一种基于数据相关性特征的网络数据流 HASH 函数

夏青[1, 2]， 丁伟[1, 2]

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

摘要：网络流是众多网络分析与网络安全等应用的数据来源，而针对流的 hash 结构是这些应用普遍使用的数据结构， hash 算法作为 hash 结构的核心直接关系着这些应用的效率。本文针对网络结点所采集的数据进行分析，发现结点数据具有相关性，并定义了比特串相关性测度。传统的流 hash 函数设计方法没有考虑数据可能存在相关性的特点。本文提出的 CRLT hash，基于对数据相关性特征的统计信息进行设计，经实验证明比现有的流 hash 函数更适合处理网络结点的流数据。

关键字：流 hash 函数；报文 hash 函数；数据相关性

1 引言

hash 函数是 IP 报文聚类的基本手段，所谓报文聚类，就是按照一定的标准或者规范，将海量的报文分为若干个集合，通过研究报文集合的特点，从更高的层面反映网络的运行特征。所谓网络数据流就是符合特定的流规范和超时约束的一系列数据包的集合，所以从 IP TRACE 流生成网络数据流的过程也是一种 IP 分类的过程。目前广泛使用的流规范是 IP 流的五元组流规范和思科定义的 netflow 七元组的流规范。所谓五元组就是把源地址、宿地址、源端口、宿端口、协议类型这五个元组作为定义数据流的依据，而流记录中除了这五元组通常还包括流开始和结束时间、流内报文数、流内总字节数等统计信息。七元组就是在这五元组的基础上增加了 TOS 和路由器信息。但是路由器的信息在报文中无法得到，90%以上的 TOS 值都为 0，因此对于 netflow 规范分类也只能使用 5 元组作为输入。

目前组流使用的 hash 函数有思科路由器的硬件 hash，IPSX[8]，Cheng hash[10]和文献中使用的流 hash 函数，其中思科路由器的硬件 hash 没有对外公布。文献[10]中证明 Cheng hash 函数是目前性能最好的组流函数，该算法主要通过对数据源的随机性特征的分析来安排位移和异或进而达到提高均匀性的目的。该算法在输入数据相关性较小的情况下性能较好。而在接入网节点采集的数据中，由于地址部分字节之间相关性较大，所以算法本身还有改进的余地。据此，本文提出的 CRLT 方法改进了 Cheng 的 hash 函数，依据对数据的随机性特征以及相关性特征进行了综合的统计及分析，设计了一个能够对数据特征进行自适应的 hash 函数构造方法，更适用于更靠近接入网节点采集数据的组流。

对结果的分析和验证采用了三个实测的 IP Trace，它们均采自 CERNET 某省网边界，采集时间是，2008 年 12 月 14 日 13:55 分至 15:04 分，具体描述如下：

数据 1 是全部数据，总量为：616809400 个报文。数据 2 是数据 1 中流量最大的七个单位的数据之和，数据总量为 185042820。数据 3 是数据 2 中流量最大的一个单位的数据，总量为 102876743 个报文。

2 流 hash 函数运算符

流 hash 函数属于位操作 hash，一般通过异或、位移、加减三种操作来增强输入值的随机性。本文用到前面的 2 种，简单介绍如下。

2.1 异或

比特 A 和比特 B 异或后结果如图 1：

基金项目：本文获国家科技支撑项目（2008BAH37B04）资助。

作者简介：夏青（1985～），硕士研究生，主要研究领域网络行为学。丁伟，教授，博士研究生导师，主要研究领域计算机集成制造、通用搜索引擎、PKI 证书体系、网络环境下的远程教育和网络行为学等。

比特A	0 0 1 1
比特B	0 1 0 1
⊕	0 1 1 0

图 1 异或运算表

2.2 位移

本文定义的位移运算涉及 2 个长度分别为 L1 和 L2 的比特串 B1 和 B2，且由循环右移和异或两个动作组成，具体描述如下：

Shift(shift, B1,B2,L1,L2){// shift 为右移位数//

将 B1、B2 两个比特串左端对齐，如果 L1<>L2，将短的一个右端用 0 补齐；

将字符串 B2 向右循环移动 shift 位；

将 B1 和 B2 按位做异或运算并将结果放在 B2 中

}

如果 B1=1001010(L1=7)，B2=01011(L2=5)，则 Shift(4,B1,B2,7,5)的结果是 B1=1001010（保持不变），B2=0101000。

3 IP TRACE 比特随机性和相关性

3.1 随机性

定义 1 IP Trace 特定比特为 1 的概率 p：设该 TRACE 所包含的报文数量为 n，统计这 n 个报文中某特定比特为 1 的报文数量 m，则 p=m/n。

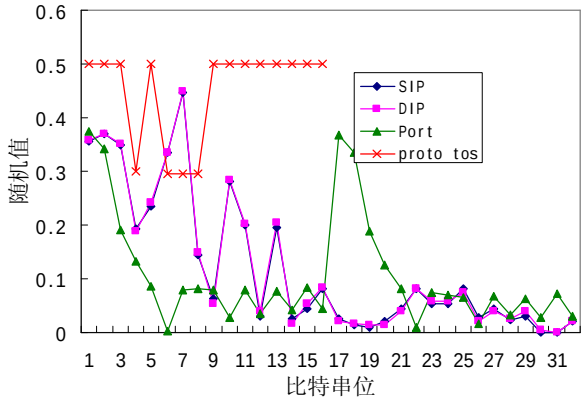

图 2 网络元组比特随机性分析

定义 2 IP Trace 特定比特随机性 ram: ram=|0.5-p|。p 为定义 1 中所描述的该比特为 1 的概率。ram∈(0,0.5)，越靠近 0 表明该比特的随机性越好，反之亦然。

图 2 为数据 1 中源宿 IP、源宿端口，协议位和 TOS 位各比特的随机性值。

3.2 相关性

3.2.1 比特相关性

定义 3 IP Trace 特定比特组合随机性 p_{ij}：设该 TRACE 所包含报文数量为 n，统计这 n 个报文中某两个特定比特 i、j 同时为 1 的数据个数，计为 m。则 m/n 为比特组合为 1 的概率值。

文献[9]中证明两个比特位做异或时，如果比特不相关，即满足下式：

$$|r| \le \min\left(\left|\frac{2p+q-1-2pq}{2\sqrt{pq(1-p)(1-q)}}\right|, \left|\frac{q-2pq}{2\sqrt{pq(1-p)(1-q)}}\right|\right) \quad (1)$$

的情况下，异或后随机性会增加。其中 ρ 为相关系数。 $r = \mathrm{cov}(e,h)/\sqrt{\mathrm{cov}(e,e)\,\mathrm{cov}(h,h)} = (E(e,h) - pq)/$

$\sqrt{pq(1-p)(1-q)}$。p 为第一个比特为 1 的概率，q 为第二个比特为 1 的概率，$E(e,h)$ 为两比特同时为 1 的概率值。

本文可以证明如果两比特满足下式，异或后随机性会增大：

$$|p_1 + p_2 - 2p_{12} - 0.5| < \min(|0.5 - p_1|, |0.5 - p_2|) \quad (2)$$

证明：b_1、b_2 为参加异或的两个比特，异或后生成比特 b_3，p_1、p_2、p_3 代表分别表示 b_1、b_2、b_3 为 1 的概率，p_{12} 表示 b_1、b_2 同时为 1 的概率（定义 3），假设比特 b_1 为 1 是事件 A，比特 b_2 为 1 是事件 B，则 $p_3 = P(A\overline{B}) + P(\overline{A}B) = (P(A) - P(AB)) + (P(B) - P(AB)) = P(A) + P(B) - 2P(AB) = |p_1 + p_2 - 2p_{12}|$。$b_3$ 的比特随机性距离为 $|0.5 - p_3| = |p_1 + p_2 - 2p_{12} - 0.5|$。因此异或后比特随机性增大需要满足的条件为 $|0.5 - p_3| < \min(|0.5 - p_1|, |0.5 - p_2|)$ 证毕。

相对于（1）式，（2）式使用相同的参数，但计算要更加简单直观。

3.2.2 比特串的相关性：

本节在比特相关性的基础上讨论比特串的相关性。

定义 4 比特串相关矩阵：设比特串 B1、B2 的长度分别为 m 和 n。从两串中各取一个比特构成比特对，共有 m*n 对，把每对比特分别代入不等式（2），满足不等式的，在矩阵中的对应值为 1，否则为 0。由此形成的矩阵称为 B1、B2 的比特串相关矩阵。

根据定义 1 和定义 3，（2）式中 p_1、p_2、p_{12} 均可由统计获得。例：统计数据 1 的源 IP 的第一、二字节，在本例中构成 8*8 的矩阵。如图 3：

源IP第一字节

源IP第二字节	0	1	2	3	4	5	6	7
0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	1	1	0
2	0	0	0	1	0	0	0	0
3	0	1	0	1	0	0	0	0
4	0	0	0	1	0	0	1	0
5	1	1	1	0	1	1	1	0
6	1	1	0	1	0	1	0	0
7	0	1	0	0	0	0	1	0

图 3 源 IP 第一、二字节相关性矩阵

定义 5 比特串相关性值 r：设相关性矩阵中 1 的个数 i，矩阵中共有 k 个值，则 r =i/k。

该值越大说明相关性越小，否则相关性越大。

本文面向的数据是在边网节点上采集出来的，其入流量中的宿地址相对集中，而出流量中的源地址相对集中，几个主要的 IP 前缀加起来占据了总量的 73%，因此至少 IP 地址前两个字节具有较强的相关性。

根据数据 1 统计结果发现，源 IP 之间，宿 IP 之间字节相关性较大。将源 IP 的 4 个字节各自作为 1 个比特串，两两作相关性分析，结果如表 1，其相关性值都在 0.8 以下，说明相关性较大。表 2 中将源 IP 的四个字节分别与宿 IP 的四个字节两两作分析，值基本都在 0.8 以上，说明源、宿 IP 的相关性较小。

	字节 1	字节 2	字节 3	字节 4
字节 1	0.56	0.47	0.67	0.77
字节 2	0.47	0.34	0.63	0.61
字节 3	0.67	0.63	0.69	0.78
字节 4	0.77	0.61	0.78	0.38

表 2 源地址与宿地址 4 字节相关性值

	字节 1	字节 2	字节 3	字节 4
字节 1	0.98	0.84	0.97	0.91
字节 2	0.89	0.82	0.92	0.83
字节 3	0.97	0.92	0.97	0.87
字节 4	0.91	0.83	0.87	0.70

表 1 源地址 4 字节的相关性值

经过相同的分析方法发现源、宿端口之间，端口与 IP 地址之间的相关性值大于 0.8，可以认为是不相关的。但源、宿端口各自的第一第二字节的相关性较大，值在 0.5 左右。

4 基于相关性特征的 CRLT 流 hash 函数设计

本论文提出的算法是对 Cheng Hash[10]的改进，因此先简单介绍一下该算法

4.1Cheng hash

Cheng hash 步骤如下：使用源宿地址、源宿端口共 96 位，每 8 位为一个比特串单元，共 12 个比特串。用 Bi，i=1,2,...12 表示这 12 个串。

步骤 1（统计）：取一段 IP TRACE 统计这 96 个比特的随机性。

步骤 2（求初始串）：假设 hash 函数的输出比特串长 L，在源地址中选取 L 个比特作为初始化值 key。

步骤 3（求各位移值）：根据统计值可得到 key 和 B1 的各位随机性。选择 key 中随机性最差的一位 b1，和 B1 中随机性最好的一位 b2。因为 key 和 B1 要作异或，所以通过对 B1 位移 shift1 位，使得 b1、b2 两个比特对齐。然后将 B1、B2、shift1 代入可以预测异或后结果随机性的 XOR 函数，得到结果串的随机性预测数组 hash1。从 hash1 中选取随机性最差的一位，同时在 B2 中选择随机性最好的一位，按同样的方法位移 shift2 位后，再异或得到比特串 hash2，然后参与到下一轮异或当中。按照此方法，直到将 B12 的异或位移步骤完成。

步骤 4（构成 hash 函数）：在以上过程中计算出的 shifti,i=1,2,…12 这 12 个位移，逐个代入 2.2 节的 Shift 函数，即为最终所使用的 hash 函数。

步骤 1-3 是生成 hash 函数的步骤，步骤 4 的输出结果作为程序使用的 hash 函数。经实验统计发现数据特征具有一定的稳定性。因此，前 3 个步骤只需要隔一段时间运行一次。

此方法假设异或字节都为不相关字节，因此不适用于数据有相关性的情况。即使对于不相关性字节的异或操作，该方法还是可以作进一步优化。

4.2 优化方法

4.2.1 提高不相关比特串随机性方法

Cheng 方法的核心思想是使两个比特串异或结果的随机性更平均。即通过异或一个比特串中随机性最好的一位与前串中随机性最差的一位异或，来增加该位的随机性。由于两个相关性较强的比特串异或时，可以用位移的方法使更少的相关性比特在一起作异或，所以下述方法产生的比特串随机性更好。

算法涉及两比特串 B1，B2 分别对应的随机性数组 p1[]、p2[]，以及各自的长度 L1 和 L2。

```
URshift(p1[],p2[],L1, L2){
  While（尝试两比特串异或时所有可能的位移值）{
      计算异或后比特串各位随机性
      查找结果串各比特位中最大的随机性值，记录到集合 S 中，同时记下当前位移
  }
  选取 S 中最小值所对应的位移作为最佳位移
}
```

4.2.2 减弱比特串相关性影响的方法

为了完成上述算法，首先定义两个长度分别为 m 和 n 的比特串，串间位移值为两个串第一个比特间的距离。这个值的取值范围为：0~max(m,n)。

图 4 显示了两个 8 位比特串，位移为 7 时的情况。

假设两比特串分别为源地址第一、二字节。根据数据 1 计算出的它们的相关性矩阵如图 5。不同斜线上 1 的数量对应于相应位移后随机性的大小，1 的数量越多表明随机性越强，用该方法对应于上述算法中的最大随机性。具体描述如下：

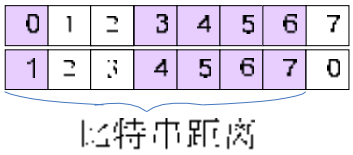

图 4 相关性比特串异或

算法输入的参数依次为比特串 B1、B2 的长度 L1、L2，及比特串的相关性矩阵 BR[][ ]，算法如下：

```
BRshift(L1, L2, BR[][ ]){
   While(遍历所有可能的位移值){
       求出每个位移下，矩阵 BR[ ][ ]中斜线上 1 的个数记录到集合 S 中，并记下对应位移值
     }
     找到集合 S 中最大值所对应的 1 个或多个位移
  返回位移集合
}
```

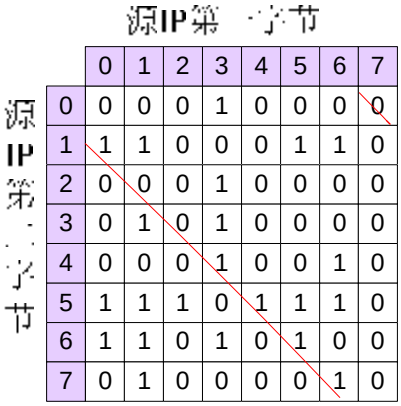

图 5 第一二字节相关性矩阵

根据文献[10]，可以预测比特串异或后结果的随机性值。设两比特随机性值分别为 ram1，ram2，异或后其结果的随机性值为 ram3。当两个比特不相关时 ram3=2*ram1*ram2，相关时 ram3∈（2*ram1*ram2，min(ram1,ram2)）。可以通过该式，估算异或后比特串随机性值范围。然而比特串异或 2 次以上后将无法判断串内比特与其它串中比特的相关性，所以只能使用一种近似预测值公式，本算法和 cheng hash 一样，使用 XOR 函数，即 ram3=2*ram1*ram2 来预测异或后结果随机性。

4.3 CRLT hash 函数构造算法:

本小节完整地给出改进后的 CRLT hash 算法。

步骤 1：统计各参数

hash 函数的输入为源宿地址、源宿端口共 96 位。统计 IP TRACE 为每个字段建立其相关性矩阵。其中源宿 IP 分别对应矩阵 RSIP、RDIP，源宿端口对应矩阵 RSPORT、RDPORT。并统计出每个字节的比特随机性值。

步骤 2： hash 函数初始串

设 hash 值长 L，对源宿 IP 分别取其前 L 个比特，构成源 IP 初始串 initSIP 和宿 IP 初始串 initDIP。

步骤 3：求各异或字节位移值

源宿地址，每 8 位为一个比特串单元，分别用 SIPi、DIPi，i=12,3,4,表示各字节随机性值。SPORT、DPORT 为源宿端口随机性值，SPORT2、DPORT2 分别表示源宿端口的第 2 字节随机性值。

将函数 BRshift、URshift、XOR 结合使用，可以求出多个比特串相异或时，使最终结果随机性最好，并且相关性字节结合最少的各比特串之间位移。

算法如下：

1. 用 BRshift 根据 RSIP 分别求出 SIP3、SIP4 与 initSIP 之间的相对位移值集合 set1、set2。因为 3、4 字节之间的相关性不大，所以不考虑他们之间的位移。同理对 DIP3、DIP4 与初始串 initDIP 求出集合 set3,set4。
2. 用 BRshift 根据端口相关性矩阵，分别求出 SPORT2 与 SPORT、DPORT2 与 DPORT 的相对位移值集合 set5,set6。
3. While(尝试 set1~set6 中所有的位移值组合)

{//shifti∈seti。hashi 表示异或后结果随机性预测值

1） 求 XOR(initSIP, SIP3,shift1)的结果 hash1，和 XOR(initSIP, SIP4,shift2)的结果 hash2。同理，求出宿地址对应的结果 hash3、 hash4。

2） 用 XOR(SPORT, SPORT2,shift5)、XOR(DPORT,PDPORT2,shift6)分别更新 SPORT 和 DPORT。

3） 用 URshift(hash2,hash4)求出位移 shift7，并带入 XOR(hash2,hash4,shift7)求出 hash5。

4） 用 URshift(hash5, SPORT)求出 shift8。再代入 XOR（hash5,SPORT, shift8）求出 hash6。

5） 对 hash6 和 DPORT 重复 4）中的步骤分别求出 shift9 和 hash7。

6） If(hash7 的随机性是目前最好的)

记录下位移值 shift1~shift9 至集合 S_Shift。

}

4. S_Shift 中的各位移值作为最佳位移。

步骤 4：构成 hash 函数

本过程与 Cheng hash 相同。将位移值代入 Shift 函数求得最终 hash 函数。

5 hash 函数性能评估

5.1 hash 函数散布均匀性测试

评测hash函数的散布均匀特性．有许多可选的方法[8][9][10]，本文选择R．Jain etal[11]提出的测度方法并结合我们的工作背景作了适应性修改，表述如下：

假定xi是hash函数h分配给第i(1≤i≤M)个槽的键值数，记为随机变量x，$\sum_{i=1}^{M} xi = N$，则h的均匀性定义为：

$$fairness\ (h) = \frac{(E(x))^2}{E(x^2)} = \frac{(\sum_{i=1}^{M} x_i)^2}{M(\sum_{i=1}^{M} x_i^2)} \quad (3)$$

易证0<fairness(h)<1。fairness(h)越接近1，表明h越公平(或越具有均匀散布特性)。若fairness(h)=0.30，意味着对30％的槽来说是公平的，对70%的槽是不公平的，它能反映与理想分布情况(fairness(h)=1)的整体偏离程度，所以我们选择它作为评价公平性（均匀性)的指标。

对本文所使用的数据集1，2，3，分别对其使用CRLT hash，并与Cheng hash， Gao hash，以及BOB hash作比较，结果见表3。

数据	函数名	Fairness（%）	平均槽占用率（%）	最大链长	hash长度	时间（s）
1	CRLT	69.8	90.5	35	16	1615
1	Cheng	68.7	90.5	27	16	1616
1	Gao	60.8	83.8	26	16	1620

1	BOB	10.1	53.7	274	16	1627
2	CRLT	58.7	75.8	13	16	1571
2	Cheng	57.3	75.4	13	16	1572
2	Gao	56.8	75.7	14	16	1577
2	BOB	10.6	48	163	16	1578
3	CRLT	34.9	40.1	8	15	109
3	Cheng	33.3	40.9	8	15	124
3	Gao	33.2	42.6	8	15	148
3	BOB	18.8	35.6	74	15	184

表 3 hash 函数随机测试结果

从表3中数据可以看出CRLT hash比Cheng hash性能略好，在数据量大时Gao hash的表现和前两个差距更大，当数据量小时Gao hash和前两者的性能更接近。而BOB hash针对本数据集则表现较差。

5.2 代价分析

5.2.1 hash 函数的时间复杂度分析：

CRLT算法共产生了2个初始值和9个位移值，共11个操作。Cheng hash则产生了一个初始值和12个位移值，需要13次操作。Gao算法在本例中有18次操作，而BOB算法需要37次。

5.2.2 函数生成所用的时间代价分析：

Gao和BOB算法代码固定，所以不需要生成时间。而Cheng hash和CRLT hash需要统计数据特征来更新hash函数。为了减少统计量，采用1：64抽样统计数据特征。每读到一个抽样数据CRLT需统计640次比特位值。Cheng需要统计96次。本文使用的组流程序采多线程，且读、写线程分离。其中统计位于速度较快的读线程当中，因此采用1/64抽样统计后对运行时间影响不明显。CRLT hash函数的构造算法比较复杂但运行时间只需3毫秒。每隔一小时更新一次其时间花费可以忽略。在相同的实验环境下对4种流hash函数分别使用数据1，2，3组流，运行时间相差并不大，具体见表3。

6 结论

hash 函数作为组流操作的核心技术，其性能性的好坏直接决定了整个系统的效率。本文在对该相关领域进行了全面分析的基础上，根据接入网边界数据相关性高的特点，提出了一个具有针对性的 hash 函数算法 CRLT hash。经实测数据检验并与其它 hash 算法比对，该函数能达到预期的设计目标，越靠近末端接入网的边界，其性能越好。

参考文献：

[1]IP Flow information export (ipfix).2004.http://www.ietf.org/html.charters/ipfix-charter.html

[2] Cheng Guang, Gong Jian, Ding Wei, A Traffic Sampling Model for Measurement Using Packet Identification,[J] ICON 2002.409-413.

[3]CiscoNetflow.2004.http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml

[4] Haoyu Song，Fast Hash Table Lookup Using Extended Bloom Filter[J] SIGCOMM’05, August 22－26, 2005

[5] Woo-jin Yang, Tae-il Kim, Hae-won Jung，Optimizing Hash Table Structure of Flow Exporting Software[J] IEEE Feb20-22, 2006 ICA0T2006

[6]http://tools.ietf.org/html/draft-ietf-psamp-sample-tech-05#section-17

[7]http://tools.ietf.org/html/draft-ietf-psamp-sample-tech-05#ref-Jenk97

[8] Cheng Guang, Zhao Wei, Gong Jian, XOR Hashing Algorithms to Measured Flows at the High-Speed Link,[C] Second International Conference on Future Generation Communication and Networking 2008

[9] 程光 ，龚俭 ，丁伟，等 面向 IP 流测量的 hash 算法研究[J]软件学报 2005，65：62

[10] Cheng Guang，Gong Jian，A Method to Device Flow Hash Algorithm Based Traffic Character[C] IEEE International Conference on Communication Technology Proceedings 2008 11th
[11] Bhargava R，Goel A，meyerson A. Using approximate majorization to characterize protocol fairness[C]. In：Proc. SIGMETRICS／Performance 01.2001

A Flow Hash Algorithm Based Data Relativity

Xia Qing1,2, Ding Wei1,2
(1.School of Computer Science and Engineering,Southeast University, Nanjing 210096;2. Jiangsu Provincial Key Laboratory of Computer Network Technology,Nanjing 210096)

Abstract Flow can be analyzed for several applications, such as network security and traffic analysis. Hash table designed for flow is used commonly in these applications, while hash algorithm is one of the key measuring technologies. We analyze the data get from a network node and find out that the data is related, but no algorithm is designed with this character. So we proposed a new hash algorithm base on this kind of analysis, and is proved that when it used in our network node it is better than other algorithms.

Keywords flow hash; packet hash; data relativity