软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.007159]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

意图驱动的网络流量分布式测量方法*

张宇鑫, 李福亮, 元禹博, 王兴伟

(东北大学 计算机科学与工程学院, 辽宁 沈阳 110819)

通信作者: 李福亮, E-mail: lifuliang@cse.neu.edu.cn

摘　要: 可编程交换机的网络流量测量技术凭借其特性可以处理高速网络流量, 在灵活性、实时性等方面均有巨大的优势. 然而, 由于需要使用复杂的 P4 语言配置交换机的内部逻辑, 测量任务部署复杂且易错. 此外, 测量准确度往往受限于交换机内部可用的测量资源. 详细研究基于意图的网络及网络流量测量技术, 提出一种意图驱动的网络流量分布式测量方法. 首先, 设计基于测量意图原语的意图表示形式, 构建意图编译器将抽象意图表示转译为可执行的 P4 代码. 其次, 提出网络流量分布式测量方法, 使用多台交换机的资源以分布式的方式协同完成一个测量任务, 以大流测量为例介绍测量资源动态分配及计数器配置算法. 最后, 实验结果表明所提出的方法可行并且具有一定的优越性.

关键词: 网络流量测量; 基于意图的网络; 意图表示; 意图转译; 可编程交换机

中图法分类号: TP393

中文引用格式: 张宇鑫, 李福亮, 元禹博, 王兴伟. 意图驱动的网络流量分布式测量方法. 软件学报. http://www.jos.org.cn/1000-9825/7159.htm

英文引用格式: Zhang YX, Li FL, Yuan YB, Wang XW. Intent-driven Distributed Network Traffic Measurement. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7159.htm

Intent-driven Distributed Network Traffic Measurement

ZHANG Yu-Xin, LI Fu-Liang, YUAN Yu-Bo, WANG Xing-Wei

(School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China)

Abstract: The network traffic measurement technology of programmable switches is capable of handling high-speed network traffic and offers significant advantages in terms of flexibility and real-time processing. However, due to the necessity of configuring the internal logic of switches using the complex P4 programming language, the deployment of measurement tasks becomes intricate and error-prone. Furthermore, measurement accuracy is often constrained by the available measurement resources within the switch of measurement tasks. This study proposes a detailed exploration of intent-based networking and network traffic measurement technology, introducing an intent-driven network traffic distributed measurement method. Firstly, an intent representation format based on measurement intent primitives is designed, and an intent compiler is developed to translate abstract intent representations into executable P4 code. Secondly, a network traffic distributed measurement approach is introduced, utilizing the resources of multiple switches to collaboratively complete a measurement task in a distributed manner. The dynamic allocation of measurement resources and counter-configuration algorithms are exemplified with heavy-hitter measurements. Finally, experimental results demonstrate the feasibility and certain advantages of the proposed method.

Key words: network traffic measurement; intent-based networking (IBN); intent representation; intent translation; programmable switch

网络测量技术是状态监测、性能管理、安全防御等网络研究工作的基础, 在网络研究领域具有重要地位[1]. 传统网络环境中, 流量测量工作的重点在流量的采集和压缩[2], 其中流量采集是在交换机上对流量进行主动或被动的采集进而分析流量特征, 流量压缩则使用抽样技术和 Sketch[3]数据结构对流量进行概率性存储, 使用少量的

* 基金项目: 国家自然科学基金 (62072091)
收稿时间: 2023-09-19; 修改时间: 2023-11-14; 采用时间: 2024-01-11; jos 在线出版时间: 2024-07-03

存储资源就可以保存大量的统计信息. 随着流量激增, 在面对海量流量时传统的测量技术受限. 随着可编程交换机和数据平面编程语言 P4[4]的出现, 网络测量有了新的发展. 通过利用可编程交换机的特性可以实现更加实时与细粒度的测量任务[1], 相比于传统的基于流量采样和流量压缩的测量方法, 在灵活性、实时性以及准确性方面均有巨大的优势. 然而, 可编程交换机的测量技术面临着以下两个挑战.

(1) 可编程交换机的网络测量技术需要使用 P4 语言配置交换机的内部逻辑, 由于底层设备之间存在差异, 编写过程需要考虑大量与特定设备绑定的接口和参数. 此外, 一旦切换测量场景, 就需要重新编写代码, 进而带来大量不必要的重复工作, 导致测量任务的部署变得复杂且容易出错.

(2) 可编程交换机使用三态内容寻址寄存器 (ternary content addressable memory, TCAM)[5]进行快速匹配, 而由于 TCAM 资源稀缺且价格昂贵, 所以交换机内 TCAM 资源并不多, 且大部分的资源要留给路由、访问控制等功能, 留给测量任务的资源很少[6]. 因此, 在 TCAM 资源受限的情况下, 流量测量的准确率往往不够高.

针对上述挑战, 本文提出了一种意图驱动的网络流量分布式测量方法. 将意图的理念引入到网络测量中, 设计基于测量意图原语的意图表示形式, 构建意图编译器将抽象意图表示转译为可执行的 P4 代码. 根据用户意图自动完成测量任务的部署, 为解决测量任务部署复杂易错的问题提供了良好的解决方案. 当完成足够多的测量意图后, 意图网络可以积累大量模板代码, 快速生成测量策略, 进一步解决反复编写重复代码的问题, 提高测量方法的灵活性和拓展性. 此外, 本文利用多台交换机的资源以分布式的方式协同完成一个测量任务, 设计交换机资源动态分配算法以及计数器配置算法, 能更充分地利用交换机资源, 进而提升测量的准确率.

本文的主要贡献如下.

(1) 提出一个高效、稳定、可拓展的意图驱动的网络流量测量框架, 将基于意图的网络和可编程交换机的网络测量技术进行融合, 推动网络测量向智能化发展.

(2) 设计基于领域特定语言的意图表示形式, 构建意图编译器对意图的表示进行转译, 生成相应的数据平面 P4 代码.

(3) 设计网络流量分布式测量机制, 利用多个交换机以分布式的方式协同完成一个测量任务, 设计交换机资源动态分配算法以及计数器重新配置算法, 充分利用交换机资源, 提升测量的准确率.

本文第 1 节介绍相关工作, 包括基于意图的网络和可编程交换机的网络测量技术. 第 2 节展示意图驱动的网络流量分布式测量方法整体框架. 第 3 节详细阐述测量意图原语的设计与转译. 第 4 节提出网络流量分布式测量方法, 并以网络大流测量为例介绍了测量资源动态分配算法以及计数器配置算法. 第 5 节对本文提出的意图驱动的网络流量分布式测量方法进行实验评估. 第 6 节总结全文.

1 相关工作

本节介绍基于意图的网络以及可编程交换机的网络测量技术相关工作.

1.1 基于意图的网络

随着互联网规模不断扩大, 网络流量激增, 网络管理也在向智能化、自动化方向发展. 在这个背景下, 基于意图的网络 (intent-based networking, IBN) 应运而生. IBN[7]是一种在掌握全局网络状态下, 根据用户意图自动配置和搭建的网络架构. 用户只需要声明想要达到的网络状态, 而不用声明如何实现这个状态, 网络便可以根据“意图”自驱动地完成整个网络的配置[8].

现有的基于意图的网络研究综述从两个方面展示了研究进展. 在文献 [7] 中, 对基于意图的网络的闭环结构, 包括意图获取、意图转译、策略验证、意图下发与执行、意图的实时反馈, 进行了详尽地介绍, 并展望了基于意图的网络在网络测量和网络业务编排这两个场景中的应用. 而文献 [8] 则全面探讨了基于意图的网络的发展流程、关键阶段以及相关研究成果.

此外, 学术界和工业界已经有一些工作对 IBN 的应用进行探索. Kiran 等人[9]提出了 iNDIRA 系统, 该系统采用描述性语言表达用户对网络远程连接和网络配置的意图, 并应用自然语言处理和本体论的知识分析提取意图的关键信息, 根据这些信息对意图进行建模以生成相关的网络配置策略. Scheid 等人[10]将 IBN 应用到网络功能虚拟

化中, 将用户意图转译为一系列网络服务功能链. Tian 等人[11]针对阿里巴巴企业园区网络 ACL 配置更新操作复杂的问题, 设计了一种名为 LAI 的意图语言. LAI 以高级别的抽象形式表达用户的意图, 并利用配置综合的方法[12]生成 ACL 代码, 同时进行有效性验证, 实现了 ACL 的自动配置和更新, 从而避免了重大的服务中断. Kang 等人[13]提出了在云环境下实现意图的方法, 通过分析意图在云设备中的应用和关系实现用户需求.

1.2 可编程交换机的网络测量技术

相较于传统的网络流量测量技术, 可编程交换机的网络流量测量技术具有许多显著的优势. 由于能在可编程交换机内部进行存储和计算, 它具备更好的实时性、更精细的测量精度和更高的测量灵活性.

Yu 等人[3]提出了基于可编程交换机的 OpenSketch 测量框架, 该框架在可编程交换机的数据平面中实现了哈希算法、分类统计、计数的三级数据处理管道. 哈希算法用于流量数据的压缩, 分类统计则利用 TCAM 的通配符属性对流量进行过滤和分类, 最后使用交换机内的计数器来统计流量信息. 这一框架首次将 Sketch 结构引入可编程交换机, 极大提高了测量效率, 为后续基于 Sketch 的流量测量方法如 SketchVisor[14]、SketchLearn[15]和 Elastic Sketch[16]等提供了坚实的基础. Gupta 等人[17]基于可编程交换机提出了一个网络遥测系统 Sonata, 该系统将网络数据流类比为结构化的大数据流, 并将大数据处理框架 Flink 中的 API 引入到网络测量领域, 提出了 map、reduce、filter 等针对网络流的处理操作, 解决了网络测量部署复杂等问题. Laffranchini 等人[6]借鉴 Sonata 的思想设计了网络测量关键字, 提升了网络测量的灵活性和拓展性.

此外, 基于可编程交换机的网络流量测量技术发展受限于交换机内的资源限制, Moshref 等人[18]针对 SDN 交换机中测量资源不足的问题, 提出了 DREAM 机制, 通过对交换机资源进行动态分配, 充分利用测量资源以提高测量准确性. 文献 [19] 进一步扩展了 DREAM, 将 Sketch 结构引入到系统中, 从而使得测量的种类变得更加丰富, 且测量速度更快. Wang 等人[20]借鉴了 DREAM 的思想, 通过优化算法对网络大流的测量方法进行优化, 提升了网络大流的测量速度. 本文也借鉴了 DREAM 的思想, 将其拓展到 P4 可编程交换机环境中, 改进了其资源分配和流量测量的方法, 通过量化资源分配, 使得该机制可以更好地拓展到 P4 可编程交换机, 从而提高了测量的速度. 此外, 本文引入了“流量树”概念, 以便于更清晰地分析网络流量特征, 克服了 DREAM 因流量突变使其无法工作的问题.

2 整体框架

本文提出的一种意图驱动的网络流量分布式测量方法整体框架如图 1 所示, 其中主要包括意图表示模块、意图引擎模块以及网络流量分布式测量模块, 此外还包括测量策略下发和测量数据上传等机制.

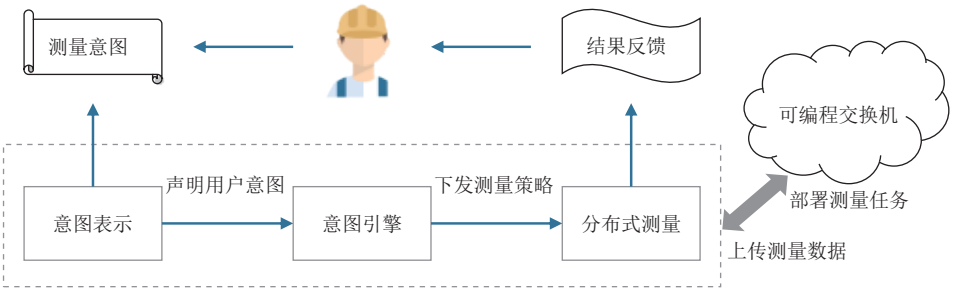


图 1 意图驱动的网络流量分布式测量方法框架

意图表示模块负责用户意图的显式声明, 本文设计了一种基于领域特定语言 (domain specific language, DSL) 的意图表示方法, 以一种抽象级别较低但灵活的意图原语形式声明用户意图. 意图表示模块屏蔽了大量底层代码的细节, 使用户只专注于测量算法的设计, 提高了测量方法的灵活性和拓展性, 减少了大量重复工作, 不仅为未来提出的新测量指标提供复用的可能性, 还为测量任务部署复杂易错的问题提供了良好的解决方案.

意图引擎模块是意图驱动的网络流量分布式测量方法的核心部分, 主要负责测量意图的转译. 其中针对基于 DSL 的意图表示, 设计了意图转译方法, 并实现了意图编译器. 意图编译器首先对意图原语文件进行预处理和解

析, 生成意图原语的抽象语法树, 然后将其进一步细分为原子操作, 结合原语模块和底层设备代码的格式, 生成可以在交换机上执行的 P4 代码.

网络流量分布式测量模块主要对测量算法进行优化, 利用多个交换机的资源协同完成一个测量任务. 首先, 根据预定的测量策略初始化测量任务对象和资源分配方式, 并配置需要监测的节点. 然后, 对测量结果进行评估, 动态调整资源分配, 以确保准确率较低的任务能够获得更多的资源支持. 最后, 根据测量时间是否到达, 进行新一轮迭代或结束测量任务.

3 测量意图原语的设计与转译

3.1 测量意图原语的设计

本文设计了基于 DSL 的意图表示形式, 使用测量意图原语表示用户测量意图. DSL 是针对计算机某个领域内一种表达受限的程序设计语言, 与通用的高级编程语言相比, 具有更简单的语法和语义. 因此当开发环境在某个特定的场景时, DSL 的开发效率会更高且容易理解.

设计基于 DSL 的意图表示形式的关键在于设计一组合理的意图原语. 这些原语可以执行一系列操作, 如创建测量数据结构并对其进行处理. 结合本文中意图在网络测量场景的应用以及 DSL 的特性, 设计意图原语时需要遵循以下原则.

(1) 意图原语不仅可以表示常见的测量指标, 还需要为未来提出的新测量指标提供复用的可能性, 从而提高意图原语的可扩展性.

(2) 意图原语应处于适当的抽象级别, 既可以处于较低抽象级别以实现多种测量任务, 体现其灵活性, 又可以处于较高抽象级别以简化用户操作, 减少工作量.

(3) 意图原语应考虑可编程交换机的特性, 充分利用可编程交换机的优势.

以上述 3 点原则为指导, 通过总结网络流量测量方法中常用的数据结构和相关操作, 设计了如表 1–表 3 所示的意图原语, 这些原语不仅屏蔽了底层复杂性, 还可以通过组合展现丰富的语义, 从而更灵活地实现多种流量测量任务.

表 1 数据结构类型的意图原语

数据结构	原语表示	说明
时间戳结构	Timestamp(name)	创建当前时间戳
计数器结构	Counter(name, width)	创建计数器
Sketch结构	Sketch(name, type, key, hashNum, width)	创建Sketch
Bloomfilter结构	Bloomfilter(name, type, key, hashNum, width)	创建Bloomfilter

表 2 操作类型的意图原语

操作	原语表示	说明
对计数器操作	Counter_op(name, action, object)	更新计数器
对Sketch操作	Sketch_op(name, action, object)	更新Sketch
对Bloomfilter操作	Bloomfilter_op(name, action, object)	更新Bloomfilter
数据过滤操作	Filter(condition)	对数据匹配过滤

表 3 操作顺序的意图原语

操作顺序	原语表示	说明
顺序操作	action1 >> action2	action1先于action2执行
并行操作	action1 + action2	action1与action2同时执行

表 1 设计的意图原语用于声明在创建测量任务时需要使用的数据结构, 下面具体说明这些原语的用法和含义.

(1) Timestamp: name 参数表示该结构在 P4 语言中的名称, Timestamp 原语用于读取当前交换机的本地时间,

主要用于与时间参数相关的测量任务. 由于可编程交换机自带 Timestamp 数据结构, 因此可以直接调用该接口.

(2) Counter: width 参数表示位数, Counter 原语用于对数据包的数量、大小等指标进行计数, 主要用于统计类型的测量任务. 可编程交换机自带 Counter 数据结构, 使用 TCAM 资源进行快速匹配, 因此在使用 Counter 时需要注意资源问题.

(3) Sketch: type 参数表示 Sketch 的类型, 目前仅支持 Count-min Sketch, key 参数表示进行哈希操作所使用的关键字, hashNum 参数表示哈希函数的数量. Sketch 原语可以对数据包的数量、大小等进行统计, 相较于 Counter, 其统计是概率性的, 结果可能会有一定误差. 但 Sketch 可以使用 SRAM 资源实现, 不用考虑资源限制的情况, 且在海量数据情况下, 可以计算 Sketch 的误差率.

(4) Bloomfilter: type 参数表示 Bloomfilter 的类型, 目前支持 membership 和 counting 类型, 分别用于成员关系的高效查找以及概率性计数.

表 2 设计的操作类型的意图原语用于对数据结构原语和数据包进行操作, 下面具体说明这些原语的用法和含义.

(1) Counter_op: name 参数表示该操作在 P4 代码中的名称, action 参数表示对 Counter 结构进行的操作, 其中 set 操作表示设置计数器的值, object 表示操作的对象.

(2) Sketch_op: action 参数表示对 Sketch 结构进行的操作, 其中 set 操作表示设置新值, sum 操作表示获取 Sketch 结构中数量的总和, min、max、avg 功能类似.

(3) Bloomfilter_op: action 参数表示对 Bloomfilter 结构进行的操作, 对于 membership 型的 Bloomfilter, insert 负责插入新值, test 负责检查值是否在其中; 对于 counting 类型的 Bloomfilter, 其操作与 Sketch 的操作类似.

(4) Filter: condition 参数表示当前匹配条件, filter 类似高级语言中的 if, 表示过滤操作.

表 3 设计了用于对不同操作原语之间的操作顺序, 下面对这两种顺序进行说明.

(1) 顺序操作: “>>”前后的原语按顺序执行, 之前的原语会对后续的原语产生影响, 如果之前的原语更新了计数器, 那么后续的原语将读取更新后的值.

(2) 并行操作: “+”前后的原语互相独立执行, 对数据包进行不同的处理.

通过对意图原语的组合可以表示丰富的测量意图, 下面以网络大流测量为例进一步展示意图原语的使用. 网络大流通常指在单位时间内流量超过一定阈值且粒度最小的流量, 本例的网络大流测量意图为测量来自端口 10086 的超过流大小 50% 的流, 设计如图 2 所示的基于 DSL 的网络大流测量意图表示.

```
1  flowKey = [ip.src, ip.dst, tcp.sport, tcp.dport, ip.proto]
2  totalSize = Counter(name=totalSize, width=32)
3  hhSize = Counter(name=hhSize, width=32)
4  hhSet = Bloomfilter(name=hhSet, type=membership, key=flowKey, hashNum=4, width=32)
5  statisticSize = Sketch(name=statisticSize, type=count_min_sketch, key=flowKey, hashNum=4, width=32)
6  incTotal = Counter_op(name=incTotal, action=totalSize.set(totalSize.get()+pkt.size), totalSize)
7  incStatistic = Sketch_op(name=incStatistic, action=statisticSize.set(pkt.size), statisticSize)
8  testHh = filter(statisticSize.get()/totalSize.get() > 0.5) >> Bloomfilter_op(name=testHh, action=hhSet.\
       insert(flowKey), hhSet)
9  incHh = filter(hhSet.test()) >> Counter_op(name=incHh, action=hhSize.set(hhSize.get()+pkt.size), hhSize)
10 measurement = incTotal >> filter(pkt.sport==10086) >> incStatistic >> (testHh+incHh)
```

图 2 基于 DSL 的网络大流测量意图表示

图 2 中的意图共需要 10 行代码就可以描述大流测量意图. 这些意图原语屏蔽了大量底层代码的细节, 使用户只专注于测量算法的设计就可以完成测量任务, 提高了测量方法的灵活性和拓展性, 减少了大量重复工作, 为解决测量任务部署复杂易错的问题提供了良好的解决方案.

3.2 意图转译

设计基于 DSL 的意图表示方法的目标是通过一系列操作将 DSL 转换为相应的 P4 代码, 以完成意图的转译.

在 P4 代码的结构中, header 和 parser 部分是固定的, 意图编译器主要负责生成 P4 代码中 apply 块逻辑代码以及对应的 table 和 action 的代码. 本文设计了如图 3 所示的意图编译器, 其工作流程如下.

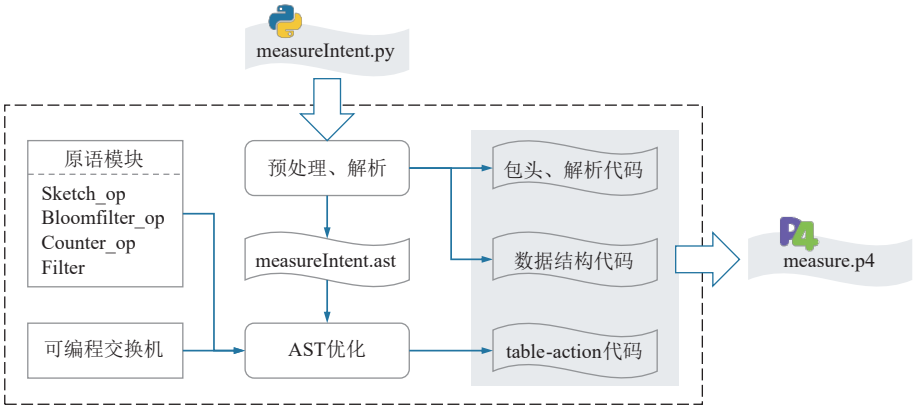


图 3 意图编译器

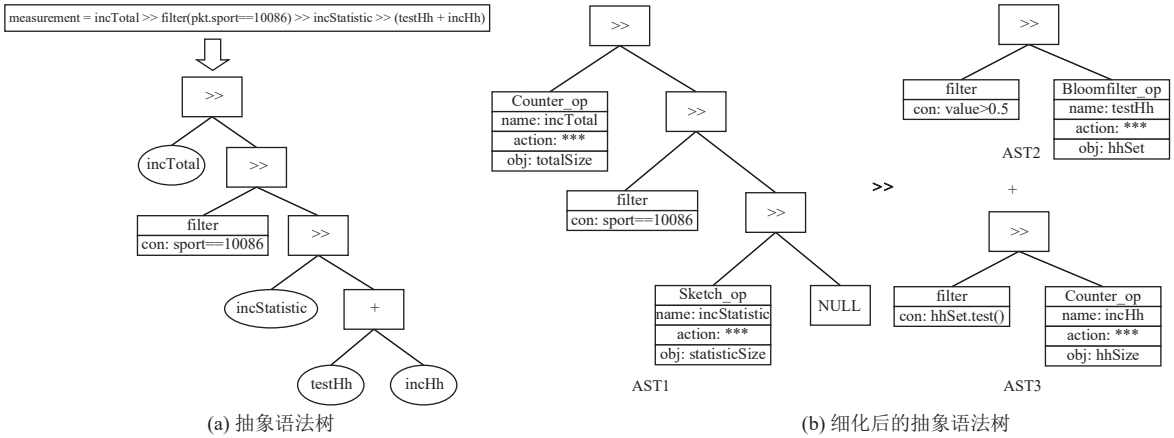
(1) 首先, 对编写好的意图原语文件进行预处理和解析, 通过预处理去掉注释以及多余符号后, 解析生成以下 3 个内容: 意图原语的抽象语法树、header 以及 parser 代码、数据结构代码.

(2) 然后, 对意图原语的抽象语法树进行优化, 将“+”原语的操作拆分为新的语法树后进一步细分为原子操作. 结合原语模块和底层设备代码的格式生成 P4 代码中 table 和 action 的代码以及 apply 块对应的控制逻辑.

(3) 最后, 通过结合上述完成的各部分代码, 生成可以在交换机上执行的 P4 代码.

显然, 意图编译器的重点在于生成 apply、table、action 相关的代码. 相比传统编译器使用词法分析、语法分析、文法分析等一系列的代码生成及优化操作, 本文设计的意图编译器核心部分是使用抽象语法树进行代码生成. 接下来以图 2 为例, 介绍如何通过抽象语法树生成目标代码.

本文定义 DSL 文件中的最后一行代码即以 measurement 为开头的一系列原语的组合作为操作的入口. 将图 2 的基于 DSL 的网络大流测量意图表示转换为如图 4(a) 所示的抽象语法树的形式.



(a) 抽象语法树 (b) 细化后的抽象语法树

图 4 意图原语抽象语法树

遍历整个语法树, 从意图表示文件中搜索组合操作的定义, 并将其进一步分解为原子操作. 对于“+”原语, 将其左右节点从树中分离出来, 创建一个新的抽象语法树, 并递归地对语法树进行细化. 对于“>>”原语, 无需进行任何操作. 图 4(a) 中的抽象语法树可以进一步细化为图 4(b) 所示.

由于每棵树的内部结点都是“>>”原语, 其中左子树的操作优先于右子树, 因此通过对每棵树后序遍历, 可以

获得该语法树中原语操作的顺序. 如图 4(b) 所示, 3 棵语法树的执行顺序可以是 AST1 → AST2 → AST3 或 AST1 → AST3 → AST2 , 原语执行顺序对应着 P4 代码 apply 块中的逻辑执行顺序.

P4 数据平面通过调用原子表 (table) 与其定义的动作 (action) 对数据包进行操作, 因此, 需要将意图原语转译为相应的 table 和 action. 对于数据结构操作的 action, 在解析器生成数据结构时已经生成. 至于 filter 原语, 它用于生成条件判断语句, 对于每种操作类型的意图原语, 都会生成一个 table. 该表的名称由 name 参数指定, 执行的动作由 action 参数指定, 对于 filter 原语则是使用预定义的名称作为表名, 并利用预先编写的意图原语模块以及对应的交换机类型来生成 table 和 action 的代码.

以上两个步骤分别生成了原语操作顺序及对应的 table 和 action, 将其组合可以生成对应的 apply 块代码.

4 网络流量分布式测量方法

网络流量分布式测量方法主要以网络大流的测量为例进行深入分析和优化. 传统方法通常在单个交换机上进行操作以获得大流测量结果, 但由于资源限制, 这种方法可能无法提供准确的实时结果. 本文通过观察流量在多个交换机之间传输的情况, 提出了协同利用多个交换机资源进行分布式测量的解决方案.

4.1 算法流程

网络流量分布式测量方法的流程图如图 5 所示.

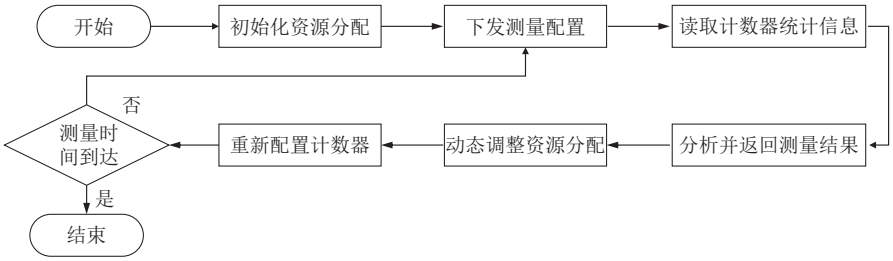


图 5 网络流量分布式测量流程图

首先, 根据大流测量策略确定要测量的网络前缀, 初始化测量任务对象和资源分配, 并根据网络流量的前缀信息配置计数器需要监测的节点. 然后, 将计数器配置下发到数据平面, 在每个测量周期结束后, 检查计数器统计的流是否超过阈值, 如果超过则标记前缀为网络大流. 接下来, 根据测量结果评估任务的准确率, 并动态重新分配资源, 以确保准确率低的任务获得更多计数器资源. 资源重新分配后, 重新配置计数器, 随后根据测量时间进行新一轮迭代或结束测量任务.

4.2 测量资源动态分配方案

在测量准确率达到 80% 后, 随着资源的增加, 准确率的提升速度逐渐减缓. 为了将准确率提升至 100%, 需要投入数倍的资源. 考虑到这一现象, 默认网络大流测量任务的目标准确率为 80%.

为了评估网络大流测量的准确率, 本文引入了召回率 $recall$, 如公式 (1) 所示. 召回率是指探测为真的大流数量占所有大流数量的比重, 而所有大流数量需要对流量进行采样和离线分析才能得到. 通过观察流量树评估一个大流前缀可能会漏掉的大流数量, 能够估算出该网络流内一共有多少个大流, 如公式 (2) 所示, 其中 x 表示通配符数量, vol 表示该节点上监测到的值, th 表示大流探测阈值. 如果检测到的前缀为叶子节点, 则一定不会漏掉大流, 而如果检测到的前缀为内部节点, 则可以取 2^x 和 $\lfloor vol/th \rfloor$ 之间的最小值作为漏掉的大流数量评估值.

$$recall = \frac{detectedNum}{detectedNum + missedNum} \tag{1}$$

$$missedNum = \begin{cases} 0, & \text{如果检测的是叶子节点} \\ min(2^x, \lfloor vol/th \rfloor), & \text{其他情况} \end{cases} \tag{2}$$

为了方便描述, 将大流测量的召回率统称为准确率. 为了进一步描述何时进行动态资源分配, 定义了 ga_i 和 $la_{i,s}$ 分别表示任务 i 的全局准确率和任务 i 在交换机 s 上的局部准确率, 并定义了 ra_i 表示用户需要或系统默认的准确率. 每一轮测量周期结束后, 通过获取交换机上计数器的信息, 可以分析出每个任务的全局准确率以及每个任务在各自交换机上的局部准确率. 当 $ga_i < ra_i$ 时, 表明准确率未达到目标, 此时需要对该任务进行资源分配.

资源分配过程是在每个交换机上单独进行的, 如果任务 i 的 $ga_i < ra_i$, 就给所有 $la_{i,s} < ra_i$ 的交换机上的任务 i 分配更多计数器资源, 这些计数器资源应优先从备用空间中分配. 如果备用空间中的资源不足, 需要在不同任务之间进行资源分配. 为了描述任务间的资源分配, 提出了两个概念: 富任务和穷任务. 其中富任务是指在交换机 s 上, 某一个任务的 ga_i 和 $la_{i,s}$ 均大于 ra_i , 而穷任务是指需要分配资源的任务, 即任务的 ga_i 和 $la_{i,s}$ 均小于 ra_i . 将富任务中的一部分资源分配给穷任务不会明显降低富任务的准确率, 却能显著提高穷任务的准确率. 重新进行资源分配并重新配置计数器后, 可以进行新一轮的测量迭代. 通过多轮迭代, 使每个任务的测量准确率都高于默认准确率.

算法 1. 测量资源动态分配算法.

输入: 当前测量任务集合 *tasks* ;
输出: 测量迭代的资源分配方案.

1. 读取计数器的统计信息, 并存入集合 *switches* ;
2. **for** $task \in tasks$
3. $total \leftarrow 0,\ missed \leftarrow 0$;
4. **for** $s \in switches$
5. $total \leftarrow total$ +交换机 s 中检测到的大流个数;
6. $missed \leftarrow missed$ +根据公式 (2) 计算交换机 s 中 *task* 漏掉的大流个数;
7. 计算 *task* 在该交换机上的局部准确率并存到 $task.la_{s,i}$;
8. 根据 *total* 和 *missed* 计算 *task* 的全局准确率并存到 *task.ga* ;
9. **for** $s \in switches$
10. $rich \leftarrow [],\ poor \leftarrow []$;
11. **for** $task \in s.tasks$
12. 根据 *task* 的全局准确率和局部准确率将任务划分到 *rich* 和 *poor* 中;
13. **if** $poor \neq \varnothing$
14. $totalRich \leftarrow 0,\ totalPoor \leftarrow 0$;
15. **for** $task \in poor$
16. 计算该任务增加的计数器个数 *num* ;
17. $totalPoor \leftarrow totalPoor + num$;
18. **if** $totalPoor <$ 交换机 s 的备用空间大小
19. 使用备用空间将资源分配给穷任务;
20. continue;
21. **for** $task \in rich$
22. 计算该任务减少的计数器个数 *num* ;
23. $totalRich \leftarrow totalRich + num$;
24. **if** $totalPoor \leqslant totalRich$
25. 将 *totalRich* 的资源分配给所有的穷任务, 并将剩余资源归还备用空间;
26. **else**
27. **for** $task \in poor$

28. 计算该任务增加的计数器个数 num;
29. 分配给该任务 $totalRich \times (num/totalPoor)$ 数量的资源;

在每轮测量算法的迭代过程中, 需要考虑每次计数器分配的数量大小, 这个大小对应着每轮迭代的“步长”. 由于富任务对资源的需求没有穷任务那么“迫切”, 把穷任务每次获得的计数器的数量设定为 $totalCounter \times q\% + 4$, 而将富任务每次释放的计数器的数量设定为 $totalCounter \times p\% + 2^x$ (x 为迭代次数, q 和 p 可以根据任务的数量调整). 随着迭代周期的增加, 富任务释放的资源会逐渐增加, 而穷任务获得的资源则保持不变.

根据上述分析, 本文设计了算法 1 测量资源动态分配算法. 首先, 统计每个任务在不同交换机上计数器的使用情况, 并计算每个测量任务的全局准确率以及在每台交换机上的局部准确率. 然后, 在每台交换机上根据任务的全局准确率和局部准确率找到穷任务的集合和富任务的集合, 并根据资源可分配的大小关系分 3 种情况进行资源分配.

4.3 计数器配置方案

计数器的配置方案是指在每轮测量的迭代周期中, 该任务所拥有的计数器资源应对哪些流的前缀进行监测并统计. 每个任务得到初始的计数器资源后, 需要根据网络大流测量的参数平均分配计数器资源, 并根据待测网络大流的前缀信息配置计数器. 每个任务对计数器配置的实质是维护该任务的流量树, 其中该流量树的叶节点是待监测的网络大流的前缀, 通过对这些前缀进行回溯, 维护流量树. 在每轮测量迭代结束后通过遍历流量树可以得到测量结果并生成新的计数器配置方案.

如图 6 所示, task1 要检测 X*** (X 表示 IP 位) 流中的所有大流, 初始分配两个计数器资源, task2 要检测 Y*** 流中的所有大流, 初始分配一个计数器资源. 图 6 中红色圈标记的是初始要监测的 IP 前缀, 其中 task1 监测 X0** 和 X1**, 当监测到这两个前缀的流量值后就可以得到 X*** 的流量值, 从而构建出一颗不完整的流量树, task2 同理. 为了在每轮迭代后都能维护该流量树, 需要持续测量每个初始节点, 其本质是维护 N 棵子流量树, 这 N 棵子流量树可以进一步构建为一棵完整的流量树. 对于 task1 来说, 由于所有流量都经过 S1、S2、S4、S5, 所以 task1 上的两个计数器资源既可以来自以上任意两个交换机, 也可以来自一个交换机, 只要资源足够并且可以维护该任务的流量树即可.

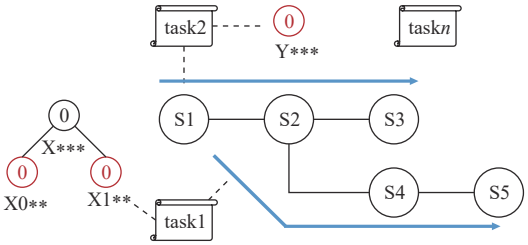


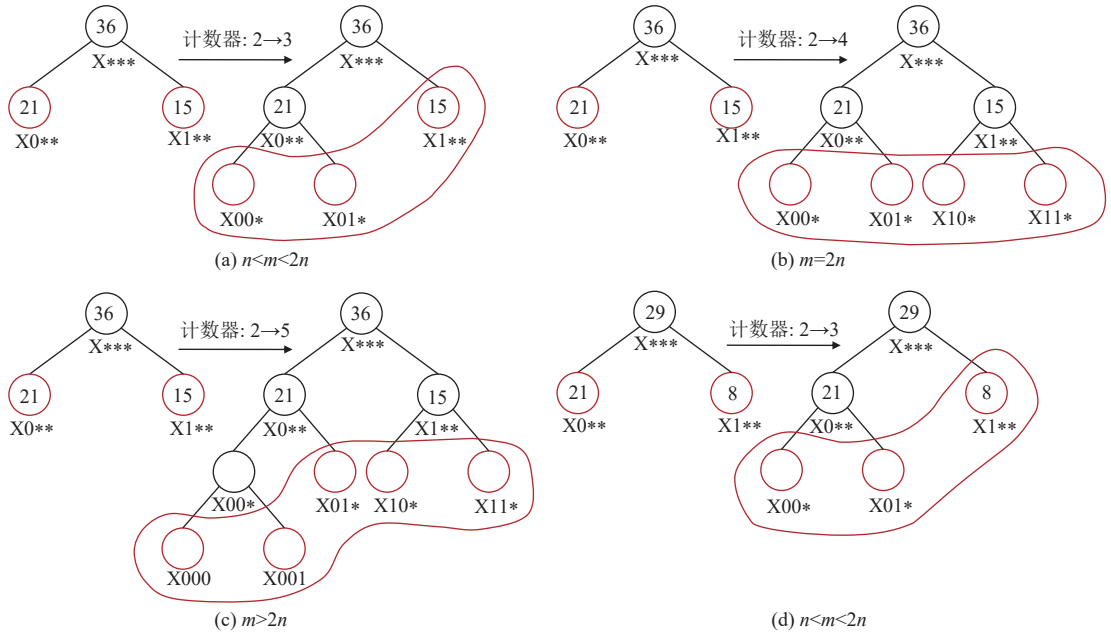
图 6 计数器初始配置方案

在获得新的资源分配方案后, 各个任务将根据新的计数器分配数量进行调整, 以确保在下一轮迭代中, 每个任务都能获得更高的测量准确率.

计数器重新配置算法的核心思想是将每个测量任务在所有交换机上拥有的资源视为一个整体. 由于所有待测流量都会经过这些交换机, 因此每个待测的子流都可以获得计数器资源识别网络中的大流, 并为下一轮迭代配置计数器. 计数器的重新配置是通过对疑似大流, 即不是最细粒度的大流进行划分、对小流进行合并实现的. 其中划分会占用更多的计数器, 合并会释放一些计数器, 同时划分和合并操作要以新的资源分配方案为限制.

划分操作是将当前穷任务检测到的疑似大流进一步分解, 以确定是否存在更细粒度的大流. 图 7 展示了图 6 中 task1 在经历一轮迭代后计数器的统计情况及新的计数器配置方案. 将上一轮计数器的数量记为 n, 新分配的计数器数量记为 m. 图 7(a) 展示了 $n < m < 2n$ 且有 n 个疑似大流的情况, 按照计数器的监测值从大到小的顺序逐个划分每一个疑似大流的节点, 并为监测无法划分的节点分配一些计数器资源. 这个过程可以类比于求解方程 $2 \times X1 + (n - X1) = m$, 其中 $X1$ 为要分解的疑似大流的数量. 图 7(b) 展示了 $m = 2n$ 且有 n 个疑似大流的情况, 因此

可以划分所有的疑似大流. 图 7(c) 展示了 $m > 2n$ 且有 n 个疑似大流的情况, 依次从最大的流进行进一步的划分. 图 7(d) 展示了 $n < m < 2n$ 有不足 n 个疑似大流的情况, 尽管 X1** 并非大流, 但它是初始子流量树的根节点, 因此需要持续监测这个节点, 消耗一个计数器, 其余的计数器继续对疑似大流进行划分.

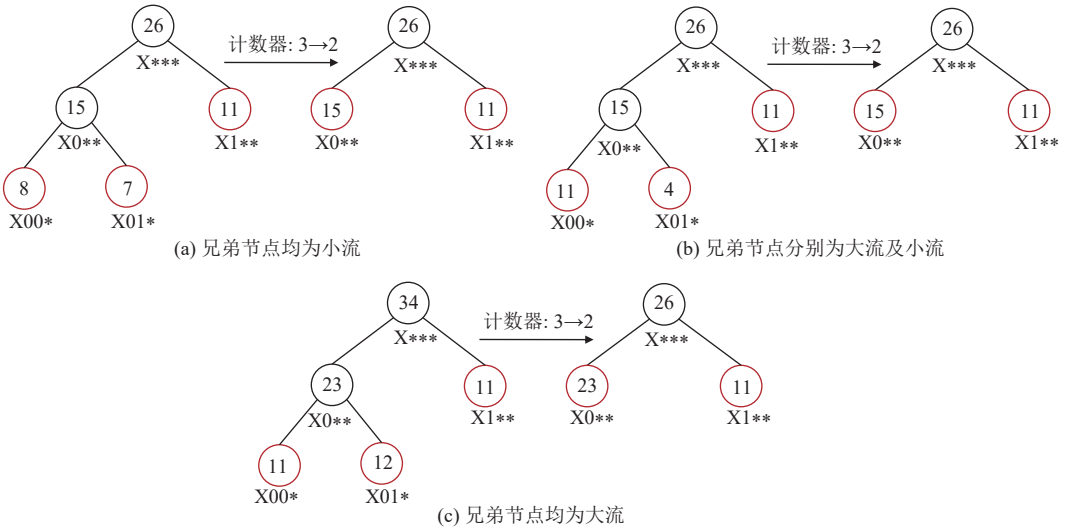


(a) $n<m<2n$ (b) $m=2n$

(c) $m>2n$ (d) $n<m<2n$

图 7 划分操作示例

合并操作是将富任务中的一些小流或者相对较小的大流进行合并, 使得富任务释放计数器资源, 进而匹配新的资源分配方案. 如图 8 所示, 图 8(a) 展示了两个兄弟节点都是小流的情况, 优先合并为其父节点. 图 8(b) 展示了两个兄弟节点一个是大流一个是小流的情况, 这种情况的合并优先级略低. 图 8(c) 展示了兄弟节点都是大流的情况, 这种情况的优先级最低, 通常只在没有其他节点可供合并时才会进行此类合并操作. 合并操作同样也需要维护该任务的流量树, 且不能合并流量子树的根节点.



(a) 兄弟节点均为小流 (b) 兄弟节点分别为大流及小流

(c) 兄弟节点均为大流

图 8 合并操作示例

综上所述, 计数器重新配置算法如算法 2.

算法 2. 计数器重新配置算法.

输入: 当前测量任务集合 *tasks*;
输出: 测量迭代的计数器的配置方案.

1. **for** $task \in tasks$
2. 根据计数器的统计信息维护当前任务的流量树;
3. $rich \leftarrow []$, $poor \leftarrow []$;
4. **for** $task \in tasks$
5. 若 $task.ga \geqslant ra$, 则将 *task* 存入 *rich*, 否则存入 *poor*;
6. **if** $poor = \varnothing$
7. 退出算法, 使用上一轮迭代的计数器配置;
8. **for** $task \in poor$
9. $hh \leftarrow []$, $n1 \leftarrow 0$; 遍历 *task* 流量树叶子节点, 大流存入 *hh*, 小流数量记为 $n1$;
10. 将 *task* 新计数器数量记为 m; 对 *hh* 中大流由大到小排序;
11. $total \leftarrow m - n1$, *total* 记录总计数器个数; $X1 \leftarrow 0$, $X1$ 记录待分解的大流个数;
12. 由方程 $2 \times X1 + (hh.size() - X1) = total$, 计算 $X1$ 的值;
13. **if** $X1 \leqslant 0$
14. 将 *hh* 中前 $X1$ 个大流划分, 并使用计数器监测;
15. **else**
16. 将 *hh* 中所有大流划分, 对前 $X1$ 个大流进行多次划分;
17. 将以上的配置记录到 *task* 中;
18. **for** $task \in rich$
19. $siblingNodes \leftarrow []$; 遍历流量树, 将相应节点以 [node1, node2] 形式存入 *siblingNodes*;
20. 将该 *task* 上一轮迭代中计数器数量记为 $n2$, 新的计数器数量记为 $n3$;
21. 将 *siblingNodes* 按照节点值的和由小到大排序;
22. $diff \leftarrow n2 - n3$;
23. **while** $diff > 0$
24. 取 *siblingNodes* 中最小的节点进行合并, 并将该节点从 *siblingNodes* 中取出;
25. $diff \leftarrow diff - 1$;
26. 重新维护流量树, 并执行第 19 行和第 21 行的操作;
27. 将以上的配置记录到 *task* 中;

5 实验评估

为验证本文提出的意图驱动的网络流量分布式测量方法的有效性和可行性, 本节对意图表现力、意图驱动效率及意图转译时间进行分析, 并对网络流量分布式测量方法性能进行评价. 针对意图驱动的网络流量分布式测量方法实验, 设计了如图 9 所示的拓扑.

5.1 意图语言效果评价

意图语言效果评价主要包括 3 个方面: 意图表现力、意图驱动效率及意图转译时间. 意图表现力是指意图原语是否能准确描述多样的测量需求; 意图驱动效率则关注基于意图的方法是否能简化用户操作, 降低测量任务的复杂性和错误率; 意图转译时间为能否在合理时间内转化为实际的测量策略. 具体示例如表 4 所示.

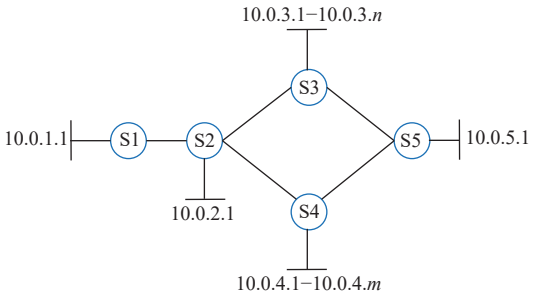


图 9　实验拓扑

表 4　意图驱动测量示例

测量指标	意图原语行数	生成P4代码行数	转译时间 (ms)
Heavy hitter	10	164+72+235=471	18–32
Flow size	18	144+32+101=277	15–69
Top-k flow	8	152+62+162=376	17–52
Path change	8	166+64+284=514	19–57
Flow cardinality	13	140+18+62=220	13–22
Path lantecy	9	104+30+72=206	13–23
SuperSpread	14	142+54+117=313	15–31

意图表现力: 对于各种不同的测量需求, 仅需 8–18 行意图原语代码即可精确表达用户的测量意图. 灵活组合不同的意图原语, 用户可以用较少的输入实现对测量需求的表达, 体现出测量意图原语具有较强的表现力.

意图驱动的效率: 仅使用 10 行网络大流测量意图原语, 就能转译生成 471 行 P4 代码, 其中包括 164 行 header 以及 parser 代码、72 行 table 代码以及 235 行 action 代码, 能在可编程交换机上正确运行. 这极大地降低了测量任务实施的复杂性, 减少了大量的重复工作, 进而提高了测量任务的实施效率.

意图转译时间: 由于意图编译器需要生成抽象语法树并对其进行遍历和细化, 因此意图转译需要一定的时间. 尽管有个别意图的最长转译时间为 69 ms, 但大多数测量意图的转译时间都在 20–30 ms 之间, 因此对于持续时间较长的测量任务来说, 该转译时间在可接受范围之内.

5.2　网络流量测量方法性能评价

本节用网络大流测量速度、并发执行任务数量这两个指标对网络流量分布式测量方法进行评价. 网络大流测量速度指在达到期望准确率之前的迭代次数, 并发执行任务数量则表示可以同时执行的任务数量. 此外, 由于软件交换机无法直接量化 TCAM 资源的数量, 因此以计数器数量作为 TCAM 资源的量化指标.

(1) 网络大流测量速度

首先, 在交换机测量资源不同的条件下对网络大流测量速度进行评估. 为了模拟交换机测量资源不同的情况, 分别将每台交换机的计数器数量限制为 25、50、75、100, 之后下发大流测量任务, 并将本文提出的分布式测量的参数 N 设为 2, q 设为 10%, p 设为 5%, 期望准确率为 80%, 结果如图 10 所示.

由图 10 可以看出, 在每台交换机拥有 100 个计数器的情况下, 只需要 3 次迭代就可以达到 80% 以上的准确率, 而在每台交换机只拥有 25 个计数器的情况下, 则需要 6 次迭代才能达到 80% 以上的准确率. 每台交换机只有 25 个计数器的情况对应真实网络环境中测量资源较少的情况, 通过分布式测量方法使得该任务最多可以使用 100 个计数器, 因此可以通过较少的迭代得到准确的测量结果.

为更全面评价本文提出的网络流量分布式测量方法, 使用 DREAM[18]机制作为参考. DREAM 采用动态资源分配的方式来分配测量资源, 提升测量准确率. 相比于 DREAM, 本文提出的网络流量分布式测量机制在资源分配和计数器配置算法上均做出优化. 如图 11 所示, 通过比较每台交换机拥有不同数量的计数器, 两种方法在达到期望准确率前的迭代次数, 发现在资源较多的情况下, 两种方法都能在较少迭代次数内达到期望准确率, 但在资源较

少的情况下, 本文提出的方法优于 DREAM, 因为它能够更有效地分配测量资源, 使测量任务能够更快速地达到期望准确率.

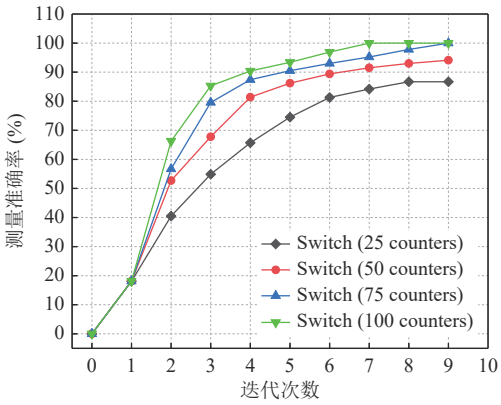


图 10 不同资源下测量任务的迭代次数

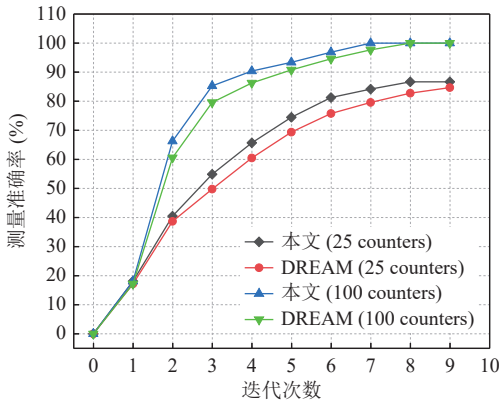


图 11 两种机制下测量任务的迭代次数

(2) 并发执行任务数量

并发执行的任务数量不仅能够反映出测量机制在高强度任务下系统的承受能力, 还能够反映出测量机制对于资源的充分利用程度. 为了评估这一指标, 将每台交换机的可用计数器数量限制为 1 000 个, 并将本文提出的分布式测量的参数 N 设为 2, q 设为 5%, p 设为 5%, 期望准确率为 80%, 计算每个任务达到稳定状态时的平均测量准确率, 如表 5 所示.

表 5 不同测量任务数量下两种机制的平均测量准确率对比

测量任务个数	本方法的平均测量准确率 (%)	DREAM的平均测量准确率 (%)
10	99.8	99.6
20	98.3	97.6
30	95.4	94
40	93.3	90.5
50	89.6	87.6
60	85.3	85.9
70	80.8	83.5
80	75.6	81.7
90	65.1	72.5
100	53	63.8

在达到平均测量准确率 80% 以上的情况下, 本文提出的网络流量分布式测量方法最多可同时执行约 70 个测量任务, 而 DREAM 则可以同时执行约 80 个测量任务. 这表明 DREAM 在高负载网络环境中的承受能力要优于本文提出的方法. 然而, 当执行任务较少时, 平均测量准确率要高于 DREAM. 这表明在并发执行任务较少的情况下, 本文提出的网络流量分布式测量方法能更充分地利用测量资源, 获得准确的测量结果. 虽然 DREAM 在同时执行任务数量较多的情况下具有优势, 但它可能面临着“流量突变”的问题, 即在某一时刻, 非疑似节点的流量突然增大, 疑似节点的流量减小, 而 DREAM 无法检测到这种情况, 因此错过了这些流量. 本文提出的网络流量分布式测量方法通过维护流量树可以检测到“流量突变”, 从而克服了 DREAM 所面临的问题.

6 结 论

本文提出了一种意图驱动的网络流量分布式测量方法. 将意图的理念引入到网络测量中, 设计基于测量意图原语的意图表示形式, 构建意图编译器将抽象意图表示转译为可执行的 P4 代码. 根据用户意图自动完成测量任务

的部署, 为解决测量任务部署复杂易错的问题提供了良好的解决方案. 此外, 利用多台交换机的资源以分布式的方式协同完成一个测量任务, 以大流测量为例设计交换机资源动态分配算法以及计数器配置算法, 能更充分地利用交换机资源进而提升测量的准确率. 实验结果表明本文设计的意图驱动的网络流量分布式测量方法可以通过十几行代码生成数百行可执行的 P4 代码, 极大降低了测量方法实现的复杂性, 提升了测量算法部署的效率. 不仅可以快速获得准确的测量结果, 还能充分利用测量资源, 允许同时执行更多的测量任务, 并且克服了 DREAM 机制无法处理的“流量突变”问题.

References:

[1] Dai M, Cheng G, Zhou YY. Survey on measurement methods in software-defined networking. Ruan Jian Xue Bao/Journal of Software, 2019, 30(6): 1853–1874 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5832.htm [doi: 10.13328/j.cnki.jos.005832]

[2] Hu ZG, Tian CQ, Du L, Guan XQ, Cao F. Current research and future perspective on IP network performance measurement. Ruan Jian Xue Bao/Journal of Software, 2017, 28(1): 105–134 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5127.htm [doi: 10.13328/j.cnki.jos.005127]

[3] Yu ML, Jose L, Miao R. Software defined traffic measurement with OpenSketch. In: Proc. of the 10th USENIX Conf. on Networked Systems Design and Implementation. Lombard: USENIX Association, 2013. 29–42. [doi: 10.5555/2482626.2482631]

[4] Bosshart P, Daly D, Gibb G, Izzard M, McKeown N, Rexford J, Schlesinger C, Talayco D, Vahdat A, Varghese G, Walker D. P4: Programming protocol-independent packet processors. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87–95. [doi: 10.1145/2656877.2656890]

[5] Qiu K, Yuan J, Zhao J, Wang X, Secci S, Fu XM. FastRule: Efficient flow entry updates for TCAM-based OpenFlow switches. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 484–498. [doi: 10.1109/JSAC.2019.2894235]

[6] Laffranchini P, Rodrigues L, Canini M, Krishnamurthy B. Measurements as first-class artifacts. In: Proc. of the 2009 IEEE Conf. on Computer Communications. Paris: IEEE, 2019. 415–423. [doi: 10.1109/INFOCOM.2019.8737383]

[7] Li FL, Fan GY, Wang XW, Liu SC, Xie K, Sun Q. State-of-the-art survey of intent-based networking. Ruan Jian Xue Bao/Journal of Software, 2020, 31(8): 2574–2587 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6088.htm [doi: 10.13328/j.cnki.jos.006088]

[8] Pang L, Yang CG, Chen DY, Song YB, Guizani M. A survey on intent-driven networks. IEEE Access, 2020, 8: 22862–22873. [doi: 10.1109/ACCESS.2020.2969208]

[9] Kiran M, Pouyoul E, Mercian A, Tierney B, Guok C, Monga I. Enabling intent to configure scientific networks for high performance demands. Future Generation Computer Systems, 2018, 79: 205–214. [doi: 10.1016/j.future.2017.04.020]

[10] Scheid EJ, Machado CC, Franco MF, Dos Santos RL, Pfitscher RP, Schaeffer-Filho AE, Granville LZ. INSpIRE: Integrated NFV-based intent refinement environment. In: Proc. of the 2017 IFIP/IEEE Symp. on Integrated Network and Service Management. Lisbon: IEEE, 2017. 186–194. [doi: 10.23919/INM.2017.7987279]

[11] Tian BC, Zhang XY, Zhai EN, Liu HH, Ye QB, Wang CS, Wu X, Ji ZM, Sang YH, Zhang M, Yu D, Tian C, Zheng HT, Zhao BY. Safely and automatically updating in-network ACL configurations with intent language. In: Proc. of the 2019 ACM Special Interest Group on Data Communication. Beijing: ACM, 2019. 214–226. [doi: 10.1145/3341302.3342088]

[12] Stoenescu R, Popovici M, Negreanu L, Raiciu C. SymNet: Scalable symbolic execution for modern networks. In: Proc. of the 2016 ACM SIGCOMM Conf. Florianopolis: ACM, 2016. 314–327. [doi: 10.1145/2934872.2934881]

[13] Kang JM, Lee J, Nagendra V, Banerjee S. LMS: Label management service for intent-driven cloud management. In: Proc. of the 2017 IFIP/IEEE Symp. on Integrated Network and Service Management. Lisbon: IEEE, 2017. 177–185. [doi: 10.23919/INM.2017.7987278]

[14] Huang Q, Jin X, Lee PPC, Li RH, Tang L, Chen YC, Zhang G. SketchVisor: Robust network measurement for software packet processing. In: Proc. of the 2017 Conf. of the ACM Special Interest Group on Data Communication. Los Angeles: ACM, 2017. 113–126. [doi: 10.1145/3098822.3098831]

[15] Huang Q, Lee PPC, Bao YG. SketchLearn: Relieving user burdens in approximate measurement with automated statistical inference. In: Proc. of the 2018 Conf. of the ACM Special Interest Group on Data Communication. Budapest: ACM, 2018. 576–590. [doi: 10.1145/3230543.3230559]

[16] Yang T, Jiang J, Liu P, Huang Q, Gong JZ, Zhou Y, Miao R, Li XM, Uhlig S. Elastic Sketch: Adaptive and fast network-wide measurements. In: Proc. of the 2018 Conf. of the ACM Special Interest Group on Data Communication. Budapest: ACM, 2018. 561–575. [doi: 10.1145/3230543.3230544]

[17] Gupta A, Harrison R, Canini M, Feamster N, Rexford J, Willinger W. Sonata: Query-driven streaming network telemetry. In: Proc. of the

2018 Conf. of the ACM Special Interest Group on Data Communication. Budapest: ACM, 2018. 357–371. [doi: 10.1145/3230543.3230555]

[18] Moshref M, Yu ML, Govindan R, Vahdat A. DREAM: Dynamic resource allocation for software-defined measurement. In: Proc. of the 2014 ACM Conf. on SIGCOMM. Chicago: ACM, 2014. 419–430. [doi: 10.1145/2619239.2626291]

[19] Moshref M, Yu ML, Govindan R, Vahdat A. SCREAM: Sketch resource allocation for software-defined measurement. In: Proc. of the 11th ACM Conf. on Emerging Networking Experiments and Technologies. Heidelberg: ACM, 2015. 14. [doi: 10.1145/2716281.2836099]

[20] Wang WT, Yang YJ, Wang E. A distributed hierarchical heavy hitter detection method in software-defined networking. IEEE Access, 2019, 7: 55367–55381. [doi: 10.1109/ACCESS.2019.2905526]

附中文参考文献:

[1] 戴冕, 程光, 周余阳. 软件定义网络的测量方法研究. 软件学报, 2019, 30(6): 1853–1874. http://www.jos.org.cn/1000-9825/5832.htm [doi: 10.13328/j.cnki.jos.005832]

[2] 胡治国, 田春岐, 杜亮, 关晓蔷, 曹峰. IP 网络性能测量研究现状和进展. 软件学报, 2017, 28(1): 105–134. http://www.jos.org.cn/1000-9825/5127.htm [doi: 10.13328/j.cnki.jos.005127]

[7] 李福亮, 范广宇, 王兴伟, 刘树成, 谢坤, 孙琼. 基于意图的网络研究综述. 软件学报, 2020, 31(8): 2574–2587. http://www.jos.org.cn/1000-9825/6088.htm [doi: 10.13328/j.cnki.jos.006088]

张宇鑫(1999－), 女, 硕士生, 主要研究领域为网络智能运维, 下一代互联网.

元禹博(1997－), 男, 硕士生, 主要研究领域为网络智能运维.

李福亮(1986－), 男, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为网络智能运维, 下一代互联网.

王兴伟(1968－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为未来互联网, 云计算, 网络空间安全.