MgdFlow：微电网场景下的多粒度数据流管理算法

荆有波[1,2]，曹清越[1]，朱瑞[1]

（1. 郑州大学河南先进技术研究院，河南 郑州 450003；2. 中国科学院微电子研究所，北京 100029）

摘 要：针对软件定义网络（SDN）设备被动响应式插入流表项导致表项空间占用和资源调度频繁的问题，提出了一种基于业务功能的多粒度数据流管理算法（MgdFlow）。通过延迟排序和多重聚合与分流，减少了流表项插入的数目和调度指令的次数。实验数据表明，所提算法在负载均衡性能方面比自适应两级流表提升 18%，在平均表项占用方面比传统 OpenFlow 方案提升 9%，在定义的控制器调度资源比率 ImproveQoS 方面比 2 种对比方案提升 24%和 12%。

关键词：软件定义网络；流表管理；负载均衡；服务质量

中图分类号：TP393.02

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2023175

MgdFlow: multi-granularity data flow management algorithm in microgrid scenario

JING Youbo[1,2], CAO Qingyue[1], ZHU Rui[1]

1. Henan Institutes of Advanced Technology, Zhengzhou University, Zhengzhou 450003, China

2. Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029, China

Abstract: Aiming at the problem of frequent entry space occupation and resource scheduling caused by passive response insertion of flow entries by software defined network (SDN) devices, a operational function-based multi-granularity data flow management algorithm (MgdFlow) was proposed. Through delayed sorting and multiple aggregation and splitting, the number of flow entry insertions and scheduling instructions were reduced. Experimental data shows that the load balancing performance is 18% higher than the adaptive two-stage flow, and the average entry occupation is 9% lower than the traditional OpenFlow scheme. There are also 24% and 12% improvements in the defined controller scheduling resource ratio ImproveQoS compared to other schemes.

Keywords: software defined network, flow table management, load balancing, QoS

0 引言

微电网是指由分布式电源、储能装置、能量转换装置等组成的小型配电系统，它以分布式电源为主，利用储能和控制装置进行调节，实现网络内部的电力平衡。微电网的出现丰富了传统电网的运行模式，提高了供电系统的灵活性和可靠性[1]。由于其发电设备多为分布式[2]，具有分布稀疏、发电不连续且不确定、运行随机性大等缺陷，传统的集中式优化方式难以揭示多主体之间的交互行为[3]，因此对微电网运行情况的调度控制需要全局动态信息，同时还要保证在电网灾变等特殊情况下供电不受影响。

软件定义网络（SDN, software defined network）是一种网络虚拟化方法，被誉为下一代网络的关键技术[4]，其通过 OpenFlow 分离网络的控

收稿日期：2023−02−17；**修回日期：**2023−08−31

通信作者：曹清越，cqy@gs.zzu.edu.cn

基金项目：国家重点研发计划基金资助项目（No.2018YFC0823900）

Foundation Item: The National Key Research and Development Program of China (No.2018YFC0823900)

制平面和数据平面，从而实现对网络的可编程性。近年来，随着网络应用的迅速发展，SDN 逐渐在学术界和工业界备受关注[4-6]。SDN 通过将网络设备的控制平面与数据平面分离，以及逻辑上的集中控制，可以实现对网络流量的灵活控制，促使网络资源得到更充分的利用，推动网络创新的进一步发展。

SDN 的控制转发分离技术能很好地满足微电网全局调度、及时处置突发情况的需求，通过集中控制特性，能够感知全局网络拓扑结构及通信资源使用情况，具有合理分配带宽、灵活调度业务、改善服务质量（QoS, quality of service）的功能[7]，突发的、不在预设方案内的网络情况会触发 packet-in 消息，从而通过控制器下发流表解决实际网络问题，但在 SDN 应用中这种 OpenFlow 被动式插入流表项[6]的控制方式容易导致存储占用和调度资源过频，从而影响微电网中网络的健康状况。

在 SDN 中，通常在网络的初始化阶段采用主动式放置流表项以满足对网络带宽和网络时延方面有要求的应用，而被动式放置流表项的方式通常被侧重负载均衡的应用所采用，2 种方式结合起来能够让 SDN 充分调整网络中的负载压力，让拥挤的路由得到及时的缓解，让空闲的设备得到更充分的利用。然而，每台 OpenFlow 交换机所能存储的流表项数目和流条目处理能力有限[8]，导致现有的流量调度方案存在以下问题。1) 负载均衡得不到保证：目前商用 OpenFlow 交换机至多能支持存储数万条流表项，突发流量的激增、流处理能力的不足和流表溢出[9]会导致活动路由的流量分布不均，部分路由过分拥挤甚至堵塞，影响负载均衡的实现。2) 服务质量产生波动：随着涌入网络的流量增加，丢包率也不断提升，那些没有主动式配置流表项的流量会促发 packet-in 消息风暴，导致控制器不得不做出大量响应处理这些携带流信息的数据包，与此同时，交换机设备的流表项也无法及时进行增删改操作，进一步延缓了后续流量的处理，恶化了网络服务质量。

为了解决以上问题，实现微电网中活动路由的负载均衡和服务质量，SDN 控制器需要考虑每条路由的链路负载和每个流带来的负载，并且主动式地插入引导式流表项使瞬间大量涌入的流量可以按照网络中的已有路由均匀分发，在多个节点对这些流量进行引导将极大提升网络的负载均衡性能以及网络服务质量，最后阶段利用细粒度的被动式调度插入相应流表项完成流的运输任务。

1　相关工作

为研究在保持网络设备正常负载的情况下，减少各设备的存储占用和冗余损耗，国内外的诸多学者已经做了大量的研究。Shirali-Shahreza 等[10]提出了延迟安装与加速驱逐 2 种技术，通过使用 TCP RST/FIN 数据包预测 TCP 流何时终止来模拟及时驱逐流表规则以及利用延迟安装来弥补针对非 TCP 流方案的缺陷，从而极大地减少了流表项占用率。Guo 等[11]提出了动态流调度方案（AggreFlow），通过流集路由、延迟重路由和自适应重路由 3 种技术灵活地对流进行调度，并且将大量同时进行的重路由操作分摊在一段相对较长的时间内，实现了在较低开销内活动路由的负载均衡和较低功耗与延迟。Soliman 等[12]提出了一种基于多协议标签交换（MPLS）标签的转发方案，交换机根据这些携带路由信息的 MPLS 标签来转发数据包，从而降低了对流表项的占用，但也带来了额外的损耗，尤其在小数据流情况下更加明显。文献[13]提出利用数据包包头的可用 VLAN 标识符（VID）来携带路由信息，并且在指定的交换机上放置流表项减少了 SDN 中流表的使用，在不同新流量到达的单播和多播场景，降低了流拒绝率，极大地推迟了第一次流量拒绝时间。李佳[14]提出的自适应两级流表超时机制，采用了两级流表架构，根据流的到达时间特征不断修改流表项中主流表的空闲超时与次级流表的硬超时，从而减少了 packet-in 消息的产生，并且提高了流表空间的利用率。刘振鹏等[15]提出了基于时序与集合的 SDN 流表更新策略，将网络中的交换机按照新旧路由分类，分类集合按次序分别更新流表，保证了流表更新的一致性，并且减少了流表空间的占用。曾友雯等[16]采用 OpenFlow 交换机的服务器负载均衡策略，利用多地址定向流表对服务请求进行分类，通过蚁群算法求解最佳负载重定向方案，降低了流表项数目规模，比传统负载策略的性能更优越。马晓航[17]通过动态调节流表项的超时时间和超时方式，对控制通道占用率、流表项存活数、大象流侦测精度 3 个目标进行联立优化，其设计的混合超时机制的优化效果优于硬超时和空闲超时。严可意[18]提出的

区分大象流和老鼠流的检测算法以及动态预测并设置流表项超时时间，不仅减少了交换机与控制器的频繁通信开销，还提升了流表利用率。齐婵等[19]针对 SDN 配置更新所引起的数据包处理不一致问题，提出一种基于分类和时序的 SDN 流表更新一致性方案，有效控制了交换机存储空间的占用率并降低了控制器的负载。

以上方案在微电网场景下的应用中都存在不足。Shirali-Shahreza 等[10]提出的方案针对 TCP 流具有良好的节省表项占用效率，但对于非 TCP 流的处理方案不能满足微电网设备运行间歇性大、具有随机性的需求。Guo 等[11]提出的 AggreFlow 对微电网设备出现的随机性流进行聚合不够高效。Soliman 等[12]提出的方案处理分布稀疏的设备产生的小流量时会导致更大的资源损耗。李佳[14]提出的方案能够减少重定向流的调度资源，但比传统 OpenFlow 方式又增加了流表项数量。刘振鹏等[15]提出的流表更新一致性策略减少了控制负载和流表项占用，但不适于微电网场景下复杂的网络环境。曾友雯等[16]提出的负载均衡策略较传统算法有更优的性能，但是要求预先部署多地址定向流表；文献[17-19]分别提出的动态优化机制和流表更新一致性能够提升流表的利用率，但对具备相同特征的流表项没有聚合利用。

针对微电网设备容易带来的调度指令过频和存储占用问题，本文提出了一种通过延迟排序和多重聚合分流的数据流（MgdFlow, multi-granularity data flow）管理算法，在保证良好网络通畅的情况下，比传统 OpenFlow 方案[6]和自适应两级流表方案[14]减少了交换机中平均流表项的安装数目和调度指令下发次数，在负载均衡和降低流表项占有率方面的性能更优越。

2 系统模型

2.1 目标函数

1) 负载均衡。负载均衡主要考虑 2 个指标：交换机的平均负载情况和交换机的平均表项占有率[20-22]。交换机的最大负载均衡路由决定了自身负载均衡表项的上限，而通过粗粒度聚合的流表划分，将路由尽可能均匀地发送到网络中的各台交换机，避免极端情况下交换机超出最大负载利用率。由网络平均负载与各交换机负载的均方根误差（RMSE）来评估网络中整体负载均衡，计算式为

$$\mathrm{fun}_1=\sqrt{\frac{\left(\mathrm{rate}_s-\mathrm{rate}_{\mathrm{avg}}\right)^2}{h}} \tag{1}$$

交换机的硬件条件限制了其存储空间的大小，因而对流表项的存储存在着严重的限制[8-9,23]。通过聚合与分流可以极大限度地减少交换机中流表项的插入，以交换机平均表项占有率衡量该算法的效果，计算式为

$$\mathrm{fun}_2=\frac{\sum_1^j \mathrm{so}_j+\sum_1^k \mathrm{so}_k}{h} \tag{2}$$

2) ImproveQoS。通常 QoS 考虑的指标为可用性、吞吐量、时延变化等，在本文 SDN 应用场景下，主要考虑 2 个指标：丢包率、控制器下发消息总数量。一是因为控制器下发控制新流消息的开销会带来路由开销，在聚合与分流策略的作用下，可以尽可能减少新流加入时控制器的相应消息，在入口交换机中用于粗粒度聚合的流表项增多时效果更明显。二是因为丢包率与流集匹配路由有关，更多的流能够直接匹配流集路由进入网络中，而不需要等待控制器下发指令，网络的处理能力更好。因此本文提出一种新的考量指标 ImproveQoS，即需要重路由的流占所有流的比例为

$$\mathrm{fun}_3=\frac{\sum f-\sum_{f\in \mathrm{FS}} \mathrm{fs}_n}{\sum f} \tag{3}$$

其中，f 表示进入网络中的任意一个流，fs_n 表示对网络中的流聚合成的第 n 个流集。$\sum f$ 表示进入网络的所有流，用其减去在流集中匹配处理的流后得到需要控制器指令下发调度的流数量。

2.2 限制条件

1) 交换机负载限制。S 表示交换机的集合，$s_i\in S$ 表示每台交换机，Route_i 表示在时间 t 经过交换机 s_i 的路由，每台交换机处理流的总速率不能超过上限 v，则有

$$\mathrm{rate}(\mathrm{Route}_i,t)\leqslant v \tag{4}$$

2) 交换机存储空间限制。SO 表示所有交换机流表项的集合，每台交换机存储的流表项 $\mathrm{so}_i\in \mathrm{SO}$ 不能超过上限 r，则有

$$\mathrm{so}_i\leqslant r \tag{5}$$

3) 交换机数量限制。粗粒度聚合阶段交换机总数和细粒度分流阶段交换机总数之和等于网络总交换机数，则有

$$j+k=h \tag{6}$$

2.3　问题描述

针对当前微电网场景下的实际问题，通过减少交换机平均流表项占有率和减小交换机的平均负载，从而达到网络整体负载均衡；同时还需要减少丢包率和控制消息的下发数目，从而提高 ImproveQos 整体表现。因此，针对这 3 个优化目标，问题可以被量化为

$$\min\left(w_1 \mathrm{fun}_1 + w_2 \mathrm{fun}_2 + w_3 \mathrm{fun}_3\right) \tag{7}$$

其中，w_1, w_2, w_3 为赋予该 3 个优化目标的权重常量，且 $w_1 + w_2 + w_3 = 1$。当 $w_1 = 1$ 且 w_2, w_3 为 0 时，表示仅考虑网络的平均负载能力；当 $w_2 = 1$ 且 w_1, w_3 为 0 时，表示仅考虑交换机的存储占用空间；当 $w_3 = 1$ 且 w_1, w_2 为 0 时，表示仅考虑控制器调度资源的损耗。在本文实验中，设 $w_1 = w_2 = w_3 = \frac{1}{3}$，即网络平均负载，交换机存储空间和控制器调度资源损耗具有相同的优先级。

3　MgdFlow 算法设计

3.1　主要思想

以单个数据包为粒度，将进入网络的流先重定向加入一个 200 μs 的缓冲队列，设置队列中可以缓冲的最大数据包个数为 16 个，在该缓冲时间内对队列中的数据包进行排序，初步聚合属于同一流的数据包，从而降低同一流的数据包中交换机中匹配表项的 miss 概率，减少 CPU 调度指令的数量进而增加吞吐量、提高网络的负载均衡性能。针对加入一段时延缓冲队列排序是否会对整体时延造成更大影响的问题，Hamid 等[24]所构建的 reframer 工具中已经充分验证了数据包排序能够极大地提高服务器内存层次的利用率，从而提升了时延、CPU 利用率等性能指标，因此在后续的求解算法中不再考虑这一问题。

图 1 展示了一个延时排序的简单示例。在设定的 200 μs 的缓冲队列中，充斥着属于不同流 A、B、C 的数据包，通过对乱序数据包打上不同的标签进行分类，随后将这三类标签对于流的处理规则写入入口交换机中，从而完成了不同流中交织的乱序数据包的处理。

以主要业务功能为条件，将经过初步时延排序的不同流再次进行基于流的粗粒度聚合和细粒度分流。在粗粒度聚合阶段，以目的 IP 地址为分流依据，将不同的流分为 N 个流集，分别指向 N 个中转交换机，同时在入口边缘交换机处插入 N 个用作粗粒度聚合流的流表项，从入口交换机到 N 个不同的交换机之间共存在 M 条路由，并在这些路由经过的交换机上插入指向相应中转交换机的流表项，从而实现粗粒度聚合阶段流到达中转交换机的功能。在细粒度分流阶段，以源 IP 地址为分流依据，将所需大业务下的不同小业务流进行分发，在中转交换机处插入细粒度分流的流表项；同时，对于中转交换机可能存在的流表项占用率高以及网络中 mice flow 占据了绝大多数的流表项的情况[18,25]，将所有中转交换机的流表项的 idle_time 从一般的 5 s 修改为 1 s，达到快速淘汰 mice flow 对应表项。

图 2 展示了聚合与分流示例，在经过初步时延排序处理后，当前网络的处理对象由有各种具备不同属性（如源 IP 地址、目的 IP 地址、源 MAC

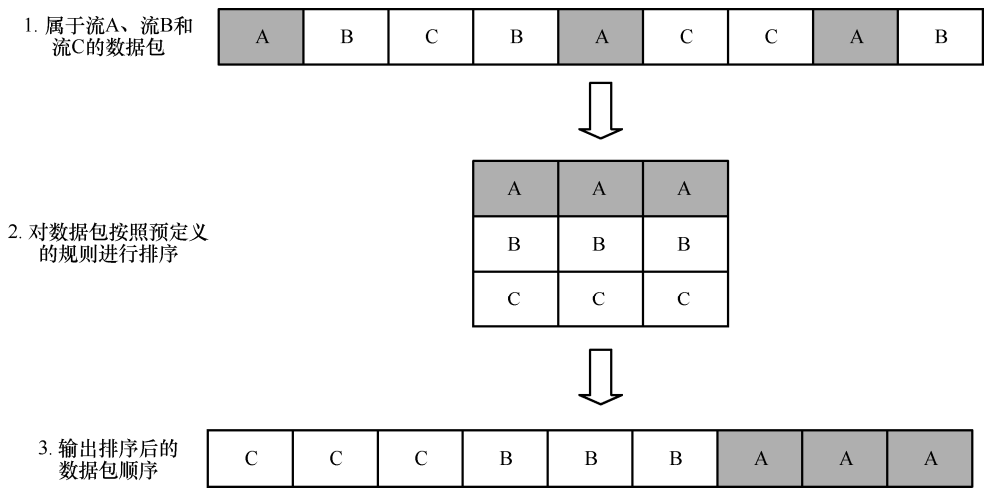


图 1　延时排序示例

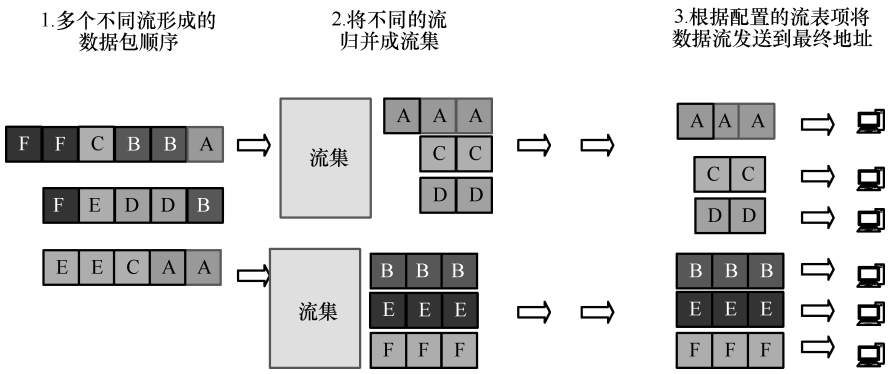


图 2　聚合与分流示例

地址、目的 MAC 地址等）的属于不同流 A、B、C、D、E、F 的数据包组成，通过主动式放置在入口交换机的流表项粗粒度聚合成流集，在本例中，流集 a 包括流 A、C、D，流集 b 包括流 B、E、F，在第一步的粗粒度聚合中，根据流集分成两大类；在第二步的细粒度分流中，由交换机响应式细粒度地插入流表项完成转发到终端的任务，经过 N 次粗粒度聚合的筛选，最终流集中的其他流也发送到了终端。

本文将微电网场景下的负载、存储占用和指令调度抽象为一个多目标优化问题，MgdFlow 模型的符号及含义如表 1 所示。使用 $N=(S,L)$ 来描述网络，其中，S 表示交换机集合包含 h 台交换机，并将其分别划分给 j 个粗粒度阶段网络和 k 个细粒度阶段网络，故 $S=\left[s_1,s_2,\cdots,s_j,\cdots,s_h\right]$；$L$ 表示 S 中交换机之间的 q 条链路，故 $L=\left[l_1,l_2,\cdots,l_q\right]$。

表 1　MgdFlow 模型的符号及含义

符号	含义
s_h	第 h 台交换机
l_q	交换机之间的第 q 条链路
Rate_f	流 f 的速率
fs_n	对网络中的流聚合成的第 n 个流集
route_m	网络设备和链路组成的第 m 条路由
so_j	粗粒度网络中第 j 台交换机的表项
so_k	细粒度网络中第 k 台交换机的表项
$a_{\text{route}}^{\text{fs}}$	若流集 fs_n 要通过路由 Route，则为 1，否则为 0
$b_{\text{route}_m'}^{\text{fs}_g'}$	若流集 fs_g' 要经过路由 Route′，则为 1，否则为 0

3.2　算法描述

在网络运行时间 t 中，流 f 具有以下属性：流速率 Rate_f、源 IP 地址 A_{src}^f、目的 IP 地址 A_{des}^f、开始时间 T_{start}^f、结束时间 T_{end}^f。因此，流 f 可以表示为 $f=\left[\text{Rate}_f,A_{\text{src}}^f,A_{\text{des}}^f,T_{\text{start}}^f,T_{\text{end}}^f\right]$。在粗粒度聚合流阶段，在入口交换机处，对所有流 f 进行聚合后形成 n 个流集，即 $\text{FS}=\left\{\text{fs}_1,\text{fs}_2,\cdots,\text{fs}_n\right\}$。将 n 个用来分类聚合的流表项插入入口交换机，网络中所有设备和链路共同组成了 m 条传输路由，即 $\text{Route}=\left\{\text{route}_1,\text{route}_2,\cdots,\text{route}_m\right\}$。一条路由上的一台交换机可能会有多个流集经过，因此，在粗粒度聚合阶段的所有交换机流表项数 $\text{SO}=\left[\text{so}_1,\text{so}_2,\cdots,\text{so}_j\right]$，任一台交换机中流表项数量表示为

$$\text{so}_j=\sum_{s\in\text{Route}}\text{route}_m\sum_{f\in\text{fs}_n}x \tag{8}$$

其中，$x=1$ 表示流集被分配给 route_m，$x=0$ 表示流集没有被分配给 route_m。

使用 AF_n 表示经过某一流集 fs_n 的所有流，若流集 fs_n 要通过路由 Route，则设 $a_{\text{route}}^{\text{fs}}=1$，否则 $a_{\text{route}}^{\text{fs}}=0$；则在时间 t，并且满足 $T_{\text{start}}^f\leqslant t\leqslant T_{\text{end}}^f$ 时，令条件参数 $\theta=1$，否则 $\theta=0$；交换机 $s\in\text{Route}$，通过的流的总速率表示为

$$\text{rate}(\text{Route}_i,t)=\sum_{s\in\text{Route}}\text{route}_m\sum_{f\in\text{AF}_n}\text{Rate}_f\,\theta a_{\text{route}}^{\text{fs}} \tag{9}$$

在细粒度分流阶段，在 n 台中转交换机处，对已经粗粒度过滤的流集再进行细粒度的划分，将其分为 g 个流集，并在每台中转交换机处将 g 个用于进一步分流的流表项插入，此时将有 ng 个流表项插入；网络中所有设备和链路组成了 m' 条传输路由，即 $\text{Route}'=\{\text{route}_1',\text{route}_2',\cdots,\text{route}_m'\}$。

对流集 fs_n 进行再次划分，则有 $\text{fs}_n=\{\text{fs}_1',\text{fs}_2',\cdots,\text{fs}_g'\}$。

细粒度分流阶段的所有交换机流表项数量 $SO=[so_1,so_2,\cdots,so_k]$，任一台交换机中流表项数量表示为

$$so_k=\sum_{s\in Route'} route'_m \sum_{f\in fs'_g} x \tag{10}$$

其中，$x=1$ 表示流集被分配给 $route'_m$，$x=0$ 表示流集没有被分配给 $route_m$。

采用 AF'_n 表示经过某一流集 fs'_g 的所有流，若流集 fs'_g 要经过路由 $Route'$，则设 $b^{fs}_{route^{fs'_g}_{m'}}=1$，反之则为 0；同理，在时间 t，并且满足 $T^f_{start}\leqslant t\leqslant T^f_{end}$ 时，令条件参数 $\theta=1$，否则 $\theta=0$；交换机 $s\in Route'$，通过的流的总速率表示为

$$rate(Route',t)=\sum_{s\in Route'} route'_m \sum_{f\in AF'_n} Rate_f\,\theta b^{fs}_{route^{fs'_g}_{m'}} \tag{11}$$

3.3　算法求解

根据微电网场景下整体路由建立的过程，可以概括如下。

1) SDN 控制器根据流表项占用和网络链路负载更新粗粒度聚合节点的表项和整体动态视图拓扑。

2) 粗粒度聚合节点发出路由请求，转到步骤 3)，否则转到步骤 1)。

3) SDN 控制器依次响应插入流表项引导达到每一跳的节点并记录该条路由的表项数量。

4) 重复步骤 1)~步骤 3)，直至获取所有链路和路由的流表项占用和负载情况。

5) 更新粗粒度聚合节点的流表项并生成新的流集。

6) 当表项空间不足或负载指标达到阈值时，增大粗粒度聚合节点的范围并在损耗多的链路路由进行再次聚合与分流。

7) 网络节点根据插入的表项完成路由。

4　实验仿真及分析

4.1　实验环境与配置

为模拟微电网场景下的 SDN 设备情况，研究在 ubuntu 环境下利用 2.3.0 版本的 Mininet 网络仿真器搭建 SDN 仿真环境，并且采用基于 Java 的 15.0.0 版本的 OpenDayLight（ODL）作为 SDN 控制器，Open vSwitch 作为虚拟交换机，软件版本和硬件配置如表 2 和表 3 所示。

表 2　软件版本

软件名称	版本号
Mininet	2.3.0
OpenDayLight	15.0.0
Open vSwitch	2.10.0
Apache Maven	3.6.3

表 3　硬件配置

硬件名称	型号
主机 CPU	Intel(R) Core(TM)i5-7200U 2.5 GHz
主机内存	16GB DDR4 2133 MHz
主机系统	Windows10 专业版
虚拟机内存	4 GB
虚拟机硬盘	100 GB
虚拟机系统	Ubuntu16.04

构建实验网络拓扑如图 3 所示。实验网络由 18 个节点组成，其中包含 9 台交换机、8 台终端主机和一台控制器 c_0。网络拓扑中的 s_1 为入口交换机，插入了不同粗粒度的聚合流集的流表项，用来对进入流量进行初步聚合发送；s_4、s_5、s_6 为中转交换机，插入了进行第二次聚合的粗粒度的流表项，用来对进入的流量进行二次分流发送；s_7、s_8、s_9 为出口交换机，插入了精确的细粒度流表项，完成最后的传输任务。h_8 主机以固定速率向 h_1～h_7 主机发送数据包，网络的传输速率控制在 500~800 Mbit/s，对实验节点进行 10 次仿真测试，并利用 Wireshark 在各个节点进行抓包分析与记录。

4.2　评估指标

1) 交换机平均负载 RMSE。交换机与链路的负载决定了整个网络的最大流量的容载上限，每条链路和交换机的负载都影响到整体的最大容载率，某条链路的过载可能导致某条路由甚至几条路由的拥塞和崩溃，因此针对已有网络中的交换机负载和链路承载上限，本文通过计算每条路由上的交换机平均负载情况，并计算整体的 RMSE 来评估网络中整体链路的承载状态。

2) 交换机平均流表项占有率。在具备一定规模的 SDN 中，OpenFlow 交换机需要通过存储大量的流表项来应对处理进入网络的流量，受限于三态内容寻址存储器（TCAM, ternary content addressable memory）内存容量，流表所能存储的流表项数目也是有限的，而 TCAM 造价昂贵且耗能，所以通过增

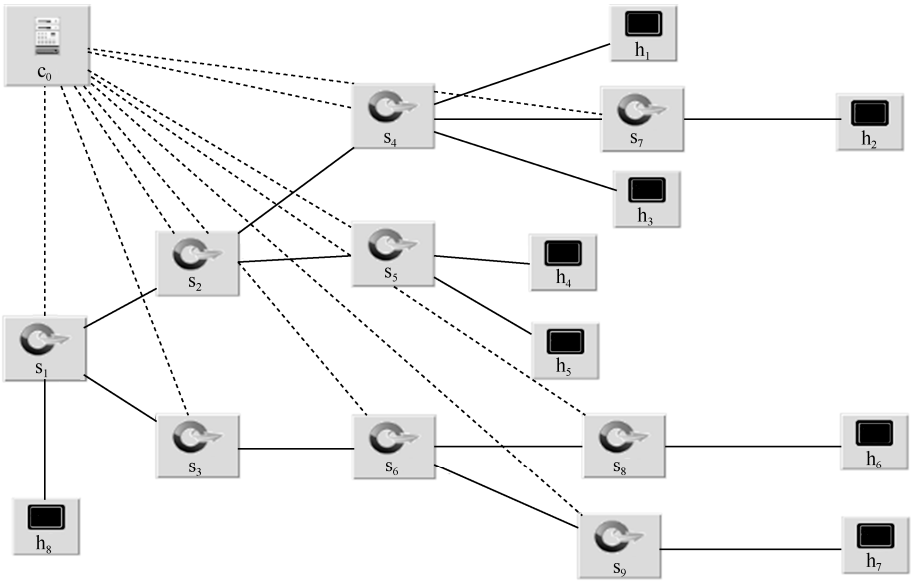


图 3 ODL 控制器下的网络拓扑

加 TCAM 容量来应对大流量的流表项数目是不现实的[8-9,26]，因此针对一台 OpenFlow 交换机的存储空间十分有限的现状，聚焦于通过尽可能减少流表项占用，以最精简的存储策略来完成网络流量的处理。交换机流表项的占用空间反映了维持一个网络正常运行下该节点所需要占用的空间代价，平均流表项占用率则反映了整个网络设备中所有节点的平均存储空间占用情况，这一指标不同于最大流表项占用率和中位数流表项占用率可能受制于极大值或者极小值存储占用的影响，而是由整体设备的存储占用所决定，能够真实地反映出网络各设备的存储情况。

3) ImproveQoS。随着网络负载的增加，涌入的流量逐渐增多，超过了原本设定的阈值，导致丢包率增大，流量因得不到及时的处理而阻塞在各个节点。同时，没有匹配流表项的流量会激发 table-miss 表项，使交换机向控制器发送 packet-in 消息并将这种类型的数据包发给控制器处理，导致控制器调度信息的下发[27]，增加了资源的损耗与负载。通过关注丢包率与控制器下发消息的数目，并将其归结为需要重定向处理消息的流占总流的比例，从而衡量在不同策略下网络的服务质量。

4.3 多目标优化结果与分析

1) 负载均衡性能

通过对比传统 OpenFlow 方案[6]、自适应两级流表方案[14]以及本文提出的 MgdFlow，得到 3 种方案下网络链路负载情况，如图 4 所示。其中，RMSE 越小则性能越好。箱线图的上横线和下横线分别代表第三四分位数（Q3）和第一四分位数（Q1），也就是数据的上下 $\frac{1}{4}$ 分位，箱线图的高度表示数据的 $\frac{1}{2}$ 分位；同时，每个箱线图数据的平均数用三角形标出，可以看出不同方案下得出的指标数据的总体分布。

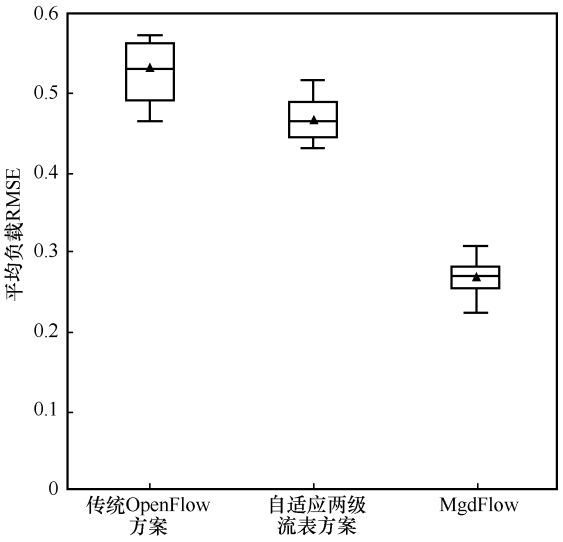


图 4 3 种方案不同方案下网络链路负载情况

2) 平均流表项占有率

通过对比传统 OpenFlow 方案、自适应两级流表方案以及 MgdFlow，得到 3 种方案在不同路由节点数量下的网络所需流表项，如图 5 所示。从图 5 可知，随着网络规模逐渐扩大，路由节点数量逐渐增多，被添加到各台交换机中的流表项数量不断增加，自适应两级流表方案对流表项的损耗最多，传

统 OpenFlow 方案其次，MgdFlow 方案损耗的流表项数始终低于这 2 种方案。

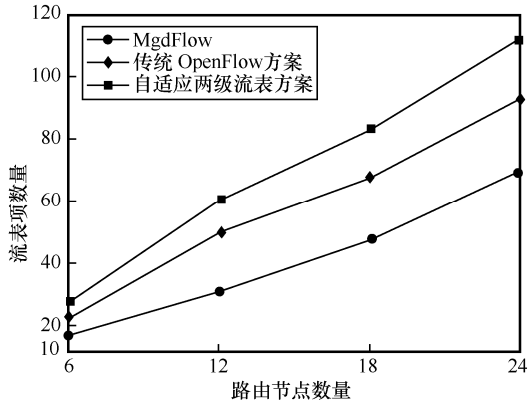


图 5　3 种方案在不同路由节点数量下的网络所需流表项

3) ImproveQoS

通过对比传统 OpenFlow 方案、自适应两级流表方案以及 MgdFlow，得到 3 种方案在不同路由节点数量下出现网络拥塞时需要重定向流量占比，如图 6 所示。从图 6 可知，MgdFlow 的 ImproveQoS 指标始终低于另外 2 种方案。当流量激增时，网络的丢包率增加，对交织在不同流中的数据包进行访问就会反复调用线程，此时会增加对调度信息的需求，在需要重定向的流中，大象流占比越少、老鼠流占比越多，则同一丢包率下的 ImproveQoS 越高。

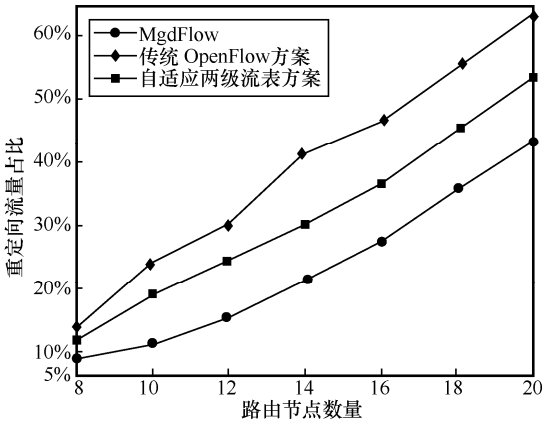


图 6　3 种方案在不同路由节点数量下出现网络拥塞时需要重定向流量占比

3 种方案对比如表 4 所示。从表 4 可以看出，与传统 OpenFlow 方案和自适应两级流表方案相比，在负载均衡方面，MgdFlow 分别提升了 25%和 18%；在平均流表项占有率方面，MgdFlow 分别提升了 9%和 16%；在 ImproveQoS 方面，MgdFlow 分别提升了 24%和 12%。

表 4　　3 种方案对比

方案	负载均衡	平均流表项占有率	ImproveQoS
传统 OpenFlow 方案	53%	36%	47%
自适应两级流表方案	46%	43%	35%
MgdFlow	28%	27%	23%

5　结束语

针对微电网需要全局调度、及时处理突发情况的需求，结合 SDN 的控制转发分离技术，提出了针对微电网业务场景下的多粒度数据管理算法 MgdFlow，通过对数据包进行延时排序和对数据流完成聚合与分流，在 SDN 中有效提高负载均衡性能和服务质量，降低平均交换机流表项的占有率和调度指令的下发次数，提高整体网络对于流量的处理能力。对比传统 OpenFlow 方案和自适应两级流表方案，本文方案在负载均衡性能、存储表项利用和调度指令占用方面都有明显的改进和较高的可用性。虽然延时缓冲队列排序会增加一段时延，但对整体网络性能的提升却是巨大的，本文中加入的缓冲队列是静态设置的，未来的工作将研究在微电网场景下的自适应延时缓冲队列排序，从而进一步提升性能。

参考文献：

[1] 郭金明, 李欣然, 邓威, 等. 基于 2 层规划的间歇性分布式电源及无功补偿综合优化配置[J]. 中国电机工程学报, 2013, 33(28): 25-33, 6.
GUO J M, LI X R, DENG W, et al. Comprehensive optimal allocation of intermittent distributed generation and reactive power compensation based on bilevel planning[J]. Proceedings of the CSEE, 2013, 33(28): 25-33, 6.

[2] 陈其森, 汪湘晋, 池伟, 等. 多微电网互联系统能量管理方法研究[J]. 电力系统保护与控制, 2018, 46(11): 83-91.
CHEN Q S, WANG X J, CHI W, et al. Research on energy management method of multi-microgrids[J]. Power System Protection and Control, 2018, 46(11): 83-91.

[3] 曹正斐, 张忠辉, 董治成, 等. 基于区块链的多互联微电网分布式协调优化调度[J]. 电力系统及其自动化学报, 2022, 34(9): 138-145.
CAO Z F, ZHANG Z H, DONG Z C, et al. Distributed coordination and optimization scheduling of multi-interconnected microgrids based on block chain[J]. Proceedings of the CSU-EPSA, 2022, 34(9): 138-145.

[4] 郭泽华, 窦松石, 齐力, 等. 面向软件定义广域网的路径可编程性保障研究综述[J]. 电子与信息学报, 2023, 45(5): 1899-1910.
GUO Z H, DOU S S, QI L, et al. A survey of maintaining the path programmability in software-defined wide area networks[J]. Journal of Electronics & Information Technology, 2023, 45(5): 1899-1910.

[5] 张朝昆, 崔勇, 唐翯祎, 等. 软件定义网络(SDN)研究进展[J]. 软件学报, 2015, 26(1): 62-81.
ZHANG C K, CUI Y, TANG H Y, et al. State-of-the-art survey on software-defined networking (SDN)[J]. Journal of Software, 2015, 26(1): 62-81.

[6] 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.
ZUO Q Y, CHEN M, ZHAO G S, et al. Research on OpenFlow-based SDN technologies[J]. Journal of Software, 2013, 24(5): 1078-1097.

[7] 张进进. 基于 SDN 的电力通信网 QoS 技术研究[D]. 重庆: 重庆邮电大学, 2020.
ZHANG J J. Research on QoS technology of power communication network based on SDN[D]. Chongqing: Chongqing University of Posts and Telecommunications, 2020.

[8] 陈志鹏, 徐明伟, 杨芫. SDN 交换机转发规则 TCAM 存储优化综述[J]. 计算机学报, 2021, 44(7):1341-1362.
CHEN Z P, XU M W, YANG Y. A survey on TCAM storage optimization for SDN switch forwarding rules[J]. Chinese Journal of Computers, 2021, 44(7): 1341-1362.

[9] 谢升旭, 邢长友, 张国敏, 等. OpenFlow 交换机流表溢出缓解技术研究综述[J]. 计算机研究与发展, 2021, 58(7): 1544-1562.
XIE S X, XING C Y, ZHANG G M, et al. Survey of OpenFlow switch flow table overflow mitigation techniques[J]. Journal of Computer Research and Development, 2021, 58(7): 1544-1562.

[10] SHIRALI-SHAHREZA S, GANJALI Y. Delayed installation and expedited eviction: an alternative approach to reduce flow table occupancy in SDN switches[J]. IEEE/ACM Transactions on Networking, 2018, 26(4): 1547-1561.

[11] GUO Z H, XU Y, LIU Y F, et al. AggreFlow: achieving power efficiency, load balancing, and quality of service in data center networks[J]. IEEE/ACM Transactions on Networking, 2021, 29(1): 17-33.

[12] SOLIMAN M, NANDY B, LAMBADARIS I, et al. Exploring source routed forwarding in SDN-based WANs[C]//Proceedings of 2014 IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2014: 3070-3075.

[13] GUO Z H, XU Y, CELLO M, et al. JumpFlow: reducing flow table usage in software-defined networks[J]. Computer Networks, 2015, 92: 300-315.

[14] 李佳. 面向 SDN 的流表超时和路由算法研究[D]. 广州: 华南理工大学, 2020.
LI J. Research on flow table timeout and routing algorithm for SDN[D]. Guangzhou: South China University of Technology, 2020.

[15] 刘振鹏, 李明, 王鑫鹏, 等. 基于时序与集合的 SDN 流表更新策略[J]. 河北大学学报(自然科学版), 2020, 40(4): 427-432.
LIU Z P, LI M, WANG X P, et al. Update strategy of SDN flow table based on time series and sets[J]. Journal of Hebei University (Natural Science Edition), 2020, 40(4): 427-432.

[16] 曾友雯, 李双庆, 邹东升. 采用 OpenFlow 交换机的服务器负载均衡策略[J]. 重庆大学学报, 2021, 44(11): 48-56.
ZENG Y W, LI S Q, ZOU D S. Server load balancing strategy using OpenFlow switch[J]. Journal of Chongqing University (Natural Science Edition), 2021, 44(11): 48-56.

[17] 马晓航. 基于流表项超时的 SDN 网络性能优化[D]. 桂林: 桂林电子科技大学, 2021.
MA X H. Performance optimization of SDN network based on timeout of flow table entry[D]. Guilin: Guilin University of Electronic Technology, 2021.

[18] 严可意. 基于机器学习的 SDN 流表优化研究与实现[D]. 北京: 北京邮电大学, 2021.
YAN K Y. Research and implementation of optimization of SDN flow table based on machine learning[D]. Beijing: Beijing University of Posts and Telecommunications, 2021.

[19] 齐婵, 刘建伟, 毛剑, 等. 基于分类和时序的 SDN 流表更新一致性方案[J]. 计算机应用研究, 2018, 35(11): 3405-3408, 3412.
QI C, LIU J W, MAO J, et al. Classification and sequence based consistent flow update scheme in SDN[J]. Application Research of Computers, 2018, 35(11): 3405-3408, 3412.

[20] PETALE S, THANGARAJ J. Failure-based controller placement in software defined networks[J]. IEEE Transactions on Network and Service Management, 2020, 17(1): 503-516.

[21] KAMATH S, SRIVASTAVA A, KAMATH P, et al. Application aware multiple constraint optimal paths for transport network using SDN[J]. IEEE Transactions on Network and Service Management, 2021, 18(4): 4376-4390.

[22] CHANG Y C, LIN H T, CHU H M, et al. Efficient topology discovery for software-defined networks[J]. IEEE Transactions on Network and Service Management, 2021, 18(2): 1375-1388.

[23] LAI W K, WANG Y C, CHEN Y C, et al. TSSM: time-sharing switch migration to balance loads of distributed SDN controllers[J]. IEEE Transactions on Network and Service Management, 2022, 19(2): 1585-1597.

[24] HAMID G , BARBETTE T, KATSIKA G P, et al. Packet order matters! improving application performance by deliberately delaying packets[C]//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2022: 807-827.

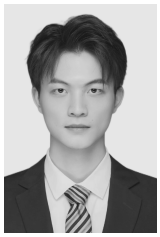
[25] ISYAKU B, ABU B K B, GHALEB F A, et al. Performance evaluation of flowtable eviction mechanisms for software defined networks considering traffic flows variabilities[C]//Proceedings of 2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE). Piscataway: IEEE Press, 2022: 71-75.

[26] QIU K, YUAN J, ZHAO J, et al. FastRule: efficient flow entry updates for TCAM-based OpenFlow switches[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 484-498.

[27] SHANG Z H, WU H, WOLTER K. Buffer management for reducing packet-in messages in OpenFlow networks[C]//Proceedings of 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN). Piscataway: IEEE Press, 2019: 458-465.

[作者简介]

荆有波（1982− ），男，山东潍坊人，郑州大学高级工程师，主要研究方向为无线通信。



曹清越（2000− ），男，湖南益阳人，郑州大学硕士生，主要研究方向为计算机网络。

朱瑞（1998− ），男，江西赣州人，郑州大学硕士生，主要研究方向为边缘计算。