基于流的哈希函数比较分析研究

强士卿[1], 程 光[2]

(1. 东南大学 计算机科学与工程学院, 江苏 南京 210096; 2 东南大学 江苏省计算机网络重点实验室, 江苏 南京 210096)

[摘要] 为了缓解高速网络测量与硬件资源之间的矛盾, 需要对网络流进行抽样处理, 基于哈希的流抽样技术的广泛应用, 流哈希函数性能是整个测量系统的核心. 高速网络测量中对于哈希函数的研究主要集中在报文哈希函数性能方面, 目前还未对现有的流哈希函数的性能进行分析比较研究. 从理论分析和实验验证的角度出发, 提出了几种流哈希函数的性能测度, 并使用 CERNET主干流量比较验证了一些通用的流哈希函数的均匀性、冲突率等性能测度, 为流哈希函数的选择与使用提供依据.

[关键词] 流哈希函数, 报文哈希函数, 均匀性, 冲突率

[中图分类号] TP 393 [文献标识码] A [文章编号] 1672-1292(2008)04-0025-04

Comparison and Analysis of Hash Algorithm Based on Flows

Qiang Shiqing[1], Cheng Guang[2]

(1. School of Computer Science and Engineering, Southeast University, Nanjing 210096, China;
2. Jiangsu Provincial Key Laboratory of Computer Network Technology, Southeast University, Nanjing 210096, China)

Abstract In order to relax the contradiction between the high-speed network measurement and the hardware resources it is necessary to make a sampling treatment with network flow. The emerging techniques based on hash-sampling have been widely used, and the performance of flow hash algorithm is one of the key measuring technologies. The study of hash in high-speed network measurement was mainly concentrated on the performance of the packet hash function, yet there haven't been the analysis and comparative study for the performance of the existing flow hash. From the theoretical analysis and experimental verification point, this paper puts forward several performance estimations of flow hash, and uses the CERNET backbone traffic to make comparative study in uniformity and low collision rate of general flow hash, and thus provides a foundation for selecting flow hash.

Key words flow hash, packet hash, uniformity, collision rate

网络流的测量和分析是许多网络应用的基础. 通过网络测量系统的报文可以聚类成流信息进一步分析应用, 比如检测 DDOS攻击. 发现 P2P流量等. 在高速网络中, 平均每 8 ns就会来一个报文, 这对路由器 CPU 资源要求很高. 网络流量的骤增会造成网络测量设备中路由器 CPU 资源枯竭. 一般采用报文抽样技术对原始报文进行过滤[1].

抽样采集的报文聚类成的流信息维护在内存的流表中, 但是高速网络线速的进一步提高, 即使进行报文抽样后, 维护所有的流需要大量的内存空间. 网络管理员主要关注的是占用网络带宽超过一定阀值的流即重尾流. 所以采用基于哈希函数的流抽样技术, 对网络中的流有选择地维护并在内存中进行测量. 流哈希函数是流抽样技术的核心.

流哈希函数主要应用在流记录的抽样、流存在性检查等. 文献[2]中对流哈希函数的计算速度进行了分析研究. Cheng[3]设计了一种新的流哈希函数, 并验证了其优越性.

本文的重点就是分析比较常用流哈希函数的性能, 并使用 CERNET 主干流量通过实验来检验流哈希函数的性能. 本文提出了一种新的测度活跃流估算法, 改善了现有的哈希函数性能测度不能明显区分流哈希函数的性能, 并通过实验验证了这种测度的可用性.

收稿日期: 2008-06-18

基金项目: 国家"973"计划(2003cb314804)、东南大学优秀青年教师项目和广东省计算机网络重点实验室开放研究(CCNL 200706)资助项目.

通讯联系人: 程 光, 博士, 副教授, 研究方向: 网络测量、网络行为学、网络安全等. E-mail: gcheng@njnet.edu.cn

1 哈希函数性能测度

1.1 均匀性

一般期望设计的哈希函数的哈希值均匀落入哈希空间, 避免发生聚焦. 将哈希空间 n 等分, 得到 p 个哈希值, 那么平均落入每个哈希子空间的哈希值是 $f_0 = p/n$, 落入第 i 个子空间的哈希值个数是 f_i. 式 (1) 反映了样本与理论分布的偏离, 统计量 χ^2 表示 f 到均匀分布的偏离度. 哈希函数均匀性可用卡方拟合优度检验来判断,

$$\chi^2 = \sum_{i=1}^{n} \frac{(f_i - f_0)^2}{f_0}. \tag{1}$$

1.2 冲突率

在高速网络环境下抽样采集 T 个报文进行处理, 将这些报文按四元组聚合成 N 个聚类点 (流) 作为哈希函数 f 的输入, 运算结果输出到 M 个槽中. 哈希的槽数和其输入值有直接关联, 对于千万量集的报文序列, 槽数大于百万量级才可能有良好的性能. 如果,

$flow\_1(ip\_src, ip\_dst, port\_src, port\_dst) \neq flow\_2(ip\_src, ip\_dst, port\_src, port\_dst)$ 且 $f(flow\_1) = f(flow\_2)$, 则定义为哈希函数冲突. 统计哈希函数输出到第 i 号槽中所有的流数目 N_i, 当 $N_i > 1$ 时此槽发生冲突, 可以得到此槽中发生哈希冲突的次数为: $\frac{N_i(N_i + 1)}{2}$. 综合考虑哈希空间的所有的槽, 使用下面的平均冲突次数来表示哈希函数 f 的冲突率, Collision越小冲突率越低, 哈希函数的性能也越好.

$$\text{Collision} = \frac{1}{\frac{N(N+1)}{2}} \sum_{i=1}^{M} \frac{N_i(N_i+1)}{2}, \tag{2}$$

其中, $N_i > 1$ 且 $\sum_{i=1}^{M} N_i = N$.

1.3 活跃流估算法

$$\hat{n} = m \ln\left(\frac{m}{n}\right), \tag{3}$$

其中, m 是哈希空间的总槽数; n 是哈希空间中被映射到聚类点的槽数.

此算法的前提是假设哈希函数尽可能随机, 哈希函数随机性越好, 估算出来的活跃流的数量就越接近真实值. 论文使用的哈希函数主要用在高速主干网络中, 报文的数量巨大而且报文特征字段的分布范围很广, 所以 m 和 n 很大. $\hat{n}$ 是活跃流数 n 的最大似然估计. 证明详见文献 [4].

1.4 计算速度

哈希函数对于每个流记录操作的时间是哈希函数性能的直接体现. 从理论上讲计算时间的直接体现就是基本操作步数越少、运算速度越快. 另外, 算法的程序实现也会影响其运算速度, 要尽量避免使用跳转语句、子函数调用等低效率语句. 在实际的使用中, 机器的硬件性能都影响着哈希函数的速度. 所以下面的几种哈希函数性能的比较要对每个代码块进行同级别的优化, 在资源公平的条件下比较几种哈希算法.

2 哈希函数实验比较分析

哈希函数的性能比较选取本文提出的几种性能指标, 分析数据来自 CERNET数据. CERNET数据是 2005年 11月 10日从 18: 48~ 19: 58, 在我国华东 (北)地区网络中心对 CERNET主干网络主干流量连续测量 10m in的数据. 流量 Trace采用 L ibpcap的数据格式, 每个报文记录默认保存 8字节的时戳和 64字节的报文首部信息, 总共测量到 24G的报文. 本文中使用的 CERNET数据全部来自这组数据源.

网络流的定义通常采用五元组 (源地址、宿地址、源端口、宿端口、协议类型). 由于网络中 90%以上的流为 TCP流, 因此不考虑将协议字段作为哈希算法的输入参数. 文中讨论的流哈希函数的输入参数是四元组的流记录, 经过哈希运算输出的是 16~ 20位的比特串. 流哈希函数和报文哈希函数之间区别是输入的比特串是否与报文的网络传输路径相关. 实验中比较的流哈希函数有 IPSX[5]、Bob[6]、CRC32[5]和 MD5等.

2.1 卡方检验

在卡方检验中, 假定样本的分布是均匀的, 设定可信度 $\partial = 0.99$, 样本数量为 512, 则 $\chi^2_{(0.99)}(511) =$

588.7. 我们以每 30 000条流记录作为输入, 进行 40次独立的实验. 每次实验统计量值大于 588.7, 则拒绝样本分布均匀的假设. 观察图 1, uniform曲线就是 $\chi^2_{(0.99)}(511)$, MD5和 IPSX函数的分布不均匀, Bob和 CRC32有很好的均匀性, Bobs稍好.

2.2 冲突率的比较

从实验数据中聚合并选取 75万条流记录, 以每 5万条流记录为间隔输入并测试哈希函数冲突率. 实验的结果如图 2所示, 横轴表示流记录数, 纵轴表示冲突率. 由图 2可以看出, 随着输入流记录数的增加, 哈希函数冲突率逐渐降低且趋于平稳.

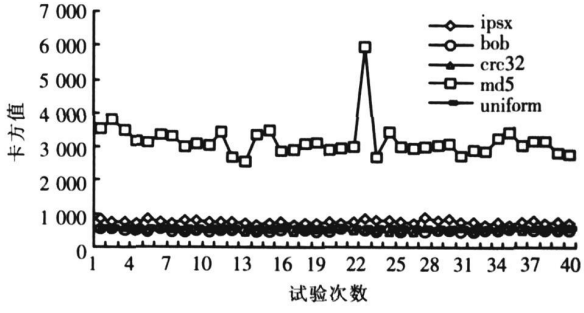

图 1 卡方检验(置信度∂=0.99, 样本量 512)

Fig.1 Chi-square test (confidence level ∂=0.99, samples 512)

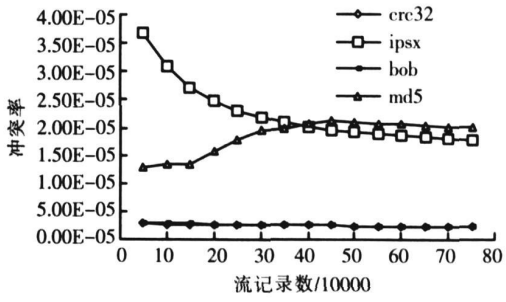

图 2 冲突率比较

Fig.2 Comparison in collision rate

2.3 活跃流估测度

选取 20组 5 s trace, 估算活跃流的数量并与真实值进行比较. 观察图 3, MD5估算值远差于其它哈希函数. IPSX估算值在真实值的上下波动. 而 Bob和 CRC32都接近真实值, 图 3中无法明显区分这两者性能. 从相对误差的角度来分析, 由图 4可以看出, CRC32估算误差曲线在 Bob的下面, 因而 CRC32的均匀性优于 Bob.

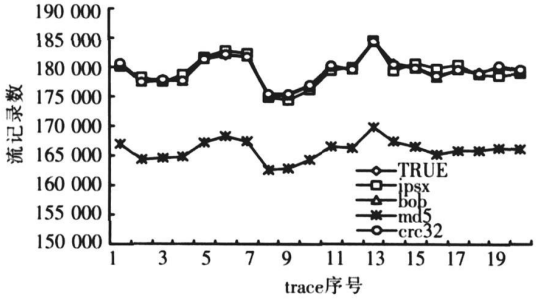

图 3 活跃流估算量比较

Fig.3 Comparison in flow estimation

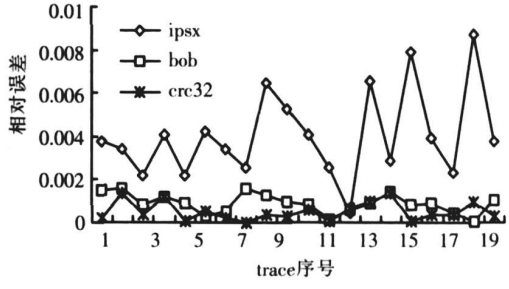

图 4 活跃流估算相对误差比较

Fig.4 Comparison in relative error of flow estimation

2.4 速度比较

表 1中对 3种流哈希哈数的操作数进行了比较, MD5的操作步数远大于这 3者的操作步数. 理论上, IPSX在效率方面具有很大的优势. 实验使用的服务器配置: Intel(R) Pentium(R) III CPU, 主频 1 266 MHz, 内存 1 032M; 缓存 900M; 操作系统 Linux(Fedora Core 5). 这里选取了 1 min的 trace作为输入, 分别对 759 041个流记录进行运算, 平均每个流记录进行哈希计算的时间如图 5所示.

表 1 操作数比较

Table 1 Comparison in operations

哈希函数	操作数			
	与	或	移位	异或
IPSX			77	256
CRC32	3 072			1 536
Bob		648	650	2 336

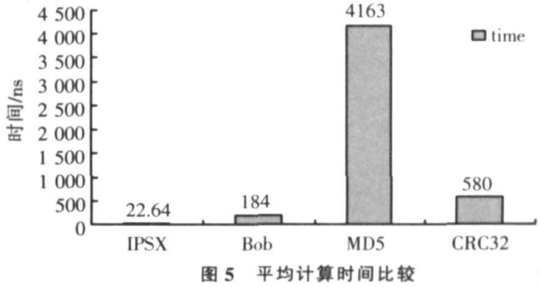

图 5 平均计算时间比较

Fig.5 Comparison in average operational time

3 结论

论文提出几个流哈希函数性能的测度,通过实验比较分析了一些常用的流哈希函数的性能.实验的结果可以作为选择哈希函数的基础.一般情况下建议选用 Bob.对于速度要求很高,选用 IPSX.流哈希函数的输入对输出产生很大的影响,论文下一步的工作就是对于不同的数据输入时流哈希函数性能所体现出来的变化进行分析比较.

[参考文献](References)

[1] Molina M, Niccolini S, Duffield N G. A comparative experimental study of hash functions applied to packet sampling[C]// International Teletraffic Congress(ITC-19). Beijing 2005.

[2] Molina M, Tartarelli S, Raspall n F, et al. Implementation of an IPFIX compliant flow trafficmeter challenges and performance assessment[C]// IEEE Workshop on IP Operations and Management, 2003: 61-67.

[3] 程光,龚俭,丁伟, 等. 面向 IP流测量的哈希算法研究[J]. 软件学报, 2005, 16(5): 652-658.
Cheng Guang, Gong Jian, Ding Wei, et al. A hash algorithm for IP flow measurement[J]. Journal of Software, 2005, 16(5): 652-658. (in Chinese)

[4] Whang K, Vander Zanden B, Taylor H. A Linear-time probabilistic counting algorithm for database applications[J]. ACM Trans on Database Systems, 1990, 15(2): 208-229.

[5] Zseby T, Molina M, Raspall F, et al. Sampling and filtering techniques for IP packet selection[EB/OL]. IETF 2004: 32-33. http://tools.ietf.org/html/draft-ietf-psamp-sample-tech-05#section-17.

[6] Jenkins B. Algorithm Alley, Dr Dobb's Journal, September 1997[J/OL]. http://burtleburtle.net/bob/hash/doobs.html

[责任编辑: 刘 健]