基于数据到达间隔的网络大流检测方法*

赵　亮[1]，林　栎[2]

（1. 西南科技大学　信息工程学院，四川　绵阳　621010；2. 新疆电子研究所股份有限公司，新疆　乌鲁木齐　830010）

摘　要：大流检测是网络监管中的重要任务。现有的检测方法主要围绕流的大小来进行筛选评判。然而，真实的大流往往与数量众多的老鼠流混合在一起，尤其是当路由器存储空间有限时，依靠所能记录下来的流大小信息通常不能准确实现大流检测。为此提出一种新的大流检测方法，该方法根据流的新旧及数据到达间隔来滤除老鼠流。实验显示该方法在有限的存储空间条件下，表现出良好的检测准确性。

关键词：网络监管；大流检测；存储空间；数据到达间隔

中图分类号：TP393　　**文献标识码：**A　　**DOI：**10. 19358/j. issn. 2097 - 1788. 2024. 08. 007

引用格式：赵亮，林栎. 基于数据到达间隔的网络大流检测方法［J］. 网络安全与数据治理，2024，43(8)：40 - 43.

Elephant flow detection method based on data arrival interval

Zhao Liang[1]，Lin Li[2]

（1. School of Information Engineering，Southwest University of Science & Technology，Mianyang 621010，China；
2. Xinjiang Electronics Research Institute Company，Urumqi 830010，China）

Abstract：Detecting elephant flows is a critical task in network monitoring and management. The existing detection methods mainly focus on the size of the flows，where they filter the elephant flows by size. However，the real elephant flows in high-speed network tend to be swamped by a large number of mouse flows，especially under limited memory size，resulting in the algorithm not being able to correctly detect elephant flows by size. In this paper，we propose a novel method which separates the elephant flows from the mouse flows by expelling the oldest flow which also takes on big arrival interval of data. Experimental results show that the new method achieves good precision with limited memory size.

Key words：network monitoring and management；elephant flow detection；memory size；data arrival interval

0　引言

流量检测是网络领域进行拥塞控制、异常识别、负载均衡等工作的基础[1-3]。网络领域中具有相同流 ID 的数据包集合被称为流，其中流 ID 通常是数据包头部特定字段的组合，如源 IP、源端口、目的 IP、目的端口等，集合中数据包总数或字节总数对应流的大小。在现实网络中，流的大小通常服从重尾分布，即大多数流非常小，这类流又称为老鼠流，而一小部分流非常大，这类流通常称为大流或大象流[4-5]。相比之下，大流更容易造成网络堵塞和负载不均衡等问题，故对大流进行检测识别是网络监控的重要任务。

通常来说，现代网络为实现高速信息传输，所用的路由器主要使用 SRAM 之类具有低延迟特性的存储器。受 SRAM 的成本制约，路由器的内存往往有限，由此带来的问题便是硬件处理速度难以满足高速网络流量测量的需要[6]。这种情况下，如何在有限的硬件条件下完成大流检测成为了网络领域的研究热点。针对这一需求，目前已发展出不同类型的大流检测方法，这些检测方法从策略上大致可分为三类：count-all 策略、admit-all-count-some 策略和其他策略。其中 count-all 策略基于 sketch 计算所有流的大小[7]，这类方法更适用于均匀分布的数据，当网络流量不均匀时，其大流检测结果中会含有大量的假阳性。admit-all-count-some 策略将所有新流都视为大流，并只记录部分流的信息以节省内存空间[8-11]，

*基金项目：四川省产教融合示范项目（23cjkc24）；中国科技城网络应急管理研究中心项目（WLYJGL2023YB07）；自治区科技支疆项目（2022E02080）

www.pcachina.com

这类方法存在的问题在于对真正大流的漏检率偏高。其他具有代表性的大流检测策略包括：Cold Filter 方法采用双层 sketch 结构对网络流进行过滤，结合 Space－Saving 算法记录大流[12]，但其过早对流的类型做出判断，将老鼠流误识为大流；HeavyKeeper 方法使用指数衰减的方式来剔除老鼠流并记录大流[13]，但其识别某些老鼠流的用时较长，由此可能导致后续的大流无法被记录下来；ActiveKeeper 方法[14]则是在 HeavyKeeper 基础上引入双模式计数器，通过不同的计数模式来分别对大流和老鼠流进行计数，以此提升内存使用效率。

针对现有方法的局限性，本文提出一种新的大流检测策略。该策略基于网络流量在大小、到达时间间隔方面的统计特性，在有限的内存空间中重点记录新的、持续到达的流，并及时抛弃旧流为记录新流腾出空间。实验表明，本文方法可以在有限的片上内存条件下实现高精度和高吞吐量，由此可为网络监控提供及时准确的决策支持。

1　基于数据到达间隔的大流检测方法

1.1　设计思路

考虑到在一个网络测量周期中，大流的大小总是远大于老鼠流，属于同一个大流的数据包应呈现出更高的集中度，即大流的平均数据包到达间隔应小于老鼠流。故在进行大流检测时，除了从流的大小入手进行区分之外，还可以根据数据到来的间隔进行识别。具体而言，在内存中，当接收的流的数据包大小还不足以判断是否为大流时，若一个流的数据包以更小的时间间隔不断到达，则根据网络流量中有关数据到达间隔的统计特性，可认为该流发展为大流的概率更大，这种情况下将该流在存储空间中保留并做持续记录。当存储空间不足以插入新数据包时，抛弃到达间隔最大的流，而不是当前最小的流。这样一来，被保留下来的新流都是最可能成长为大流的，由此使大流检测过程具有更高的时间与空间效率。

1.2　算法设计

如图 1 所示，本文大流检测方法的数据结构由一个哈希表 H 构成，对应的哈希函数记为 $h(\cdot)$。哈希表 H 包含 L 个桶，每个桶被分为了 E-flows 和 U-flows 两部分，每部分都包含了多个存储单元，每个单元分别记录一个流的 ID、流大小及当前数据到达间隔特征值。其中 E-flows 部分所包含的 l_1 个单元用于记录已经确认的大流，U-flows 部分所包含的 l_2 个单元用于接受新流，并舍弃不具备大流特征的流，为后续新流腾出空间。

考虑网络流由一系列数据包 $P=p_1, p_2, \cdots, p_N$ 组成，具有相同 ID 的数据包 p_i（$1\leqslant i\leqslant N$）属于同一流 f_j。围绕图 1 所示数据结构的具体操作策略包括下述插入与更新两类。

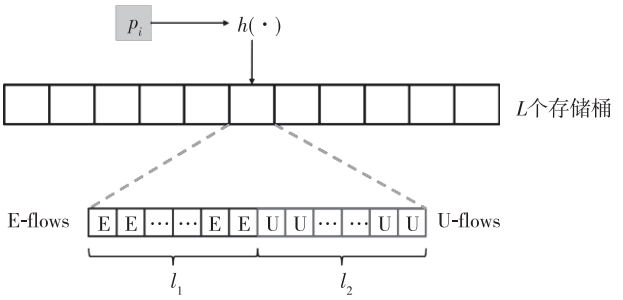

图 1　大流检测的数据结构

1.2.1　新流插入

检测开始时，哈希表中所有字段都置 0。对于一个流 ID 属于 f_j 的数据包 p_i，通过哈希函数 $h(f_j)$ 将 p_i 映射到桶 $H[h(f_j)\%L]$，并首先尝试将 p_i 插入到 E-flows 部分，具体情况包含以下三种：

（1）若流 f_j 已被记录在 E-flows 的一个单元里，则将相应单元的流大小字段加 1，该单元的当前数据到达间隔特征值字段清零，E-flows 中所记录的其他流的当前数据到达间隔特征值字段加 1；

（2）若流 f_j 还未被记录在 E-flows 中，且 E-flows 仍有空闲单元，则将 f_j 插入 E-flows 的空闲单元，令该单元中 f_j 的流大小字段为 1，同时清零当前数据到达间隔特征值，E-flows 中所记录的其他流的当前数据到达间隔特征值字段加 1；

（3）若流 f_j 还未被记录在 E-flows 中，且 E-flows 没有空闲单元，则流 f_j 无法插入 E-flows 中，此时将流 f_j 视为一个新流并插入到该桶的 U-flows 中。

U-flows 接收所有新流，并根据各流的数据到达间隔来从庞大的流量中剔除老鼠流，保留大流。其在进行新流插入时，考虑以下三种情况：

（1）若流 f_j 已被记录在 U-flows 的一个单元里，则将相应单元的流大小字段加 1，将该单元的当前数据到达间隔特征值字段清零，并将 U-flows 中所记录的其他流的当前数据到达间隔特征值字段加 1；

（2）若流 f_j 还未被记录在 U-flows 中，且 U-flows 中仍有空闲单元，则将 f_j 插入 U-flows 的空闲单元，将该单元中 f_j 的流大小字段设为 1，该流的当前数据到达间隔特征值设为 0，并将 U-flows 中所记录的其他流的当前数据到达间隔特征值字段加 1；

（3）若流 f_j 还未被记录在 U-flows 中，且 U-flows 没有空闲单元，则此时将数据到达间隔特征值字段最大的流视为老鼠流丢弃，腾出的空间用于记录流 f_j。

www.cachina.com

1.2.2 大流更新

每当流 f_j 的数据包 p_i 被插入 U-flows 后，检查 f_j 的流大小字段数值是否大于 E-flows 中流大小字段数值最小的流 $F_{\text{E-min}}$。如果 E-flows 中存在多个流具有同样最小的流大小，或者流大小排在最后的几个流之间，流大小差距小于设定阈值（例如将阈值设为 30），则进一步比较 E-flows 中最小的几个流的数据到达间隔，选取其中具有最小数据到达间隔的流作为 $F_{\text{E-min}}$。若 f_j 的流大小字段数值大于 E-flows 中的 $F_{\text{E-min}}$，则使用 f_j 更新 $F_{\text{E-min}}$，f_j 被记录到 E-flows，原来的 $F_{\text{E-min}}$ 被重新插入 U-flows 中。

2 性能分析

2.1 时间复杂度

本文方法的时间复杂度主要由哈希次数和内存访问次数及范围决定。单次数据插入通常需要计算一个哈希函数，其时间复杂度为 $O(H)$。除此以外还需要最多遍历 E-flows 的 l_1 个单元 1 次，遍历 U-flows 的 l_2 个单元 1 次。则插入数据包 p_i 的时间复杂度为 $O(H+l_1+l_2)$。该时间复杂度受内存空间及对内存单元进行计数操作的影响，相较于其他算法[7,13]，本文中方法具有更少的计算哈希函数的时间复杂度。

2.2 空间复杂度

本文方法的空间复杂度取决于哈希表的大小。当哈希表长度为 L 时，方法所需空间为 $S=L\times l_1\times B+L\times l_2\times B$ 个字节，其中 B 代表 E-flows 和 U-flows 中每个单元所占字节数。当接收到 N 个数据包时，该方法占用的空间始终为 S 个字节，即空间复杂度为 $O(S)$，且 S 通常远小于 N。

3 实验

实验在 Ubuntu 环境下采用 CAIDA 网络流量数据集[15]测试本文方法，在测量时段内，设占总流量百分比 λ 以上的流为大流，λ 分别取 0.01、0.05、0.1 三个值。采用式（1）、式（2）所示误检率指标 F_{PR} 与漏检率指标 F_{NR} 比较本文方法与 CM Sketch[12]、HeavyKeeper[13]方法在准确性方面的差异。其中本文方法使用一个哈希函数，在大流更新时设定流大小的筛选阈值为 25，即将排在最后且大小差距小于 25 的流进行数据到达间隔的比较。CM Sketch 方法中数组个数取 6，分别对应 6 个哈希函数。HeavyKeeper 方法中数组个数设为 2，对应 2 个哈希函数，指数衰减值取 1.06。

$$F_{\text{PR}}=\frac{F_{\text{P}}}{F_{\text{P}}+T_{\text{N}}} \tag{1}$$

$$F_{\text{NR}}=\frac{F_{\text{N}}}{F_{\text{N}}+T_{\text{P}}} \tag{2}$$

式（1）中，F_{P} 为被误检测为大流的老鼠流数量，T_{N} 为被正确抛弃的老鼠流数量。式（2）中，F_{N} 是被当作老鼠流抛弃的大流数量，T_{P} 是被正确检出的大流数量。除此之外，式（3）所示相对误差 E_{R} 是指检测出的大流大小与实际大流大小的相对误差之和的平均值。其中 k 为检测出的大流总数，$\hat{n}_j$ 为检测所得大流 f_j 的大小，n_j 为流 f_j 的实际大小。

$$E_{\text{R}}=\frac{1}{k}\sum_{j=1}^{k}\frac{|n_j-\hat{n}_j|}{n_j} \tag{3}$$

式（4）所示吞吐量指标 T 用于比较几种方法在处理速度方面的差异：

$$T=\frac{N}{t} \tag{4}$$

其中 N 为数据包总数，t 为处理数据包所用总时间，吞吐量 T 的单位为 Mp/s（Million Packet Per Second）。

考察 U-flows 中的单元格数量 l_2 对本文方法的相对误差及吞吐量的影响。图 2 所示为 l_2 取值从 2 至 10 变化时，相对误差及吞吐量的变化趋势。

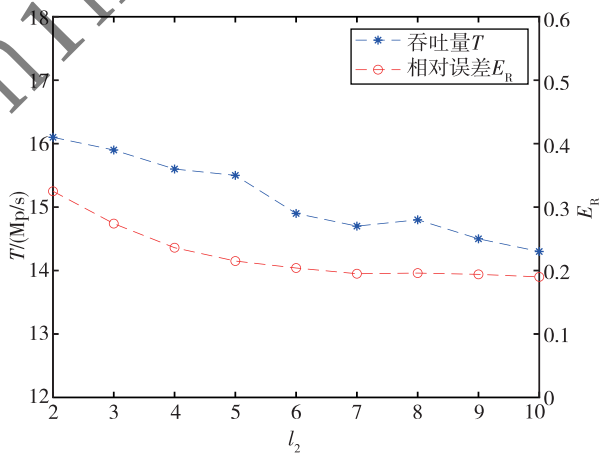

图 2 相对误差及吞吐量受单元格数量 l_2 的影响

由图 2 可见，本文方法在进行大流检测时的相对误差 E_{R} 随 l_2 增大而减小，但吞吐量 T 也随之下降。其原因在于检测过程中每个流在被丢弃前需要经过 l_2 个不同流的数据包，l_2 增大会导致存储空间访问次数的相应增加，由此降低吞吐量。图 2 显示当 l_2 大于 6 以后，相对误差的减小速率放缓。在图 2 所示的实验过程中，被本文方法选中作为 E-flows 中 $F_{\text{E-min}}$ 流的数据到达间隔在 4 ~ 83 之间波动，显然该值受实验数据的影响具有较大的分布范围。综合考虑检测误差、吞吐量与存储资源消耗的关系，在本文后续实验中将 l_2 的值设为 6，即每个桶的 U-flows 中包含 6 个单元格。

表 1 所示为本文方法与 CM Sketch 方法、HeavyKeeper

www.pcachina.com

方法在误检率 F_{PR} 与漏检率 F_{NR} 方面的差异。

表 1　算法错误率比较　　(%)

大流认定阈值 λ	错误指标	检测方法		
		本文方法	CM Sketch	HeavyKeeper
0.01	F_{PR}	1.82	3.21	1.72
	F_{NR}	2.41	2.19	3.22
0.05	F_{PR}	1.25	2.75	1.24
	F_{NR}	1.87	1.58	2.31
0.1	F_{PR}	1.13	2.17	1.08
	F_{NR}	1.45	1.38	1.93

表 1 中数据显示，CM Sketch 方法与 HeavyKeeper 方法在误检率和漏检率方面各有侧重。而本文方法在误检率方面与 HeavyKeeper 方法接近，显著低于 CM Sketch 方法；在漏检率方面本文方法较 HeavyKeeper 方法更低，高于 CM Sketch 方法。总体而言，本文方法在误检率和漏检率方面取得了较好的平衡。同时，三种方法的错误率均随着阈值 λ 的增大而减小，这显示网络中流的大小越大，越容易从其他老鼠流中被分辨出来，在大流检测中出错的概率也就越低。

4　结论

针对当前网络监控在进行大流检测时所存在的路由存储资源与检测准确性相互矛盾的问题，基于网络流量的统计特性，提出了一种根据流的新旧及数据到达间隔来进行筛选的大流检测方法。对所提的大流检测方法从时间和空间维度进行了复杂性分析。实验结果表明，在有限的存储开销下，所提方法具有较为平衡的大流误检率与漏检率。

参考文献

[1] LEI K, LIANG Y Z, LI W. Congestion control in SDN-based networks via multi-task deep reinforcement learning [J]. IEEE Network, 2020, 34 (4): 28 – 34.

[2] RAJENDRAN S, MEERT W, LENDERS V, et al. Unsupervised wireless spectrum anomaly detection with interpretable features [J]. IEEE Transactions on Cognitive Communications and Networking, 2019, 5 (3): 637 – 647.

[3] XIE S X, HU G Y, XING C Y, et al. Online elephant flow prediction for load balancing in programmalbe switch-fased DCN [J]. IEEE Transactions on Network and Service Management, 2024, 21 (1): 745 – 758.

[4] FONTUGNE R. Scaling in Internet traffic: a 14 year and 3 day longitudinal study, with multiscale analyses and random projections [J]. IEEE/ACM Transactions on Networking, 2017, 25 (4): 2152 – 2165.

[5] 杜鑫乐，徐恪，李彤，等. 数据中心网络的流量控制：研究现状与趋势 [J]. 计算机学报，2021，44 (7)：1287 – 1390.

[6] KHOOI X Z, CSIKOR L, LI J, et al. Revisiting heavy-hitter dectection on commodity programmable switches [C]//2021 IEEE 7th International Conference on Network Softwarization (NetSoft). Tokyo: IEEE, 2021: 79 – 87.

[7] CORMODE G, MUTHUKRISHNAN S. An improved data stream summary: the count-min sketch and its applications [J]. Journal of Algorithms, 2005, 5 (1): 58 – 75.

[8] DEMAINE E, ALEJANDRO L, MUNRO J. Frequency estimation of Internet packet streams with limited space [C]//Algorithms-ESA2002. Rome: University of Waterloo, 2002: 11 – 20.

[9] GURMEET S M, RAJEEV M. Approximate frequency counts over data streams [C]//Proceedings of the VLDB Endowment. Istanbul: VLDB, 2012: 346 – 357.

[10] METWALLY A, AGRAWAL D, ABBADI A E. Efficient computation of frequent and top-k elements in data streams [C]//Database Theory-ICDT 2005. Edinburgh: Springer, 2005: 398 – 412.

[11] BEN – BASAT R, EINZIGER G, FRIEDMAN R, et al. Heavy hitters in streams and sliding windows [C]//The 35th Annual IEEE International Conference on Computer Communications. San Francisco: IEEE, 2016: 1 – 9.

[12] ZHOU Y, YANG T, JIANG J, et al. Cold filter: a meta-framework for faster and more accurate stream processing [C]//ACM SIGMOD Conference 2018. Houston: ACM, 2018: 741 – 756.

[13] GONG J Z, YANG T, ZHANG H W, et al. Heavykeeper: an accurate algorithm for finding top-k elephant flows [J]. IEEE/ACM Trans. Netw., 2019, 27 (5): 1845 – 1858.

[14] WU M K, HUANG H, SUM Y E, et al. Activekeeper: an accurate and efficient algorithm for finding top-k elephant flows [J]. IEEE Communications Letters, 2021, 25 (8): 2545 – 2549.

[15] WANG J, LIU W, KUMAR S, et al. Learning to hash for indexing big data: a survey [J]. Proceedings of the IEEE, 2016, 104 (1): 34 – 57.

（收稿日期：2024 – 06 – 20）

作者简介：

赵亮（1983 – ），男，博士，高级工程师，主要研究方向：通信系统。

林栎（1981 – ），通信作者，男，硕士，高级工程师，主要研究方向：软件工程。E-mail：lonlystring@163.com。

www.pcachina.com

版权声明

凡《网络安全与数据治理》录用的文章，如作者没有关于汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权等版权的特殊声明，即视作该文章署名作者同意将该文章的汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权授予本刊，本刊有权授权本刊合作数据库、合作媒体等合作伙伴使用。同时，本刊支付的稿酬已包含上述使用的费用，特此声明。

《网络安全与数据治理》编辑部

www.pcachina.com