中国科学 E辑：信息科学 2008 年 第 38 卷 第 10 期：1679 ~ 1696
www.scichina.com info.scichina.com
《中国科学》杂志社
SCIENCE IN CHINA PRESS

基于自适应抽样的超点检测算法

程光*, 龚俭, 丁伟, 吴桦, 强士卿

东南大学计算机科学与工程学院, 南京 210096;
江苏省计算机网络重点实验室, 南京 210096
* E-mail: gcheng@njnet.edu.cn

收稿日期: 2008-06-03; 接受日期: 2008-08-21
国家重点基础研究发展计划(批准号: 2003cb314804)资助项目

摘要 超点是在一个测量时间区间内链接了大量源 IP(宿 IP)的宿 IP(源 IP), 实时超点检测对网络安全和管理具有重要意义. 现有的算法不能控制内存空间的使用和超点的测量精度, 论文提出了一个具有自适应抽样功能的超点实时检测算法. 该算法采用流抽样保留技术以减少非超点的测量并提高超点的测量精度; 设计一个数据流结构维护流记录, 并统计补偿 Hash 映射中产生的冲突; 提出一个基于不等概率的自适应策略以维护内存空间. 采用实际网络数据将论文的算法和其他算法进行分析比较, 实验和数学分析表明论文算法在资源可控性、测量精度等方面优于现有的其他算法.

关键词
超点检测
自适应过程
流抽样保留
冲突补偿

网络中的很多安全事件, 如分布式DDoS攻击、蠕虫扫描 [1]、端口扫描等对网络的安全和管理具有重要影响, 如何在高速链路上对这些事件进行实时检测是目前的研究热点和难点. 这些安全事件都具有类似的行为特征, 如: DDoS攻击的特点是受害者的宿IP收到来自于大量不同源IP的链接; 蠕虫扫描的特点是攻击者的源IP向大量的宿IP发送链接; 端口扫描的特点是扫描源IP向大量的宿端口发起链接; 这些攻击的共同特点是源IP/宿IP发送或接收到大量来自于不同宿IP/源IP的链接, 如Slammer扫描器 [2]每秒能够产生 30000 个扫描. 一个测量系统往往需要使用几百兆的内存以维护流记录信息, 实现对攻击源的检测. 但是DDoS等攻击会产生大量随机的源IP地址, 蠕虫扫描随机产生大量的宿IP地址, 因此现有设备难以在内存中存放通过链路的所有流和IP记录信息.

超点是在一个测量时间内链接了大量宿 IP(源 IP)的源 IP(宿 IP), 超点检测就是要检测出在测量时间内发送或接收到流数超过阀值的源 IP 或宿 IP. 为了能够检测超点, 需要维护流存在的状态信息和 IP 的状态信息. 超点检测的主要难点在于如何维护流和 IP 两个状态空间信息, 以及如何对这两个状态空间进行操作, 在有限测量资源情形下对网络流量超点精确、实时检测.

论文提出了一个具有自适应抽样功能的超点实时检测算法(superpoints detection algorithm on adaptive sampling, SDAS), 它由更新过程和自适应过程构成. 更新过程记录流和IP状态信息, 而自适应过程调整抽样参数和控制测量内存空间. SDAS算法的更新过程和自适应过程并行运行, 通过在更新过程和自适应过程之间的相互转换动态平衡IP Hash链表空间. 由于流和IP的数量巨大, 维护每个流状态信息和每个IP状态信息需要消耗大量的内存资源, 因此合理的流状态记录结构和IP记录方式是该算法设计的核心. SDAS算法设计中采用了多种网络测量技术, 如网络数据流算法 [3~5]技术维护流状态空间, 抽样保留技术 [6~8]减少非超点IP的测量并提高超点的检测精度, 不等概率的抽样技术 [9,10]减少自适应调整过程中超点被淘汰的概率等.实验和数学分析表明SDAS算法在自适应性、资源可控性、测量精度等方面优于现有其他算法, 能够实现高速网络超点高精度实时检测.

论文的第 1 节将讨论超点和自适应测量相关的研究状况; 第 2 节给出论文的自适应抽样超点检测算法, 分析 SDAS 算法的测量精度; 第 3 节分析 SDAS 的测量空间; 第 4 节采用实际网络数据分析比较 SDAS, Bitmap, Sampled 等超点检测算法的测量精度和性能; 最后总结全文.

1 相关研究

超点检测最简单的方法是在内存中维护测量时间区间内所有流和IP信息, 如: Snort[11]和Flowscan[12]. 由于网络中的流和IP数量巨大, 其算法需要消耗大量的内存空间维护流和IP信息, 只能用于低速链路的超点测量. 2005 年Venkataraman等 [13]提出超源IP检测算法, 他们采用流抽样算法随机抽样流, 以维护抽样到的所有流和IP信息. 2005 年Zhao等 [14]提出了抽样和数据流算法相结合的超点检测方法, 采用一个Bitmap结构维护流状态, 但是没有考虑IP空间的维护, 使得测量到的IP不能够完全记录在高速SRAM中. 另外这个Bitmap算法对每个IP的流都采用相同的抽样概率, 降低了超点流数估计的精度. 他们的第 2 个算法是采用比特矩阵来存储流和IP流数信息, 由于比特矩阵列向量大小决定了最大的IP流数的数量, 如果某些IP的流数较大, 其IP对应的列向量中全部或大多数比特被赋值而导致算法失效.

2004 年Moore等 [2]提出使用一个Bloom Filter结构进行流数测量; 2007 年Kamiyama等 [15]也提出了一个基于Bloom Filter的参数化的流数测量算法. 他们的算法都采用一个Bloom Filter结构维护流状态空间, 由于在测量中他们没有补偿Bloom Filter的Hash冲突对测量精度的影响, 同时Bloom Filter使用的空间较大, 因而难以在SRAM中维护Bloom Filter结构以适应高速网络链路测量. 另外, 文献[15]的算法在Hash链表空间大小的维护时只是简单地将链表中的后续结点丢弃.

目前的超点测量方法都采用了固定的测量参数, 但是在DDoS等突发流量情形下网络中流数将有很大的变化, 固定的测量方案将无法应对网络流量异常情形, 异常流量可能将会导致测量系统的崩溃. 现有的自适应测量研究主要是针对报文测量, 如Estan提出的具有自适应抽样能力的NetFlow(ANF)[16], 这个算法在每次调整时要重新抽样流缓存中的记录信息. 2005 年, Keys等 [17]提出了按照源IP、宿IP、源端口、宿端口分别聚类的热点流量检测技术. 这个算法采用了非淘汰结点的自适应算法, 其主要缺点是没有给出自适应过程中抽样参数的更新方案, 而且测量过程中需要使用较多内存以维护 4 个状态. 2007 年Cheng等 [18]提出了采用逐步递减测

量精度的网络流自适应测量技术. 同年Cohen等[19]提出了测量流量子集的自适应测量技术, 该算法要维护所有结点的大小及其抽样序号, 然后根据抽样序号进行抽样概率的自适应调整.

2 自适应超点检测算法

2.1 算法概述

假设一个测量时间区间Φ持续 t 秒, 一个 IP 在测量区间内产生或收到 m 个流, 我们定义一个阀值 m^*, 如果 $m \geqslant m^*$, 则认为这个 IP 是一个超点. 为了能够在高速链路上实时识别超点, 需要维护流和 IP 两个状态信息, 论文算法的核心是设计好流状态空间和 IP 状态空间, 以便控制 SRAM 内存资源的使用, 实现高速链路超点实时检测.

图 1 是自适应抽样超点检测算法结构图. 从图中可以看出, 论文的算法由更新和自适应两个过程构成. 更新过程的功能是记录流和 IP 信息, 自适应过程调整抽样参数并从内存中移出部分 IP 记录信息以控制测量内存空间. 更新过程由数据流和流抽样保留两个模块构成, 其中数据流模块的功能是记录流信息, 流抽样保留模块功能是更新 Hash 链表中的 IP 流数信息. 自适应模块则从 Hash 链表空间中移除 IP 记录信息, 并调整流抽样参数. 系统通过对更新过程和自适应过程并行运行, 平衡 IP Hash 链表空间的大小, 以实现超流的检测.

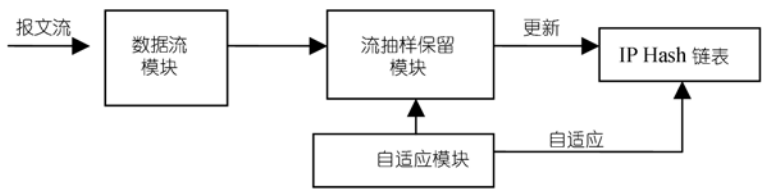

图 1 自适应抽样超点检测算法

SDAS 算法由 Bitmap 比特向量和 IP Hash 链表两个存储结构构成, Bitmap 的数据流结构存放流状态信息, 以判断到达的流是否是一个新流; IP Hash 表存放 IP 标识信息和 IP 流数信息. 由于在流检测过程中 Bitmap 可能存在冲突, 为了能够补偿流冲突带来的流数估计误差, 算法中考虑采用误差补偿技术补偿 Bitmap 冲突中带来的误差. SDAS 算法中的 Bitmap 具有两个方面功能, 首先是记录流存在信息, 根据 Bitmap 中的映射情形判断一个流是否已经存在; 其次采用 Bitmap 作为流抽样机制, 将 Bitmap 空间中的冲突映射成流抽样, 采用抽样误差估计技术实现冲突补偿.

为了节省 IP 状况空间, 算法需要能够尽可能过滤非超点的 IP, 同时抽样保留超点的记录. SDAS 算法采用流抽样保留技术, 对于每个流采用相同的概率抽样, 一旦一个 IP 在内存中被记录, 则其后续的所有流都将被抽样. 由于超点的流数较多, 非超点的流数较少, 因此采用流抽样保留技术将使得超点被测量的概率较大, 而非超点被测量的概率较少. 由于 IP Hash 空间的大小由抽样到的 IP 数量决定, 与测量的流数没有关系, 因此流抽样保留技术可以在不增加内存使用的情形下提高超点测量精度.

若测量时间没有结束, IP Hash 链表中的 IP 记录数目已经超过阀值, 系统没有更多的空间存储新到达的 IP 信息, 这说明更新过程中设定的抽样参数太大, 因此自适应阶段需要调整抽

样参数并从IP Hash链表中移出一些可能非超点的IP记录. 算法的目的是为了检测超点信息而不是所有IP信息, 因此在自适应抽样阶段需要考虑两个因素: 1) 从IP Hash链表中移出的IP记录信息应该是非超点信息; 2) 自适应过程中不减少已经测量到的超点信息. 由于在测量没有结束的时候, 除非IP的流数已经超过了阀值, 否则我们不知道一个IP是否是超点, 因此为了满足自适应测量的两个条件, 自适应过程采用不等概率的抽样技术统计地对非超点淘汰, 保留超点的测量精度. 下面将分别从更新过程和自适应过程两个部分讨论SDAS算法.

2.2 更新过程

更新过程涉及到两个模块: 数据流模块和流抽样保留模块. 数据流模块的功能是以很小的内存维护流状态信息, 流抽样保留模块的功能一方面减少对非超点的检测概率, 另一方面提高超点流数的测量精度.

由于数据流模块采用Bitmap(比特向量)结构存储流状态信息, 但是Bitmap检测一个新流是具有一定的冲突概率的, 如果不考虑冲突对流数测量的影响, 则测量到的流数会小于实际流数. SDAS算法在更新过程中采用了新流冲突补偿技术, 降低流数的估计误差. 设Bitmap的大小为 m, 其初始值为全部设置为0, 如果一个流标识的Hash值映射的比特位置为1, 则认为是一个已经存在的流, 否则如果映射的比特位置为0, 则认为是一个新流.

每个到达测量器的报文首先通过Bitmap数据流算法检测该流是否是一个新流, 如果是一个新流, 则进入下面的处理工作流程, 如果不是一个新流, 则丢弃该报文. 由于Bitmap算法只需要消耗很少的内存空间, 完全可以在SRAM中在线检测高速链路中的新流, 只有通过新流检测的流将会被后续过程处理. 每个流最多只有一个报文会被后续过程处理, 而其后续所有报文都将在新流检测过程中被丢弃, 大大减少了数据处理时间. 由于后续的抽样保留算法是采用基于Hash的流标识映射, 一个流如果没有被抽样, 其后续也不会被抽样, 因此没有被抽样流的所有后续报文都将不被处理. 为了提高超点流数的测量精度, 在更新阶段采取保留技术: 如果一个IP已经被记录在IP Hash链表中, 则其后续所有的流都将被记录, 避免后续流抽样过程中带来的测量误差.

如果一个新流被映射到Bitmap时为0的数目为 u, 则这个新流是可能被检测的概率为 $q=u/m$, 其含义是 $1/q$ 个新流中只有1个新流可能被Bitmap检测为新流. 为了补偿Bitmap冲突造成的 $1/q-1$ 流数损失, IP Hash链表更新到一个新流同时, 累加补偿 $1/q-1$ 的冲突损失. 设Hash链表中一个IP地址 A 的流数是 n, 当检测到 A 产生的一个新流时, Bitmap的冲突概率是 q, 则 A 更新后的流数为 $n+1+(1/q-1)=n+1/q$.

在抽样保留过程中, 采用流抽样保留技术减少非超点IP信息的记录. 由于超点IP比非超点IP有更多的流数, 假设每个流被抽样的概率是 p, 则流数为 n 的IP没有被抽样的概率为 $(1-p)^n$, 因此 n 越大, 其IP被检测的概率也越大. 通过调整流抽样概率 p 可以控制IP Hash链表中记录IP信息的数量. 如果定义的超点阀值流数等于 T, 则流数为 T 的IP没有被检测到的概率为 $(1-p)^T$. 如果定义一个漏检概率为 δ, 超点没有被检测到的概率 $(1-p)^T \leqslant \delta$, 因此为了满足这个条件, 流抽样概率不能小于 $p \geqslant 1-\delta^{1/T}$. 如果 T 取值为100, $\delta=0.0001$, 则

$p \geqslant 0.088$.

自适应过程要调整抽样概率以控制IP Hash链表中IP的数量. 为了实现自适应前后流抽样过程采用不同的抽样概率, IP Hash 链表的流数更新过程中以流数估计值代替测量值. 假设当前的流抽样概率为 p, 一个 IP 地址 A 的一个流被测量到, 那么 A 流数的一个无偏估计值是 $1/p$. 将 $1/p$ 作为 A 已经产生流数的初始值, 其含义是在流抽样概率 p 情形下, A 中 $1/p$ 个流才有 1 个流可能会被检测到.

更新阶段的算法代码如图 2. 算法第 2 行定义了一个大小为 m 比特的 Bitmap 结构记录流存在信息, 比特向量初始值为 0. 第 3 行定义一个变量 u 记录 Bitmap 中 0 比特的数量, u 初始值为 m. 如果 Hash 函数为 $h()$, 第 6 行对应的报文流标识为 pkt.flow_ID, 其 Hash 值为 $r=h$ (pkt.flow_ID). 第 7 行表明 Hash 值 r 映射到 Bitmap 比特位置的值 BM[r]如果为 0, 则说明是一个新流到达, 则继续执行下面的命令; 否则, 如果这个位置中对应的比特为 1, 则说明这个报文所属的流已经被测量过, 将这个报文直接丢弃. 第 8 行 BM 的 r 比特位置的比特赋值为 1, 标识该流已经被处理过. 由于每个流的所有报文具有相同的流标识, 因此同一个流所有报文流标识都映射到同一个 Bitmap 比特空间位置. 当一个流的一个报文被新流检测过程发现是一个新流以后, 其对应位置的Bitmap比特被设置为1, 于是其后续的所有报文都将不能通过新流检测, 每个流中只有第一个报文能够通过新流检测. 第 9 行判断新流所属的 IP 是否在 IP Hash 表中, 如果在 IP Hash 表中, 则进入第 10 行对 IP 流数信息更新. 由于一个 IP 只要被检测, 其所有后续的新流都被记录, 一个新流被 Bitmap 判断为新流的概率为 u/m, 因此第 10 行 IP 流数的更新估计值为 $1/(u/m)$. 如果第 9 行判断新流所属的 IP 不在 IP Hash 表中, 则进入第 11 行流抽样过程. 设一个 Hash 函数 $h'()$, 其映射的流标识空间为[0, 1], p 为流抽样概率, 如果 h'(pkt.flow_ID)< p 则抽样该流. 如果一个流通过第 11 行被抽样, 则在 IP Hash 链表中增加一条 IP 记录信息, 将其 IP 标识设置为 pkt.flow_ID. pkt.flow_ID 被检测到的概率由两个部分构成:

```
1.  Initialize
2.  BM[i] : = 0, i = 0, 1, … , m–1      //m 是 Bitmap 的大小
3.  u : = m     //变量 u 是用于跟踪 Bitmap 中 0 的数量，初始值为 m
4.  Measurement
5.  Each incoming packet pkt
6.    r := h(pkt.flow_ID);
7.    if (BM[r] == 0);              //n 是 k 个 Hash 函数映像位置上有多个 0
8.          BM[r]=1;
9.          if (pkt.SIP in AH)
10.               AH(pkt.SIP) =+ 1/(u / m)
11.         else if (h'(pkt.flow_ID) < p)     // 抽样流
12.               AH(pkt.SIP) = 1/(u / m) /p
13.         u =u–1
```

图 2 数据测量算法

1) 这个新流在 Bitmap 结构中被判断为新流的事件, 如果这个 Hash 函数映像流标识到 Bitmap

位置的值不为 1, 则这个流被判断为新流. 由于 Bitmap 空间中有 m 个比特位置, 其中 u 个位置为 0, 因此新流标识到 1 的概率为 $1-u/m$, 不为 1 的概率为 u/m. 2) 流抽样过程: 一个流被抽样的概率为 p. 因此结合这两个过程, 一个报文的 IP 如果第一次出现在 IP Hash 表中, 其抽样概率为 $p(u/m)$, 因此该 IP 流数的初始值的估计值为 $1/(u/m)/p$.

如果在新流检测过程的一个新流被检测到的概率为 $q=u/m$, 其中 u 是 Bitmap 中 0 的数量, m 是 Bitmap 向量的大小. 这个过程类似于一个新流按照 q 的概率进行随机抽样, 假设当 IP 的第一个被抽样流的 Bitmap 冲突概率为 $p_{bm}=u/m$, 流抽样概率 p_f, 则一个流被抽样的概率 $p=p_{bm}\cdot p_f$.

定理 1 更新过程中, 当一个 IP 被抽样到时, 则该 IP 已经通过流的数量估计均值为 $1/p_f(u/m)$, 是一个无偏估计, 其方差为 $(3+p_f\cdot u/m-2u/m-2p_f)/(u/m)^2 p_f{}^2$.

证明 流事件由两个独立事件构成: 新流事件 X 和流抽样事件 Y, 新流事件的抽样概率为 q, 其 $E(X)=1/(u/m)$, $D(X)=(1-(u/m))/(u/m)^2$. 流抽样事件 Y 的抽样概率为 p_f, 其 $E(Y)=1/p_f$, $D(Y)=(1-p_f)/p_f^2$. 因此一个 IP 被抽样时流数的期望为

$$E(XY)=E(X)E(Y)=1/p_f(u/m),$$

$$\begin{aligned}D(XY)&=E(X^2)E(Y^2)-(EX)^2(EY)^2=D(X)D(Y)+(E(X))^2D(Y)+(E(Y))^2D(X)\\&=(1-(u/m)/(u/m)^2)\cdot(1-p_f)/p_f^2+1/(u/m)^2\cdot(1-p_f)/p_f^2\\&\quad+(1-(u/m))/(u/m)^2\cdot 1/p_f^2\\&=(3+p_f\cdot u/m-2u/m-2p_f)/(u/m)^2p_f^2.\end{aligned}$$

定理 2 更新过程中, 一旦一个 IP 被测量, 其后续的所有流只要通过 Bitmap 新流检测过程都将被记录, 因此 IP 流数 f 的无偏估计和方差为

$$E[f]=1/p_f(u_Y/m)+\sum_{i=1}^{n}1/(u_i/m),$$

$$D[f]=(3+p_f\cdot u_Y/m-2u_Y/m-2p_f)/(u_Y/m)^2p_f^2+\sum_{i=1}^{n}(1-u_i/m)/(u_i/m)^2.$$

证明 设任何一个被测量的 IP 事件 A 由第一个被抽样流事件为 Y, 其后续各流被抽样的事件定义为 X_i 的独立事件构成.

$$E\left[Y+\sum_{i=1}^{n}X_i\right]=E[Y]+\sum_{i=1}^{n}E[X_i]=1/p_f(u_Y/m)+\sum_{i=1}^{n}1/(u_i/m),$$

$$\begin{aligned}D\left[Y+\sum_{i=1}^{n}X_i\right]&=D[Y]+\sum_{i=1}^{n}D[X_i]\\&=(3+p_f\cdot u_Y/m-2u_Y/m-2p_f)/(u_Y/m)^2p_f^2+\sum_{i=1}^{n}(1-u_i/m)/(u_i/m)^2.\end{aligned}$$

证毕.

式中 n 表示一个 IP 被测量到以后, 一共有 n 个流被测量. u_i 表示 IP 第 i 个流被测量到时

Bitmap 中有个 u_i 比特是 0. u_Y 表示 IP 被抽样测量到时, Bitmap 中有 u_Y 个比特是 0.

推论 1 假设自适应过程开始时 Bitmap 中有 u_t 个比特是 0, 定义 $p_t=(u_t/m)$. 则 IP 流量的估计方差可以近似为 $(3+(n-2)p_f-(n-1)p_f p_t-2p_t)/p_t^2 p_f^2$,

$$\begin{aligned}D[f] &\leqslant (3+p_f\cdot u_t/m-2u_t/m-2p_f)/(u_t/m)^2 p_f^2+\sum_{i=1}^{n}(1-u_t/m)/(u_t/m)^2\\ &=(3+p_f\cdot p_t-2p_t-2p_f)/p_t^2p_f^2+\sum_{i=1}^{n}(1-p_t)/p_t^2\\ &=(3+(n-2)p_f-(n-1)p_f p_t-2p_t)/p_t^2p_f^2.\end{aligned}$$

2.3 自适应过程

更新过程不能控制 IP Hash 表的大小, 也不能根据流量实际情形调整抽样参数, 因此需要使用自适应过程维护 IP Hash 表空间, 自适应地调整抽样参数. 若当测量时间没有结束, IP Hash 链表中 IP 记录数目已经超过阀值, 系统没有更多的空间存储新到达的 IP 记录, 说明更新过程中设定的抽样参数太大, 因此自适应阶段需要调整抽样参数并从 IP Hash 链表中移出一些可能非超点的 IP 记录. 当 IP Hash 表流数量超过一个阀值, 则重新计算流抽样参数, 并根据新的流抽样参数进行重抽样. 因此自适应过程需要 3 项工作: 1) 计算自适应调整阀值; 2) 计算新的流抽样比率; 3) 进行 IP Hash 空间记录重抽样.

假设更新阶段的流抽样概率为 $1/n$, 测量到 N 个 IP 记录的流数大小为 x_i, $i\in[1,N]$, 为了减少 IP Hash 链表的记录数量, 将抽样概率调整为 $1/z$, $1/z<1/n$. 由于抽样概率减少, IP Hash 链表中的部分 IP 在抽样概率 $1/n$ 下能够被抽样, 而在新的抽样概率 $1/z$ 就可能不能被抽样. 对于 N 个 IP 流数记录来看, $x_i, i\in[1,N]$, 由于这 N 个 IP 流数是实际 IP 流数的估计值, 如果 $x_i\geqslant z$, 即 $x_i/z\geqslant 1$, 则认为这个流在抽样概率 $1/z$ 情形下, 也有可能被抽样, 因此对于流数 $x_i\geqslant z$ 的 IP 记录不做调整. 而对于 $x_i<z$, 即 $x_i/z<1$, 则这个 IP 以 x_i/z 新的概率重新被抽样, 如果被抽样, 则该 IP 记录被保留, 并且设定该 IP 流数的估计值为 z; 如果这个 IP 没有被抽样, 则该 IP 记录被从 Hash 链表中删除.

下面要讨论的问题是如何计算新的抽样概率 $1/z$. 设 IP Hash 链表中的流数为 N, 每次自适应阶段需要移出的 IP 记录的数量为 αN, 需要根据淘汰数量 αN 计算出新的抽样概率 z. 为了能够计算阀值 z, 需要在测量器 SRAM 中维护一个 IP 流数分布数组. 设 $x_i, i\in[1,N]$ 个 IP 记录的长度分别为 $l_i, i\in[1,K]$, l_i 表示长度为 i 流数的 IP 数量. 设新调整的抽样概率初始值设定为 $z=n+1$, 则移出的 IP 记录数量为 $S=\sum_{i=1}^{z}(1-i/z)l_i$; 如果 S 小于 αN, 设定新的 $z=z+1$, 重新计算 S 值, 直到 $S\geqslant\alpha N$. 这时新的抽样概率设定为 $1/z$, IP Hash 链表中的 IP 记录按照 z 阀值进行重抽样. 如果 p_f 是自适应前流抽样概率, p_t 是在自适应前 Bitmap 的 Hash 冲突概率, 则自适应调整后的流抽样参数 p_f 设置为 $p_f=1/(z*p_t)$.

假设一个 IP 地址 A 流估计值为 $x=E(A)$, 其估计方差为 $D(A)$, 如果设定的 IP 流数阀值定义为 z, 则定义流抽样概率为 $1/z$. 如果 x 的流量大于等于 z, 则 A 不做任何处理, 其估计方差不变, 还是 $D(A)$. 如果 x 小于 z, 则按照 x/z 的概率抽样 A, 如果 A 被抽样, 则 A 的流数为 z, 如果 A 没有被抽样, 则将 A 的记录删除. 即(1)式表示流长为 x 的 IP 流重抽样概率.

$$p(Y \mid X)=\begin{cases}1, & x \geqslant z;\\ x/z, & x<z.\end{cases} \tag{1}$$

图 3 是自适应过程算法. 第 5 行是判断自适应过程的触发条件. 每次更新过程中如果在 IP 缓存中增加了一条记录, 将检测 IP 中的记录数量 AH.number 是否到达触发条件, 如果 IP 中的记录数量不小于定义的阀值 h, 则开始自适应过程. 第 6~8 行是淘汰阀值 z 的计算. 第 6 行是计算自适应调整参数的初始值 z. IP 缓存超过了阀值, 说明原来制定的流抽样参数设置过大, 需要降低原来的流抽样参数值. 由于原来的流抽样比率是 p, Bitmap 中在自适应调整开始时的“0”数量是 u, 其冲突率为 $1-u/m$, 因此一个流实际被抽样的概率的 $p(u/m)$, 因而我们定义初始的 z 为流实际被抽样概率的倒数. 第 7 行是判断 z 条件. 如果设置的 z 值使得淘汰的 IP 数量超过了预定的淘汰阀值, 则将这个 z 值作为淘汰阀值, 否则采用第 8 行设置新的 z 值继续计算. 第 9~14 行是对 IP 缓存中的记录采用淘汰阀值 z 进行淘汰. 其中第 10 行表明如果 IP 的流数 AH[i]小于 z 才进行淘汰操作, 否则 IP 流数保持不变. 第 11 行是对 IP 流数按照 AH[i]/z 的概率抽样, 如果被抽样, 则设置新的 IP 数量为 z, 否则第 14 行将这个 IP 记录从 IP Hash 表中删除.

```
初始过程和测量过程同测量过程算法, 初始过程补充以下定义:
1.   Initialize
2.   f[i]: = 0, i = 0, 1,…, t      //定义 IP 流数分布
3.   p, h, g                       //p 是流抽样比率, h 是自适应过程触发时 IP 缓存记录数量, g 是自
                                    适应过程中需要淘汰的 IP 记录数量.
4.   Self_Adaptive
5.   If (AH.number >= h)           //如果 IP 缓存中的流数量超过阀值, 则进入自适应过程
6.        z=⌈1/p·(u/m)⌉            //设置自适应开始的初始值, 其中 p 为流抽样概率, u 为 Bitmap
                                    中 0 的数量, m 为 Bitmap 空间的大小.
7.        while  g ⩽ ∑_{i=1}^{z-1} [1-i/z]·f(x)   //计算 z
8.        z=z+1                     //如果 z 不满足淘汰条件, 则继续设置新的 z 重新计算
9.        For each AH(i) in AH
10.            If AH[i]<z
11.                if 按照 AH[i]/z 的概率被抽样
12.                    AH[i] = z;
13.                Else if AH[i]没有被抽样
14.                    则从链表 AH 中将 AH[i]删除
15.       p=1/(z·u/m)               //计算新的流抽样概率
```

图 3　自适应过程的算法

定理 3　自适应过程后是不需要对 Bitmap 空间进行调整的.

证明　SDAS 算法在 IP Hash 空间中的记录数超出了阀值才会进行自适应调整, 降低流抽样的测量参数. 假设抽样概率为 p、流标识为 a、抽样函数 $f()$、抽样函数的取值为 0(不抽样)

或 1(抽样)、Hash 函数 $h()$取值为[0,1]. 则一个流被抽样的事件为

$$f(a)=\begin{cases}1, & h(a)\leqslant p;\\ 0, & h(a)>p.\end{cases}$$

定义一个流标识抽样集合$\{Ap,\ h(a)\leqslant p, a\in A_p\}$, 则定义 Ap 为在抽样概率 p 下能够被抽样的所有流标识集合. 假设自适应调整之前的抽样概率为 p_1, 其对应的流标识抽样集合为 Ap_1; 调整后的抽样概率为 p_2, 其对应的流标识抽样集合为 Ap_2. 由于自适应抽样调整后, 抽样参数降低, $p_1>p_2$, 从定义中我们很容易知道 $Ap_1\supset Ap_2$, Ap_2 是 Ap_1 的子集. 自适应调整之前没有被抽样的流记录, 在自适应调整之后也不会被抽样. 因此自适应前后的 Bitmap 不需要进行调整. 证毕.

定理 4 定义一个以 z 为函数的抽样概率函数 $p_z(\hat{f})=\min\{1,\hat{f}/z\}$, 一个流数估计值为 $\hat{f}$ 的 IP 被抽样的概率为 $p_x(\hat{f})$. 参数 z 是一个阀值, 对于估计流数大于等于 z 的 IP 必然被抽样, 而小于 z 的 IP 被抽样的概率为 $\hat{f}/z$. 设 Y_i 是一个独立随机变量, 定义

$$Y=\begin{cases}1, & p_z(\hat{f});\\ 0, & 1-p_z(\hat{f}).\end{cases}$$

则 $E[\hat{f}]=Y\cdot\hat{f}/p_z(\hat{f})$ 是 IP 流数 f 的无偏估计, 其方差为

$$D[\hat{f}]\leqslant\left(1/p_f(u_Y/m)+\sum_{i=1}^{n}1/(u_i/m)\right)\max\left\{\left(z-\left(1/p_f(u_Y/m)+\sum_{i=1}^{n}1/(u_i/m)\right),0\right)\right\}$$
$$+(3+p_f\cdot u_Y/m-2u_Y/m-2p_f)/(u_Y/m)^2p_f^2+\sum_{i=1}^{n}(1-u_i/m)/(u_i/m)^2+(n+1)/4.$$

证明 $E[\hat{f}]=E[Y\cdot\hat{f}/p_z(\hat{f})]=E[E[Y\cdot\hat{f}/p_z(\hat{f})\mid f]]=E[f]$.

其方差为

$$D[\hat{f}]=E[(\hat{\hat{f}}-f)^2]=E[(\hat{\hat{f}}-\hat{f}+\hat{f}-f)^2]$$
$$=E[(\hat{\hat{f}}-\hat{f})^2]+E[(\hat{f}-f)^2]+2E[(\hat{\hat{f}}-\hat{f})(\hat{f}-f)].$$

我们首先证明 $E[(\hat{\hat{f}}-\hat{f})(\hat{f}-f)]=0$, 即 $\hat{\hat{f}}-\hat{f}$ 和 $\hat{f}-f$ 近似正交, 由于 $\hat{\hat{f}}-\hat{f}$ 和 $\hat{f}-f$ 是表示两个独立测量过程的误差, 因此 $\hat{\hat{f}}-\hat{f}$ 和 $\hat{f}-f$ 是独立随机变量, $E[(\hat{\hat{f}}-\hat{f})(\hat{f}-f)]=E[\hat{\hat{f}}-\hat{f}]E[\hat{f}-f]=(E[\hat{\hat{f}}]-\hat{f})(E[\hat{f}]-f)=0$.

因为 $\hat{\hat{f}}$ 是 $\hat{f}$ 的无偏估计, $\hat{f}$ 是 f 的无偏估计. 因此我们有

$$D(\hat{\hat{f}})=E[(\hat{\hat{f}}-\hat{f})^2]+E[(\hat{f}-f)^2]=D[\hat{f}]+D[f].$$

$$D(\hat{f}) = D(y \cdot \hat{f} / p_z(\hat{f})) = \frac{\hat{f}^2}{p_z(\hat{f})^2} D(y) = \frac{\hat{f}^2}{p_z(\hat{f})^2} p_z(\hat{f})(1 - p_z(\hat{f})) = \hat{f} \max\{(z - \hat{f}), 0\}$$

$$= \left(1/p_f(u_Y/m) + \sum_{i=1}^{n} 1/(u_i/m)\right) \max\left\{\left(z - \left(1/p_f(u_Y/m) + \sum_{i=1}^{n} 1/(u_i/m)\right), 0\right)\right\}.$$

如果 $E[f] \geqslant z$, 则 IP 流数不做任何调整, 因此没有引入误差, 所以自适应调整对 IP 流数大于等于 z 的 IP 误差没有影响.

$$D[\hat{f}] = (3 + p_f \cdot u_Y/m - 2u_Y/m - 2p_f)/(u_Y/m)^2 p_f^2 + \sum_{i=1}^{n} (1 - u_i/m)/(u_i/m)^2.$$

因此经过自适应测量的 IP 估计方差为

$$D[\hat{\hat{f}}] = \left(1/p_f(u_Y/m) + \sum_{i=1}^{n} 1/(u_i/m)\right) \max\left\{\left(z - (1/p_f(u_Y/m) + \sum_{i=1}^{n} 1/(u_i/m)), 0\right)\right\}$$

$$+ (3 + p_f \cdot u_Y/m - 2u_Y/m - 2p_f)/(u_Y/m)^2 p_f^2 + \sum_{i=1}^{n} (1 - u_i/m)/(u_i/m)^2.$$

证毕.

式中 n 表示一个 IP 被测量到以后, 一共有 n 个流被测量. u_i 表示 IP 第 i 个流被检测到时, Bitmap 中有个 u_i 比特是 0. u_Y 表示 IP 被抽样测量到时, Bitmap 中有 u_Y 个比特是 0.

推论 2 假设测量结束时 Bitmap 中有 u_t 个比特是 0, 定义 $p_t = (u_t/m)$. 则流数量的估计方差为

$$D[\hat{f}] \leqslant (3 + (n-2)p_f - (n-1)p_f p_t - 2p_t)/p_t^2 p_f^2, \; E[f] \geqslant z,$$

$$D[\hat{f}] \leqslant \frac{(1 + np_f)(zp_f p_t - 1 - np_f) + (3 + (n-2)p_f - (n-1)p_f p_t - 2p_t)}{p_f^2 p_t^2}, \; E[f] < z.$$

3 算法空间分析

3.1 Bitmap 内存空间

假设一个大小为 b 的 Bitmap, 一个指定的流映射到一个比特上的概率是 p=1/b. 假设有 n 个流, 则一个比特没有被任何流映射的概率是 $p_z = (1-p)^n \approx (1/e)^{n/b}$, 在测量结束以后, 在 Bitmap 中没有被赋值的比特数量的期望值是 $E(z) = bp_z \approx b(1/e)^{n/b}$.

已知测量 n 个流, 为了保证测量精度, 测量结束以后, 我们希望 Bitmap 中没有被赋值的比特数量不小于总比特数的比率为 a, 因此可以建立 a 和 b 之间的关系

$$a \geqslant E[z]/b = p_z \approx (1/e)^{n/b}, \; b \geqslant n/\ln(1/a).$$

如果 b=n, 则 a=0.368, 如果 b=2n, 则 a=0.606, 如果 a=0.5, 则 b=1.44n, 如果我们控制冲突比率不高于 90%, 则 b=0.43n. 如果希望测量结束的时候, 冲突率不超过 50%, 则 Bitmap 的设置大小必须大于 1.44n, 每个流仅仅消耗不到 1.44 个比特的 SRAM 内存, 因此可以使用 SRAM 进行高速链路的流测量. 如平均每个流 10 个报文, 每个报文的平均长度是 500 Byte=

4000 bit, 因此 128 kB 的 SRAM 可以测量 0.7 M 个流, 支持 28 Gbit 流量的测量. 可以支持 OC48 链路 10 s 的满负载的连续测量, 对于 10 Gbps 的链路, 可以支持 3 s 的满负载的流量连续测量. Bitmap 中的冲突等同于流抽样, 在这种情形下类似于对网络流量采用了 50%~100%的概率进行流抽样, 在更新误差估计时最多会产生 50%概率的流抽样误差.

如果允许最低流抽样概率达到 10%, 即 a=10%, 则 b=0.43n, 则每个流仅仅消耗不到 0.43 个比特的 SRAM, 因此可以使用 SRAM 进行高速链路的流测量. 如平均每个流 10 个报文, 每个报文的平均长度是 500 Byte=4000 bit, 因此 128 kB 的 SRAM 可以测量 2.3 M 个流, 支持 90 Gbit 流量的测量. 可以连续 36 s 对 OC48 链路的满流量测量, 连续 10 s 对 OC192 的 10 Gbps 链路的满流量测量.

3.2 IPHash 表内存空间

通过自适应抽样技术, 算法保证产生固定数量的 IP 记录信息, 当 IP 流数占总流量超过一个比率时, 我们能够估计出其最坏情形下的相对误差.

定理 5 假设 Hash 表期望记录 IP 的数量为 N, T 为测量期间的流数总数, 超点阀值定义为 $T\cdot f$, 则超点相对误差最多不会超过 $\sqrt{1/(N\cdot f)}$.

证明 假设采用流抽样比率 N/T, 则其抽样的流数量为 N. 因此 IP 数量为 N 的抽样比率至少为 $p\geqslant N/T$. 超点流数阀值为 $f\cdot T$, 则其抽样流数的均值是 $p\cdot f\cdot T$, 方差是 $p(1–p)fT$. IP 流数的估计值是采用 $1/p$ 乘以测量到的流数, 因此估计的方差是 $(1/p^2)p(1-p)fT=(1-p)fT/p<fT/p\leqslant fT/(N/T)=fT^2/N$, 其标准方差最多为 $T\sqrt{f/N}$, 因此超点相对误差最多为 $T\sqrt{f/N}/fT=\sqrt{1/(N\cdot f)}$. 证明完毕.

如果已知相对误差 ε 和超点阀值比率 f, 则 IP 空间大小设置为 $N\geqslant 1/(\varepsilon^2 f)$. 如果设置 f=0.01, 相对误差控制 10%, 则 N=10000.

在自适应调整过程中, 我们需要检测和处理 IP 空间中所有的记录信息, 但是在自适应期间如果采用一个缓冲存储新到达的流信息是不符合实际情形的. 因此在实际自适应过程中, 更新过程和自适应过程是同步进行的, 这样必须保证自适应过程中淘汰 IP 结点的数量应该要比新的 IP 产生的速度要快, 即: 为了避免自适应过程的淘汰出现落后现象, 必须保证自适应过程中释放新的 IP 空间数量高于 IP 产生的数量.

由于在自适应调整过程中, 设置的 IP 抽样概率和其流数成正比, 对于大于阀值的 IP 记录不进行调整. 假设 IP 缓存空间记录为 L, 淘汰后的 IP 缓存记录为 N, 也就是每次淘汰的 IP 记录的数量为 L–N. 假设一条 IP 记录访问一次需要消耗的时间是 t_1, 重抽样计算一个 IP 需要消耗的时间为 t_2, 淘汰每个 IP 需要消耗的时间为 t_3. 每次自适应调整过程中, 被访问 IP 的数量为 L, 被重抽样计算的 IP 数量最多可能为 L, 淘汰的 IP 数量为 L–N. 因此自适应过程需要消耗 CPU 的总时间为

$$t=t_1\cdot L+t_2\cdot L+t_3(L-N)=(t_1+t_2+t_3)L-N\cdot t_3.$$

定理 6 假设 IP 空间记录为 L, 淘汰后的 IP 缓存记录为 N, 也就是每次淘汰的 IP 数量为

L–N, 则 $L \geqslant N(l - t_3 \cdot B \cdot p)/[l - (t_1 + t_2 + t_3)Bp]$. 其中一个网络链路速率为 Bbit/s, 流的平均长度为 l, 流抽样概率为 p.

证明 在时间 t 内总共有流到达的数量为 F, $F = t \cdot B / l$ 个流, 当前抽样到的流为 Fp. 极端情形下, 在自适应期间所有被测量到的流都是新的 IP, 因此在自适应淘汰期间其新 IP 记录产生的数量必须小于等于淘汰 IP 记录数, 即 $F \cdot p \leqslant L - N$, $t \cdot B / l \cdot p \leqslant L - N$,

$$[(t_1 + t_2 + t_3)L - N \cdot t_3] \cdot B / l \cdot p \leqslant L - N,$$

$$L \geqslant \frac{N(l - t_3 \cdot B \cdot p)}{[l - (t_1 + t_2 + t_3)Bp]}.$$

其中要求: $l - t_3 \cdot B \cdot p > 0$, $l - (t_1 + t_2 + t_3)Bp > 0$. 证毕.

假设一个 OC192 的链路, 带宽为 10 Gbps, 在第一次调整以后, 由于淘汰阀值的最小值为 2, 因此流抽样概率 p 至少小于等于 1/2. 如果平均每个流 10 个报文, 每个报文 500 Byte, 每个流都对应唯一一个 IP, 在 SRAM 内存中, t_1, t_2, t_3 的处理访问时间在 10 ns 以内能够完成, 则 IP 记录数量至少为 $L \geqslant 1.003N$. 如果设置 f = 0.01, 相对误差控制 10%, N = 10000 情形下, 则 L 的数量为 10030.

考虑到一个 IP 记录中 IP 地址 32 bit, 流数需要 16 bit, 指针 16 bit, L=11000, 设置 IP Hash 空间大小为 64/8*11000=86 kB, 为了维护这些IP记录, 如果设置头指针 10240, 指针大小 16 bit, 因此头指针需要空间 10240*16/8=20kB. 前面分析的 Bitmap 空间 128 kB, 因此为了实现超点检测, 我们维护 Bitmap 和 IP Hash 的空间需要内存 128+86+20=234 kB, 这样完全允许采用低容量的高速 SRAM 进行超点的实时检测.

为了防止自适应过程中出现如 DDoS 突发流量, 某一瞬间到达的 IP 数高于释放的 IP 数量, 算法可以设置 IP 的空间比自适应淘汰阀值的 IP 空间大. 比如淘汰阀值空间大小为 K, IP 缓冲空间大小为 L, 则 IP 空间中有 L–K (如 30) 的多余 IP 记录空间用于缓存突发流量. 这个过程可以通过建立 Markov 模型进行分析, 由于文章的长度限制, 这部分内容将不进一步讨论.

4 SDAS 算法实验分析

在这一节中, 我们将使用 NLANR 和 CERNET 网络数据对论文的 SDAS 算法进行评估. 评估部分分为 3 个部分: 测试环境的描述; SDAS, Sampled, Bitmap 算法比较; SDAS 算法的性能、参数分析等.

4.1 实验说明

4.1.1 测试数据集

论文采用NLANR数据 [1)]和江苏省计算机网络重点实验室采集的CERNET网络流量 [2)]数据进行测试. 第 1 部分NLANR数据(IPKS0 和IPKS1)是 2004 年 6 月 1 日NLANR从Internet2 的 Indianapolis Abilnene路由器通往Kansas城的OC192c POS两个方向链路上收集. 第 2 部分数据

1) NLANR PMA: Special Traces Archive. http://pma.nlanr.net/Special/

2) 江苏省计算机网络技术重点实验室 IP TRACE 发布. http://ntds.njnet.edu.cn/home/intro.php

是 2005 年 11 月 10 日 18 点 48 分从CERNET华东北地区网络主干节点OC48 链路上下载, 其中 C5 数据持续 5 s, C10 数据持续 10 s, C30 数据持续 30 s, C60 数据持续 60 s. 表 1 是测试数据的相关概要信息, 其中第 1 列是采用 6 组数据的名字, 第 2 列是数据中的源IP数, 第 3 列表示测试数据中流的数量, 第 4 列是超点阀值定义为总流数 1‰时对应的流数, 第 5 列表示符合定义超点阀值的超点数量, 最后一列是数据中的报文总数.

表 1 实验中使用的流量日志

日志	源 IP 数	流数	流数 1‰的阀值	超点数	报文数
IPKS0	31235	833227	833	92	28698731
IPKS1	47341	597342	597	90	28253549
C5	52746	180650	180	67	3084047
C10	66888	248421	248	76	6168094
C30	99369	468977	468	85	18504282
C60	128350	759041	759	85	37008564

实验中采用的流是按照报文字段四元组{源 IP, 宿 IP, 源端口, 宿端口}聚类的, 即将相同{源 IP, 宿 IP, 源端口, 宿端口}的报文集合称为流. 采用源 IP 为聚类点标识定义, 将具有相同源 IP 的流合并成 IP 点. 超点是流数超过了一个事先定义阀值的 IP, 实验中将超点阀值定义为测量过程中总流数的 1‰.

4.1.2 测量测度

下面给出用于对算法评估的 3 个相关测度: 超点测量相对误差 Relative.err, 超点测量平均相对误差 Average.err, 超点漏报相对误差 Super.err.

超点流数测量相对误差 Relative.err, Relative.err 是超点流数的测量值和实际值之间的误差绝对值和超点实际流数之间的比值. (2)式中 $\hat{F}_s$ 表示超点 s 的流数测量值, F_s 是超点 s 的流数实际值.

$$\text{Relative.err} = \frac{\left|\hat{F}_s - F_s\right|}{F_s} \times 100\%. \tag{2}$$

超点测量平均相对误差 Aver.err, Aver.err 是所有超点相对误差的平均值, 其中(3)式中 n 是超点的个数, $\hat{F}_i$ 表示超点 i 的流数测量值, F_i 是超点的 i 流数实际值.

$$\text{Average.err} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|\hat{F}_i - F_i\right|}{F_i}. \tag{3}$$

超点漏报误差 Super.err, Super.err 用于评估超点检测算法漏报超点的性能指标, (4)式中 $\bar{F}_{\max}$ 表示没有被测量到的超点的流数最大值, T 表示超点检测的阀值.

$$\text{Super.err} = \frac{\bar{F}_{\max} - T}{T}. \tag{4}$$

4.2 算法性能比较

4.2.1 比较算法

论文将 SDAS 自适应算法和第 2 节中介绍的 Venkataraman 流抽样的超点检测算法(Sampled)、Qi 的比特向量算法(Bitmap)进行比较. 论文中的 SDAS 和 Bitmap 算法都采用了比特向量存储流记录信息, 在算法性能比较中, 所有的实验采用的比特向量均设置一个 128 kB, 即 128*8*1024=1048576 bit. SDAS 算法中 IP 空间缺省大小设置 11000, 每次自适应过程淘汰数量为 1000, 流抽样概率初始值 p=1. Bitmap 算法和 Sampled 算法在检测 4 组 CERNET 数据中, 其流抽样比率设置为 1/8, 对于 IPKS0, IPKS1 数据设置流抽样比率为 1/4.

4.2.2 算法测量精度的比较

图 4 是 C5, IPKS0 两组数据采用 SDAS, Sampled, Bitmap 三种算法检测的测量值和实际值之间的关系图, 其中 X 轴表示的超点实际流数, Y 轴表示检测算法的测量值, 超点检测阀值定义在表 1 中, 设超点阀值为 T, X 轴读数的取值范围是[T, 2T], C5 数据 X 轴取值是[180, 360], IPKS0 数据 X 轴取值是[833, 1666]. 其中中间对角线表示对应流数的实际值, 如果图中分布点接近中间的对角线, 则表示测量值接近实际值. C5 中间对角线上面的虚线表示测量值比实际值大 10%, 中间对角线下面的虚线表示测量值比实际值小 10%, 两对角线中间的点表示测量误差在 10%以内. 从图 4(a)中可以看出, SDAS 检测算法的测量误差都在两条虚线之间, 表明算法的检测误差小于 10%. 而 Sampled 算法和 Bitmap 算法的检测超点有很多落在两条虚线以外, 这些落在虚线以外的点表示测量误差大于 10%, 离中心线越远, 表示误差越大.

图 4(b)的 IPKS0 中间对角线上面的虚线表示测量值比实际值大 5%, 中间对角线下面的虚线表示测量值比实际值小 5%, 两对角线中间的点表示测量误差在 5%以内. 图 4(b)表明 SDAS 检测算法的测量误差大多都在两条虚线之间, 表明算法的检测误差小于 5%. 而 Sampled 算法和 Bitmap 算法的检测超点有很多落在两条虚线以外, 这些落在虚线以外的点表示测量误差大于 5%, 离中心线越远, 表示误差越大.

图 4 表明 SDAS 算法检测误差远好于 Sampled 和 Bitmap 算法, Sampled 和 Bitmap 两个算法的检测精度相近. 由于 CERNET 数据中, Bitmap 和 Sampled 数据的流抽样采用 1/8, 而 IPKS 数据中流抽样采用 1/4, 因此图 4(a)测量误差采用 10%, 而图 4(b)测量误差设置为 5%.

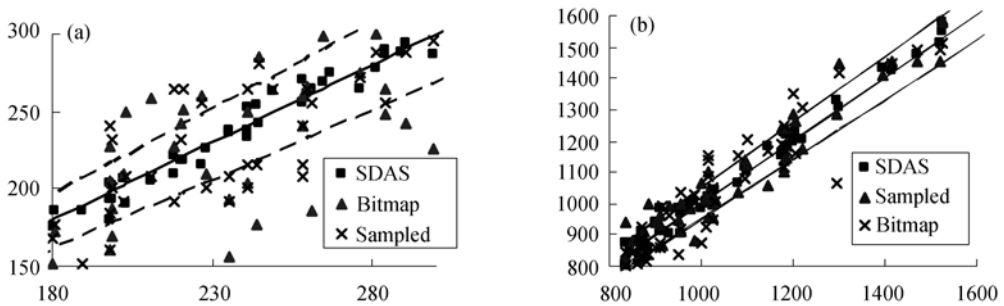

图 4 超点测量值和实际比较

(a) C5 数据; (b) IPKS0 数据

图 5 C60 数据中流数在前 60 位的超点实际值, SDAS 测量值, Sampled 测量值和 Bitmap 测量值等四个值之间的比较, 图中 *X* 轴是超点流数实际值大小的排序, *Y* 轴是超点流数的测量值和实际值. 图 5 表明, SDAS 的测量值更好地接近实际值, 说明对于这些超点 SDAS 算法更为精确.

图 6 是 SDAS, Bitmap, Sampled 算法对于超点阀值设置为 1‰ 超点流数估计值在 1%, 5%, 10%, 20%, >20%的超点数占总超点数的比率, 其中 *X* 轴是超点测量相对误差, *Y* 轴是误差分布. 从图 6 中可以看出, SDAS 算法的估计误差大多数小于 5%, 其估计误差远远好于 Bitmap 和 Sampled 算法. 其中 Sampled 算法的精度略好于 Bitmap 算法, 这其中原因是 Bitmap 算法采用一个比特矩阵维护流记录会引入冲突.

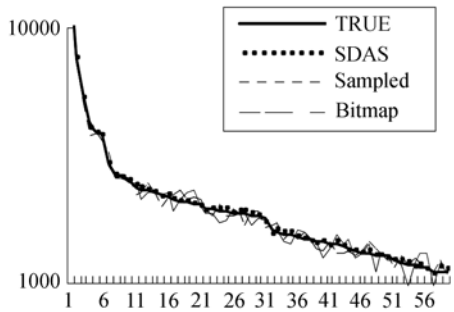

图 5 C60 数据的流数前 60 的超点

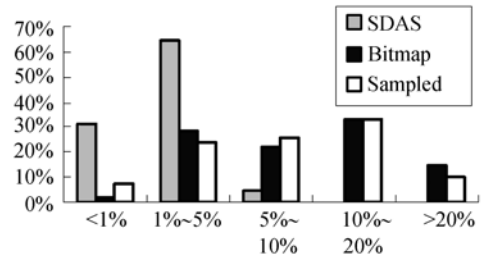

图 6 C5 数据的超点估计误差分布比较

图 7 是表 1 中 6 组数据采用 SDAS, Bitmap, Sampled 等 3 种超点检测算法测量到超点相对误差小于 5%的比率、超点检测算法中 IP 内存使用情形之间的关系. 图 7 中 *X* 轴是对应 6 组测试数据, 左 *Y* 轴是超点相对测量误差小于 5%的比率, 其对应的数据是 SDAS 测量误差(A-E), Sampled 测量误差(S-E), Bitmap 测量误差(B-E), 右 *Y* 轴是测量算法中使用的 IP 空间大小, 其对应的数据有 SDAS 算法的 IP 空间(A-IP), Sampled 算法的 IP 空间(S-IP), Bitmap 算法的 IP 空间(B-IP). 从图 7 可以看出, SDAS 算法的超点检测相对误差小于 5%的比率大于其他的两种算法, 而其 IP 空间使用的内存使远小于其他两种方法. 由于 Bitmap 算法和 Sampled 算法都采用相同的测量参数, 因而这两个算法在测量误差和 IP 空间使用上都比较接近. 但是 Sampled 算法需要大量的内存空间维护被抽样的流记录完整信息, Bitmap 算法和自适应算法只需要少量的比特向量维护流存在信息.

图 8 是超点平均相对误差, 其中 *Y* 轴是超点平均相对误差, SDAS 超点平均相对误差小于 Sampled 和 Bitmap 算法, Sampled 算法的测量平均相对误差略小于 Bitmap 算法.

从上面的分析来看, 由于 SDAS 算法采用了自适应抽样调整、流抽样保留、测量冲突误差补偿等技术, SDAS 算法的超点测量精度优于 Sampled 算法、Bitmap 算法, 而其使用的内存空间小于 Sampled 和 Bitmap 算法. 在设置相同抽样参数情形下, 由于 Bitmap 算法在测量流时候采用比特向量维护流记录存在冲突误差, Sampled 算法的测量精度略好于 Bitmap 算法, 而 Bitmap 算法使用的内存小于 Sampled 算法.

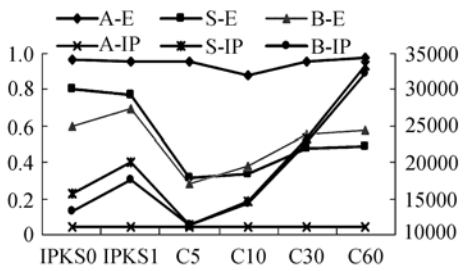

图 7　数据和误差、IP 空间的关系

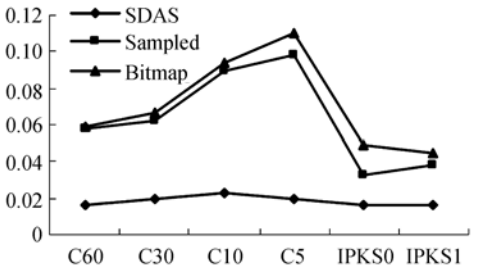

图 8　超点平均相对误差比较

4.3　自适应超点检测算法性能分析

4.3.1　测试数据和算法精度之间的关系

设置一个 128 kB 大小的比特向量 m=128*8*1024=1048576, IP 空间大小设置 11000, 每次淘汰数量 1000, 流抽样概率初始值 p=1. 在算法参数不变的情形下, 采用表 1 的 CERNET 四组 C5, C10, C30, C60 进行处理数据分析, 比较它们的误差关系. 表 2 是不同数据日志的 SDAS 自适应调整次数和最后一次调整的自适应阀值 z. 从表 2 我们可以知道, 随着测量数据的增加, SDAS 算法的自适应调整次数和淘汰阀值也会相应增大, 以适应自适应测量的需求.

表 2　SDAS 自适应调整次数和抽样阀值 z

	CERNET5s	CERNET10s	CERNET30s	CERNET60s
调整次数	8	9	12	16
最后 z	11	13	21	40

图 9 是 SDAS 算法中测量到超点阀值为总流数 1‰ 的超点测量误差的分布, <1%表示超点测量相对误差小于 1%的超点所占的比率, 1%~5%表示超点相对测量误差大于等于 1%且小于 5%的超点所占比率, 5%~10%表示超点相对误差大于等于 5%且小于 10%的超点所占的比率. 从实验结果来看, SDAS 算法在处理这 4 组不同时间粒度的流量数据, 均能够保证其测量误差在 10%以内, 而且相对测量误差分布接近, 实验结果表明测量数据量的增大对超点流数的估计误差影响不大. 从图 9 对不同时间粒度的数据测试结果表明, SDAS 具有很好的测量数据的自适应能力, 对于不同数据流的测试数据能够很好地进行自适应维护聚类点空间, 并能保证测量的超点流数精度没有明显降低.

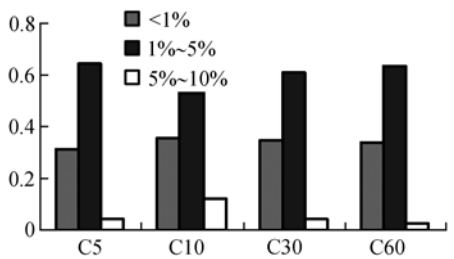

图 9　不同测试数据的相对误差比较

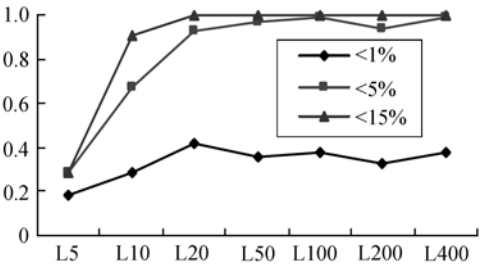

图 10　IP 空间和误差之间的关系

4.3.2 IP 空间和测量精度的关系

对于 C5 日志数据, 我们分别设置 IP 空间大小 L 分别设置为 500(L5), 1000(L10), 2000(L20), 5000(L50), 10000(L100), 20000(L200), 40000(L400), 每次淘汰的 IP 数为 IP 空间的 10%, 分析其自适应参数 z 调整的变化情形, 以及误差之间的关系比较. 图 10 是 IP 空间和误差之间的关系, 从图中可以看出, 当 IP 空间大于 2000 结点以后, IP 空间对超点误差的估计影响将很小. 除了 IP 空间设置为 2000 情形下出现误差超过 10%的点, IP 空间大于 2000 以上的情形下, 超点流数估计误差没有明显受 IP 空间大小的影响.

4.3.3 自适应淘汰 IP 数和测量精度的关系

下面采用 C5 数据分析, 淘汰结点数和误差、自适应次数之间的关系. 设置 IP 空间 L 为 11000, 淘汰结点数设置为 1000(R1), 2000(R2), 3000(R3), 4000(R4), 5000(R5), 6000(R6), 7000(R7), 8000(R8), 9000(R9), 10000(R1).

其中 R1~R10 表示淘汰结点数分别为 1000~10000, Time 是自适应的次数, Error 是所有超点流数估计绝对误差的平均值, “1<%”表示相对误差小于1%的超点所占的比率, “5<%”表示相对误差小于 5%的超点所占的比率. 图 11 可以知道, 每次自适应淘汰结点越多, 则自适应的次数将越少, 当淘汰结点为 10000 时, 测量误差将有较为明显的增加, 而在等于 9000 个淘汰结点时, 其误差没有明显的关系.

图 12 是 SDAS 算法测量 C5 数据过程中的自适应调整次数、自适应调整最后的测量参数 z 和测量数据的流数、IP 数之间的关系, X 轴是表 1 中的测试 4 组数据, 左 Y 轴是测量次数和自适应调整最后参数 z, 右 Y 轴表示流数和 IP 数, 图标“Time”表示自适应的次数, “z”表示 SDAS 算法最后一次自适应调整参数, “Flow”表示测量数据的流数, “SIP”表示测量数据中源 IP 的数量. 从图 12 中可以看出, 随着测量流数和 IP 数量的增加, SDAS 的自适应次数和调整参数也相应增大.

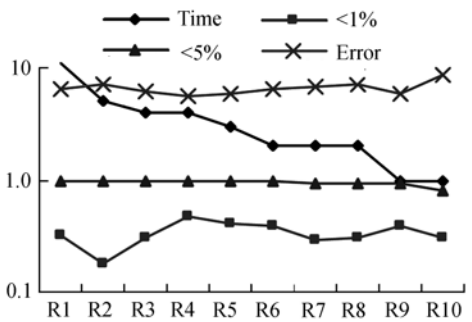

图 11 淘汰结点和误差、自适应次数的关系

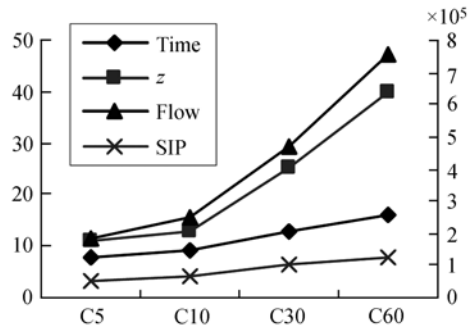

图 12 自适应参数之间的关系

5 结论

高速链路的网络流量超点实时检测是网络安全和网络测量中的重要研究价值, 而超点检测问题的核心是超点检测算法的内存空间和测量精度. 论文采用了数据流算法、冲突误差补

偿、抽样保留、基于不等概率抽样等多种自适应技术, 设计了一种基于自适应抽样的超点检测算法 SDAS. 和其他类似的算法相比, SDAS 算法能够自适应调整内存的消耗、提高超点流的测量精度、减少冲突所引起的估计误差、能够使用多个抽样测量参数. 利用实际流量数据对 SDAS 算法和其它算法进行了比较和分析, 实验和数学分析表明 SDAS 算法在自适应性、资源可控性、测量精度等方面优于现有的 Sampled 和 Bitmap 等算法, 能够实现高速网络超点高精度实时检测.

参考文献

1 Yang F, Duan H X, Li X. Modeling and analyzing of the interaction between worms and antiworms during network worm propagation. Sci China Ser F-Inf Sci, 2005, 48 (1): 91—106

2 Moore D, Paxson V, Savage S, et al. Inside the slammer worm. Secur Privacy Mag, 2003, 1(4). 33—39 [DOI]

3 Abhishek K, Xu J, Li L, et al. Space code bloom filter for efficient traffic flow measurement. In: Proceedings of ACM/USENIX Internet Measurement Conference. Miami: ACM Press, October 2003. 167—172

4 Abhishek K, Minho S, Xu J, et al. Data streaming algorithms for efficient and accurate estimation of flow size distribution. In: ACM Sigmetrics. New York: ACM Press, June 2004. 177—188

5 Estan C, Varghese G, Fisk M, et al. Bitmap algorithms for counting active flows on high speed links. In: Proc Internet Measurement Conference. Miami: ACM Press, 2003. 153—166

6 Estan C, Varghese G. New directions in traffic measurement and accounting. In: SIGCOMM. Pittsburgh: ACM Press, 2002. 323—336

7 Frederic R, Sebastia S, Josep Y. Shared state sampling. In: Internet Measurement Conference. Rio de Janeriro: ACM Press , 2006. 1—14

8 Bruno R, Towsley D, Ye T, et al. Fisher information of sampled packets: an application to flow size estimation. In: Internet Measurement Conference. Rio de Janeriro: ACM Press, 2006. 15—26

9 Duffield N, Lund C, Thorup M. Charging from sampled network usage. In: ACM Sigcomm Internet Measurement Workshop. San Francisco: ACM Press, 2001. 245—256

10 Duffield N, Lund C, Thorup M. Estimating flow distributions from sampled flow statistics. In: SIGCOMM. Karlsruhe: ACM Press, 2003. 325—336

11 Roesch M. Snort-lightweight intrusion detection for network. In: Proc USENIX Systems Administration Conference. Seattle: USENIX Assoc, 1999. 229—238

12 Plonka D. Flowscan: a network traffic flow reporting and visualization tool. In: USENIX LISA. New Orleans: USENIX Assoc, 2000. 305—317

13 Venkataraman S, Song D, Gibbons P, et al. New streaming algorithms for fast detection of superspreaders. In: Proc NDSS. San Diego: Internet Society, 2005

14 Zhao Q, Kumar A, Xu J. Joint Data Streaming and Sampling Techniques for Detection of Super Sources and Destinations. In: IMC. Berkeley: ACM Press, 2005. 77—90

15 Kamiyama N, Mori T, Kawahara R. Simple and adaptive identification of superspreaders by flow sampling. In:Infocom. Anchorage: IEEE Press, 2007. 2481—2485

16 Estan C, Keys K, David M, et al. Building a better netflow. In: SIGCOMM. Portland: ACM Press, 2004. 245—256

17 Keys K, David M, Estan C, et al. A robust system for accurate real-time summaries of internet traffic. In: ACM Sigmetrics. Banff: ACM Press, 2005. 85—96

18 Cheng G, Gong J. A resource-efficient flow monitoring system. IEEE Commun Lett, 2007, 11(6): 558—560 [DOI]

19 Cohen E, Duffield N, Kaplan H. Algorithm and estimators for accurate summarization of internet traffic. In: IMC. San Diego: ACM Press, 2007. 265—278