doi:10.11959/j.issn.1000-436x.2017203

基于 sketch 的软件定义测量数据平面硬件模型

戴冕[1,2]，程光[1,2]

（1. 东南大学计算机科学与工程学院，江苏 南京 211189；
2. 东南大学教育部计算机网络和信息集成重点实验室，江苏 南京 211189）

摘 要：提出一种基于 sketch 数据结构的软件定义测量数据平面硬件模型，并在以现场可编程逻辑门阵列（FPGA）为核心的可编程网络设备 NetMagic 上进行了实现。利用部署在硬件 FPGA 高速 SRAM 中的通用 sketch 数据结构高效地采集数据平面流量数据，控制平面收集并缓存统计数据，提供给上层的测量应用使用。使用 count-min sketch 和 2-universal 散列函数实现了在高速流量下实时的分组处理和流量统计；使用 Bloom filter 在控制平面恢复流量的原始 5 元组信息，解决了 sketch 数据结构的不可逆问题。使用 CERNET 骨干网流量数据对原型系统进行的评估结果表明，该原型系统使用极其有限的硬件资源实现了对较大规模网络流量的实时测量，同时具备较好的测量精度。

关键词：软件定义测量；现场可编程逻辑门阵列；全域散列

中图分类号：TP393 **文献标识码：**A

Sketch-based data plane hardware model for software-defined measurement

DAI Mian[1,2], CHENG Guang[1,2]

(1. School of Computer Science and Engineering, Southeast University, Nanjing 211189, China;
2. Key Laboratory of Computer Network and Information Integration of Ministry of Education, Southeast University, Nanjing 211189, China)

Abstract: A sketch-based data plane hardware model for software-defined measurement was introduced, and it was implemented in the programmable network device NetMagic. A generic sketch model for collecting flow-level data using high-speed memories on the FPGA was proposed, the control plane collected and cached the data for further process. Count-min sketch and 2-universal hash functions in the SRAM of FPGA for real-time traffic counting of high-speed traffic were implemented; Bloom filter was used to rebuild the original 5-tuple data which solved the irreversibility of sketch. The CERNET backbone trace to evaluate the prototype system was used, the result shows that it has the ability to use the limited hardware resource to measure a large amount of network traffic data with a proper measurement accuracy at the same time.

Key words: software-defined measurement, FPGA, universal hash

1 引言

实时、准确的网络流量测量是网络管理的基础。在传统高速网络测量中，为了对海量网络流量进行实时的处理，需要对数据量进行大规模的压缩，同时保留流量数据的特征信息。针对高速网络流量测量的问题，目前主要有抽样技术和数据流技术 2 种解决方案。

抽样技术通过选择具有代表性的网络流量数据分组子集，以该子集推断总体流量的特征信息，可以直接降低流量数据的处理时间和设备内存的占用空间。NetFlow[1,2]作为抽样技术的标准工具，

收稿日期：2016-12-10；**修回日期：**2017-06-15
通信作者：程光，gcheng@njnet.edu.cn
基金项目：国家高技术研究发展计划(“863”计划)基金资助项目(No.2015AA015603);赛尔网络下一代互联网技术创新基金资助项目(No.NGII20150108)

Foundation Items: The National High-Tech R&D Program of China (863 Program) (No.2015AA015603), CERNET Innovation Project (No.NGII20150108)

在一个测量周期内需要为每条流维护一个独立的计数器，较高的抽样速率需要消耗大量的性能资源，而较低的抽样速率会导致丢失部分原始流量的统计特征，使测量结果产生较大的误差。

数据流技术不对原始网络流量进行筛选，它以 sketch 数据结构为基础，通过散列算法将庞大的信息压缩到较小的存储空间内，并且保证精确的流量测度估计。sketch 数据结构可以在高速网络中实时地汇总并存储流量特征信息，只占用较小的内存空间，通过查询计算可以获得针对某一种测度的精确估计。针对不同流量测度选择合适的 sketch 数据结构能够提高数据流算法的执行效率和估计精度，降低计算和存储开销。sketch 数据结构的更新效率极高，可以保证在高速网络环境中进行线速下的测量，并且具备在理论上可证明的估计精度与内存的平衡特性，即调整 sketch 数据结构的大小就可以使测量值的估计精度在要求的范围内。sketch 数据结构还具有线性计算的性质，即可以将若干个 sketch 线性串联，以实现对复杂度的测量。因此，基于 sketch 的数据流算法被广泛地应用于高速网络的流量测量应用中，包括大流检测（heavy hitter detection）、超点检测（super spreader detection）、异常检测（change detection）、流长分布估计（flow distribution extimation）、熵估计（entropy estimation）、带宽利用率估计、网络容量估计等[3~6]。

然而，sketch 数据结构本身也存在缺点。在基于 sketch 数据结构的数据流算法中，sketch 的更新操作包含大量的“读取—修改—写入”操作，对于 IO 性能要求较高，因此，sketch 数据结构一般需要在硬件的 SRAM 中实现。但是 sketch 数据结构与特定测量任务的耦合性很大，不同的测量任务需要若干个不同的 sketch 相组合来实现，同时，不同的 sketch 又具有不同的数据结构，用于更新 sketch 的流字段各不相同，占用的内存空间大小也不一致。这样导致在硬件上实现的 sketch 数据结构的灵活程度非常有限，很难被不同的流量测量应用所复用。此外，由于基于 sketch 数据结构的数据流算法具有单向性和一次性的特点，即算法只对网络流执行一次计算，且散列过程不可逆，因此，sketch 无法保留原始流量的分组信息。目前很多工作[7~10]在研究如何设计 sketch 数据结构，使其计算过程可逆，从而获取原始流量信息，但是需要以牺牲内存空间和 sketch 更新速率为代价。

软件定义网络通过将控制平面从网络设备的数据平面中剥离，部署为逻辑上集中的控制器，大大增强了网络管理的简单性和灵活性。数据平面和控制平面相分离同时，意味着网络流量测量的数据平面也和控制平面相分离，这大大增加了网络流量测量的灵活程度。同时，软件定义网络中基于流的转发规则也使对网络流量进行更细粒度的测量成为可能。在传统网络测量中，往往需要通过交换机端口镜像或光纤分路的方式将待测流量转发至专门的测量服务器，而在软件定义网络中，可以将测量应用直接部署在控制器中，直接利用控制器中保存的全局网络信息辅助测量任务的计算。

本文在文献[11]的基础上，提出了一种软件定义测量数据平面硬件模型，并基于 SDN 化的 FPGA 设备 NetMagic[12]进行了原型实现和评估。NetMagic 作为一个可编程的独立网络设备，其核心组件为 FPGA，拥有 4 MB 的片上 SRAM 和 4×1 000 bit/s 的以太网端口。该设备通过硬件语言编程，具备基本的 SDN 交换机功能。它可以对任意端口输入的任意分组的全部字段进行解析，并且根据事先下发的匹配规则转发、丢弃或存储。NetMagic 支持自定义的 NMAC 协议对其进行访问控制，在控制器端部署 OpenFlow 协议和 NMAC 协议的适配层之后，可以通过 OpenFlow 控制器直接进行访问。本文在 NetMagic 的 SRAM 上部署 count-min sketch 作为网络流量测量的数据结构，存放流的分组统计和流长统计信息，因为其具有实现简单、更新速率快的优点，所以不需要对输入分组进行抽样就可以在线速下实时地处理每一个输入分组。count-min sketch 包含 4 张散列表，由于全域散列函数具有较高的性能，本文使用 4 个 2-universal 散列函数对输入分组的头部进行散列。这里要说明的是，本文将分组头部散列后的值直接作为 SRAM 的地址索引来更新 count-min sketch，一方面减少了读写 SRAM 时的寻址时间，另一方面也减少了存储分组头部所需要的内存空间。本文没有在数据平面通过复杂的设计来恢复原始流的信息，而是利用在控制器中部署和 NetMagic 中同样的散列函数和同样的 count-min sketch 数据结构来达到相同的效果。当每个待测分组到达 NetMagic 时，其分组头部 5 元组被解析，然后通过一个 Bloom filter 来判断是否为新流，新流的 5 元组信息会被封装在 NMAC 协议分组中发送

给控制器进行相应的处理和存储。每个测量周期之后，结合 NetMagic 和控制器中 count-min sketch 的信息可以重建出原始流的 5 元组信息。本文使用真实的 CERNET 骨干网流量数据对模型的原型实现进行了评估，实验结果表明在 FPGA 有限的板载资源上，该模型可以在大规模的数据量下达到较好的精度和效率。

2　相关研究

传统高速网络流量测量中，很多流量检测算法将动态抽样算法、散列算法、sketch 数据结构、数理统计相结合来优化测量结果。软件定义网络的出现使在测量控制平面动态的配置测量平面参数、实例化测量任务、调整测量资源成为可能。因此，有研究开始将注意力集中在如何把软件定义测量架构与传统的测量算法和数据结构相结合，以达到更高的通用性、测量效率和测量精度。

OpenSketch[4]使用了基于散列算法的测量数据平面和 sketch 数据结构来统计网络流量数据。其研究目标是在测量控制平面和测量数据平面的算法相分离的架构下提供一种通用且高效的网络流量测量手段。OpenSketch 保证了数据平面功能简单易于实现、可以灵活配置，在控制器端实现了可编程的支持不同测量任务的算法。OpenSketch 的测量支持加上 OpenFlow 的控制支持，可以构成一个完全的 SDN 测量控制闭环架构。OpenSketch 重新设计了交换机的测量 API，使其具备通用性和高效性。与基于流的测量不同，OpenSketch 允许更加自定义和高效的数据收集方案，来决定测量哪些流（使用散列和 TCAM 通配符规则分类）、测量哪些数据（不仅是比特/分组计数，如平均流量大小）、如何存储测量数据（使用更加简洁的数据结构而不是简单的每条流的 5 元组计数器）。OpenSketch 设计了一个三级数据平面管道，易于配置和操作，支持在高链路速率、低内存开销下的多种测量任务。OpenSketch 在测量控制平面部署了一个测量库，使用这个测量库可以自动地为不同的 sketch 配置数据平面管道并且分配相应的交换机资源使任务达到最大的精度。在一个测量周期后，FPGA 上 SRAM 中存储的 sketch 计数器被发送到控制器，由控制器上部署的通用测量库和多种测量应用进行分析和处理。在 OpenSketch 中，FPGA 上的 SRAM 被划分为若干个逻辑区域，每个区域分别被实现为不同结构和大小的计数器，以满足不同 sketch 数据结构的需要，这种实现方法需要复杂的 SRAM 寻址机制来支持。

UnivMon[5,6]在之前研究的基础上，提出了一个通用的网络流量监测架构。通过在网络设备上部署简单且通用的测量原语，可以实现对多种测量任务的支持并且可以达到较高的测量精度。所有测量任务的精度至少都不低于使用与该测量任务相适应的 sketch 数据结构所达到的测量精度。UnivMon 提出了一种 universal sketch 的数据结构，并且使用 P4[13]语言进行了实现。它同样利用了测量控制平面与测量数据平面的分离，在控制平面下发 sketching manifest 来调度数据平面的 sketch 资源，支持一定程度的分布式软件定义测量。数据平面则是将抽样方法和数据流方法相结合，通过并行流水线的方式处理流量数据。原始流量经过多级采样，每级采样后的流量数据并行的通过 count sketch 数据结构，最终每级 sketch 的统计值被发送到控制平面，由部署在控制平面的测量算法递归计算出最终的 top-k 流量。

OpenSketch 和 UnivMon 提出的在硬件上实现通用 sketch 的解决方案，实际上都消耗了大量的硬件资源。OpenSketch 是在硬件上部署了大部分常用的 sketch，在执行测量任务时由控制平面根据具体要计算的测度来调度数据平面所需的 sketch，通过将它们组合成流水线来进行流量统计。这种方法虽然具备了通用性，但是以牺牲硬件资源为代价，未参与计算的 sketch 数据结构浪费了其占有的硬件 SRAM 空间。UnivMon 虽然只采用了一种 sketch 数据结构，但是实际测量的时候需要对 N 级抽样流量数据并行的进行计算，实际上需要将 sketch 数据结构复制 N 份，同样要消耗大量的硬件资源。另外，UnivMon 虽然通过 P4 语言实现了原型系统，但是实验验证是基于 P4 提供的软件仿真网络环境，实际上并没有在硬件上实现。sketch 数据结构和上层测量算法都是通过 CPU 进行计算，性能结果与硬件存在较大的差距。

本文的研究工作从基于硬件 FPGA 实现的角度出发，综合考虑测量控制平面、测量数据平面的复杂程度和测量效率、精度，使两者在有限的硬件资源的前提下达到一种平衡。在硬件实现中，sketch 的更新速度是一个性能瓶颈。为了在硬件 FPGA 上实现线速的 sketch 数据结构更新，本文采用了 count-min sketch 数据结构作为数据平面的计数器。

count-min sketch[14]是一个次线性的空间数据结构，可以用一个二维数组表示，也可以理解为 n 个长度为 m 的一维数组。n 个相互独立的散列函数被均匀、随机地选择，用于更新 sketch。当更新项(k,v)到达时，键值 k 分别通过 n 个散列函数进行计算，计算结果为 k_1、k_2、…、k_n，然后数组 1 的第 k_1 项增加 v 个计数，数组 2 的第 k_2 项增加 v 个计数，以此类推，如图 1 所示。

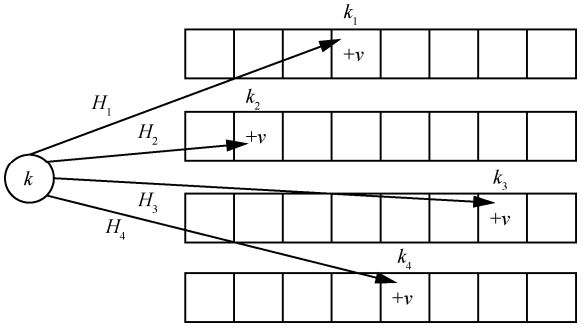

图 1 count-min sketch 的数据结构和更新过程

count-min sketch 一共需要 $n\times m$ 个存储空间，容易实现。在查询 count-min sketch 时，以 n 个数组中计数最小的值作为查询值。count-min sketch 数据结构虽然无法完全避免散列冲突导致的测量误差，但是可以保证精度在可接受的范围内。使用链式方法或线性探测方法来处理散列表中的散列冲突可以完全消除散列冲突带来的误差，但是链式方法需要占用额外的内存空间，而线性探测方法会引入额外的插入和查询操作时间，在最坏的情况下，链表的长度和线性探查的次数会变得非常高，导致硬件 FIFO 队列阻塞。此外，count-min sketch 更新所需的内存 IO 次数为一个常量 n，即 count-min sketch 所包含的独立散列函数的个数。

Bloom filter[15]是用于查找一个元素是否在集合中的高效数据结构。它可以用一个 m 位的数组表示一个集合 $S=\{x_1,x_2,\cdots,x_n\}$，初始化值为全 0。Bloom filter 同样使用 k 个独立的散列函数，每个散列函数的取值都为$\{1,2,\cdots,m\}$。对于任意 $x\in S$，散列函数 h_i 映射到数组的位置 $h_i(x)$就会被置 1（$1\leqslant i\leqslant k$），如果一个位置被多次置 1，那么只有一次会起作用。如果要查询一个元素 y 是否属于 S，那么用 k 个散列函数对 y 进行 k 次散列，如果数组中所有的 $h_i(y)$ 的位置都是 1，则判断 $y\in S$，否则 $y\notin S$。Bloom filter 的插入和查找过程如图 2 所示。

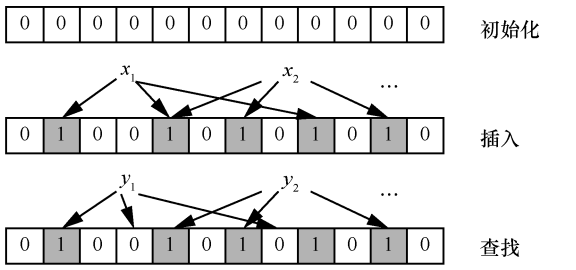

图 2 Bloom filter 的插入和查找过程

Carter[16]所提出的全域散列函数是指一族具有特殊的随机性质的散列函数。如果在这一族的散列函数中均匀、随机地选择散列函数，那么该函数将 2 个键值计算为同一散列值的概率是有边界的。因此，选择合适的全域散列函数可以大大减少散列冲突的概率。在 2-universal 的散列函数族中，冲突概率是两两独立的，即对于所有的 $x\neq y\in U$ 且 $s,t\in B, P_{h\in H}[h(x)=s,h(y)=t]=\dfrac{1}{R^2}$。$k$-universal 散列函数族可以由式(1)推导，即 5-universal 散列函数族中 k=5，其中，$x<p$ 且 p 是一个素数，通常情况下在梅森素数（如 $2^{31}-1$ 或 $2^{61}-1$ 等）中选择，这样可以避免长时间的除法计算。参数 a_i 在 0~p 之间随机选择。

$$h(x)=\sum_{i=0}^{k-1}((a_i x^i)\bmod p)\bmod m \tag{1}$$

$$P_{h\in H}[h(x_i)=v_i]=\frac{1}{m^k}, v_i\in[m], i\in(0,1,\cdots,k-1)$$

3 原型系统设计

3.1 NetMagic 简介

本文所描述的数据平面测量模型原型系统完全基于可编程网络设备 NetMagic 实现，因此，在本节中首先对基于该设备的前期工作作一个简要的介绍。如前所述，NetMagic 是一个基于 FPGA 实现的独立网络设备，由于 FPGA 可以通过 Verilog 语言灵活地进行编程，因此可以任意修改 NetMagic 中的分组解析规则、转发规则、SRAM 读写规则，使其成为一个二层交换机、三层交换机、SDN 交换机或完成任何特定任务的网络设备。在本文中，将 NetMagic 实现为标准的 OpenFlow 交换机，然后在板载 SRAM 上部署 count-min sketch 和 Bloom filter。同时还通过 Java 语言实现了软件控制器，可以远程配置 NetMagic 并且收集 SRAM 中的数据。NetMagic 上部署的基础硬件代码，从逻辑上可以分为 3 个模块，具体如图 3 所示。

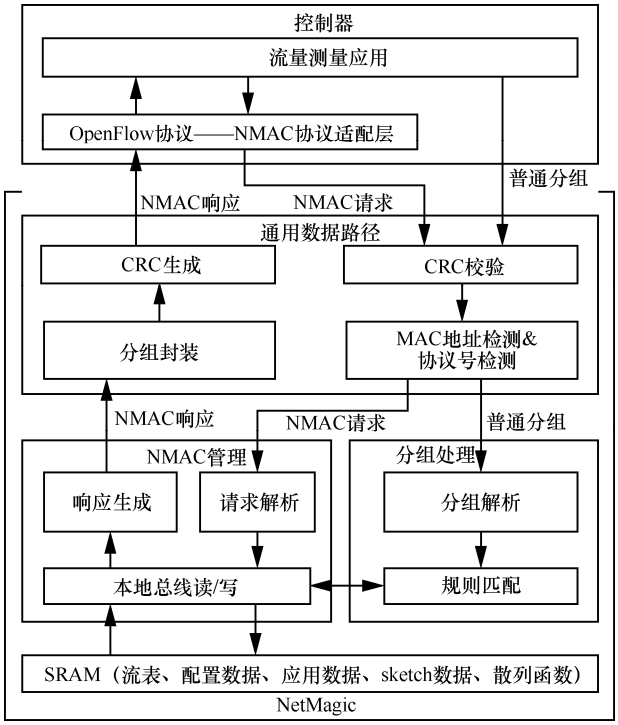

图 3　NetMagic 逻辑结构和工作流程

通用数据路径模块负责网络设备处理以太网分组时所有通用不变的操作，包括发送/接收分组、帧校验、维护和调度进出队列等。NMAC 管理模块处理控制器发送至 NetMagic 的 NMAC 请求分组，解析并执行其中的命令，读/写 SRAM 并且返回相应的响应分组。分组处理模块负责对非 NMAC 分组的普通分组进行解析、规则匹配和转发。NMAC 协议是对 NetMagic 进行访问和控制的自定义协议，协议号指定为 253，协议命令封装在普通 IP 分组的数据域中传输。NMAC 协议具有 7 种类型的命令，包括控制器与 NetMagic 设备的连接与释放、读/写 NetMagic 的 SRAM 的请求与响应、NetMagic 对控制器的异步请求。

综上所述，NetMagic 具备 SDN 交换机的全部基础功能。首先通用路径模块实现了以太网中二层分组交换的所有通用操作；其次通过 NMAC 管理模块读写 SRAM 可以在 SRAM 上轻松的实现流表规则的部署、修改、删除和查询；最后，NMAC 协议使 NetMagic 具备了与控制器通信的能力，虽然 NMAC 协议是自定义的通信协议，但是通过在 OpenFlow 控制器中部署转换 OpenFlow 协议分组和 NMAC 协议分组的软件适配代码，可以使 NetMagic 直接与标准的 OpenFlow 控制器进行通信。

3.2　总体架构

本文提出的基于 NetMagic 实现的软件定义测量系统的架构如图 4 所示。当网络流量从 NetMagic 的以太网端口进入设备后，对于普通分组，在通过通用数据路径模块的校验后，会被发送到分组处理模块对分组头部进行解析，得到 5 元组字段（源 IP、目的 IP、源端口、目的端口、协议号）和分组长度

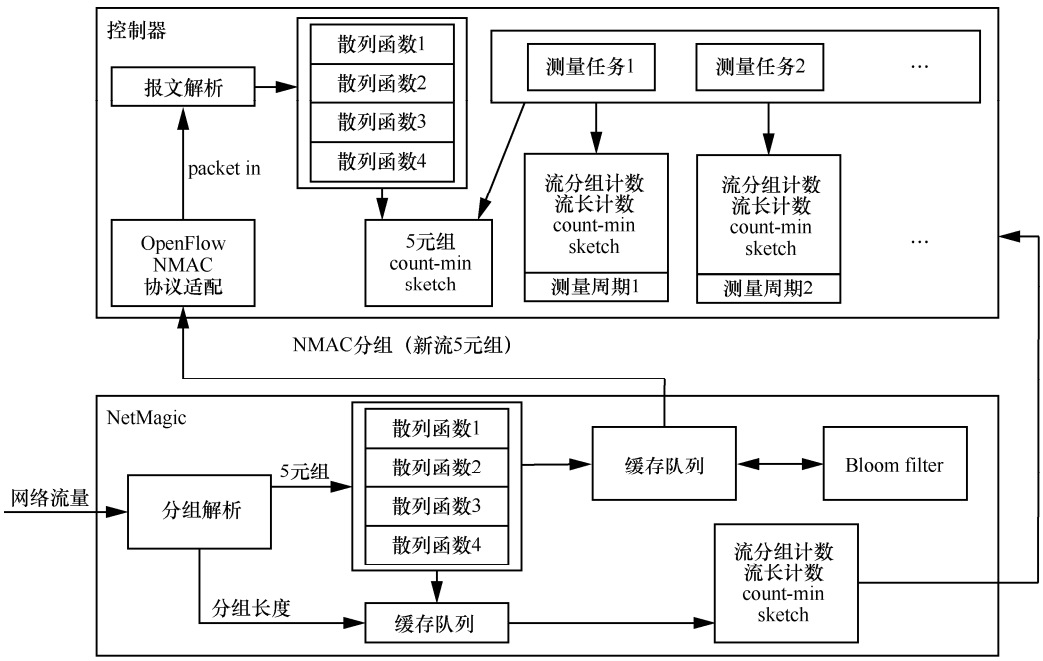

图 4　系统总体架构

字段。其中，5 元组字段被合并为一个 104 bit 的值进入散列模块进行计算。散列模块中部署了 4 个 2-universal 散列函数，一般情况下，4~8 个独立的散列函数可以满足大部分基于 sketch 的测量算法的需求。NetMagic 的 SRAM 上划分了 4 个大小相等的区域，作为 count-min sketch 的计数器结构，4 个区域通过地址索引前缀 00、01、10、11 区分，因此，每个区域的相对地址索引都是相同的。4 个散列函数输出值的范围与每个区域的相对地址索引范围完全相同，因此，更新 count-min sketch 时，直接分别使用 4 个散列函数的输出值作为地址索引，更新对应位置的计数器值。count-min sketch 中的每个计数器存放 2 个计数值，一个是流分组计数，即每更新一个分组，计数值加 1，一个是流长计数，每更新一个分组，计数值加该分组的长度。

为了解决 sketch 算法无法恢复原分组头部 5 元组信息的问题，本文设计了一个在控制器端恢复原始分组的方案。在 NetMagic 的 SRAM 中，本文还部署了一个 Bloom filter 用于快速检测分组 5 元组是否属于一条新流。如果是新流，那么 5 元组信息会被封装到 NMAC 异步请求分组中发送给控制器，如果控制器使用的是标准 OpenFlow 协议控制器，那么协议适配层会将 NMAC 消息转换为 packet-in 消息发送给控制器。控制器部署着软件代码实现的与 NetMagic 中完全相同的 4 个独立的 2-universal 散列函数，同时维护着与 NetMagic 中完全相同的 count-min sketch 计数器。不同的是，packet-in 中的分组 5 元组被散列后，更新计数器值的不是流分组个数和流长，而是将该分组的 5 元组本身存放在散列表中，同时使用链式方法解决散列冲突，散列表的每个表项维护一张链表，保存 5 元组的原始信息。这样，当测量任务从每个测量周期的某个计数表中筛选出潜在的大流或异常流量时，就可以在控制器端的相同表的相同地址索引中获取一个链表的分组原始 5 元组信息。测量任务用这些 5 元组信息再到 count-min sketch 的 4 张计数表中去查询，就可以剔除错误的流，从而获取正确的原始流信息。

NetMagic 在一个测量周期内连续的进行测量，每个测量周期后，控制器会通过 NMAC 消息读取 NetMagic 中 count-min sketch 中的所有数据，保存在控制器端，同时初始化 NetMagic 中的 count-min sketch 和 Bloom filter 计数器。

3.3 测量数据平面

NetMagic 中数据总线的位宽为 128 bit，分组头部 5 元组的长度为 104 bit，因此，2-universal 散列算法的输入模块在一个硬件周期内就可以完成全部输入数据的采集，时钟频率为 125 MHz。2-universal 散列函数在 FPGA 上的硬件逻辑如图 5 所示。

NetMagic 总共具有 4 MB 的 SRAM，地址索引空间为 20 bit，总大小为 2^{20} bit，每个地址索引可以存放 32 bit 的数据。在本文的设计中，SRAM 需要存储 count-min sketch，包含 4 个一维数组；Bloom filter 包含 1 个一维数组；4 个 2-universal 散列函数；设备配置数据；分组匹配规则流表。因此本文将 SRAM 规划如下。首先将 20 bit 的地址索引划分为前缀为 0 和前缀为 1 的两部分，前缀为 0 的地址空间的 011**********部分用于存放 Bloom filter，其他部分存放散列函数、设备配置数据和分组匹配规则流表；前缀为 1 的部分用于存放 count-min sketch，划分为 100**********、101**********、110**********、111**********4 部分。这样，Bloom filter 和 count-min sketch 中 5 个计数数组的相对地址空间的地址索引完全一致，使 5 元组通过散列函数后不需要任何额外的操作就可以获得要更新的地址索引。SRAM 的地址划分如图 6 所示。

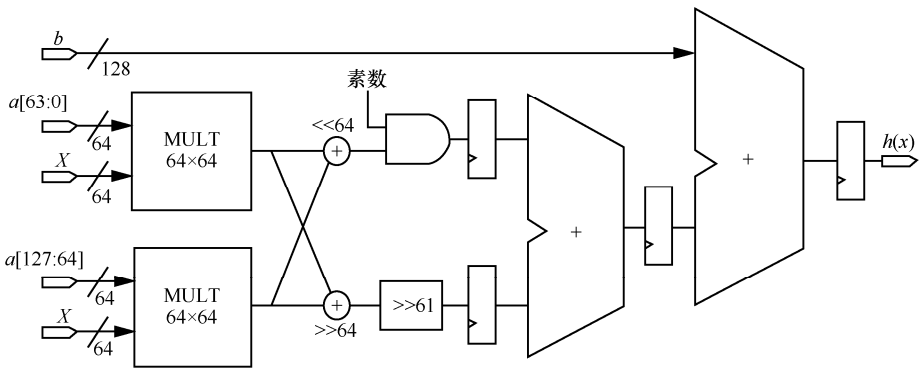

图 5 2-universal 散列函数在 FPGA 上的硬件逻辑

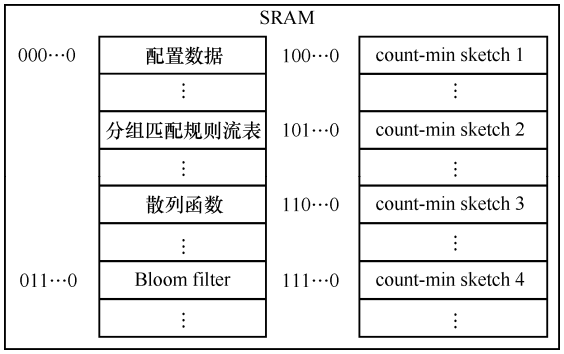

图 6　SRAM 地址划分

3.4　测量控制平面

为了保证原型系统的通用性，本文采用 Floodlight 作为 NetMagic 的控制器。由于 NetMagic 通过 NMAC 协议通信，因此，需要在控制器和 NetMagic 之间增加 OpenFlow-NMAC 协议适配层。控制平面的测量应用作为扩展模块集成在 Floodlight 中，作为控制器之上的应用程序。即使剥离测量应用的模块，NetMagic 也可以正常地与控制器交互，像普通的 OpenFlow 交换机一样工作，完成分组的查表转发。

每个测量周期结束后，NetMagic 上 SRAM 中 count-min sketch 数据会被发送到控制器。由于 NetMagic 支持基于 SRAM 地址的连续读取，控制器获取 SRAM 中 sketch 数据的时间小于 100 ms，远远小于测量间隔，因此，不会影响到系统的正常运行。控制平面根据需求缓存若干个测量周期的数据。控制器上部署的测量模块通过查询 count-min sketch 找出潜在的满足测量要求的流量，然后根据控制器端的记录流原始 5 元组信息的 count-min sketch 进行筛选，最终得到符合要求的流的 5 元组信息。

本文在控制器中部署了异常检测算法来对原型系统进行验证。异常检测是指识别出分组数量或分组大小在数个测量周期内变化超过给定阈值的流。以检测分组大小发生剧变的流为例，假定 $P_t = p_1, p_2, \cdots$ 是一个测量周期内进入 FPGA 的待测分组，对于每个分组 p_i，其用于更新 count-min sketch 的键为 5 元组，值为分组大小。假设 $C(t)$ 为第 t 个测量周期结束后 count-min sketch 中的值，$C_f(t)$ 是使用滑动平均（MA）算法根据测量周期 t 之前 W 个测量周期的结果计算出的第 t 个周期的预测值。其中，W 是滑动窗口的大小，其取值影响到需要缓存的历史测量周期数据的个数，可以根据每个测量周期的流量规模进行调整。那么本文可以得出第 t 个周期的真实流量统计数据 $C(t)$ 和预测统计数据 $C_f(t)$ 的差值 $C_e(t) = C(t) - C_f(t)$，根据给定的阈值 T，通过比较 $C(t)$ 和 $C_e(t)$ 即可找出符合变化条件的流量。

4　实验结果和分析

实验验证环境如图 7 所示。NetMagic 基于 Arria II EP2AGX45 系列 FPGA 开发，使用 Verilog 语言进行编程，板载 36 100 个逻辑单元，4 Mbit SRAM，硬件周期为 8 ns。控制器使用 Floodlight，部署在一台 Dell PowerEdge R720 服务器上，服务器的硬件配置为 INTEL Xeon E5-2620 v3×2 CPU、32 GB RAM，操作系统为 Ubuntu 16.04。

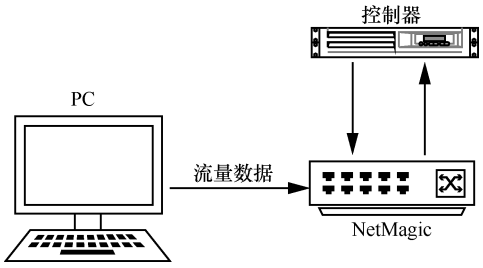

图 7　实验验证环境

NetMagic 中 SRAM 的配置如前所述，count-min sketch 包含 4 张计数表，每个具有 128 bit k 个地址索引，每个地址索引存放 32 bit 数据，其中，前 11 bit 用于分组计数，后 21 bit 用于流长计数；Bloom filter 包含 1 个计数表，具有 128 k 个地址索引。具体参数如表 1 所示。

表 1　　sketch 数据结构参数配置

sketch	散列函数	地址空间
count-min sketch	4	$128k \times 4$
Bloom filter	4	$128k$

测量数据源使用 IPTAS[17]提供的 CERNET 骨干网流量数据。450 s 的统计数据包含 7 485 823 个分组，其中，不同的 5 元组个数为 352 727 个。根据 FPGA 的硬件测量资源和测试数据的 5 元组规模，本文将测量周期设置为 10 s，即每个测量周期发送 10 s 的测试数据。测试数据集的统计信息如表 2 所示。

本文首先验证 2-universal 散列函数在 NetMagic 和控制器中的计算效率，结果如图 8 所示。

表 2　　测试数据集的统计信息

数据统计量	Trace(450 s)	Trace(10 s)
分组数量	7 485 823	226 843
源 IP 数量	68 998	2 204
目的 IP 数量	72 620	2 445
(SIP&SPort,DIP&DPort) 数量	352 727	22 176

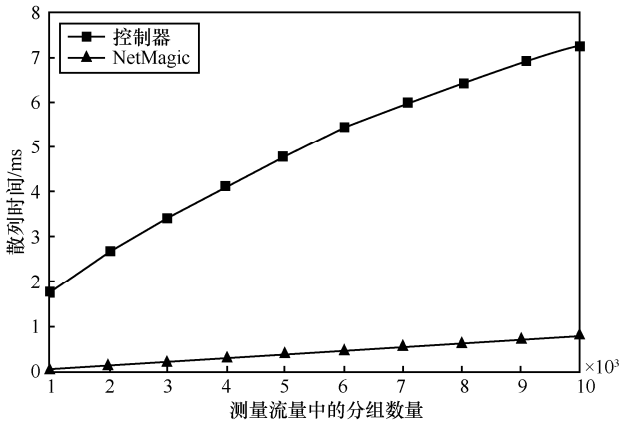

图 8　2-universal 散列函数在 NetMagic 和控制器上的散列时间

如 3.3 节所述，104 bit 的 5 元组数据小于 NetMagic 的硬件数据总线长度，因此，全部比特可以在一个硬件时钟周期内完成采集。由于 FPGA 中的门电路可以进行硬件级别的流水线运算，所以只需要数个硬件周期就可以完成计算，计算效率高于控制器上通过 Java 语言实现的散列函数。本文可以看到软件代码在启动执行时，还存在一个潜在的启动延迟，这一点在执行的散列计算数量较少时尤为明显，该延迟在毫秒级别，所以并不会对系统运行造成影响。

本文在原型系统的控制平面部署了异常检测算法，下面对该算法的精确度进行验证。异常检测算法的原理已经在第 3 节进行了说明，本实验中滑动窗口 W 设置为 3 个测量周期，因此，控制器中至少需要缓存 4 个测量周期的统计数据。

如前所述，本文通过在 FPGA 中部署 Bloom filter 来识别新流，将 5 元组信息保存在控制器中。由于 Bloom filter 存在不同流的 5 元组散列成相同值的情况，会带来额外的误报率，因此，本文需要尽量降低 Bloom filter 带来的测量误差。对于大小为 m 的 Bloom filter，如果在其中存储 n 个数据，那么误报率 $FPR=\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^{k}\approx\left(1-\mathrm{e}^{\frac{-kn}{m}}\right)^{k}$，其中，$k$ 为使用的散列函数的数量。理论上对于固定大小的 m，k 的值取 $\left(\frac{m}{n}\right)\ln 2$ 可以最小化误报率。在本实验中，散列函数个数 k 固定为 4，根据 10 s 测量周期的大致数据量可以计算出最优的 Bloom filter 大小 m 为 128 kbit。

熵是代表一组数据的随机性的度量，熵为 0 代表所有数据取值都相同，熵为最大值代表每个数据取值都不同，因此，熵估计算法被广泛地用于流量异常检测任务中。本文使用 count-min sketch 中的流长信息和 Bloom filter 区分出的流 5 元组信息来计算流量熵值，与真实的流量熵值比较，测度用相对误差来表示，实验结果如图 9 所示。在实验中本文使用了不同大小的 Bloom filter，可以看出在采用合适大小的 Bloom filter 的情况下，本文的算法可以有效地将异常检测的误差控制在可接受的范围内。随着 Bloom filter 大小的增加，熵估计算法的精度也大幅增加，当 Bloom filter 的大小超过 256 kbit 时，相对误差小于 1%。

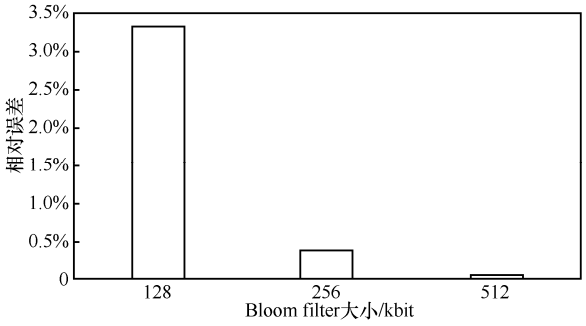

图 9　熵估计

本文选取了多个阈值 T 进行异常检测实验，测量精度采用误报率(*FPR*)和漏报率(*FNR*)来表示，结果如图 10 所示。从结果可以看出由于采用了大小

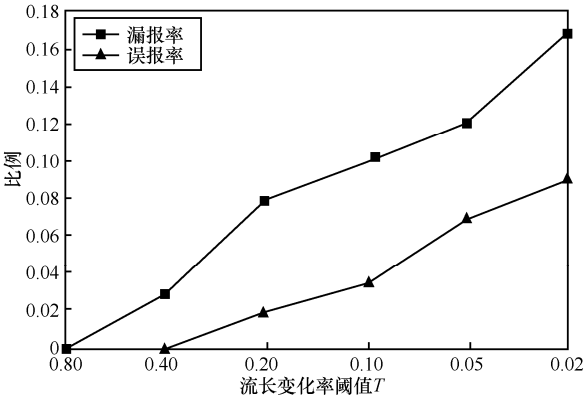

图 10　流量异常检测算法的误报率和漏报率

合适的测量周期，异常流量检测的误报率较低。漏报率稍高于误报率，这是由于待测流量中 5 元组散列存在一定的冲突，隐藏了部分异常流量。在较大的阈值 T 下，两者都保持了较低的水平。

5　结束语

本文提出了一种基于 Sketch 数据结构的软件定义测量数据平面模型，并在可编程设备 NetMagic 上实现了原型系统。该数据平面模型部署在 FPGA 的板载 SRAM 中，实现了在线速下实时地统计流量的分组计数和流长计数。控制平面负责收集和缓存统计信息，提供给上层测量应用使用，并且能够恢复被测流量的原始 5 元组信息。本文使用真实的骨干网 CERNET 流量数据和异常检测算法对该数据平面模型的测量效率和精度进行了评估，结果表明，该原型系统在对较大规模网络流量进行异常检测的测量任务中，使用非常有限的硬件资源达到了较好的测量精度。在下一步的工作中，本文将对分组处理流程、硬件资源分配、测量任务调度等问题进行深入研究，然后将该数据平面移植到性能更强的 FPGA 上进行验证和优化。

参考文献：

[1] HOFSTEDE R, ČELEDA P, TRAMMELL B, et al. Flow monitoring explained: from packet capture to data analysis with NetFlow and IPFIX[J]. IEEE Communications Surveys & Tutorials, 2014, 16(4): 2037-2064.

[2] SOMMER R, FELDMANN A. NetFlow: information loss or win?[C]// The 2nd ACM SIGCOMM Workshop on Internet Measurment. 2002: 173-174.

[3] MOSHREF M, YU M, GOVINDAN R, et al. SCREAM: sketch resource allocation for software-defined measurement[C]//The 11th ACM Conference on Emerging Networking Experiments and Technologies. ACM, 2015: 14.

[4] YU M, JOSE L, MIAO R. Software defined traffic measurement with OpenSketch[C]//Presented as Part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). 2013: 29-42.

[5] LIU Z, VORSANGER G, BRAVERMAN V, et al. Enabling a RISC approach for software-defined monitoring using universal streaming[C]// The 14th ACM Workshop on Hot Topics in Networks. ACM, 2015: 21.

[6] LIU Z, MANOUSIS A, VORSANGER G, et al. One sketch to rule them all: rethinking network flow monitoring with UnivMon[C]//The 2016 Conference on ACM SIGCOMM 2016 Conference. 2016: 101-114.

[7] CORMODE G, MUTHUKRISHNAN S. What's new: finding significant differences in network data streams[J]. IEEE/ACM Transactions on Networking (TON), 2005, 13(6): 1219-1232.

[8] LIU Y, CHEN W, GUAN Y. A fast sketch for aggregate queries over high-speed network traffic[C]//INFOCOM. 2012: 2741-2745.

[9] SCHWELLER R, GUPTA A, PARSONS E, et al. Reversible sketches for efficient and accurate change detection over network data streams[C]//The 4th ACM SIGCOMM Conference on Internet Measurement. 2004: 207-212.

[10] SCHWELLER R, LI Z, CHEN Y, et al. Reversible sketches: enabling monitoring and analysis over high-speed data streams[J]. IEEE/ACM Transactions on Networking (ToN), 2007, 15(5): 1059-1072.

[11] DAI M, CHENG G, WANG Y. Detecting network topology and packet trajectory with SDN-enabled FPGA platform[C]//The 11th International Conference on Future Internet Technologies. 2016: 7-13.

[12] LI T, SUN Z, JIA C, et al. Using NetMagic to observe fine-grained per-flow latency measurements[C]//ACM SIGCOMM Computer Communication Review. 2011, 41(4): 466-467.

[13] 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.

[14] 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.

[15] SONG H, DHARMAPURIKAR S, TURNER J, et al. Fast hash table lookup using extended Bloom filter: an aid to network processing[J]. ACM SIGCOMM Computer Communication Review, 2005, 35(4): 181-192.

[16] CARTER J L, WEGMAN M N. Universal classes of hash functions[C]//The Ninth Annual ACM Symposium on Theory of Computing. 1977: 106-112.

作者简介：

戴冕（1988-），男，江苏南京人，东南大学博士生，主要研究方向为软件定义网络、数据中心网络和网络测量技术等。

程光（1973-），男，安徽黄山人，东南大学教授、博士生导师，主要研究方向为网络测量、网络安全和网络管理等。