活跃节点检测算法的分析与研究

李翔[1, 2]，程光[1, 2]

（1. 东南大学计算机科学与工程学院，南京，211189；
2. 计算机网络与信息集成教育部重点实验室，南京，211189）

摘　要： 活跃节点的研究在当前网络安全、管理以及行为学研究等方面都有着十分重要的意义。本文先简单介绍了当前的活跃节点研究情况，然后提出了一种新的活跃节点检测算法。该算法先对报文数据进行流记录和报文层面上的抽样统计，通过 Bitmap 结构来维护流记录的状态信息，并利用 IP hash 表来记录被抽样到的 IP 的相关信息。针对 Bitmap 中存在的 hash 冲突，设计了一种流记录数的补偿策略，提高了算法的测量精度。最后通过计算分析 IP hash 表中的统计信息，得到活跃节点的测量结果。通过实验的结果分析，该算法在活跃节点测量时具有较高的精度和实用性。

关键词： 活跃节点，抽样，Bitmap，冲突补偿

Research on the Active-Node Detection Algorithm

LI Xiang[1, 2], CHENG Guang[1, 2]

(1. School of computer science & engineer, Southeast University, Nanjing, 211189;
2. Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing, 211189)

Abstract: The research on active-node has a very important significance in the network security, management and behavioral research. This article briefly describes the current research on active-node, and then proposes a new detection algorithm. The algorithm samples on flow and packet level, and uses the Bitmap to maintain the flow status information and IP hash table to record the information of sampled IP. Then we used a strategy of compensating the number of flow for the hash conflicts in Bitmap. Through the analysis of the results of the experiment, the algorithm has higher accuracy and usefulness of the active nodes measure.

Key words: Active-Node, Sample, Bitmap, Conflict-Compensation

1. 引言

高速网络流量检测和分析以及相应行为的研究能够给网络管理者提供当前网络的性能、负载以及潜在故障等关键性的管理信息。随着网络应用的不断丰富和网络速率的不断提高，网络管理者对于网络流量和行为监控等方面的需求变得日益突出。而在研究者们以往对网络流量进行的分析和研究中发现，少量的主机在网络中建立了大量的通信连接数，同时发送了占大部分网络总流量的报文数据。因此，通过分析研究这些特殊的主机，我们就能较好的实现对整个网络的管理和研究。

我们将这样的主机或节点定义为活跃节点。活跃节点是具有大量报文、字节和流等多维流量属性的主机。网络中很多安全事件和应用都体现出活跃节点的特性，如 DDoS 攻击、蠕虫扫描、端口扫描等异常流量，P2P 流量、热点 Web、FTP 服务器等应用流量，这些流量对网络运行会产生重要影响，是网络管理员关注的重点所在。在 P2P 网络节点中总数 1%的节点占据了 80%的连接数和流量，如果能够准确及时地发现这些占总数不足 1%的活跃节点，就有可能以较小的代价达到控制整个 P2P 网络流量的目的，因此检测和分类活跃节点的研究成果将具有广泛的应用价值。

2. 相关工作

目前活跃节点检测的研究主要是针对单维特性流量的重尾流检测和超点检测。重尾流节点是指在测量时间内发送或接收到大量报文或字节的节点，超点是指在测量时间内发送大量流的节点。基于重尾流或超点的活跃节点检测算法只是考虑报文数、字节数或流数等单维流量属性，没有综合考虑多种

作者简介： 李翔，（1988-），男，硕士研究生，E-mail: xli@njnet.edu.cn；程光，（1973-），男，教授，E-mail: gcheng@njnet.edu.cn.

流量特性对网络的影响。

在网络重尾流检测方面，2001年Nick Duffield [1]提出基于流大小的不等概率抽样重尾流测量方法以提高流统计估计精度。2002年Estan[2]提出的报文抽样和保留（PSH）算法使相同的计数器能够记录比NetFlow更多的报文。2006年Raspall[3]提出了一个比PSH算法能更精确地检测出高速网络重尾流的S3算法。2004年 Abhishek Kumar[4]使用最大似然技术估计bloom filter结构中的每个流大小。2006年Zhao[5]采用基于概要的数据流算法检测分布式的重尾流。2008年Edith Cohen[6]对网络测量过程中采用多次抽样和聚类所带来的误差进行理论估计。

在网络超点检测方面，在 2000 年以前 Snort[7]就使用哈希表直接记录所有网络流量 IP 地址的流数。2005 年 Venkataraman[8]提出两种基于流抽样的超点检测技术，其方法类似 Snort 也需要消耗大量的内存空间以存储抽样流。2005 年 Qi[9]提出抽样和数据流算法相结合的检测超点方法，但是该方法需要大量空间记录流标识信息。2006 年 Li[10]使用 IP 流随机聚类方法和子空间划分的方法识别异常发生的原因，类似的方法也可以用于超点异常的检测。2007 年 Noriaki Kamiyama[11] 提出了类似于 Venkataraman 算法的可以进行网络超点参数化的检测算法。

国内也有很多学者在研究网络超点检测技术。2008 年程光[12]提出了一种基于流抽样保留技术的自适应超点检测方法。该方法使用 Bitmap 进行流抽样，并对其进行流数上的 hash 冲突补偿。同时使用一个 IP Hash 链表来记录 IP 信息，通过不等概率抽样机制来淘汰非超点 IP 的自适应过程来控制 IP Hash 所消耗的资源。2008 年王洪波[13]基于 Bloom Filter 进行流抽样，根据 Bloom Filter 存在的 hash 冲突进行流抽样概率补偿，并通过实验与各种评价指标验证了该方法的有效性。

3. 基于多维属性的活跃节点检测算法

3.1 算法概述

由于重尾流或超点的检测方法一般只关注了主机节点在某个单维属性上的表现，不能综合的从多维属性方面考虑主机的活跃程度，可能会出现较高程度的误判。本文提出了一种基于主机节点多维属性的活跃节点检测算法，综合考虑了主机节点在检测时间段内的流记录数、报文数以及字节数，采用流记录和报文双重抽样的方法，同时将活跃节点的检测计算从一维扩展到多维。

根据上述算法，在活跃节点检测过程中，我们需要维护每一个被抽样到的 IP 的流记录、报文数以及字节数。因此，需要设计好流记录的状态信息和 IP 的统计信息。论文中使用 Bitmap 来维护流记录的状态信息，同时使用一个 IP Hash 表来记录每一个 IP 的统计信息。在流抽样过程中，若一个流记录被抽样，则利用 Bitmap 来检测当前流记录是否为新流。同时，由于 Bitmap 存在 hash 冲突，为了能够补偿流冲突带来的流数估计误差，所以采用相应的冲突补偿技术来对每一个被抽样的 IP 的流数进行补偿，以提高测量结果的精确性。在报文抽样的过程中，若一个报文被抽样，则根据报文的源宿 IP 去更新 IP Hash 表中对应 IP 节点的报文数和字节数。

在通过上述抽样过程得到测量数据后，需要读取 IP Hash 表中存储的初步统计数据，然后对每一个统计数据进行计算以判断当前 IP 是否为超点。论文参考超点和重尾流检测的计算方法，并将其从一维向多位扩展，充分考虑了 IP 在多维属性上的综合表现。下面将从抽样更新过程和结果计算方法两个方面来介绍本算法。

3.2 抽样更新过程

抽样更新过程共包含两个过程：对报文流数据的抽样过程和根据抽样结果对 IP Hash 表中统计信息的更新过程。图 1 是抽样更新流程图。

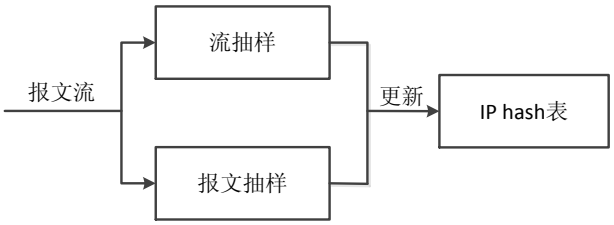

图 1 抽样更新流程图

在对报文进行抽样的过程中，每到达一个报文，都要进行如下的 3 个操作：

i. 获取报文的源宿 ip 和源宿端口，将报文的这 4 个属性作为流记录的关键字，利用流抽样函数对其进行抽样操作，若被抽样，则进行步骤 2；否则进行步骤 3；

ii. 利用 Bitmap 结构来检测当前报文所属的流是否为新流（相应位是否为 1）：若不是新流，则进行步骤 3。若是新流，先将当前流记录的信息插入到 Bitmap 中（将相应位置为 1），然后在 IP Hash 表中更新当前报文源宿 IP 节点的流记录数。

iii. 对当前报文进行系统抽样，若被抽样，则根据当前报文的源宿 IP，在 IP Hash 表中更新其对应 IP 节点的报文数和字节数。

按照活跃节点的定义，与其相关联的流数、报文数以及字节数都是超过阈值的。因此，如果流抽样与报文抽样函数足够随机，则有较大的概率出现原来为活跃节点的IP通过流抽样操作之后得到的流数应该比非活跃节点更多，通过报文抽样函数之后得到的报文数与字节数也应该比非活跃节点多。而通过流抽样和报文抽样两个操作后，能够大量减少程序处理的数据量，使得当前算法在高速网络中也有可用之处。

上述的3个操作中也包含了更新过程，即得到数据后对IP Hash表的更新操作。值得注意的是，在更新过程中采取的冲突补偿策略。假设Bitmap初始大小为m，且此时Bitmap中0的位数为k。当一个新流通过抽样后在Bitmap中进行存在性验证时，则这个新流被检测出（hash后得到的值在Bitmap中为0）的概率为q=k/m。这个值表明，在1/k中被抽样到的新流中，只有1个新流可能被Bitmap检测为新流，同时有1/q-1个新流由于Bitmap的hash冲突而未被检测出。因此，为了补偿由于冲突而丢失的流数，在Bitmap每检测出一个新流并在IP Hash表中进行流数更新时，在当前流的源宿IP统计节点上补偿1/q-1个流，以降低由于Bitmap存在的hash冲突所造成的误差。

3.3 计算方法

根据超点的定义，当一个主机与其他主机之间的流数超过一定的数值时，就会判定当前主机为超点。设定当前主机的流数为 R_f，预先设定的阈值为 T_f，则上述定义可以表示为：

$$\frac{R_f}{T_f} \geq 1 \qquad (1)$$

在活跃节点的检测中，如果采用节点的流数、报文数以及字节数等更多维属性来进行计算，我们可以将此计算方法扩展到了相应的多维，具体的计算公式如下：

$$\sqrt[n]{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{R_i}{T_i}\right)^n} \geq 1 \qquad (2)$$

其中 R_i 表示节点的属性 i 的测量值，T_i 表示预先设置的属性 i 的阈值。

在本文的实际计算中，我们采用了主机的流数、报文数以及字节数来进行计算，从而实际的计算公式如下：

$$\sqrt[3]{\frac{1}{3}\left(\left(\frac{R_f}{T_f}\right)^3+\left(\frac{R_p}{T_p}\right)^3+\left(\frac{R_b}{T_b}\right)^3\right)} \geq 1 \qquad (3)$$

其中 R_f、R_p 和 R_b 分别为单个主机（IP）经过抽样统计得到的流数、报文数和字节数，T_f、T_p 以及 T_b 分别为流数、报文数以及字节数的阈值。

在设定阈值的时候，如果采用固定值作为阈值，操作比较简单且容易实现，但结果不能很好的反映当前网络中的真实情况。若根据经验设定固定阈值，则在网络流量高峰期时，会因为阈值较小而产生较多数量的活跃节点（IP）；而在网络流量低谷时，又会因为阈值过大而导致活跃节点（IP）数量减少甚至检测不到结果。因此，本文采用了当前比较常用的比值法来确定每个属性的阈值。通过抽样统计，能够得到在一个抽样时段内被抽样到的总流数、总报文数以及总字节数，分别记为 $T_f^{'}$、$T_p^{'}$ 和 $T_b^{'}$，然后取每个属性的 N 分点值作为阈值，即流数、报文数和字节数的阈值为 $N*T_f^{'}$、$N*T_p^{'}$ 和 $N*T_b^{'}$。这样做的好处是能够根据被测网络的实际状态来进行检测，得到的结果可以很好的反映当时网络的真实情况。

4. 算法实验分析

4.1 实验数据集

在本节中，我们将在真实网络流量中运行上述算法代码，并从多个评价指标上来分析当前算法的具体性能。实验的数据来自江苏省计算机网络重点实验室采集的 CERNET 网络流量，日期为 2013 年 3 月 10 日，时长为 30 分钟。在实验中，我们认为的将这 30 分钟划分为 6 个测量子区间，每个子区间长度为 5 分钟，同时将这 6 个子区间标号为 1~6，方便表述和理解。表 1 为本次实验数据的相关信息，其中第 1 列为测量子区间的标号，第 2 列为当前子区间内出现的 IP 数，第 3 列为当前子区间内的流记录数，第 4 列为当前子区间内的报文数，第 5 列为当前子区间内的字节数，第 6 列为当前子区间内的超点数。在实验过程中计算超点时，我们将各个属性的阈值设定为对应属性统计值的 1%。

表 1 实验数据的概要信息

标号	IP 数	流数	报文数	字节数	超点数
1	76,528	727,378	16,098k	18.2G	39
2	82,488	747,323	15,829k	17.8G	39
3	72,062	755,310	14,418k	15.5G	39
4	68,705	754,987	14,001k	15.0G	36
5	83,997	787,752	14,616k	15.7G	38
6	90,224	800,765	152,57k	16.4G	34

4.2 实验结果分析

在进行实验结果分析之前，我们先定义两个比较常用的评价指标：活跃节点的误检率和漏检率。误检（false positive）是指本来不是活跃节点的主机（IP）在测量过程中被错误的判断成活跃节点，而误检率是被误检为活跃节点的非活跃节点主机数与活跃节点主机总数的比值。漏检（false negative）是指本来是活跃节点的主机在测量中没有被检测出来，定义漏检率为被漏检为非活跃节点的活跃节点主机数与总的活跃节点主机数的比值。

由于标号 1~6 的数据集从实际意义上来说，每一个数据的作用都与其他几个数据相似，因此，在实验结果分析中，我们采用了同时测量 6 个数据集并取结果平均值的方法。这样操作既不会影响对算法结果的具体分析，同时还能从更大的时间区间上来对算法进行观察。

图 2 为在固定流抽样比为 1/128 的条件下，不同报文抽样比下误检率与漏检率的结果图。从图中可以看出，算法具有较低的误检率，即将很少的非活跃节点判断成活跃节点，且这个特性基本上有流抽样比来决定，并没有随着报文抽样比例的变动而产生相应的变化。同时，漏检率会跟随报文抽样比的变化而变化，而这种变化趋势也没有呈现出线性规律。

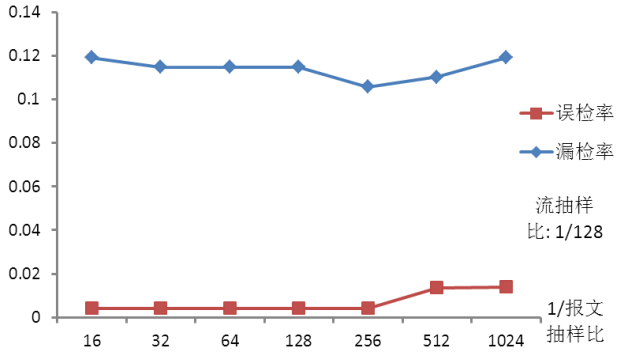

图 2 不同报文抽样比误检率与漏检率的变化图

图 3 为固定报文抽样比（1/128）下，不同流抽样比条件下误检率与漏检率的结果变化图。从图中可以看出，算法的误检率非常低，并且具有很好的稳定性，随着流抽样比的变化有轻微的变化。同时，在流抽样比增大时，算法的漏检率也随之增大，呈现了一种正相关的态势。

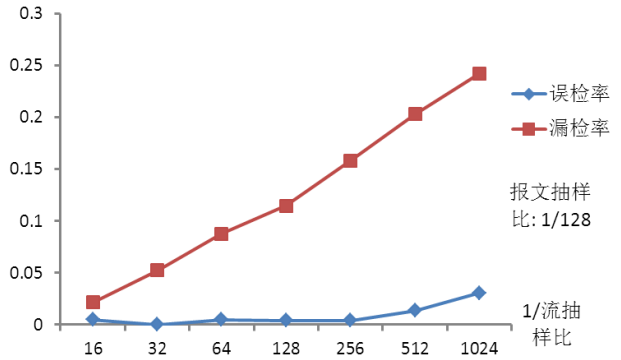

图 3 不同流抽样比误检率与漏检率的变化图

结合图 2 和图 3，我们可以得出如下结论：算法在误检率方面具有很好的稳定性，在流抽样或报文抽样比发生变化时，结果数据表现平稳，没有太大的波动；算法的漏检率基本由流抽样比决定，且与流抽样比呈现一种正相关的变化趋势，同时漏检率与报文抽样比基本无关。

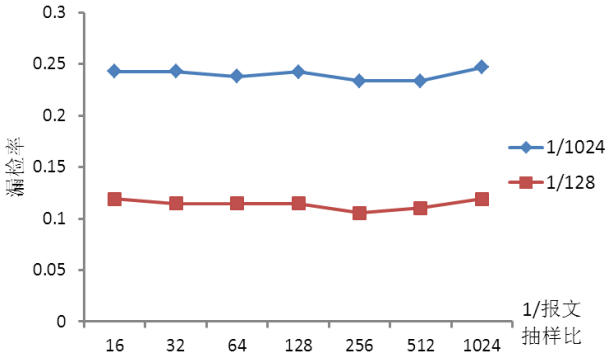

图 4 不同流抽样比漏检率比较图

图 4 为采用不同的固定流抽样比时，在变化的报文抽样比下算法漏检率的比较。从图中可以很明显的得出如下的结论：在采用相同的流抽样比时，不同报文抽样比对结果的影响很小；在采用相同的报文抽样比时，不同的流抽样对结果的影响十分明显。这也与图 2、图 3 中得出的结论一致。

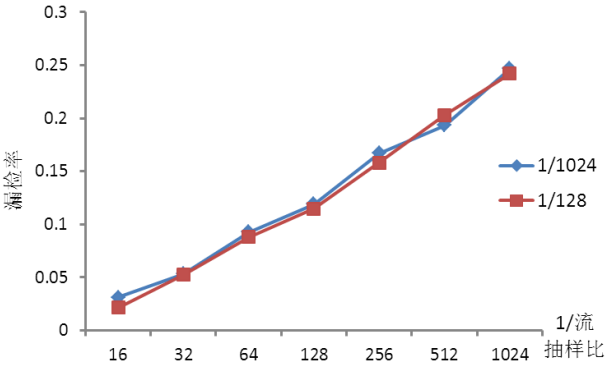

图 5 不同报文抽样比漏检率比较

图 5 为采用不同的固定报文抽样比时，流抽样比对算法漏检率的影响。很明显，当流抽样比发生变化时，算法的漏检率有很大变化。而在同一流抽样比的条件下，不同的报文抽样比对结果没有太多影响。

图 6 和图 7 是算法误检率方面的比较图。从图 6 中可以看出，当流抽样比从 1/128 增大到 1/1024 时，误检率有比较明显的变化，同时报文抽样比的发生变化时，算法的误检率也会随之产生正相关的变化，且波动的幅度很小。图 7 是固定报文抽样比后，观察不同流抽样比下误检率的变化。从图中可以看出，算法的误检率会随着流抽样比的变动产生明显的变化：流抽样比增大，误检率也增大。与此同时，相同流抽样比下不同报文抽样比的误检率之间并没有显著的差别。

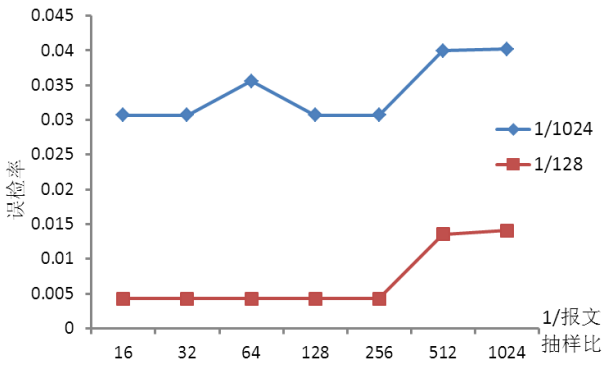

图 6 不同流抽样比误检率比较图

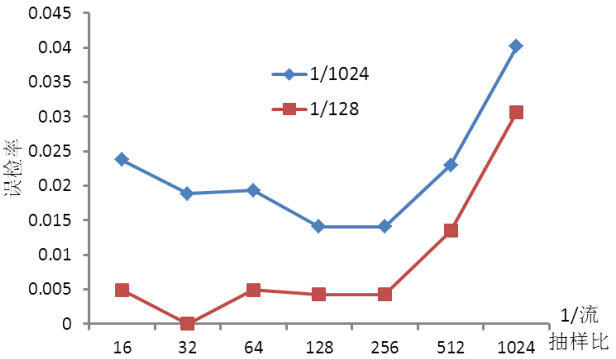

图 7 不同报文抽样比误检率比较图

综合上述的实验结果分析，我们可以得出如下的结论：在算法中，结果的误检率较低，且不会随着流和报文抽样比的变化产生很大的波动，表现十分稳定；而在漏检率方面，流抽样比基本决定了算法的表现：流抽样比越大，漏检率越高，基本上漏检率与流抽样之间呈现出一种正相关的态势。

5. 总结

活跃节点是指在一段测量时间内流数、报文数以及字节数等多个属性的综合表现超过一定阈值的主机节点。在当前的网络管理需求下，对活跃节点的研究具有非常重要的现实意义。而现有的测量方法一般都只注重于节点在某个单维属性上的表现，不能很好的考虑节点的多维属性。文章提出了一种基于 Bitmap 结构并综合节点多维属性的活跃节点检测算法，并通过 hash 冲突补偿来提高抽样之后的测量精度。通过实验结果分析，该算法在误检率和漏检率方面具有较高的精确度，能够在实际应用中发挥很好的作用。

参考文献：

[1] Nick Duffield, Carsten Lund, Mikkel Thorup. Charging from Sampled Network Usage [C], In Proc. ACM SIGCOMM Internet Measurement Workshop 2001.

[2] C. Estan, G. Varghese. New directions in traffic measurement and accounting [J], Computer. Commun. Rev., vol.32, no.4, pp. 323–338, 2002.

[3] Frederic Raspall, Sebastia Sallent, Josep Yufera. Shared State Sampling [C], Internet Measurement Conference 2006.

[4] Abhishek Kumar, Jun Xu, Li Li, et al. Space Code Bloom Filter for Efficient Traffic Flow Measurement [C], In Proc. of ACM/USENIX Internet Measurement Conference,, October 2003.

[5] Qi Zhao, Mitsunori Ogihara, Haixun Wang, et al. Finding Global Icebergs over Distributed Data Sets [C], to appear in Proc. of ACM Syposium on Principles of Database Systems (PODS) 2006, June 2006.

[6] Edith Cohen, Nick G. Duffield, Carsten Lund, et al. Confident estimation for multistage measurement sampling and aggregation [C]. In Proc. SIGMETRICS 2008: 109-120

[7] M.Roesch. Snort-lightweight intrusion detection for network [C]. In Proc. USENIX Systems Administration Conference, 1999.

[8] S. Venkataraman, D. Song, P. Gibbons, A. Blum. New streaming algorithms for fast detection of superspreaders [C]. In Proc. NDSS, 2005.

[9] Qi Zhao, Abhishek Kumar, Jun Xu. Joint Data Streaming and Sampling Techniques for Detection of Super Sources and Destinations [C]. In Proc. of IMC 2005, 77 – 90.

[10] Xin Li, Fang Bian, Mark Crovella, et al. Detection and Identification of Network Anomalis Using Sketch Subspaces [C] , In Proc. of Internet Measurement Conference, 2006.

[11] Noriaki Kamiyama, Tatsuya Mori, Ryoichi Kawahara. Simple and Adaptive Identification of Superspreaders by Flow Sampling [C]. 2481-2485, In Proc. INFOCOM 2007.

[12] 程光，龚俭，丁伟等.基于自适应抽样的超点检测算法[J]，中国科学 E 辑: 信息科学 2008 年第 38 卷 第 10 期: 1679 ~ 1696.

[13] 王洪波，程时端，林宇. 高速网络超连接主机检测中的流抽样算法研究[J]. 电子学报，2008 年第 6 卷第 4 期:809 ~ 818.