# 一种时间复杂度为$\Theta(n)$的网络流量一维聚合分析方法

彭莹[1,2]，龚俭[1,2]

（1.东南大学计算机科学与工程学院，江苏省南京市，211189；2.江苏省网络技术重点实验室，江苏南京，211189）

**摘　要：**IP 流记录中某些字段（例如，源 IP 地址）的取值集合按照集合间包含关系自然地形成了层次结构。沿着层次结构对网络流量进行聚合分析，获取占用资源（例如，带宽）较多的数据集合，对于网络性能测量和流量异常检测都很有意义。Estan 等人首先提出来了这样一种流量分析方法，自底向上地离线构建占用资源较多的数据集合。Cheng 等人对 Estan 的方法进行改进，提出了一种自顶向下的聚合方法。然而算法的时间复杂度为 $O(n\log n)$。在本文中，我们做了 Cheng 的方法对进一步的改进，通过以空间换时间和分治等策略，使算法的时间复杂度降低至 $\Theta(n)$。

**关键词：**网络测量；流量分析；流量聚合

# A One-dimensional Network Traffic Aggregation with Time Complexity of $\Theta(n)$

Peng Ying[1,2], Gong Jian[1,2]

(1.School ofComputer Science & Engineering, Southeast University, Nanjing, 211189; 2. Jiangsu Provincial Key Laboratory of Computer Network Technology, Southeast University, Nanjing, 211189)

**Abstract:** In the IP flows, the natural hierarchies exist for some field (e.g., source IP address). Aggregating along hierarchies to get the clusters with significant amount of resource consumption is useful for both network performance measurement and traffic anomaly detection. Estan et al. first proposed this kind of network traffic analysis method. They designed an offline algorithm constructing significant clusters in a bottom-up way. Cheng et al. improved this algorithm by aggregating in a top-down way. However, the computational complexity of their method is $O(n\log n)$. In this paper, we introduce a practical technique that further improved Cheng et al.'s algorithm by taking the strategies, "space for time" and "divide and conquer". The computational complexity of our method is reduced to $\Theta(n)$.

**Key words:** Network Measurement; Traffic Analysis; Traffic Aggregation

## 1 引言

IP 报文头部有多个字段，其中由源宿 IP 地址、源宿端口和协议五个字段形成的五元组流规范被广泛采用，五个字段上具有相同取值的报文组成一条流。IP 流中的这些字段的取值集合按照集合包含关系可以自然地形成层次结构，根据层次结构进行流量聚合，获取流量数据集合及其占用的资源信息，是一个被用于网络管理很多应用中的分析检测技术。例如，网络感知聚合基于访问模式识别网络群组（例如，在相同管理域下的若干主机），特别是产生了对一个网站大部分请求的那些组（从 IP 流数目方面来测量）。关于繁忙群组的信息，对路由器中的报文分类、QoS 的区分、代理的设置以及服务器的备份等都很有用处。另外，在一个分布式的 SYN 泛洪 DOS 攻击中，攻击者向受害者发送大量的 SYN 包（启动会话），但却不以 ACK 包响应受害者发来的 SYN-ACK 包来完成三次握手协议，从而占用受害者的资源。当一个主机收到的 SYN 和 ACK 数据包数目悬殊时，这样一种 DoS 攻击就可能被检测到。然后可以在不同层次上，为 IP 地址前缀维护 ACK 包数目和 SYN 包数目的比值，从而监视被攻击的宿地址（受害者）以及源地址（攻击者）。实际上，为了防御这种 DoS 攻击，基于聚合的自动化拥塞控制技术（例如，基于子网前缀）也已被提出[1]。

**作者简介：**彭莹，（1991-），女，硕士研究生，E-mail: ypeng@njnet.edu.cn; 龚俭，（1957-），男，教授，博导，E-mail: jgong@njnet.edu.cn.

我们用流量簇来描述流量数据集合。报告给管理员的簇数目越多，提供的流量细节就越多，管理员对流量就越了解，但是过多的细节同时也使得报告难以阅读，管理员不一定能吸收这些信息完成快速响应。因此需要流量聚合提供真正重要的占用资源较多的簇，我们用重击点（*heavy hitter*, HH）来描述占用资源较多的簇。

各个字段定义了流在不同维度上的特征，所以网络流量是一种具有多维属性的数据。一维聚合是指在一个维度上对流量进行聚合，是流量聚合的最简单情况，也是多维流量聚合的基础。一维聚合分析针对流的某个标识字段，分析从网络中采集到的流数据，报告其中超过管理员设置阈值$\phi$的流量模式，即重击点。在本文中，我们研究的是在数据集合中发现层次重击点（*hierarchical heavy hitter*, HHH）：给定一个数据集合 $S$ 和一个分数$\phi$，其中的数据子集按照集合包含关系可以自然地形成层次结构，我们希望在层次结构中找到那些扣除已经被子 HH 含盖的元素后，所含盖的元素数目仍然不小于$\phi|S|$的结点，即 HHH。层次重击点提供了层次域数据集合中数据的拓扑统计图。下面我们给出 HH 和 HHH 的形式化的定义。

**定义 1（重击点）**给定一个基数为 $N$ 的多重集合 $S$ 和一个阈值$\phi$，重击点（*HH*）是指在 $S$ 中出现的频度不小于$\lceil \phi N \rceil$的一个元素。记 $f_e$ 为 $S$ 中的元素 $e$ 的频度，那么 $HH = \{e \mid f_e \geq \lceil \phi N \rceil\}$。

**定义 2（层次重击点）** 给定一个多重集合 $S$，其中的元素来自高度为$h$的层次域$D$。记*elements(T)*为层次域中前缀集合 $T$ 子元素的并集。给定一个阈值$\phi$，我们归纳地定义 $S$ 的层次重击点。记 $HHH_0$ 为第 0 层（最底层）的层次重击点，其实就是 $S$ 的重击点。给定一个层次结构中第 i 层的前缀 $p$，记 *F(p)* 为 $\sum f_e$，其中 $e \in elements(\{p\}) \wedge e \notin elements(\bigcup_{l=0}^{i-1} HHH_l)$。$HHH_i$ 为第 i 层的层次重击点，$HHH_i = \{p \mid F(p) \geq \lceil \phi N \rceil\}$。记 *HHH* 为重击点集合，$HHH = \bigcup_{i=0}^{h} HHH_i$。因为 $\sum_p F(p) = N$，层次重击点的数目不会超过 $1/\phi$。

我们用一个例子来解释一下层次重击点的定义，多重集合 $S$ 中的元素是那些作为 Web 请求源地址的 IP 地址。$S$ 中有 $N$(=100,000)个元素，其中与 IP 地址前缀相关联的 IP 地址数目统计如下：135.207.50.250/24(2003)，135.207.50.250/25(1812)，135.207.50.250/26(1666)，135.207.50.250/27(1492)，135.207.50.250/28(1234)，135.207.50.250/29(1001)，135.207.50.250/30(767)，135.207.50.250/31(404)，和 135.207.50.250/32(250)，其中 *ipaddr/b(c)*代表了通过取 *ipaddr* 的前 *b* 比特获得的 IP 地址前缀含盖了 *c* 个 IP 地址。使用阈值$\phi$=0.01，仅有 135.207.50.250/29 和 135.207.50.250/24 是 HHH，前者是 HHH 的原因是包含的 IP 地址数目超过了阈值（100,000*0.01=1000），后者是因为扣除它的子 HHH——135.207.50.250/29 后，它包含的 IP 地址数目仍超过了阈值。

Estan 等人在[2]中首先提出了这样一种沿着层次结构聚合的流量分析方法，但是这个算法有潜在的排序操作和大量的内存分配操作，时空消耗都比较大，是一个离线算法。CHENG 等人在[3]中在 Estan 算法上进行了改进，提出了一种资源节约的快速一维网络流量聚合方法，采用了自定向下的方法建立起层次树，并且通过子集约束等策略减少了流记录的遍历次数，节省了时间。但是算法仍需要排序操作，时间复杂度为$\Theta(n \log n)$（$n$ 为流记录数目），不能负荷大流量。在本文中，我们提出进一步的改进。通过以空间换时间和分治等策略，多使用了 2*n*~6*n* 个存储空间，算法的时间复杂度降低至$\Theta(n)$，流记录的遍历次数也大大减少。

本文组织如下。我们在第二节描述了我们的算法过程；在第三节对我们的算法进行了分析和实验评估；在第四节中做了总结。

## 2、算法描述

层次结构可以用树来刻画，树的所有叶子节点是取值集合中的单个元素。例如，对于 26 层 IP 前缀层次树，根节点有 256 个子节点，分别连接到 256 个/8 前缀；每棵/8 子层次树采用每层前缀增加 1bit 的二叉树结构，叶节点是/单个 IP 地址，加上根节点的一层一共 26 层。

对于取值集合较小的协议、端口，聚合算法很简单，所以以下我们仅讨论 IP 地址维度。但对于 IP 地址，因为 IP 地址空间按照前缀形成的层次树节点数目过多，不可能为层次树中所有节点维护

对应的计数器。为此设计了下面的算法对网络流量进行聚合并输出带宽占用超过阈值$\phi$的 IP 地址前缀，我们把 IP 前缀层次树的 17~26 层（/24~/32）称为低层次, 1~16 层（/8~/23）称为高层次。算法过程如图 1 所示。

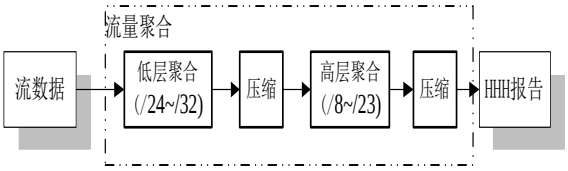


图 1 一维聚合算法过程

整个聚合模块由四部分组成：1、通过低层聚合获得低层的原始 HHH，2、通过压缩低层次原始报告获得低层的 HHH 报告，3、通过高层聚合获得高层的原始 HHH，4、通过压缩高层次原始报告获得高层的 HHH 报告。网络采集到的流数据通过流量聚合模块后，最终呈现给管理员一张 HHH 列表。

具体的算法如下：

**1) 分配三块存储区域**

流缓存、层次重击点缓存、由/24 前缀标识的一组桶。流缓存是一个大小为 N 的一维数组，记录流的标识字段和流量等信息。由于每层至多有 $1/\phi$ 个重击点，最多只可能有 $32/\phi$个重击点，所以分配大小 $26/\phi$（采用 26 层的 IP 前缀层次树）的二维数组作为重击点缓存。桶通过哈希表实现，每个桶记录一个/24 前缀含盖每个 IP 地址在流缓存中的位置，以及其中所有 IP 地址的总流量。哈希表的溢出通过链接方法来处理。这也是我们相比[2]中的方法多使用的一部分内存。

**2) 预处理**

收集流记录，将其中出现的 IP 地址写入流缓存，并按照在相应桶中插入相关信息。

**3) 通过底层聚合获得低层的原始 HHH**

对桶进行遍历，如果总流量在阈值之上，则记为第 24 层中的重击点，写入到重击点缓存数组的第 24 行，将其中的 IP 地址在流缓存中的位置索引转存，然后销毁桶。通过遍历每个第 24 层的重击点包含的 IP 地址子集，在第 25 层的重击点（长度增加 1 的 IP 地址前缀）都能被找到，迭代这个过程，可自顶向下的逐层获得重击点，直到建立了从第 24 层到第 32 层（最底层）的原始报告。

**4) 通过压缩低层次原始报告获得低层的 HHH 报告**

可以看到如果某个重击点流量超过阈值，那么从该重击点直至根节点路径上的所有重击点都会被写入报告。这样，原始报告中就引入了冗余聚合点。原始报告压缩的目的就是去除原始报告中聚合点重叠流量，从而将报告中重击点的数目确定在上界 $1/\phi$之内。这里我们采用 Estan 在[1]中提出的压缩算法，在原始报告中重击点形成的拓扑结构上进行的后序遍历，去除按照层次聚合时产生的重叠流量，得到低层的 HHH 报告。请参看文献，这里不再赘述。

**5) 通过高层聚合获得高层的原始 HHH**

此时，桶结构只剩下了那些总流量在阈值之下的桶。我们遍历桶结构，用总流量小于阈值的/24 IP 地址前缀和相应的总流量刷新流缓存。在第 3 步的压缩过程中，第 24 层的一些重击点流量未被全部计入低层次 HHH 报告，我们将这些重击点对应的/24 IP 地址前缀和未被计入的流量信息也插入到流缓存中。此时流缓存中存放的其实是/24 IP 地址前缀和其流量等信息，信息条目数远远小于原始流记录的数目。使得高层的聚合可以更加快速的完成。

向步骤 2 中的聚合过程一样，可以自顶向下通过遍历上一层重击点含盖的 IP 地址前缀子集，获得下层的重击点，直到第 23 层。这样就获得了高层的 HHH 原始报告。

**6) 通过压缩高层次原始报告获得高层的 HHH 报告**

压缩操作同第 3 步，从而得到高层的 HHH 报告。

这样最终我们获得了一份完整的 HHH 报告。

## 3、算法分析和实验评估

我们利用从 CERNET 江苏省网边界上采集到流量数据对本文提出的聚合方法进行测试。数据采集从 2013 年 4 月 10 日 24:00 点开始至 2013 年 4 月 11 日 24:00 点结束，以 5min 作为时间间隔，获得 288 个流量数据文件作为算法输入，将阈值$\phi$设置为 0.01，并且只进行源 IP 地址的聚合。

整个聚合算法是通过对流记录的多次遍历完成的，所以整个算法的时间复杂度为$\Theta(n)$，$n$ 为流记录的数目。具体的在聚合算法中：步骤 1 存储区域是预先设置的，时间复杂度为$\Theta(1)$。步骤 2 预处理过程的时间复杂度为$\Theta(n\alpha)$，$\alpha$ 为哈希表的装载密度；步骤 3 低层次聚合的时间复杂度为$\Theta(n)$；步骤 4 压缩过程的时间复杂度为$\Theta(1)$；步骤 5 高层次聚合的时间复杂度为$O(n)$；步骤 6 的时间复杂度为$\Theta(1)$。在空间占用方面，相对于[2]中的方法，我们只是增加了一个 hash 表结构，其中存放的是每个 IP 地址在流缓存中的位置索引（每个记录占用 2 个存储单元），以及/24 IP 地址前缀及其流量信息（每个记录占用 4 个存储单元），共需$(2n+4m)$个存储空间，其中 $m$ 为/24 IP 地址前缀的数目，在实验过程中我们 $m>>n$，如图 2 所示。

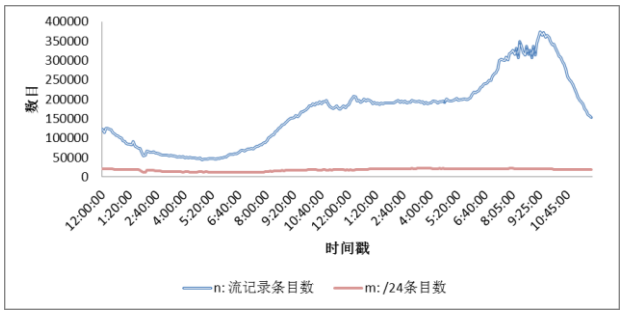


图 2 流记录数目和/24 IP 地址前缀条目

同时，我们还观察了算法的运行时间并计算了运行时间与流记录数目的比值。从图 3 中，可以看出运行时间与流记录数目的比值并没有因为流记录数目的增大而增大，变化相对稳定。可见我们的算法时间复杂度确实是$\Theta(n)$的。

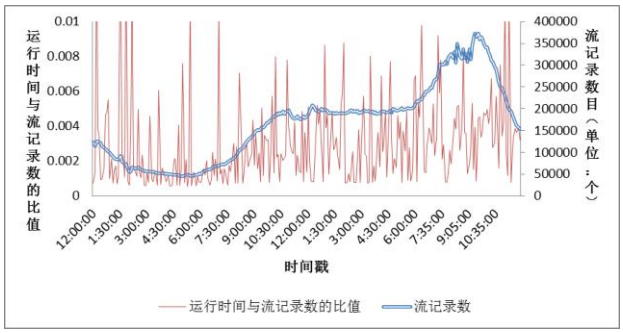


图 3 运行时间与流记录数目

## 4、总结

流量聚合分析是一种流量分析总结技术，提供真正重要的值得关心的数据，避免网络管理人员“大海捞针”。IP 维度上的聚合是一维聚合分析中最复杂的情况，本文使用了以空间换时间、分治等策略改进了算法，避免了排序操作，降低了算法的时间复杂度。我们使用 CERNET 江苏省网边界的流数据，对算法进行了评估和比较，实验结果也表明算法的时间复杂度确实降为$\Theta(n)$。

## 参考文献：

[1]. R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. "Controlling high bandwidth aggregates in the network." [J]. Computer Communications Review, 32(3):62–73, 2002.

[2]. Cristian Estan, S. Savage, and G. Varghese. "Automatically Inferring Patterns of Resource Comsumption in Network Traffic. "[A]. Proceedings of ACM SIGCOMM[C]. 2003.

[3]. G Cheng. "Real-time inferring network traffic patterns." [A]. The 8th Annual IEEE Consumer Communications and Networking Conference - Security and Content Protection[C]. 2011.

[4]. Cormode, Graham, et al. "Finding hierarchical heavy hitters in data streams." [A]. Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment[C]. 2003.