doi:10.11887/j.cn.201806011 http://journal.nudt.edu.cn

流的大小与传输速率相结合的双门限检测算法*

李春强[1],董永强[1,2],吴国新[1,2]
(1. 东南大学 计算机科学与工程学院,江苏 南京 211189;
2. 东南大学 计算机网络和信息集成教育部重点实验室,江苏 南京 211189)

摘 要:提出基于流传输速率与数据量的双门限检测算法。为满足高速网络传输的性能要求,使用Hash表存储流检测的数据结构,将Hash表的冲突处理与基于流速率的缓存替换相结合以实现高效的大流检测,通过限制Hash桶的容量,确保报文的处理性能。真实网络数据的仿真测试结果表明:所提算法在相近的存储开销下,保持了较高的处理性能,准确性优于基于最近最少使用算法的大流检测及其派生算法以及基于统计计数的紧凑型空间节省算法。

关键词:流检测;Hash表;传输速率;Hash冲突;缓存替换
中图分类号:TP393 **文献标志码**:A **文章编号**:1001-2486(2018)06-075-07

Dual threshold elephant flow detection algorithm combined flow size with transmission rate

LI Chunqiang[1], *DONG Yongqiang*[1,2], *WU Guoxin*[1,2]
(1. School of Computer Science and Engineering, Southeast University, Nanjing 211189, China;
2. Ministry of Education Key Laboratory of Computer Network and Information Integration, Southeast University, Nanjing 211189, China)

Abstract: A novel algorithm which is based on both the flows' size and transmission rate was proposed for elephant flow detection. In order to meet the performance requirement of high speed networks, the data structure of elephant flow detection was indexed by Hash table, which combines the Hash conflict resolution with the eviction of cached flow entry to identify the elephant flow efficiently. A theoretical analysis was conducted to demonstrate the accuracy, performance and memory overhead of the detection algorithm. Experimental results on real data sets show that the proposed algorithm outperforms least recently used detection algorithm, its derivations and compact space saving detection algorithm in terms of accuracy and performance with similar memory overhead.

Key words: flow detection; Hash table; transmission rate; Hash collision; cache eviction

大量研究[1-5]表明:网络中少数具有较大数据量的数据流生成了网络的大部分流量,而其他大量的数据流的数据量总和仅占小部分的网络流量;数据流在带宽占用上也表现出不均衡性,其中少量具有较高速率的数据流消耗了大量的网络带宽[3]。网络中数据流的这种分布特征严重影响了网络传输的有效性[3-4],导致了数据流传输带宽占用上的不公平性,对报文的传输时延产生了较大影响[6],严重时甚至发生拥塞导致报文丢失。在网络设备上,有效检测出这些数据流,并进行适当的管理与控制[7-11],可以改善网络传输的有效性,比如缓解网络拥塞[12]、增加网络的有效吞吐率、降低报文传输时延及报文丢失率、改善网络传输的公平性[13]等。

目前的研究根据数据流的统计特征定义了大流与小流。文献[6-8,12]将数据量超过一定门限(如64 KB、1 MB、100 MB等)的流定义为大流,这一定义主要关注的是数据流的数据量;文献[14]将数据传输速率达到链路容量的一定比例(如1%、0.1%)的流定义为大流,这一定义重点关注的是数据流的传输速率。链路的带宽容量是以数据传输速率衡量的,因此准确及时地检测出具有高传输速率的数据流是实施流量的管理与控制、改善网络传输有效性的关键。然而对具有较高的传输速率但仅有少量数据的数据流而言,其生存期短,对网络传输的有效性不会产生持续性影响,因此在进行传输控制时可以被忽略。对于具有较高的传输速率和较大数据量的数据流,

* **收稿日期**:2017-09-28
基金项目:国家863计划资助项目(2013AA013503);国家自然科学基金资助项目(61272532);赛尔网络下一代互联网技术创新资助项目(NGII20160407)
作者简介:李春强(1975—),男,山东沾化人,博士研究生,E-mail:lcq9432@163.com;
吴国新(通信作者),男,教授,硕士,博士生导师,E-mail:gwu@seu.edu.cn

不断竞争网络的可用带宽对网络传输的有效性产生持续影响,成为网络传输控制的主要对象。综上所述,本文将同时具有较高传输速率和较大数据量的数据流定义为大流。由于网络中传输的数据报文大小并不统一,大报文的数据量甚至相当于小报文的数据量的几十倍,数据流的报文发送频率与数据流的传输速率并不是完全等价的,为了确保检测的准确性,将数据流的传输速率而不是报文的发送频率作为大流检测的重要依据。

随着网络技术的发展,传输链路的带宽容量和数据流的传输速率越来越大。高速网络的转发能力对检测算法的处理提出了很高的性能要求。本文针对高速网络中以流量管理与控制为主要目标的大流检测,提出了基于流的大小与传输速率的双门限检测(Dual threshold flow detection based-on flow Size and transmission Rate, DSR)算法。

1 相关工作

大流检测在网络管理等领域的应用,引起了研究人员的关注。根据存储结构中是否维护数据流的标识符信息可分为无状态流统计检测方案与部分流统计检测方案。尤其是随着应用场景的拓展,网络链路带宽的增加,不断涌现新的研究成果;这些成果从准确性、性能、存储开销等方面对已有的研究进行了改进,涌现许多组合型方案。

1.1 无状态流统计检测方案

在该类方案中不必存储数据流标识符,只需通过 Hash 函数将数据流的统计信息与一组计数器关联,当与之关联的所有计数器都超过所设定的门限后将其标识为大流。由于存在 Hash 冲突,因此一个计数器会与多个数据流相关。这类方案的代表有多阶段过滤器(Multi-Stage Filter, MSF)[14],基于计数布鲁姆过滤器(Count Bloom Filter, CBF)[15]的检测算法等。这类方案具有较小的存储开销;但可能将速率和数据量都非常小的流误识别为大流,误识别率与总的流数目及存储开销密切相关。尤其是对网络数据流而言,随着检测时间的推移,进入系统中的数据流数目越来越多,Hash 冲突率随之增加,从而导致检测的误报率增加。为了降低这种错误率则需要对部分低于一定门限的计数进行清理,清理时需要扫描整个存储结构,由于报文是连续到达的,清理操作可能会导致部分到达的报文来不及被处理。λ-HCount[16]、节省空间的前向衰减计数最小(Forward Decay Count Min Space Saving, FDCMSS)算法[17]给出了基于时间衰减模型的流检测方案,这类方案可以实现常数级的清理操作,但其检测的准确性跟衰减参数的设置密切相关。

1.2 部分流统计检测方案

这类方案需要存储流的标识符及相应的统计信息,由于存储空间是有限的,只能维护一部分流的信息,需要根据一定的策略对检测缓存中的条目进行淘汰。根据检测缓存条目的淘汰方式可分为:集中式淘汰与分散式淘汰。

在集中式淘汰的部分流统计方案中,每次进行缓存条目淘汰时,需要扫描整个缓存空间,按照预先设定的规则淘汰非大流缓存条目。这类方案的代表有基于采样保持(Sample and Hold, S&H)的检测算法[14],有损计数(Lossy Counting, LC)检测算法[18],概率型有损计数(Possibility Lossy Counting, PLC)检测算法[19]等。

分散式淘汰每次只要淘汰一个缓存条目。基于最近最少使用(Least Recently Used, LRU)[5]的检测算法采用了分散式淘汰,该算法使用队列维护流条目被淘汰的顺序,根据数据流报文的到达情况修改流在队列中的位置;然而大量突发到达的小流会严重影响算法的准确性。为了缓解突发到达的小流对准确性的影响,文献[20-21]提出了基于界标的 LRU(Landmark LRU, LLRU)算法,与 LRU 算法相比,LLRU 算法进一步提高了大流检测的准确性。LRU 类检测算法的处理复杂度是常数级的,能够满足高速网络的性能需求,但其准确性易受存储开销及数据流到达分布特征等因素的影响。

空间节省(Space Saving, SS)算法[22]也是基于分散式淘汰的部分流统计检测方案。该算法为了实现常数级的淘汰操作,需要动态分配存储空间并维护大量的指针,不利于硬件实现。紧凑型空间节省(Compact Space Saving, CSS)算法[23]对 SS 算法进行了改进,虽然实现了静态存储空间分配,但其处理操作仍然比较复杂。

1.3 组合型检测方案

通常,为了降低系统的资源开销,减少单位时间需要处理的报文数以及流数目,将采样技术[24]作为辅助措施应用于高速网络的大流检测中。文献[25]提出了通过周期性报文采样实现大流检测的方案,该方案存在较大的检测误差且无法避免误报率。采样还可以用于减少突发小流的到达对大流检测的干扰,比如采样型最近最少使用(Sample LRU, SLRU)算法[5]。对基于采样的方案而言,采样率的设置以及数据流的分布特征对

大流检测的准确性有着非常大的影响。

为了改善大流检测算法的准确性、性能以及存储开销,还出现了其他组合型大流检测方案,比如定时 CBF-LRU(Time CBF-LRU, TCBF-LRU)[26]、LRU-BF[27-28]等。TCBF-LRU 与 LRU-BF 算法的主要问题是,随着流数目的增加会导致检测的准确性逐步下降。

2　双门限大流检测算法

定义 1　网络中传输速率超过一定门限的数据流定义为高速流;所有高速流构成了高速流集合。

定义 2　网络中数据量超过一定门限的数据流定义为长流;所有长流构成了长流集合。

定义 3　网络中同时满足高速流和长流特征的数据流定义为大流;所有大流构成了大流集合。

为了有效检测出网络中的大流,避免将小流误识别为大流,采用部分流统计检测方案。下面从方案的基本思路、详细设计及分析逐步展开。

2.1　算法的基本思路

当到达的报文属于新的数据流,但无可用的存储空间时,需要依据流的传输速率淘汰一个流条目。流的传输速率等于流传输的数据量除以流的生存时间,流的生存时间为流的创建时间与当前时间之差,因此系统中所有流的传输速率随着时间的推移而不断变化,需要动态地计算。这一方法的优势在于:对于存储结构中已过期的流,由于不再有报文到达,其生存时间越长,传输速率越小,从而被淘汰;尤其对于仅包含单个报文的流,可以较快地将其从存储结构中淘汰,有效缓解了大量单报文流的突发到达对检测效果的干扰。

当报文到达时,需要在存储器中查找当前报文所属的数据流;如果未发现所属的数据流,且无可用存储空间,则还需要进一步扫描存储器,根据数据流的传输速率淘汰一条数据流。内容可寻址存储器(Content Addressable Memory, CAM)具有优异的查找性能,可以满足高速网络中报文处理的查找需求。然而若选择淘汰一条流,则需要扫描已存储的多个流条目才能确定出要淘汰的流,这将导致检测算法处理性能急剧下降,难以满足高速网络的处理要求。Hash 表具有良好的平均查找性能且易于实现,将 Hash 表应用到网络报文处理上得到了广泛的研究。然而对于普通的 Hash 表,当待查找的条目存在 Hash 冲突时,需要多次存储器访问才能获得查找结果,无法确保最差情形下的查找性能。由于 Hash 冲突的存在,采用 Hash 表组织流检测的存储结构,对于流条目的查找与淘汰操作,也需要扫描位于同一 Hash 桶中的多个流条目才能选择出应该淘汰的条目。

2.2　双门限检测算法的详细设计

使用 Hash 表作为大流检测的存储结构的主要问题是,Hash 冲突的存在导致流条目的查找与淘汰的处理难以满足大流检测的性能要求。因此,可以通过限制 Hash 冲突出现的数目(即单个 Hash 桶中流的条目数)来确保大流检测的性能。流的大小与传输速率相结合的双门限检测算法的具体描述见算法 1。

算法 1　流的大小与传输速率相结合的双门限检测

Alg. 1　Dual threshold elephant flowd etection combined flow size with transfer rate

输入:收到的报文 *Packet*

输出:*Packet* 对应的流是否为大流

```
1:报文 Packet 到达;
2:提取报文 Packet 的头部字段,生成所属的流 Flow 的
  标识符 Flow. Id;
3:Index = Hash(Flow. Id);
4:IF(在 Index 对应的 Hash 桶中查找到 Flow 对应的条
  目)
5:{    Flow. Size + = Packet. Size;
6:     IF(Flow. Size≥大流数据量门限)
7:     {  计算数据流 Flow 的传输速率 Flow. Rate;
8:        IF (Flow. Rate≥大流的速率门限)
9:        {  Flow 对应的流标识为大流; }
       }
  }
10:ELSE//在 Index 的 Hash 桶中未找到 Flow 对应的
   条目
11:{  min_Rate = MAX;
12:   取 Index 对应 Hash 桶的第一个流条目 F;
13:   WHILE((F. Size≠0)&& (F 不是桶的最后条目))
14:   {  计算流 F 的速率 F. Rate;
15:      IF(F. Rate < min_Rate)
16:      {  min_Rate = F. Rate;
17:         position = F 在 Index 桶中的位置;
         }
18:      ELSE
         {取 Index 的 Hash 桶中的下一流条目 F;}
      }
19:   IF(min_Rate < 大流的速率门限)
20:   {将 Index 桶中 position 的空间分配给新流 Flow;
21:      存储新流 Flow 的生成时间及字节数;}
  }
END
```

新的数据流到达,若无可用的存储空间,则在当前新流 Flow 对应的 Hash 桶中找出传输速率最小的流 minFlow。如果该数据流的传输速率小于淘汰门限,则将该流淘汰,释放出的空间分配给当前的新流 Flow,记录新流 Flow 的生成时间及其字节数;如果 minFlow 的速率高出淘汰门限,则忽略新流 Flow。这样的处理可能会导致大流的漏检,因此需要对该方案的漏检率进行分析。

2.3 算法的漏检率分析

设 Hash 桶的数目为 H,最大并发高速流的数目为 n,则 n 个流条目随机分布在 H 个 Hash 桶中,对于任意一个特定的 Hash 桶,每个流条目进入这一 Hash 桶的概率为 $1/H$。随机变量 η 表示任一 Hash 桶中包含的高速率条目数,n 个流中恰好有 k 个进入这一 Hash 桶的概率服从二项分布 $B(n,1/H)$,即 $P(k)=\Pr(\eta=k)=C_n^k(p)^k(1-p)^{n-k}\approx\frac{\lambda^k}{k!}e^{-\lambda}$,其中,$p=1/H$,$\lambda=N\cdot p=n/H$。设 Hash 桶的容量为 ω,即每个 Hash 桶最多存储 ω 个流条目;当 $\eta>\omega$ 时,根据算法可知,新到达报文所属的流无法存储到缓存中,从而导致流的漏检;随机变量 γ_i 表示第 i 个 Hash 桶中漏检的高速流数目,则

$$\gamma_i=\begin{cases}\eta-\omega, & \eta\geqslant\omega\\ 0, & \eta<\omega\end{cases}$$

γ_i 的期望值为:

$$E(\gamma_i)=\sum_{i=\omega+1}^{\infty}[(i-\omega)\cdot P(i)] \tag{1}$$

由式(1)得:

$$E(\gamma_i)=\sum_{i=0}^{\omega-1}[(\omega-i)\cdot P(i)]-(\omega-\lambda) \tag{2}$$

整个 Hash 表漏检的高速流条目数 $S=\sum_{i=1}^{H}\gamma_i$,高速流条目数的漏检率 $\varepsilon=S/n$,ε 的期望值 $E(\varepsilon)=H\cdot E(\gamma_i)/n$,即

$$E(\varepsilon)=\{\sum_{i=0}^{\omega-1}[(\omega-i)\cdot P(i)]-(\omega-\lambda)\}/\lambda \tag{3}$$

由于大流一定是高速流,因此大流的数目不超过高速流的数目,大流的条目数漏检率期望值小于等于 $E(\varepsilon)$。

定理 1 一条数据量为 V 字节的大流被漏检的概率为 Δ,则 $\Delta<\varepsilon^{[1+(V-E_{th})/\bar{S}]}$,$\bar{S}$ 表示报文的平均大小,E_{th} 是大流的数据量门限值,ε 是高速流漏检率的期望值。

证明:一条数据量为 V 字节的大流,其报文数 $N_p\approx V/\bar{S}$。由于 Hash 冲突的存在,大流的报文以 φ 的概率被存储到流检测缓存中。设收到第 X 个报文,该大流被存储到流检测缓存中;随机变量 X 服从几何分布,即 $X\sim G(\varphi)$,$\varphi=1-\varepsilon$。当大流条目在检测缓存中被创建时,其数据量的统计值少于门限值 E_{th} 时被漏检,即当 $V-(X-1)\bar{S}<E_{th}$ 时,该大流被漏检。因此当 $X>+(V-E_{th})/\bar{S}$ 时,大流被漏检,漏检概率为:

$$\Delta=\sum_{x=n}^{N_p}P(X=x)<\sum_{x=n}^{\infty}[\varepsilon^x(1-\varepsilon)] \tag{4}$$

由式(4)可得:

$$\Delta<\varepsilon^{[1+(V-E_{th})/\bar{S}]} \tag{5}$$

□

因为报文是网络设备收发与处理的基本单位,所以定理 1 从报文而不是字节数的角度分析了大流及其数据量的漏检率,这一分析的准确性是建立在大流的报文数与字节数线性相关的基础上的。从定理 1 可以看出:随着大流数据量的增加,任一大流被漏检的概率逐渐降低。

3 实验与结果分析

在满足一定的处理性能和存储开销的前提下,大流检测算法的准确性一直是研究与关注的重点。通常流检测算法的准确性包含漏检率(False Negative Rate, FNR)与误报率(False Positive Rate, FPR)两个方面。DSR 算法的检测门限同时根据速率和数据量识别活动大流,不存在误报问题,因此只对 DSR 算法的漏检率进行测试与分析。对于流量管理类应用,主要关注的是所检测的大流的总数据流量,因此测试了流条目数的漏检率与数据量的漏检率。记算法检测出的大流条目数为 N_f,网络流量中实际的大流条目数为 N_r,则流条目数的漏检率 $FNR_N=(N_r-N_f)/N_r$;记算法检出的大流数据量为 V_f,网络流量中实际的大流总数据量为 V_r,则数据量的漏检率 $FNR_V=(V_r-V_f)/V_r$。

3.1 网络数据集

不失一般性,测试采用了 6 组网络数据,这些网络流量数据分别是:来自广泛集成的分布式环境(Widely Integrated Distributed Environment, WIDE)项目中 MAWI 工作组[29]的骨干链路的流量数据 T2014、T2015、T2016,WAND 网络研究组[30]提供的网络边缘链路的流量数据 T2011 以及两所高校数据中心[31]的流量数据 UNV1 和 UNV2。流量数据的概要信息见表 1。

为了便于硬件实现,DSR 算法采用固定存储空间分配的方案。测试选取了 0.01% 链路容量作为大流检测的速率门限。对于网络流量数据 T2014、T2015、T2016,分配 4000 个流条目空间;对于网络流量数据 T2011、UNV1、UNV2,分配 800 个流条目空间。

表 1　网络流量数据概要

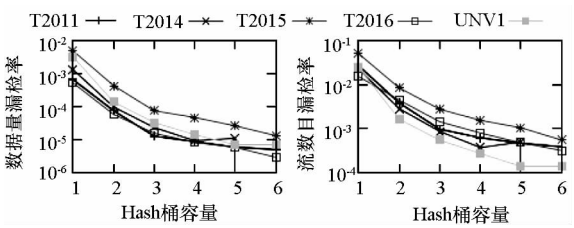
Tab. 1　Summary of network tracing data

流量数据	日期	时长/s	数据量/GB	报文数 ($\times 10^4$)	流数目 ($\times 10^4$)	平均速率/(MB/s)
UNV1	2009 - 12	3925	11.25	1732.3	55.7	2.935
UNV2	2010 - 01	9480	69.14	9881.7	19.1	7.468
T2011	2011 - 06	18001	135.76	16 768	398.7	7.723
T2014	2014 - 12	900	28.61	4435.6	372.4	32.552
T2015	2015 - 12	899	56.27	8829.8	318.7	64.094
T2016	2016 - 12	899	45.07	6011.3	629.9	53.797

3.2　Hash 桶容量对 DSR 算法准确性的影响

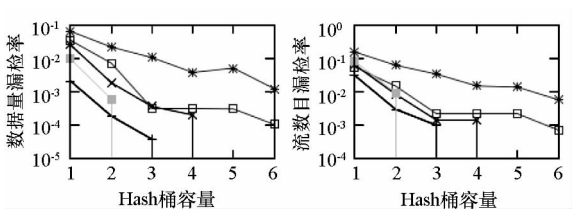
本节通过仿真实验测试了 Hash 桶的容量对 DSR 检测算法漏检率的影响。通常一个数据流至少包含一个满窗口尺寸的数据量时,才值得去进行管理与控制,因此选取 64 KB 作为一个大流检测的数据量门限值;少量的数据流超过了 4 MB,因此选择 4 MB 作为大流检测的一个数据量门限。

图 1 给出了在相同存储开销条件下,DSR 算法漏检率随 Hash 桶容量的变化趋势。从图 1 可以看出,随 Hash 桶容量的增加,无论是 DSR 算法流数目的漏检率还是数据量的漏检率均逐渐降低。图 1(a)、图 1(b)分别给出了检测门限为〈64 KB, 0.01%〉、〈4 MB, 0.01%〉时流数目与数据量漏检率的变化趋势。从图 1 可以看出,在 Hash 桶容量不小于 4 的情况下,所有数据量的漏检率均低于 0.6%。由于网络流量 UNV2 包含的大流数目较少,在各种检测门限下测试出的漏检率都为 0,因此在图 1 中没有体现出来。



(a) 检测门限为〈64 KB, 0.01%〉

(a) Detection threshold of 〈64KB, 0.01%〉



(b) 检测门限为〈4MB, 0.01%〉

(b) Detection threshold of 〈4MB, 0.01%〉

图 1　检测算法的漏检率随 Hash 桶容量的变化趋势

Fig. 1　Variation tendency of false negative rate with the capacity per Hash bucket

对比图 1(a)、图 1(b)的左右两图可以看出,数据量的漏检率比流数目的漏检率低至少一个数量级。这是因为一条大流包含的数据量越多,其漏检的概率就越低,从而被漏检的流通常包含的数据量相对较少,数据量的漏检率要低于流数目的漏检率,这与定理 1 的理论分析是一致的。定理 1 对漏检率的分析是建立在大流的报文数与字节数线性相关的基础上的,数据流的报文数与字节数的相关性如图 2 所示。从图 2 可以看出,大流的报文数与字节数高度线性相关,因此定理 1 的理论分析具有较高的准确性。

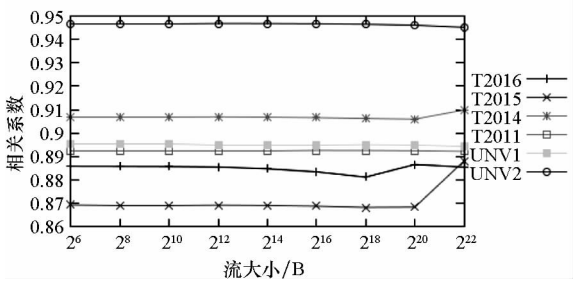


图 2　数据流的报文数与字节数的相关性

Fig. 2　Correlation between packets number and bytes number of data flows

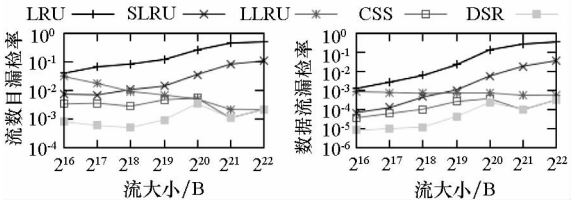
3.3　DSR 与其他检测算法准确性的比较

由于 LRU 类检测算法及 CSS 算法在进行流数目的淘汰或统计更新上只需要常数级的操作,选取 LRU、SLRU、LLRU、CSS 与 DSR 算法在检测的准确性上进行比较。LRU 算法及 CSS 算法主要依据流的数据量作为评价指标,而未考虑数据流的速率,即 DSR 与 LRU 及 CSS 算法的检测标准不完全一致,因此只进行大流漏检率的比较。对于 LRU 及其派生算法,每当报文到达,至少需要 4 次以上的存储器访问;对于 Hash 桶容量为 N 的 DSR 算法,每当报文到达,至多需要 N 次存储器的访问,因此选择 Hash 桶容量为 4 的 DSR 算法与 LRU 及其派生算法进行漏检率的比较。

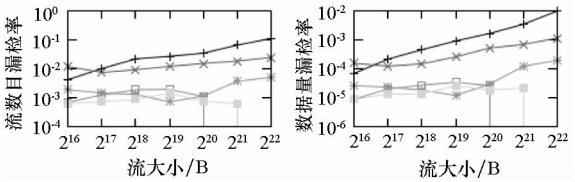
对于 SLRU 算法而言,采样的方式有基于字节数的周期性采样、基于字节数的随机采样、基于报文数的周期性采样、基于报文数的随机采样。

根据网络流量数据 T2011、T2014、T2015、T2016、UNV1、UNV2 的测试表明，基于字节数的随机性采样[15]优于其他采样方式，并且当采样率为1/1000时算法的准确性较好。因此图 3 中 SLRU 算法采用了采样率为 1/1000 的随机性采样方式。

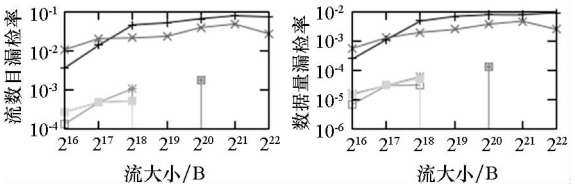
使用网络流量数据 T2011、T2014、T2015、T2016、UNV1、UNV2 测试了不同数据量门限(64 KB、128 KB、256 KB、512 KB、1 MB、2 MB、4 MB)和速率门限(0.01%)组合下的几种算法的漏检率，如图 3 所示。其中图 3(a)～(c)分别是使用数据集 T2016、T2011、UNV1 测试得出的结果。T2014、T2015 的测试结果与 T2016 相近，因此图 3 没有给出。网络流量数据 UNV2 中的大流数目较少，DSR 的检测出的漏检率均为 0，因此图 3也没有给出。



(a) T2016



(b) T2011



(c) UNV1

图 3 不同检测门限下几种算法漏检率的对比

Fig. 3 Comparison of different detection algorithm under different threshold

从图3 可以看出，总体而言，DSR 算法流数目的漏检率、数据量的漏检率均低于 LRU、SLRU、LLRU、CSS 等算法。LRU、SLRU、LLRU 算法漏检率高的主要原因在于：这类算法主要根据短时期内报文的命中率，而不是数据量的字节传输速率进行流的淘汰，同时报文大小并不是一致的，通常报文大小从几十字节到 1500 B 变化，所以在存储空间不大的情形下，即使数据流具有较高的传输速率但单位时间内的报文数相对较低时也会被淘汰，导致漏检。通过采样的方案，可以缓解由于大量突发到达的小流导致的漏检，即 SLRU 算法在采样率设置适当的情况下，准确性要优于 LRU 算法；LLRU 使用分级淘汰策略，可以进一步缓解突发小流对检测准确性的干扰，尽量将包含数据量多的流保持在缓存中。所以仅从漏检率的角度来看，与 LRU 及 SLRU 相比，LLRU 算法具有较高的准确性。CSS 算法依据报文的发送频率而不是传输速率，导致了检测的误差。而基于速率的淘汰机制可以有效抑制突发小流的干扰，将低速率的数据流及时从缓存中淘汰，因此 DSR 算法相比上述算法具有较高的准确性。

4 结论

针对高速网络中以流量管理与控制为主要目标的大流检测，提出了基于速率与数据量的 DSR 双门限检测方案。为了满足高速网络传输的性能需求，DSR 使用 Hash 表存储流检测的数据结构，将 Hash 表的冲突处理与流缓存替换相结合以实现高效的大流检测。对所提 DSR 算法的存储开销、性能及准确性进行了理论分析。实际网络流量数据测试结果表明：在相同存储开销下，随着 Hash 桶容量的增加，DSR 算法的漏检率逐渐降低；在相似的性能及存储开销条件下，DSR 算法优于 LRU 及其派生算法以及基于统计计数的 CSS 算法。

参考文献(References)

[1] Benson T, Akella A, Maltz D A. Network traffic characteristics of data centers in the wild[C]//Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, 2010: 267－280.

[2] Roy A, Zeng H, Bagga J, et al. Inside the social network's (datacenter) network [J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 123－137.

[3] Mahajan R, Floyd S, Wetherall D. Controlling high-bandwidth flows at the congested router[C]//Proceedings of the Ninth International Conference on Network Protocols, 2001: 192－201.

[4] Fred S B, Bonald T, Proutiere A, et al. Statistical bandwidth sharing: a study of congestion at flow level [J]. ACM SIGCOMM Computer Communication Review, 2001, 31(4): 111－122.

[5] Smitha, Kim I, Reddy A. Identifying long-term high bandwidth flows at a router [C]//Proceedings of the 8th International Conference on High Performance Computing, 2001: 361－371.

[6] Alizadeh M, Greenberg A, Maltz D A, et al. Data center TCP (DCTCP) [J]. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 63－74.

[7] Baker F, Fairhurst G. IETF recommendations regarding active queue management: RFC 7567[R/OL]. (2015－07－01)[2017－08－10]. http://www.rfc-editor.org/info/rfc7567.

[8] Papagiannakit K, Taft N, Diot C. Impact of flow dynamics on traffic engineering design principles [C]//Proceedings of Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, 2004: 2295 – 2306.

[9] Wang W, Sun Y, Salamatian K, et al. Adaptive path isolation for elephant and mice flows by exploiting path diversity in datacenters[J]. IEEE Transactions on Network and Service Management, 2016, 13(1): 5 – 18.

[10] Agarwal S, Kodialam M, Lakshman T V. Traffic engineering in software defined networks [C]//Proceedings of IEEE INFOCOM, 2013: 2211 – 2219.

[11] Cui W Z, Yu Y, Qian C. DiFS: Distributed flow scheduling for adaptive switching in FatTree data center networks [J]. Computer Networks, 2016, 105: 166 – 179.

[12] Le L, Aikat J, Jeffay K, et al. Differential congestion notification: taming the elephants[C]//Proceedings of the 12th IEEE International Conference on Network Protocols, 2004: 118 – 128.

[13] Voorhies S, Lee H, Klappenecker A. Fair service for mice in the presence of elephants[J]. Information Processing Letters, 2006, 99(3): 96 – 101.

[14] Estan C, Varghese G. New directions in traffic measurement and accounting: focusing on the elephants, ignoring the mice[J]. ACM Transactions on Computer Systems, 2003, 21(3): 270 – 313.

[15] Bai L, Chen C. Frequent items mining algorithm over high speed network flows based on double hash method [J]. International Journal of Future Generation Communication and Networking, 2016, 9(5): 75 – 82.

[16] Chen L, Mei Q L. Mining frequent items in data stream using time fading model [J]. Information Sciences, 2014, 257: 54 – 69.

[17] Cafaroa M, Pulimenoa M, Epicocoa I, et al. Mining frequent items in the time fading model[J]. Information Sciences, 2016, 370/371: 221 – 238.

[18] Manku G S, Motwani R. Approximate frequency counts over data streams[C]//Proceedings of the VLDB Endowment, 2012: 1699 – 1699.

[19] Dimitropoulos X, Hurley P, Kind A. Probabilistic lossy counting: an efficient algorithm for finding heavy hitters[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(1): 5 – 15.

[20] Che L C, Qiu B, Wu H R. Improvement of LRU cache for the detection and control of long-lived high bandwidth flows[J]. Computer Communications, 2005, 29 (1): 103 – 113.

[21] Choi Y, Joung J. A novel algorithm for detection of elephant flows: landmark-LRU with recycle [C]//Proceedings of International Conference on Future Information & Communication Engineering, 2013: 159 – 167.

[22] Metwally A, Agrawal D, Abbadi A E. Efficient computation of frequent and top-*k* elements in data streams [C]//Proceedings of International Conference on Database Theory. Germany: Springer, 2005: 398 – 412.

[23] Ben-Basat R, Einziger G, Friedman R, et al. Heavy hitters in streams and sliding windows[C]//Proceedings of the 35th Annual IEEE International Conference on Computer Communications, 2016.

[24] Silva J M C, Carvalho P, Lima S R. Analysing traffic flows through sampling: a comparative study[C]//Proceedings of IEEE Symposium on Computers and Communication, 2015: 341 – 346.

[25] Mori T, Uchida M, Kawahara R, et al. Identifying elephant flows through periodically sampled packets[C]//Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement. USA: ACM, 2004: 115 – 120.

[26] 白磊, 田立勤, 陈超. 基于 TCBF-LRU 的高速网络大流检测算法[J]. 计算机研究与发展, 2014, 51(s2): 122 – 128.
BAI Lei, TIAN Liqin, CHEN Chao. A TCBF-LRU algorithm for identifying and measuring elephant flows in high speed network[J]. Journal of Computer Research and Development, 2014, 51(s2): 122 – 128. (in Chinese)

[27] Zhang Z, Wang B Q, Lan J L. Identifying elephant flows in internet backbone traffic with bloom filters and LRU[J]. Computer Communications, 2015, 61: 70 – 78.

[28] 张震, 汪斌强, 张风雨, 等. 基于 LRU-BF 策略的网络流量测量算法[J]. 通信学报, 2013, 34(1): 111 – 120.
ZHANG Zhen, WANG Binqiang, ZHANG Fengyu, et al. Traffic measurement algorithm based on least recent used and Bloom filter[J]. Journal on Communications, 2013, 34(1): 111 – 120. (in Chinese)

[29] MAWI Working Group Traffic Archive[DB/OL]. [2017 – 08 – 10]. http://mawi.wide.ad.jp/mawi/.

[30] Waikato Internet Traffic Storage[DB/OL]. [2017 – 08 – 10]. http://wand.net.nz/wits/waikato/8/20110606 – 000000 – 0.php.

[31] Data Set for IMC 2010 Data Center Measurement[DB/OL]. [2017 – 08 – 10]. http://pages.cs.wisc.edu/ ~ tbenson/IMC10_Data.html.