

基于 CBF-SS 策略的大流识别算法^{*}

赵小欢^{1†}, 李明辉²

(1 中国人民解放军 95034 部队, 广西 百色 533616; 2 空军后勤部, 北京 100720)

(2014 年 3 月 31 日收稿; 2014 年 7 月 22 日收修改稿)

Zhao X H, Li M H. Large flow identification based on counting Bloom filter and space saving [J]. Journal of University of Chinese Academy of Sciences, 2015, 32(3): 391-397.

摘 要 在分析大流识别算法中的散列方法和计数方法的优缺点的基础上, 针对网络流的重尾分布特性, 提出一种能够有效结合散列方法和计数方法优点的大流识别算法 CBF-SS (counting Bloom filter & space saving). 该算法首先采用改进的计数型布鲁姆过滤器 (counting Bloom filter, CBF) 过滤掉大部分的小流, 然后通过 SS (space saving) 计数算法识别出网络中的大流. 理论分析和实验结果表明, CBF-SS 算法具有较低的时间复杂度和空间复杂度, 在大流识别效果上远优于 SS 等算法.

关键词 网络流; 大流; 计数型布鲁姆过滤器; space saving 算法

中图分类号: TP393 **文献标志码:** A **doi:** 10. 7523/j. issn. 2095-6134. 2015. 03. 015

Large flow identification based on counting Bloom filter and space saving

ZHAO Xiaohuan¹, LI Minghui²

(1 95034 Unit of PLA, Baise 533616, Guangxi, China; 2 Air Force Logistics Department, Beijing 100720, China)

Abstract Aiming at the characteristics of the heavy-tailed distribution of network flows, we propose a large flow identification algorithm, CBF-SS (counting Bloom filter and space saving), on the basis of analyzing advantages and deficiencies of hashing and counting methods used for large flow identification. It has the capability of combining the advantages of hashing and counting methods efficiently. The algorithm CBF-SS uses the counting Bloom filter to filter mass of small flows at first. Then, CBF-SS uses the SS (space saving) counting method to our large flows. Both theoretical and experimental results show that CBF-SS is very space-saving and time-efficient and it performs much better than the SS algorithm in the precision of large flow identification.

Key words network flows; large flows; counting Bloom filter; space saving

网络中的大流是指在一定的测量时间内字节数或报文数超过特定阈值的流, 大流识别问题属于数据流频繁项挖掘问题的一种典型应用. 在当

前骨干网络链路速率呈几何倍数增长的情况下, 实时准确地挖掘出网络中的大流对于网络管理和网络安全具有重要的意义^[1-2]. 依据大流识别原

^{*} 国家自然科学基金 (61201209) 和陕西省自然科学基金重点项目 (2012JZ8005) 资助

[†] 通信作者, E-mail: zxhxh_2012@163.com

理的不同,大流识别方法通常可以分为基于抽样的大流识别方法、基于散列的大流识别方法和基于计数的大流识别方法 3 类^[3].

文献[4-5]采用抽样的方法实现大流识别,尽管该类方法实现简单,但抽样方法通常需要假定网络流服从特定的分布以便更好地抽取到需要的信息以及更好地从抽样数据中估算原始流信息.然而,由于网络流的突发性及多样性,目前网络流并不存在精确的数学描述模型,在数据到达之前并不能事先预知网络流的到达.因此,基于抽样的大流识别方法在大流召回数量和统计流长指标上通常会存在较大的误差.

文献[6-8]通过哈希映射将具有相同散列值的流项不断累加实现大流识别,该类方法属于基于散列的大流识别方法.此类方法具有良好的时空特性,能够将网络 IP 流中较长的五元组信息映射成很短的位数,并且原始频率超过阈值的 IP 流被映射后的频率一定也会超过该阈值.此类方法的缺点在于由于哈希冲突的存在,导致大流误报的情况时有发生;另外由于哈希函数可逆性较差,导致散列方法较难恢复出原始流信息,这一缺点进一步限制了散列方法的使用.

基于计数的大流识别方法在有限的缓存空间内不断地计数和剪枝操作实现大流识别,计数方法能够保存原始流信息,在保证计数方法时间和空间复杂度的情况下,该类算法通常具有较好的精度和流完整性.计数方法的关键在于“在不确定数据流中各数据到达顺序的情况下,每次剪枝操作时需要尽量删除最不可能发展为大流的流项”.文献[9-14]提出一些经典的基于计数的大流识别方法.由于 SS 算法在剪枝操作时能够综合考虑时间和流长因素, Cormode 和 Hadjieleftheriou^[15]和 Liu 等^[16]分别指出“在所比較的基于计数的频繁项统计方法中,SS 算法是所对比计数方法中的最优算法”.

文献[17-20]则进一步将散列方法和计数方法相结合实现大流识别,其中文献[17]在 LRU 页面置换算法的基础上,通过布鲁姆过滤器(Bloom filter, BF)标记大流提高大流识别的准确率;文献[18]将空分编码布鲁姆过滤器(space code Bloom filter, SCBF)与 LRU 页面置换算法结合实现大流识别,该方法的复杂度较高;文献[19-20]采用计数型布鲁姆过滤器(counter Bloom filter, CBF)过

滤掉部分小流,然后再通过计数算法识别网络中的大流,达到较高的大流识别准确率,算法的复杂度要高于单一的大流识别算法.

在综合考虑散列方法和 SS 计数算法优缺点的基础上,本文提出一种能够充分结合散列方法和 SS 计数算法优点的大流识别方法 CBF-SS,算法的基本思想在于:首先通过 CBF 过滤掉网络流量中大量的小流,然后通过 SS 算法挖掘出网络中的大流.由于散列方法中原始流长超过阈值的 IP 流被映射后流长一定会超过阈值,因此网络中的大流能够顺利地通过 CBF,同时由于 SS 计数算法保存有网络流的基本信息,因此即使部分小流由于哈希冲突通过了 CBF,SS 计数算法也能很快将其淘汰.

1 CBF-SS 算法设计

1.1 算法基础理论

1.1.1 布鲁姆过滤器 CBF

计数型布鲁姆过滤器 CBF 是一种支持集合的近似表示以及近似从属查询的数据结构,BF 由 k 个 Hash 函数 h_1, h_2, \dots, h_k 和一个 m 位比特向量 V 构成.对于集合 $S = \{s_1, s_2, \dots, s_n\}$ 中的元素 x ,当 x 需要插入向量 V 时,计算 x 对应的 k 个哈希值,并将向量 V 中 $V[h_i(x)] (i = 1, 2, \dots, k)$ 置为 1;当查询元素 x 是否出现过时,需要检查向量 V 中 $V[h_i(x)] (i = 1, 2, \dots, k)$ 是否为 1,如果其中任意一个不为 1,则判断 x 未出现过,否则认为 x 已出现过.由于存在哈希冲突,可能出现某一元素对应的 k 个位置全部被其他元素置为 1 的情况,BF 的假阳性概率为

$$P^{\text{BF}}(n, m, k) = (1 - (1 - 1/m)^{kn})^k \approx (1 - e^{-kn/m})^k. \quad (1)$$

BF 较好地支持元素的插入和查询,但是不能支持元素的删除及出现频率查询,为解决这一问题,Fan 等^[21]提出计数型布鲁姆过滤器数据结构,CBF 初始条件与 BF 的区别在于将向量 V 中每个单元由 1 bit 扩展为 1 个计数器,当元素 x 使用 k 个哈希函数插入到向量 V 时,将映射到的 k 个单元的计数值都增加 1,当元素 x 需要从 CBF 中删除时,若 x 映射到的 k 个单元的计数值都大于 0,则将这 k 个计数值均减 1,否则认为 x 未出现过.

1.1.2 SS 算法

SS 算法是 Metwally 等^[13]提出的一种基于计

数的频繁项挖掘算法, 算法的原理为: 算法共维护 n_{ss} 个表项 $\langle e, f(e), \Delta(e) \rangle$, 其初始值全部为 0, 其中 e 表示表项的标志, $f(e)$ 表示频率估计值, $\Delta(e)$ 表示最大误差, 当数据项 i 到达时, 若数据项 i 对应的表项 e_i 正在被监控时, 表项 e_i 更新为 $\langle e_i, f(e_i) + 1, \Delta(e_i) \rangle$, 否则找到 n_{ss} 个表项中 $f(e)$ 值最小的 1 个表项 $\langle e_j, f(e_j), \Delta(e_j) \rangle$, 并将表项 e_j 替换为 $\langle e_i, f(e_j) + 1, f(e_j) \rangle$.

尽管 SS 算法具有较好的频繁项挖掘效果, 但若将 SS 算法应用于大流识别, 由于 SS 算法统计的流长会随着新流的出现而不断累加, 同时网络流空间的数目远远大于缓存器的大小, SS 算法不可能为网络流分配足够的缓存空间, SS 算法缓存中后期出现的流的流长会由于前期的累加操作而被严重高估. SS 算法对于前期出现的流是不公平的, 即使前期出现的流的流长超过大流阈值, 若后期新出现的流的初始流长被累加至超过前期出现的大流, 此时前期出现的大流将被淘汰. 因此在网络流数目远大于缓存数目的情况下, 有必要对 SS 算法进行优化实现大流识别.

1.2 CBF-SS 算法

网络流长呈现出明显的重尾分布特性, CBF-SS 算法利用网络流的这一特性, 将 CBF 散列方法和 SS 计数方法结合起来实现网络流频繁项挖掘, 算法的基本流程如图 1 所示: 当一个报文 X_i 到达时, 首先查看报文 X_i 对应的流项 e_{xi} 是否正在被 SS 算法监控. 若流项 e_{xi} 正在被 SS 算法监控, 则按照 SS 算法的原理更新流项 e_{xi} , 否则查看 CBF 向量 V 中 $V[h_j(X_i)] (j = 1, 2, \dots, k)$ 是否等于 T_{CBF} , 若 $V[h_j(X_i)] = T_{CBF} (j = 1, 2, \dots, k)$, 则由 SS 算法处理报文 X_i , 否则通过 CBF 处理报文 X_i .

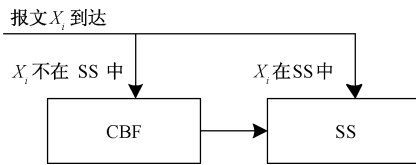


图 1 CBF-SS 算法原理

Fig. 1 Principle of CBF-SS

CBF-SS 算法中 CBF 流过滤的基本原理为: 当一个流项映射到的 k 个计数器的值全部等于阈值 T_{CBF} 时, 判定此流项通过 CBF 过滤器, 否则更新 CBF 中对应计数器的值. 由于哈希冲突的存

在, CBF 的过滤效果会随着 CBF 中计数器值的增大而变差, 为确保算法的过滤效果, 对 CBF 过滤过程增加 2 点约束: 1) 当 CBF 中计数器值等于 T_{CBF} 的计数器数目 N_{CBF} 超过特定阈值 N_0 时, CBF 中全部非 0 计数器的值均自减 1; 2) 每次 CBF 计数器需要执行加 1 操作时, 只更新映射到的 k 个计数器中值最小的计数器的值, 其余计数器的值不变. 另外, 由文献[12, 22]可知, 网络中流长小于 16 的流在总体流量中所占比例极高, 因此 CBF 每个计数器设置为 4 bit 即可过滤大部分的小流, 即 $T_{CBF} = 15$, 且当 CBF 计数器的值等于 T_{CBF} 时, 该计数器值不再增加.

CBF-SS 算法的具体步骤为:

1) 报文 X_i 到达, 查看 X_i 对应的流项 e_{xi} 是否正在被 SS 算法监控, 若 e_{xi} 在 SS 缓存中, 则流项 e_{xi} 更新为 $\langle e_{xi}, f(e_{xi}) + 1, \Delta(e_{xi}) \rangle$, 结束此次循环, 等待下一个报文到达; 否则, 转入步骤 2);

2) 判断 CBF 向量 V 中 $V[h_j(X_i)] (j = 1, 2, \dots, k)$ 是否等于 T_{CBF} , 若 $V[h_j(X_i)] = T_{CBF} (j = 1, 2, \dots, k)$, 则查找 SS 缓存中 $f(e)$ 值最小的 1 个表项为 $\langle e_j, f(e_j), \Delta(e_j) \rangle$, 并将表项 e_j 替换为 $\langle e_{xi}, f(e_j) + 1, f(e_j) \rangle$, 结束此次循环, 等待下一个报文到达; 否则, 转入步骤 3);

3) $V[h_j(X_i)] (j = 1, 2, \dots, k)$ 中值最小的计数器执行加 1 操作, 判断 CBF 向量 V 中 $V[i] = T_{CBF} (i = 1, 2, \dots, m)$ 的计数器的数目 N_{CBF} , 若 $N_{CBF} \geq N_0$, 则 CBF 中非 0 计数器均执行减 1 操作. 结束此次循环, 等待下一个报文到达.

1.3 CBF-SS 算法分析

由于 CBF 中 $T_{CBF} = 15$, 大流阈值 $T_{big} \gg T_{CBF}$, 由哈希函数的特性可知, 大流一定会通过 CBF 过滤器. 对于流长 $f = 1$ 的流 e , 其能通过 CBF 意味着流 e 映射到的 k 个计数器的初始值全部等于 T_{CBF} , 而流 e 映射到这 N_{CBF} 个计数器的概率为 $(N_{CBF}/m)^k$, 由 $N_{CBF} < N_0$ 可知流 e 能够通过 CBF 的概率小于 $(N_0/m)^k$, 因此流 e 被 CBF 过滤器阻止的概率大于 $1 - (N_0/m)^k$. 对于流长为 $f (f < T_{CBF})$ 的小流 e , 其 f 个报文都被 CBF 过滤器阻止的概率大于 $(1 - (N_0/m)^k)^f$. 由此可见, 对于流长越短的流, CBF 的过滤效果越好, 而网络流中小流的数量远远大于大流, 因此 CBF 能够在几乎不影响大流通过效果的情况下过滤掉网络中大

部分的小流.

由文献[13]可知,对于 SS 算法给定错误率 ε ,只要算法表项总数 $n_{ss} \geq 1/\varepsilon$,就能保证流长大于 εN 的网络流被监控,其中 N 为到达 SS 算法的报文总数.以 $n_{ss} = 10\,000$ 为例,若 $N = 3.2 \times 10^7$ (以每个报文 40 byte 为例,OC192 链路 1 s 内到达的报文数就能达到 N),此时 SS 算法能保证流长大于 3 200 的网络流被监控.而若要在 N 保持不变的情况下监控到全部流长大于 1 000 的流,则需要表项总数 $n_{ss} = 32\,000$.由于高速缓存的容量是极其有限的,在实际应用中不断增加表项总数 n_{ss} 是不切实际的.而在 CBF-SS 算法中,CBF 过滤器过滤掉一部分报文,因此到达 SS 的报文数 N 减小,其能够在 n_{ss} 不变的情况下有效改善 SS 算法的性能.更为重要的是,CBF-SS 算法能够过滤大量小流,从而使得到达 SS 的流数大为减少,这样 SS 算法剪枝的次数将大大降低,极大地避免了 SS 剪枝操作时将大流误删除的情况.

由于 CBF 过滤器不需要保存 IP 流五元组及指针等信息,且 CBF 中每个计数器设置为 4 bit,与之相对的是基于计数的大流识别算法中每个流项需要保存 IP 流五元组、计数器、指针大约 200 bit 的信息^[10],因此 CBF 的空间利用率是很高的.通过 2.1 节的实验可知,CBF 占用的缓存相对于 SS 算法来说是可以忽略的.CBF 算法中对于每个报文需要执行 k 次哈希运算及计数器更新,时间复杂度为 $O(k)$,CBF 算法执行 1 次剪枝操作的时间复杂度为 $O(m)$,若 1 次剪枝操作删除的报文数为 \tilde{m} ,则 CBF 每报文时间复杂度为 $O(k + \frac{m}{\tilde{m}})$.由于通常情况下 CBF 中 k 很小,并且哈希运算及计数器更新操作可以通过 FPGA 等芯片实现并行处理,同时在 $N_{CBF} \geq N_0$ 时 CBF 向量 V 中存在 $V[i] = 0 (i = 1, 2, \dots, m)$ 的概率非常小,因此 $\tilde{m} \approx m$,综上考虑,可以认为 CBF 平均每报文时间复杂度为 $O(1)$.而由文献[13]知 SS 算法每报文时间复杂度为 $O(1)$,空间复杂度为 $O(1/\varepsilon)$,其中 $1/\varepsilon$ 为 SS 算法的表项数目.因此整个 CBF-SS 算法的每报文时间复杂度为 $O(1)$,空间复杂度接近 $O(1/\varepsilon)$.

2 实验及评价

本文采用 Plab 软件^[23] 获取网络流量的基准

信息,通过 Matlab 软件编程实现各种大流识别算法.选用 3 组互联网骨干链路流量数据作为测试数据集,如表 1 所示.其中 trace_2003 为 CAIDA^[24] 2003 年一条 OC48 链路流量,而 trace_2011 与 trace_2013 为 MAWI^[25] 在 2011 年和 2013 年采集的一条跨太平洋骨干链路流量.

表 1 网络流量基本信息

Table 1 Basic information of three traces

数据	持续 时长/s	报文数	流数	流长 > = 1 000 的流数
trace_2013	899	37 094 876	9 076 416	1 842
trace_2011	900	41 776 906	3 823 497	2 511
trace_2003	299	17 758 098	1 107 637	2 394

2.1 CBF-SS 算法过滤效果分析

令 CBF-SS 算法中 CBF 的计数器数量为 $m = 2^{14}$,计数器位宽为 4 bit,哈希函数数目 $k = 3$, $T_{CBF} = 15$, $N_0 = m/4$,SS 表项数目 $n_{ss} = 10\,000$,则 CBF 过滤器占用空间不足 SS 算法 500 个表项占用的空间.哈希函数选自于 Arash Partow 提供的 general purpose Hash 函数簇.

表 2 描述 CBF-SS 算法中原始流量信息和到达 SS 模块流量信息的对比情况,可以看出,CBF 的过滤效果是非常好的,CBF 能够在保证大多数网络报文通过的情况下,使得到达 SS 模块的流数大为减少.因此 CBF 能够有效过滤掉网络中绝大部分的短流.

表 2 CBF-SS 算法中 CBF 过滤效果

Table 2 Filtering performance of CBF in CBF-SS

数据	原始报文数	到达 SS 的 报文数	原始流数	到达 SS 的 流数
trace_2013	37 094 876	23 807 136	9 076 416	1 314 849
trace_2011	41 776 906	32 449 878	3 823 497	949 370
trace_2003	17 758 098	12 013 101	1 107 637	356 550

图 2 以 trace_2003 为例描述 $n_{ss} = 10\,000$ 情况下 CBF-SS 算法中原始大流流长和到达 SS 模块的大流流长之差,可以看出,原始流量中全部的大流均能够通过 CBF 过滤器,其中绝大部分大流流长失真较小,流长差值在 40 报文以上的流数仅为 6,而全部 2 394 条流的流长差值之和为 23 136,平均到每个大流上流长差值仅为 9.66,该值远小于大流流长,因此,CBF 能够在几乎不影响大流

通过效果的情况下过滤掉网络中绝大部分的短流。

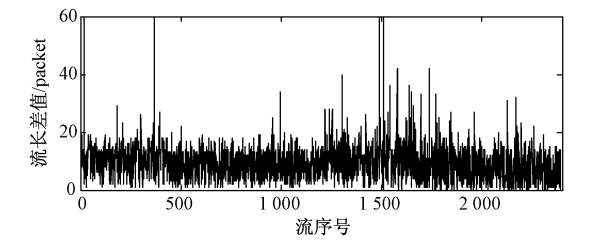


图2 CBF-SS 算法中 CBF 大流通过效果

Fig.2 Large flow pass performance of CBF in CBF-SS

2.2 大流识别效果对比

本文选用 SS 算法^[13]、LLR 算法^[9]、LRU2 算法^[10]和 CBF-SS 算法进行比较. 为 4 种算法分配表项数目均为 L . 其中 SS 算法以 $f(e) - \Delta(e)$ 作为流长估计值; LLR 算法两级缓存各分配 $L/2$ 表项, 算法阈值设置为 2; LRU2 算法两级缓存各分配 $L/2$ 表项, 算法阈值设置为 300; CBF-SS 算法中 CBF 的计数器数量为 $m = 2^{14}$, 计数器位宽为 4 bit, 哈希函数数目 $k = 3$, $T_{\text{CBF}} = 15$, $N_0 = m/4$, 由于 CBF 高的空间效率, 这里暂不考虑 CBF-SS 算法中 CBF 暂用的缓存, 令 CBF-SS 算法中 $n_{\text{ss}} = L$, 且以 CBF-SS 算法中 SS 模块估计流长加上 10 作

为 CBF-SS 算法流长估计值.

定义流长大于等于 1 000 报文的流为大流, 召回率 E_R 为算法正确检测出的大流数量与原始流量中大流数量之比, 定义 N_p 为误报的大流数目, 定义频数均差率 E_L 为算法正确检测出的大流的统计流长与原流长的绝对差和原流长之比的平均值.

图3 描述 $L = 10\,000 \sim 6\,000$ 时 3 组 traces 的大流召回率 E_R 的对比情况, 可以看出, 1) 对于不同的流量数据, LRU2 和 LLR 算法召回率波动较大, 2 种算法对于 CAIDA 流量的大流召回率仅在 8% ~ 22% 之间, 而 2 种算法对于 MAWI 流量的大流召回率较高, 说明 LRU2 和 LLR 2 种算法对于流量的特性比较敏感; 2) SS 算法的大流召回率较为稳定, 但由于 3 组 traces 中网络流数量均远大于缓存数量, 因此 SS 算法的召回率不能够达到较高的水平, 且 SS 算法的大流召回率随着缓存 L 的减小而不断降低. 相比而言, 本文提出的 CBF-SS 算法的大流召回率远高于其他 3 种算法, 对于 3 组数据, CBF-SS 算法在 $L = 10\,000$ 时的召回率接近 100%, 即使是在 $L = 6\,000$ 的情况下, CBF-SS 算法的大流召回率仍高于其他 3 种算法在 $L = 10\,000$ 时的召回率.

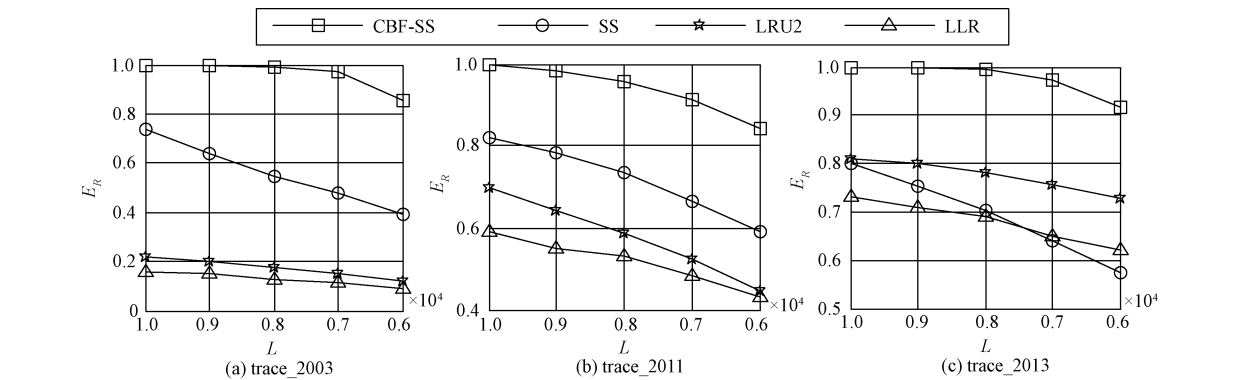


图3 4 种算法大流召回率对比

Fig.3 Comparison of E_R among four algorithms

图4 描述 $L = 10\,000 \sim 6\,000$ 时 3 组 traces 的频数均差率 E_L 的对比情况, 可以看出, 在各种情况下, CBF-SS 和 SS 算法的频数均差率远低于 LRU2 和 LLR 2 种算法, 说明 SS 算法用于网络流时以 $f(e) - \Delta(e)$ 作为流长估计值是合适的, 且 CBF-SS 和 SS 2 种算法能够比较准确地估计大流的流长.

由于 LRU2 和 LLR 算法在新流到来且需要保存时均是 将新流流长设置为 1, 随后 只在该流有报文到达时其流长才增加, 因此 LRU2 和 LLR 算法不存在误报大流的情况, 而 SS 算法以 $f(e) - \Delta(e)$ 作为流长估计值时同样不会存在误报大流的情况, 因此 SS、LRU2 和 LLR 3 种算法中误报的大流数目 N_p 等于 0. 在 CBF-SS 算法中, 为了补偿

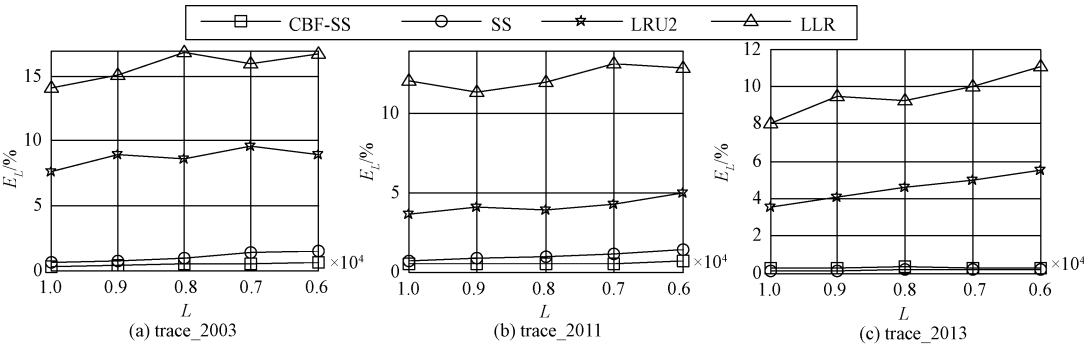


图 4 4 种算法频数均差率对比

Fig. 4 Comparison of E_L among four algorithms

CBF 过滤造成的影响,以 SS 模块估计流长加上 10 作为 CBF-SS 算法流长估计值,因此 CBF-SS 算法存在误报大流的可能性. 表 3 描述了 $L = 10\,000 \sim 6\,000$ 时 3 组 traces 中 N_p 的变化情况,可以看出, CBF-SS 算法误报的大流数量相对于总体大流数是非常小的. 尽管 CBF-SS 算法会引入少量的误报情况,但 CBF-SS 算法的大流召回率与 SS 算法相比具有明显提升,将 CBF 与 SS 算法相结合能够显著提高 SS 算法的大流识别效果.

表 4 进一步描述 CBF-SS 算法在缓存极其有限情况下的大流识别效果,可以看出, CBF-SS 算

表 3 CBF-SS 算法误报大流数					
Table 3	Numbers of false positive large flows in CBF-SS				
数据	10 000	9 000	8 000	7 000	6 000
trace_2013	4	8	8	6	2
trace_2011	6	5	6	5	4
trace_2003	10	10	9	7	6

法在极端情况下仍然具有较好的大流识别效果,对于部分流量数据, CBF-SS 算法在 $L = 3\,000$ 时的大流识别效果甚至要高于其他算法在 $L = 10\,000$ 时的大流识别效果.

表 4 极端情况下 CBF-SS 算法大流识别效果
Table 4 Performance of large flow identification for CBF-SS in extreme situations

	trace_2013			trace_2011			trace_2003		
	5 000	4 000	3 000	5 000	4 000	3 000	5 000	4 000	3 000
E_R	0.846	0.751	0.614	0.772	0.648	0.513	0.716	0.553	0.405
$E_L/\%$	0.3	0.28	0.31	0.85	0.98	1.15	0.72	0.89	1.15
N_p	1	1	0	1	0	0	5	4	3

综上所述, CBF-SS 算法能够有效结合散列方法和计数方法的优点. 算法利用散列函数不需要保存流信息的特性,使用极少量的缓存空间快速淘汰大量的小流,使得到达 SS 算法的流数量大为减少,降低了 SS 算法剪枝操作的次数;同时由于 SS 算法本身具有较为稳定的性能, SS 算法的不足之处在于需要较大的缓存空间,采用 CBF 和 SS 算法相结合正好可以弥补 SS 算法缓存空间不足的问题. 因此,对于 3 组不同的网络流量,本文提出的 CBF-SS 算法的大流识别效果要远优于 SS 等算法.

3 结束语

在实际网络应用中,网络流量分布具有明显的重尾效应,依据网络流量的这一特性,本文提出一种能够充分结合散列方法和计数方法优点的大流识别方法 CBF-SS. CBF-SS 算法先采用计数型布鲁姆过滤器淘汰掉网络中大量的小流,然后算法采用 SS 算法挖掘网络中的大流. 通过 3 组不同的骨干链路流量数据测试表明, CBF-SS 算法的大流识别效果远优于 SS 等算法,且 CBF-SS 算法具有 $O(1)$ 的时间复杂度及高效的空间利用率,算法能够有效应用于高速网络环境.

参考文献

- [1] Hyunsang C, Heejo L. Identifying botnets by capturing group activities in DNS traffic[J]. *Computer Networks*, 2012, 56(1): 20-33.
- [2] 周爱平, 程光, 郭晓军. 高速网络流量测量方法[J]. *软件学报*, 2014, 25(1): 135-153.
- [3] 张玉, 方滨兴, 张永铮. 高速网络监控中大流量对象的识别[J]. *中国科学: 信息科学*, 2010, 40(2): 340-355.
- [4] Estan C, Varghese G. New directions in traffic measurement and accounting: focusing on the elephants, ignoring the mice[J]. *ACM Transactions on Computer Systems*, 2003, 21(3): 270-313.
- [5] Manku G S, Motwani R. Approximate frequency counts over data streams[C]//Proc of the 28th International Conference on Very Large Data Bases, Hong Kong, 2002:346-357.
- [6] Cormode G, Muthukrishnan S. What's hot and what's not: tracking most frequent items dynamically [J]. *ACM Transactions on Database Systems*, 2005, 30(1): 249-278.
- [7] 张震, 汪斌强, 陈庶樵, 等. 基于多维计数型布鲁姆过滤器的大流检测机制[J]. *电子与信息学报*, 2010, 32(7): 1608-1613.
- [8] 吴桦, 龚俭, 杨望. 一种基于双重 Counter Bloom Filter 的长流识别方法[J]. *软件学报*, 2010, 21(5): 1115-1126.
- [9] 王风宇, 云晓春, 王晓峰, 等. 高速网络监控中大流量对象的提取[J]. *软件学报*, 2007, 18(12): 3060-3070.
- [10] 裴育杰, 王洪波, 程时端. 基于两级 LRU 机制的大流检测算法[J]. *电子学报*, 2009, 37(4): 684-691.
- [11] Karp R M, Shenker S, Papadimitriou C H. A simple algorithm for finding frequent elements in streams and bags[J]. *ACM Transactions on Database Systems*, 2003, 28(1): 51-55.
- [12] 夏靖波, 赵小欢, 柏骏, 等. 基于时间和流长约束的网络流频繁项挖掘算法[J]. *中国科学技术大学学报*, 2013, 43(10): 790-798.
- [13] Metwally A, Agrawal D, Abbadi A E. Efficient computation of frequent and Top-k elements in data streams[C]//Proc. of the International Conference on Data Theory. Edinburgh: Springer-Verlag, 2005:398-412.
- [14] 王风宇, 郭山清, 李亮雄, 等. 一种高效率的大流提取方法[J]. *计算机研究与发展*, 2013, 50(4): 731-740.
- [15] Cormode G, Hadjieleftheriou M. Finding the frequent items in streams of data[J]. *Communications of ACM*, 2009, 52(10): 97-105.
- [16] Liu H Y, Lin Y, Han J W. Methods for mining frequent items in data streams: an overview [J]. *Knowledge and Information System*, 2011, 26(1): 1-30.
- [17] 张震, 汪斌强, 张风雨, 等. 基于 LRU_BF 策略的网络流量测量算法[J]. *通信学报*, 2013, 34(1): 111-120.
- [18] 谢冬青, 周再红, 骆嘉伟. 基于 LRU 和 SCBF 的大象流提取及其在 DDoS 防御中的应用[J]. *计算机研究与发展*, 2011, 48(8): 1517-1523.
- [19] 赵小欢, 夏靖波, 付凯. 基于散列和计数方法的网络流频繁项挖掘算法[J]. *华中科技大学学报: 自然科学版*, 2013, 41(9): 57-62.
- [20] 孙昱, 夏靖波, 赵小欢, 等. 基于 LEAST 和 CBF 两级结构的大流检测算法[J]. *华中科技大学学报: 自然科学版*, 2014, 42(4): 40-44.
- [21] Fan L, Cao P, Almeida J, et al. Summary cache: a scalable wide-area web cache sharing protocol [J]. *IEEE/ACM Transactions on Networking*, 2000, 8(3): 281-293.
- [22] 周明中. 大规模网络 IP 流行为特性及其测量算法研究[D]. 南京: 东南大学, 2006.
- [23] Dainotti A, Pescapè A, Ventre G. A packet-level characterization of network traffic[C]//Proc of the 11th Int Workshop on CAMAD. Piscataway: IEEE, 2006: 38-45.
- [24] The Cooperative Association for Internet Data Analysis. The CAIDA anonymized OC48 Internet traces dataset[EB/OL]. (2012-11-10)[2014-03-01]. <http://www.caida.org/>.
- [25] MAWI Working Group. MAWI working group traffic archive[EB/OL]. (2013-04-26)[2014-03-01]. <http://mawi.wide.ad.jp/2011>.