计算机科学
COMPUTER SCIENCE

# ACCF:时间预测机制驱动的top-*k*流测量

胡永庆, 杨含, 刘子源, 秦广军, 戴庆龙

引用本文

胡永庆, 杨含, 刘子源, 秦广军, 戴庆龙. ACCF:时间预测机制驱动的top-*k*流测量[J]. 计算机科学, 2025, 52(10): 98-105.

HU Yongqing, YANG Han, LIU Ziyuan, QING Guangjun, DAI Qinglong. ACCF:Time Prediction Mechanism-driven Top-*k* Flow Measurement[J]. Computer Science, 2025, 52(10): 98-105.

---

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.241000033

# ACCF:时间预测机制驱动的 top-$k$ 流测量

胡永庆　杨　含　刘子源　秦广军　戴庆龙
北京联合大学智慧城市学院　北京 100101
(huyongqing2021@163.com)

**摘　要**　针对当前 top-$k$ 流测量过滤算法依赖固定计数器阈值的问题,提出了基于活跃度预测机制的 ACCF(Activity Counting Cuckoo Filter)测量结构。ACCF 通过引入活跃度预测机制,利用时间序列分析和指数加权移动平均(Exponentially Weighted Moving Average,EWMA)机制,动态计算网络流的活跃度,实现对潜在的 top-$k$ 流的实时识别与提前过滤。针对哈希冲突可能导致的精度损失,ACCF 引入了自刷新存储表(Self-Refreshing Storage Table,SRST),用于存储踢出路径上的网络流信息。当踢出操作达到设定的 MaxNumKicks 值时,SRST 会在局部范围内优先踢出活跃度最小的网络流项,避免重要流量信息丢失。实验结果证明,ACCF 与 SRST 在合适的参数组合条件下,可以提前过滤 65%左右的大流并减少 41%左右的插入操作,并显著提升了在 top-$k$ 流量测量中的精度,尤其是在与传统的 Space Saving(SS),CM Sketch,LUSketch 和 Cuckoo Counter 算法对比时,展现了明显的优势。

**关键词**:top-$k$;活跃度;时间序列;EWMA;SRST;Sketch
**中图分类号**　TP393

## ACCF:Time Prediction Mechanism-driven Top-$k$ Flow Measurement

HU Yongqing,YANG Han,LIU Ziyuan,QING Guangjun and DAI Qinglong
Smart City College of Beijing Union University,Beijing 100101,China

**Abstract**　In addressing the problem that current top-$k$ flow measurement filtering algorithms depend on fixed counter thresholds, a measurement structure named ACCF based on the activity prediction mechanism has been put forward. ACCF incorporates the activity prediction mechanism, utilizing time series analysis and the EWMA mechanism, to dynamically compute the activity of network flows and accomplish real-time identification and early filtering of potential top-$k$ flows. With respect to the accuracy loss that may be induced by hash conflicts, ACCF introduces a SRST for storing the network flow information on the evicted paths. When the eviction operation reaches the predefined MaxNumKicks value, SRST will give priority to evicting the network flow item with the lowest activity within the local scope to avoid the loss of crucial traffic information. Experimental results indicate that, under suitable parameter combinations, ACCF and SRST can filter out approximately 65% of the major flows in advance and reduce insertion operations by approximately 41%, significantly improving the accuracy in top-$k$ traffic measurement, especially when compared with traditional algorithms such as Space Saving (SS), CM Sketch, LUSketch, and Cuckoo Counter, thereby demonstrating distinct advantages.

**Keywords**　Top-$k$, Activity, Time series, EWMA, SRST, Sketch

## 1　引言

精确识别 top-$k$ 流是网络管理中的核心任务,其测量结果为负载均衡[1-2]、容量规划[3-4]、流量工程[5-6]、拥塞控制[7]以及异常行为检测[8]等多种网络任务提供了重要的数据支持。在流量测量中,网络流通常被定义为具有相同流标识(如源 IP、目的 IP、协议、源端口号和目的端口号)的数据包序列,并以数据包数量来衡量其大小。因此,top-$k$ 流测量就是从网络流中寻找大小排名前 $k$ 位的网络流。

现有研究表明,网络流量的大小分布呈现高度的倾斜性[9-10],即少数网络流占据大部分流量(称为大象流),而绝大多数网络流仅占很小一部分流量(称为老鼠流),尽管其数量远远多于大象流。随着网络规模的持续扩大,网络流量呈现出高速、大量和不可预测的特点,给网络管理与

到稿日期:2024-10-09　返修日期:2025-02-03
基金项目:面向海量事件流的流式计算框架的研究(CCFIS2019-01-01);智慧交通中的光与无线融合接入网组网技术研究(KM202111417010);拓扑结构引导的深度学习优化技术研究(ZK10202403)
This work was supported by the Research on Stream Computing Frameworks for Massive Event Streams(CCFIS2019-01-01),Research on Optical and Wireless Converged Access Network Technologies in Smart Transportation Systems(KM202111417010)and Research on Topology-Guided Deep Learning Optimization Techniques(ZK10202403).
通信作者:戴庆龙(xxtqinglong@buu.edu.cn)

维护带来了诸多挑战。

传统测量方法往往依赖于昂贵的硬件资源，如 TCAM (Ternary Content Addressable Memory)和片上内存 SRAM (Static Random-Access Memory)[11-12]，难以在大规模网络中广泛部署。目前的诸多研究引入了计数器[13-14]和 Sketch[15-16]技术，并提出了两种策略来实现 top-$k$ 流测量：1)count-all 策略(如 Count-Min Sketch[17]等)，使用基于计数器的 Sketch 统计所有网络流量，提供每个流的近似估计，并通过维护一个最小堆来跟踪 top-$k$ 流；2)admit-all-count-some 策略(如 LossyCounting(LC)[18]和 Space-Saving(SS)[19]等)，使用以键值对(〈流标识，流大小〉)为基础的计数器数组进行流量统计，并通过替换策略采样大流而驱逐小流。

近年来，一些研究将上述两种策略结合，通过过滤机制实现了大小流的分离测量。例如，Roy 等[20]提出的 Augmented Sketch(ASketch)，通过预过滤队列与 Sketch 结合，将大流存储于过滤队列，实现了大小流分离统计。ColdFilter(CF)[21]则利用由两个计数器阈值队列组成的过滤器筛选大流，并结合 Space-Saving 等算法实现 top-$k$ 流测量。然而，这类过滤机制通常依赖固定的计数器阈值，难以适应动态的网络环境。此外，阈值选择不当会直接影响过滤性能：阈值过高会导致频繁插入操作，阈值过低则会误报小流，特别是在高流量和突发流量场景中，这些问题更加突出。

为了解决上述问题，本文提出了一个新型算法—活跃度计数布谷过滤器(ACCF)，这是一种基于活跃度阈值的过滤器算法，其关键技术是将数据包序列根据其携带的时间戳抽象为时间序列，并通过分析时间序列来预测网络流的未来增长趋势，实现对大流的提前过滤。本文的创新点和贡献如下：

1)通过量化每个数据包对网络流活跃度的贡献，得到一组时间上连续的权重序列，并结合 EWMA 机制，实时计算网络流活跃度，实现对大流的快速识别和提前过滤。

2)为 ACCF 引入一个额外的自刷新存储表 SRST，用于处理插入过程中发生的哈希冲突。

3)使用 CAIDA 的隐私数据集对 ACCF 在大流过滤和 top-$k$ 流测量上的性能进行了验证，实验证明基于活跃度预测机制的 ACCF 在大流检测中能够提前过滤 63%左右的大流，且在过滤精度方面达到 99%以上，表现出良好的性能。

本文第 2 章介绍了本课题的研究背景和相关工作；第 3 章介绍了 ACCF 的数据结构及核心算法；第 4 章对 ACCF 的性能进行了评估；最后对全文进行总结和讨论。

## 2 背景和相关工作

本章对 top-$k$ 流测量问题进行了陈述并回顾了学术界在 top-$k$ 流测量方面进行的诸多研究。

### 2.1 问题陈述

给定时间窗口内，在测量节点统计到包含 $m$ 个数据包的网络流量数据$\{\mathbb{P}_1,\mathbb{P}_2,\cdots,\mathbb{P}_m\}(m>0)$，记为 $P$。根据每个数据包$\mathbb{P}_l(0<l\leqslant m)$携带的流标签(使用五元组作为流标签)将网络流量划分为 $n$ 个网络流 $F=\{f_1,f_2,\cdots,f_n\}(n>0)$，流大小为 $S=\{s_1,s_2,\cdots,s_n\}$。top-$k$ 流测量描述如下：

$$F_k'=\underset{F}{\operatorname{argmax}}\sum_{f_i'\in F_k'}s_i'(F_k'\in F,|F_k'|=k) \tag{1}$$

其中，$F_k'=\{f_1',f_2',\cdots,f_k'\}$为 top-$k$ 测量结果，$s_i'$为网络流 $f_i'(f_i'\in F)$的测量大小，$|F_k'|$表示集合$F_k'$的大小为 $k$。

### 2.2 相关工作

在 top-$k$ 流测量方面，目前的研究可以分为以下 3 类：

1)count-all 策略使用基于计数器的 Sketch 结构(如 CM Sketch，CountMax Sketch[22]和 LUSketch[23])来测量每个流的大小，并通过数据格式为〈流标签，流大小〉的最小堆来跟踪记录 top-$k$ 流。典型的 CM Sketch 由 $d$ 个长度为 $w$ 的计数器数组组成，并分别与一个哈希函数关联。当数据包到达时，通过其流标签计算出对应的计数器位置，并增加该位置的计数器值，随后根据更新后的统计值调整最小堆。然而，Sketch 策略存在固有缺陷：哈希冲突导致无法区分大小流，且为所有流分配相同大小的计数器，内存利用效率不高。

2)admit-all-count-some 策略接受全部网络流，但只采样部分流信息。典型算法包括 Lossy Counting 和 Spacing Saving 等。以 Space Saving 算法为例，它维护一个固定大小的计数器数组 Stream-Summary。当数组满时，Space Saving 算法会用新流替换当前计数最小的网络流〈flowID，$n_{\min}$〉，并将其计数值设置为$n_{\min}+1$。这种最小计数替换策略优先保留大流，但由于流量分布的倾斜性，可能会高估大流。为了解决这一问题，后续研究提出了多种改进机制，以尽可能保留大流信息[24-25]。例如，HeavyKeeper[26]算法引入指数衰减机制，通过更频繁地衰减小流的计数器，使大流更有可能被保留。HeavyGuardian[27]算法则使用指数衰减机制设计了两个统计部分：heavypart 使用次级候选桶(护卫桶)降低国王桶(计数最大的桶)被衰减的次数；lightpart 用于记录和管理小流。尽管这些改进机制降低了小流对大流的影响，但在识别较晚出现的大流时仍面临挑战。

3)过滤思想，分离大小流。目前，过滤思想在分离大小流方面得到了广泛关注，例如 Augment Sketch，ColdFilter，LogLogFilter(LLF)[28]和 LadderFilter[29]等。ASketch 在传统 Sketch 结构前增加一个预过滤队列。当队列满时，通过与 Sketch 的双向通信将大流从 Sketch 中分离并插入队列中，然而，这种双向通信和队列扫描带来了较高的资源开销。与之不同，CF 使用两层尺寸不同的 CU Sketch 作为单向过滤结构，过滤小流并记录超过阈值的大流，但频繁的过滤操作会产生多次哈希和内存访问操作，资源开销较大。LLF 用 LogLog 结构代替 CF 中的 CU Sketch 结构，扩大了过滤范围，但在高性能场景下，LLF 在内存和时间效率方面存在一定不足。LadderFilter 通过实验观察到在当前时间窗口内，无希望小流(频率小于 5)发展成为有希望大流(频率大于 5)的概率较低，因此设计了多级 LRU 桶队列过滤器。当桶满时，优先移除时间上最近最少使用的键值对，从而实现对小流的过滤。然而，这种过滤器方案通常需要与已有的 Sketch 结合使用，存储需求更高。

## 3 ACCF 的设计

本章将详细介绍基于活跃度的网络流预测推荐机制，并深入解析 ACCF 的数据结构和操作原理。ACCF 通过网络流活跃度的动态计算，并结合过滤器的高效存储和查询特性，实

时评估网络流的增长趋势，并准确预测和推荐未来可能成为 top-$k$ 流的网络流。

### 3.1 基本原理

ACCF 的设计目标是精确预测和过滤出期望 top-$k$ 流，即识别并筛选出可能发展为大流的网络流。然而，现有的大多数测量方案侧重于网络流标识及简单的数据包计数，难以有效预测网络流趋势。由于在实际网络中，数据包通常会携带一个时间戳（发送或捕获时间戳），因此网络流的数据包序列天然可以抽象为一段时间序列。基于此，提出了网络流活跃度预测机制，通过实时计算网络流的活跃度，判定该网络流是否可能成为期望 top-$k$ 流。

为了更好地处理这些时间序列数据，本文引入了指数加权移动平均（EWMA）机制。EWMA 能够平滑数据波动，捕捉网络流的变化趋势，以此预测网络流是否可能成为期望 top-$k$ 流。将该机制同 Cuckoo Filter 结合，并针对其随机踢出机制的潜在误差进行了优化，提出了 ACCF 算法，用于实现对期望 top-$k$ 流的准确预测和过滤。

### 3.2 数据包权重

在将网络流的数据包序列抽象为时间序列后，需要对其进行建模，为每个时间点赋予一个观测值，以量化每个数据包对网络流活跃度的贡献。基于 LadderFilter 的实验验证和设计原则，可以合理地假设，一旦网络流开始被统计记录，其后续数据包对该流的活跃度贡献将呈现逐步递减的趋势。

为了更加精确地量化每个数据包对网络流活跃度的实际影响，本文引入了一个时间衰减因子 $\lambda$ 进行建模，利用时间衰减函数计算每个数据包权重的具体值。在该模型中，网络流的第一个数据包的时间戳被设定为该网络流的基准时间 $t$，其权重设定为 1。后续每个数据包的权重将根据其携带的时间戳与基准时间之间的时间间隔来计算，其权重计算式为：

$$w_i = e^{-\lambda \Delta t} \tag{2}$$

其中，$w_i$ 为第 $i+1$ 个数据包计算的权重值；$\lambda$（$\lambda \in (0,1)$，如 $\lambda=0.1$）为预定义的时间衰减因子，用于控制数据包权重的衰减速率，其值越大，数据包权重衰减越快；$\Delta t$ 为当前到达数据包携带的时间戳 $t_i$ 与基准时间 $t$ 的差值，即 $\Delta t = t_i - t$。

### 3.3 时间序列预测算法

将网络流的活跃度定义为衡量网络流发展成为 top-$k$ 流的潜力。通过网络流量进行时间序列建模，获得了一组时间上连续的权重观测值序列，表示为 $W=[1, w_1, w_2, w_3, \cdots, w_i]$（该被观测网络流的大小为 $i+1, i>0$）。为预测网络流未来是否可能成为流量最大的前 $k$ 个流，本文引入 EWMA 机制对这些时间序列数据进行平滑处理与动态分析处理。

EWMA 是一种常用的时间序列平滑技术，通过赋予最新数据点更高的权重，增强对流量变化的敏感性，同时保留历史数据的趋势。该机制通过加权平均计算，使较新的数据对结果的影响更大，而较旧的数据影响逐渐衰减，从而能够快速响应网络流的变化。基于这种特性，将其应用于网络流活跃度预测，通过对时间序列数据进行加权平滑处理，捕捉网络流在不同时间点的动态趋势。网络流活跃度的计算式如下：

$$\mathbb{A}_i = (1-\alpha) w_i + \alpha\, \mathbb{A}_{i-1} \tag{3}$$

其中，$\mathbb{A}_i$ 表示第 $i$ 个数据包到达时的网络流活跃度；$\alpha$ 是 EWMA 的平滑系数，取值范围为 $[0,1]$，用于控制当前与历史活跃度的平衡，较小的 $\alpha$ 值意味着对当前数据的敏感度较高，而较大的 $\alpha$ 则使得历史数据占据更大的权重；$\mathbb{A}_{i-1}$ 是第 $i-1$ 个数据包到达时网络流的活跃度。通过活跃度计算和时间序列预测算法，能够准确捕捉网络流的实时变化和长期趋势，从而预测哪些网络流将在未来成为 top-$k$ 流。

### 3.4 ACCF 数据结构

ACCF 是基于 Cuckoo Filter 改进的过滤结构，二者数据结构相似，均是由一个长为 $d$ 的桶数组组成的哈希表，其中每个桶包含 $b$ 个插槽（Slot），最多容纳 $b$ 条网络流信息，如图 1 所示。ACCF 的条目（Entry）由指纹、计数器、时间戳、活跃度和 flag 字段组成，分别表示该网络流标识的 $n$ 位哈希比特序列、数据包计数、首次到达时间戳、活跃度值和期望 top-$k$ 流标记。

ACCF 沿用了 Cuckoo Filter 的索引计算规则，对于属于网络流 $f$ 的数据包 $\mathbb{P}_f$，ACCF 通过 $h_1(f)=hash(f)$ 和 $h_2(f)=h_1(f) \oplus hash(f's fingerprint)$ 计算两个候选桶索引，以在 ACCF 哈希表中准确插入网络流。$h_2(f)$ 使用了基于部分键布谷鸟哈希技术的异或运算，使得 ACCF 只需要指纹和任一索引桶位置便可完成重定位操作 $h_{another} = h_{current} \oplus hash(f's fingerprint)$。

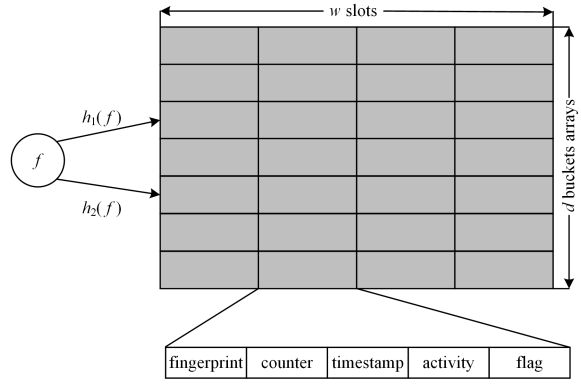



图 1 ACCF 的数据结构

Fig. 1 Data structure of ACCF

由于哈希计算的引入，ACCF 不可避免地会出现哈希冲突。为了解决位置冲突，ACCF 采用了最小活跃度踢出策略（Minimum Activity Removal Strategy，MARS）。不同于 Cuckoo Filter 的随机踢出机制（Random Kick Out Mechanism，RKOM），当插入网络流项的两个候选桶都满时，ACCF 踢出选定候选桶中活跃度最小的网络流项，为新网络流项腾出插入空间，被踢出的网络流项会被重定位到另一个候选桶。重定位操作会持续进行，直到插入成功或重定位次数达到预设的 MaxNumKicks。

然而，在高负载或高冲突情况下，采用 MARS 的 ACCF 会由于频繁的重定位操作显著降低插入效率。此外，当重定位次数达到预设值后，ACCF 将丢弃重定位桶中活跃度最小的网络流项，但未必是踢出路径上全局活跃度最小的网络流项，这会影响过滤准确性。为了优化该策略，ACCF 引入

SRST 表，用于记录踢出路径上的网络流项。当插入失败时，ACCF 会先将被踢出的网络流项记录到 SRST 中，然后尝试重定位该网络流，直到插入成功或达到设定最大重定位次数。若达到重定位上限，ACCF 选择踢出 SRST 中活跃度最小的网络流项，并回写其余网络流项。每次插入操作结束后，SRST 进行刷新操作，为下次插入操作腾出内存空间。

### 3.5 ACCF 操作

ACCF 提供插入和查询两种操作，并在插入过程中执行过滤功能，下面将以网络流 $f_i$ 的数据包 $\mathbb{P}_f$ 为例，详细介绍 ACCF 的两种操作流程。为了简化表示，使用 $b[i][j]$ 表示 ACCF 中第 $i$ 个桶中的第 $j$ 个插槽，并使用 $FP$，$C$，$AC$ 和 $FG$ 分别表示指纹字段、计数器字段、活跃度字段和期望 top-$k$ 流标记字段。

插入：对于网络流 $f$ 的数据包 $\mathbb{P}_f$，ACCF 首先计算其指纹 $\mathbb{F}$ 和候选桶索引 $h_1$ 和 $h_2$，并检查 ACCF 中是否记录指纹 $\mathbb{F}$。根据以下 3 种检查情况，ACCF 执行不同的操作（见算法 1）。

情况 1　ACCF 中已记录网络流 $f$。ACCF 首先检查 $FG$ 字段的值是否为 1，若为 1，说明网络流 $f$ 已被标记为期望 top-$k$ 流，不执行其他操作；否则，ACCF 更新该流的活跃度并增加计数器值，然后将当前实时活跃度与设定的活跃度阈值 $\mathbb{A}_{thre}$ 进行比较，判定该流是否为期望 top-$k$ 流。

情况 2　ACCF 中没有记录网络流 $f$ 且候选桶中有空位置。ACCF 检查两个候选桶，并从中随机选择一个空位置插入网络流 $f$。

情况 3　ACCF 中没有记录网络流 $f$ 且候选桶已满。ACCF 随机选择一个候选桶，踢出活跃度最小的网络流项，并以〈网络流项，踢出索引位置，活跃度〉三元组的形式记录到 SRST 表中，然后将网络流 $f$ 插入。被踢出的网络流项通过异或运算执行重定位操作，若该候选桶中有空位置，则直接插入；否则，从中踢出活跃度最小的网络流项，并将其信息记录到 SRST 中。ACCF 循环执行踢出和重定位操作，直到插入成功并将 SRST 中记录的网络流数据重新插入 ACCF，或达到最大踢出上限，踢出 SRST 中活跃度值最小的网络流并重新插入剩余网络流。

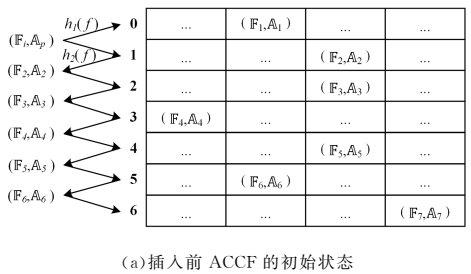
**算法 1**　ACCF 插入算法

输入：网络流 f 的数据包 $\mathbb{P}_f$

输出：无

1. 初始化：$\mathbb{F}$←fingerprint(f)，$h_1$←hash(f)，$h_2$←$h_1$⊕hash($\mathbb{F}$)
2. /* 情况 1：ACCF 中已存在流 f */
3. while i∈{$h_1$，$h_2$} && j∈{0，w} do
4. 　if b[i][j]. FP== $\mathbb{F}$ && b[i][j]. FG≠1 then
5. 　　b[i][j]. AC←$\mathbb{A}_P$ and b[i][j]. C+=1；
6. 　　if b[i][j]. AC> $\mathbb{A}_{thre}$ then
7. 　　b[i][j]. FG←1，return；
8. /* 情况 2：ACCF 未插入流 f 且有空 slot */
9. if b[$h_1$] ‖ b[$h_2$] has an empty slot then
10. 　insert f into ACCF，return；
11. /* 情况 3：ACCF 未插入流 f 且无空 slot */
12. i←$h_1$ or $h_2$；
13. j←argmin(b[i][j]. AC) && j∈{0，w}；
14. isreach=true；/* 设置 SRST 控制参数 */
15. 　for n←0 to MaxNumKicks do
16. 　〈$\mathbb{F}_n$，$\mathbb{A}_n$〉←b[i][j]；
17. 　append (〈$\mathbb{F}_n$，$\mathbb{A}_n$〉，(i，j)) to SRST；
18. 　i←i⊕hash(b[i][j]. FP)；/* 重定位 */
19. 　if b[i] has an empty slot then
20. 　　isreach←false，break；
21. 　j←argmin(b[i][j]. AC) && j∈{0，w}
22. /* 回填 SRST 记录的信息并刷新 */

插入示例：图 2 展示了 $b$ 值为 4 的 ACCF 插入操作。为了简化表示，示例使用($\mathbb{F}$，$\mathbb{A}$)和$\mathbb{F}$分别表示 ACCF 和 SRST 中的网络流项。插入示例设置 MaxNumKicks 的值为 5，并模拟一个特殊的高负载场景，即网络流 $f$ 的插入达到最大踢出上限。图 2(a)展示了 ACCF 的初始状态，其中重点标识了踢出的网络流项并省略了其他网络流项的描述。ACCF 计算网络流 $f$ 的指纹$\mathbb{F}$和候选桶索引 $h_1$ 和 $h_2$，并检查到 ACCF 中没有记录流 $f$ 且无空 slot 插入，则随机选择一个桶（候选桶 $h_2$），踢出活跃度最小的网络流项（$\mathbb{F}_2$，$\mathbb{A}_2$）并记录到 SRST 表中，然后对网络流项（$\mathbb{F}_2$，$\mathbb{A}_2$）执行重定位操作。ACCF 后续执行循环踢出和重定位操作，直到达到最大踢出上限，并依次将踢出的网络流项记录到 SRST 表中，如图 2(b)所示。SRST 表中记录的网络流项满足 $\mathbb{A}_4<\mathbb{A}_5<\mathbb{A}_3<\mathbb{A}_6<\mathbb{A}_2$，因此 ACCF 将舍弃网络流项（$\mathbb{F}_4$，$\mathbb{A}_4$），并将其他网络流项重新插入，如图 2(c)所示。




(a)插入前 ACCF 的初始状态

| Flow Entry | $\mathbb{F}_2$ | $\mathbb{F}_3$ | $\mathbb{F}_4$ | $\mathbb{F}_5$ | $\mathbb{F}_6$ |
|---|---|---|---|---|---|
| Index | (1,2) | (2,2) | (3,0) | (4,2) | (5,1) |
| Value(A) | $\mathbb{A}_2$ | $\mathbb{A}_3$ | $\mathbb{A}_4$ | $\mathbb{A}_5$ | $\mathbb{A}_6$ |

(b)SRST 记录数据

| | | | |
|---|---|---|---|
| ... | ($\mathbb{F}_1$,$\mathbb{A}_1$) | ... | ... |
| ... | ... | ($\mathbb{F}_f$,$\mathbb{A}_p$) | ... |
| ... | ... | ($\mathbb{F}_3$,$\mathbb{A}_3$) | ... |
| ($\mathbb{F}_4$,$\mathbb{A}_4$) | ... | ... | ... |
| ... | ... | ($\mathbb{F}_5$,$\mathbb{A}_5$) | ... |
| ... | ($\mathbb{F}_6$,$\mathbb{A}_6$) | ... | ... |
| ... | ... | ... | ($\mathbb{F}_7$,$\mathbb{A}_7$) |

(c)插入后 ACCF 的状态

图 2　ACCF 插入示例

Fig. 2　ACCF insert example

查询算法：查询网络流 $f_i$，ACCF 计算该网络流的指纹 $\mathbb{F}_i$ 和两个候选桶索引 $h_1$ 和 $h_2$，并查询桶 $h_1$ 和桶 $h_2$ 中是否已记录网络流 $f_i$。如果存在 $bk[i][j].FP=\mathbb{F}_i$，则表明网络流 $f_i$ 已被记录。其查询算法的伪代码如算法 2 所示。

**算法 2**　ACCF 查询算法

输入：网络流 f 的数据包 $\mathbb{P}_f$

输出：无

1. 初始化：$\mathbb{F}$←fingerprint(f)，$h_1$←hash(f)，$h_2$←$h_1$⊕hash($\mathbb{F}$)

```
2. while i∈{h1,h2} && j∈{0,w} do
3.   if b[i][j].FP== 𝔽 then
4.     return True;/*查询成功*/
5. return false;/*查询失败*/
```

## 4 实验评估

### 4.1 实验设置

1)实验平台

本实验运行在一台高性能服务器上,服务器配备了第 12 代 Intel(R) Core(TM) i7-12700H 处理器,拥有 14 核 20 线程架构和 128GB 内存,能够提供强大的计算能力和高效的多线程性能,支持内存密集型应用和大规模数据处理,确保实验高效稳定运行。

2)数据集

本实验使用五元组(源 IP、源端口、目的 IP、目的端口和协议)作为网络流的唯一标识,并选取来源于 CAIDA 的被动追踪数据集作为实验数据集。为了确保实验数据的时效性和实验结果的可靠性,选择了 CAIDA passive-2019 数据集中的一个子集进行实验分析。该子数据集由 Equinix-nyc 监视器于 2019 年 1 月 20 日 20 时 43 分捕获,包含 1 min 匿名被动流量数据(pcap 文件),其中记录了 36 892 281 个数据包,涵盖了多种网络协议,主要包括 IPv4 和 IPv6 协议以及传输层的 ICMP、TCP 和 UDP 协议。由于本研究聚焦于对 IPv4 数据包的处理,因此对原始数据集进行了必要的预处理,表 1 列出了预处理后的数据集协议的分布情况,为后续的实验分析提供了基础数据支持。

表 1 数据集信息

Table 1 Dataset information

| Transport Layer Protocol | Packet Count | Percentage/% |
|---|---|---|
| TCP | 29 177 989 | 80.912 |
| UDP | 6 883 310 | 19.088 |

3)系统实现

本文通过编写超过 1 500 行的 C++代码实现了 ACCF,并提供了插入和查询两个核心功能。本实验将 ACCF 与其他算法(如 Space Saving, CM Sketch, Cuckoo Counter 和 LUSketch)相结合,用于实现 top-$k$ 流测量。这些组合方法旨在评估 ACCF 在不同算法框架下的性能和准确性,尤其是在精度、空间效率和处理速度方面的优越性。

### 4.2 性能指标

在性能评估中,本文对测量系统的各项指标进行了详细分析,主要评估了 ACCF 大流过滤的精确率(Precision)、召回率(Recall)、插入吞吐量、top-$k$ 流测量精确率和 F1-Score。

精确率:衡量大流检测数据中实际为大流的比例。

$$Precision=\frac{TP}{TP+FP} \tag{4}$$

其中,$TP$ 表示被正确识别为大流的流量,$FP$ 表示被错误识别为大流的流量。

召回率:衡量实际为大流的流量中,有多少网络流被正确识别为大流。

$$Recall=\frac{TP}{TP+FN} \tag{5}$$

其中,$FN$ 表示未被识别为大流的流量。

F1-Score 的计算式如下:

$$F1\text{-}Score=2\times\frac{Precision\times Recall}{Precision+Recall} \tag{6}$$

大流的提前过滤量(Pre-Filtering Ratio):表示基于活跃度预测机制的 ACCF 在大流过滤操作中,提前成功过滤的大流所占过滤大流总数的比例。

插入操作减少量(Reduction in Insertion Ratio, RIR):表示基于活跃度预测机制的 ACCF 在大流过滤操作中减少的插入次数比例。

吞吐量(Throughput):评估 ACCF 性能的关键指标,用于表示 ACCF 在单位时间内成功插入的数据包数量。吞吐量的计算式为:

$$throught=N/t \tag{7}$$

其中,$N$ 表示成功执行插入操作的数据包总数,时间 $t$ 表示算法执行的总时间。

### 4.3 ACCF 参数

ACCF 是一种基于 Cuckoo Filter 进行优化设计的测量结构,其指纹长度的计算式为:

$$f\geqslant\lceil\log_2(2b/\epsilon)\rceil \tag{8}$$

其中,$\epsilon$ 为查询假阳性率,$b$ 为每个桶中的插槽数量。较低的假阳性率 $\epsilon$ 能够减少误判,但会增加指纹长度,进而带来额外的内存开销,而增加插槽数量 $b$ 可以降低冲突概率,提高插入和查询操作效率。在验证实验中,设定插槽数量 $b$ 为 4,指纹长度 $f$ 为 16,在确保足够假阳性率的同时优化了 ACCF 的存储需求。

#### 4.3.1 活跃度计算参数

在 ACCF 中,引入活跃度计算的目的是为了实现大流的提前过滤,以减少不必要的插入操作,降低计算开销并提升整体过滤效率。活跃度计算通过结合数据包权重衰减因子 $\lambda$ 和 EWMA 的平滑系数 $\alpha$ 来动态预测网络流的发展趋势,二者的设置会直接影响 ACCF 的大流过滤性能。

以实验数据集中最大网络流为例,图 3 展示了数据包权重在衰减因子 $\lambda$ 不同取值时的变化。$\lambda$ 的值越近于 0 时,权重变化曲线越平滑,数据包权重衰减较慢;相反,随着衰减因子 $\lambda$ 的增大,权重衰减得更快。基于活跃度的过滤机制倾向于在短期内保持较高的数据包权重,以捕捉流量近期的变化,同时避免长期内权重过快衰减,从而保证历史数据对活跃度的影响。

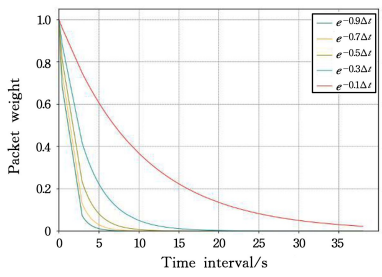



图 3 数据包权重变化

Fig. 3 Changes in packet weights

平滑因子 $\alpha$ 平衡了网络流短期动态变化与历史趋势对活跃度的影响。较大的 $\alpha$ 使系统更关注近期流量的波动，而较小的 $\alpha$ 更重视网络流的历史趋势。在本实验中，通过设置不同的衰减因子 $\lambda$ 和平滑系数 $\alpha$ 的组合，以验证 ACCF 在大流识别中的性能，并评估基于活跃度的预测机制对过滤性能的提升。实验中特别选取了 $\alpha=0.2$，$\alpha=0.5$ 和 $\alpha=0.8$ 作为平衡短期变化与历史趋势的 3 个典型值，用于分析这两种极端情况下系统的表现，而 $\lambda$ 取 0.1 和 0.3 两项指标值。其中，ACCF 的大流识别性能由 Precison 和 Recall 两个实验结果衡量，而过滤性能关注大流的提前过滤量和插入次数减少量。

从表 2 中可以看出，当衰减因子 $\lambda$ 和平滑系数 $\alpha$ 更关注历史数据时，ACCF 在保证高精确率的前提下，能够显著提升大流的提前过滤能力，并减少插入操作的次数。具体而言，随着 $\alpha$ 的增大，PFR 显著提升，可达 65.24%，同时 RIR 也达到了 41.22%。

表 2 $\lambda$ 和 $\alpha$ 对 ACCF 的性能影响

Table 2 Effect of $\lambda$ and $\alpha$ on the performance of ACCF

| $(\lambda,\alpha)$ | *Precision* | *Recall* | *PFR*/% | RIR/% |
|---|---|---|---|---|
| (0.1,0.2) | 0.9525 | 0.9779 | 55.33 | 37.45 |
| (0.1,0.5) | 0.9749 | 0.9776 | 59.59 | 38.32 |
| (0.1,0.8) | 0.9981 | 0.9774 | 65.24 | 41.22 |
| (0.3,0.2) | 0.9598 | 0.9778 | 55.74 | 39.67 |
| (0.3,0.5) | 0.9761 | 0.9775 | 58.13 | 40.44 |
| (0.3,0.8) | 0.9999 | 0.9774 | 62.67 | 43.56 |

### 4.3.2 MaxNumKicks 值

MaxNumKicks 是影响 ACCF 过滤效率的重要参数，较大的 MaxNumKicks 值会增加 SRST 的长度，从而扩大局部最小值的选择范围，提高过滤的准确率。然而，过大的值可能导致 ACCF 失去 SRST 的优势，降低计算效率，并占用更多存储空间。实验通过研究 MaxNumKicks 从 1～20 变化过程中的 ACCF 过滤性能变化，系统分析了 ACCF＋SRST，ACCF＋MARS 和 ACCF＋RKOM 这 3 种策略的性能表现，如图 4 所示。

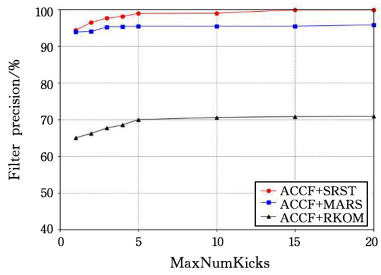



图 4 过滤精确率变化

Fig. 4 Filtration precisions change

从图 4 可以看出，随着 MaxNumKicks 值的增加，3 种 ACCF 版本的准确率均有所提升，但在表现上存在明显差异。SRST 策略在 MaxKicksNum 达到 10 后，过滤精确率趋于稳定，接近 100%，展现了极高的识别精度。相比之下，MARS 策略的过滤精确率略低于 SRST；而 RKOM 策略表现最差，随着 MaxNumKicks 的增加，过滤精确率只是从 60%提升至约 75%，其性能明显低于前两种方案。

考虑到 MaxNumKicks 增加对测量系统吞吐量的影响，从图 5 可以看出，SRST 方案的吞吐量并没有明显的下降，这说明在 ACCF 结构之外增加一个扩展表不仅提升了大流的识别率，而且不会对系统的吞吐量产生负面影响。

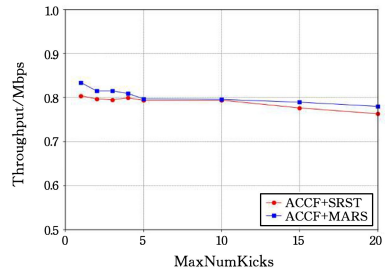



图 5 测量系统吞吐量变化

Fig. 5 Measure changes in system throughput

综合上述对 ACCF 参数的实验分析可得，ACCF 的参数取插槽数 $b$ 为 4、指纹长度 $f$ 为 16 的情况下，衰减因子 $\lambda$ 为 0.1、平滑系数 $\alpha$ 为 0.8 和 $MaxNumKicks=10$ 时，ACCF 能够维持较高的精确率和召回率，还能够显著提高大流的提前过滤比例，同时大幅减少不必要的插入操作次数。

### 4.4 top-$k$ 流测量实验

本节通过将 ACCF 作为前置大流过滤器，与 SS，CM Sketch，Cuckoo Counter 和 LUSketch 结合进行 top-$k$ 流测量的比较实验。通过将 ACCF 与这 4 种已有的方法结合，旨在系统地评估 ACCF 在提升 SS，CM Sketch，Cuckoo Counter 和 LUSketch 的 top-$k$ 流测量性能方面的优势。实验通过设置定值 $k$ 和动态变化的 $k$ 值，观察了 ACCF 对 SS，CM Sketch，Cuckoo Counter 和 LUSketch 的性能变化。

1）定值 $k$：实验设置 $k$ 值为 1 000，针对不同内存条件下，比较了 SS，CM Sketch，Cuckoo Counter 和 LUSketch 单独测量以及与 ACCF 进行组合测量的性能表现，实验结果如图 6 示。单独使用 SS，CM Sketch，Cuckoo Counter 和 LUSketch 进行 top-$k$ 流测量时，随着内存的增大，其 F1-Score 的提升有限，表现出较为平缓的增长趋势。然而，当 4 种测量算法与 ACCF 进行组合测量时，4 种算法的 F1-Score 得到显著提升，尤其是在内存大于 32 kB 时，F1-Score 值达到了 0.85 以上。因此，ACCF 的引入极大地提升了现有测量算法对 top-$k$ 流的识别精度，尤其是在内存较大时表现尤为出色。

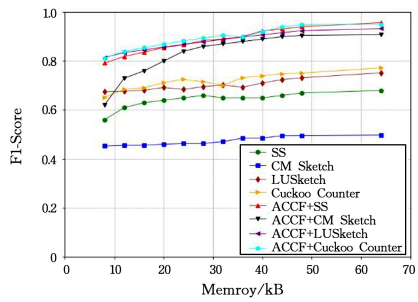



图 6 定值 $k$：top-$k$ 测量变化

Fig. 6 Fixed value $k$: top-$k$ measures change

2）动态变化的 $k$ 值：实验将内存设置为 32 kB，针对不同 $k$ 值，分别比较了 SS，CM Sketch，Cuckoo Counter 和 LUSketch 单独测量，以及与 ACCF 进行组合测量的识别精度，实验结果如图 7 所示。实验结果表明，随着 $k$ 值的增加，不管是单独

测量还是组合测量方案，其精度均有所下降，但与 ACCF 的组合测量方案明显优于单独使用 SS，CM Sketch，Cuckoo Counter 和 LUSketch 的方案。在小 $k$ 值范围内（$k<300$），4 种算法与 ACCF 组合的测量方案 top-$k$ 流测量精度接近 1，表现出极高的识别准确性，尤其是 ACCF+SS，在 $k$ 值较大时仍能维持较高的精度。相比之下，单独使用 SS，CM Sketch，Cuckoo Counter 和 LUSketch 进行 top-$k$ 时，精度下降更明显，尤其是 CM Sketch 在较大 $k$ 值时表现出较大的波动，说明其对内存和 $k$ 值变化更为敏感。

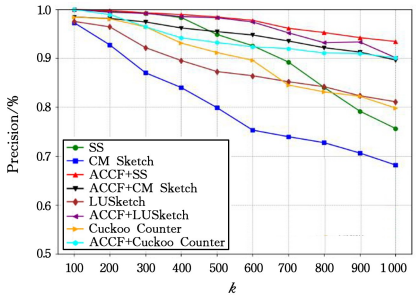



图 7 动值 $k$：top-$k$ 测量变化

Fig. 7 Dynamic value $k$: top-$k$ measures change

**结束语** 本文针对 top-$k$ 流过滤算法中存在的固定阈值问题，提出了一种基于活跃度预测机制的 ACCF 测量结构，并结合 SRST 实现大流的提前过滤。该算法的核心思想是：通过流的历史活跃度信息，利用 EWMA 机制动态预测网络流的未来活跃度，从而替代传统的固定阈值过滤策略。ACCF 将活跃度较高的流优先标记为潜在大流，减少了不必要的插入操作和资源消耗。同时，结合自刷新存储表的设计，使得系统能够实时更新和调整对流量的识别，从而提高了在不同流量场景下的适应性和过滤效率。实验结果表明，该算法在 top-$k$ 流检测中的精确率和资源利用效率相比传统方法有显著提升。

然而，本文仅采用静态的活跃度计算参数进行分析，即衰减因子 $\lambda$ 和平滑系数 $\alpha$ 未随网络流量的动态变化进行自适应调整。尽管静态参数能够在某些场景下提供良好性能，但在实际复杂网络环境中，流量的时序特性和突发性波动可能导致活跃度预测的准确性下降。因此，未来的工作可以考虑引入自适应参数调整机制，根据流量的实时变化动态优化 $\lambda$ 和 $\alpha$，以进一步提高 ACCF 在不同网络条件下的检测精度和响应能力。

## 参 考 文 献

[1] ZHANG T, ZHANG Q, LEI Y, et al. Load balancing with traffic isolation in data center networks[J]. Future Generation Computer Systems, 2022, 127: 126-141.

[2] LIU S, HE M, WU Z, et al. Spatial-temporal graph neural network traffic prediction based load balancing with reinforcement learning in cellular networks[J]. Information Fusion, 2024, 103: 102079.

[3] HUANG Q, LEE P P C, BAO Y. Sketchlearn: Relieving user burdens in approximate measurement with automated statistical inference[C]// Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. 2018: 576-590.

[4] ZHANG X, WANG T. Elastic and reliable bandwidth reservation based on distributed traffic monitoring and control[J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(12): 4563-4580.

[5] BISWAS R, WU J. Traffic engineering to minimize the number of rules in sdn datacenters[J]. IEEE Transactions on Network Science and Engineering, 2021, 8(2): 1467-1477.

[6] ROBIN D D, KHAN J I. P4TE: PISA switch based traffic engineering in fat-tree data center networks[J]. Computer Networks, 2022, 215: 109210.

[7] KUMAR R, VENKANNA U, TIWARI V. Optimized traffic engineering in Software Defined Wireless Network based IoT (SDWN-IoT): State-of-the-art, research opportunities and challenges[J]. Computer Science Review, 2023, 49: 100572.

[8] SHI G, SHEN X, XIAO F, et al. DANTD: A deep abnormal network traffic detection model for security of industrial internet of things using high-order features[J]. IEEE Internet of Things Journal, 2023, 10(24): 21143-21153.

[9] FERNÁNDEZ Y, SOTO J E, VERA S, et al. A streaming algorithm and hardware accelerator to estimate the empirical entropy of network flows[J]. Computer Networks, 2023, 237: 110035.

[10] WU M, HUANG H, SUN 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.

[11] WANG H, XU H, HUANG L, et al. Fast and accurate traffic measurement with hierarchical filtering[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(10): 2360-2374.

[12] KANNAN K, BANERJEE S. Compact TCAM: Flow entry compaction in TCAM for power aware SDN[C]// International Conference on Distributed Computing and Networking. Berlin: Springer, 2013: 439-444.

[13] OZERY A, DIAMANT J, FEIBISH S L. RecenTo: Finding Top-K Flows of the Recent Past[J]. Proceedings of the ACM on Networking, 2024, 2(CoNEXT3): 1-20.

[14] SHI Q, XU Y, QI J, et al. Cuckoo counter: Adaptive structure of counters for accurate frequency and top-k estimation[J]. IEEE/ACM Transactions on Networking, 2023, 31(4): 1854-1869.

[15] ZHANG B, HUANG H, SUN Y E, et al. Jigsaw-Sketch: a fast and accurate algorithm for finding top-k elephant flows in high-speed networks[J]. Science China Information Sciences, 2024, 67(4): 142101.

[16] ZHU Y, SONG Q, LUO Y. Differentially Private Top-$k$ Flows Estimation Mechanism in Network Traffic[J]. IEEE Transactions on Network Science and Engineering, 2023, 11(3): 2462-2472.

[17] CORMODE G, MUTHUKRISHNAN S. An improved data stream summary: the count-min sketch and its applications[J]. Journal of Algorithms, 2005, 55(1): 58-75.

[18] DIMITROPOULOS X, HURLEY P, KIND A. Probabilistic lossy counting: An efficient algorithm for finding heavy hitters [J]. ACM SIGCOMM Computer Communication Review, 2008,

38(1):5-5.

[19] MITZENMACHER M,STEINKE T,THALER J. Hierarchical heavy hitters with the space saving algorithm[C]// 2012 Proceedings of the Fourteenth Workshop on Algorithm Engineering and Experiments (ALENEX). Society for Industrial and Applied Mathematics,2012:160-174.

[20] ROY P,KHAN A,ALONSO G. Augmented sketch:Faster and more accurate stream processing[C]// Proceedings of the 2016 International Conference on Management of Data. 2016:1449-1463.

[21] ZHOU Y,YANG T,JIANG J,et al. Cold filter:A meta-framework for faster and more accurate stream processing[C]// Proceedings of the 2018 International Conference on Management of Data. 2018:741-756.

[22] YU X,XU H,YAO D,et al. CountMax:A lightweight and cooperative sketch measurement for software-defined networks [J]. IEEE/ACM Transactions on Networking,2018,26(6):2774-2786.

[23] LU J,CHEN H,ZHANG Z. LUSketch:A fast and precise sketch for top-k finding in data streams[C]// 2022 International Conference on Computer Communications and Networks (ICCCN). IEEE,2022:1-10.

[24] WANG C,LI X,ZENG J,et al. Prob-CS:A Probabilistic Cuckoo Sketch for Accurate Network Traffic Measurement[J]. IEEE Internet of Things Journal,2024,11(22):36965-36978.

[25] XIONG B,LIU Y,LIU R,et al. ActiveGuardian:An accurate and efficient algorithm for identifying active elephant flows in network traffic[J]. Journal of Network and Computer Applications,2024,224:103853.

[26] YANG T,ZHANG H,LI J,et al. HeavyKeeper:an accurate algorithm for finding Top-k elephant flows [J]. IEEE/ACM Transactions on Networking,2019,27(5):1845-1858.

[27] YANG T,GONG J,ZHANG H,et al. Heavyguardian:Separate and guard hot items in data streams[C]// Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018:2584-2593.

[28] JIA P,WANG P,ZHAO J,et al. Loglog filter:Filtering cold items within a large range over high speed data streams[C]// 2021 IEEE 37th International Conference on Data Engineering (ICDE). IEEE,2021:804-815.

[29] LI Y,WANG F,YU X,et al. Ladderfilter:Filtering infrequent items with small memory and time overhead[C]// Proceedings of the ACM on Management of Data. 2023:1-21.

**HU Yongqing**, born in 1999, postgraduate, is a member of CCF (No. G9824G). His main research interests include network traffic measurement and software-defined networking.

**DAI Qinglong**, born in 1988, Ph.D, associate professor, master supervisor, is a member of CCF (No. H6517M). His main research interests include edge computing, big data and cloud computing.

(责任编辑:喻藜)