可扩展的网络验证技术:研究现状与发展趋势

黄翰林[1],徐 恪[1,2,3*],李 琦[2,3,4],李 彤[5],付松涛[1],高翔宇[4]

(1. 清华大学计算机科学与技术系,北京 100084;2. 北京信息科学与技术国家研究中心,北京 100084;
3. 中关村实验室,北京 100094;4. 清华大学网络科学与网络空间研究院,北京 100084;
5. 数据工程与知识工程教育部重点实验室(中国人民大学),北京 100872)

摘 要: 互联网作为国家信息基础设施的重要组成部分,已经在各个领域发挥着巨大的作用. 随着其规模不断扩大和应用持续深入,我们也面临着意图不一致的网络行为可能导致的灾难性危害. 为了确保互联网的正常运行和网络行为的一致性,我们迫切需要可部署的网络验证技术,以确保网络运行时的行为与网络运维人员的意图一致. 当前已经有许多关于网络验证技术的研究,这些研究帮助用户实现自动检测网络错误,并进一步分析错误产生的原因. 然而,为了满足互联网规模不断扩大的需求,可扩展性问题成为在互联网部署网络验证技术的一项重要挑战. 即如何在满足时间和空间复杂度约束的前提下,快速发现并排查网络策略的错误,真正将网络验证技术应用于实际,成为一个研究热点. 本文从数据面验证和控制面验证两个方面出发,深入研究和总结了现有的网络验证研究工作,并探索了基于时空优化的可扩展性技术,对这些方案的特点进行了系统性分析. 最后,本文总结和展望了网络验证可扩展技术的未来研究趋势,为该领域的研究人员提供一定的参考.

关键词: 网络验证;可扩展性;网络配置;时空优化;数据面验证;控制面验证

基金项目: 国家重点研发计划(No.2022YFB3102300,No.2022YFB3102301);国家自然科学基金(No.61932016,No.62132011,No.62202473,No.U22B2031,No.61825204);北京高校卓越青年科学家计划(No.BJJWZYJH01201910003011)

中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112(2024)04-1083-20

电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20230682

Scalable Network Verification Technologies: State of the Art and Future

HUANG Han-lin[1], XU Ke[1,2,3*], LI Qi[2,3,4], LI Tong[5], FU Song-tao[1], GAO Xiang-yu[4]

(1. *Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084, *China*;
2. *Beijing National Research Center for Information Science and Technology*, *Beijing* 100084, *China*;
3. *Zhongguancun Laboratory*, *Beijing* 100094, *China*;
4. *Institute for Network Science and Cyberspace*, *Tsinghua University*, *Beijing* 100084, *China*;
5. *Key Laboratory of Data Engineering and Knowledge Engineering* (*Renmin University of China*), *Beijing* 100872, *China*)

Abstract: The Internet, as a critical component of a nation's information infrastructure, has played a significant role in various domains. However, as its scale continues to expand and its applications deepen, we also face the potential catastrophic consequences of inconsistent network behaviors. To ensure the normal operation of the Internet and the consistency of network behaviors, there is an urgent need for deployable network verification technologies that align network operations with the intentions of network operators. Extensive research has been conducted on network verification technologies, assisting users in automating the detection of network errors and analyzing their root causes. However, to meet the increasing demands of the expanding Internet, scalability has become a crucial challenge in deploying network verification technologies. Specifically, how to quickly identify and diagnose errors in network policies, while satisfying time and space complexity constraints, has become a research hotspot in effectively applying network verification technologies in practice. To address this problem, this paper delves into and summarizes cutting-edge research on the temporal and spatial scalability of network verification. It begins by introducing the background knowledge related to network verification and then describes the current issues and challenges faced in network verification. Focusing on the core issue of scalability, the paper thoroughly analyzes existing work in achieving scalable verification from both the data plane and control plane perspectives. It

收稿日期:2023-07-21;修回日期:2023-11-18;责任编辑:宋甲英
*通讯作者:徐恪

provides a systematic analysis of the characteristics of these approaches, showcasing the distinctions and connections among related studies. According to the existing researches, we find that: (1) The scalability of data plane verification is primarily constrained by header space and forwarding matching rules, while the scalability of control plane verification is mainly limited by the complexity of multiple protocols and policies. (2) Although both data plane and control plane research employ similar scalable verification techniques, they address different but interconnected targets. For example, incremental computation in the data plane primarily focuses on updating packet equivalence classes, while incremental computation in the control plane primarily deals with network models affected by configuration changes. When applying network slicing techniques, both data plane and control plane independently validate the network by dividing it into multiple segments. (3) Compared to spatial scalability, current research places greater emphasis on temporal scalability, where reducing verification time overhead appears to be the primary pursuit of verification tools. (4) Previous research predominantly adopted a centralized verification approach, which involved collecting control plane or data plane information and then performing centralized analysis and verification. However, there has been a recent trend towards distributed verification, such as Coral and Tulkun in control plane verification. Lastly, based on the current research landscape, the paper concludes by summarizing and forecasting the research trends in scalable network verification technologies, offering valuable insights for researchers in this field. In conclusion, this paper presents a comprehensive review and outlook on the topic of scalability in network verification. It emphasizes the importance of aligning network behaviors with the intentions of network operators to ensure the reliable and consistent operation of the Internet. By addressing the challenges of scalability, researchers can advance the development of network verification technologies that can effectively verify large-scale networks within the constraints of time and space complexity. Ultimately, this contributes to enhancing the reliability and security of the Internet as a critical information infrastructure.

Key words: network verification; scalability; network configuration; time-space optimization; data plane verification; control plane verification

Foundation Item(s): National Key Research and Development Program of China (No.2022YFB3102300, No. 2022YFB3102301); National Natural Science Foundation of China (No. 61932016, No. 62132011, No. 62202473, No.U22B2031, No.61825204); Beijing Outstanding Young Scientist Program (No.BJJWZYJH01201910003011)

1 引言

现代计算机网络部署了大量复杂的功能，涵盖各种路由协议，如BGP（Border Gateway Protocol）、OSPF（Open Shortest Path First）等、有状态防火墙、代理、负载均衡器等，共同维护网络运行时的平稳状态和功能，保证这些功能正确运行给网络运维人员带来了巨大的挑战．随着网络规模不断扩大，网络配置和功能变得复杂、容易出错；网络应用持续深入，涉及生产生活的各个方面，意料之外的错误一旦发生，很可能造成灾难性影响．大量网络中断事件表明，配置错误成为导致网络中断和安全漏洞的主要原因[1~3]．例如，最近的一次BGP错误导致世界上的每个路由器都删除了所有与Facebook有关的路由，全球所有流量无法到达其服务器[4]．

作为检查网络是否满足重要属性的技术框架，网络验证技术为提前发现潜在的网络错误提供了可能．网络验证技术以符号化的形式抽象表达网络的各种语义，通过数学推导或图建模等方法，检查网络行为和网络策略意图是否一致．例如，可达性策略、过滤性策略、隔离性等，保证这些属性的正常运行对网络来说十分重要．

通过对网络语义进行抽象，将编码后的符号或公式输入特定的求解器可以发现策略是否成立，然而这种做法会导致空间状态爆炸的问题，对于一个属性的验证可能会花费长达数个小时，难以满足属性覆盖率高的网络对网络验证的需求[5]．为了提高验证效率，许多研究工作在网络建模和验证阶段对时间和空间开销进行优化，利用网络和属性的特征，从整体到局部，压缩了语义状态空间，然后再输入至网络验证器进行验证．这种优化是极其有效的，可以将普通验证数小时的开销缩短为几分钟甚至几秒，而且在空间上减少内存开销、增强语义表达，实现了时间和空间上的可扩展性．

有研究者也对关于网络验证的研究进行过总结[6~8]．Li等人[6]广泛调研了近年来网络验证和网络测试的工作，介绍网络验证的通用技术如模型检测、符号执行、定理证明等，并探讨了验证和测试的关系；Zhang等人[7]综述了控制面的无状态验证和有状态验证，数据面验证中的基于探测和基于标记的技术；方星等人[8]与Li等人类似，也是从数据面和控制面展开，对验证技术的划分更全面，另外还综述了有状态网络验证的工作．

不同于以上研究者的工作，本文首次针对网络抽象和验证过程的可扩展性技术进行了综述．本文将详细介绍网络验证中可扩展性相关的最新研究进展，研

究如何在满足时间和空间复杂度约束前提下,快速发现并排查大规模网络的潜在错误. 时间上主要验证某个任务消耗的时间短,能服务于大规模网络;空间上主要指内存空间、表达空间的优化,使其实用性更强. 本文从数据面验证和控制面验证两个方面对网络验证的可扩展性技术进行分析,尽管二者的优化方法有共同之处,但是所解决的问题并不相同. 控制面考虑多协议、多策略的路由收敛,协议栈的复杂程度更能制约验证技术的可扩展性;数据面考虑数据包的匹配转发,包头空间更能制约验证技术的可扩展性. 通过对前沿可扩展问题的总结,本文进一步分析未来网络验证可扩展性的发展趋势,并提出可能的技术思路.

2　网络验证背景知识

2.1　网络验证的概念及基础技术

网络验证是通过建模网络系统、分析网络属性,以检验网络行为是否与运维人员意图一致的过程. 图1描述了网络验证的宏观过程. 其中,系统描述包括网络配置、转发路由表、网络拓扑、环境等,待验证策略包括可达性、隔离性等属性;通过网络抽象将系统和策略分别抽象、建模,归一表示,然后采用特定的验证技术,判断该策略在当前网络环境下是否成立,结果正确表明策略成立,否则输出反例. 在此过程中,建模过程和验证过程都会影响网络验证的可扩展性,需要通过一些优化手段才能保证二者的可扩展性,在时空上实现优化,提高验证效率.

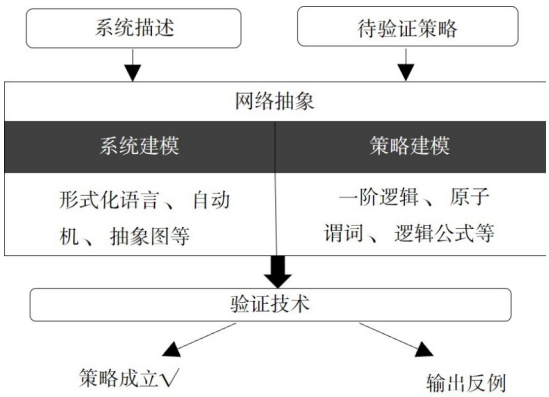


图1　网络验证的流程

抽象建模与验证技术并不是完全独立的,而是具有一定的对应关系,建模的方法决定了验证技术的选择. 例如,使用有限状态机建模系统,则需要将所有状态输入模型检测程序中,判断策略在该状态下是否成立;若按照网络设备和拓扑结构建模系统,则需要运用图算法,根据图的强连接、弱连接等属性,检验网络策略. 表1列出了五种基本的网络验证技术:模型检测、符号执行、图算法、定理证明和布尔可满足性问题(Boolean SATisfiability problem,SAT)/可满足性模态理论(Satisfiability Modulo Theories,SMT)求解.

表1　网络验证基础技术

网络验证技术名称	技术简介
模型检测[9~12]	建立待测试系统的状态模型,穷举网络状态
符号执行[13~16]	程序分析技术,将网络建模为符号表达式和变量
图算法[17~20]	将网络策略或属性转换为图之间的关系属性
定理证明[21~24]	利用公理和推导规则,验证系统的一阶逻辑形式
SAT/SMT求解[18,25~27]	在考虑不同理论约束下,判断是否存在满足其条件的解

(1)模型检测

模型检测[27]是一种形式化验证技术,基于对系统行为进行建模和分析. 可以对数据转发行为或者网络配置进行建模,通常采用有限状态机、时序逻辑或模型检测特定的描述语言(如Promela或TLA+)等模型. 然后,通过自动化的算法和工具,对建模后的系统进行状态空间搜索和性质检查,以发现潜在的策略违反问题. 模型检测十分依赖程序自动行为空间,对每一个状态穷举遍历,当网络规模较大时,极易造成状态爆炸,这也限制了该技术的应用. NetSMC[11]采取的是模型检测验证技术,基于线性时序逻辑建模网络策略语言,验证算法遍历网络状态空间,并且通过对网络语义进行压缩,避免了状态爆炸问题.

(2)符号执行

符号执行[28]是一种静态分析技术,通过对数据面或控制面的程序进行符号化执行,将待验证的属性转化为符号变量和约束条件,一起输入到约束求解器,遍历程序执行路径空间并记录所有约束条件,验证数据转发逻辑是否符合规范等其他策略或输出反例. 符号执行能够发现各种可能的输入条件和路径,但在大规模网络中可能面临路径爆炸和约束求解效率的问题. NICE[15]通过应用事件处理程序的符号执行来执行应用程序的代码路径,以系统地探索网络状态空间(控制器、交换机和主机),在实际应用中有效地找出应用

错误.

(3)图算法

图算法[29]常用于拓扑分析、路径搜索和可达性分析. 工作原理是将网络拓扑和数据平面转发规则等信息建模为图结构,并使用图算法进行分析和搜索. 例如,常用的图算法广度优先搜索(Breadth First Search, BFS)、深度优先搜索(Depth First Search, DFS)、最短路径算法等,可以用于验证连通性、路由正确性和访问控制规则一致性等. 但对于复杂的网络结构,图算法面临着探索缓慢的问题. Tiramisu[5]建模路由邻接图和流量转发图,利用图的邻接、通路等特性验证网络的可达性.

(4)定理证明

定理证明是一种基于数学逻辑和推理规则的验证方法,工作原理是使用形式化的逻辑规则和推理规则,将网络的规约或策略表达为逻辑公式,然后使用自动或交互式的定理证明器进行推理和验证,但是不能提供反例进行调试. 比较代表性工作 VeriCon[22]通过关系代数,将应用程序结构和待验证属性都转化为一阶逻辑表达式,输入进自动定理证明器 Z3 中,验证该属性在当前环境下是否成立.

(5)SAT/SMT求解

布尔可满足性问题(boolean SATisfiability problem, SAT)求解技术用于解决可满足性问题,即判断一个逻辑公式是否存在满足其所有约束的解. 在网络验证中,SAT求解器可以用于处理布尔逻辑公式,其中变量表示系统的状态或控制决策,约束表示系统的性质或规约条件. 可满足性模态理论(Satisfiability Modulo Theories, SMT)求解技术扩展了SAT求解技术,可以处理更丰富的逻辑和约束表达. SMT求解器结合了SAT求解器和具有理论支持的约束求解器(如整数理论、位向量理论等),以处理包含不同理论的混合约束问题,如 Minesweeper[30]使用了 SMT 建模大型网络,描述网络转发在不同路由协议相互作用下的稳定状态. 然而,对于复杂的网络系统和大规模的约束问题,SAT/SMT求解技术可能面临效率和可扩展性的挑战,需要结合其他优化技术和抽象方法来提高求解的效率和准确性.

综上,尽管上述方法可以成功验证网络的属性,但是单纯运用这些技术效率较低,难以满足网络运维的要求. 为此,当前各个验证工作不再仅局限于验证技术,而是更加关注如何在验证过程中利用网络特点对网络优化抽象,增强网络验证的可扩展性和实用性.

2.2 网络验证的详细流程

上述是对网络验证流程和技术的总体概述,为了进一步让读者对网络验证有更深的了解,本文基于一个综合的示例详细阐述网络验证的过程以及各技术在示例的具体应用.

图2(a)呈现了一个由四个路由器和两台主机组成的系统,其中两台主机 X 和 Y 分别与路由器 S_A 和 S_B 相连. 每个路由器都包含有针对特定IP前缀流量和相应出端口的路由表,例如交换机 S_A 转发表项之一规则 R_{A1} 是将目标IP为192.168.11.0/24网段的报文通过接口 P_{a0} 发出去. 我们以可达性策略为例,即验证由主机 X 发出的数据包能否达到主机 Y,分别描述上述各技术的工作原理.

模型检测. 如图2(c),网络系统的状态States由数据包 p 以及所在位置 l 决定,例如初始状态数据包是一个全空间的,对应的位置是主机 X,那么初始状态Initial State可以编码为(p, X). 交换机上数据包的转发规则可以被认为是状态转换State Transitions,例如(p, S_A)→(p', S_B)(p.dst∈192.168.12.0/24, p'.dst=p.dst)表示 S_A 设备上 R_{A2} 的转发逻辑. 当对报文初始状态以及状态之间的转换编码完成后,可以定义属性(property),例如检查是否存在一条路径,使得数据包能够从主机 X 转发到主机 Y. 输入模型和属性到模型检查器引擎(如NuSMV),如果引擎返回pass,则说明该属性在系统模型上成立. 如果模型不满足该属性,系统将返回一个反例. 上面的例子中,模型满足输入的属性.

符号执行. 图2(d)展示了一段伪代码程序,表示了设备接口之间如何转发报文的转发规则. 每次迭代处理一个报文,分为两个步骤:第一,转发数据包到当前链路;第二,将其作为参数转发到与此链路相连的设备. 我们可以定义一个断言声明:定义从主机 X 的数据包转发到主机 Y, P[i].s_Tag 表示原始报文, P[i].d_tag 表示当前位置上的报文. 我们将程序和断言输入到符号执行引擎,该引擎输入一个符号值表示报文,然后分析程序的执行路径,最终会返回断言是否成立.

定理证明. 根据网络拓扑和转发规则构造逻辑公式,以图2(b)为例, link(S_A, P_{a0}, P_{c0}, S_C)可以表示为交换机 S_A 接口 P_{a0} 连接到交换机 S_C 接口 P_{c0}. 输入转发规则,例如 S_A.recv(p.dst∈192.168.11.0/24, P_{a2})表示交换机 S_A 可以在接口 P_{a2} 接受目的IP为192.168.11.0/24数据包 p. 判断来自 X 的数据包是否可以到达 Y,我们可以证明公式 X.sent(p.dst∈0.0.0.0/0)→Y.recv(p.dst∈192.168.11.0/24)公式为真. 但是当不满足公式时,定理证明不能提供反例,所以用户没办法依据反例进行分析和调试.

SAT/SMT求解. 图2(e)表示对设备转发表进行编码, $G(S_A, S_B)$=p.dst∈192.168.12.0/24是一个布尔公式,表示为设备 S_A 到设备 S_B 之间数据报文转发的约束. 然后构造所验证属性对应的逻辑公式,如果想验证报文从主机 X 是否能到主机 Y,将两条路径 X→S_A→S_C→S_D→

Y 和 $X \to S_A \to S_B \to S_C \to S_D \to Y$ 的逻辑表达的公式输入到求解器中进行运算,如果不满足会输出一个反例.

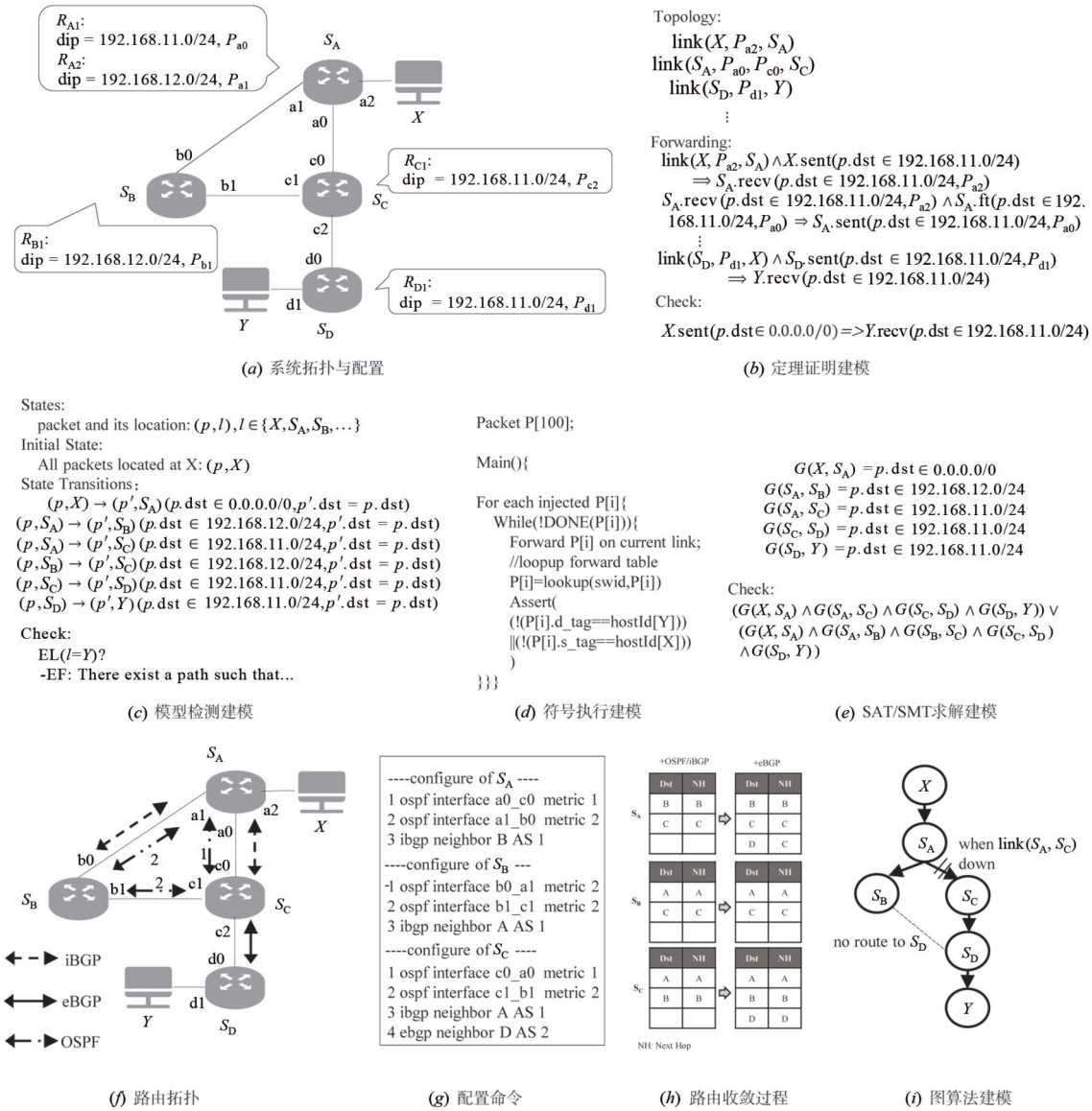


图2　网络验证综合示例

图算法. 在前面的例子中,我们基于转发规则对各个验证技术的工作原理进行了详细阐述,这些验证技术既适用于数据面也适用于控制面,因为控制面本质上也是通过分析网络配置模拟出转发行为. 为了更充分地说明这一点,我们以图2的后四个子图为例展示从网络配置为输入到最终使用图算法验证网络策略的流程. 保持上例的拓扑不变,各设备建立了内部边界网关协议(Internel BGP,iBGP),外部边界网关协议(External BGP,eBGP)和OSPF的路由关系,具体的配置如图2(f)和图2(g),例如SA中的"ospf interface a0_c0 metric 1"表示 S_A 的 $a0$ 接口与 S_C 的 $c0$ 接口建立了OSPF路由,其中metric为1. 假设按照OSPF,iBGP,eBGP的顺序依次建模路由收敛过程,如图2(h)所示,在添加iBGP时并不会改变各个路由表,但是当添加eBGP协议后 S_A 则获得了"Dst:D,NH:C"这一到 S_D 的路由,S_C 也获得了到邻居 S_A 的路由,但是由于从eBGP对等体获得的BGP路由,BGP设备发布给它所有eBGP和iBGP对等体,因此 S_B 没有到 S_D 的路由. 我们根据收敛之后的路由表构建了从 X 到 Y 的有向图,如图2(i)所示. 不难看出存在一条 $X \to S_A \to S_C \to S_D \to Y$ 的可达路径. 但是一旦link(S_A,S_C)发生故障,X 则无法达到 Y,因为 S_B 无法获得到 S_D 的路由.

尽管上述验证流程已经能完整地实现网络验证,但是前文提到,仅靠基础的验证技术是不够的,例如模型检测会遭受到状态爆炸、符号执行面临路径爆炸和

约束求解效率的问题、图算法难以表示大规模网络场景以及在建模复杂系统面临开销大的问题等. 此时就需要优化建模和验证的过程,针对简化后的待验证网络,再利用这些验证技术,可取得事半功倍的效果. 例如根据状态依赖性,在检查 X 到 Y 的可达性时,可以提前剪枝掉与 S_B 相关的状态,按照类似的思想,可以在建模中删除掉许多无关状态和变量,使时间和空间上得到优化;再比如对于图的多协议交互路由问题,则需要更强的规范语义准确表达出协议的行为,避免给模型精度带来损失. 优化系统验证过程、减小开销的技术是本文关注的重点. 根据待分析网络的特点,选择不同的优化技术,不仅可以加速完成策略验证的过程,还能减少状态变量节省内存空间.

2.3 网络验证的可扩展性

2.3.1 定义与评价

本文对网络验证可扩展性的关注点集中在时间和空间两个维度,如表2所示. 时间指的是验证系统完成验证整个过程花费的时间,而空间上本文分别从内存空间和表达空间来切入,探究验证系统的内存需求、利用率,以及对网络协议、功能建模的表达能力. 我们分别就时间、内存空间和表达空间详细展开描述.

(1)时间. (a)可以通过验证处理时间的增长率来评估. 如果验证处理时间随着网络规模的增加呈线性增长,那么可以认为其可扩展性良好,如图3中的B区域. 而对于C区域中变缓的增长率,则认为其可扩展性较好. 对于A区域中逐渐激增的增长率,则认为其可扩展性较差. (b)可以通过并行处理的能力来评估. 如果验证过程能够并行处理多个验证任务,以提高效率和减少验证时间,那么可以认为验证在时间上是可扩展的. 例如,Libra[31]将网络划分为多个切片,并行分析每个切片的转发图,以检测路由故障.

表2 时空可扩展含义

角度		评价标准
时间		处理时间增长率、并行处理能力、灵活和可配置性
空间	内存空间	内存需求增长率、内存资源利用率
	表达空间	协议表达能力、功能表达能力

(2)内存空间. (a)可以通过内存需求的增长率来评估,与上述时间可扩展的评估类似. (b)可以通过内存复用的能力来评估. 如果验证器(在验证多个任务时)能共享状态、数据,降低网络验证过程中的内存消耗,那么认为其是可扩展的. 例如,APKeep[32]为了解决设备异构性问题,提取设备之间相同的功能集合,在功能级别上使用单元(因特网协议(Internet Protocol,IP)转发、访问控制列表(Access Control List,ACL)等)为每个设备建模,避免了为每个设备都编写一个模型.

(3)表达空间. (a)可以通过验证系统是否具备多协议和多属性(功能)验证能力来评估. 例如,Tiramisu[5]能够捕获不同协议(如BGP和OSPF等)之间的依赖关系,完成可达性、路径偏好、最小割等属性的验证. (b)可以通过验证系统是否具备高级逻辑建模语言来评估. 如果验证器能通过高级声明式语言隐藏底层的细节,支持用户在程序里定义网络需要满足的行为,那么认为其是可扩展的. 例如,VeriCon[22]使用编程语言CoreSDN和语义模型,将网络控制集中化并与网络硬件分离,使网络管理员可以在控制器上运行程序来指定转发规则、访问控制策略等,而无需直接与网络硬件或其他软件定义网络(Software Defined Network,SDN)程序进行交互.

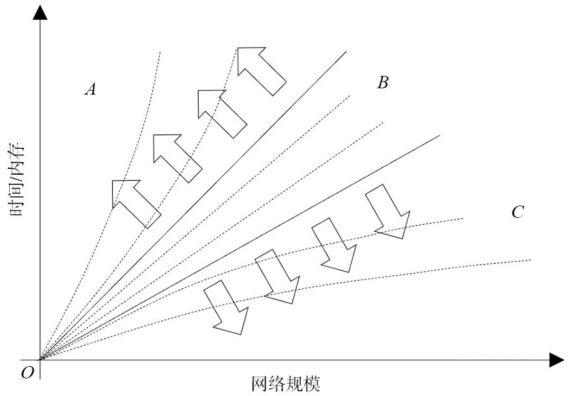


图3 时间(内存)随网络规模的变化趋势

2.3.2 影响因素

在网络验证可扩展性方面,存在众多因素会对建模和验证过程产生影响. 本文对常见的7种因素进行了总结,并进行了简要的分析.

(1)策略的复杂性. 需要验证的网络策略繁多,对于验证系统来讲需要更多的时间来检查这些策略的正确性和一致性. 例如,对于模型检测,当需要同时验证的策略过多时,需要编码大量网络状态,导致有限状态机爆炸.

(2)转发和路由规则的复杂性. 如果转发表中包含大量的条目和复杂的规则,涉及规则之间相互作用导致网络的状态空间庞大,系统需要更多的内存来完成验证. 例如,其会导致符号执行技术生成的路径约束数量急剧增加,给系统内存带来压力.

(3)协议的交互性. 一个路由器上的路由协议可能包含OSPF,BGP,中间系统-中间系统(Intermediate System-Intermediate System,IS-IS)[33]等,报文在路由器上的转发是由多个路由协议根据路由偏好、代价等因素共同决定的,要求验证系统能够准确建模各个协议的决策. 例如,在策略建模阶段,某个策略会受到多种协议影响,SAT/SMT求解需要考虑各种协议的可满足属性,增加可满足性问题的复杂性.

(4)网络的动态性. 控制平面或者数据平面更新过快会导致“更新风暴”,验证工具需要能够及时捕捉这些变化并进行更新. 例如,路由表频繁更新、路径选择调整时,需要不断重新产生数据面快照输入上述的技术中,增加了验证的计算负担.

(5)拓扑结构的差异性. 扁平的拓扑结构相对简单,而具有层次结构或多级结构的网络可能需要更复杂的验证方法. 例如,在树状拓扑结构中,需要构建的节点数与影响待验证策略所在的层级数目成指数增长.

(6)设备的异构性. 网络设备行为模型的正确性受到特定厂商行为的影响,难以准确、统一地表达各设备的行为. 例如,在系统建模阶段,不同厂商以不同的方式实现协议的部分功能,难以用统一的语言建模各个功能状态.

(7)网络环境的复杂性. 除了拓扑、协议外,还需要准确建模网络配置、链路状况等信息. 例如,在系统和策略建模阶段,需要手动分析网络中的各种协议、功能、故障、安全隐私等因素并准确建模各因素的综合系统,这对运维人员是不小的挑战.

3　问题与挑战

根据第2.1节的阐述,网络验证过程主要包括两个核心环节:建模(包括系统建模与策略建模)和验证. 在前述部分,本文概括性地阐述了影响可扩展验证的各种因素(如策略复杂、协议交互等). 这些因素之所以会对网络验证的时空扩展性产生影响,主要在于它们为建模和验证过程带来了挑战,增加了相应的开销. 而随着网络规模的增加,这种挑战也会进一步加剧. 我们分别从数据面验证和控制面验证两个角度进行分析.

3.1　数据面可扩展验证挑战

在建模过程中,数据面涉及多个网络设备、协议和功能,其中包括路由、交换、转发、过滤等. 建模这些复杂的数据面功能和行为需要理解各种协议规范和设备特性,并将其准确地抽象和表示为可验证的模型. 而且需要在精确性和抽象度之间寻找平衡,过于详细的模型可能会导致复杂性和计算开销的增加,而过于抽象的模型可能会忽略关键细节导致验证精度降低.

在验证过程,需要对输入流量进行匹配和处理,以验证网络设备的正确性和性能. 随着流量规模和复杂性的增加,流量匹配和处理所需的时间也会增加,并且巨大的包头空间对验证所需的内存也是不小的挑战. 这是由于网络数据流包含大量的数据包,如果单独分析每个数据包的转发行为,会出现大量的可能状态组合,在没有做优化处理的情况下,对于上述技术来说会导致状态空间呈指数级增长,例如根据将数据包按位表示为<0,1,x>的集合,其状态空间为3^n,其中n表示包头长度. 网络数据面是动态的,包括链路状态的变化、流量负载的波动和实时数据包的传输,验证工具需要能够在短时间内处理和验证实时的动态行为,及时发现潜在问题.

3.2　控制面可扩展验证挑战

在建模过程中,验证系统需要收集配置信息,模拟出数据面状态或网络行为,因此准确表达出各配置、各策略的行为规范对推断出数据面状态和建模至关重要. 网络存在多种控制平面协议和策略,如OpenFlow,BGP,OSPF等,需要一定时间达到收敛状态,协议的复杂程度也会影响建模的时间.

在验证过程中,与数据面类似,控制面的状态空间往往也非常庞大,因为它涉及多个控制器和网络设备之间的状态和交互. 验证过程需要处理庞大的状态空间,以确保对所有可能的状态和交互进行全面的验证,对存储和维护大规模状态空间所需的内存消耗是一个挑战. 并且配置更新也会导致网络控制面发生改变,如果重新模拟出完整的数据面并验证,无疑加剧了整个过程的开销.

为了解决这些挑战,需要采用合适的建模和验证优化技术,通过压缩状态空间、改进算法和数据结构以及增强语义规则的建模能力,在时间和空间上增强网络验证的可扩展性,以提高验证的效率和准确性. 下面将从数据面验证和控制面验证两个方面分析现有工作是如何在满足时空可扩展的前提下完成属性、策略验证的.

4　现有研究现状与分析

本节我们首先介绍了网络验证可扩展技术及其在数据面和控制面对应的优化思想,然后分别从数据面和控制面分别深入分析每个研究工作是如何利用相应的可扩展思想完成验证的. 尽管数据面和控制面会用到相同的可扩展技术,但是它们所解决的问题并不相同.

4.1　可扩展验证技术的分类

通过对现有网络验证工作的总结与分析,本文整理了当前最具代表性的网络验证面对时间和空间可扩展性问题的解决方案,如表3所示. 其中,每种可扩展技术又分别应用在数据面验证、控制面验证方面,或者两方面均有应用. 我们列举了每种技术在数据面验证和控制面验证上的联系与区别.

数据面和控制面对可扩展性的影响因素存在差异性,如数据面验证考虑数据包的匹配转发,包头空间制约验证技术的可扩展性;而控制面验证考虑多协议、多策略的路由收敛,协议栈的复杂程度能制约验证技术的可扩展性. 对此,我们将分别从数据面和控制面两个角度,具体分析现有技术是如何在时间和空间上解决可扩展问题的.

表3 可扩展验证技术

可扩展验证技术	数据面	控制面
基于对称性	利用网络的对称结构对网络进行压缩	—
基于网络切片	基于网络的模块化结构的切割方法,将网络切割成多个片段进行独立验证	与数据面类似
基于增量等价类/计算	数据面验证的增量计算主要体现在对数据包等价类的增量计算,对受影响的等价类进行更新	控制面更关注网络配置更新情况,通过只更新受配置变化影响的系统模型部分,或只验证受配置变换影响的网络策略
基于状态依赖性	根据策略和流量的交互逻辑,检测内容依赖式的策略,减少状态的数量	利用属性依赖关系进行拓扑剪枝和建立约束,减小搜索空间
基于层次分解	—	将编码与验证算法分离,针对不同类别的策略使用不同的验证算法
基于逻辑编程语言	声明式规范语言用以描述网络服务和验证策略	与数据面相同,用高级的编程语言,简单地表达出复杂的网络策略
基于分布式控制	一种分布式、设备上的数据平面验证(Data Plane Verification,DPV)框架,通过将DPV转化为基于有向无环图(Directed Acyclic Graph,DAG)上的计数问题	—
基于转发相似性	网络中许多不同的数据包共享相同的转发行为,据此建立等价类	与数据面类似
基于FASTPLANE模拟	—	利用FASTPLANE的路由模拟计算加速技术,选择合理的路由传播次序,加快收敛速度

4.2 数据面可扩展验证技术

4.2.1 基于对称性

(1)相关工作

Plotkin等人[34]首先提出对无状态网络数据面进行验证的SVU,利用网络对称性压缩网络规模,将功能相同的设备合并为一个整体. 他们指出,数据中心网络多是“胖树”的层次结构[35],特定层级的路由器对称连接到相邻层级的路由器. 如果对称位置的路由器互换,那么它们的端口以及相邻层级路由器的端口也会互换,全局拓扑结构仍然保持不变. 因此可以将对称路由器替换为一个单独的等价转发路由器,“胖树”的结构也自然转变为“瘦树”. 例如在图4中,R_3,R_4被对称放置:因为如果它们互换位置,它们的端口和它们邻居R_1,R_2,R_5的端口也相对互换,那么总体的拓扑结构仍然保持不变. 因此通过设计,可以期望在给定级别(例如R_3和R_4)对称放置路由器的规则集也是对称的,这意味着在转发规则上,R_3和R_4可以被等价的路由器$\overline{R}_3$所替代,同时不会影响网络的属性. 文章还提供了实验证据,表明他们的网络转换方法可以将验证大型数据中心网络中所有虚拟机之间通信的任务加速65倍. 原本需要5.5天的全对之间可达性计算可以在2 h内完成,并且可以轻松并行化以在几分钟内完成.

利用与SVU相似的思想,Panda等人[24]的研究仍然聚焦在数据中心网络,不同点在于其提出的VMN是针对包含多个中间盒的易变网络数据面,通过合并中间盒减少需要分析的网络状态变量. 他们认为在大网尤其是在易变网络中,都有很高的对称性. 他们通过使用对称性,减少验证整个网络需要的变量的数量,使整个

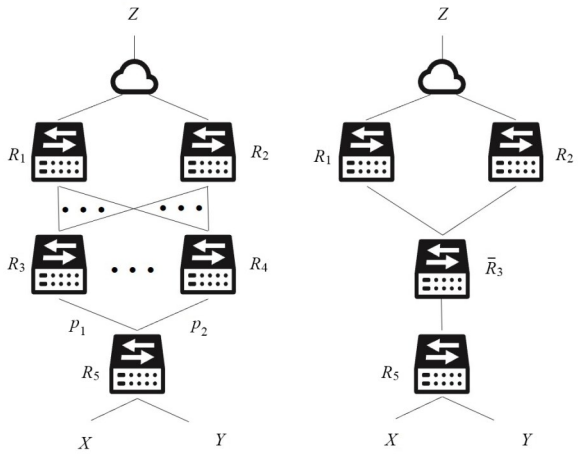


图4 “胖树”到“瘦树”的转化[34]

具有易变网络功能的验证变得高效可行. 该工作的不足是,抽离了中间盒的绝大部分功能,只关心中间盒的转发功能.

(2)区别与联系

SVU利用数据中心架构的特性,删除与待验证属性无关的结构,有效压缩了网络规模,减少了验证整个网络所需的时间. 但是该方法对网络结构特性要求较高,只限于“胖树”的网络结构,对于对称性较弱的网络不能很好地扩展,而且尽管已经抽离了与属性无关的拓扑结构,但是对于依赖性策略,仍然会有许多状态变量影响验证效率. 除了数据中心,许多大网也存在对称拓扑和对称协议,VMN通过合并易变网络的中间盒减少需要验证的变量空间. 基于对称性的抽象保留了许多网络属性,将具体的网络抽象为具有等价行为的抽

象网络,然后再输入到特定的验证工具中,到达快速验证的目的.

4.2.2　基于网络切片

(1)相关工作

Kazemian等人[14]提出HSA,最先通过对包头信息进行分析,为每个独立的网络空间执行静态检查的工作.HSA采用了网络切片的思想,每一个切片由一个三元组构成:切片网络空间,操作权限,切片转移功能.每个切片都有自己的控制平面,允许其拥有者决定如何路由转发和处理数据包.网络运营商通常希望控制哪组主机(或用户)可以相互通信,将每个切片看作独立自治的管理单元,更好地保证终端、用户或者流量的隔离性.尽管HSA通过切片的技术缩小了网络规模,但是当网络的控制依赖关系较强时,每个切片规模会随之变大,很难实现高效验证.

与HSA不同,Zeng等人[31]提出的Libra仅对转发规则进行切片,对每个切片采用MapReduce[36]技术完成验证.如图5所示,从构成整个转发图的交换机开始,每个交换机都有自己的前缀转发表,分两个阶段进行.首先,在Map阶段,它将图划分为一系列切片,每个前缀地址为一个切片,切片里面包含转发包到特定子网地址的转发规则和交换机;在Reduce阶段,Libra独立地分析每个切片,将其作为转发图,并行处理路由失败的情况.然后每个子网单独建立转发图,可达性检查被表达为从网络中的任何交换机都可以到达该切片子网,循环检测可以表达为切片图中是否存在一个强连接.Libra可以在不到一分钟的时间内分析包含10 000个交换机的网络,运行时间随网络规模线速增长.即使是输入最大的数据集DCN-G[31](大约100万个节点和1 000万条边),也只消耗412 MB的内存.但是Libra基于转发规则的静态分析,不能预测网络运行时的动态行为,无法检查出所有的配置错误.

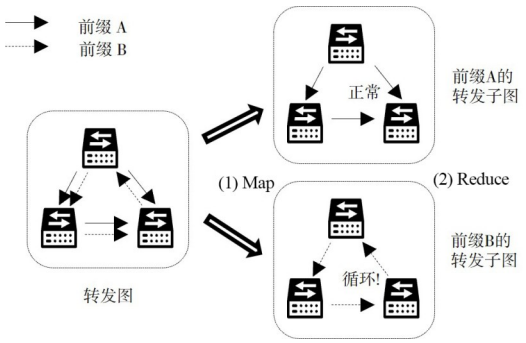


图5　Libra的MapReduce示意图[31]

Guo等人[37]2022年提出了Flash,为每个数据面快照生成一个子空间验证器,同时结合分治思想,有效降低了计算开销.子空间由模型管理和一致性验证器两部分组成:模型管理基于收到的路由更新维护数据面快照,一致性验证器使用一致性检测算法维护验证图.子空间验证器提供与原始数据面验证器相同的功能,接受路由转发信息的更新,重构数据面状态,验证属性是否成立.最后,将各分治子空间聚合,达到全局验证的目的.在验证同样的网络LNet-smr下,Flash消耗了15 MB,而Delta-net[38]需要6 792 MB内存.

(2)区别与联系

HSA和Flash都是基于拓扑单元进行子网划分,无需模拟整个数据面,更快地对属性进行验证如隔离性、可达性;不同的是,Libra是对转发规则进行划分的,在网络切片技术的基础上,通过分治策略不仅达到了小范围网络快速验证的目的,而且只存储切片的转发图,节约了内存空间,无需保存所有的网络状态.

4.2.3　基于状态依赖性

(1)相关工作

在无状态网络中检查策略是否正确执行是静态的过程,回答可达性问题是相对简单的.然而事实上,网络管理员往往在有状态网络中实现了许多内容依赖式的策略.例如,防火墙检测到某源端发起的连接数目超过一定数量时,会给该源端打上标签,后续来自该源端的数据包会被送往异常检测系统.在这种情况下,检测内容依赖式的有状态策略是非常有挑战性的.

Fayaz等人[39]针对该问题设计了BUZZ,这是一项关于网络中测试依赖上下文的策略的研究,利用新的高级流量单元(Buzz Data Unit,BDU)概念和模拟复杂的网络功能为有限状态机集合,提供了可扩展性的数据平面模型.为了建模可追溯模型,其分解每个网络功能内逻辑独立的任务或者流量单元,创造一个有限状态机(Finite State Machine,FSM)表征整体,而不是一个巨大单一的FSM.与之前针对IP报头,在头部空间进行搜索的工作[31,40,41]不同,BUZZ需要对所有可能的流量单元序列进行搜索,BDU通过聚合提高了可扩展性.但是BUZZ需要为每个流量单元都生成测试流量,由于测试间的干扰,并行化运行所有测试容易产生不正确的结果.

Zhang等人[42]于2020年提出了Gravel,根据规则依赖执行状态转移,验证完整的中间盒属性.与VMN把中间盒仅抽象为转发模型不同,Gravel关注的是验证中间盒本身程序实现的正确性,例如是否能够在流量异常之后做区分以发现程序实现的bug等,包含了中间盒负责的所有功能.当一个包到来时,分类器使用依赖规则执行状态转移,直到状态机达到某个最终状态,状态转换表的内存区域的内容在程序执行期间保持不变,每个元素或模块执行的操作数量是有限且较小的.Gravel可以通过最少的代码更改避免在现有中间盒中发现的错误,并且给中间盒带来的延迟仅在15 μs左

右. 尽管Gravel实现了对中间盒功能的全面验证,但是其目前仅支持基于Click体系结构的中间盒,当今涌现的基于P4架构[43]、SONiC架构[44]的中间盒往往对每个包处理操作复杂,Gravel验证难以适用.

(2)区别与联系

BUZZ和Gravel均是将复杂的网络功能建模为有限状态机,根据策略和流量的交互逻辑,检测内容依赖式的策略,执行状态转移和剪枝,减少策略不相干的状态数量,缩小了寻找测试流量的范围空间,时间和空间效率都有所提升.

4.2.4 基于转发相似性

(1)相关工作

第一个利用该思想对SDN网络进行验证的是VeriFlow,2013年由Khurshid等人[41]提出. VeriFlow突破了传统方法的限制,利用数据包行为相似性对网络进行划分,动态跟踪规则的改变,成功将验证时间降低到毫秒级别. 图6展示了VeriFlow位于SDN应用程序和设备之间感知并检查新插入的规则. 首先为了在几秒内对规则的插入或删除进行快速验证,VeriFlow根据新的规则和交叉的规则,把网络分成一组数据包等价类(Packet Equivalence Class,PEC)集合,每个等价类是一组经历相同转发行为的数据包. 等价类划分完毕后,Khurshid等人注意到每个规则的改变,只会影响少数几个等价类,从而主要关注验证那些受影响PEC的不变量.

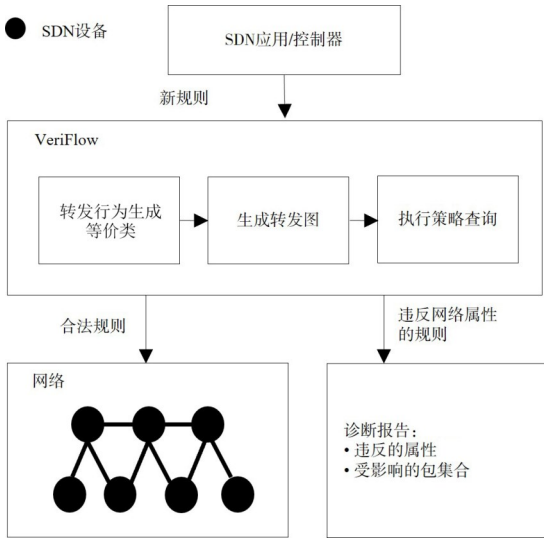


图6 VeriFlow感知并检查每个进入网络的规则[41]

Yang等人[45]首次提出了新的形式化验证工具APV,由原子谓语呈现完整谓语的表达,实现SDN网络的实时验证,其中原子谓语表示一类具有相同转发行为的数据包集合. 每个表示网络功能的谓语,都可以分解为许多原子谓语的组合,两个谓语的计算可以被作为两组整数集合的运算. 通过对谓语的计算来判断网络策略的正确性,例如转发谓语的正误决定数据包是否被转发.

(2)区别与联系

VeriFlow用于实时验证网络属性,该方法通过在SDN中使用增量计算实时观测数据平面,及时在规则发生改变时检测网络内的策略违反现象. 而APV则利用原子谓词的概念来加速数据包等价类的计算. 此外,APV还特别关注实时性,能够处理网络中动态的变化并实时验证网络属性的符合性.

4.2.5 基于增量等价类

(1)相关工作

基于APV提出的原子谓语(Atomic Predicates,AP),Wang等人[46]提出了APC,APC利用原子谓语构建了AP树,对一个谓语从上到下拆分,叶子节点代表原子谓语,表示具有相同转发行为数据包的等价类,一个数据包会被分配到对应的原子谓语中. 如果网络规则过多,那么构建的AP树的深度也会增加,这对验证的效率是一个挑战. APC通过增量计算的方法只对受状态变化影响的节点进行更新. 不仅降低了树的深度,而且减少了需要存储的原子谓语. APC在一般计算机上只使用几兆字节的内存,每秒支持超过两百万个查询,并且能够在实时更新中实现很低的延迟.

Delta-net[38]是一种实时网络验证方法,旨在自动检测数据平面中网络范围可达性不变式的违规情况. 该系统使用数字区间建模数据包头集合,每个区间匹配具有相同转发行为的数据包,可以增量地维护网络中所有数据包流的紧凑表示,从而支持更广泛的场景和查询. 通过使用实际部署的软件定义网络应用程序(SDN-IP)和来自实际部署网络的拓扑和BGP更新生成的数亿个IP前缀规则进行实验. 实验结果表明,Delta-net平均在约40 μs内检查规则的插入或删除,相比VeriFlow效率提高了10倍以上.

Zhang等人[32]于2020年提出的APKeep旨在解决实际网络中的验证问题. 基于一个支持增量计算的模块化网络模型(Port-Predicate Map,PPM),使用PPM技术构建有向图,通过符号标识的等价类表示图中边的信息. 该模型能够更准确地表达实际设备的功能,当数据面变化时对PPM增量更新,将网络行为相同但不属于同一等价类的进行合并,有效减少了等价类数量. APKeep支持对包括IP转发、ACL、网络地址转换(Network Address Translation,NAT)、基于策略的路由等的验证,并提供了相应的算法来实现低内存占用和快速更新速度. 实验展示了APKeep可以在大多数情况下在1 μs以内验证NAT规则的更新,能够随NAT规则的数量可扩展,保证验证器追赶上网络的平均更新速度(如每秒一次设备更新).

KATRA[47]构建了一个分层网络的增量验证系统,解决跨层依赖性问题. 它指出网络的分层设计在一定程度上促成了网络的成功,但也给网络的可靠运行带来了挑战,为了确保多层网络的安全运行,Beckett等人[47]开发了一种用于具有由头部堆栈组成的数据包的网络的验证算法. Beckett等人[47]从协议层次分解,将包头协议栈(如IP-in-IP[41]、虚拟扩展本地局域网(Virtual eXtensible Local Area Network, VXLAN)[48], IP通用路由封装协议(IP Generic Routing Encapsulation, IP GRE)[49]等)的分解表示作为一组集合,部分等价类捕获在网络中的每个节点具有相同转发行为的数据包集,增量计算部分等价类的最小集合. 与基于报头重复的解决方案APKeep相比,对于两层协议,KATRA的验证速度达到了它的5~12倍,这种速度优势随着网络规模和层次数量的增加而增加.

(2)区别与联系

上述均是采用了增量等价类的数据面验证可扩展技术,它们均是先基于数据包转发行为建立等价类,然后对等价类进行增量更新的. 其中,APC都是基于树结构增量计算受到数据平面变化影响的等价类;Delta-net采用数字区间运算进行增量计算受影响的数据包集合;APKeep提出了一个增量计算模型PPM,对等价类进行增量更新;而KATRA则是基于协议栈分解增量验证使用数据包封装实现的任意分层网络数据平面.

4.2.6　基于逻辑编程语言

(1)相关工作

Renganathan等人[50]提出一种名为Hydra的系统,利用运行时验证的思想来实时检查网络中每个数据包是否按照规范正确处理. 他们提出了一种用于编写属性的领域特定语言Indus,并开发了一个编译器,将指定的属性转换为可在转发时线速速率运行的可执行P4代码. 通过使用Indus对各种属性进行建模,他们评估了所提方法,并展示了它足够表达先前研究中的示例. 如在具有源路由的数据中心网络中实现路径验证,确保数据包遵循"valley-free"路径,使用Hydra来检测Aether程序过滤实现中的一个细微错误.

4.2.7　基于分布式控制

(1)相关工作

Xiang等人[51]提出了一种分布式、设备上的数据面验证(DPV)框架Coral,通过将DPV转化为基于有向无环图(DAG)上的计数问题,并在网络设备上执行轻量级任务来实现可扩展性. 评估结果显示,该框架的原型在各种设置下实现了可扩展的DPV,并且对普通网络设备的开销很小.

Tulkun[20]是Coral的完整扩展版本,基于分布式、设备内验证的方法. Xiang等人[20]提出一种分布式验证消息传递协议,用于指定设备内验证器如何高效地通信任务结果以共同验证不变式,将计数问题自然地分解为在网络设备上执行的轻量级任务,从而实现在各种规模和类型的网络中快速进行数据平面检查. 实验证明,Tulkun在真实数据集(广域网/局域网/数据中心)上可以在41 s内验证一个真实的大型数据中心,而其他工具需要数分钟甚至数十小时,并且在商用网络设备上增量验证的80%分位数提速高达2 355倍.

(2)区别与联系

Coral是迈向分布式设备执行数据平面验证的第一步,Tulkun则完整回答了Coral未解决的如何有效地指定和验证通用不变量、如何通过数据包转换来验证数据平面、如何最小化设备间的信息交换以减少开销以及如何有效地验证不变量的容错性. 尽管前面提到的Libra通过划分子网实现并行验证,但它仍然是使用中心化的设计使用集中式服务器从每个网络设备收集数据平面并验证不变量.

4.2.8　分析与总结

本节主要介绍了数据面验证工作的可扩展性技术,分析了各研究工作是如何利用对应技术减小数据面状态空间,描述了它们在各自验证场景下具体时空性能的表现. 我们也对同一种优化技术下的相关工作做了联系和区分,这是由于尽管一些研究工作用了相同的优化方法,但是它们的针对对象可能是不同的,例如基于网络切片的HSA和Flash,前者针对拓扑单元,而后者针对转发规则进行划分.

值得注意的是,增量技术被广泛应用于数据平面验证工具,要么针对等价类进行增量计算,要么直接增量计算受影响的数据包集合. 分布式验证是一个非常新颖的可扩展技术,旨在解决现有中心化架构在可扩展性方面的挑战,对未来验证技术的发展具有参考意义. 表4总结了每种可扩展技术对应的数据面验证相关工作,并且将每个工作在每一时空维度上的可扩展性分为"高中低"三个层次,展现不同研究工作的可扩展能力,"—"表示在该维度没有体现相应的可扩展性.

4.3　控制面可扩展验证技术

4.3.1　基于网络切片

(1)相关工作

Batfish[52]通过分析网络配置文件模拟数据面的生成,尽管可以动态地检测各个属性策略,但是模拟带来的开销是巨大的. 为了解决这个问题,ARC[16]对局部网络进行抽象,为每对源和目的节点组合的子网构建一个转发图,每当控制面配置发生改变时,仅针对受影响的子网进行分析,支持在任意故障情况下对关键属性快速验证. 但是ARC需要为每个子网都建立转发模型,即使链路的故障仅影响少部分网络,这也会带来不

表4 数据面可扩展验证相关工作总结

可扩展性技术	相关研究	时间	内存空间	表达空间
基于对称性	SVU[34]	中	—	—
	VMN[24]	低	—	—
基于网络切片	HSA[14]	低	—	—
	Libra[31]	高	中	—
	Flash[37]	中	高	—
基于增量等价类	Delta-net[38]	中	—	—
	APKeep[32]	高	高	中
	VeriFlow[41]	中	—	—
	APV[45]	高	低	—
	APC[46]	高	高	—
基于状态依赖性	BUZZ[39]	高	中	—
	Gravel[42]	低	—	—
基于层次分解	KATRA[47]	高	—	高
基于逻辑编程语言	Hydra[50]	中	—	高
基于分布式控制	Tulkun[20]	高	—	高
	Coral[51]	中	—	—

必要的冗余计算.

ERA[18]使用二进制决策图(Binary Decision Diagrams,BDD)[53]压缩控制面消息的集合,基于图建模丰富的路径解决协议交互问题. 在控制面中两个路由公告会交互,而且代表图网络的每个顶点和边需要编码大量的路由交互逻辑,默认的编码方式会产生大量公式和约束,给验证带来不可忽视的挑战. 为了减少变量和约束的数量,Minesweeper[30]用基于IP地址的网络切片技术移除不影响最终结果的网络变量和约束,实现了高网络设计覆盖率和数据面覆盖率. Minesweeper通过使用SMT约束建模网络和执行验证,但是牺牲了一部分正确性和表达性. 这是由于它忽略了在收敛之前网络暂态的属性而造成错误判断,例如临时的路由环路或路由撤销期间的黑洞;表达性是由于Minesweeper全局使用逻辑公式建模的方式在编码某些协议行为上十分低效,例如建模iBGP模型时需要创造n个网络副本,其中,n是运行iBGP协议的路由器数量. Bonsai[54]是一种利用切片技术将大型网络压缩为具有相似控制平面行为的小型网络的算法. 例如,研究人员成功将一个196个节点的操作性数据中心网络压缩到26个节点,并将边的数量减少了大约99%. 一种使用eBGP,iBGP,OSPF和静态路由的1 086个节点的广域网被压缩到137个节点,并将边的数量减少了7倍.

Thijm等人[55]介绍了一种基于SMT的分布式控制平面路由行为分析方法Kirigami. 为了提高SMT求解的可扩展性,他们引入了一种模块化的网络控制平面验证方法,将网络切割成较小的片段. 用户可以指定一个带注释的切割方式来生成这些片段,然后独立地对每个片段进行验证. 结果显示,使用Kirigami进行端到端的验证时间提高了10倍,SMT求解时间提高了6个数量级. LIGHTYEAR[17]是一种基于模块化的方法来验证边界网关协议(Border Gateway Protocol,BGP)控制平面的工具. 它的模块化方法使得定位配置错误变得容易,并且支持增量重新验证. 相比传统的方法,LIGHTYEAR在可扩展性方面有显著改进,并且已经被成功应用于大型实际网络的属性验证.

(2)区别与联系

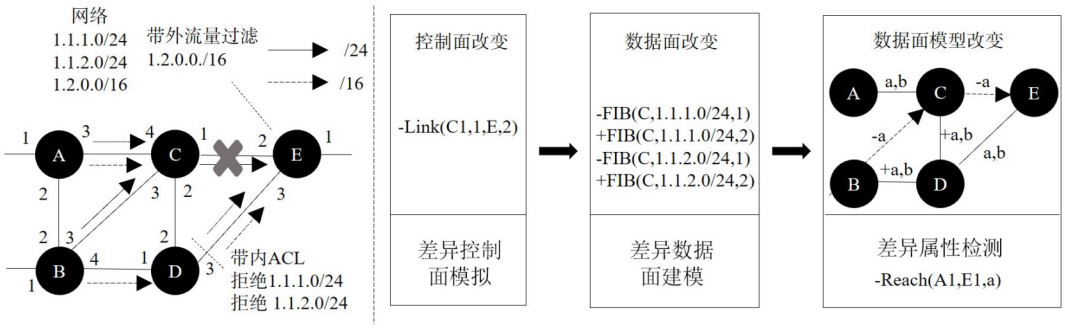
与HSA和Flash类似,ARC和Kirigami也基于拓扑单元进行子网划分,独立地对每个片段进行验证,提高了验证工具的求解速度;Minesweeper基于IP地址空间进行划分,开发了一套模型切片和提升优化技术,可将大型网络的验证时间减少多达99.8%,但是另一方面建模过程需要大量的SMT公式和变量,也会影响可扩展性;LIGHTYEAR根据用户提供的本地约束,以捕获配置的模块化结构,通过对单个节点和边缘节点进行本地检查来验证端到端网络属性.

4.3.2 基于增量计算

(1)相关工作

Zhang等人[56]于2020年提出了增量网络配置验证INCV,计算受配置改变后状态的有效性,识别配置改变后,之前的那些路由转发等状态仍然是有效的,仅仅重复计算受配置改变所影响的部分. 但是如果想将INCV成功利用起来,就必须将其与现有的配置验证器相结合,需要增强或重新设计现有的验证工具,特别是模型检查器、约束求解器等需要确定哪些之前的验证结果在配置更新后仍然有效.

继INCV之后,Zhang等人[57]于2022年提出了DNA,逐步识别配置变化引起的行为差异,成功将增量配置计算应用于实际的验证过程中. DNA使用基于differential datalog(DDlog)的控制平面模型,由一组“事实”和“规则”组成,只关注如何派生路由“事实”,而不关心连接、映射或过滤路由的顺序,基于路由协议的配置、网络拓扑和外部路由,输出包含派生“事实”的转发信息库(Forward Information dataBase,FIB),这使路由计算的建模更加容易. 它将整个验证过程分为三个模块,即控制面模拟、数据面建模和属性检查,每个模块提取每个阶段的差异性. 具体拓扑如图7(a)所示,网络有五个运行BGP的路由器,分别标注了各路由器的发布、过滤以及拒绝流量的情况. 为了模拟网络的改变,考虑一个单链路失败的情况,即路由器C和E之间的链路失败,结果路由器A的流量无法到达两个24前缀的网络. 按照图7(b)中DNA的三个模块,第一阶段首先模拟控制面的差异性,C的端口1和E的端口2之间的链路失败被编码为记录-Link(C,1,E,2)的删除,作为阶段一的



(*a*) 验证示例　　(*b*) DNA的工作流程

图7　DNA验证流程图[57]

输入:模拟控制面和FIB的差异性,即在路由器C,两个24前缀路由的输出端口1被删除,输出端口2被添加.在第二阶段,接受控制面状态的差异输入,输出数据面模型的差异,即流量之间的转发从边(C,E)到边(C,D),从边(B,C)到边(B,D),涉及四个边的删除和插入状态.第三阶段接受数据面模型的差异,增量计算相关的转发行为,输出属性的差异.该实例会输出-Reach(A1,E1,a),意味着属于等价类a的包不能再由路由器A的端口1转发至路由器E的端口1,这表示随着链路CE的失败,该部分的可达性属性已不再成立.

DNA能增量地只对受影响的网络策略进行验证,使验证更加高效,在几秒钟内计算出由控制平面变化引起的可达性差异,比现有的控制平面验证方法如Minesweeper快3个数量级.不足的是,目前DNA只针对转发行为相关策略进行增量配置验证,许多其他的网络策略如多路一致性、隔离性等并不在考虑范围内.

(2)区别与联系

上述两个工作都是采用增量计算实现的控制面可扩展验证的,与数据面增量验证工作如APC和Delta-net等只针对数据包转发进行增量更新不同,INCV是探讨网络配置验证中的增量验证方法,并介绍了一种名为RealConfig的增量配置验证器,可以在一秒钟内检查配置变化.DNA是INCV的扩展版本,其分别将配置、外部路由、可用链路、路由器的差异作为输入,基于控制平面的差异增量计算转发行为的差异,将计算分为三个模块阶段,即DDlog控制平面仿真、数据平面建模和属性检查,增量计算控制和数据平面状态,输出端到端的行为差异,使每个阶段都是增量的.

4.3.3　基于状态依赖性

(1)相关工作

与验证确定性属性不同,NetDice[58]关注属性成立的概率问题,根据依赖性删除与验证属性无关的状态遍历.考虑图8的一个包含6个路由器和链路权重的OSPF网络,假设需要验证在链路失败的情况下流量从D到C经过中间节点E的概率.假设每条链路都有2种情况,即故障和工作,那么总共需要探索的有2^7=128种.但是暴力搜索速度非常慢,复杂度与链路的数量成指数增长,NetDice发现当锁定转发路径是D-E-C后,其余五条链路可以被删除,如图8右侧所示,这五条链路的故障与否均不影响待验证的目标属性.

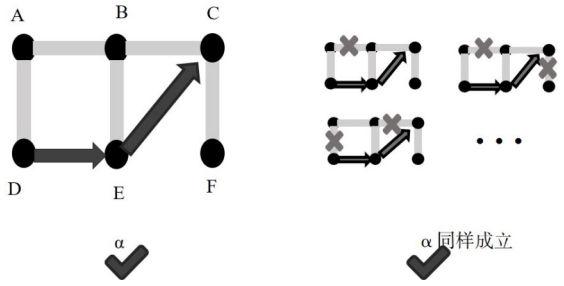


图8　NetDice探测链路失败情况[58]

通过删除与待验证属性不相干的链路,只关注属性依赖性链路,NetDice支持快速对网络进行概率性验证.但是NetDice仅能验证包含数百条链路网络的相关属性,对于更大规模的网络,NetDice无法快速地选择与属性不具有依赖关系的链路.而且随着依赖性链路变多,针对单个链路失败建模的概率具体不确定性,NetDice根据概率预测的精度也会降低.在该工作基础上,Zhang等人[19]提出的SRE考虑了确定性属性有综合了概率性属性,通过对给定的验证属性进行分析,利用路由剪枝、地址剪枝和抽象表示三种方法,简化了计算复杂度.在检查不同大小的"胖树"的全对可达性时,节点数量增多不会同比例增加对内存消耗,当有500个节点4 000条链路时,内存占用不超过10 GB.

Snowcap[59]是第一个验证重配置的框架,可以综合任意的配置更新分析,不同的配置顺序会导致意外的网络行为.为了验证配置顺序是否满足配置对应的功能,需要保证各个配置符合相应的依赖关系.给定初始和最终配置,Snowcap根据配置间关系,把安全重配置建模为带依赖约束的最优化问题,自动生成满足网络属性的配置顺序.即使对于大型拓扑,Snowcap也在最多几

秒钟内找到了一个有效的重配置顺序. 但是Snowcap从初始配置到最终配置,中间需要经历许多转换,无法在不受依赖性约束的情况下直接生成安全的最终配置.

Shao等人[10]提出了一种将路由策略验证问题转化为满足模型理论问题的方案biNode. 其关键思想是以策略感知的方式构建SMT模型,尽可能减少、消除变量之间的相互依赖关系. 此外,通过修剪,减小了生成的SMT模型的大小. 研究团队通过一个现成的SMT求解器实现并评估了策略感知模型. 实验结果显示,即使在规模较小的拓扑中,策略感知模型也能够将验证时间减少多达99%. 对于包含数万个节点的拓扑,只需要几分钟就能回答一个查询.

Yang等人[13]根据引发和执行这些计算的路由进程间依赖关系,提出了一种通过捕获路由进程间因果关系,将分布式路由计算转换成串行执行程序的方式,从而可以利用各种顺序程序分析理论和工具来诊断和修复路由配置错误. 他们给出了两种路由配置错误诊断的初步设计方法:使用最小不可满足核和错误不变式进行数据流分析;使用选择性符号执行进行控制流分析. 每个网络中的源拒绝错误诊断在不到50 ms的时间内完成;对于大多数存在路由传播错误的网络,诊断时间通常小于50 ms,最多可达230 ms.

(2)区别与联系

NetDice和SRE都关注了属性成立的概率问题. 利用属性依赖关系进行拓扑剪枝和建议约束,这种“剪枝”思想极大减小了NetDice的搜索空间,可以快速对网络进行概率性验证;Snowcap可以根据约束依赖自动生成安全的配置顺序,在几秒钟内为现实的网络拓扑结构和场景找到良好的重新配置计划;biNode以策略感知的方式构建SMT模型,并通过减少变量间的相互依赖关系和修剪模型的大小来简化验证过程;Yang等人[13]的方法捕获不同设备路由计算过程中的因果关系,将每个目的前缀路由的计算过程都表示成一个有向无环图.

4.3.4 基于层次分解

(1)相关工作

Tiramisu[5]将编码从验证算法分离,进行多层次协议建模,是首个支持属性定制化验证算法的多层控制面模型. 传统的控制面验证工具难以在性能和通用性之间权衡,它们大多数忽略了多层协议间的交互如虚拟局域网(Virtual Local Area Network,VLAN)等. 其关键思想是将编码从验证算法中分离出来,Tiramisu利用了一种新的、丰富的流量传播图编码,可以模拟各种控制平面特征和网络设计结构. 其图模型是多层次的,使用多属性的边权值,从而捕获协议之间的依赖关系,以便为每组独立的协议使用不同的验证工具. 例如图9展示的,针对多层次的图模型,对于偏好策略可以使用路径枚举的方法,对于链路失败相关的属性可以使用量化的图属性等. Tiramisu能捕捉协议之间的依赖关系(例如,iBGP依赖OSPF提供的路径)以及虚拟和物理链路之间的依赖关系. Tiramisu在路径是否存在的策略验证上只花了3 ms,速率比Minesweeper快了多达600倍. 在iBGP网络上,Tiramisu的速率在故障情况下比Plankton快了多达300倍.

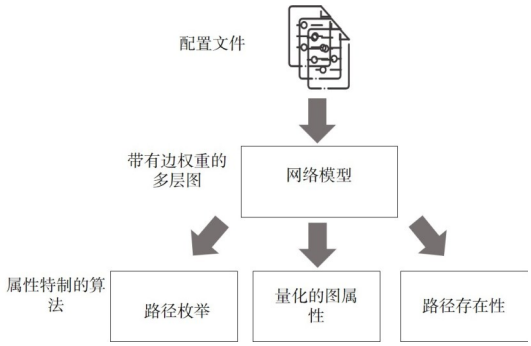


图9 Tiramisu单独编码模型和验证算法[5]

(2)区别与联系

Tiramisu将编码和验证算法分层,允许对不同类别的策略使用针对性的自定义验证算法,在提高性能的同时,也增强了跨层次验证的空间维度. 而前面介绍的数据面验证工作KATRA是对协议栈进行层次分解,依次剥离处于同一个包头但属于不同协议栈的依赖关系,空间上支持多协议跨层验证.

4.3.5 基于逻辑编程语言

(1)相关工作

Guha等人[21]首次将NetCore编程语言应用于SDN验证. NetCore是一种声明性语言,它允许程序员描述他们想要的网络行为,为应用层提供描述接口,在Coq证明助手[60]中,通过形式化规范和详细的SDN可操作模型,开发一个经过验证的SDN控制器. 在控制器中,运维人员使用NetCore编程语言描述网络的行为,该语言抽象了底层交换机硬件和分布式系统的细节,并允许程序员用简单的逐跳数据包处理步骤进行推理. 但是该编程语言只能支持特定控制器上的特定编程语言的配置程序,如该工作中采用的是轻量级的OpenFlow[61],对于业界流行的开源控制器,例如ONOS[62]和Floodlight[63]上的配置程序并不支持,缺少通用性.

FlowLog[12]是另一项扩展表达空间的研究工作. 研究指出虽然NetCore可以方便地描述转发策略,但缺乏联系或改变控制器状态的能力. 为了更好地支持控制器编程,其创建了FlowLog,一种无层网络编程语言,基于模型检测来支持验证功能. 它通过解决现有语言在控制平面和数据平面处理上的限制,提供了一个统一的编程抽象,并具备内置的验证和主动编译功能. 这种语言在实际应用中表现出了良好的性能和灵活性,并

为SDN开发者提供了一种强大的工具.

VeriCon[22]是第一个能够验证SDN程序在所有可接受的拓扑和所有可能的(无限)网络事件序列中是否正确的系统,使用编程语言CoreSDN和语义模型,将网络控制集中化并与网络硬件分离,使网络管理员可以在控制器上运行程序来指定转发规则、访问控制策略等,而无需直接与网络硬件或其他SDN程序进行交互.通过符号推理来验证网络(潜在的无限)状态,以验证对于任何事件序列(例如,控制器从交换机接收数据包头或链路故障)和所有可接受的拓扑,验证失败时会返回一个反例. Ball等人[22]认为VeriCon是实现验证SDN程序网络范围不变量的实用机制的第一步.

Aura[64]是一个用于数据中心路由策略的生产级综合系统,它包括一个高级语言RPL(Routing Policy Language),用于表达所需的行为,以及一个编译器,自动生成交换机的配置. 通过声明性语言,RPL允许操作员定义具有灵活交换机粒度和激活条件的策略意图,此外,Aura系统分阶段和利用标签来激活配置,从而最小化了每次更改配置时重新配置网络的需求. 现有验证工具使用RPL作为事实来源,并根据手动生成的BGP配置对其进行验证. RPL可以在Aura部署之前和期间进行全局验证,从而保证策略迁移的正确性.

(2)区别与联系

与Hydra通过使用Indus对各种属性进行建模的目的类似,上述几个工作都是为了增强验证语义的表达空间,NetCore和VeriCon方便了运维人员编码网络行为,只需指定上层行为语义,而不用关心底层实现,因为策略行为在Coq中得到了验证,所以可以一劳永逸地确定了控制器的正确性,而不需要像大多数控制器那样进行运行时或事后验证. FlowLog则是提供了与外部程序交互的接口和抽象,程序员可以根据自己的分析目标,根据需要自由地调用现有的、功能齐全的库.

4.3.6　基于转发相似性

(1)相关工作

Kinetic[65]致力于验证动态的网络控制平面. 针对SDN的动态变化,采用有限状态机,但是状态机会面临状态爆炸问题. 为此,Kinetic根据数据包来源进行划分,相同源端的流量被作为一个等价类,在PEC的思想下,该程序会将同一对端点之间的带内流量和带外流量赋予相同的状态,不需要为每个端节点都单独编码. Kinetic的验证时间跟PEC的数量呈线性关系,有效提高了验证效率,属性的数量不会明显影响验证时间,文中展示的所有的验证都在35 ms内完成. 但是如果网络中小流数量比较多,只根据包行为相似性对流空间划分仍然会存在状态爆炸问题,小流庞大的数量会导致更大的划分空间,使Kinetic验证效率下降.

Plankton[9]是一个配置验证平台,数据面验证工具可以分析大的头部空间去决定所有可能的数据面行为,软件验证技术可以探索软件所有的执行路径,包括测试遗漏的执行,Plankton结合二者的优势,使用数据包划分管理大的头部空间和清晰的状态模型检查探索协议执行. Plankton不仅通过数据包在协议执行过程中的相似性建立等价类分析,而且通过有序调度解决了PEC之间存在依赖关系的问题. Plankton支持OSPF,BGP和静态路由验证,并通过实验证明它可以在实际规模的网络中验证策略(对于超过300个设备的网络,验证时间不到1 s). 在验证“胖树”中的OSPF等策略中,当节点数增加,占用内存增加的并不明显,总的消耗不超过0.1 GB. 但是Plankton缺乏对BGP多路径的支持和对路由聚合的有限支持,当检查跨PEC依赖的网络时,也可能产生假阳性,因为它预期一个PEC的每个收敛状态可能与依赖它的其他PEC的每个收敛状态共存.

(2)区别与联系

尽管上述两个工作都基于行为相似性建立了等价类,但是它们的等价类含义有所区别. Kinetic根据来源对数据包建立等价类,同一对端点之间的带内流量和带外流量赋予相同的状态,不需要为每个端节点都单独编码. 不同的是,Plankton利用符号划分来管理大型报文头空间,将具有相同协议行为的控制面指令建立等价类,并进一步通过有序调度解决PEC之间的依赖关系. 在前面提到的数据面工作中,APKeep也利用了数据包转发的相似性的思想,但等价类含义有所差别,它通过PPM技术构建有向图,使用符号标识的等价类表示图中边的信息,对行为相似的等价类进行合并.

4.3.7　基于FASTPLANE模拟

(1)相关工作

无论数据平面还是控制平面,现有的网络验证工具服务的主要对象都是单管理域网络. 多管理域网络故障的检测手段还主要依赖不同管理域的网络管理员进行人工沟通与排查. 为了应对多管理域网络的网络故障检测问题,Xu等人[66]提出了基于安全多方计算的多管理域网络验证系统InCV,该系统结合了FASTPLANE的路由模拟计算加速技术[67],通过选择合理的路由传播模拟次序,加快收敛速度,最终提升系统性能. FASTPLANE的关键思想是首先发送全局最优路由公告,初步实验结果表明InCV可在可接受时间开销内对中小型域间网络进行验证(在至多32个参与域的网络中以不超过52 min耗时完成验证),并保护参与者配置隐私信息.

4.3.8　分析与总结

本节主要介绍了控制面验证工作的可扩展性技术,尽管和数据面验证会用到相同的技术思想提高可扩展性,但是所解决的问题并不相同. 控制面更关注对

网络配置的建模优化，数据面则更关注对转发状态的优化．与此同时，我们将每个相关工与利用相同优化思想的数据面研究工作进行了差异性分析．

表5分析了每个相关工作的时空可扩展性特点，与数据面工作类似，它们都能在不同的时空维度上可扩展，不同的是控制面工作在表达空间的可扩展性上表现得相对更加突出，特别是具备建模多种不同的协议能力和表达复杂的功能规范和约束能力的相关工作更加丰富．

表5 控制面可扩展验证相关工作总结

可扩展技术	相关研究	时间	内存空间	表达空间
基于网络切片	ARC[16]	高	—	—
	ERA[18]	中	—	—
	Minesweeper[30]	中	—	中
	Bonsai[54]	高	高	—
	LIGHTYEAR[17]	中	—	—
	Kirigami[55]	高	—	—
基于转发相似性	Plankton[9]	高	高	低
	Kinetic[65]	中	—	—
基于状态依赖性	NetDice[58]	中	高	—
	Snowcap[59]	低	—	—
	SRE[19]	高	低	—
	biNode[10]	低	—	—
	Yang et al[13]	低	—	—
基于增量计算	INCV[56]	中	—	—
	DNA[57]	高	—	低
基于层次分解	Tiramisu[5]	高	—	高
基于逻辑编程语言	VeriCon[22]	—	—	高
	NetCore[21]	—	—	中
	FlowLog[12]	—	—	中
	Aura[64]	中	—	高
基于FASTPLANE模拟	InCV[66]	中	—	—

5 发展趋势与展望

为了满足时空约束的可扩展性，前文已经从网络结构、数据包行为、路由关系描述等方面进行了充分论述，保证验证过程快速高效地完成．随着现代网络和技术的发展，未来网络验证的可扩展性也会迎来新的问题和发展方向．

5.1 基于策略相关性提高多策略验证效率

当网络规模不断扩大，需要验证的策略也会增多，此前的研究多数只针对某一类策略进行验证，如与路由相关的策略、与过滤规则相关的策略，针对同一类中的各个策略再单独验证．当待验证策略数量较多时，这种串行方式很难在短时间内发现配置错误．

实际上，网络策略并不是相互独立的，而是具有一定的层次结构和关联性的．例如以可达性为出发点，如果流量不可达，那么可能存在转发循环或者黑洞等综合属性；如果流量可达，可能会有基于该子属性的经过单点（waypoint）和经过多点两种综合属性待验证．通过优先调度综合属性（如waypoint），可以让其余部分属性的验证不攻自破（可达性自然成立），减少了需要实际验证的属性数量，在短时间内验证更多的网络策略．而且对于不相干的属性，即分别属于不同的类别，可以从不同的子属性同时出发，以达到并行验证的目的．通过策略调度优先验证综合属性达到“以一验多”的目的，减少所需的验证时间．

5.2 基于分布式验证提高并发处理能力

现有的传统网络下的网络验证工具，大都采用的是集中式验证的方法，即需要收集控制面或数据面信息，然后进行集中式的分析验证．这种方式在大型网络中面临可扩展性问题，因为验证器成为性能瓶颈和单点故障，并需要可靠的管理网络．

通过构建分布式验证系统，将验证任务分散到多个节点或服务器上进行并行处理．这种方式可以提高验证的并发性和处理能力，适应大规模网络环境下的需求．同时，通过合理的任务调度和负载均衡算法，确保验证任务在整个系统中均匀分布和高效执行．目前，已有的分布式验证研究有Coral[51]和Tulkun[20]．

5.3 基于跨平台验证工具解决设备协议异构性

网络设备行为模型的正确性很难保证，这是由于厂商特定行为的存在．不同厂商以不同的方式实现协议的部分功能，而在构建验证工具时开发人员可能并不知道这些差异．如果未能捕获一个或多个行为差异，这可能会严重影响验证结果的准确性．

开发能够跨不同平台和设备进行验证的工具，这些工具应支持多种设备厂商和操作系统，能够适应不同设备的配置和特性．同时，这些工具应提供灵活的配置选项和自定义的验证规则，以满足异构网络环境的需求．Hoyan[26]是第一个考虑设备供应商在协议行为实现方面差异的控制平面验证工具，增强了协议建模的表达能力．

5.4 基于运维大模型解决语义建模问题

网络验证过程涉及复杂的设备和协议交互，目前运维人员使用定制化的语言建模协议和策略的行为规范，具有一定人工的开销，在未来可以考虑利用运维大模型[68]解决这个问题．

将定义的规则和属性输入到运维大模型中进行验证和分析，模型会根据已学习的知识和模式，对网络状态和行为进行推断和预测，判断是否满足所定义的规则和属性．如果某个属性不成立，模型可以提供反例或故障定位信息，并且运维大模型将验证结果可视化展

示,生成详细的报告,包括问题描述、验证结果、原因分析和解决方案,帮助运维人员理解问题的本质和采取相应的措施.

5.5 小结

针对未来网络验证可扩展性问题,我们提供了几种解决思路,包括基于策略相关性提高验证效率、基于分布式验证提高并发处理能力、基于跨平台验证工具解决设备协议异构性以及基于运维大模型解决语义建模问题,从时间和空间上提高验证效率.

6 总结

本文从数据平面和控制平面两个角度,总结并分析了现有网络验证工作是如何解决时空可扩展性问题的,并对各类方案的优缺点和突出特点进行了分析,提出了未来的研究趋势和潜在的可扩展性方法. 总的来说,现有的网络验证已经得到了较好的发展,从强假设到弱假设,从静态验证到实时验证,无论是验证覆盖率还是效率都得到了提升,网络验证的可部署性也得到了改善.

但是将这些设计理论完全成功应用于实践还存在一定距离,目前的可扩展技术针对网络结构本身做了一定“瘦身”,如抽离了复杂的功能、简化了中间设备角色、未考虑网络环境的变化等,这些问题仍是部署的“卡脖子”难点. 如今网络验证仍处于发展热潮,解决这些问题需要综合运用最新技术成果,科学设计全面的可扩展性技术,同时考虑系统的可部署性,为网络运维人员和用户提供安全可靠、行为一致的网络环境.

参考文献

[1] JANGJOU M, SOHRABI M K. A comprehensive survey on security challenges in different network layers in cloud computing[J]. Archives of Computational Methods in Engineering, 2022, 29(6): 3587-3608.

[2] DHATRAK A, SARKAR A, GORE A, et al. Cyber security threats and vulnerabilities in IoT[J]. International Research Journal of Engineering and Technology, 2020, 7(3): 2921-2925.

[3] DEB R, ROY S. Dynamic vulnerability assessments of software-defined networks[J]. Innovations in Systems and Software Engineering, 2020, 16(1): 45-51.

[4] PAN L, QIU R, WANG A, et al. Measuring the resolution resiliency of second-level domain name[C]//InScience and Information Conference. Cham: Springer, 2022: 742-755.

[5] ABHASHKUMAR A, GEMBER-JACOBSON A, AKELLA A. Tiramisu: Fast multilayer network verification[C]//Proceedings of the 17th USENIX Conference on Networked Systems Design and Implementation. New York: ACM, 2020: 201-220.

[6] LI Y, YIN X, WANG Z, et al. A survey on network verification and testing with formal methods: Approaches and challenges[J]. IEEE Communications Surveys & Tutorials, 2018, PP(99): 1.

[7] ZHANG X, WANG C, LI Q, et al. Toward comprehensive network verification: Practices, challenges and beyond[J]. IEEE Network, 2020, 34(1): 108-115.

[8] 方星, 胡波, 马超, 等. 网络验证研究综述[J]. 软件学报, 2023, 34(1): 351-380.
FANG X, HU B, MA C, et al. Survey on network verification[J]. Journal of Software, 2023, 34(1): 351-380. (in Chinese)

[9] PRABHU S, CHOU KY, KHERADMAND A, et al. Plankton: Scalable network configuration verification through model checking[C]//Proceedings of the 17th USENIX Conference on Networked Systems Design and Implementation. New York: ACM, 2020: 953-968.

[10] SHAO X, GAO L. Verifying policy-based routing at internet scale[C]//Proceedings of IEEE International Conference on Computer Communications (INFOCOM). Piscataway: IEEE, 2020: 2293-2302.

[11] YUAN Y, MOON SJ, UPPAL S, et al. NetSMC: A custom symbolic model checker for stateful network verification[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2020: 181-200.

[12] NELSON T, FERGUSON AD, SCHEER MJ, KRISHNAMURTHI S. Tierless programming and reasoning for software-defined networks[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2014: 519-531.

[13] YANG R, XING F, LIZHAO Y, et al. Diagnosing distributed routing configurations using sequential program analysis[C]//Proceedings of the Asia-Pacific Workshop on Networking (APNet). New York: ACM, 2023: 34-40.

[14] KAZEMIAN P, GEORGE V, NICK M. Header space analysis: Static checking for networks[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2012: 113-126.

[15] RUCHANSKY N, PROSERPIO D, CANINI M, et al. A nice way to test openflow applications[C]//Proceedings of the USENIX Symposium on Networked Systems Design

and Implementation (NSDI). New York: ACM, 2012: 127-140.

[16] GEMBER-JACOBSON A, VISWANATHAN R, AKELLA A, MAHAJAN R, et al. Fast control plane analysis using an abstract representation[C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2016: 300-313.

[17] TANG A L, BECKETT R, BENALOH S, et al. Lightyear: Using modularity to scale bgp control plane verification[C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2023: 94-107.

[18] FAYAZ S K, SHARMA T, FOGEL A, et al. Efficient network reachability analysis using a succinct control plane representation[C]//Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI). New York: ACM, 2016: 217-232.

[19] ZHANG P, WANG D, GEMBER-JACOBSON A. Symbolic router execution[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2022: 336-349.

[20] XIANG Q, HUANG C Y, WEN R D, et al. Beyond a centralized verifier: Scaling data plane checking via distributed, on-device verification[C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2023: 152-166.

[21] GUHA, A, REITBLATT M, FOSTER N. Machine-verified network controllers[J]. ACM SIGPLAN Notices, 2013, 48(6): 483-494.

[22] BALL T, BJØRNER N, GEMBER A, et al. VeriCon: Towards verifying controller programs in software-defined networks[C]//Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM, 2014: 282-293.

[23] ANDERSON C J, FOSTER N, GUHA A, et al. NetKAT: semantic foundations for networks[C]//Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM, 2014: 113-126.

[24] PANDA A, LAHAV O, ARGYRAKI K, et al. Verifying reachability in networks with mutable datapaths[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2017: 699-718.

[25] MAI H, KHURSHID A, AGARWAL R, et al. Debugging the data plane with anteater[C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2011: 290-301.

[26] YE F, YU D, ZHAI E, et al. Accuracy, scalability, coverage: A practical configuration verifier on a global WAN [C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2020: 599-614.

[27] WEBSTER M, WESTERN D, ARAIZA-ILLAN D, et al. A corroborative approach to verification and validation of human-robot teams[J]. The International Journal of Robotics Research, 2020, 39(1): 73-99.

[28] DANIEL LA, BARDIN S, REZK T. Binsec/rel: Efficient relational symbolic execution for constant-time at binary-level[C]//Proceedings of the IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE, 2020: 1021-1038.

[29] HANDIGOL N. Intent-based verification leading a new wave of network automation[EB/OL]. (2019-05-21) [2023-07-01]. https://www. networkcomputing. com/networking/intent-based-verification-leading-new-wave-network-au - tomation.

[30] BECKETT R, GUPTA A, MAHAJAN R, et al. A general approach to network configuration verification[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2017: 155-168.

[31] ZENG H Y, ZHANG S D, YE F, et al. Libra: Divide and conquer to verify forwarding tables in huge networks[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2014: 87-99.

[32] ZHANG P, LIU X, YANG H, et al. APKeep: Realtime verification for real networks[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2020: 241-255.

[33] AWEYA J. IP Routing Protocols: Link-State and Path-Vector Routing Protocols[M]. First edition. Boca Raton: CRC Press, 2021.

[34] PLOTKIN G D, BJØRNER N, LOPES N P, et al. Scaling network verification using symmetry and surgery[J]. ACM SIGPLAN Notices, 2016, 51(1): 69-83.

[35] AL-FARES M, LOUKISSAS A, VAHDAT A. A scalable, commodity data center network architecture[C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2008: 63-74.

[36] ABDALLA H B, AHMED A M, SIBAHEE MA A L. Op-

timization driven mapreduce framework for indexing and retrieval of big data[J]. KSII Transactions on Internet and Information Systems, 2020, 14(5): 1886-1908.

[37] GUO D, CHEN S S, GAO K, et al. Flash: Fast, consistent data plane verification for large-scale network settings [C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2022: 314-335.

[38] HORN A, KHERADMAND A, PRASAD M R. Delta-net: Real-time network verification using atoms[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2017: 735-749.

[39] FAYAZ S K, YU T L, TOBIOKA Y, et al. BUZZ: Testing context-dependent policies in stateful networks[C]// Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2016: 275-289.

[40] ZENG H Y, KAZEMIAN P, VARGHESE G, et al. Automatic test packet generation[C]//Proceedings of the International Conference on Emerging Networking Experiments and Technologies (CoNEXT). New York: ACM, 2012: 241-252.

[41] KHURSHID A, ZOU X, ZHOU W X, et al. VeriFlow: Verifying network-wide invariants in real time[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2013: 15-28.

[42] ZHANG K Y, ZHUO D Y, AKELLA A, et al. Automated verification of customizable middlebox properties with gravel[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2020: 221-240.

[43] The P4 Language Consortium. P4 portable NIC architecture[EB/OL]. (2021-05-18)[2023-11-28]. https://p4.org/p4-spec/docs/PNA.html.

[44] ZHANG Y C. SONiC system architecture[EB/OL]. (2022-07-21) [2023-11-28]. https://github. com/sonic-net/SONiC/wiki/Architecture.

[45] YANG H K, LAM S S. Real-time verification of network properties using atomic predicates[J]. IEEE/ACM Transactions on Networking, 2015, 24(2): 887-900.

[46] WANG H Z, QIAN C, YU Y, et al. Practical network-wide packet behavior identification by AP classifier[J]. IEEE/ACM Transactions on Networking, 2017, 25(5): 2886-2899.

[47] BECKETT R, GUPTA A. Katra: Realtime verification for multilayer networks[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2022: 617-634.

[48] MAHALINGAM M, DUTT D, DUDA K, et al. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks: RFC 7348: 2014[S/OL]. (2020-01-21) [2023-12-20]. https: //doi.org/10.17487/RFC7348.

[49] LI J Q, ZHOU M. Research on VPN in experimental simulation environment based on GRE and IPSec[C]//Proceedings of the International Conference on Robotics and Artificial Intelligence. New York: ACM, 2020: 220-224.

[50] RENGANATHAN S, RUBIN B, KIM H, et al. Hydra: Effective runtime network verification[C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2023: 182-194.

[51] XIANG Q, WEN R D, HUANG C Y, et al. Network can check itself: Scaling data plane checking via distributed, on-device verification[C]//Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets). New York: ACM, 2022: 85-92.

[52] FOGEL A, FUNG S, PEDROSA L, et al. A general approach to network configuration analysis[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2015: 469-483.

[53] RUMREICH L. The Binary Decision Diagram: Formal Verification of a Reference Implementation[R]. Columbus: The Ohio State University, 2021.

[54] BECKETT R, GUPTA A, MAHAJAN R, et al. Control plane compression[C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2018: 476-489.

[55] THIJM T A, BECKETT R, GUPTA A, et al. Kirigami, the verifiable art of network cutting[C]//Proceedings of the IEEE International Conference on Network Protocols (ICNP). Piscataway: IEEE, 2022: 1-12.

[56] ZHANG P, HUANG Y H, GEMBER-JACOBSON A, et al. Incremental network configuration verification[C]// Proceedings of the ACM Workshop on Hot Topics in Networks. New York: ACM, 2020: 81-87.

[57] ZHANG P, GEMBER-JACOBSON A, ZUO Y, et al. Differential network analysis[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2022: 601-615.

[58] STEFFEN S, GEHR T, TSANKOV P, et al. Probabilistic verification of network configurations[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2020: 750-764.

[59] SCHNEIDER T, BIRKNER R, VANBEVER L. Snowcap: Synthesizing network-wide configuration updates [C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2021: 33-49.

[60] CHLIPALA A. Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant[M]. Cambridge: MIT Press, 2022.

[61] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: Enabling innovation in campus networks [C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM). New York: ACM, 2008: 69-74.

[62] BERDE P, GEROLA M, HART J, et al. ONOS: Towards an open, distributed SDN OS[C]//Proceedings of the Third Workshop on Hot Topics in Software Defined Networking (HotSDN). New York: ACM, 2014: 1-6.

[63] MORALES LV, MURILLO AF, RUEDA SJ. Extending the floodlight controller[C]//Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA). Piscataway: IEEE, 2015: 126-133.

[64] RAMANATHAN S, ZHANG Y, GAWISH M, et al. Practical intent-driven routing configuration synthesis[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2023: 629-644.

[65] KIM H, REICH J, GUPTA A, et al. Kinetic: Verifiable dynamic network control[C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI). New York: ACM, 2015: 59-72.

[66] XU H, QIN Q, FANG X, et al. Toward privacy-preserving interdomain configuration verification via multi-party computation[C]//Proceedings of the Asia-Pacific Workshop on Networking (APNet). New York: ACM, 2023: 28-33.

[67] LOPES NP, RYBALCHENKO A. Fast BGP simulation of large datacenters[C]//International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI). Cham: Springer, 2019: 386-408.

[68] 饶琛琳. 大模型在运维领域的应用展望[EB/OL]. (2023-08-10) [2023-11-28]. https://mp.weixin.qq.com/s/ewuSv2wX0uouyg9JOQDIMA.

作者简介

黄翰林 男，1998 年 7 月出生于河南省信阳市. 清华大学计算机系博士研究生. 主要研究方向为网络验证和网络安全.
E-mail: hhl21@mails.tsinghua.edu.cn

徐 恪 男，1974 年 12 月出生于江苏省淮安市. 现为清华大学计算机系教授、博士生导师. 获得国家技术发明奖二等奖 1 项，国家科技进步奖二等奖 1 项，省部级特等奖 1 项，省部级一等奖 5 项. 主要研究方向新一代互联网体系结构、网络空间安全、区块链系统. 中国电子学会会员编号：E190008630F.
E-mail: xuke@mail.tsinghua.edu.cn

李 琦 男，1979 年 1 月出生于浙江省长兴县. 现为清华大学网络研究院副教授、博士生导师. 主要研究方向为互联网安全. 中国电子学会会员编号：E190026656S.
E-mail: qli01@tsinghua.edu.cn

李 彤 男，1989 年 9 月出生于湖南省涟源市. 现为中国人民大学副教授、杰出学者、硕士生导师. 主要研究方向为新一代互联网体系结构、分布式系统、大数据.
E-mail: tong.li@ruc.edu.cn

付松涛 男，1982 年 2 月出生于贵州省遵义市. 2023 年获清华大学计算机系博士学位. 现为卫星通信中心工程师. 主要研究方向为计算机网络、卫星通信. 中国电子学会会员编号：E190024348A.
E-mail: fust18@tinghua.org.cn

高翔宇 男，1999 年 5 月出生于辽宁省沈阳市. 清华大学网络研究院硕士研究生. 主要研究方向为计算机网络和网络安全. 中国电子学会会员编号：E190028664A.
E-mail: gaoxy21@mails.tsinghua.edu.cn