文章编号:1007-5321(2020)01-0097-07 **DOI**:10.13190/j.jbupt.2019-076

一种面向软件定义网络的大流检测机制

邢长友[1], 李东阳[1], 谢升旭[1], 张国敏[1], 魏 伟[2]
(1.陆军工程大学 指挥控制工程学院, 南京 210001; 2.31106 部队, 南京 210016)

摘要: 为解决大流检测过程中普遍存在的精度低、耗费高等问题,提出了一种面向软件定义网络的大流检测机制 SampleFlow. 通过综合 sFlow 和 OpenFlow 技术优势,使用粗粒度的采样技术识别出疑似大流,在 OpenFlow 交换机上安装测量流表项,对这些疑似大流进行细粒度测量判别,以达到准确检测大流的目标,通过采样点优化选择算法,还可降低采样的冗余性. 实验结果表明,SampleFlow 能够有效降低测量负载,并提升大流检测的精度.

关 键 词: 软件定义网络; 网络测量; 大流检测; 采样
中图分类号: TP393 **文献标志码**: A

A Heavy Hitter Detection Mechanism in Software Defined Networks

XING Chang-you[1], LI Dong-yang[1], XIE Sheng-xu[1], ZHANG Guo-min[1], WEI Wei[2]
(1. Army Engineering University, Command and Control Engineering College, Nanjing 210001, China;
2. Corps 31106, Nanjing 210016, China)

Abstract: SampleFlow, a heavy hitter detection mechanism in software defined networks, is proposed to solve the problems of low detection accuracy and high measurement cost. By combining the technical advantage of sFlow and OpenFlow, SampleFlow firstly detects a set of suspicious heavy hitters by using the coarse-grained sFlow sampling method, and then installs measurement flow entries on specific OpenFlow switches to perform a fine-grained measurement on these suspicious heavy hitters, so as to determine the true heavy hitters. Besides, SampleFlow also uses a sampling position optimization method to decrease the sampling redundancy. Experiment results show that SampleFlow can decrease the measurement cost, and increase the heavy hitter detection accuracy effectively.

Key words: software defined network; network measurement; heavy hitter detection; sampling

软件定义网络技术(SDN, software defined networking)为流量工程、服务质量优化等提供了一种更好的解决途径. 在全网范围内对时变的动态流量进行准确和及时的测量,能够有效支持故障诊断、攻击检测、流量工程、负载均衡等. 然而,流量测量面临的一个主要挑战是网络流量巨大,对于当前的流量规模和链路速率,可能会同时存在数百万条并发的数据流,导致测量系统无法为这些流同时维护状态信息. NetFlow[1]、sFlow[2]等提出的采样测量方法存在误差较高等问题,Mai 等[3]的分析结果表明,采样方法所导致的偏差会明显影响网络攻击检测的准确性.

研究显示,网络中大量的流为小流,少数流为大流,并且绝大部分流量分布集中在这些少量的大流中[4]. 这里的大流是指在一个测量时间段内传输的报文或者字节数量超过一定数量的流. 因此,快速

收稿日期: 2019-05-11
基金项目: 国家自然科学基金项目(61379149, 61772271); 国家博士后科学基金项目(2017M610286)
作者简介: 邢长友(1982—), 男, 副教授, 硕士生导师, E-mail:changyouxing@126.com.
通信作者: 李东阳(1996—), 男, 硕士生, E-mail: dongyangli_nj@126.com.

测量以识别大流对于整个工作具有至关重要的作用. 针对 SDN 环境下的大流检测问题,ProgME 提出了一种可编程流量测量体系结构,能够针对用户定义的流量集合收集统计数据[5]. 为了支持多种流量测量任务,Elastic Sketch[6] 提出了一种弹性动态的 sketch 模型,并能够对其进行动态配置,以提升流量测量的灵活性. 为了避免对交换机架构的修改,Moshref[7] 和 Harrison[8] 等分别提出了运行于不同商业交换机上的流量测量方案,Moshref 等[9] 提出了一种在测量过程中优化使用 OpenFlow 交换机流表资源的方法. Xing 等[10] 初步讨论了将采样和流表测量相结合进行测量的可行性和如何优化测量流表项的问题,并未研究如何高效地进行流量采样的问题. 笔者提出一种 SDN 流量采样与流表测量相结合的两阶段大流检测机制,一方面,能够基于粗粒度的流量采样技术在源地址或目的地址等维度上以聚合的方式快速高效地寻找出疑似大流;另一方面,针对这些疑似大流,采用在 OpenFlow 交换机中加载测量流表的方式进行细粒度的监测,从而能够有效压缩 OpenFlow 交换机中的测量流表项,以较高的精度测量网络中的大流,并且不需要修改现有交换机的实现方法,方便在网络中进行部署.

1 两阶段大流检测机制

OpenFlow 为每条流提供了分组、字节、持续时间 3 种类型的统计计数器. 集中式控制器通过周期性地向各个交换机发送 Read-State 消息来获取这些计数器的最新值以及网络状态信息等. 理论上,通过在存储器中为每条流增加一个测量表项,网络管理员可以实现对每条流的大小、分组数等信息的统计,然而,一方面这种测量将极大地消耗宝贵的三态内容寻址存储器(TCAM, ternary content addressable memory)资源;另一方面,频繁的查询会对交换机和控制器带来极大的开销. 受 TCAM 资源的约束,为每条流建立一个测量流表项并不现实,需要设计相应的优化方案.

1.1 两阶段检测机制设计

采样是进行高速网络流量测量的一个通用解决方案,但采样率对测量精度会产生重要影响. 若采用读取 OpenFlow 交换机流表方式进行测量时,又会受制于 OpenFlow 交换机的资源限制. 例如,通常情况下,sFlow 检测系统每秒能够处理约 100 万条流,而 POX 控制器每秒仅能够处理约 3 万条流[11]. 因此若利用 sFlow 技术进行粗粒度的大流检测,随后再利用流表测量机制对检测结果进行细粒度处理,则可以实现两者优势互补,有效提升大流检测的精度. 基于此,笔者提出一种采样和流表读取相结合的两阶段动态大流检测机制 SampleFlow.

在 SampleFlow 中,为了准确地测量出网络内的大流,首先通过 sFlow 采样机制对流量进行采样,并将发现的疑似大流结果上报至 SDN 控制器;其次,控制器根据寻找出的疑似大流向相应的交换机下发测量流表项对疑似大流进行流测量,进而准确发现网络中的大流. 上述通过两阶段的采样和读流表操作,能够有效解决采样率影响测量精度、流表测量导致 TCAM 测量流表空间严重膨胀等问题.

由于网络中大流的数量只占整个流数量的很小一部分,能够通过 SampleFlow 第 1 阶段的疑似大流检测过程的流数目相对于原始流数目非常有限,有效降低了第 2 阶段需要在交换机加载的测量流表. 尽管采样会带来一定的误差,但通过适当降低判别阈值,允许一定的误判,而尽量减少漏判,进而在第 2 阶段的测量中实现准确的测量.

1.2 基于 sFlow 采样的疑似大流检测

受路由策略的约束,不同的流可能会经过不同的转发设备,如图 1 所示. 假设网络中共有 4 条流 t_1、t_2、t_3、t_4,根据路由约束,经过交换机 S_1、S_2、S_3、S_4 的流集合分别为 $\{t_1, t_2\}$、$\{t_2, t_4\}$、$\{t_2, t_3\}$、$\{t_3, t_4\}$,此时若在每个交换机处都进行采样,则 t_1、t_2、t_3、t_4 被采样的次数分别为 1、3、2、2,冗余采样率为 100%. 事实上,若仅在 S_1 和 S_4 处进行采样,即可保证每条流恰好被采样 1 次,相对在所有位置进行全采样的方式能够有效节约测量资源.

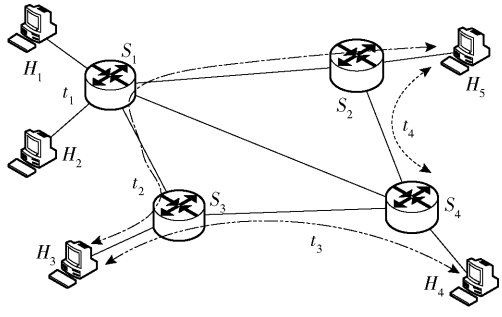

图 1 sFlow 采样点设置示例

综上,在进行流量采样的过程中,如何合理确定采样点是一个需要解决的问题. 对此,利用控制器具有全网流量路由信息这一特点,结合路由信息进

行采样点部署的决策，达到优化采样设计的目标. 假设网络中共有 m 台可以进行采样的交换机 $S=\{s_1,s_2,\cdots,s_m\}$ 和 k 条流 $T=\{t_1,t_2,\cdots,t_k\}$，矩阵 A 为 1 个 $k\times m$ 的布尔矩阵，代表流 t_i 是否经过交换机 s_j.

对于 A 中的任一元素 a_{ij}，其值满足式(1)的约束：

$$a_{ij}=\begin{cases}1, & \text{若流 } t_i \text{ 经过交换机 } s_j\\ 0, & \text{否则}\end{cases} \tag{1}$$

以图 1 所示的具有 4 台交换机和 4 条流的网络为例，其路由矩阵 $\boldsymbol{A}$ 具有如下形式：

$$\boldsymbol{A}=\begin{pmatrix}1&0&0&0\\1&1&1&0\\0&0&1&1\\0&1&0&1\end{pmatrix} \tag{2}$$

式(2)表示流 1 经过交换机 1；流 2 经过交换机 1、2、3 等，而测量点的选择问题本质上就是在上述 m 台交换机中选择能够覆盖全部流的最小化交换机集合，以降低测量耗费.

上述问题可以抽象为集合覆盖问题：从 1 个 k 行、m 列的 0-1 矩阵 $\boldsymbol{A}_{k\times m}$ 中选出若干列，使得其中具有 1 的行覆盖所有的行，并能够使得这一过程中的代价最小. 因此，该问题可以建模为

$$\min\sum_{j=1}^{m}\beta_j x_j \tag{3}$$

$$\text{s.t.}\sum_{j=1}^{m}a_{ij}x_j\geqslant 1,\ i=1,2,\cdots,k \tag{4}$$

$$\sum_{s_j\in \mathbf{S}}a_{ij}\geqslant 1 \tag{5}$$

$$x_j=\begin{cases}1, & \text{若交换机 } s_j \text{ 被选择为采样点}\\ 0, & \text{否则}\end{cases} \tag{6}$$

其中 β_j 代表选择交换机 S_j 作为采样点所需要的耗费，约束条件(4)确保每条流都至少被 1 台交换机所采样，约束条件(5)表示每条流至少经过 1 台交换机，否则无法完成采样任务，约束条件(6)说明 x_j 是一个 0-1 变量，代表是否选择交换机 S_j 作为采样点. 该问题是一个典型的 NP 完全问题，需要设计相应的启发式算法进行求解. 对此，基于 0-1 分支定界的思想，提出一种采样点选择的启发式求解算法，如算法 1 所示.

算法 1　基于分支定界的采样点选择

输入：OpenFlow 交换机集合 S，流路由矩阵 $\boldsymbol{A}$

输出：采样点集合 W

```
1   W = ∅, k = 0, queue = NULL;
2   g = gain(W, A);
3   node = (W, k, g);
4   queue.add(node);
5   while (true)
6     node = get_first(queue);
7       if node is empty
8         break;
9     end if
10        k = node.k + 1;
11    for each sw in S-node.s
12          TS = node.s ∪ {sw};
13          g = gain(TS, A)
14          if (TS, k, g) in queue or g < gain.min_
bound(k)
15              continue;   //剪枝
16          elseif g > gain.max
17              W = TS;
18        end if
19          node = (TS, k, g);
20          queue.add(node);
21    end for
22  end while
23  return W
```

在算法 1 中的第 1 行，首先初始化最终采样点集合 W、选择的采样点数目 k、结点队列 queue，在算法第 2 ~ 4 行计算当前所选择采样点获得的收益值，构造结点，并将结点加入状态空间树(用队列保存状态结点). 第 5 ~ 22 行循环进行采样点的选择，其中在每次选择 1 个新的交换机 sw 后，第 13 行计算其收益，若小于最小收益下界，则进行剪枝；否则，若大于当前收益，则更新 W 集合，最终返回所选择的采样点集合.

确定采样点之后，每台采样交换机上的 sFlow 代理对经过的分组按照预设的采样率进行采样，并将采样到的分组信息发送至 sFlow 采集器. 如图 2 所示，sFlow 采集器针对所有采样到的流维护 1 个哈希表 HT，假设某一采样分组 p 具有流标识 id_p，且 $f_i=\mathrm{Hash}(\mathrm{id}_p)$，则在收到该分组后，sFlow 采集器将判断 f_i 是否已经在哈希表 HT 内，若是，则 sFlow 采集器将 HT$[f_i]$ 的值增加 volume(p)；否则，其将创建一个新的初始数值为 volume(p) 的表项 HT$[f_i]$. 一旦HT$[f_i]$ 的数值超过预定的疑似大流阈值，则 sFlow 采集器将立即发送流标识 id_p 到 SDN 控制器，并标

志该流为正在被 SDN 控制器测量的疑似大流.

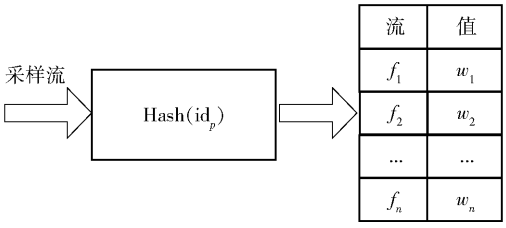

图 2 sFlow 采集器中的疑似大流检测

在大流检测过程中,假阳性和假阴性是 2 个广泛使用的评价指标,前者代表检测结果中的误判情况,而后者表明了检测结果中的漏判情况. 本步骤的主要目标是找出疑似大流,因此漏判比误判具有相对更严重的后果,后者可以在流表测量阶段进行过滤,只是会带来一定的流表空间浪费,而前者将直接导致测量结果的错误. 因此,在疑似大流的判定过程中,通过引入一个松弛因子 λ 来适当降低疑似大流的判断阈值,容许一些误判,并随后在流表测量中将误判流排除.

1.3 基于流表查询的大流检测机制

在完成疑似大流的检测后,sFlow 采集器将所检测到的疑似大流发送给 SDN 控制器,SDN 控制器根据接收到的疑似大流向相应的 OpenFlow 交换机下发测量流表项开启测量任务,并随后读取这些测量流表项,从而获得每条测量流的实际流量数据,据此检测出真正的大流及其流量值等. 图 3 给出了大流检测机制的示意,其中加载测量规则过程中,控制器根据 sFlow 采集器所上报的疑似大流信息生成测量规则,并按照一定的策略[10]加载到相应的 OpenFlow 交换机上. 而读取测量数据则是由控制器周期性地从交换机处读取最新的流量测量数据,进而通过流量值判断出真正的大流.

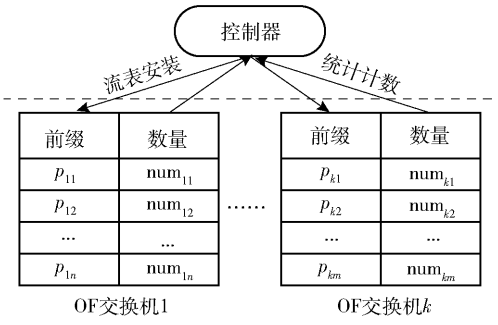

图 3 基于流表查询的大流检测机制

算法 2 给出了 SampleFlow 大流检测机制的总体流程.

算法 2 SampleFlow 大流检测算法

输入: OpenFlow 交换机集合 S, 流路由矩阵 $\boldsymbol{A}$, 采样率 r, 疑似/真实大流阈值 γ/γ', 测量流表分发策略 $\boldsymbol{\Phi}$

输出: 大流标识及其测量值矩阵 $\boldsymbol{M}$

$W=\varnothing$, $M=\mathrm{O}$; //W 为采样点集合
$N=\varnothing, j=0$; //N 为上报的疑似大流集
获取 sFlow 采样点集合 W; //算法 1
在采样点 W 处依照采样率 r 进行 sFlow 采样;
while (true) //采样开始
  更新采样流集合 F 及测量值 HT;
  for each f_i in F //F 为采样流集合
    if HT(f_i) $\geqslant\gamma$ //流 f_i 测量值超出阈值
      $N=N\cup\{f_i\}$;
    end if
  end for
  为疑似大流 F 生成相应测量流表项;
  Vol = ReadState. obtain(F); //读测量流表项
  for each f_i in F
    if Vol(f_i) $\geqslant\gamma'$
      $M[j][0]=f_i$, $M[j][1]=\mathrm{Vol}(f_i)$, $j=j+1$;
    end if
  end for
end while //采样结束
return M

在基于流表查询的大流检测机制中,流表空间资源是 OpenFlow 交换机最宝贵的资源,每台交换机的 TCAM 资源非常有限,而 SampleFlow 首先利用相对充裕的静态随机存取存储器(SRAM, static random access memory)资源来过滤出疑似大流,从而有效降低了对 TCAM 资源的消耗,并且解决了仅采用 sFlow 测量带来的高误差率问题.

2 试验评价及分析

笔者分别在 sFlow 采集器和 POX 控制器上实现了 SampleFlow 的采样点选择、测量流表项分配以及最终的大流识别等模块,并利用仿真和因特网实际测量数据从不同方面对其性能进行分析.

场景 1 采样点选择算法性能分析

为了验证 SampleFlow 在确定采样点方面的有效性,实现了随机和最大结点介数(max # of OD)2

种采样点选择算法. 前者随机进行采样点选择,后者则始终选择介数(网络中经过该结点的流数目)最大的点为采样点,当能够覆盖网络中所有的流时,算法终止.

首先利用 Abilene 的网络和流量数据[12]进行性能评估,为了便于讨论,这里假设在每台 OpenFlow 交换机上进行采样的代价相同. 此外,选取采样点数量和冗余流数量 2 个指标来对算法的性能进行评价,其中采样点数量代表为了实现网络中流的全覆盖,至少需要在多少个结点进行采样. 冗余流数量则代表在测量过程中有多少额外的冗余测量.

图 4 所示为在不同规模的流数目情况下,为了实现流采样的全覆盖,用 3 种算法得到的最小采样点数目的变化趋势. 可以看出,在 Abilene 拓扑中,随着流数目的增加,所需要的采样点数目也呈现增加的趋势(随机选择算法的波动相对较大),但 SampleFlow 算法得到的采样点数目始终优于另外 2 种算法,表明 SampleFlow 能够以相对更小的代价实现流采样. 这是因为 SampleFlow 算法采取的是一种启发式的类贪心机制,它存在一个针对当前阶段采样备选节点的筛选过程,在每一轮的选择中通过利用分支定界的思想来不断剔除那些效益较差的节点,从而获得总体效益较优的最终结果. 相比之下,随机选取和节点介数最大策略在理论和实践上都没有体现出一种相对稳定的"去差选优"趋势,尤其对于随机选取策略,其本身的随机性更是给该方法带来了较大的不确定性. 因此, SampleFlow 算法在采样点选择方面有着更好的优越性,能够以较小的代价实现流采样.

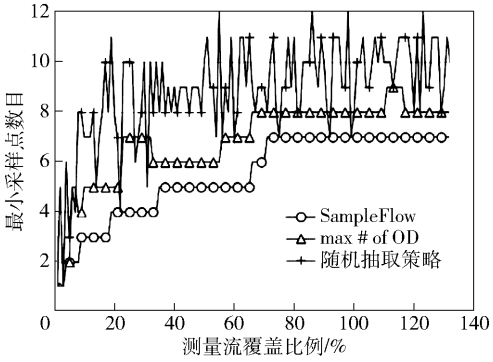

图 4　Abilene 不同流数目时所需采样点变化情况

图 5 所示为基于 3 种算法所选择的采样点进行采样测量时冗余测量流所占的比例,可以发现, SampleFlow 算法冗余流的比例相对最低. 这样的实验结果符合预期,因为 SampleFlow 算法的采样点选择模块优化了各个采样点在网络流特征方面的代表能力,从而显著降低了整个测量系统的工作冗余度.

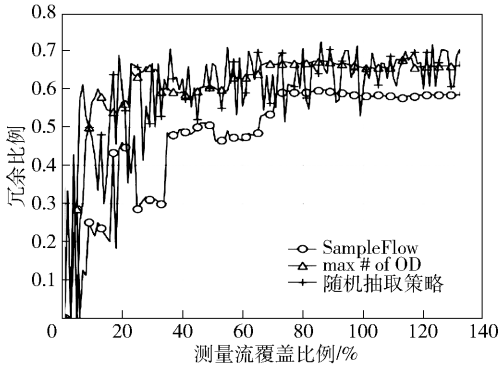

图 5　Abilene 不同流数目时测量流冗余情况

由于 Abilene 网络拓扑只包含 12 个路由器结点,规模相对较小,笔者进一步又利用 Brite 拓扑生成器[13]生成 100 个结点的网络拓扑,且拓扑中的网络流量定义为源和目的节点对(OD, origin destination)之间的流量,路由按照最短路规则进行路由选择.

图 6 所示为在不同规模的流数目情况下全覆盖采样所需的最小采样点数目变化趋势. 可以看出,在 Brite 生成的拓扑中,SampleFlow 算法得到的采样点数目同样始终优于另外 2 种算法,进一步验证了 SampleFlow 在采样点选择方面的有效性.

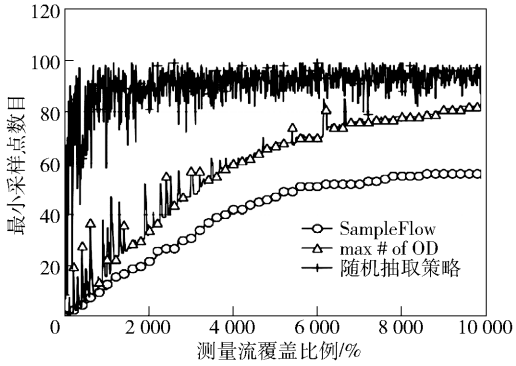

图 6　Brite 不同流数目时所需采样点变化情况

图 7 示出了在 Brite 生成的网络拓扑中,基于 3 种算法所选择的采样点进行采样测量时冗余测量流所占的比例,可以发现 SampleFlow 算法所得到的冗余流比例相对最低. 此外,随着网络规模的扩大,SampleFlow 与其他 2 种方案之间的差距也就变得愈加明显,而这种现象产生的原因在于采样

巧合性地对网络系统带来的影响会随着规模的扩大和结构的复杂而逐渐减弱,故而在仿真实验环境中, SampleFlow 算法展现出更加明显的性能优势.

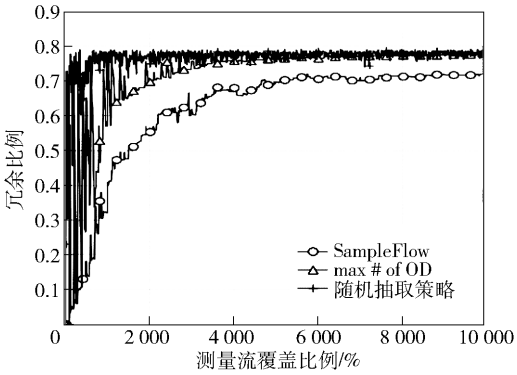

图7 Brite 不同流数目时测量流冗余情况

综合本部分的试验结果可以发现,无论对于真实网络拓扑及流量数据,还是仿真网络拓扑及流量数据,SampleFlow 的采样点选择机制均能够得到相对好的结果.

场景2 SampleFlow 大流检测精度分析

对于 sFlow 采集器上报的疑似大流,SampleFlow 中 SDN 控制器通过直接读取 OpenFlow 交换机中对应测量流表项的计数器获取流的真实大小,因此所检测出的大流一定为网络中真实的大流,而不存在误判现象,亦即其结果的假阳性为0. 另一方面,在 sFlow 采样阶段,可能会有一些真正的大流因采样误差而被过滤,导致无法被检测出,会存在漏判的可能,存在一定的假阴性. 因此,在测量精度评价方面,应以假阴性指标对 SampleFlow 进行评价.

实验以因特网应用数据分析中心(CAIDA, center for applied internet data analysis)测量数据集[14]为基础,这里大流定义为发送数据量超过1 MB 的流. 对于 sFlow 采样方法,分别以 0.01 ~ 0.05 不同的采样率进行流量采样,然后基于采样结果推断每条流的大小,并找出那些大于 1 MB 的流作为检测出的大流.

在 SampleFlow 中,其第1阶段同样采用类似的采样方法,但为了减小漏判率,笔者以不同的松弛因子 λ 放松对大流检测标准的约束,分别以 0.9、0.8 和 0.7 倍于 1 MB 的流判断为疑似大流,随后在第2阶段的流表测量方法中再从这些疑似大流内检测出真正的大流. 整个网络中 OpenFlow 交换机总的 TCAM 可用流表空间设置为1 000. 这种松弛判断方法不适合于仅采用 sFlow 采样的机制,因为其会带来较高的误判率,而 sFlow 无法消除这种误判.

图8 所示为 SampleFlow 方法和仅采用 sFlow 采样方法得到的大流检测精度对比情况,可以看出,在同样的采样率设置下,具有最大松弛度(λ 最小)的 SampleFlow 方法其检测精度也最高. 然而,随着检测标准中松弛度的增加,也会带来 TCAM 资源消耗的同步增加. 因此,在选择 λ 时,需要结合测量精度的要求和测量资源的限制进行合理折中. 此外,大流检测的精度同样随着采样率的增加而增加.

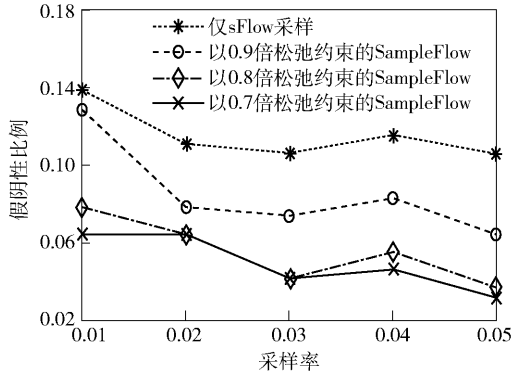

图8 SampleFlow 与 sFlow 精度对比

3 结束语

快速、准确地识别出网络中的大流对于网络管理和网络安全具有重要意义,然而,当前无论是采样机制还是流表测量机制都存在一些问题. 笔者针对 SDN 大流检测问题,提出了一种基于采样和流表查询的两阶段方法 SampleFlow. SampleFlow 首先通过粗粒度的采样方法寻找出网络内的疑似大流,随后通知 SDN 控制器对这些疑似大流采样流表测量的方法进行细粒度监测,从而找出真正的大流. 由于网络中大流在数量上仅占总的流数很少的一部分,这一方法相对于直接测量有效降低了对 OpenFlow 交换机 TCAM 资源的耗费,同时又解决了直接采样方法所存在的精度不高等问题. 实验结果显示,SampleFlow 能够有效提升大流检测的精度,降低 TCAM 资源的耗费. 此外,SampleFlow 可以直接基于现有 OpenFlow 的工作逻辑实现,具有良好的扩展性. 在下一步工作中,拟研究混合 SDN 网络环境中的大流快速检测机制,并考虑将其应用至具体的流量工程中.

参考文献：

[1] The Internet Society. Cisco systems NetFlow services export version 9 [EB/OL]. (2004-10) [2019-04-15]. http://www.rfc-editor.org/rfc/rfc3954.

[2] InMon. Traffic monitoring using sFlow [EB/OL]. (2013-03) [2019-04-15]. https://sflow.org/sFlow-Overview.pdf.

[3] Mai J, Chuah C N, Sridharan A, et al. Is sampled data sufficient for anomaly detection [C]//2006 ACM SIGCOMM Conference on Internet Measurement. Rio De Janeiro: ACM, 2006: 165-176.

[4] Kekely L, Kucera J, Pus V, et al. Software defined monitoring of application protocols [J]. IEEE Transactions on Computers, 2016, 65(2): 615-626.

[5] Yuan L, Chuah C N, Mohapatra P. ProgME: towards programmable network measurement [J]. IEEE/ACM Transactions on Networking, 2011, 19(1): 115-128.

[6] Yang Tong, Jiang Jie, Liu Peng, et al. Elastic sketch: adaptive and fast network-wide measurements[C]//2018 ACM SIGCOMM Conference. Budapest: ACM, 2018: 561, 575.

[7] Moshref M, Yu M, Govindan R. Resource/accuracy tradeoffs in software-defined measurement [C]//2013 ACM SIGCOMM HotSDN. Hong Kong: ACM, 2013: 73-78.

[8] Harrison R, Cai Q, Gupta A, et al. Network-wide heavy hitter detection with commodity switches [C]//2018 ACM Symposium on SDN Research. Los Angeles: ACM, 2018: 1-7.

[9] Moshref M, Yu M, Govindan R, et al. DREAM: dynamic resource allocation for software-defined measurement [J]. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 419-430.

[10] Xing Changyou, Ding Ke, Hu Chao, et al. Sample and fetch-based large flow detection mechanism in software defined networks [J]. IEEE Communications Letters, 2016, 20(9): 1764-1767.

[11] Suh J, Kwon T T, Dixon C, et al. OpenSample: a low-latency, sampling-based measurement platform for commodity SDN [C]//2014 IEEE International Conference on Distributed Computing Systems. Madrid: IEEE Press, 2014: 228-237.

[12] MattR. 2004 Abilene dataset [EB/OL]. (2004-05-15) [2019-04-16]. http://www.maths.adelaide.edu.au/matthew.roughan/project/traffic_matrix/.

[13] Medina A, Lakhina A, Matta I, et al. BRITE: an approach to universal topology generation[C]//2001 International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. Cincinnati: IEEE, 2001: 346-353.

[14] CAIDA. Anonymized internet traces 2014 [EB/OL]. (2014-10-09) [2019-04-17]. http://www.caida.org/data/passive/passive_2014_dataset.xml.