敏度不高。Lu L 等人[6]提出一种协作式拥塞控制算法,该算法主要应用于多层网络中,实时监测核心层和边缘层的链路负载状态,针对不同的拥塞情况采用不同的重路由策略。但是,这种算法使用的是终端处的拥塞管理思想,信令开销较大,给控制器带来了一定的处理压力,不利于提升网络的性能。朱超[7]提出一种 CMTA-OF 算法,该算法同时考虑了数据流的业务属性和流量分布,选择转移代价最小的数据流进行调度。然而该转移代价模型是基于传统的业务模型,并不适用于数据中心网络的典型特征,同时无法满足网络整体链路利用率需求。吴志强等人[8]提出一种 CC-OF 拥塞控制机制,通过对拥塞节点的上一跳节点进行重路由来进行拥塞控制。但是该算法不能有效利用网络中大量处于空闲状态的链路,浪费了大量的网络资源。Hwang J 等人[9]提出了一个可扩展的拥塞控制协议(SCCP),利用 SDN 交换机有效地限制了 TCP 数据包的传送速率,使总的链路利用率不超过瓶颈链路容量,此算法对网络整体的链路利用率不高。Kanagevlu R 等人[10]提出了一种基于 OpenFlow 的本地重路由控制架构,该架构在云数据中心中根据网络负载变化自适应地对流量本地重路由。

综上所述,当前网络侧的拥塞控制算法主要采用动态重路由机制,存在的问题主要有:1)控制器开销较大,不利于网络性能的提升;2)无法提升网络整体链路利用率;3)算法灵活性较差。本文从网络侧出发提出一种基于流量调度的拥塞控制算法,此算法采用集中控制的数据流调度机制,通过多路径

樊自甫(1977-),男,副教授,主要研究方向为通信运营管理、下一代网络技术,E-mail:59200515@qq.com;李 书(1989-),女,硕士,主要研究方向为下一代网络技术、SDN;张 丹(1989-),女,硕士,主要研究方向为下一代网络技术、SDN。

路由方式，旨在有效缓解网络拥塞，同时降低控制器开销。

## 2　基于流量调度的拥塞控制算法

### 2.1　算法思路

本文设计一种拥塞控制算法（Congestion Control Per-Elephent Flow，CC-PEF），该算法能充分利用数据中心网络中存在的冗余路径，在完成细粒度流量调度的同时，能很好地克服控制器的计算开销，完成拥塞控制的目标。

该拥塞控制算法的思路流程如图 1 所示。首先，SDN 控制器对网络拓扑进行初始化，一旦 SDN 控制器检测到网络拓扑，控制器计算每一对节点之间的所有可能的路径，并将所有路径按跳数升序的方式排序，再从基于跳数的多条最短路径中选取可用带宽最大的一条路径作为主路径。控制器周期性地向交换机发送 State-Request 消息问询交换机及其端口的状态，并对所有链路进行流量监控且计算其负载情况。当链路负载大于设定的拥塞门限时，首先调度链路关键度最大链路上的大流，然后对该链路上的大流进行重路由计算，选择调度开销最小的流，最后下发新的流表，同时降低源主机的发包速率，完成整个拥塞控制算法的流程。

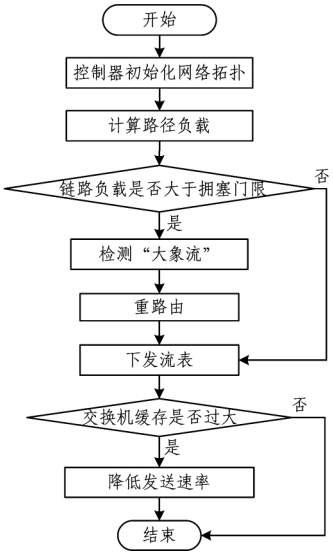


图 1　拥塞控制算法的流程

### 2.2　算法设计

#### 2.2.1　路径负载计算

SDN 控制器能实时监测网络中节点各端口和链路的流量信息，因此可以计算出每条链路任意时刻的链路利用率 $L_{i,j}(t)$，如式(1)所示：

$$L_{i,j}(t)=\frac{load_{i,j}(t)}{B_{i,j}}\times 100\% \tag{1}$$

其中，$load_{i,j}(t)$ 表示链路 $(i,j)$ 上传输数据流所占用带宽的大小，$B_{i,j}$ 表示链路 $(i,j)$ 上最大带宽的大小，即最大传输速率。

假设一条路径 $P$ 包含了链路 $(i,j)$，$(m,n)$，…，$(o,p)$，则该路径的负载为 $L_p(t)$，如式(2)所示：

$$L_P(t)=\mathrm{MAX}[L_{i,j}(t),L_{m,n}(t),\cdots,L_{o,p}(t)] \tag{2}$$

#### 2.2.2　链路负载判断

当控制器检测到链路负载大于设定的拥塞门限时，判定该链路产生了拥塞，需要对该链路上的数据流重新选择路径。本算法所涉及的拥塞门限阈值 $B_{th}$ 取链路最大带宽的一定比例，在实际网络中可以通过测量在一定链路利用率下的丢包率来确定比例的选取，链路拥塞的判断可采取式(3)进行计算：

$$L_{i,j}(t)>B_{th} \tag{3}$$

#### 2.2.3　大小流区分

当控制器监测到网络产生拥塞，就要对数据流重新选择可用路径。根据数据中心网络流量特征可知，为了减少网络开销、提高资源利用率，需要检测并调度网络中的大流。由于交换机支持监控特性（如 sFlow），控制器可以周期获得交换机内计数器记录的传输数据包数及字节数。数据流的大小由其所含字节数决定，数据流越大，持续的时间越长，反之，数据流越小，持续时间越短。通过周期性地监测流总共传输的字节变化量可以区分大小流。因此，流的大小分类可采取式(4)进行计算：

$$\psi=\frac{b_{t_2}-b_{t_1}}{(t_2-t_1)B} \tag{4}$$

其中，$\psi$ 是根据统计信息对流量大小进行的估计，$b_{t_1}$，$b_{t_2}$ 分别表示 $t_1$，$t_2(t_1<t_2)$ 时刻交换机接收到的字节数，$\psi$ 表示流的速率占带宽的比例，通过对 $\psi$ 进行界定即可区分大小流。不同的 Fat-Tree 拓扑网络中的链路带宽不同，导致对大流的判定标准也不同。为了适应不同的 Fat-Tree 网络链路带宽，当 $\psi$ 超过 10% 时，就认定该流为一个“大象流”。在启用动态路由时，针对不同大小的流制订相应调度策略，为了降低控制器开销，不需要对小流量进行重路由。

#### 2.2.4　重路由计算

为了充分利用节点对之间存在的多路径，需要考虑链路的权值优化问题才能充分利用数据中心网络中存在的冗余路径。

首先，通过定义链路关键度对关键链路进行量化，可采取式(5)进行计算：

$$\lambda(l_{i,j})=\frac{AVE(l_{i,j})}{B_{l_{i,j}}} \tag{5}$$

其中，

$$AVE(l_{i,j})=\frac{b_{l_{i,j}}}{M_{l_{i,j}}} \tag{6}$$

其中，$\lambda(l_{i,j})$ 表示链路 $l_{i,j}$ 的关键度；$B_{l_{i,j}}$ 表示链路 $l_{i,j}$ 的最大带宽；$AVE(l_{i,j})$ 定义为链路的平均期望负载，该期望值越高说明链路质量越好，则数据源端选择这条链路的概率值就大，$b_{l_{i,j}}$ 定义为所有节点对之间通过链路 $l_{i,j}$ 的“大象流”流量之和，$M_{l_{i,j}}$ 表示通过链路 $l_{i,j}$ 的“大象流”数目之和。

为了充分利用网络中的冗余链路资源，本文采用多路径传输，提出对调度开销进行优化的方法。目前最常用的就是定义链路最大带宽或可用带宽的倒数作为链路的边权值。例如，某一刻一条链路剩余带宽较大，按照以往定义的倒数权值关系，那么权值相对较小，越容易吸引流量，但是如果这是一条关键度值很高的链路，根据 $AVE(l_{i,j})$ 的物理意义，那么在下一刻这条链路有可能接收更多的流量，给它分配过多的流量会导致拥塞问题加重。因此，当链路发生拥塞时，调度链路关键度最大的拥塞链路上的大流，最大链路关键度通过式(7)计算：

$$\lambda(l_{s,t})=\mathrm{MAX}[\lambda(l_{i,j}),\lambda(l_{m,n}),\cdots,\lambda(l_{o,p})] \tag{7}$$

链路 $(s,t)$ 上大流的条数为 $M$，其中 $f_k$ 的流速为 $v_k$，$P$ 为其流经的路径，$n(P)$ 为路径中的跳数，$\lambda(P)$ 为其所有链路的链路关键度之和，现将 $f_k$ 重路由至网络中的另一条非拥塞路径 $P'$，$B_{av}(P')$，$n(P')$，$\lambda(P')$ 分别为路径 $P'$ 的可用路径负载、跳数和所有链路的链路关键度之和。流的调度开销设为 $C(f_k)$，如式(8)所示：

$$C(f_k)=\lambda\frac{n(P')-n(P)}{n(P)}+\mu\frac{v_k}{B_{av}(P')}+\psi\frac{\lambda(P')-\lambda(P)}{\lambda(P)} \tag{8}$$

其中，$\frac{n(P')-n(P)}{n(P)}$ 表示新路径跳数变化增加的开销，$\frac{v_k}{B_{av}(P')}$ 表示占用新路径可用带宽的开销，$\frac{\lambda(P')-\lambda(P)}{\lambda(P)}$ 表示新路径关键度变化增加的开销，为了防止流量调度后产生新的拥塞，在为大流进行路由时，需要小于路径 $P'$ 上的可用路径负载 $B_{av}(P')$，定义 $B_{av}(P')$ 为：

$$B_{av}(P')=B_{th}(P')-BL_{P'} \tag{9}$$

而 $\lambda$，$\mu$ 和 $\psi$ 分别为跳数增量、可用带宽和路径关键度的权重，其取值范围都在[0，1]的区间内，可根据实际网络的需求进行设置。控制器最终下发的路由策略是在节点对之间所有可能的多条路径上进行流量调度，以缓解网络拥塞。

2.2.5 降低发送速率

在一定程度上，当前流表项的数目反映了交换机的负载状况。本文指定交换机负载的度量参数 $F$ 的定义如下：

$$F=\frac{F_c}{F_m} \tag{10}$$

其中，$F_c$ 代表当前的流表条目数，$F_m$ 代表交换机能够维护的流表条目的上限。$F$ 的取值范围在 0 到 1 之间。当 $F_c=0$ 时，$F$ 取值最小为 0，此时 OpenFlow 交换机的流表为 0；当 $F_c=F_m$ 时，$F$ 取值最大为 1，此时 OpenFlow 交换机的流表条目已满。通过对交换机负载的度量参数 $F$ 进行界定，可以判断当前交换机是否过载，一旦过载，控制器应当控制源主机的发包速率。

当链路发生拥塞时，首先调度拥塞链路中关键度最大的链路上的大流，对该链路上的大流进行重路由计算，选择调度开销最小的流，并使用式(8)进行调度代价计算，最后对调度代价最小的流进行调度。拥塞控制算法如算法 1 所示。

算法 1 CC-PEF Algorithm

输入：网络拓扑 G(V，E)、流量矩阵 T、链路最大带宽、拥塞判断阈值 $B_{th}$

输出：重路由后的路径 $p'(f_k)$

1. G(V，E)中任意两个节点间的路径集 p←Depth-First Traversa Algorithm (V，E，n(P))
2. $L_{i,j}(t)$←链路(i，j)的链路利用率，$L_p(t)$←路径负载
3. $\lambda(\lambda_{i,j})$←链路 $l_{i,j}$ 的关键度，路径关键度 $\lambda(p)\leftarrow\Sigma\lambda(l_{i,j})$
4. $F=\frac{F_c}{F_m}$←交换机负载度量值
5. while $L_{i,j}(t)\leqslant B_{th}$
6. if $\psi_k>0.1$
7. 判定流量矩阵 T 中的 $f_k$ 为“大象流”
8. end if
9. $\lambda(l_{s,t})=\max=[\lambda(l_{i,j}),\lambda(l_{m,n}),\cdots,\lambda(l_{o,p})]$
10. M←链路 $l_{s,t}$ 上大流的条数，最小的调度开销 $C_{min}\leftarrow\infty$
11. for k=1，k≤M，k++
12. $P'(f_k)$←有效可用路径，可用路径负载 $B_{av}(P')\leftarrow B_{th}(P')-BL_{p'}$
13. 流 $f_k$ 的调度开销 $C(f_k)\leftarrow\frac{n(P')-n(P)}{n(P)}+\frac{v}{B_{av}(P')}+\frac{\lambda(P')-\lambda(P)}{\lambda(P)}$
14. if $C(f_k)<C_{min}$
15. $C_{min}=C(f_k)$
16. end if
17. end for
18. 选择具有 $C(f_k)$ 最小值的流进行调度，设为流 $f_k$，将 $P'(f_k)$ 发送给流表下发模块，路由拓扑中流 $f_k$ 经过的相应链路负载增加 $v_k$
19. end while
20. if 超过设定的负载值
21. 使用 ovs-vsctl 命令降低发送速率
22. end if

## 3 实验与性能分析

### 3.1 实验环境

为了更清晰地展示算法的有效性和性能，本文在 Ubuntu 系统上使用了轻量级网络仿真工具 Mininet，并在其上搭建网络拓扑，包含了支持 OpenFlow 协议的交换机 OpenvSwitch、一般的主机和链路。同时在 Mininet 外部连接了一个外部控制器，实验环境中使用的控制器为 Ryu[12]，并通过增加 Ryu 的功能模块来实现算法。实验采用 Fat-Tree[13] 网络拓扑，其中包括 4 台边缘层交换机(S5，S6，S9，S10)，4 台聚合层交换机(S3，S4，S7，S8)和 2 台核心层交换机(S1，S2)。每一个边缘层交换机通过两个端口分别与两台主机相连接，每 4 个聚合层交换机或边缘层交换机组成一个 Pod，每个 Pod 之间有 4 条可用链路，仍然具有多路径的特性，能够满足本文所设计算法的需求，如图 2 所示。

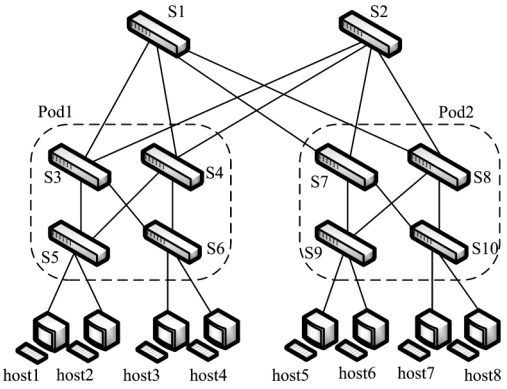


图 2 实验拓扑图

为实现 CC-PEF 算法的功能，作者在 Ryu 控制器中编程实现了以下功能模块：路径计算模块(Path Calculation)、拥塞监测模块(Congestion Monitoring)、大流检测模块(Elephant Flow Ddetection)、拥塞控制重路由模块(Re-routing)及流表管理模块(FlowTable Managing)，如图 3 所示。

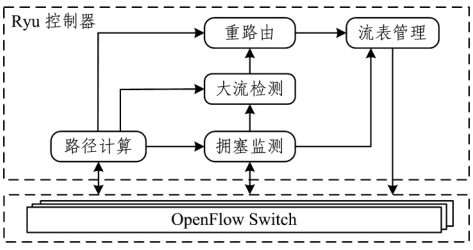


图 3 基于 Ryu 的拥塞控制算法模块

实验使用的设置如下：各交换机的路由接口速率均为 200Mbit/s，设置的拥塞门限为 180Mbit/s，实验中构造 4 条 UDP 数据流，分别为 f1(host5、host7)，f2(host5－host8)，f3(host6－、host7)，f4(host6－host8)。由最小路径算法可知，4 条数据流以 S9－S7－S10 为主路径进行传输，路径 S9－S8－S10 和 S9－S7－S2－S8－S10 作为可用路径。实验按对一般数据中心链路容量进行一定比例的缩小来进行设置，放大后仍符合真实网络情况。

### 3.2 实验结果分析

为了验证 CC-PEF 的网络拥塞控制性能，实验拟采用路径负载、链路利用率和丢包率作为对比参数，并与文献[11]中

的基于每个数据包转发的拥塞控制(Congestion Control Per-Packet，CC-PP)算法和文献[8]中基于 OpenFlow 的拥塞控制(Congestion Control in OpenFlow，CC-OF)算法进行了对比分析。

(1)路径负载

首先，为了验证算法对网络中链路拥塞的有效性，本文对主路径 S9－S7－S10 和可用路径 S9－S8－S10，S9－S7－S2－S8－S10 进行了考察，如图 4 所示。实验开始后路径 S9－S7－S10 的负载一直增大，到达最高 94.5%，超过设置的 90%门限值，表明该路径发生了拥塞，启动了 CC-PEF 拥塞控制算法，把该路径上的一部分流量分配到可用路径 S9－S8－S10 和 S9－S7－S2－S8－S10，但是路径 S9－S7－S10 的拥塞并没有得到立刻缓解，原因是拥塞链路上转移的流的数目和大小未能使路径 S9－S7－S10 的负载降低到门限值以下。随着数据流的不断转移，在第 7 秒以后，路径 S9－S7－S10 的负载最后稳定在 78%左右就不再变化，缓解了路径 S9－S7－S10 的拥塞问题，使其负载一直低于门限值。

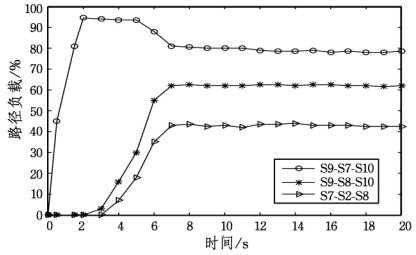


图 4　路径负载变化情况

同时，为了分析 3 种算法缓解拥塞路径的性能，对比了 3 种算法针对拥塞路径 S9－S7－S10 的负载变化情况，如图 5 所示。从图 5 中可以看出，CC-PEF 算法在第 6 秒时使得路径 S9－S7－S10 的拥塞明显缓解，路径负载从 94.5%降低到 88.7%，最终路径负载稳定在 78%左右；CC-PP 算法在第 8 秒时将路径负载降低到 79%，但在第 11 秒时导致了新的拥塞，最后稳定在 85%左右；CC-OF 算法的拥塞路径负载下降得比较缓慢，最后稳定在 86%左右。本文所设计的 CC-PEF 算法能够有效地缓解网络拥塞，同时算法性能更加稳定，不会造成新的网络拥塞，使得网络性能得到明显的改善。

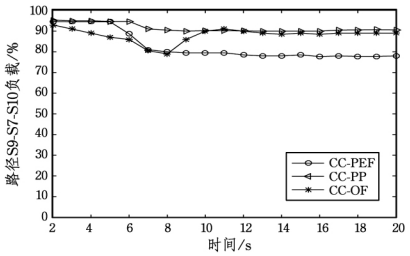


图 5　拥塞路径负载对比

(2)链路利用率

为了分析 3 种算法缓解拥塞路径的性能，图 6 对比了 3 种算法链路利用率的变化情况。由图 6 可知，CC-OF 算法的链路利用率平均为 69%，CC-PEF 算法为 79%，CC-PP 算法的链路利用率在 86%左右。虽然 CC-PP 算法有更高的链路利用率，但是网络性能在第 9 秒时产生恶化，导致链路利用率迅速下降，这是因为在接收端产生数据包失序而导致的。由于 CC-PEF 算法在一定程度上能够避免数据包失序问题，且充分利用了网络的空闲资源，因此网络的链路利用率比较高且相对稳定。从图 6 可知，CC-PEF 算法的链路利用率比 CC-OF算法高 10%左右，这是因为 CC-OF 算法会导致新的拥塞问题，使得网络的资源不能得到充分的利用。

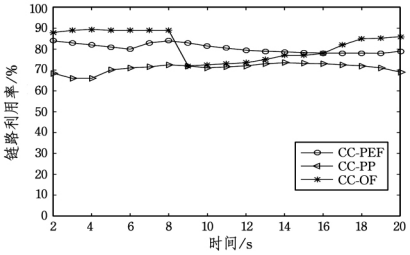


图 6　链路利用率对比

(3)丢包率

由图 7 可以看出，随着数据流量的变大，3 种算法的丢包率都有不同程度的上升，但都小于 0.1%，在满足网络稳定性范围之内。其中，当流量为 350M 到 400M 时，CC-PP 算法的丢包率增长速度比 CC-PEF 算法更快，这是由数据包失序问题导致的。而 CC-OF 算法性能很差，当数据流比较大时，更容易发生拥塞，丢包率几乎达到了 0.044%，远高于 CC-PEF 算法的 0.023%。而且，CC-PEF 算法的性能比 CC-PP 和 CC-OF 算法更好、更稳定。

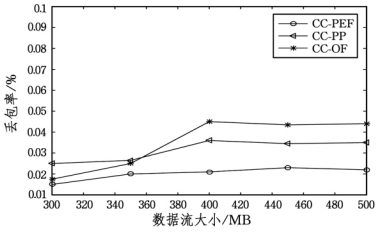


图 7　丢包率对比

结束语　本文在 SDN 技术背景下，结合了数据中心网络结构、应用及流量等典型特征，设计并实现了一种拥塞控制算法。该算法采用 SDN 集中控制方式的流量调度，通过对全局视图的网络管控，充分利用数据中心网络中存在的冗余路径，在完成细粒度流量调度的同时，能较好地降低控制器的计算开销，完成拥塞控制的目标，并能克服分布式只能达到局部最优的缺点。通过在实验平台上模拟数据中心网络流量特征的数据流，并和两种典型算法进行对比分析，验证了所设计算法在本实验环境中能够有效地缓解网络拥塞，同时提高了网络资源的整体利用率，降低了丢包率，算法性能更稳定、有效。

## 参 考 文 献

[1] 左青云，陈鸣，赵广松，等. 基于 OpenFlow 的 SDN 技术研究[J]. 软件学报，2013，24(5)：1078-1097.

[2] 张卫峰. 深度解析 SDN：利益、战略、技术、实践[M]. 北京：电子工业出版社，2014.

[3] 邓罡，龚正虎，王宏. 现代数据中心网络特征研究[J]. 计算机研究与发展，2014(2)：395-407.

[4] 郑毅，杨艳松，刘思聪. SDN 在 IDC 网络的应用方案研究[J]. 邮电设计技术，2014(6)：58-61.

[5] HUI L，YAO S，GUO M Y，et al. LABERIO：Dynamic load-balanced Routing in OpenFlow-enabled networks[C]// 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). IEEE，2013：290-297.

[6] LU L Y，YANG X，DU H F. OpenFlow control for cooperating AQM scheme[C]// 2010 IEEE 10th International Conference on Signal Processing (ICSP). IEEE，2010：2560-2563.

(下转第 273 页)

率即为 UA 可识别数据集大小与数据集总大小的比值；另一部分的识别率，即 C4.5 决策树分类模型的准确率，因此模型总体的识别率为：

$$Accuracy_{total}=Accuracy_{UA}+(1-Accuracy_{UA})\times Accuracy_{C4.5Tree} \quad (8)$$

UA 识别率和模型整体识别率如表 7 所列。

表 7　模型识别率/%

| 数据集 | 连接数 | UA 识别率 | 模型识别率 |
|---|---|---|---|
| DataSet1 | 204770 | 60.4429 | 94.5469 |
| DataSet2 | 756709 | 70.4304 | 96.1369 |
| DataSet3 | 989383 | 63.8979 | 95.0509 |
| DataSet4 | 210455 | 62.9275 | 95.0922 |
| DataSet5 | 195770 | 61.0982 | 95.7423 |
| DataSet6 | 232267 | 58.0349 | 95.8503 |
| DataSet7 | 244082 | 70.2940 | 96.8677 |
| DataSet8 | 236705 | 66.1511 | 95.4947 |
| DataSet9 | 215315 | 58.9462 | 94.1787 |
| DataSet10 | 658120 | 65.5054 | 95.5851 |
| 平均 | 394357.6 | 63.7729 | 95.4546 |

### 4.2　效率评估

模型效率是另一项评估模型的重要指标。模型对手持与非手持终端流量的区分主要由两部分组成：1）UA 识别部分，直接通过关键字进行匹配；2）对于 UA 无法识别的部分，需训练 C4.5 决策树进行分类。UA 识别部分，所需匹配关键字预先已经获取，因而模型建立所需的时间即为建立 C4.5 决策树分类模型的时间。各个数据集的建模时间如表 8 所列。

表 8　模型建立时间/s

| 数据集 | 连接数 | 模型建立时间 |
|---|---|---|
| DataSet1 | 204770 | 26.86 |
| DataSet2 | 756709 | 363.30 |
| DataSet3 | 989383 | 626.85 |
| DataSet4 | 210455 | 30.58 |
| DataSet5 | 195770 | 21.73 |
| DataSet6 | 232267 | 26.26 |
| DataSet7 | 244082 | 36.23 |
| DataSet8 | 236705 | 43.13 |
| DataSet9 | 215315 | 33.43 |
| DataSet10 | 658120 | 225.85 |

从表 8 中的数据可以看出，模型的建立效率很高，主要原因在于训练 C4.5 决策树分类模型所用属性只有 5 个，避免了高维数据带来的时间损耗，因而可以获得较快的建模速度。

结束语　通过机器学习的方法进行网络流量的识别和优化是目前网络流量管理的研究热点。本文利用 C4.5 决策树结合传统 UA 识别方法建立手持终端流量分类识别模型，对手持终端流量识别有较高的准确率。必须指出，目前 WF-C4.5 方法不能精确识别具体的设备型号、操作系统类型和浏览器类型等，如何利用机器学习方法，结合传统的识别技术，高精度和高准确率地识别出设备型号、浏览器型号、操作系统型号等将是本文下一步的主要研究工作。

## 参 考 文 献


[1] 中国互联网络信息中心. 第 38 次中国互联网络发展状况统计报告[EB/OL]. [2016-08-01]. http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/201608/P020160803367337470363.pdf.

[2] Cisco Visual Networking Index:Global Mobile Data Traffic Forecast Update,2015-2020 White Paper[OL]. http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html#~mobile-forecast.2015.

[3] MAIER G,SCHNEIDER F,FELDMANN A. A first look at mobile hand-held device traffic[C]// International Conference on Passive and Active Network Measurement. Springer Berlin Heidelberg,2010:161-170.

[4] CHEN X,JIN R,SUH K,et al. Network performance of smart mobile handhelds in a university campus WiFi network[C]// Proceedings of the 2012 ACM Conference on Internet Measurement Conference. ACM,2012:315-328.

[5] GEMBER A,ANAND A,AKELLA A. A comparative study of handheld and non-handheld traffic in campus Wi-Fi networks [C]// International Conference on Passive and Active Network Measurement. Springer Berlin Heidelberg,2011:173-183.

[6] GEMBER A,ANAND A,AKELLA A. Handheld vs non-handheld traffic:Implications for campus wifi networks[EB/OL]. [2016-08-06]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.391.4705&rep=rep1&type=pdf.

[7] ZyTrax:Mobile browser id strings[OL]. http://zytrax.com/tech/web/mobile_ids.html.

[8] 关晓蔷. 基于决策树的分类算法研究[D]. 太原：山西大学，2006.

[9] 徐鹏，林森. 基于 C4.5 决策树的流量分类方法[J]. 软件学报，2009,20(10):2692-2704.

[10] DAI Q,ZHANG C,WU H. Research of Decision Tree Classification Algorithm in Data Mining[J]. International Journal of Database Theory and Application,2016,9(5):1-8.

[11] The Bro Network Security Monitor[OL]. http://www.bro.org.

[12] SELLY D,TROELSEN A. Expert ASP . NET 2.0 Advanced Application Design[M]. John Wiley & Sons,2006:29-63.

[13] MARKOV Z,RUSSELL I. An introduction to the WEKA data mining system[J]. ACM SIGCSE Bulletin,2006,38(3):367-368.


（上接第 269 页）


[7] 朱超. 基于 Openflow 的软件定义网络路由技术研究[D]. 合肥：中国科技大学，2014.

[8] 吴志强，吴艳浩. 基于 Openflow 的拥塞控制机制研究[J]. 河南理工大学学报，2015(4):1673-9787.

[9] HWANG J,YOO J,LEE S H,et al. Scalable Congestion Control Protocol Based on SDN in Data Center Networks[C]// 2015 IEEE Global Communications Conference (GLOBECOM). San Diego,CA,2015:1-6.

[10] KANAGEVLU R,AUNG K M M. SDN Controlled Local Rerouting to Reduce Congestion in Cloud Data Center[C]// 2015 International Conference on Cloud Computing Research and Innovation (ICCCRI). Singapore,2015:80-88.

[11] CHIM T W,YEUNG K L,LUI K S. Traffic distribution over equal-cost-multi-paths[J]. Computer Networks the International Journal of Computer & Telecommunications Networking,2005,49(4):465-475.

[12] Ryu 3.20 documentation[EB/OL]. [2015-3-12]. http://ryu.readthedocs.org/en/latest/. 2012.

[13] GREENBERG A,HAMILTON J,MALTZ D A,et al. The cost of a cloud:research problems in data center networks [J]. ACM SIGCOMM,2008(8):68-73.