计算机科学
COMPUTER SCIENCE

RBFRadar:基于可编程数据平面检测价值突发流

吴艳妮, 周政演, 陈翰泽, 张栋

引用本文

吴艳妮, 周政演, 陈翰泽, 张栋. RBFRadar:基于可编程数据平面检测价值突发流[J]. 计算机科学, 2024, 51(4): 48-55.

WU Yanni, ZHOU Zhengyan, CHEN Hanze, ZHANG Dong. RBFRadar:Detecting Remarkable Burst Flows with Programmable Data Plane [J]. Computer Science, 2024, 51(4): 48-55.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231000213

RBFRadar:基于可编程数据平面检测价值突发流

吴艳妮[1,2]　周政演[3]　陈翰泽[1]　张　栋[2,4]
1 福州大学计算机与大数据学院　福州 350108
2 泉城省实验室　济南 250100
3 浙江大学计算机科学与技术学院　杭州 310013
4 福州大学至诚学院　福州 350002
(231020045@fzu.edu.cn)

摘　要　在各种网络流量中,突发是一种常见且重要的流量模式。突发会增大网络时延并影响应用性能,因此对突发流的检测、分析和缓解对于提升网络性能和鲁棒性是有意义的。然而,当前基于逐次突发的检测方案存在显著的带宽开销和高用户负担问题。文中通过观察并分析多个场景下的突发流量特征,提出了价值突发流(Remarkable Burst Flow,RBF)检测,在降低带宽开销的同时,减少了传统突发检测中的密集手工劳动和专家经验要求,减轻了网络管理者的负担。RBFRadar是基于Sketch数据结构的框架,支持可编程数据平面上的RBF检测,在一段时间内观察流级别的突发性。该框架仅产生有限的内存占用和低时间复杂性,其原型可在PISA架构上实现。实验结果表明,在检测RBF的准确性方面,RBFRadar的F1分数是现有方案的5.6～23.4倍;在带宽开销方面,与基于逐次突发的检测方案相比,RBFRadar可降低84.62%～98.84%的带宽开销。

关键词:突发流检测;Sketch;网络测量;可编程数据平面;数据中心网络
中图分类号　TP393

RBFRadar:Detecting Remarkable Burst Flows with Programmable Data Plane

WU Yanni[1,2],ZHOU Zhengyan[3],CHEN Hanze[1] and ZHANG Dong[2,4]
1 College of Computer and Data Science,Fuzhou University,Fuzhou 350108,China
2 Quan Cheng Laboratory,Jinan 250100,China
3 College of Computer Science and Technology,Zhejiang University,Hangzhou 310013,China
4 Zhicheng College,Fuzhou University,Fuzhou 350002,China

Abstract　Burst is a common and important traffic pattern in diverse network traffics. Since bursts may increase network latency and have a non-trivial impact on application performance,the efforts to detect,analyze and mitigate burst flows are meaningful for improving the performance and robustness of network. However,existing per-burst-based detection schemes face the limitations of significant bandwidth overheads and high user burdens. This paper proposes the detection of remarkable burst flows(RBFs) via observing and analyzing the characteristics of burst flows in various scenarios. The detection of RBFs reduces the bandwidth overheads. At the same time,such detection process avoids the requirements of intensive manual labor and expert experience,and mitigate the burdens of network operators. We propose RBFRadar,a Sketch-based RBF detection framework that supports RBF detection on programmable data plane,observing flow-level burstiness in a period. We prototype RBFRadar in PISA architecture with limited memory footprints and low time complexity. Experiments demonstrate that the F1-score of RBFRadar in RBF detection is 5.6 times to 23.4 times higher than that of existing schemes. Compared with per-burst detection,the bandwidth overhead could be reduced by 84.62% to 98.84%.

Keywords　Burst flow detection,Sketch,Network measurement,Programmable data plane,Data center network

1　引言

突发是一种流速在短时间内激增的流量模式。突发流在各种生产网络中广泛存在,例如数据中心网络[1]、广域网[2-3]等。突发可能对网络和应用的各方面性能产生显著的影响。对于如万维网搜索[4-5]等对时延敏感的应用而言,突发流可能

到稿日期:2023-10-29　返修日期:2024-01-26
基金项目:国家重点研发计划专项(2023YFB2904000,2023YFB2904005);泉城省实验室(QCLZD202304);山东省实验室项目(SYS202201)
This work was supported by the National Key R&D Program of China(2023YFB2904000,2023YFB2904005),Quan Cheng Laboratory(QCLZD202304) and Research Project of Provincial Laboratory of Shandong,China(SYS202201).
通信作者:张栋(zhangdong@fzu.edu.cn)

引入高响应延迟、数据包丢失和性能波动[6],导致服务质量(Quality of Service,QoS)显著下降,甚至违反服务水平协议(Service Level Agreement,SLA)等。

对突发进行检测是缓解突发造成的网络质量下降的前提条件[7]。目前,部分研究将突发抽象为数据流模式[8],在主机端利用数据项到达速率的突增和突降,标志并检测数据流中突发的出现。还有一部分研究[7,9-10]致力于在可编程交换机上检测微突发,即持续时间在微秒甚至毫秒级别的突发。通常来说,当队列长度突然增大至超过某个预先定义的阈值时,交换机会检测到微突发,并将微突发的相关信息上报至控制平面用于进一步分析。与主机端相比,数据平面对网络内的信息有更强且更及时的可见性,并且新兴的 P4 语言[11]支持灵活的自定义数据平面操作。因此,这种基于可编程数据平面的解决方案是有优势的。

现有的研究工作都采用基于逐次突发的检测方案,认为每次突发都是无关联的,独立地检测并报告每次突发。这种基于逐次突发的方案可以捕获所有的突发事件,但也引入了高带宽开销,并明显增大了网络管理者的负担。其中,在带宽开销方面,数据平面向控制平面上报所有的突发事件。由于突发在流量中普遍存在[7,9-10],这样的突发报告方式会消耗大量数控平面之间的带宽资源。在用户负担方面,网络管理者需要在控制平面处理大量的流量信息以定位感兴趣的突发流,这个过程需要许多额外劳动甚至专家经验。以数据中心网络为例,控制器可以在短时间内积累数量众多的突发信息[1,12-13],但其中只会有少数的突发流指示有价值的网络事件,例如网络攻击[14-16]、需要被优化的应用层行为[17]等。对于网络管理者而言,在许多良性的突发流中区分出重要的突发流,需要密集的手工劳动并且非常耗时[14]。因此,较高的用户负担会导致无法快速排查网络故障、及时检测攻击以及优化网络需求[14,18-21]。

针对上述现有研究的局限性,本文研究在突发检测中,如何能够以低成本获取对突发流的深入见解。一种直观的思路是,针对重要的突发进行检测,并过滤掉不重要的突发。这种差异化对待的方式能够降低控制器端的数据量并减轻用户负担。具体地,例如流量模式、攻击等网络事件,具有明显影响且管理者可以执行干预操作。这样的网络事件在生成突发流的同时,倾向于长期反复地引入网络拥塞。良性突发流[14]在几乎所有流量中都存在,相比之下,这些网络事件引入的突发流数量较少,但值得被专门定位以执行对应的操作。为了在生产流量中定位上述网络事件,需要对这些事件所产生的突发流特征进行分析。为此,本文在接下来一些场景中对突发特征进行了观察,详细的验证实验将在第 5 章中进行说明。

1)通信模式。在分布式系统中,依赖于分布式通信的应用(例如,分布式机器学习[22]、大数据并行计算[23]、万维网搜索等)在通信过程中反复生成具有 incast 模式[17]的流量。在 incast 中,多个信息源在短时间内向一个接收方同时发送数据。因此,持续时间短、数据量大的突发频繁出现[24]。

2)网络攻击。脉冲波拒绝服务(Pulse-wave DoS)攻击[16]由一组持续时间短的、使用不同攻击向量的高速率突发组成。低速率拒绝服务攻击(LDoS)的特征是一系列周期性的、高速率短时长的突发[15,25]。

3)应用层优化需求。在数量和类型迅速增长的各种服务中,一部分应用生成具有显著突发性的流量。例如视频流服务周期性地请求大量高质量的视频块,在线流媒体中在各时间段产生不同类型的流量峰值[26]。

在上述场景中,由于流量数据集中以突发的形式出现,这些对网络性能有明显影响的事件产生了大量的突发数据。因此,与呈现良性突发的流相比[14],这部分流的突发数据在总数据中的占比更高。为了支持数据平面上的有效检测,这些流量的特征需要被总结为一种流量模式。因此,本文基于流中的突发数据占比,定义了一种新的流量模式,称为价值突发流(Remarkable Burst Flow,RBF)。其中,突发数据占比情况可通过计算突发数据包占总数据包数量的比例得到。为满足数据平面的线速率操作需求,突发数据包的定义主要基于数据包间的时间间隔特征。在应用场景方面,RBF 检测的应用范围远大于上述列举的场景,还可用于网络瓶颈定位、微突发缓解等各种任务。

为了在可编程数据平面有效地检测 RBF,本文提出了基于 Sketch 的检测框架 RBFRadar。总的来说,该框架针对 RBF 的特性,设计了对应的桶字段用于跟踪潜在的 RBF,并通过基于突发性的投票规则,降低因检测过程中哈希冲突导致的抖动而造成的影响。RBFRadar 可在 PISA 架构[27]上部署并获得对网内信息的可见性。本文在公开数据集[28-29]上对该框架的准确性、参数敏感性和带宽开销进行了测试。实验结果表明,RBFRadar 能够以较高的准确度检测 RBF,并降低 84.62%以上的带宽开销,由于 RBFRadar 具有紧凑数据结构的良好特性,在检测任务中只产生小且静态的内存开销和低时间复杂度。

综上所述,本文的贡献包含以下 3 个方面:

1)在突发检测中,首次提出检测价值突发流(RBF)问题以及公式化的流量模式定义,以克服现有基于逐次突发的检测方案在带宽开销和用户负担方面的局限性。

2)提出了一个基于 Sketch 数据结构的框架 RBFRadar,作为在可编程数据平面检测 RBF 的有效方案,并提出了基于突发性的投票机制,以处理 RBF 检测过程中的哈希冲突问题。

3)基于 PISA 架构实现 RBFRadar 原型,并使用公开数据集 CAIDA 进行准确性和带宽开销测试。实验结果表明,在 RBF 检测的准确性方面,该框架的 F1 分数是现有方案的 5.6~23.4 倍。在带宽开销方面,与逐突发检测方案相比,可减少 84.62%~98.84%的带宽需求。

本文第 2 章介绍了背景知识和动机;第 3 章对相关定义和系统问题进行了描述;第 4 章阐述了基于 Sketch 的检测框架 RBFRadar 的具体设计;第 5 章进行了实现和实验验证,以及实验结果分析;第 6 章回顾了相关工作;最后总结全文并展望未来工作。

2 背景和动机

2.1 突发流与微突发

在各种生产网络中,突发是一种常见的流量模式,以持续

时间短的高流速为特征。如图1中所展示的例子，与速率稳定的流(见图1中的灰色线条)相比，虽然突发流(见图1中的黑色线条)在大部分时间内速率较低，但在突发期间(如t_1到t_2)流速突然增大，瞬时速率可达到平均速率的数十倍至数百倍。微突发是突发的一种，通常作为一种持续时间极短的瞬时拥塞事件被检测[7,9-10]，造成交换机队列快速堆积，导致较长的排队时延甚至丢包。突发的产生存在通信模式、传输协议、系统批处理方案等多种因素的影响[2]，因此完全避免突发的出现是非常困难的。

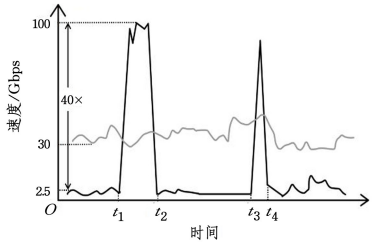

图1 突发流与稳定流量示例

Fig. 1 Example of burst flow and stable traffic

2.2 可逆 Sketch

Sketch是一种基于散列的数据结构，可以在高速网络环境中，将庞大的数据流信息记录在小的存储空间中。与传统的基于采样的方案相比，基于Sketch的网络测量是一种良好的替代方法[28]，提供了高资源效率以及可数学证明的误差上界保证。其中，有一类Sketch属于可逆Sketch，支持通过数据结构本身恢复流键。以重流(即heavy hitter与heavy changer)检测为例，Count-Min Sketch[29]这样的非可逆Sketch需要检索整个流键集合才能恢复所有重流，而MV-Sketch这样的可逆Sketch能够仅通过数据结构自身，恢复所有重流的流键[30-31]。对于许多网络测量任务而言，可逆性是一个良好的特性，支持快速获取网络测量结果。

2.3 可编程数据平面

由于突发的持续时间可能很短，传统网络的端到端测量方案难以准确检测突发，并且其固定功能的数据平面只能提供有限的信息和操作。可编程数据平面[32]和P4语言[11]的出现，为检测和处理流量突发提供了可自定义的操作，并可以提供细粒度的网络状态观察，例如队列和内存占用等。许多现有工作[7,9-10]利用可编程数据平面的特性，获取突发期间的详细队列情况用于肇事流分析[7]等任务。

2.4 基于逐次突发方案的局限性

现有研究致力于对逐次突发进行独立的检测、分析和缓解，但基于逐次突发的思路将引入显著的带宽开销和网络管理者的高用户负担。在带宽开销方面，由于突发在流量中普遍存在，数量较大，对每个突发事件进行报告容易与其他测量任务产生带宽资源的竞争。以数据中心网络为例，该网络中有100台交换机，每秒报告100次突发事件，其中每次突发将上报一个大小为49.125 KB的MV-Sketch[10,30]，将需要39.3 Gbps的数控平面间带宽。在用户负担方面，基于逐次突发的方案将所有检测到的突发不加区别地报告，其中大部分突发是良性的，对网络性能的影响不明显或非常短暂[2,14]，而小部分突发意味着重要的网络事件(例如LDoS攻击)或网络优化需求(例如网络瓶颈)的存在。例如，在上述场景中，控制器仅在10 s内就积累了10^6次突发的信息，而其中可能只有少数几个突发流指示攻击的存在。因此，网络管理者需要大量的额外工作量甚至专家经验来定位需要被关注的突发流。

2.5 网络层对突发性测量指标的需求

通常而言，网络管理者通过获取网络层设备上的测量指标，执行具体的管理操作(例如故障排除)。网络管理者通过直接观察一些测量指标捕获重要的信息(例如携带特殊标志的流)，并将其作为管理决策的参考。现有用于逐次突发检测的指标包括队列长度[9-10]和流速特征[8]，但其中队列长度只对整体流量进行观察，提供粗粒度的信息而非流级别的信息，且流速特征难以在数据平面进行检测，无法及时获取网络信息。考虑到现有指标的不足，网络管理者需要一个网络测量指标，用于反映具有显著且持续突发性的流。本文提出的RBF作为流级别的指标，通过过滤掉突发性较弱的流，提升数据的信息价值密度。此外，RBF的检测可部署在数据平面上，只需要观察数据包之间的时间间隔，不需要复杂的浮点计算。因此，RBF可以作为网络测量中的一个有效指标。

3 定义与问题描述

3.1 定义描述

本文中常用的符号如表1所列。

表1 常用符号

Table 1 Common symbols

符号	含义
$\mathcal{T}$	数据包之间的平均间隔
d	Sketch配置的行数
w	Sketch配置的列数
α	价值突发流的阈值参数
α'	驱逐规则的配置参数
β	突发数据包的阈值参数
γ	RBF在总数据流中的占比
D	一段时间内的流数据包总数
$P(\cdot)$	事件发生的概率
b	当前流突发数据包数量
u	当前流总数据包数量
χ	周期内未到达数据包数

目前对于突发还没有完全统一的定义，但突发通常被理解为持续时间短的高流速时段，在突发期间，流速率远远高于平均速率[6,13]。在数据量较大的突发频繁发生的流中，由于数据集中在突发期间发送，突发数据在总流量中的占比高于其他流。基于这些观察，本文提出了价值突发流(RBF)的定义，公式化的定义如下。

在一个示例中，考虑一个流键为给定〈源IP地址，目的IP地址〉的流f。在示例中，由于流f是一次脉冲波拒绝服务攻击的一部分，在测量点上f频繁出现突发，且在突发时注入大量攻击数据。

定义1(突发数据包) 每个流都由一个流键(例如五元组等)进行标识。给定一个流F，该流由一系列时间有序的数据包构成，其中两个相邻数据包p_i和p_j之间的时间间隔被记为$t_{i,j}$。在时间维度上，流F被分割为若干个等长的时间段。在第i个时间段内，数据包之间的平均时间间隔被记为$\mathcal{T}$。给定一个常量参数$\beta(\beta>1)$，如果在第i个时间段内，存在两个

相邻数据包p_i和p_j(其中$i<j$)满足式(1),那么这两个数据包被认为是第i个时间段内的突发数据包。

$$t_{i,j}<\frac{1}{\beta}\cdot\mathcal{T} \tag{1}$$

示例说明:在第i个时间段中,流f的平均数据包间隔$\mathcal{T}$为200 ns,同时参数β被设置为20。如果存在两个相邻数据包之间的间隔为5 ns,由于$5ns<\frac{1}{20}\cdot 200ns$,这两个数据包被认为是突发数据包。

定义2(价值突发流) 给定一个流F,该流被分割为若干个等长的时间段。在第i个时间段内,来自F的所有数据包总数记为D,其中突发数据包的数量被记为d_{bursty}。给定一个常量参数$\alpha(0<\alpha<1)$,如果满足式(2),则在第i个时间段内,F被认为是价值突发流。

$$d_{\text{bursty}}\geqslant\alpha\cdot D \tag{2}$$

示例说明:在第i个时间段中,流f的所有数据包总数为5×10^7,在这些数据包中,f生成了4.6×10^7个突发数据包。阈值α被设置为0.7。因此,突发数据包的占比为0.92,高于阈值α,来自脉冲波拒绝服务攻击的流f被检测为价值突发流。

3.2 系统与问题描述

为了将RBF的定义应用于实际的检测工作,需要在数据平面检测RBF并报告,以进行进一步分析。该过程主要包括3个步骤:首先,用户通过配置部分参数来指定检测的突发流;其次,可编程数据平面自动检测价值突发流,并在每个时间段结束时向控制器进行通报;最后,控制器基于所接收的信息,进行进一步的分析和管理操作。

在有限的内存资源下观察流级别的突发性是系统需要解决的问题之一。据我们所知,目前在可编程数据平面上已有工作根据队列特征定义和检测突发,没有将突发作为流级别的流量模式进行检测[9-10]。为了解流的突发性,一种简易的方法是为每个流分配若干个寄存器以监控流速的变化情况。但由于网络设备提供的硬件资源(例如SRAM存储)有限,为每个流维护时间序列数据是不现实的。因此,在RBF检测中,如何选择被监控的流子集是一个关键的问题。

系统应处理的第二个问题是减少哈希冲突导致的抖动所带来的影响。准确的RBF检测需要以持续的流级别观察为基础。为了在数据平面简化哈希冲突的处理,一种简易的方法是,一旦发生碰撞,就驱逐正在被监控的流并插入新到达的流。但在这样的操作下,发生哈希冲突的桶中将出现频繁的替换,进而导致这些桶无法有效地跟踪任何一个流。基于频率的投票机制(例如主票选算法[33])可以筛选出高频率的数据项,但目前还缺乏一种基于突发性的投票机制,以动态地票选出具有更强突发性的流。

针对RBF检测过程中所采用的突发性指标较为单一的问题,本系统可与网络管理者所使用的其他网络测量指标兼容,从而满足不同的测量需求。

4 设计

4.1 设计总览

RBFRadar是一个基于Sketch数据结构的框架,支持以数据包的线性处理有效地检测价值突发流。为适应可编程数据平面的高线速特征,RBFRadar应无循环地处理数据流。由于数据平面上的存储资源有限,框架的部署应尽可能轻量化。RBFRadar关注以下设计目标。

1)高RBF检测准确性。在RBF检测任务中,RBFRadar应达到良好的精度、召回率、F1分数等准确性要求。

2)小且静态的内存需求。RBFRadar应维护一个紧凑的数据结构,仅产生小且有限的内存占用。这样的静态内存分配方式允许在检测过程中的轻量化内存访问,并为硬件加速提供潜力。

3)可逆性。由于可编程数据平面上流键搜索空间庞大,通过枚举流键在Sketch中搜索价值突发流是不可行的,因此RBFRadar应是可逆的,在每个时间段结束时,能够只通过其数据结构本身恢复出所有检测到的RBF。

4.2 基于Sketch的RBF检测

为满足内存要求,RBFRadar采用Sketch作为基础数据结构。与Count-Min Sketch[29]相似,RBFRadar被初始化为二维的桶数组,其中每个桶记录一条流的突发性信息。为了缓解哈希冲突造成的抖动,每条流会被哈希到多个桶中,但只对其中一个桶进行操作。

图2给出了RBFRadar的数据结构,由d行w列的二维桶数组构成。位于第i行第j列的桶被记为$B_{(i,j)}$,其中$1\leqslant i\leqslant d\wedge 1\leqslant j\leqslant w$,每个桶中包含5个字段,具体功能如下。RBFRadar与d个两两独立的哈希函数关联,每个哈希函数h_i将每个输入数据包的流键哈希至第i行的w个桶中的一个桶。图2中相关变量的含义如下:

1)$key_{(i,j)}$,记录当前正在被监控的流键。

2)$\mathcal{T}_{(i,j)}$,该数值表示被跟踪的流的平均数据包间间隔。

3)$Times_{(i,j)}$,该字段记录当前该流最后一个到达的数据包时间戳。

4)$bursty_{(i,j)}$,对该流在当前时间段的突发数据包进行计数。

5)$total_{(i,j)}$,该字段表示流在当前时间段的所有数据包总数。

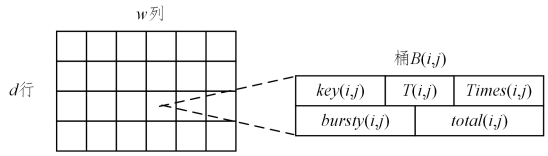

图2 RBFRadar的数据结构

Fig. 2 Data structure of RBFRadar

RBFRadar中的每个桶支持两个基本操作:更新操作,将每个输入数据包插入到数据结构中;查询操作,返回在当前时间段中被跟踪的流的突发数据包占比。算法1体现了更新操作的主要过程。当一个数据包被哈希到d个桶之后,存在3种情况:1)该数据包所在的流已经被其中一个桶跟踪(见算法1中的第1—2行),则该数据包可直接更新对应流的信息(见算法1中的第3—6行);2)至少存在一个桶为空(见算法1中的第8—9行),则该新流可将信息插入至空桶中(见算法1中的第10行);3)所有被哈希到的桶都被其他流占用,在这种

情况下，将查找其中突发数据包占比最小的桶(见算法1中的第12行)并进行检查，如果该流的突发数据包占比低于预先定义的阈值，则该流将被新流替换(见算法1中的第13－14行)。否则，如式(3)和式(4)所示，$bursty_{(i,j)}$字段和$total_{(i,j)}$字段将同时递减，以降低该桶中的突发数据包占比(见算法1中的第15－16行)。算法2同时演示了查询操作(见算法2中的第10－12行)，在查询操作中，突发数据包的占比通过桶中字段计算得到，该结果可与参数α进行比较，获得该流是否为价值突发流的判断。

$$bursty_i \cdot (total_i - 1) > total_i \cdot (bursty_i - 1) \tag{3}$$

$$\frac{bursty_i}{total_i} > \frac{bursty_i - 1}{total_i - 1} \tag{4}$$

算法1 更新算法

输入：流键k，时间戳t，哈希函数h_i

1. for i←1 to d do
2. if $key_{(i,h_i(k))}$==k then
3. if t－$Times_{(i,h_i(k))} \leqslant \frac{1}{\beta} \cdot \mathcal{T}_{(i,h_i(k))}$ then
4. $bursty_{(i,h_i(k))}$←$bursty_{(i,h_i(k))}$+1
5. end if
6. $total_{(i,h_i(k))}$与$Times_{(i,h_i(k))}$更新后结束
7. end for
8. for i←1 to d do
9. if $key_{(i,h_i(k))}$==NULL then
10. INSERT(k,t,$B_{(i,h_i(k))}$,origin_$\mathcal{T}$)
11. end for
12. (x,y)←MINP(k)
13. if $bursty_{(x,y)}/total_{(x,y)} \leqslant \alpha'$ then
14. INSERT(k,t,$B_{(i,h_i(k))}$,origin_$\mathcal{T}$)
15. else
16. $bursty_{(x,y)}$与$total_{(x,y)}$－－=1
17. end if

如图3所示，RBFRadar的工作流程可分为以下阶段：1)参数设置。在检测工作开始前，管理者可通过控制平面设置参数，以指定价值突发流。2)更新操作。在RBF检测的工作流程中，每个输入的数据包都将触发更新操作，具体可包括哈希映射、投票操作以及追踪更新。3)查询操作。在每个时间段结束时，控制平面对每个桶执行查询操作，并报告所有检测到的RBF。查询操作包含数据清除这一子过程，为了满足RBF检测的持续性，正在跟踪价值突发流的桶将在相邻时间段之间保留$key_{(i,j)}$，$\mathcal{T}_{(i,j)}$，$Times_{(i,j)}$字段用于持续跟踪，而没有跟踪到RBF的桶将被清空用于下一个时间段。

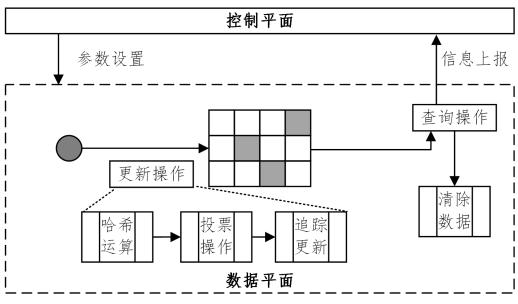

图3 RBFRadar的工作流

Fig. 3 Workflow of RBFRadar

算法2 桶操作算法

1. function INSERT(k,t,$B_{(i,h_i(k))}$,origin_$\mathcal{T}$)
2. $key_{(i,h_i(k))}$,$Times_{(i,h_i(k))}$,$\mathcal{T}_{(i,h_i(k))}$初始化
3. $bursty_{(i,h_i(k))}$←0,$total_{(i,h_i(k))}$←1
4. end function
5. function MINP(k)
6. for i←1 to d do
7. 记录$bursty_{(i,h_i(k))}/total_{(i,h_i(k))}$最小的桶索引
8. end for
9. end function
10. function QUERY($B_{(i,j)}$)
11. return $bursty_{(i,j)}/total_{(i,j)}$
12. end function

4.3 理论分析

RBFRadar配置有d行w列的桶，定理1说明了RBFRadar的时空复杂度，定理2说明了RBFRadar检测所跟踪价值突发流的误差上限。

定理1 RBFRadar的空间占用为$O(dw \cdot \log n)$，每个数据包的更新时间为$O(d)$，恢复并返回所有RBF的检测时间为$O(dw)$。

定理1的证明过程如下。

证明：RBFRadar中的每个桶最多需要存储一个长度为$\log n$比特的流键，两个计数器以及两个固定长度的寄存器，因此RBFRadar的空间占用为$O(dw \cdot \log n)$。由于每个数据包的更新需要进行d次哈希操作，访问到d个桶，因此消耗了$O(d)$的时间。恢复并返回所有的RBF需要遍历所有$d \cdot w$个桶，对于每个桶，只需要进行一次简单的查询操作，这个过程需要$O(dw)$的检测时间。

定理2 对于RBFRadar跟踪的价值突发流，RBFRadar检测RBF的误差上限为$\frac{\gamma^d}{w^d} \cdot P\left(\chi \geqslant \frac{\alpha' u - b}{\alpha' - 1}\right) \cdot (1-\gamma)$。

定理2的证明过程如下。

证明：在RBFRadar的一个检测周期内，价值突发流在总数据流中的占比为$\gamma(0<\gamma<1)$，当一个非价值突发流所映射到的一组桶都正在追踪RBF，则其所触发的投票操作可能导致RBF没有被报告为价值突发流。首先，非价值突发流同时与d个RBF发生哈希碰撞的概率如式(5)所示：

$$(1-\gamma) \cdot \gamma^d \cdot \left(\frac{1}{w}\right)^d \tag{5}$$

进一步地，根据投票规则，只有当前突发数据包占比最低的RBF会被处理，考虑最坏情况，该RBF之后没有数据包到达，其当前占比为b/u，则当非价值突发流的剩余数据包数量χ满足如式(6)所示的不等式时才会发生驱逐。

$$\frac{b-\chi}{u-\chi} \leqslant \alpha' \tag{6}$$

χ满足此不等式的概率使用$P(\cdot)$函数表示。因此，价值突发流检测的误差上限如式(7)所示：

$$\frac{\gamma^d}{w^d} \cdot P\left(\chi \geqslant \frac{\alpha' u - b}{\alpha' - 1}\right) \cdot (1-\gamma) \tag{7}$$

5 实现与实验

5.1 实验设置

本文实验在 DELL PowerEdge R750xs 服务器上运行，该服务器具有如下配置：Intel(R) Xeon(R) Silver 4399Y CPU (2.8 GHz，2NUMA，8 核 16 线程)，64 GB DDR4 RDIMM RAM。本文基于 HTTPDoS[28]，CAIDA[29] 等真实流量进行实验。

1)与其他流量相比，许多影响明显的网络事件(例如拒绝服务攻击)产生的突发流更符合价值突发流的特征(见实验 1、实验 2)。

2)RBFRadar 可平衡内存和准确性，并在检测并报告价值突发流的任务中获得良好的精度、召回率和 F1 分数(见实验 3－实验 5)。

3)RBFRadar 可大幅降低报告突发流信息所需的带宽开销，与现有工作相比，减少了 84.62%～98.84%的带宽开销(见实验 6)。

本文采用基于 PISA 架构的 P4 可编程软件交换机 Bmv2[34]实现 RBFRadar，主要包含 Sketch 的初始化、数据包的映射、数据结构的更新。在 Sketch 的初始化过程中，由于 P4 语言缺少对二维数组的直接支持，本文代码参考 MV-Sketch[30] 的实现，使用一组寄存器进行 Sketch 数据结构的维护。

5.2 前置验证实验

实验 1　与非攻击性的流量相比，一些网络攻击产生的流具有更高的突发数据包占比。在本实验中，采用一个公开的 HTTP 拒绝服务攻击数据集[28]进行验证，该数据集中包含良性节点和恶意节点生成的网络事件，并包含多种攻击模式。图 4 给出了 DDoS 流和非 DDoS 流的平均突发数据包占比，其中参数 β 在 10～50 之间均匀取值。可以看到，无论 β 的取值如何变化，来自拒绝服务攻击的流的突发数据包占比都显著高于非攻击性的流。

实验 2　一部分应用生成具有显著突发性的流量。图 5 给出了来自各种应用的流中，突发数据包在总数据包中的占比。实验结果表明，有一部分应用生成非常稳定的流量，例如基于 SCP 的远程文件传输。同时，来自在线文本翻译这类应用的流表现出较弱的突发性。但有一些应用，例如网页搜索和在线流媒体，生成的流量具有显著突发性，这意味着这部分应用可能需要应用层行为的优化、网络拓扑或负载均衡策略的改进[24]。

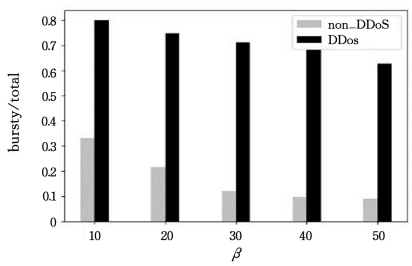

图 4　实验 1 中攻击流和非攻击流的突发数据包占比

Fig. 4　Proportion of bursty packets in DDoS flows and non-DDoS flows in experiment 1

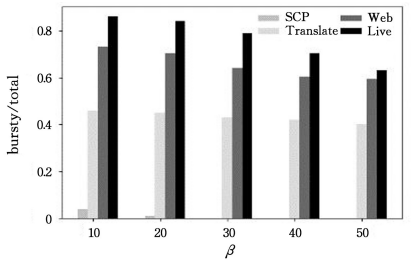

图 5　实验 2 中不同应用流量的突发数据包占比

Fig. 5　Proportion of bursty packets in different applications in experiment 2

5.3 检测精度实验

实验 3　随着两两独立的哈希函数数量的增加，RBFRadar 的检测准确性上升。在本文的实验中，本文采用 CAIDA 数据集[29]进行验证。考虑到准确性与资源的权衡，本实验采用通用配置[35]，哈希数量的取值范围为 1～4。图 6 给出了随着行数(即参数 d)的增加，RBFRadar 检测准确性的变化情况。当列数固定时，越多的行数意味着哈希碰撞越少，因此随着行数的增大，RBF 的检测精度和 F1 分数不断增加，同时召回率在整体稳定的前提下小幅下降。当 d 很小时，行数的增加对哈希冲突的缓解更为明显，因此随着行数的不断增加，F1 分数的增长速率也在放缓。

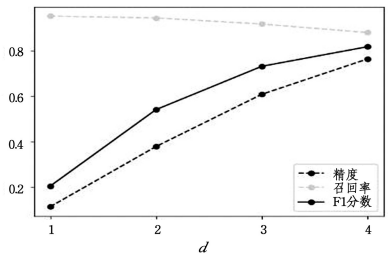

图 6　实验 3 中随行数增长的准确性变化情况

Fig. 6　Variation of accuracy as the number of row increases in experiment 3

实验 4　当列数 w 不断增加时，RBFRadar 的检测准确性先上升然后逐渐趋于稳定。当哈希函数的数量固定时，更多的列数允许更大的哈希空间，也意味着更大的内存分配。如图 7 所示，精度、召回率和 F1 分数在 w 超过 5 000 左右之后就逐渐收敛到稳定的范围。因此，在实际应用中可以在准确性和内存开销平衡之间找到一个收敛点。

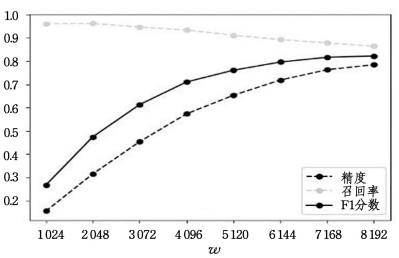

图 7　实验 4 中随列数增长的准确性变化情况

Fig. 7　Variation of accuracy as the number of colume increases in experiment 4

实验 5　在 RBF 检测任务中，RBFRadar 获得的准确性显著高于其他方案。本实验将 RBFRadar 与 3 类方案进行

比较，包括 heavy hitter 检测方案、基于队列的突发检测方案以及基于数据流模式的突发检测方案。其中，实验采用 MV-Sketch[30] 进行 heavy hitter 检测，BurstRadar[9] 和 BurstSketch[8] 则分别是基于队列和基于数据流模式的突发检测的代表性工作。如图 8 所示，无论是 heavy hitter 检测还是最新的突发检测工作，检测结果都无法很好地覆盖价值突发流。相比之下，RBFRadar 的 F1 分数是现有方案的 5.6～23.4 倍。

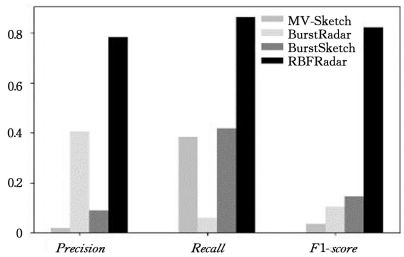

图 8 实验 5 中与现有方案相比 RBFRadar 准确性更高

Fig. 8 RBFRadar obtains higher accuracy than existing methods in RBF detection in experiment 5

5.4 带宽开销实验

实验 6 RBFRadar 只需要少量带宽用于上报。当数据平面向控制平面报告检测信息时，应避免带宽开销(Bandwidth Overhead，BO)造成各测量任务间的带宽争用。本实验在相同的流量数据中，统计不同的数据平面突发检测方案向控制器所报告的数据量。如图 9 所示，与基于逐次突发的检测方案 BurstRadar[9] 和 BurstScope[10] 相比，RBFRadar 分别减少了 98.84%和 84.62%的带宽开销。

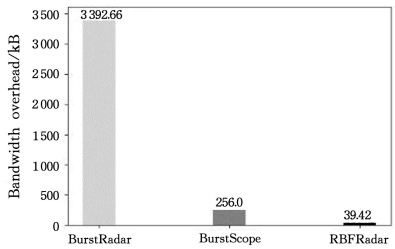

图 9 实验 6 中 RBFRadar 的带宽开销显著少于逐突发方案

Fig. 9 RBFRadar generates far less bandwidth overhead than per-burst-based schemes in experiment 6

6 相关工作

6.1 流量模式分析的研究

对流量模式(例如基于数据量的突发、基数突发[36]等)进行分析是常见的网络测量任务之一。Paul 等[37] 首次讨论在历史数据中有效检测突发事件的问题，将到达速率出现高加速度的事件认为是突发事件。相似地，Xie 等[38] 采用了基于加速度的突发定义，并只关注突发话题的检测。在最先进的工作中，Zhong 等[8] 提出了一个更完整的突发定义，其中突发由数据项到达速率的突增和突降组成。与之前的工作相比，该研究更关注实时的突发检测。

6.2 基于队列的微突发检测研究

微突发能够在生产网络中引入时延、抖动和丢包，检测微突发是进行微突发缓解和网络优化的前提条件。现有工作通过观察交换机中端口队列长度来确定微突发的发生，Joshi 等[9] 采用了快照技术，当端口队列长度超过一个指定阈值时，捕获所有涉及微突发事件的数据包，这些信息将被发送至监测控制器进行处理。为了降低该工作中的带宽开销，Gao 等[10] 提出了一个高效的微突发检测系统，在检测过程中生成更少的信使数据包。这些现有方案容易部署在数据平面上，然而，基于队列的方案只能提供整体流量粒度的见解，而不是分析每个流的突发特征。

结束语 本文分析了现有基于逐次突发的检测方案，针对其显著带宽开销和高用户负担方面的局限性，提出了价值突发流(RBF)的定义，该流量模式可作为流量突发性测量的有效指标。为了在可编程数据平面上检测 RBF，本文提出了一个基于 Sketch 的检测框架 RBFRadar，使用有限的内存开销实现高准确度的 RBF 检测。实验结果表明，在 RBF 检测准确性和节省带宽开销方面，RBFRadar 展现出了优异的性能。在未来的工作中，我们将进一步考虑在商用可编程交换机上解除硬件操作限制，进一步扩展 RBF 检测的部署场景。

参 考 文 献

[1] GHABASHNEH E, ZHAO Y, LUMEZANU C, et al. A microscopic view of bursts, buffer contention, and loss in data centers [C]// Proceedings of the 22nd ACM Internet Measurement Conference. 2022: 567-580.

[2] SHARAFZADEH E, ABDOUS S, GHORBANI S. Understanding the impact of host networking elements on traffic bursts [C]// 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23). 2023: 237-254.

[3] PAXSON V, FLOYD S. Wide area traffic: the failure of Poisson modeling[J]. IEEE/ACM Transactions on Networking, 1995, 3(3): 226-244.

[4] KIM G, LEE W. Absorbing microbursts without headroom for data center networks[J]. IEEE Communications Letters, 2019, 23(5): 806-809.

[5] FLACH T, DUKKIPATI N, TERZIS A, et al. Reducing web latency: the virtue of gentle aggression[C]// Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. 2013: 159-170.

[6] SHAN D, REN F, CHENG P, et al. Micro-burst in data centers: Observations, analysis, and mitigations[C]// 2018 IEEE 26th International Conference on Network Protocols (ICNP). IEEE, 2018: 88-98.

[7] CHEN X, FEIBISH S L, KORAL Y, et al. Catching the microburst culprits with snappy[C]// Proceedings of the Afternoon Workshop on Self-Driving Networks. 2018: 22-28.

[8] ZHONG Z, YAN S, LI Z, et al. BurstSketch: Finding bursts in data streams[C]// Proceedings of the 2021 International Conference on Management of Data. 2021: 2375-2383.

[9] JOSHI R, QU T, CHAN M C, et al. BurstRadar: Practical real-time microburst monitoring for datacenter networks[C]// Proceedings of the 9th Asia-Pacific Workshop on Systems. 2018: 1-8.

[10] GAO K, LI D, WANG S. Bandwidth-efficient Microburst Measurement in Large-scale Datacenter Networks[C]// Asia-Pacific Workshop on Networking. 2022.

[11] BOSSHART P,DALY D,GIBB G,et al. P4:Programming protocol-independent packet processors[J]. ACM SIGCOMM Computer Communication Review,2014,44(3):87-95.

[12] KAPOOR R,SNOEREN A C,VOELKER G M,et al. Bullet trains:A study of NIC burst behavior at microsecond timescales [C]// Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies. 2013:133-138.

[13] ZHANG Q,LIU V,ZENG H,et al. High-resolution measurement of data center microbursts[C]// Proceedings of the 2017 Internet Measurement Conference. 2017:78-85.

[14] SCHERRER S,VLIEGEN J,SATEESAN A,et al. ALBUS:a Probabilistic Monitoring Algorithm to Counter Burst-Flood Attacks[J]. arXiv:2306.14328,2023.

[15] SUN H,LUI J C S,YAU D K Y. Defending against low-rate TCP attacks:Dynamic detection and protection[C]// Proceedingsof the 12th IEEE International Conference on Network Protocols(ICNP 2004). IEEE,2004:196-205.

[16] ALCOZ A G,STROHMEIER M,LENDERS V,et al. Aggregate-based congestion control for pulse-wave DDoS defense [C]// Proceedings of the ACM SIGCOMM 2022 Conference. 2022:693-706.

[17] REZAEI H. Adaptive Microburst Control Techniques in Incast-Heavy Datacenter Networks[D]. Chicago:University of Illinois,2021.

[18] TANG D,ZHANG S,CHEN J,et al. The detection of low-rate DoS attacks using the SADBSCAN algorithm[J]. Information Sciences,2021,565:229-247.

[19] TAMMANA P,AGARWAL R,LEE M. Simplifying datacenter network debugging with {PathDump}[C]// 12th USENIX Symposium on Operating Systems Design and Implementation(OSDI 16). 2016:233-248.

[20] ARZANI B,CIRACI S,CHAMON L,et al. 007:Democratically finding the cause of packet drops[C]// 15th USENIX Symposium on Networked Systems Design and Implementation(NSDI 18). 2018:419-435.

[21] GKOUNIS D,KOTRONIS V,LIASKOS C,et al. On the interplay of link-flooding attacks and traffic engineering[J]. ACM SIGCOMM Computer Communication Review, 2016,46(2):5-11.

[22] MAI L,HONG C,COSTA P. Optimizing network performance in distributed machine learning[C]// 7th USENIX Workshop on Hot Topics in Cloud Computing,2015.

[23] ZAHARIA M,CHOWDHURY M,FRANKLIN M J,et al. Spark:Cluster computing with working sets[C]// 2nd USENIX Workshop on Hot Topics in Cloud Computing(HotCloud 10). 2010.

[24] REZAEI H,VAMANAN B. Superways:A datacenter topology for incast-heavy workloads[C]// Proceedings of the Web Conference 2021. 2021:317-328.

[25] YUE M,WU Z,WANG J. Detecting LDoS attack bursts based on queue distribution[J]. IET Information Security, 2019,13(3):285-292.

[26] SPITERI K,URGAONKAR R,SITARAMAN R K. BOLA:Near-optimal bitrate adaptation for online videos[J]. IEEE/ACM Transactions on Networking,2020,28(4):1698-1711.

[27] BLEULER S,LAUMANNS M,THIELE L,et al. PISA—a platform and programming language independent interface for search algorithms[C]// Evolutionary Multi-Criterion Optimization:Second International Conference,EMO 2003,Faro,Portugal. Berlin Heidelberg:Springer,2003:494-508.

[28] ZHANG Y,LIU Z,WANG R,et al. CocoSketch:High-performance Sketch-based measurement over arbitrary partial key query[C]// Proceedings of the 2021 ACM SIGCOMM 2021 Conference. 2021:207-222.

[29] CORMODE G,MUTHUKRISHNAN S. An Improved Data Stream Summary:The Count-min Sketch and its Applications [J]. Journal of Algorithms,2005,55(1):58-75.

[30] TANG L,HUANG Q,LEE P P C. Mv-Sketch:A fast and compact invertible Sketch for heavy flow detection in network data streams[C]// IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE,2019:2026-2034.

[31] HUANG Q,LEE P P C. A hybrid local and distributed Sketching design for accurate and scalable heavy key detection in network data streams[J]. Computer Networks,2015,91:298-315.

[32] 2020. Barefoot Tofino[OL]. (2020). https://barefootnetworks.com/products/brief-tofino/.

[33] BOYER R S,MOORE J S. MJRTY—A Fast Majority Vote Algorithm[M]//Automated reasoning:essays in honor of Woody Bledsoe. Dordrecht:Springer Netherlands,1991:105-117.

[34] BMv2 Software Switch[OL]. http://bmv2.org/.

[35] LV J W. Microburst detection based on programmable switch [D]. Hangzhou:Zhejiang University,2023.

[36] WANG H,MELISSOURGOS D,MA C,et al. Real-time Spread Burst Detection in Data Streaming[J]. Proceedings of the ACM on Measurement and Analysis of Computing Systems,2023,7(2):1-31.

[37] PAUL D,PENG Y,LI F. Bursty event detection throughout histories[C]// 2019 IEEE 35th International Conference on Data Engineering(ICDE). IEEE,2019:1370-1381.

[38] XIE W,ZHU F,JIANG J,et al. TopicSketch:Real-time bursty topic detection from twitter[J]. IEEE Transactions on Knowledge and Data Engineering,2016,28(8):2216-2229.

WU Yanni, born in 2001, postgraduate. Her main research interests include network measurement, computer systems and programmable networks.

ZHANG Dong, born in 1981, Ph.D, professor, Ph.D supervisor, is a member of CCF(No. 29654M). His main research interests include software-defined networking, network virtualization, and the Internet QoS.

(责任编辑:喻藜)