中国科学: 信息科学 2020 年 第 50 卷 第 12 期: 1903–1918
SCIENTIA SINICA Informationis
论文

面向互联网的 SDN 流量多粒度处理机制

卢向敏[1], 王兴伟[1*], 易波[1], 李婕[1], 黄敏[2]

1. 东北大学计算机科学与工程学院, 沈阳 110169
2. 东北大学信息科学与工程学院, 沈阳 110819
* 通信作者. E-mail: wangxw@mail.neu.edu.cn

收稿日期: 2019–09–27; 接受日期: 2020–01–20; 网络出版日期: 2020–11–20

国家重点研发计划 (批准号: 2019YFB1802800) 和国家自然科学基金 (批准号: 61872073, 61572123) 资助项目

摘要 软件定义网络 (software defined networking, SDN) 作为一种新型的网络架构, 将网络的控制平面与数据转发平面分离, 实现了可编程化控制, 为互联网提供了改善网络全局性能的新思路. 虽然 SDN 具有全局视角优势, 但在处理互联网海量数据时也存在性能瓶颈: 频繁的层间通信会使控制器计算效率下降, 海量的流表项数据使得交换机存储压力过大. 为了进一步提升 SDN 的性能使其适应互联网的海量流量处理, 本文提出了面向互联网的 SDN 流量多粒度处理机制, 将 SDN 架构应用到互联网骨干网的流量处理中, 分别从路由和调度两个方面设计并实现了流量多粒度处理机制. 仿真结果表明: 本文设计的流量多粒度处理机制能减少层间通信次数, 减少下发流表项, 维持负载均衡, 提高了路由选取的正确性和有效性, 提升了 SDN 性能, 进而提升了处理互联网海量数据的能力.

关键词 软件定义网络, 互联网, 多粒度路由, 流量调度, 性能优化

1 引言

随着互联网规模的飞速发展, 网络应用不断丰富, 海量数据的传输使得网络拥塞问题日趋严重 [1]. 此外, 传统的以硬件为中心的网络缺乏可管理性、灵活性和可扩展性 [2], 使得互联网的流量处理机制面临巨大挑战. 软件定义网络 (software defined networking, SDN) [3], 将控制平面与数据转发平面分离, 通过集中控制获取全局视图 [4], 以其独特的优势为许多具有挑战性的网络性能优化问题提供了新的解决思路. 但在处理互联网海量数据时, SDN 也存在一些性能瓶颈: 控制器计算效率下降, 交换机存储压力变大. 因此需要进一步提升 SDN 的性能以处理海量数据, 从 SDN 控制器的可扩展性考虑, 分层控制器架构能够增强控制体系结构 [5]. 另一个好的解决思路是控制流量粒度, 利用差异化服务, 对互联网流量进行不同粒度的处理, 从而为将 SDN 应用到互联网提供可能性.

引用格式: 卢向敏, 王兴伟, 易波, 等. 面向互联网的 SDN 流量多粒度处理机制. 中国科学: 信息科学, 2020, 50: 1903–1918, doi: 10.1360/SSI-2019-0214
Lu X M, Wang X W, Yi B, et al. Traffic multi-granularity processing mechanism for Internet-oriented SDN (in Chinese). Sci Sin Inform, 2020, 50: 1903–1918, doi: 10.1360/SSI-2019-0214

本文设计了面向互联网的 SDN 流量多粒度处理机制, 流量处理过程以聚集为计算单位, 基于不同的流量类型, 实现了网络流量的多粒度路由算法和多粒度调度算法. 其中, 多粒度路由分别包括离线和在线两阶段, 离线阶段对尽力而为类型的流量提供粗粒度的路由计算, 而在线阶段则对带宽和时延敏感的带宽预留类型流量提供细粒度的路由计算. 尽管如此, 这两种不同粒度的计算方式分别对应在线和离线两种情形, 这两种情形之间并不冲突, 因此提出的多粒度路由也是一种混合粒度的路由方案. 另外, 对于多粒度调度算法, 它主要是对不同粒度的流量聚集集合进行调度. 不同的聚集集合对应不同数量细粒度网络流量. 提出的多粒度调度算法在对流量汇聚集合进行调度之前, 引入了动态粒度调节的过程. 通过对流量汇聚集合进行动态拆分与合并, 实现了不同粒度的混合调度模式.

本文的主要工作如下: 首先设计了 SDN 流量多粒度处理机制的系统框架; 其次建立了网络结构模型, 包括链路模型、节点模型和流量模型; 最后分别设计了流量多粒度路由机制和流量多粒度调度机制. 本文第 2 节主要介绍 SDN 性能优化相关工作; 第 3 节详细介绍了面向互联网的 SDN 流量多粒度处理机制; 第 4 节基于基准机制对本文提出的机制进行性能评价分析; 第 5 节为结束语.

2 相关工作

要将 SDN 应用于互联网的海量流量处理中, 还需要进一步提升 SDN 的性能. 目前, 对 SDN 性能优化的研究主要从控制层和数据层两个方面展开.

2.1 控制层性能优化相关研究

为了提高 SDN 控制层性能, 文献 [6] 提出了一种自适应的多控制器分配方案, 确定控制器的数量以及控制器和交换机之间的映射关系. 文献 [7] 给出了多目标优化控制器放置问题, 着重于实现网络的高可靠性, 控制器间的负载平衡以及控制器和交换机间的低延迟. 为平衡流表利用率, 文献 [8] 提出了 DIFF 动态路由, 根据流对网络资源的影响来区分流, 为每对源/目标边缘交换机预生成一组路径, 在预生成的路径集中智能地为新流选择路径, 对大象流则自适应地进行重路由.

2.2 数据层性能优化相关研究

为了更好地利用有限流表空间, 文献 [9] 提出了在交换机中添加网络处理器的方式, 但只实现了细粒度的流量处理, 并会带来额外的负载和延迟. 文献 [10] 采用了限制型通配符策略来优化流表项, 只是数据层的处理粒度加大了, 控制层依然是细粒度处理. 文献 [11] 提出了结合多径传输的流聚合机制, 通过引入集中管理的 MPLS (multi-protocol label switching) 标签分发来实现流聚合过程. 能够显著减少流条目数, 且能够减少控制器和交换机之间的通信, 实现了细粒度的流量控制. 文献 [12] 提出了贪婪算法 KSGT (K similar greedy tree), 根据流的转发路径的重合度将所有流划分为若干簇, 来减少和平衡不同 SDN 交换机中的流条目. 然而, 它没有列举所有的解决方案空间. 文献 [13] 指出集成 SDN 和 ICN 以改善网络管理, 适应互联网流量的不断增长.

控制流量的处理粒度是影响 SDN 性能的重要因素, 通过对流量进行分类处理, 能够降低计算复杂度并实现对服务的适应性 [14]. 具体采用何种粒度 (分组级, 流级, 聚集级或其他粒度级别) 的流量控制方式, 需要在控制粒度和性能之间进行仔细权衡. SDN 性能优化的相关研究依然停留在大量的细粒度处理层面, 而缺少通过控制数据流的聚集粒度提供多种流量处理方式的多粒度流量处理方案.

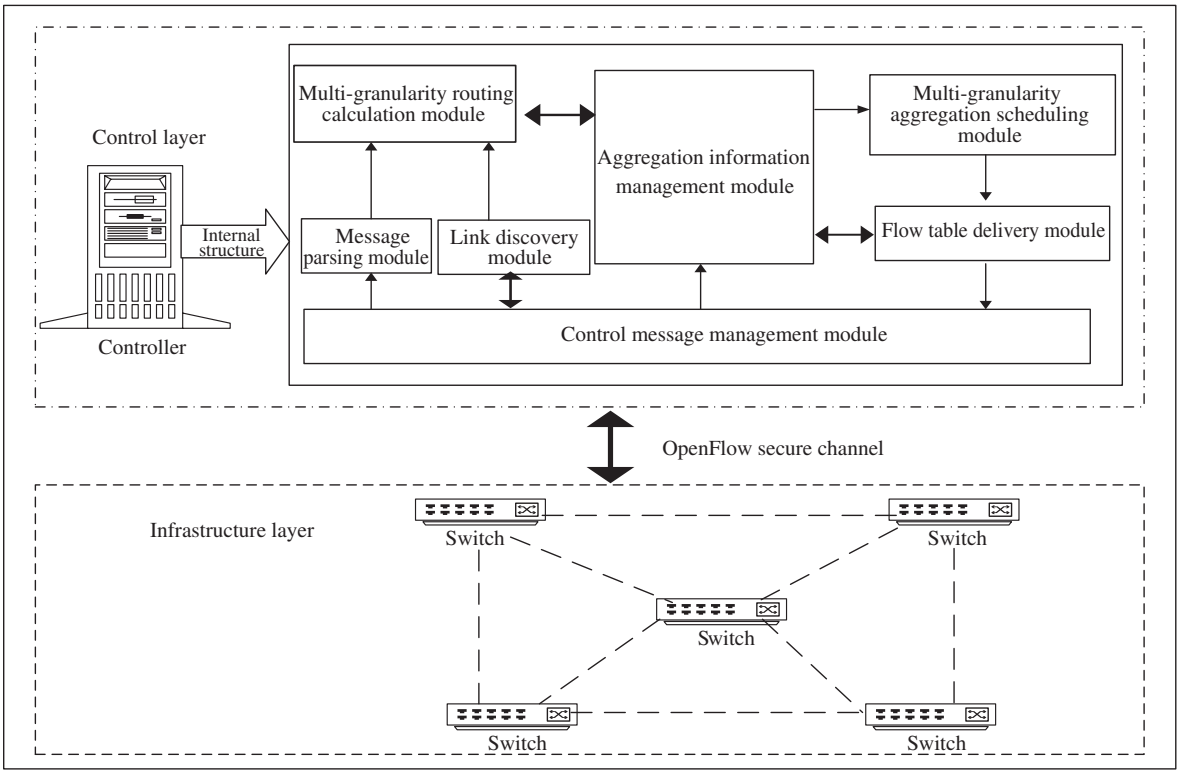

图 1 系统框架图

Figure 1 The system framework

3 面向互联网的 SDN 流量多粒度处理机制

3.1 系统设计

SDN 的性能高度依赖于控制层, 基于多控制器协作的管理方式 [15], 需要考虑控制器间的通信一致性, 通信开销以及负载均衡等问题. 本文提出的流量多粒度处理机制是在控制层中的控制器内部设计了多模块协作的系统框架, 如图 1 所示. 控制层的系统模块主要包括消息解析模块、链路发现模块、多粒度路由计算模块、聚集信息管理模块、多粒度聚集调整模块和流表下发模块. 基础设施层则主要由 OpenFlow 交换机等网络设备组成. 两层之间通过信息交互来完成任务, 即交换机和控制器相互协作, 共同完成流量的多粒度路由和调度处理.

3.2 网络模型

SDN 的网络模型可以表示为加权无向图 $G = (V, E)$. 其中 V 表示具有存储、转发和处理能力的交换机节点集合; E 表示连接两节点的链路集合. 建立如图 2 所示网络模型, 下面将依次介绍网络模型中的节点模型、链路模型和聚集流量模型.

3.2.1 节点模型

互联网骨干网节点分为核心节点和汇聚节点, 将节点建模为一个三元组 Node(id, ip_seg, type), 其中, id 能唯一标识节点, ip_seg 表示节点的 IP 地址段, type 代表节点类型, 值为 0 表示核心节点, 为 1

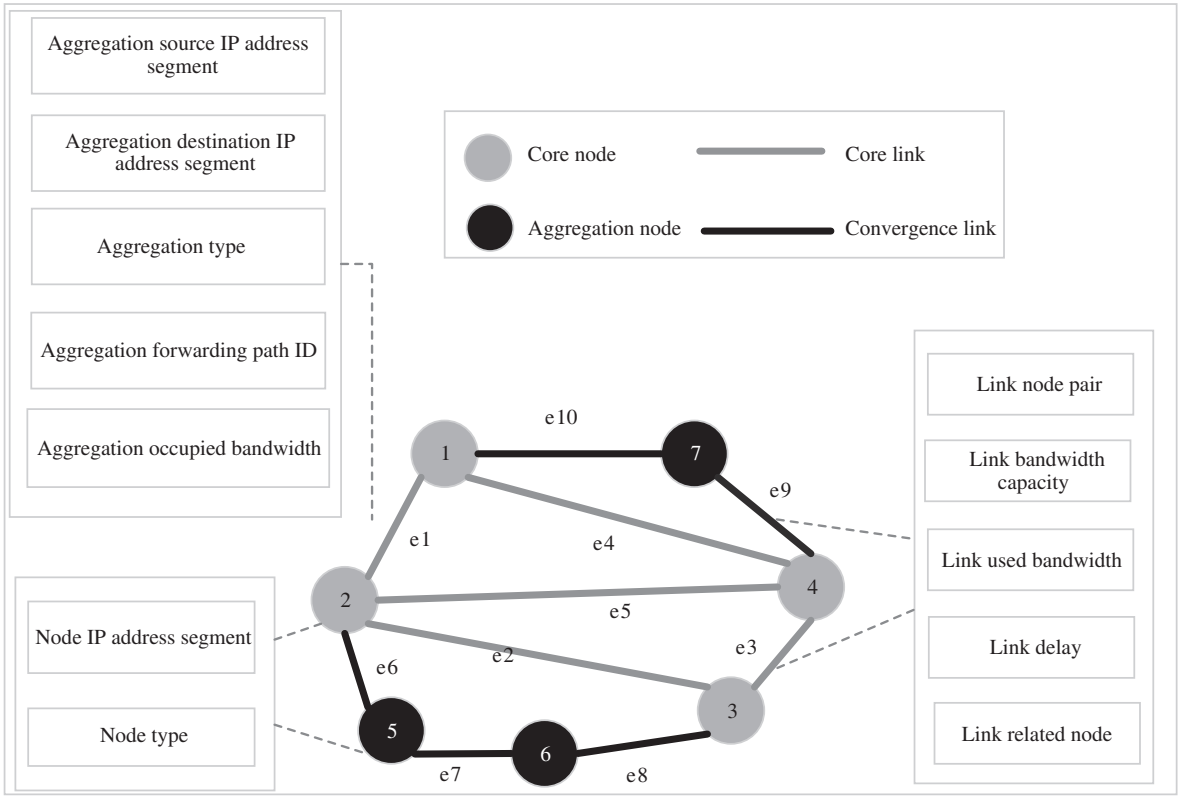

图 2　网络模型

Figure 2　The network model

表示汇聚节点.

3.2.2　链路模型

互联网骨干网中的链路分为核心链路与汇聚链路, 本文用七元组 link(id, node1, node2, total_bw, used_bw, delay, related_node) 表示每条链路. id 唯一标识一条链路, node1, node2 是链路连接的两个节点 id, total_bw 表示链路带宽总量, used_bw 表示链路已用带宽, delay 表示链路的时延, related_node 是链路的相关节点 id 集合, 可用于区分链路类型. related_node $\in \{\{0\} \cup \text{CNodes}\}$, 其中 CNodes 为汇聚节点的 id 集合. related_node 为 0 表示该链路为核心链路, 其相关节点是核心节点. related_node 属于 CNodes, 表示该链路为汇聚链路.

3.2.3　聚集流量模型

互联网骨干网网络模型中的聚集流量模型表示为五元组 aggregation (sip_seg, dip_seg, type, PID, bw). sip_seg 表示源 IP 地址段, dip_seg 表示目的 IP 地址段, type 是聚集所属流量类型, PID 是聚集所在路径的 id, bw 表示聚集所占带宽.

3.3　聚集相关设计

面对互联网海量流量的处理, 本文提出的机制以聚集流为计算单位, 在减轻控制器计算压力的同时, 能够缓解交换机的存储压力.

表 1 聚集的三元组信息

Table 1 The triples information of aggregation

sip_seg	dip_seg	type
Source IP address segment	Destination IP address segment	Traffic type

3.3.1 聚集的划分

如表 1 所示, 本文使用源、目的 IP 地址段和聚集所属流量类型定义聚集流的元信息. 这种划分使控制器能够以聚集流为单位进行路径计算和流量调度.

3.3.2 聚集之间的关系

聚集的源、目的 IP 地址段和传输路径 PID 可以用来确定两个聚集之间的关系.

(1) 最大聚集块. 当前的源/目的节点对间最粗粒度的聚集流为最大聚集块.

(2) 邻接聚集. 当两个聚集同时满足流量类型相同, 源/目的 IP 地址段相等, 源/目的 IP 地址段的前缀相同, 且能够用前缀更小的 IP 地址段表示时, 这两个聚集互为邻接聚集, 邻接聚集分同路邻接聚集和异路邻接聚集, 若其所在路径 id(PID) 相同, 则为同路邻接聚集, 不同则为异路邻接聚集.

(3) 对角聚集. 若聚集 B 是聚集 A 的两个邻接聚集的另一个共同邻接聚集, 则聚集 A 与聚集 B 互为对角聚集. 对角聚集又可以根据 PID 是否相同划分为同路对角聚集和异路对角聚集.

3.4 流量多粒度路由机制

流量多粒度路由机制的流程图如图 3 所示, 为解决控制层和数据层之间通信频繁和流表数目爆炸问题, 该机制基于流量的服务级别类型, 提供尽力而为 (best-effffort, BE) 类型的离线粗粒度和带宽预留 (bandwidth reservation, BR) 类型的在线细粒度的两阶段路由方式. 在离线阶段, 预计算出网络拓扑的全局路径, 并预下发 BE 类型流量的粗粒度聚集流表项. BE 类型流按照预下发路径汇聚转发. 而 BR 类型的新流在在线阶段, 触发交换机发送 Packet-in 消息, 并由控制器为其计算路径并下发细粒度流表项. 流量多粒度路由算法的伪代码如算法 1 所示.

3.4.1 全局路径预计算

全局路径预计算在离线阶段根据已知的全局视图, 计算拓扑图内全局路径集合. 全局路径预计算的具体步骤如下所述.

(1) 选择任意两节点 S 和 D, 借鉴深度优先搜索算法的思想 [16], 计算两节点间所有可行路径集合, 并将计算过程中得到的两节点与任一中间节点 M 的可选路径集作为 SM 节点对和 MD 节点对间的可选路径结果集保存起来.

(2) 在未计算的节点组合集合中重新选择两节点 S' 和 D', 从开始节点 S' 的邻接节点集 A 开始, 依次进行深度搜索.

(3) 若在搜索过程中发现当前中间节点 M 到目的节点 D' 的可选路径集合 P_{md} 已经存在, 则停止当前搜索, 并依次拼接节点 S' 到节点 M 的路径 P_{sm} 和 P_{md} 集合中的所有路径, 判断拼接后的路径 P 是否存在环路, 以及是否满足跳数限制, 将不存在环路的 K 跳以内的路径存到 $S'D'$ 节点对的可选路径结果集中, 最终得到包含当前分支节点 M 的路径集合.

(4) 若当前中间节点 M 到节点 D' 没有已知结果, 则同步骤 (1) 相同, 搜索 $M \to D'$ 的可行路径并将计算过程中得到的 $S'D'$ 两节点与中间节点 M 的可选路径集作为 $S'M$ 节点对和 MD' 节点对间

算法 1 流量多粒度路由算法

输入: V: 拓扑图内节点集合; E: 拓扑图内链路集合; $G(V,E)$: 网络拓扑无向图; LPVTM: 链路潜在值矩阵; $\mathrm{BWP_{oc}}$: 带宽权重比例值; SD: 当前聚集源、目的节点对; bandwidth: 当前聚集的预留带宽需求; PathSet: 全局可选路径集合;

Begin

根据 V 创建拓扑内所有 SD 节点对集合 SDList;
vNUM ⟵ V.length();
for $(s,d) \in$ SDList **do**
  $\mathrm{NLPV}_{SD} \leftarrow$ LPVTM [sd] [] ; //获取当前节点对相关的 LPV 数组
  weightTM ← getWeightTM (s, d, $\mathrm{BWP_{oc}}$, NLPV_{SD}) ; //获取当前节点对的链路权重矩阵
  visited[vNUM] , dist[vNUM][vNUM] ; //从 S 节点开始, 设置 visited[s] 和 dist[s][] 数组
  visted[s] ← true ;
  for $i = 0$ to vNUM **do**
    dist[i] ← weightTM[s][i] ;
  end for
  for $i(i! = s) \in V$ **do**
    使用 Dijkstra 算法为当前 SD 节点对的 BE 聚集选择权重最小的转发路径;
  end for
  Path[d][s] ← Path[s][d].reverse; //更新 SD 节点对对应的 DS 节点对路径信息
  BR 类型聚集;
  获取当前 SD 节点对的链路潜在值数组 NLPV, 选路径集 PathSet, 链路权重矩阵 weightTM;
  $\mathrm{min\_weight_{path}} \leftarrow$ MAX_ALUE;
  for path $\in \mathrm{PathSet}_{SD}$ **do**
    获取路径所含链路的剩余可用带宽和 $\mathrm{residual_{bandwidth_{path}}}$, 路径时延和 $\mathrm{delay_{path}}$;
    if $\mathrm{residual_{bandwidth_{path}}} <$ bandwidth or $\mathrm{delay_{path}} >$ delay **then**
      BREAK; //不考虑剩余可用带宽和时延不满足要求的路径
    end if
    $\mathrm{Weight_{path}} \leftarrow 0$;
    计算 PathSetSD 中每条路径的链路权重和, 选择 $\mathrm{Weight_{path}}$ 最小的路径作为当前 BR 聚集的转发路径 Path;
  end for
end for

End

输出: Path1: BE 类型聚集路径集合; Path2: BR 类型聚集的转发路径.

的可选路径结果集保存起来.

(5) 跳转至步骤 (2), 依次执行, 直至计算完拓扑中任意两节点路径. 最终得到全局可行路径集.

3.4.2 离线粗粒度路由

离线阶段, 根据全局路径预计算的结果构建链路潜在值矩阵和链路权重矩阵, 预计算 BE 类型聚集的初始转发路径, 预下发粗粒度流表项. 关键性链路指更容易被包含在备选路径集中的链路. BE 流量应选择包含更多非关键性链路的路径, 避免其占用关键性链路而对未来网络需求造成影响; BR 流量则选择包含更多关键性链路的路径. 为对链路关键性进行度量, 引入了链路潜在值的概念. 下面介绍链路潜在值 (link potential value, LPV) 的计算方式和 LPV 矩阵的构建方式.

(1) 计算链路潜在值数组. 借鉴 MIRA [17] 和 DORA [18] 对链路关键性的量化方式, 本文提出的链路潜在值的计算流程如下:

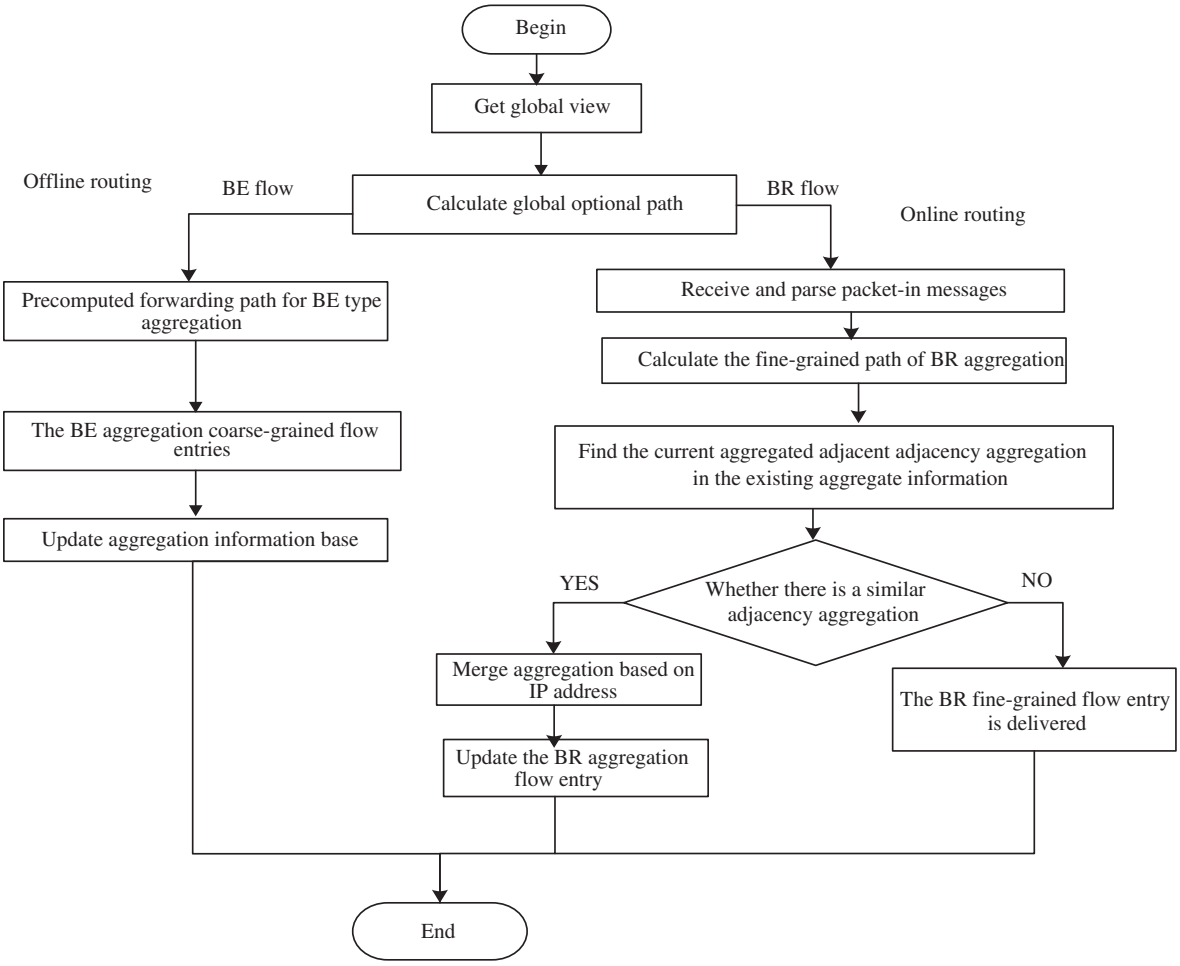

图 3 SDN 流量多粒度路由机制流程图

Figure 3 The flow chart of SDN traffic multi-granularity routing mechanism

(i) 将所有 SD 节点对对应的 LPV 数组值设为零.

(ii) 获取相应的 SD 节点对的可相交可选路径集. 链路每出现一次其对应的 LPV 值将减 1.

(iii) 检查每个链路在其他 SD 节点对的可相交可选路径集中出现次数. 每出现一次其对应的 LPV 值加 1.

(iv) 最后将该 SD 节点对的 LPV 数组归一化到 $0 \sim 100$ 的范围内, 对第 i 个 SD 节点对所对应的第 j 条链路的链路潜在值 LPV_{ij} 归一化过程如下:

$$\mathrm{LPV}_{ij} = 100 + \frac{100(\mathrm{init\_LPV}_{ij}^{-1} - \max(\mathrm{NLPV}_i^{-1}))}{\max(\mathrm{NLPV}_i^{-1}) - \min(\mathrm{NLPV}_i^{-1})}, \tag{1}$$

其中 $\mathrm{init\_LPV}_{ij}$ 表示原始的链路潜在值取值; NLPV_i 表示第 i 个 SD 节点对所对应链路潜在值数组.

(2) 构建链路潜在值矩阵. 使用 LPV 数组的集合构建链路潜在值矩阵:

$$\mathrm{LPVTM}=\begin{bmatrix}\mathrm{LPV}_{11} & \cdots & \mathrm{LPV}_{1j} & \cdots & \mathrm{LPV}_{1n}\\ \vdots & \ddots & & & \vdots\\ \mathrm{LPV}_{i1} & & \mathrm{LPV}_{ij} & & \mathrm{LPV}_{in}\\ \vdots & & & \ddots & \vdots\\ \mathrm{LPV}_{m1} & \cdots & \mathrm{LPV}_{mj} & \cdots & \mathrm{LPV}_{mn}\end{bmatrix}, \tag{2}$$

其中 n 表示拓扑中链路数, m 为 SD 节点对的数目, 计算式为

$$m=\frac{2(v-2)!}{v!}, \tag{3}$$

其中 v 表示拓扑中节点数.

链路权重由每条链路的潜在关键值和剩余带宽组合形成. 剩余带宽的影响由参数带宽权重比 (band-width proportion, BWP) 控制, 其取值范围在 0.0 ∼ 1.0 之间. 网络中第 i 个 SD 节点对对应的第 j 条链路的权重计算式为

$$\mathrm{weight}_{ij}=\begin{cases}(1-\mathrm{BWP})\times\mathrm{LPV}_{ij}+\mathrm{BWP}\times\mathrm{nBW}_j, & \text{if } \mathrm{nBW}_j<100,\\ 200, & \text{if } \mathrm{nBW}_j=100,\end{cases} \tag{4}$$

其中 nBW_j 表示归一化后的剩余带宽, 其计算式为

$$\mathrm{nBW}_{ij}=\begin{cases}99+\dfrac{99(\mathrm{resBW}_{ij}^{-1}-\max(\mathrm{RBW}^{-1}))}{\max(\mathrm{RBW}^{-1})-\min(\mathrm{RBW}^{-1})}, & \text{if } \mathrm{resBW}_{ij}>0,\\ 100, & \text{if } \mathrm{resBW}_{ij}=0,\end{cases} \tag{5}$$

其中 RBW 表示网络中链路剩余带宽值的集合, resBW_{ij} 表示链路剩余带宽, 是链路总带宽和已用带宽差值. 由式 (4) 可以得到单个节点对对应的所有链路权重的取值, 并可构建该节点对的链路权重矩阵:

$$\mathrm{weightTM}=\begin{bmatrix}\mathrm{weight}_{11} & \cdots & \mathrm{weight}_{1j} & \cdots & \mathrm{weight}_{1n}\\ \vdots & \ddots & & & \vdots\\ \mathrm{weight}_{i1} & & \mathrm{weight}_{ij} & & \mathrm{weight}_{in}\\ \vdots & & & \ddots & \vdots\\ \mathrm{weight}_{n1} & \cdots & \mathrm{weight}_{nj} & \cdots & \mathrm{weight}_{nn}\end{bmatrix}, \tag{6}$$

其中 weight_{ij} 表示对于该节点 edge_{ij} 的链路权重.

由式 (4) 可知, $\mathrm{nBW}_{ij}=100$ 时, $\mathrm{weight}_{ij}=200$, 表示链路不可用.

3.4.3 在线细粒度路由

在线阶段, 为 BR 类型聚集实时计算细粒度转发路径, 并调用聚集粒度调整算法尝试在当前聚集信息列表中寻找可合并同路邻接聚集, 若存在, 则合并聚集, 并更新流表项到相关交换机. 否则下发细粒度流表到路径相关交换机, 并将所得结果更新到聚集信息数据库中. 聚集的合并是通过增大聚集的

IP 地址掩码范围将多个细粒度聚集流合并为一条新的粗粒度聚集流, 从而减少流表项数目. 合并聚集的具体步骤如下:

(1) 在聚集集合 A 中获取当前聚集 aggre 的同路邻接聚集 adAggre$i$, 如果 i 为 0, 结束算法; 如果 i 为 1, 执行步骤 (2); 如果 i 为 2, 执行步骤 (3).

(2) 将 aggre 和 adAggre1 合并为新聚集 newAggre, 根据粒度调整触发条件, 判断 newAggre 能否保留. 结果为真则执行步骤 (7); 否则结束算法.

(3) 获取 aggre 的同路对角聚集 diAggre, 存在, 则执行步骤 (4); 否则将 aggre 与占用带宽更小的邻接聚集合并, 得到新聚集 newAggre, 根据粒度调整触发条件, 判断 newAggre 能否保留. 结果为真则执行步骤 (7); 否则结束算法.

(4) 合并 aggre, adAggre1, adAggre2 和 diAggre, 得到新聚集 newAggre, 并根据粒度调整触发条件判断能否保留. 结果为真, 则执行步骤 (7); 否则执行步骤 (5).

(5) 将 4 个聚集保存到集合 aggreSet 中, 并将其同路邻接聚集两两合并, 并判断是否符合保留条件, 若符合条件, 则保留合并后的聚集; 否则仍保留合并前的原聚集.

(6) 在 aggreSet 中选择聚集数目最少、聚集带宽标准最小的集合作为最终合并方案.

(7) 更新聚集集合信息.

(8) 根据全局预先计算好的路由信息, 为聚集流寻找最优的转发路径.

在线细粒度路由通过对细粒度流量的汇聚实现数据的高性能转发. 尽管如此, 在线路由和离线路由之间并不会相互冲突. 因此, 在基于全局路径信息已经计算的前提下, 本文采用混合粒度的路由算法来选择转发路径. 通过判断当前流量的特征 (属于 BE 或者 BR) 来对流量进行分类, 将同类型流量聚集为适当粒度, 再交付给在线路由或者离线路由模块进行处理, 从而实现混合粒度的路由算法.

3.5 流量多粒度调度机制

3.5.1 粒度调整触发条件

为解决聚集粒度过大, 网络拥塞和负载不均衡等问题, 调度机制首先对流量粒度进行调整, 然后根据调度策略将调整后的聚集调度到其他可选路径. 在粒度调整过程中, 需要判别哪些聚集可以被聚合, 哪些聚集应当被拆分. 粒度调整触发的伪代码见算法 2 的 $1 \sim 16$ 行.

3.5.2 聚集粒度动态调整

利用粒度调整触发条件和聚集间关系设计了聚集粒度的合并和拆分, 分别实现对聚集粒度的加大和减小. 下面分别介绍聚集粒度合并和拆分的思想和具体流程.

- 聚集粒度合并: 聚集合并通过增大聚集的 IP 地址掩码范围将多个细粒度聚集流合并为一条新的粗粒度聚集流, 从而减少流表项数目. 合并过程参照 3.4.3 小节中的合并过程.

- 聚集粒度拆分: 聚集拆分用来减小聚集粒度, 将一个聚集拆分为多个聚集并把交换机内的一个流表项拆分成多个流表项, 但不会改变转发路径. 根据粒度调整触发条件, 获取待拆分聚集 A, 待拆分份数下限 n 计算如下:

$$n = \left\lceil \frac{\text{bandwidth}}{\max(\text{resBW}_i)} \right\rceil, \tag{7}$$

其中 bandwidth 表示待拆分聚集的流量需求; resBW_i 表示当前聚集 A 的第 i 条备选链路的剩余可用带宽. 聚集拆分的具体过程如下:

算法 2 流量多粒度调度算法

输入: agree: 当前有待判断的聚集; α: 判断聚集粒度阈值的参数; BW$\{\text{bw}_1, \text{bw}_2, \text{bw}_3\}$: 聚集的历史带宽信息集合, 集合内元素下标越小表示数据越新; Path: 当前聚集 Aggre 所在路径.

Begin

$E \leftarrow$ 路径 Path 所含链路集合;
$\text{totalBW}_{\min} \leftarrow \min(E.\text{total\_bandwidth})$;
$X \leftarrow \text{totalBW}_{\min} \times \alpha$;
if $\text{bw}_1 > \text{bw}_2 > \text{bw}_3$ and $\text{bw}_1 < X$ **then**
  $T \leftarrow$ true;
else
  if !($\text{bw}_1 > \text{bw}_2 > \text{bw}_3$) and (集合 BW 中至少两个值小于 X) **then**
    $T \leftarrow$ true;
  else
    $T \leftarrow$ false;
  end if
end if
T: 判断当前聚集能否保留;
if 控制器检测到单一聚集粒度过大 **then**
  调用聚集拆分算法对粗粒度进行拆分, 对拆分的聚集进行调度;
end if
if 控制器检测到链路拥塞 **then**
  对拥塞链路的聚集进行调整, 对调整后的聚集进行调度;
end if

End

输出: Agree: 待调度聚集集合.

(1) 获取待拆分聚集 aggre, 并计算 n.

(2) 拆分聚集 aggre, 获取新聚集集合 newAggresip 和 newAggredip, 将聚集占用带宽标准差更小的集合作为拆分结果集合 newAggreSet. 判断 newAggreSet 内的聚集是否符合保留条件, 将符合保留条件的聚集添加到结果集合 A' 中, 不符合的添加至待拆分聚集队列 SplitQueue.

(3) 若 SplitQueue 非空或拆分后新聚集集合 A' 内聚集数量 m 比 n 小, 则执行 (4); 否则执行 (5).

(4) 如果 SplitQueue 非空, 则在 SplitQueue 中获取并删除待拆分聚集 aggre; 否则在集合 A' 中获取带宽需求最大的聚集作为待拆分聚集 aggre, 回到步骤 (2).

(5) 尝试重新合并拆分后的聚集, 直至 m 小于等于 n 或集合 A' 内不存在可合并聚集.

3.5.3 流量多粒度调度方案

本文设计的 SDN 流量多粒度调度包含粒度过粗聚集调度和链路拥塞聚集调度两种场景, 采用多粒度处理思想, 建立 0-1 整数线性规划 (integer linear programming, ILP) 模型, 基于不同的触发条件, 提出不同优化目标. 所建立 0-1 ILP 数学模型涉及的参数如表 2 所示, 流量多粒度调度算法的伪代码如算法 2 所示. 使用 MATLAB R2012b 开发, 利用其建立数学模型对 0-1 ILP 问题求解. 下面依次介绍两种触发条件下的调度策略.

(1) 粒度过粗聚集调度. 控制器周期性检测聚集占用带宽是否符合保留条件, 对不符合条件的聚集进行拆分, 并对拆分后的子聚集集合实施以最小化核心链路, 最大链路利用率为优化目标的调度策略. 优化目标表达式为

$$\text{minimize } \max(\text{UR}_I), \quad I \in \text{LC}, \tag{8}$$

表 2 0-1 整数线性规划模型参数描述

Table 2 The description of parameters in 0-1 ILP model

Parameter	Parameter description
A	Aggregations to be scheduled, a aggregation of a collection is represented by a. A_l represents the aggregations of A flowing through link l.
BR	The BR aggregations to be scheduled.
bw_a, delay_a, P_a	The bandwidth requirement, the upper limit of the delay requirement, the available paths of the a-th aggregation to be scheduled. P_{ai} represents the i-th alternative path available for aggregation a.
L	Set of links included in the collection P, where a single link is represented by l, $l \in L$. l_{aj} denotes the j-th link in the link set of the a-th aggregation alternative path.
LC	Core links contained in collection P, where a single link is represented by l_c, $l_c \in \mathrm{LC}$. LC_a represents the core link set of the a-th aggregation alternative path.
n_{lc}	The number of links included in the set LC.
$C_{lp_{ai}}$	Whether the link P_{ai} contains link l. If it contains, the value is 1, otherwise the value is 0.
T	The total amount of bandwidth of the link, and T_l represents the total amount of bandwidth of link 1.
U	The link has used bandwidth, and Ul indicates the used bandwidth of the link l.
D	Link transmission delay, Dl is the link l transmission delay.
σ	The parameter that determines the congestion of the link, the upper limit of the required bandwidth of the link must not exceed the total bandwidth of the bandwidth. The range of value is (0, 1).
$X_{P_{ai}}$	The result of the aggregation rerouting. If the aggregation a is assigned to the i-th path in the alternative path set, the value is l, otherwise the value is 0.
Ip_{ai}	Whether the i-th feasible path of the aggregation a is its original path, and if so, the value is 1, otherwise the value is 0.
N_l, UR_{l_c}	The bandwidth load of link l and the bandwidth resource utilization of core link l_c after aggregation scheduling.

优化目标的约束公式为

$$U_I - \sum_{a=1}^{A_I} \mathrm{bw}_a + \sum_{a=1}^{A_I} \left(\sum_{i=1}^{L} X_{P_{ai}} \times C_{Ip_{ai}} \right) \times \mathrm{bw}_a = N_I, \quad I \in L, \tag{9}$$

$$N_I < T_I \times \sigma, \tag{10}$$

$$\mathrm{UR}_l = \frac{N_l}{T_l}, \quad l \in \mathrm{LC}, \tag{11}$$

$$\sum_{i=1}^{P_a} X_{P_{ai}} = 1, \quad a \in A_I, \tag{12}$$

$$\sum_{a}^{\mathrm{BR}} \sum_{i}^{P_a} Ip_{ai} \times X_{P_{ai}} \times C_{Ip_{ai}} \times D_{I_{aj}} < \mathrm{delay}_a, \quad a \in \mathrm{BR}, \tag{13}$$

$$X_{P_{ai}} = 0, 1, \tag{14}$$

其中式 (9) 表示每条链路经过聚集重路由选择后的带宽负载量; 式 (10) 表示每条链路使用带宽的上限约束; 式 (11) 表示核心链路利用率的计算公式; 式 (12) 表示每个聚集的转发路径唯一性约束; 式 (13) 表示当调度聚集为 BR 类型时, 路径需要满足的 BR 聚集时延需求约束; 式 (14) 表示模型中唯一自变量 $X_{P_{ai}}$ 是一个取值为 1 或 0 的整数.

(2) 拥塞链路内聚集调度. 当控制器检测到链路拥塞时, 需要对超载链路上的聚集集合进行粒度调整, 并尝试将调整后的新聚集集合迁移至负载较轻的其他可行链路上. 算法以最小化重路由数为优化目标, 以减少流表项的更改数目. 优化目标表达式为

$$\text{minimize} \sum_{a}^{A}\sum_{i}^{P_a} I_{P_{ai}} \times X_{P_{ai}}, \tag{15}$$

其中 A 表示调整聚集粒度后的待调度聚集集合. 该模型同样以式 (9), (10), (12), (13) 作为优化目标的约束.

4 实验评估

本文提出的流量多粒度处理机制的不同模块内算法使用不同开发工具仿真实现: 多粒度路由算法采用 JAVA 语言, 在 Eclipse 平台上实现. 多粒度聚集调度模块使用 MATLAB R2012b 开发, 使用其建立数学模型对 0-1 ILP 问题求解, 并将算法 matlab 代码封装成 jar 包, 与多粒度路由算法一样在 Eclipse 内使用 JAVA 语言调用. 使用 MySQL 数据库对数据进行管理和操作.

4.1 实验设置

• 拓扑用例. 采用中国教育和科研计算机网 CERNET 的骨干网拓扑结构. 含 36 个节点, 8 个核心节点, 28 个汇聚节点; 49 条链路, 5 条核心链路, 44 条汇聚链路; 其最大节点度为 11, 平均节点度为 2.472.

• 参数确定. 基于 IP 网络中的 QoS 分类标准, 将所有 BR 类型流量请求分为 6 种类型. 在仿真实验中, 只考虑时延和带宽约束, 生成的 BR 业务流请求随机选取一种类型作为它的 QoS 参数. 6 种 BR 请求及对应的时延 (ms)/带宽 (Mbps) 分别为 1 : 80/100; 2 : 50/400; 3 : 30/300; 4 : 20/400; 5 : 15/1000; 6 : 10/1000.

• 基准机制. 将经典的 ECMP [19], GFF [20] 和 LABERIO [21] 算法与本文提出的流量多粒度处理机制 (multi-granularity processing, MGP) 进行对比分析. 为了公平起见, 本文对这 3 种经典算法进行了改进, 为其添加了全局路由与计算和粒度调整机制, 将改进后的经典算法作为系统性能评价的对比基准.

4.2 实验结果

仿真实验基于 CERNET 骨干网拓扑结构, 将本文机制与基准机制进行对比分析, 从平均路由跳数、网络容纳总带宽、链路利用率标准差、流表项个数、层间通信次数, 多个性能指标上来评价路由选择和网络性能.

(1) 平均路由跳数. 平均路由跳数表示路由成功的路径的跳数总和与成功路由的请求个数的比值. 平均路由跳数越低, 则说明路由算法越高效. 如图 4, 在网络负载较轻阶段, LABERIO 的平均路由跳数最小, 随着请求数增加, 该值也不断增大, 这是因为 LABERIO 机制在路由阶段采用剩余带宽优化策略, 但由于其调度策略的有效性, 其值始终小于另两种处理机制. ECMP 在可选路径上均匀分配请求, 而 GFF 在其基础上增加了大象流调度处理过程, 而随着网络负载加重, 缺乏流量调度的 ECMP 网络内链路拥塞现象严重, 因此其平均路由跳数增幅较大并最终超过 GFF 处理机制. MGP 处理机制在负载较轻阶段, 其值最大, 但随着请求个数增多, 逐渐达到同时段的最小值. 这是因为 MGP 在选择初始

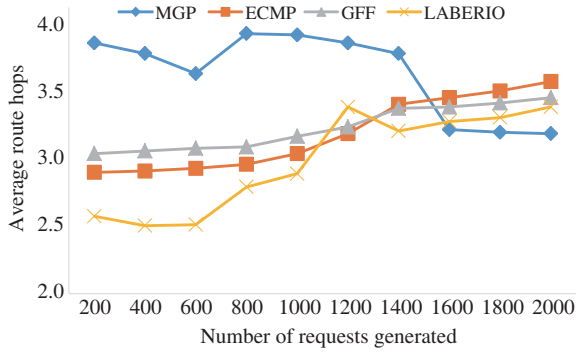

图 4 (网络版彩图) 平均路由跳数变化情况

Figure 4 (Color online) The results of average hops

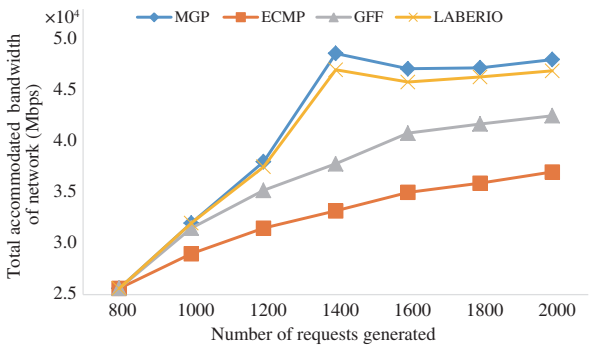

图 5 (网络版彩图) 网络容纳总带宽变化情况

Figure 5 (Color online) The results of total used bandwidth in network

路径时考虑了链路潜在值, 随着聚集粒度加大, 该机制的调度算法调整聚集粒度后, 根据流量类型将流量向跳数更少的路径迁移.

(2) 网络容纳总带宽. 网络容纳总带宽表示某时刻网络拓扑中所有链路已用带宽的总和. 网络容纳带宽总量越大, 表明路由算法性能越好. 如图 5 所示, 4 种机制的网络容纳总带宽整体都呈现上升趋势, ECMP 流量处理机制始终小于其他 3 种处理机制. 这是因为该机制在选择路径时, 没有考虑网络拓扑情况, 且缺乏流量调度. GFF 处理机制的网络容纳总带宽高于 ECMP 机制, 但始终小于另外两种机制. 其原因是该机制考虑了大象流的调度问题, 但缺乏全局视角. LABERIO 处理机制的网络容纳总带宽仅小于 MGP 处理机制, 这是因为该机制采用局部链路调整策略, 且始终维持负载均衡. MGP 处理机制的网络容纳总带宽始终高于另外 3 种处理机制, 但其增幅却逐渐减少. 这是因为随着网络可用带宽减少, 无法为数据流找到可行路径的情况逐渐增加.

(3) 链路利用率标准差. 链路利用率标准差反映了网络当前的链路负载均衡情况, 如图 6 所示. LABERIO 处理机制的链路利用率标准差呈上升趋势且同时刻处于最低水平. 这是因为该机制以维持负载均衡为优化目标. MGP 处理机制在网络负载较轻阶段数值最大, 在负载较重时段数值仅大于 LABERIO 处理机制. 产生这种变化的原因是该机制倾向于选择权重最小的路径为初始路径. 随着网络负载的增加, 流量调度机制不断地调整聚集粒度和调度流量解决链路拥塞和聚集过大问题. GFF 处理机制优于 ECMP, 但小于另外两种处理机制. 这是因为 ECMP 处理机制的大象流冲撞问题导致其网络负载失衡, 而 GFF 处理机制在一定程度上解决了大象流冲撞问题, 但网络负载较重阶段, 由于其缺乏全局视角, 因而其数值不断增加.

(4) 流表项个数. 流表项个数是数据层中某时刻活跃的流表项个数总和, 如图 7 所示, 4 种处理机制都呈现不同程度的上升趋势. MGP 处理机制的流表项个数始终远小于另外 3 种处理机制. 这是因为该机制采用多粒度处理方式, 通过聚合流量, 减少生成的流表项数目. 且对于控制器而言, 流表项拆分带来的计算压力不足以影响流表项数目减少所减轻的计算压力. 从最终效果来看, MGP 机制可以有效地减少流表项数目. 其余 3 种基准机制的数值呈现 LABERIO 处理机制大于 GFF 处理机制, 大于 ECMP 处理机制的效果. 这是由于在前两种机制内都增加了粒度调整机制以减少流表项数目, 且 LABERIO 处理机制的调度策略粒度更细, 会产生额外的流表项, 但 ECMP 处理机制不存在后续调度算法.

(5) 层间通信次数. 层间通信次数指控制层与数据层间通信次数总和, 如图 8 所示, 在负载较轻阶

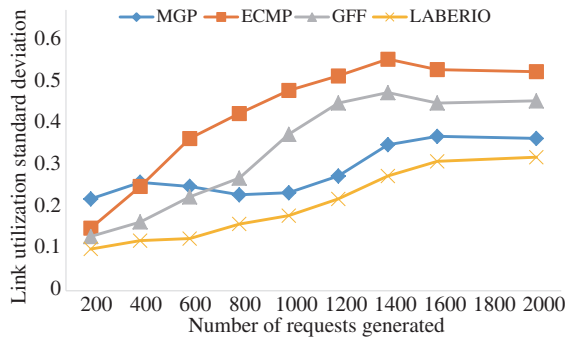

图 6 (网络版彩图) 链路利用率标准差变化情况

Figure 6 (Color online) The results of link utilization standard deviation

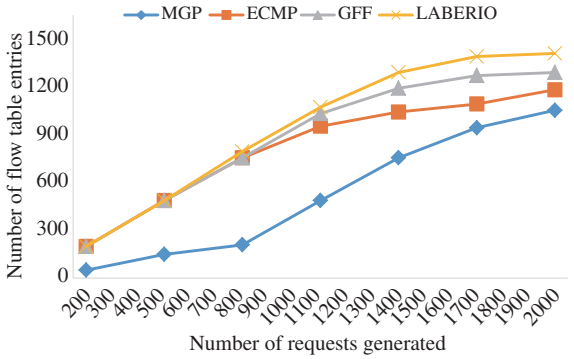

图 7 (网络版彩图) 流表项个数变化情况

Figure 7 (Color online) The results of the number of flow entries

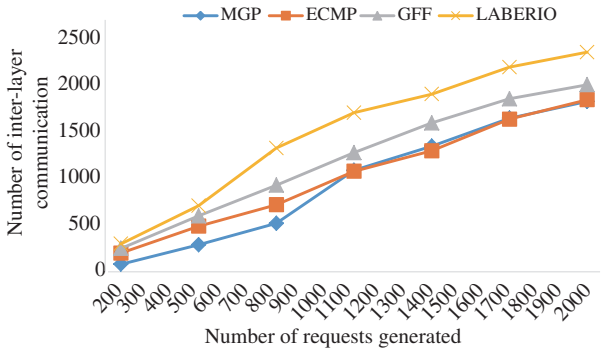

图 8 (网络版彩图) 层间通信次数变化情况

Figure 8 (Color online) The results of the number of inter-layer communication

段, MGP 处理机制的数值最小, 随着网络负载增加, 其数值也随之增加且处于次优水平, 但最终 MGP 处理机制的值略小于 ECMP 处理机制. 这是由于 MGP 处理机制在离线阶段预下发 BE 聚集粗粒度流表项, 所以在在线阶段, MGP 处理机制的新流路由请求通信次数始终小于其他 3 种机制. 随着网络负载增加, 流量调度工作也随之增加, MGP 处理机制在拥有流量调度功能的处理机制中, 表现最优, 这是因为该机制从全局角度出发, 以减少流表项数目为优化目标, 同时尽可能减少层间通信次数. ECMP 处理机制层间通信次数最少, 这是由于其缺乏流量调度机制. GFF 处理机制只针对大象流实施调度, 其层间通信次数始终少于 LABERIO 机制, LABERIO 处理机制维持负载均衡需要大量的流量调度和流表项更改操作.

5 总结

针对互联网骨干网中负载过度增加导致网络性能下降的问题, 本文提出了面向互联网的 SDN 流量多粒度处理机制, 以聚集为计算单位, 分别提出了多粒度的路由和调度两种算法. 其中, 多粒度路由算法包括面向尽力而为的离线粗粒度路由和面向带宽延迟敏感的在线细粒度路由. 多粒度调度则通过动态合并与拆分汇聚的流量集合来实现不同粒度的切换. 本文以此为基础进行了仿真实验. 研究结果表明本文提出的多粒度处理机制在平均路由跳数、网络带宽和链路利用率等方面均有较好的性能, 从

而验证本文提出算法的有效性和可行性. 本文设计的流量处理机制涉及路由计算和流量调度两个方面, 在进一步的研究工作中, 考虑从流量检测、故障恢复等方面完善本机制, 以达到更好的效果.

参考文献

1 Kreutz D, Ramos F M V, Verissimo P E, et al. Software-defined networking: a comprehensive survey. Proc IEEE, 2015, 103: 14–76

2 Yan C J, Wu D J, Xiong Y. SDN Principle Analysis: Transfer Control Separation SDN Architecture. Beijing: People's Posts and Telecommunications Press, 2016 [闫长江, 吴东君, 熊怡. SDN 原理解析: 转控分离 SDN 架构. 北京: 人民邮电出版社, 2016]

3 Meyer D. The software-defined-networking research group. IEEE Internet Comput, 2013, 17: 84–87

4 Singh S, Jha R K. A survey on software defined networking: architecture for next generation network. J Netw Syst Manag, 2017, 25: 321–374

5 Xu S, Wang X, Gao B, et al. Controller placement in software-defined satellite networks. In: Proceeedings of 2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), 2018. 146–151

6 Zhang B, Wang X, Huang M. Dynamic controller assignment problem in software-defined networks. Trans Emerging Tel Tech, 2018, 29: e3460

7 Zhang B, Wang X, Huang M. Multi-objective optimization controller placement problem in Internet-oriented software defined network. Comput Commun, 2018, 123: 24–35

8 Guo Z, Xu Y, Liu R, et al. Balancing flow table occupancy and link utilization in software-defined networks. Future Generation Comput Syst, 2018, 89: 213–223

9 Katta N, Alipourfard O, Rexford J, et al. Infinite cacheflow in software-defined networks. In: Proceedings of the 3rd Workshop on Hot Topics in Software Defined Networking, Chicago, 2014. 175–180

10 Shirali-Shahreza S, Ganjali Y. ReWiFlow: restricted wildcard openflow rules. SIGCOMM Comput Commun Rev, 2015, 45: 29–35

11 Duliński Z, Rzym G, ChoŕP. MPLS-based reduction of flow table entries in SDN switches supporting multipath transmission. 2018. ArXiv: 1805.07993

12 Jia X, Jiang Y, Guo Z, et al. Reducing and balancing flow table entries in software-defined networks. In: Proceedings of IEEE 41st Conference on Local Computer Networks (LCN), Dubai, 2016. 575–578

13 Zhang Q Y, Wang X W, Huang M, et al. Software defined networking meets information centric networking: a survey. IEEE Access, 2018, 6: 39547–39563

14 Xu S, Wang X, Huang M. Modular and deep QoE/QoS mapping for multimedia services over satellite networks. Int J Commun Syst, 2018, 31: e3793

15 Zhang B, Wang X, Huang M. Adaptive consistency strategy of multiple controllers in SDN. IEEE Access, 2018, 6: 78640–78649

16 Dijkstra E W. A note on two problems in connexion with graphs. Numer Math, 1959, 1: 269–271

17 Kodialam M, Lakshman T V. Minimum interference routing with applications to MPLS traffic engineering. In: Proceedings IEEE INFOCOM 2000, 2000. 2: 884–893

18 Riggio R, Rasheed T, Granelli F. EmPOWER: a testbed for network function virtualization research and experimentation. In: Proceedings of IEEE SDN for Future Networks and Services (SDN4FNS), Trento, 2013. 1–5

19 Dixit A, Prakash P, Hu Y C, et al. On the impact of packet spraying in data center networks. In: Proceedings of IEEE INFOCOM, Turin, 2013. 2130–2138

20 Al-Fares M, Radhakrishnan S, Raghavan B, et al. Hedera: dynamic flow scheduling for data center networks. In: Proceedings of Usenix Symposium on Networked Systems Design and Implementation, San Jose, 2010. 281–296

21 Long H, Shen Y, Guo M, et al. LABERIO: dynamic load-balanced routing in openflow-enabled networks. In: Proceedings of Advanced Information Networking and Applications, Barcelona, 2013. 290–297

Traffic multi-granularity processing mechanism for Internet-oriented SDN

Xiangmin LU[1], Xingwei WANG[1*], Bo YI[1], Jie LI[1] & Min HUANG[2]

1. *College of Computer Science and Engineering, Northeastern University, Shenyang* 110169*, China*;
2. *College of Information Science and Engineering, Northeastern University, Shenyang* 110819*, China*
* Corresponding author. E-mail: wangxw@mail.neu.edu.cn

Abstract As a new network architecture, software defined networking (SDN) separates the control and data forwarding planes and implements programmable control. SDN provides a new way to improve the overall performance of the Internet. Although SDN has global view capacity, it also suffers from performance bottlenecks when handling massive Internet data, i.e., frequent interlayer communication between the control and data forwarding planes of SDN reduces the computational efficiency of the controller, and massive flow table entry data incur very high storage costs on switches. To further improve SDN performance and make it adaptable to the massive traffic processing of the Internet, this paper proposes a traffic multi-granularity processing (MGP) mechanism for Internet-oriented SDN. The proposed MGP applies the SDN architecture to traffic processing of the Internet backbone network and implements a multi-granularity traffic processing mechanism in consideration of routing and scheduling. Simulation results demonstrate that the proposed MGP can reduce the amount of interlayer communication and flow table entries, maintain network load balancing, improve the correctness and effectiveness of route selection, improve SDN performance, and improve the processing capacity of mass data on the Internet.

Keywords software defined networking, Internet, multi-granularity routing, traffic scheduling, performance optimization

Xiangmin LU was born in 1994. She received her B.S. degree in computer science and technology from Zhengzhou University of Aeronautics, China in 2018. She is currently an M.S. candidate. Her research interests include multipath transmission and flow scheduling.

Xingwei WANG was born in 1968. He received his B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, Shenyang, China in 1989, 1992, and 1998, respectively. He is currently a professor at the College of Computer Science and Engineering, Northeastern University, Shenyang, China. His research interests include cloud computing and future Internet.

Bo YI was born in 1988. He received his B.S. and M.S. degrees in computer science from South-Central University for Nationalities, Wuhan, China in 2012 and 2015, respectively. He is currently working toward a Ph.D. degree at Northeastern University, Shenyang, China. His research interests include routing and service function chains in SDN and NFV.

Jie LI was born in 1982. She received her B.S. degree in computer science and technology from the Dalian University of Technology, Dalian, China in 2004. She received her M.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, China in 2007 and 2015, respectively. She is currently an associate professor at the College of Computer Science and Engineering, Northeastern University, Shenyang, China. Her research interests include mobile intelligent computing, mobile wireless networks, and industrial Internet.