软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(1):351−380 [doi: 10.13328/j.cnki.jos.006510]

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

网络验证研究综述*

方 星[1,2], 胡 波[1], 马 超[1], 黄伟庆[1]

[1](中国科学院 信息工程研究所, 北京 100093)

[2](中国科学院大学 网络空间安全学院, 北京 100049)

通信作者: 胡波, E-mail: hubo@iie.ac.cn

摘 要: 随着计算机网络规模和复杂度的日益增长, 网络管理人员难以保证网络意图得到了正确实现, 错误的网络配置将影响网络的安全性和可用性. 受到形式化方法在硬软件验证领域中成功应用的启发, 研究人员将形式化方法应用到网络中, 形成了一个新的研究领域, 即网络验证 (network verification), 旨在使用严格的数学方法证明网络的正确性. 网络验证已经成为当下网络和安全领域的热点研究, 其研究成果也在实际网络中得到了成功应用. 从数据平面验证、控制平面验证和有状态网络验证 3 个研究方向, 对网络验证领域的已有研究成果进行了系统总结, 对研究热点内容与解决方法进行了分析, 旨在整理网络验证领域的发展脉络, 为本领域研究者提供系统性文献参考和未来工作展望.

关键词: 网络验证; 形式化方法; 网络可靠性; 网络安全; 基于意图的网络

中图法分类号: TP393

中文引用格式: 方星, 胡波, 马超, 黄伟庆. 网络验证研究综述. 软件学报, 2023, 34(1): 351–380. http://www.jos.org.cn/1000-9825/6510.htm

英文引用格式: Fang X, Hu B, Ma C, Huang WQ. Survey on Network Verification. Ruan Jian Xue Bao/Journal of Software, 2023, 34(1): 351–380 (in Chinese). http://www.jos.org.cn/1000-9825/6510.htm

Survey on Network Verification

FANG Xing[1,2], HU Bo[1], MA Chao[1], HUANG Wei-Qing[1]

[1](Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)

[2](School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: With the increasing scale and complexity of computer networks, it is difficult for network administrators to ensure that the network intent has been correctly realized, and the incorrect network configuration will affect the security and availability of the network. Inspired by the successful application of formal methods in the field of hardware verification and software verification, researchers applied formal methods to networks, forming a new research field, namely network verification, which aims to use rigorous mathematical methods to prove the correctness of the network. Network verification has become a hot research topic in the field of network and security, and its research results have been successfully applied in actual networks. From the three research directions of data plane verification, control plane verification, and stateful network verification, this study systematically summarizes the existing research results in the field of network verification, and analyzes the research hotspots and related solutions, aiming to organize the field of network verification and provides systematic references and future work prospects for researchers in the field.

Key words: network verification; formal method; network reliability; network security; intent-based networking

现代网络已经变得非常复杂和庞大, 运行着大量不同厂商与不同类型的网络设备, 并交互有数十种不同的网络协议. 随着诸如虚拟化、云计算、物联网等技术变得越来越普遍, 网络似乎变得更加复杂, 管理起来也变得更加困难, 从而变得非常容易出错. 软件定义网络 (software-defined networking, SDN)[1]通过解耦数据平面和控制平面, 以中

* 基金项目: 国家重点研发计划 (2019YFB1005205)
收稿时间: 2021-05-24; 修改时间: 2021-09-07; 采用时间: 2021-10-13; jos 在线出版时间: 2021-11-24
CNKI 网络首发时间: 2022-11-15

心化的控制代替传统网络 (非 SDN 网络) 下的分布式控制, 实现了简单灵活的网络管理. 然而, SDN 也带来了新的问题, 例如 SDN 程序或控制器实现错误. 可以看到, 无论是传统网络还是 SDN, 保证网络按照意图正确运行都非常困难.

在面对复杂且庞大的网络时, 网络管理人员主要依靠个人经验与理解进行管理, 被称作“复杂性大师”[2]. 然而, 在数百台网络设备交互作用的复杂网络环境下, 网络管理人员即使解释一些简单的策略问题也变得非常困难, 例如“什么类型的数据包可以从主机 A 到达主机 B?”或“网络中会发生数据包转发循环吗?” 此外, 网络管理人员在配置网络设备时经常会出现理解或实现错误, 不能保证网络按照意图正确的运行, 导致发生网络中断事件[3]. 例如, 微软公司的 Azure 服务于 2019 年 5 月 2 日因人为配置 DNS 失误, 导致宕机将近 3 个小时, 使整个微软的云服务都受到了影响[4]; CenturyLink 公司于 2020 年 8 月 30 日因错误配置了边界网关协议 (border gateway protocol, BGP), 导致全球 3% 到 5% 的流量受到了影响[5]. Opengear 公司与 OnePoll 公司于 2020 年 1 月合作对 500 位 IT 领域的高级决策者进行了一项调查, 51% 的受访者表示其公司在过去一年内经历了 4 次或更多次超过 30 min 的宕机事件, 65% 的受访者表示宕机事件在过去 5 年内有所增加, 31% 的受访者表示过去一年的宕机事件带来的损失超过了 100 万美元[6]. 因此, 需要一种解决方案, 能够自动并及时地发现网络中的错误行为, 甚至能提前发现错误并进行修复, 保证网络真实行为与意图的一致, 使网络变得可靠无缺陷.

受到硬软件验证技术的启发, 2012 年左右出现了一个新的研究领域, 即网络验证 (network verification), 旨在使用严格的数学方法证明网络的正确性. 硬软件验证技术通过使用形式化方法, 能够在硬软件投入使用之前证明系统的正确性, 以避免意外错误的发生. 例如, AMD 公司形式化验证了 AMD K5 处理器的浮点数除法运算的正确性[7], 美国航空航天局 (NASA) 使用形式化方法成功发现了火星漫游者飞行软件的并发运行缺陷[8]. 既然可以使用形式化方法验证硬软件, 为什么不能验证网络呢? 斯坦福大学、伊利诺伊大学厄巴纳-香槟分校等学术研究实验室做出了开创性的网络验证研究, 基于形式化方法验证了网络行为与网络意图之间的一致性, 证明了使用形式化方法验证网络行为正确性的可行性[9,10]. 之后, 国内外高校和公司陆续开展了网络验证方向的研究, 例如国外的微软研究院、加利福尼亚大学洛杉矶分校、普林斯顿大学、苏黎世联邦理工学院等研究团队[11−14], 国内的清华大学、西安交通大学、阿里巴巴等研究团队[15−17]. 借鉴于 Beckett 等人指出的网络验证领域主要研究方向[18], 本文将网络验证研究分为了数据平面验证、控制平面验证、有状态网络验证 3 个研究方向, 图 1 展示了各个方向中的代表性研究成果[9–13,15–17,19–49]. 可以看到, 网络验证领域经过近 10 年来的发展, 已经积累了丰富的研究成果, 且仍是当前研究的热点.

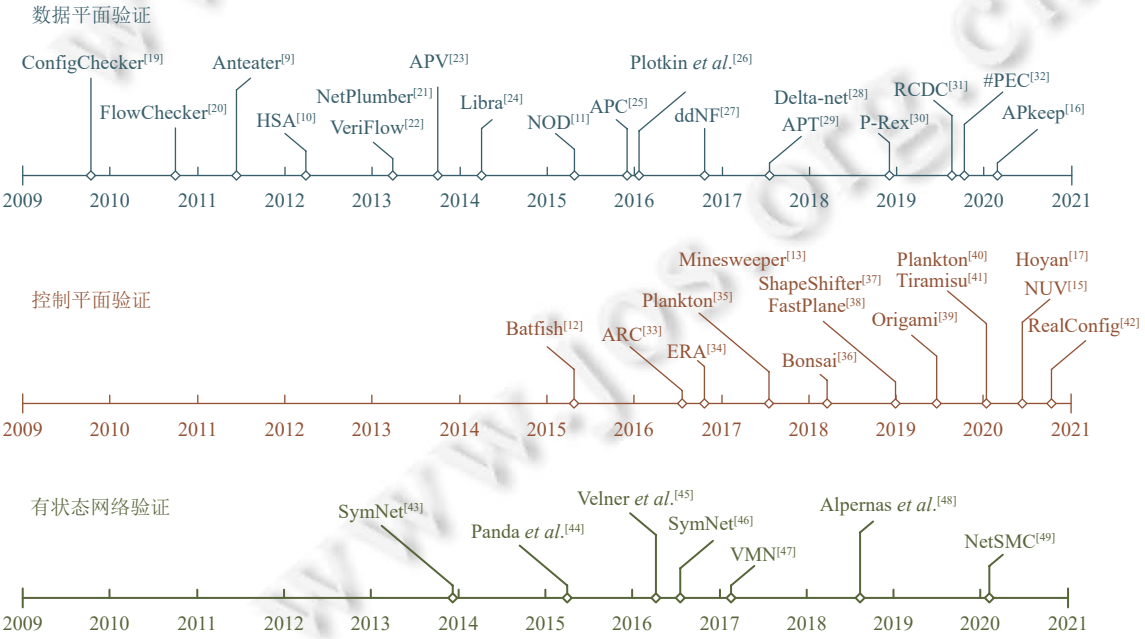


图 1 网络验证领域代表研究

网络验证使用严格的数学方法, 能够基于网络的配置或转发状态推断出所有可能的网络行为, 然后将网络行为与意图进行比较, 验证网络意图是否得到了正确实现. 相比于同样基于形式化方法进行建模分析, 但关注于验证单个协议行为正确性 (如不会出现死锁、活锁、溢出等错误状态) 的协议验证 (protocol verification) 领域[50], 网络验证关注于验证整个网络, 旨在实现全面的网络行为正确性分析 (如保证网络中无转发循环、指定节点之间相互可达、指定区域之间流量隔离等). 网络验证的研究意义在于, 可以自动化且准确全面地发现网络中潜在错误以及错误原因, 避免了繁琐易出错的手动分析. 例如, 当需要部署新的网络服务并对网络进行配置更新时, 网络验证可以在配置部署之前推断配置更新带来的影响, 保证其不会引发网络错误导致网络中断. 相比于使用网络测试方法发现错误, 网络验证通过严格的数学证明, 对网络正确性提供了全面的保证, 并可以在网络设计阶段发现错误, 提前阻止错误的发生. 目前, 网络验证研究已经在工业界得到了实际应用. 例如, 微软公司在数据中心网络 Azure 中, 使用网络验证技术分析网络设备的路由表与访问控制策略, 成功保障网络的性能意图或安全意图得到了正确实现, 如服务器集群之间的通信总是选择最优路径、私有数据中心服务器不能从因特网访问或只有部分端口开放访问等[31]; 阿里巴巴公司在其全球广域网中, 使用网络验证技术分析网络设备配置, 在配置下发前提前进行意图验证, 成功减少了因配置错误更新导致的网络服务中断, 保障了公司网络业务的正确运行与更新[17]. 此外, 网络验证领域已经出现了一些初创公司, 旨在为服务提供商和企业提供网络验证服务, 目前处于领先地位的初创公司包括 Intentionet[51]、ForwardNetworks[52]、Apstra[53]等. 近年来, 在计算机网络领域出现了一种新的网络范式, 即基于意图的网络 (intent-based networking, IBN), 其目标是使网络操作自动化并更好地使网络行为与网络意图保持一致, 被广泛认为是网络领域中的“下一件大事”[54]. 思科、华为、瞻博网络等领先网络公司, 纷纷将 IBN 作为下一步发展战略[55−57]. 其中, 网络验证作为 IBN 的关键环节, 发挥着不可或缺的重要作用.

目前, 已经有学者发表了网络验证领域的综述性文章[58−61]. Zhang 等人[58]根据研究实现的方法, 将网络验证研究分为了基于有限状态机和基于布尔可满足性的两种情况, 对其建模和验证流程进行了讨论; Qadir 等人[59]对网络验证领域进行了首次详细地综述, 其首先全面介绍了形式化方法的背景知识, 然后分别基于形式化验证技术和应用场景, 对网络验证研究进行了分类介绍; Li 等人[60]同样对网络验证领域进行了详细地综述, 相比于文献 [59] 更加注重于总结研究技术的发展情况, 且对网络测试领域进行了综述; Zhang 等人[61]对网络验证与网络测试领域的前沿研究进行了介绍. 本文与文献 [59,60] 的综述内容最为接近, 区别在于使用了不同的分类方法介绍已有网络验证研究. 本文首先根据研究内容将网络验证领域分为数据平面验证、控制平面验证和有状态网络验证 3 个研究方向, 然后根据研究主要解决的问题, 对各个方向的研究进一步地进行分类, 最后结合时间顺序与验证技术详细介绍各个研究, 旨在指出网络验证领域各个研究方向的开创性研究, 以及各个方向所面临的主要挑战以及解决方法, 从而系统梳理网络验证研究发展的整体脉络.

本文第 1 节介绍了网络验证领域中必要的背景知识; 第 2–4 节分别介绍了数据平面验证、控制平面验证、有状态网络验证 3 个研究方向的研究情况; 第 5 节介绍了网络验证相关领域的研究情况; 第 6 节对网络验证领域中的未来研究工作进行了展望; 最后对本文做出总结.

1 背景知识

为了更好地阐述网络验证领域中的研究情况, 下面分别介绍网络和验证两部分的背景知识. 网络相关的背景知识主要有数据平面、控制平面、SDN、有状态网络, 有助于理解各个研究方向的划分与研究问题; 验证相关的背景知识主要为形式化方法, 有助于理解网络验证研究的验证方法.

1.1 网络背景知识

如图 2 所示, 计算机网络根据功能可以被划分成 3 个层次: 数据平面、控制平面和管理平面[62]. 数据平面是指用于决定转发数据包的功能部分, 例如传统网络中的转发表或 SDN 中的流表; 控制平面是指通过结合网络拓扑等网络环境信息生成数据平面的功能部分, 例如传统网络中分散由各个路由设备配置文件实现的 OSPF、BGP 协议或 SDN 中集中于应用程序和控制器的逻辑控制功能; 管理平面是指用于配置或监视网络的功能部分, 例如传统网络中网络设备的命令行界面管理程序或 SDN 中的应用程序. 需要注意的是, 上述 3 个层次为抽象逻辑概念, 目

前并没有统一的标准定义. 从网络运行流程上看, 网络管理人员首先通过管理平面使用策略实现网络意图, 并下发到控制平面, 然后控制平面结合网络拓扑等环境信息生成数据平面, 最后数据包根据数据平面进行转发. 为了保证网络意图在网络中得到了正确实现, 可以在数据平面和控制平面两个层次上进行分析, 分别对应为数据平面验证和控制平面验证两个研究方向. 数据平面验证通过输入数据平面信息 (例如转发表或流表) 和网络拓扑, 验证数据平面下所有数据包转发行为与网络意图的一致性; 控制平面验证通过输入控制平面信息 (例如配置或 SDN 程序)、网络拓扑与其他环境信息 (例如链路状态、路由通告), 验证在这些信息生成的数据平面下, 所有的数据包转发行为与网络意图的一致性[63]. 此外, 还可以在管理平面层次上完成分析, 保证实现的策略完全符合意图, 此部分不在本文的讨论范围内, 属于网络配置综合 (configuration synthesis)[64]或 IBN 中意图转译的研究内容[65].

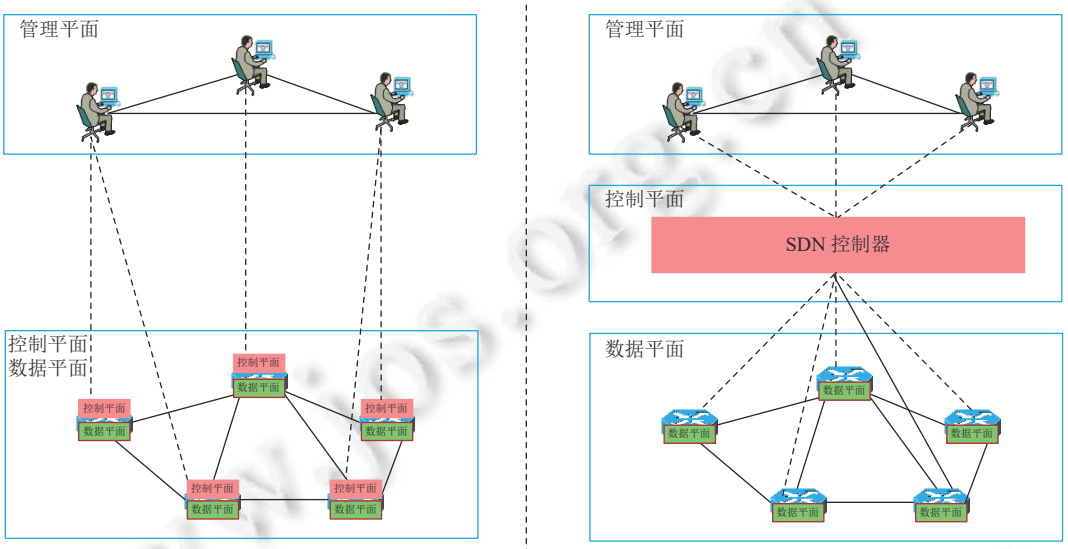


图 2 传统网络与 SDN 的网络功能逻辑分层

在传统网络中, 控制平面分散在网络中的各个设备中, 与数据平面高度耦合, 通过分布式的控制过程计算路由信息. 然而, 这种分布式的架构管理起来非常困难, 部署业务时需要逐个配置网络设备, 且需要考虑复杂多样的网络协议和厂商配置语言, 非常容易出错. 为了更好地管理网络与网络业务创新, McKeown 等人提出了 OpenFlow 方案[66], 创新性地解耦了数据平面和控制平面, 将控制平面集中到控制器上, 实现了可编程的网络. SDN 起源于 OpenFlow, 是指数据平面被集中式的控制平面控制的网络架构[62]. SDN 有以下 4 个特点: 数据平面和控制平面分离; 数据包转发基于流而不是目的地址; 控制逻辑集中到控制器; 可以通过应用程序对网络行为进行编程. SDN 的控制平面集中化, 使得用户可以使用高级语言而不是特定的厂商配置语言编写网络策略, 更加方便且不易出错. 而且, SDN 控制器掌握着网络全局状态, 有利于网络功能的开发, 使得 SDN 具备快速的业务创新能力. SDN 技术的出现, 一定程度上加速了形式化方法在网络中的应用, 促进了网络验证领域的发展[67]. 虽然 SDN 通过集中化的控制平面避免了分布式的复杂配置所导致的错误, 但也带来了新的错误因素, 例如 SDN 程序或控制器实现错误、SDN 程序之间相互冲突等[68,69]. 因此, 需要对 SDN 程序或控制器进行正确性验证, 保证其按照意图正确运行.

在传统网络架构下, 网络管理人员主要通过部署中间件 (middlebox)[70]以提升网络的安全性与转发性能, 例如防火墙、入侵检测系统、负载均衡器等. 中间件属于数据平面设备[37], 主要对网络中的数据包进行操作, 能够实现路由器所不能完成的功能, 如数据包转换、恶意检测、缓存等. 此外, 大部分中间件在运行时, 会记录处理过的数据包集合, 并根据记录的状态信息更改转发决策. 例如, 有状态防火墙在接收到可信主机发送给不可信主机的数据包之后, 允许后续来自该不可信主机的流量通过. 这类保存有状态信息的中间件被称为有状态中间件, 并将包含有中间件的网络称为有状态网络[45]. 由于有状态网络中的数据包转发行为与中间件的状态相关, 且中间件状态复

杂并具有任意性, 有状态网络下的意图验证变得非常困难甚至不可解[45]. 因此, 早期的网络验证研究主要关注于无状态网络. 随着无状态网络下的验证问题逐步得到解决, 有状态网络验证正成为网络验证领域的研究热点.

1.2 形式化方法

形式化方法 (formal methods) 是一种基于严格数学基础的技术, 用于对计算机硬软件系统规约、开发、验证[71]. 其中, 形式化意味着必须拥有可靠的、定义明确的数学基础 (如数理逻辑、自动机理论、图论), 这种数学基础可以提供方法用于证明系统的正确性[72]. 在传统的系统设计中, 采用了大量的测试 (testing) 用于验证系统正确性. 但是, 测试方法只能发现系统中存在的错误, 却不能证明系统不存在错误[73]. 形式化方法是传统系统测试方法的补充, 其严格的数学化证明可以更彻底地验证系统的属性. 虽然形式化方法带来了严格的正确性保证, 但是其也存在有一定的局限性, 如开发成本昂贵、建模程度有限等. 因此, 形式化方法往往应用于系统失败成本非常高的领域中[74]. 目前, 形式化方法已经在硬软件验证领域得到了成功的应用[7,8], 近年来正不断被应用到网络领域中用于验证网络系统的正确性, 并逐渐形成了网络验证这个新的研究领域.

形式化方法是网络验证领域的核心技术[60]. 目前, 网络验证主要使用了形式化方法中的形式化验证部分. 形式化验证是证明系统满足形式规约的过程, 通过输入系统描述和验证属性, 使用形式化验证技术进行验证[59]. 如图 3 所示, 形式化验证主要有以下步骤: 首先使用特定系统模型对系统进行建模, 并使用形式规约方法定义系统所需满足的属性, 最后使用验证方法进行验证得到验证结果. 对于不同的形式化验证技术, 系统建模和形式规约使用的方法也不相同. 系统建模一般使用通用形式建模语言或专用形式模型, 如有向图、形式化语言、自动机等; 形式规约一般使用逻辑公式, 如一阶逻辑、时序逻辑等[75].

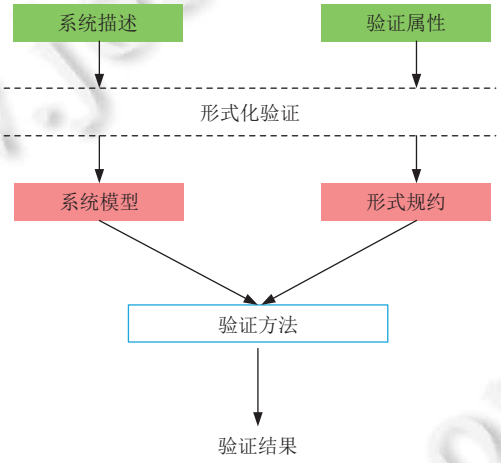


图 3 形式化验证步骤

形式化验证技术目前主要有模型检测[76]、符号执行[77]、定理证明[78]、SAT/SMT 求解器[79]、抽象解释[80]. 模型检测一般使用有限状态机 (如 Kripke 结构) 建模系统, 使用时态逻辑 (如计算树逻辑、线性时态逻辑) 公式建模属性, 然后遍历系统的有限状态空间判断公式是否得到满足, 以完成属性验证. 由于模型检测完成验证需要遍历整个状态空间, 存在状态爆炸问题, 符号模型检测[81]和有界模型检测[82]分别利用二叉决策图 (binary decision diagram, BDD) 和 SAT 求解器技术针对该问题进行了优化. 符号执行将系统建模成程序, 属性建模成程序断言, 然后输入符号值以遍历程序的路径空间, 并在遍历时记录所有的路径约束, 最后通过约束求解器 (如 SAT 求解器) 求解路径约束, 根据求解结果判断程序断言是否成立, 以完成属性验证. 与模型检测方法相似, 符号执行需要遍历整个路径空间, 面临着路径爆炸问题, 难以扩展到大型系统. 定理证明使用公式 (公理和推导规则集合) 建模系统和属性, 然后手动或使用定理证明器自动证明系统公式能否推导出属性公式, 以完成属性验证. 定理证明没有状态和路径概念, 不存在状态爆炸和路径爆炸问题, 但是其严重依赖于使用者的系统理解程度与数学经验, 需要准确构建表达系统性质的公式, 使用起来比较困难. SAT 求解器是求解布尔可满足性问题 (SAT) 的技术, 即给定一个命题

逻辑公式, 判定是否存在赋值使该公式为真. SMT 在 SAT 的基础上进行了扩充部分一阶逻辑的内容, 能够使用更加复杂的语言 (如算数运算符号) 描述问题[83]. 求解该类一阶逻辑公式可满足性问题的技术, 被称为 SMT 求解器. 在使用 SAT/SMT 求解器进行验证时, 首先生成表达系统和属性之间关系的逻辑公式, 然后放入求解器进行求解, 根据求解结果验证属性. SAT 是 NP 完全问题, 意味着任何 NP 问题可以被转换为 SAT 问题[59], 通用性强, 但是理论上求解比较困难. 抽象解释技术是一种对数学结构, 特别是涉及计算机系统语义模型的数学结构, 进行可靠抽象 (或近似) 的理论. 抽象解释的思想是只关注与验证问题相关的信息, 进行可靠的简化分析, 在分析精度和计算效率之间取得权衡. 抽象解释使用抽象域 (抽象语义以及其上的操作) 对系统和属性进行抽象, 然后在抽象域上进一步使用验证技术完成验证. 表 1 总结了上述验证技术的系统建模方法、属性建模方法、验证过程以及各自的优缺点. 需要注意的是, 上述验证方法并不是相互排斥的, 很多情况下会相互结合使用. 例如, 有界模型检测技术利用 SAT 求解器技术缓解状态爆炸问题; 符号执行技术采用 SAT 求解器技术求解路径约束; 抽象解释技术与模型检测技术结合使用进行验证[84] 等. 此外, 由于计算机网络结构本身是一种拓扑图, 网络验证研究还会将网络系统建模为图结构, 然后将验证属性转换为图相关的属性, 最后使用图算法完成验证.

表 1　形式化验证技术总结

验证技术	系统建模方法	属性建模方法	验证过程	优点	缺点
模型检测	有限状态机	时态逻辑公式	遍历状态空间判断公式是否得到满足	自动化的快速验证	状态爆炸问题
符号执行	程序	程序断言	遍历程序路径空间, 求解路径约束判断程序断言是否成立	使用少量输入完成高覆盖的测试	路径爆炸问题
定理证明	公式	公式	证明系统公式(M)是否可以推导出属性公式(S), 即M|–S	无状态爆炸问题, 应用范围广	使用复杂且不提供反例
SAT/SMT 求解器	逻辑公式	逻辑公式	使用求解器求解表达系统(M)与属性(S)的逻辑关系公式, 即M|=S	应用范围广	部分问题求解难度高
抽象解释	抽象域	抽象域	将抽象后的系统和属性作为新的验证对象, 进一步使用验证技术完成验证	计算效率高	损失分析精度

2 数据平面验证

数据平面验证是网络验证领域中的第 1 个热点研究方向, 通过分析数据平面和网络拓扑信息, 验证数据包转发行为与网络意图的一致性. 其中, 数据平面主要是指转发表或流表, 此外还包括访问控制列表 (access control lists, ACL) 等决定数据包转发的信息. 目前, 数据平面验证主要验证数据包转发相关的网络属性, 如可达性、无转发循环、区域流量隔离. 其中, 可达性用于保证网络设备之间的正常通信, 例如设备 A 发送的数据包可以正常到达设备 B; 无转发循环用于保证网络中不存在数据包转发循环; 区域流量隔离用于保证特定区域的网络设备不能相互通信, 例如办公区与公共区的网络设备不能相互访问. 需要指出的是, 网络属性与网络策略、网络意图一样, 用于表达期望达到的网络行为, 不同之处在于表达形式与表达粒度.

相比控制平面, 数据平面直接体现了数据包的转发行为, 有更好理解的语义. 因此, 数据平面验证相比控制平面验证更加容易实现, 不需要考虑控制平面中复杂的配置语言与协议交互过程. 不过, 由于数据平面验证分析的是数据平面快照信息, 且数据平面经常会发生变化 (例如接收到新的外部路由通告、链路状态改变或正处于路由收敛过程), 数据平面验证需要实现实时验证, 通过持续采集与验证数据平面信息, 保证数据平面行为与策略之间的实时一致性.

由于传统网络和 SDN 的数据平面都是转发规则信息, 两种网络架构下的数据平面验证所解决的问题在本质上是相同的, 都是通过分析转发规则信息验证网络策略. 因此, 同一数据平面验证方法可以在两个网络架构中得到使用. 两个网络架构下的数据平面验证过程的主要区别是数据平面的获取过程与数据格式, 在传统网络中可以通过终端或 SNMP 协议获取转发表信息, 在 SDN 中则可以通过控制器获取流表信息. 虽然 SDN 并没有使得数据平面验证问题变得更简单, 但其可以更方便地获取数据平面信息.

本节根据研究主要解决的问题, 将已有研究分成了离线验证研究、实时验证研究、高扩展性验证研究、高表

达性验证研究 4 个研究内容部分, 并结合时间顺序与验证技术对各个部分的研究进行了详细介绍. 其中, 扩展性是指研究工具处理大型网络验证任务的能力, 与验证速度和算法复杂度相关; 表达性是指工具的建模分析能力, 在数据平面验证中体现为数据平面行为的建模分析能力, 例如能够建模分析网络地址转换 (network address translation, NAT)、访问控制列表等特殊的转发功能.

2.1 离线验证研究

早期的数据平面验证研究, 开创性地使用不同形式化方法, 通过分析数据平面实现了网络属性的验证, 提供了数据平面验证的不同实现方案. 由于早期研究对验证速度要求较低, 主要使用离线验证的方法实现, 即不考虑数据平面的实时变化.

(1) 基于模型检测技术

Al-Shaer 等人于 2009 年提出了 ConfigChecker[19], 这是首个被正式提出的数据平面验证工具. 在此之前, 相关研究[85−88]专注于分析防火墙或 BGP 配置以发现错误情况, 但是这些研究的应用场景有限或无法验证数据包转发行为, 不能提供全面的网络行为分析. 与 ConfigChecker 最接近的研究为 Xie 等人于 2005 年提出的可达性分析算法[89], 该研究基于图算法分析数据平面以完成可达性验证. 但是, 该研究只提出了理论框架, 并未形成实际可应用的验证工具. ConfigChecker 与上述研究不同之处在于, 其通过使用模型检测技术, 在性能和应用范围上取得了提升, 能够全面地分析端对端网络行为, 验证可达性等网络属性, 且支持建模多种类型的网络设备. ConfigChecker 基于符号模型检测技术, 使用状态机建模网络系统, 将数据包的头部信息和所处位置建模为状态, 将数据平面转发信息与网络拓扑建模为状态转移关系, 然后遍历状态空间验证使用计算树逻辑 (computation tree logic, CTL) 表达的网络属性. 除了路由表, ConfigChecker 能够分析防火墙和 IPSec 的 ACL 信息, 从而支持防火墙等设备的建模以及 IPSec 相关安全属性的验证. 之后, Al-Shaer 等人在 ConfigChecker 的基础上, 针对 SDN 提出了 FlowChecker[20], 实现了 SDN 下的数据平面验证.

(2) 基于 SAT 求解器技术

Mai 等人于 2011 年提出了 Anteater[9], 是首个在真实网络中发现真实错误的数据平面验证工具. Anteater 与 ConfigChecker[19]解决的问题一致, 都旨在通过分析直接体现了网络转发行为的数据平面, 实现全面的网络行为分析, 避免局限于单个协议的正确性验证. Anteater 的验证过程主要借鉴了 Xie 等人提出的可达性验证算法[89], 即根据网络拓扑将网络系统建模为有向图, 将数据平面建模成有向边上的条件, 用于表示该有向边允许通过的数据包集合, 最后基于图算法和集合运算, 计算节点之间各路径允许通过的数据包集合, 若集合不为空则表示该路径可达. Anteater 对该算法的改进在于, 将数据包集合的计算过程使用 SAT 公式表示并放入 SAT 求解器进行求解, 能够更快确定数据包集合是否为空以判断可达性. 此外, Anteater 还在 Xie 等人[89]提出的算法上进行了改进, 能够验证转发循环、转发黑洞等更加复杂的网络属性, 且可以处理数据包转换行为, 如 NAT.

(3) 基于符号执行技术

Kazemian 等人于 2012 年提出了 HSA[10]. HSA 基于符号执行技术, 使用自定义的抽象符号即头空间 (header space) 表示数据包集合, 将数据平面与网络拓扑信息建模成头空间相关的转移函数, 将网络属性建模成头空间相关的断言. HSA 首先根据断言构造特定的符号数据包集合, 输入到转移函数进行分析, 然后通过分析转移函数对符号数据包集合的处理过程, 确定断言的满足性以完成属性验证. 相比于基于模型检测或 SAT 技术的验证方法[9,19,20], HSA 这种函数分析方法的优点在于可以找到违反网络属性的所有反例, 而不是单个反例. 例如, 网络设备 A 和 B 之间存在 ACL 规则设定 10.10.0.0/16 范围下的数据包都不允许通过. 在验证可达性时, HSA 能够直接返回 10.10.0.0/16 的反例, 而基于模型检测或 SAT 求解器技术的方法[9,19,20]只能返回其中的一个反例, 如 10.10.0.100, 仍需要进一步分析才能找到违反可达性的 ACL 规则. 当规则数量多且复杂时, HSA 这种给出全部反例的特点, 能够更好地帮助网络管理人员进行错误定位. 此外, HSA 还可以验证转发循环和区域流量隔离等网络属性.

(4) 离线验证工具总结

上述数据平面验证工具为该方向中早期的开创性研究, 分别使用模型检测、SAT 求解技术、符号执行这 3

种形式化方法, 通过分析数据平面信息完成了可达性等网络属性的验证. ConfigChecker[19]是首个正式发表的数据平面验证工具, Anteater[9]是首个在真实网络中发现真实错误的数据平面验证工具, 这两个工具的研究出发点和研究方法非常相似, 都注意到已有的通过分析配置的方法验证网络属性过于复杂, 往往只能局限于单个协议的正确性验证, 因此通过分析数据平面以避免复杂的协议和配置语言建模, 不仅扩大了验证的网络范围, 而且可以提供全面的网络行为分析. HSA[10]的研究出发点则是为了帮助网络管理人员更好地分析与管理网络, 其提供全部反例的特点能够很好地实现错误定位. 上述研究存在验证速度较慢的缺点, 需要数百甚至上千秒的时间才能完成大型网络下的验证.

2.2 实时验证研究

由于离线验证工具的验证速度普遍较慢, 难以及时地发现网络错误, 且无法处理数据平面的实时变化以保证验证结果的实时正确性. 为解决该挑战, 学术界陆续开展了实时验证的研究, 旨在开发出高效的验证算法, 能够在每次网络状态更新时实时地完成验证. 受益于集中式控制的网络架构, SDN 可以方便地获取数据平面的更新信息, 且可以通过拦截违反验证属性规则的下发, 提前阻止错误的发生. 受到分布式网络架构的限制, 实时验证在传统网络中实现起来相对比较困难.

(1) 基于等价类方法

Khurshid 等人于 2013 年提出了 VeriFlow[22], 是首个实现实时验证的数据平面验证工具. VeriFlow 将网络划分为等价类集合, 其中每个等价类是指在网络中具有相同转发行为的数据包集合. 这样, 整个网络的验证任务可以被分割成针对等价类的多个独立验证任务. 在数据平面变化时, VeriFlow 基于树结构的方法找出受到影响的等价类集合, 并为每个等价类建立转发图, 然后对各个转发图使用图算法验证网络属性. VeriFlow 利用了数据平面变化只影响小部分数据包集合转发行为的特点, 通过只验证受到数据平面变化影响的等价类, 实现了毫秒级的验证速度. 除了数据平面变化, VeriFlow 还可以处理网络链路状态变化情况. 不过, 由于网络链路状态变化影响的等价类数量较多, VeriFlow 需要数秒才能完成验证.

在 VeriFlow 被提出不久之后, Yang 等人于同年提出了更加高效的实时验证工具 Atomic Predicates Verifier (APV)[23]. Yang 等人指出, 网络的数据平面状态由网络设备的转发表和 ACL 规则决定, 可以使用断言 (predicates) 建模数据平面以表示网络设备各个端口允许进出的数据包集合. 然后, 通过计算转发路径上所有断言的交集, 根据集合是否为空即可判断该转发路径是否可达. 但是, 断言的交集计算实际上是数据包集合交并集的计算, 属于计算密集型操作, 最坏情况下的复杂度呈指数级别. 为解决该问题, APV 提出了一种新颖的方法简化了断言的交集计算. APV 首先使用 BDD 对断言进行表示, 然后基于 BDD 交集操作根据断言计算出一系列具有唯一性的原子断言 (atomic predicates), 并保证每个断言可以使用原子断言的并集进行表示. 其中, 每个原子断言表示一类具有相同转发行为的数据包集合, 也即是一种等价类. 由于每个原子断言具有唯一性, APV 使用数字对原子断言进行标识. 这样, 验证转发路径的可达性时, 断言的交集计算被转换成为数字集合之间的交集运算, 大幅简化了计算过程, 提升了可达性的计算效率. APV 的可达性验证思路与 Anteater[9]和 VeriFlow[22]的一致, 都是基于 Xie 等人提出的可达性算法[89], 通过计算路径上可以通过的数据包集合完成验证. 区别主要在于可达数据包集合的计算方式, Anteater 使用 SAT 公式表达与求解, VeriFlow 使用定制化方法计算, 而 APV 则使用数字集合计算简化了数据包集合计算. 此外, APV 使用哈希表和树结构等方法以应对链路状态和数据平面变化. 实验表明, 即使应对链路状态变化状况, APV 也可以达到毫秒级的验证速度.

(2) 基于增量计算方法

NetPlumber[21]是 VeriFlow[22]的同期研究, 基于增量计算方法实现了数据平面实时验证. NetPlumber 首先对网络系统的初始状态进行建模, 之后在每次数据平面变化时, 不重新建模系统, 只需要对系统模型进行小范围的更新. NetPlumber 将网络建模成有向图, 也可看作是一种规则依赖图, 图的节点表示数据平面中的转发规则, 图的边表示规则之间的依赖关系. NetPlumber 验证网络属性的过程基于 HSA[10], 通过分析头空间表示的数据包集合在规则依赖图的转移过程, 确定断言的满足性以完成属性验证. 当数据平面变化时, NetPlumber 对规则依赖图进行增量

更新, 并重新验证受到变化影响的网络策略. 除了实时验证, NetPlumber 还可以建模数据包的转换行为, 且提供定制语言用于表达所需要验证的策略, 但是其需要数秒才能验证链路状态变化后的情况. 与 VeriFlow 相同, NetPlumber 针对 SDN 下的网络属性验证, 可以被部署在控制器或控制器与交换机的链路上, 在规则下发之前的短时间内完成验证, 并在发现错误后阻止错误规则的下发.

(3) 基于增量计算和等价类方法

受到 APV[23]的启发, Wang 等人于 2015 年提出了验证效率更高的 AP classifier (APC)[25]. Wang 等人指出, 当数据平面发生变化时, APV 需要重新计算等价类, 影响了实时验证效率. APC 基于增量计算方法对 APV 进行了改进, 实现了等价类的增量更新, 提高了验证效率. APC 将 APV 的等价类计算过程使用二叉树表示, 树的每个叶节点表示一个等价类, 即具有相同转发行为数据包集合. 当数据平面变化时, APC 只对受到变化影响的二叉树节点更新, 避免了部分等价类的重复计算. 与其他数据平面验证工具不同, APC 专注于特定数据包的可达性验证, 即给定数据包头部信息, 判断该数据包是否能到达特定节点. APC 首先通过查找二叉树确定该数据包所属的等价类, 并记录查找路径上的断言集合, 然后根据断言集合对应的端口信息构建转发图, 最后使用图算法完成验证. 由于二叉树的平均深度影响了等价类查找效率, APC 对二叉树构建过程进行了优化, 通过调整树的构建顺序, 使得二叉树具有最小平均深度. 实验表明, 在特定数据包的查询任务上, APC 的验证速度相比 APV 提高了一个数量级, 且能够建模数据包转换行为.

虽然 APV[23]实现了较高的验证速度, 但是其等价类计算效率受到了 BDD 操作函数的限制, 且不能完成等价类的增量计算. 为解决上述问题, Bjørner 等人于 2016 年提出了 ddNF 数据结构[27]. ddNF 属于有向无环图, 图中的节点表示等价类, 并使用三元位向量 (ternary bit-vectors, TBV) 进行表示, 图中的边表示一种集合之间的差集运算关系. 由于 TBV 相比 BDD 具有更加紧凑的结构, 操作运算更加高效, ddNF 可以更快地计算等价类. 此外, ddNF 采用增量的方式计算等价类, 可以更好地处理数据平面变化情况. 作者分别基于 ddNF 和 BDD 对 APV[23]进行了实现, 经过实验, 基于 ddNF 的实现相比基于 BDD 至少快出了一个数量级.

Horn 等人于 2017 年提出了 Delta-net[28], 旨在实现高效计算等价类. Delta-net 创新性地将数据包集合使用数字区间进行表示, 将数据包集合之间的交集或差集计算, 转换成了简单的数字区间运算. Delta-net 使用平衡二叉树增量计算等价类, 然后将网络建模成有向图, 图的节点表示网络设备, 图的边使用等价类集合表示允许通过的数据包, 最后基于图算法验证网络属性. 在数据平面变化时, Delta-net 只对平衡二叉树和有向图中受到变化影响的部分进行更新. 相比 APV[23]或 ddNF[27]等计算等价类的方法, Delta-net 提升效率的关键之处在于, 其通过使用整数区间建模数据包集合, 可以使用高效的整数区间交集运算计算等价类, 达到了拟线性时间复杂度. 但是, 由于整数区间随着数据包头部位数指数变化, 存在区间状态爆炸的问题. 此外, Delta-net 存在不能建模数据包转换行为的缺点.

Horn 等人于 2019 年提出了等价类计算框架#PEC[32]. 文中指出, 已有的等价类计算方法都存在一定的局限性. 例如, APV[23]基于 BDD 的等价类计算存在计算效率瓶颈, 而基于 ddNF[27]的等价类计算虽然拥有更高的计算效率, 但是其不能确定等价类是否为空集, 导致出现误报的情况, 且受到了 TBV 数据结构低表达性的限制. #PEC 通过结合两种方法, 同时达到了 APV 的准确性和表达性, 以及 ddNF 的高计算效率. #PEC 使用哈斯图计算等价类, 图的节点为使用 TBV 表示的数据包集合, 图的偏序关系为子集包含顺序. #PEC 计算等价类采用增量计算的方式进行, 当插入或删除规则时, 根据偏序关系对哈斯图进行更新. 由于 TBV 具有紧凑的数据结构, #PEC 能够进行高效的数据包集合运算. 为了发现等价类的空集情况以保证验证准确性, #PEC 在每次节点插入时计算出对应的基数, 并对受影响的节点使用简单的减法进行更新. 相对于基于 BDD 或 SAT 求解器判断空集, 这种计数方法能有效减少操作次数与计算复杂度, 实现快速的空集判断. 通过确定空集, #PEC 能够和 APV 一样找到唯一且最小数量的等价类, 从而实现准确的验证. 为实现高表达性的验证, #PEC 使用一种抽象数据结构对 TBV 进行了优化, 使其能够表达多维度的任意范围匹配. 经过实验, #PEC 在计算等价类的过程上, 相比 APV 至少快了 10 倍.

(4) 实时验证工具总结

VeriFlow[22]和 NetPlumber[21]分别基于等价类和增量计算方法, 对离线验证工具 Anteater[9]和 HSA[10]进行了改进, 首先实现了数据平面实时验证. APV[23]创新性地使用数字对等价类进行标识, 将复杂的数据包集合计算转换为

数字集合计算, 有效地提升了验证速度. 然而, 当数据平面变化时, APV 需要重新计算等价类, 影响了验证效率. APC[25] 基于二叉树结构, 使用增量计算方法对 APV 的等价类计算过程进行了改进. 针对 APV 不能增量更新等价类以及基于 BDD 的等价类计算效率有限的缺点, Bjørner 等人提出了 ddNF[27]数据结构, 通过使用具有更加紧凑结构的 TBV 提升了等价类计算速度, 并基于有向无环图实现了等价类增量计算. Delta-net[28]创新性地使用数字区间建模数据平面, 将等价类计算过程中的数据包集合计算, 转换成了简单的数字区间运算, 有效提升了等价类计算速度, 但其建模方式存在区间爆炸的问题. #PEC[32]基于哈斯图方法实现了等价类空集的高效确定, 能够使用 ddNF 找到最小且唯一的等价类集合, 实现了在保证验证结果准确性的情况下提高等价类计算速度.

2.3 高扩展性验证研究

尽管实时验证工具已经达到了很高的验证速度, 可以在数据平面变化后的短时间内完成验证. 但是, 这些实时验证工具主要关注于校园网等中小型网络的验证, 在面临着包含数千或数万台网络设备的大型网络时, 会因为超时或内存不足而不能完成验证. 因此, 需要开发出具有高扩展性的数据平面验证工具, 实现大型网络的快速验证. 其中, 高扩展性验证除了需要较高的验证速度, 还需要具有较低的算法复杂度.

(1) 基于分布式计算技术

Zeng 等人于 2014 年提出了 Libra[24], 旨在解决已有验证工具无法扩展到大型数据中心网络的问题. Libra 基于 MapReduce 计算框架实现, 将验证任务划分成多个子任务, 通过分布式的并行计算, 实现了大型网络的高效验证. 由于数据包转发行为与网络结构高度关联, 根据网络结构进行划分难以得到独立的子任务. 一般情况下, 具有相同转发行为的数据包在一个子网中, 且不同子网的数据包转发行为相互独立. 因此, Libra 选择根据子网进行任务划分, 将子网和与子网相关的规则划分成一个独立的任务组, 每个任务组可以独立建立转发图, 然后使用图算法完成验证. 此外, Libra 支持增量计算, 进一步提高了验证效率. 经过实验, Libra 能够通过 50 台服务器的分布式验证, 在 1 min 内完成拥有 10 000 个网络设备的大型网络, 且验证时间随着网络规模增长而线性增长, 具有很好的扩展性. 除了实现高扩展性的验证, Libra 还提出了一种准确获取数据平面收敛状态的方法. Libra 通过记录数据平面变化时间, 当一定时间范围内数据平面不发生改变即判定为收敛状态.

(2) 基于模态逻辑的互模拟技术

Plotkin 等人于 2016 年提出了一种网络变换方法[26], 基于模态逻辑的互模拟技术, 利用网络结构对称性和数据包转发区域性的特点, 在保证变换前后的网络下验证结果相同的前提下, 将大型的数据中心网络变换成网络结构更加简单的网络. 在大型网络经过变换之后, 验证任务可以直接在变换后的简单网络中完成, 提高了验证效率. 该网络变换方法主要使用了对称变换和裁剪变换两种操作. 其中, 对称变换操作利用了数据中心网络结构对称的特点, 将具有相同转发行为的对称网络设备进行合并. 裁剪变换操作将与验证数据包转发行为无关的网络部分直接删除, 进一步地简化了网络结构. 实验表明, 在拥有约 10 万台虚拟机的大型数据中心网络中, 使用该变换方法实现了 65 倍的验证速度提升, 能够将 5.5 天的验证任务缩短到 2 个小时, 且可以通过并行计算将验证时间进一步缩短到数分钟之内.

(3) 基于网络结构特征

Jayaraman 等人于 2019 年提出了 RCDC[31], 该工具被部署在微软公司的大型数据中心网络 Azure 中以验证网络属性. 与其他数据平面验证工具不同, RCDC 验证的属性特别针对数据中心网络, 旨在保证服务器集群之间的通信总是选择最优路径, 从而实现最低延迟传输. 由于 Azure 网络高度结构化, 若每个层次的路由设备都符合预定义的最优路径转发模式, 整个网络也处于最优的转发状态. 那么, RCDC 所定义的属性验证可以转换成对各个层次路由设备的独立验证, 并可以使用并行计算的方法加速验证. RCDC 采用了基于 SMT 求解器的验证方法, 根据数据平面信息, 将验证属性建模为逻辑公式进行求解. RCDC 存在一个明显的缺点, 即应用范围有限, 只能验证特定结构化的数据中心网络.

(4) 高扩展性验证工具总结

上述研究通过高性能计算或简化验证任务, 实现了大型网络下的快速验证. Libra[24]基于 MapReduce 计算框

架, 根据子网将大型网络验证任务划分为多个独立的验证任务, 然后使用多个服务器进行分布式计算, 高效完成了单个机器无法完成的验证任务. Libra 可以通过使用等价类方法, 进一步优化子网和验证任务的划分过程. Plotkin 等人[26]利用网络结构和数据包转发特点, 通过变化网络结构简化了验证任务, 能够帮助验证工具提升验证速度. RCDC[31]针对验证具有特定结构的数据中心网络, 通过将全局属性的验证任务简化成为局部属性的验证任务, 实现了对大型网络的验证.

2.4 高表达性验证研究

除了高扩展性, 数据平面验证工具还需要实现高表达性, 用于建模验证真实网络中复杂多样的网络功能, 例如数据包头部转换、数据包封装与解封装、组播数据包等.

(1) 基于逻辑编程语言 Datalog

相较于实现实时验证, Lopes 等人[11,90]专注于实现一个能够对网络功能和网络策略广泛建模的工具. Lopes 等人指出, 已有的数据平面验证工具缺乏通用并且高级的策略建模语言, 难以简单地表达出复杂的网络策略, 且其通常使用硬编码的方式建模网络, 对新的网络功能建模时需要更改程序内核, 难以广泛地建模网络功能. 针对这些挑战, Lopes 等人于 2015 年提出了 NOD[11,90]. NOD 的创新之处在于, 其使用了逻辑编程语言 Datalog[91]建模网络和策略. 通用的 Datalog 语言使得 NOD 能够简单地表达所需验证策略, 而且支持对多种网络功能的建模, 如数据包的转换行为. 整体来看, NOD 首先将数据平面中的转发表和 ACL 规则建模成 Datalog 规则, 然后结合网络拓扑对 Datalog 表达的验证属性进行推理, 最后根据推理结果完成验证. 此外, NOD 还分别基于 BDD 和 TBV 数据结构对 Datalog 结构进行了优化, 实现了更快的验证速度. 然而, NOD 的验证速度不能完成实时验证, 稍慢于离线验证工具 HSA[10].

(2) 基于等价类方法

通过改进实时验证工具 APV[23], Yang 等人于 2017 年提出了具有更高表达性的数据平面验证工具 APT[29], 能够建模复杂的数据包转换行为. 相比 APV, APT 主要改进了等价类计算方法, 通过更加细粒度地建模和划分, 完成了数据包转换行为的建模. APT 除了建模转发规则, 还建模了数据包转换规则, 因此对等价类进行了重新定义, 要求等价类中的数据包在网络中不仅具有相同的转发行为, 还具有相同的数据包转换行为. 由于 APT 等价类划分粒度更细, 相对 APV 计算等价类的速度更慢. 除了数据包头部变换, APT 还能建模多协议标签交换 (multi-protocol label switching, MPLS) 这种数据包封装与解封装行为. 与 APV 相同, APT 不支持等价类的增量计算, 在数据平面变化时需要重新计算, 限制了验证效率.

Zhang 等人于 2020 年提出了 APKeep[16], 旨在实现同时具有高验证效率和高表达性的验证, 以适应具有复杂网络功能的真实网络. 为实现该目标, APKeep 提出了一种高表达性且支持增量计算的网络模型, 即 port-predicate map (PPM). PPM 将每个网络设备建模成 3 种独立的逻辑功能单元, 分别为转发、过滤和重写功能单元. 相比将网络设备建模成单一的模块, 这种细粒度的建模不仅能够有效表达数据包转换行为和厂商特定网络功能, 而且提高了增量计算效率. PPM 将网络建模成有向图, 图的节点为逻辑功能单元, 图的边为逻辑功能单元端口之间的数据平面信息, 并使用符号标识的等价类表示. APKeep 借鉴了 APT[29]的等价类划分思路, 以处理数据包转换行为. 此外, APKeep 采用增量计算的方式进行了优化, 每次数据平面变化时对等价类和 PPM 进行增量更新. 除了增量计算, APKeep 还提出了一种等价类合并方法, 基于 PPM 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, 使得在增量计算过程中等价类数量总是最少, 从而减少内存消耗且提高了验证速度. 实验结果表明, 在面对已有验证工具因为超时或内存不足的验证任务时, APKeep 仍然能够达到亚毫秒的验证时间, 具有很高的扩展性.

(3) 基于模型检测技术

Jensen 等人指出, 保证链路中断情况下的网络正确性是非常重要的, 但是已有数据平面验证工具往往无法完成链路中断情况的假设分析, 或因算法复杂度过高无法验证多条链路中断的假设情况. 此外, 由于 MPLS 网络支持可变化大小的数据包头部, 大多已有验证工具不能完成建模分析. 基于上述考虑, Jensen 等人于 2018 年提出了 P-Rex[30], 实现了 MPLS 网络中的链路中断情况的多项式时间验证. 基于 MPLS 网络根据标签而不是网络地址转

发的特点, P-Rex 使用下推自动机对网络和数据平面进行了建模, 并使用模型检测工具完成网络属性验证. 根据下推自动机中的前缀重写系统 (prefix-rewriting systems) 理论, 文中证明了 P-Rex 能够以多项式的时间完成任意数量链路失效的可达性验证. 实验表明, 相比离线验证工具 HSA, P-Rex 达到了与其相当的验证速度, 并且具有更高的扩展性. P-Rex 的缺点在于, 其验证速度不足以提供实时验证, 且只能适用于 MPLS 网络.

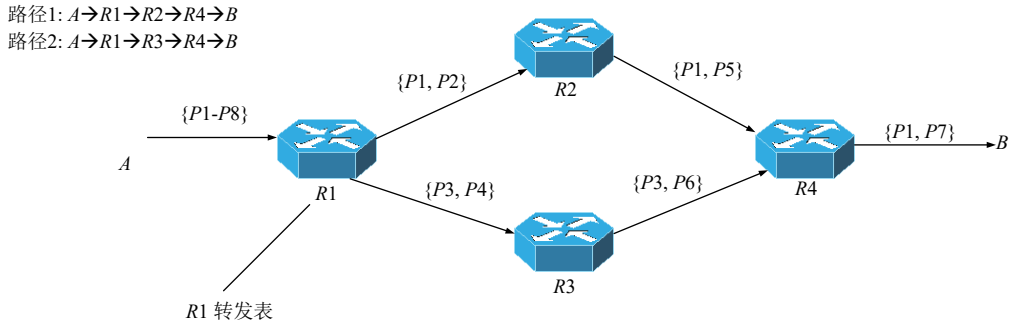
(4) 高表达性验证总结

NOD[11]是首个致力于提高表达性的数据平面验证工具, 通过使用通用的 Datalog 语言建模验证, 达到了较高的表达性, 并且可以方便地扩展新的网络功能, 但存在验证速度不足的问题. APT[29]通过改进实时验证工具 APV[23], 使用细粒度的等价类划分, 实现了数据包转换行为的建模. APKeep[16]在 APT 的基础上完成了进一步改进, 除了细粒度的等价类划分, 还实现了细粒度的网络建模, 达到了更高的表达性. 此外, APKeep 创新性地提出了一种等价类合并方法, 通过减少等价类数量, 有效提升了扩展性. 针对已有研究不能建模 MPLS 网络以及不能完成任意链路中断情况假设分析的问题, P-Rex 基于下推自动机理论进行了解决, 且达到了多项式时间的复杂度. 但是, P-Rex 存在验证速度较低以及验证范围受限的问题.

2.5 总 结

本节根据研究主要解决的问题, 将数据平面验证研究分成了离线验证、实时验证、高扩展性验证和高表达性验证 4 个研究内容部分, 并结合时间顺序和验证技术具体介绍了各个部分的研究内容. 离线验证是最早出现的数据平面验证研究, 创新性地使用不同的形式化方法, 通过分析数据平面实现了网络属性验证, 提供了实现数据平面验证的不同思路. 但是, 离线验证研究未考虑数据平面的实时变化, 不能保证验证结果的实时正确性. 实时验证研究使用等价类和增量计算技术, 通过只验证或建模受到数据平面变化影响的部分, 实现了数据平面实时验证. 高扩展性验证研究旨在实现大型网络下的快速验证, 使用了不同技术用于提升验证速度或优化算法复杂度. 高表达性验证研究旨在处理真实网络中复杂的网络功能, 以验证多种类型的网络. 需要指出的是, 部分的数据平面验证研究, 特别是前沿研究, 同时解决了多个研究部分的问题. 对于这些可以被划分到多个研究部分的研究, 本文根据其侧重解决的问题进行划分. 例如, APKeep[16]为实时验证工具, 且通过增量计算与创新的合并方法实现了较高的扩展性, 但其更加侧重于提高表达性以适应真实网络设备复杂的网络功能, 因此被划分到高表达性验证研究.

下面对数据平面验证的实现和优化方法进行总结. 实现方面, 数据平面验证与形式化验证一样, 主要分为系统建模、属性建模、属性验证 3 个流程, 其建模与验证方法取决于使用的验证技术. 例如, 基于模型检测实现时使用状态机建模数据平面等信息, 然后遍历状态空间验证使用逻辑公式表示的网络属性; 基于图算法实现时使用图模型建模数据平面等信息, 然后使用深度优先搜索等图算法验证网络属性. 目前, 数据平面验证工具主要验证可达性以及相关网络属性, 且可达性验证大多基于 Xie 等人[89]提出的算法实现. 图 4 举例阐述了数据平面验证中可达性验证的主要实现思路, 当验证节点 A 与节点 B 之间的可达性时, 首先使用图算法找到节点 A 和节点 B 之间的所有路径, 然后将数据平面信息建模为每条有向边上的信息, 用于表示该有向边允许通过的数据包集合, 最后对每条路径使用交集运算找到路径允许通过的数据包集合, 若数据包集合不为空, 则表示该数据包可经由该路径从节点 A 到达节点 B, 从而完成可达性的验证. 已有数据平面验证研究大多基于该思路实现可达性验证, 区别主要在于数据包集合的表达与数据包集合之间的交集计算操作, 例如 Anteater[9]使用 SAT 公式表示数据包集合并使用 SAT 求解器求解数据包交集是否为空集、HSA[10]使用头空间表示数据包集合并使用头空间操作函数计算数据包交集、APV[23]使用数字集合表示数据包集合并直接采用数字集合交集计算得到数据包交集等. 优化方面, 已有研究主要采用等价类方法和增量计算提升验证速度. 其中, 等价类方法是一种网络划分方法, 将具有相同转发行为数据包划分为同一等价类. 经过等价类划分, 验证工具可以对每个等价类单独进行建模验证, 且能够通过只建模验证与验证策略相关或受到数据平面变化影响的等价类, 实现快速验证. 增量计算是一种计算思想, 仅对受到数据平面变化影响的部分进行计算和或更新, 以节省计算时间, 例如只更新受到变化影响的系统模型部分, 只验证受到变化影响的网络策略等. 此外, 提高扩展性的技术还有分布式计算、网络结构变换等. 对于表达性提升, 已有研究主要通过细粒度的建模实现, 例如 APT[29]通过细粒度的等价类划分建模了数据包转换行为, APKeep[16]通过细粒度的网络设备建模处理厂商特定的网络功能.



Destination	Next hop	Interface
10.11.0.0/16 (*P1*)	*R2*	GE1/0/0
10.12.0.0/16 (*P2*)	*R2*	GE1/0/0
10.13.0.0/16 (*P3*)	*R3*	GE2/0/0
10.14.0.0/16 (*P4*)	*R3*	GE2/0/0

路径 1 可通过数据包集合 *A*: {*P*1-*P*8}∩{*P*1, *P*2}∩{*P*1, *P*5}∩{*P*1, *P*7}
路径 2 可通过数据包集合 *B*: {*P*1-*P*8}∩{*P*3, *P*4}∩{*P*3, *P*6}∩{*P*1, *P*7}
节点 *A* 与节点 *B* 之间可达: {*A*∪*B*}≠ø

图 4　数据平面验证中可达性验证的主要实现思路

表 2 对数据平面验证研究进行了总结, 分为了 4 个方面进行说明, 分别是基于技术、扩展性、表达性和简要总结. 其中, 基于技术是指工具主要使用的建模和验证方法; 扩展性是指工具处理大型网络验证任务的能力, 与验证速度和算法复杂度相关; 表达性是指工具的数据平面行为建模分析能力; 简要总结部分对各个研究的关键内容或创新处进行了简要说明.

表 2　数据平面验证技术总结

工具	基于技术	扩展性	表达性	简要总结
ConfigChecker[19]	模型检测	低	中	首个正式提出的数据平面验证工具
Anteater[9]	SAT求解	低	中	首个在真实网络中发现错误的数据平面验证工具
HSA[10]	符号执行	低	中	使用基于函数的分析方法, 能够找到违反验证属性的全部反例
VeriFlow[22]	图算法	中	低	使用等价类方法划分网络, 只对受到数据平面变化影响的等价类建模验证
APV[23]	图算法	高	低	划分等价类后, 使用数字集合替代数据包集合进行计算, 有效提升了验证速度
NetPlumber[21]	符号执行	中	中	基于HSA[10]进行改进, 对系统模型增量更新, 以及增量验证网络策略
APC[25]	图算法	高	低	基于APV[23]进行改进, 实现了等价类的增量计算, 提高了验证速度
ddNF[27]	—	—	—	一种数据结构, 相比基于BDD计算等价类更加高效, 且可以增量计算等价类
Delta-net[28]	图算法	中	低	使用数字区间表示数据包集合, 优化了等价类计算过程, 但存在区间爆炸问题
#PEC[32]	—	—	—	一种等价类计算方法, 基于ddNF[27]实现, 通过高效判断等价类是否为空, 实现了准确高效的等价类计算
Libra[24]	图算法	高	中	基于MapReduce计算框架, 根据子网将验证任务划分为多个独立的验证任务, 通过分布式计算实现了高效验证
Plotkin等人[26]	—	—	—	一种网络变换方法, 基于网络结构对称性与数据平面行为等价性简化网络结构, 并保证变换前后网络中的属性验证结果相同
RCDC[31]	SMT求解	高	低	针对特定结构的数据中心网络, 将全局属性的验证任务简化成为局部属性的验证任务
NOD[11]	Datalog	低	高	使用通用的Datalog语言有效建模了多种复杂的网络属性与网络功能, 但验证速度较慢
APT[29]	图算法	高	高	基于APV[23]进行改进, 使用细粒度的等价类划分, 实现了数据包转换行为的建模
APKeep[16]	图算法	高	高	基于APT[29] 实现了数据包转换行为的建模, 使用细粒度的网络建模实现了较高的表达性, 并通过找到最少等价类数量提升了验证速度
P-Rex[30]	模型检测	中	中	实现了多项式时间复杂度的任意链路失效情况的可达性验证, 但只能适用于MPLS网络

3 控制平面验证

控制平面验证通过分析控制平面信息、网络拓扑与环境信息, 验证在这些信息结合生成的数据平面下, 所有的数据包转发行为与网络意图一致. 与数据平面验证一样, 控制平面验证可以验证可达性、无转发循环等数据包转发相关的网络属性. 除此之外, 控制平面验证可以通过控制输入的环境信息, 对网络环境作出假设, 从而完成假设网络环境下的网络属性验证, 保证网络属性在未来可能出现的网络情况中依然能够成立. 例如, 保证在任意 k 条链路失效的情况下, 特定网络设备之间的可达性始终能够成立.

相比数据平面验证, 控制平面验证最具有吸引力的地方在于, 可以在网络配置部署之前完成验证, 提前阻止错误的发生. 此外, 控制平面验证通过直接分析配置文件, 可以在发现错误之后更加有效地定位到错误配置. 但是, 控制平面验证需要考虑不同厂商网络设备的实现差别 (如不同格式的配置语言) 以及复杂的协议交互过程. 此外, 控制平面在某些情况下存在多个收敛数据平面, 且其最终收敛到的数据平面是不确定的, 例如 BGP Wedgies[92]. 控制平面验证需要对所有可能的收敛状态, 甚至对收敛过程中的过渡状态进行验证, 以保证网络意图在网络所有可能出现的数据平面中都能得到满足. 因此, 控制平面验证相比数据平面验证实现起来更加困难.

由于传统网络中和 SDN 的控制平面格式与表示含义都不相同, 两个网络架构下的控制平面验证所解决的验证问题并不相同. 传统网络中的控制平面验证通过分析分布在各处网络设备中的配置文件验证网络策略; SDN 中的控制平面验证则通过分析控制器或应用中的程序验证网络策略, 验证方法与软件程序验证较为相似. 因此, 两种网络架构下的控制平面验证方法的差别较大. 由于传统网络目前仍然占据主要地位, 本文只介绍传统网络中的控制平面验证技术.

本节根据研究主要解决的问题, 将控制平面验证方向中的已有研究分成了开创性研究、高扩展性验证研究、高表达性验证研究 3 个研究部分, 并结合时间顺序和验证技术对各个部分的验证技术进行了详细介绍. 其中, 扩展性是指研究工具处理大型网络验证任务的能力, 与验证速度和算法复杂度相关; 表达性是指工具的建模分析能力, 在控制平面验证中体现为控制平面行为的建模分析能力, 例如能够建模分析所有网络协议、处理不同厂商的控制平面实现差异、分析所有可能收敛的数据平面状态等.

3.1 开创性研究

早期的控制平面验证研究, 创新性地使用不同技术, 通过分析控制平面实现了网络属性的验证, 提供了控制平面验证的不同实现方案.

(1) 基于逻辑编程语言 Datalog

Fogel 等人于 2015 年提出了 Batfish[12], 是首个实现控制平面多协议分析的工具, 之前的配置分析工具[85−88]局限于建模分析特定的协议, 验证范围有限, 且不能对网络行为进行全面的分析. Fogel 等人指出, 分析配置文件可以提前找出错误, 但是根据配置直接分析语义复杂的控制平面非常困难; 分析语义易理解的数据平面可以简单验证多种网络属性, 但是不能提前阻止错误的发生, 且难以定位到对应的错误配置. 为结合配置分析和数据平面分析两种方法的优点, 实现既能提前检测错误又能简单验证多种网络属性, Batfish 根据控制平面与环境等信息, 模拟生成出完整的数据平面, 然后使用数据平面验证工具完成验证. 其中, Batfish 最关键的挑战在于准确生成数据平面. Batfish 基于控制平面等信息, 使用逻辑编程语言 Datalog 建模控制平面协议行为 (例如路由通告、选择、过滤等过程), 然后执行 Datalog 程序模拟协议运行过程, 生成数据平面信息. 生成数据平面之后, Batfish 使用同样基于 Datalog 实现的数据平面验证工具 NOD[11] 验证网络属性, 然后根据验证结果生成相关数据包转发信息, 帮助网络管理人员定位错误. 此外, 为解决不同厂商的配置语言问题, Batfish 基于 ANTLR 语法分析器[93]将配置解析为统一格式, 以便于控制平面的建模. 由于 Batfish 需要完整的模拟生成数据平面, 存在一定的扩展性问题. 为提高验证效率, 最新版本的 Batfish[94]使用定制化的算法替代了基于 Datalog 的数据平面生成与验证过程.

(2) 基于图算法

Gember-Jacobson 等人于 2016 年提出了 ARC[33], 使用定制化的图算法, 通过直接分析控制平面完成了网络属性的高效验证. 文中指出, 验证可达性等转发行为相关的网络属性时, Batfish[12]这种生成完整的数据平面进行分析

是不必要的. 特别是对于任意链路失效环境下的验证, Batfish 需要对每种链路环境模拟生成数据平面, 验证速度非常慢. ARC 基于控制平面与环境等信息, 对每对源和目的子网建立一个加权有向图, 图的节点表示路由进程, 图的有向边表示路由进程之间的通信路径, 边上的加权值表示路由选择的优先级. ARC 保证图中的转发路径与真实网络一致, 并且图中源和目的节点之间的最小带权路径, 即为真实的数据转发路径. ARC 将可达性等网络属性的验证转换成加权有向图上的图属性验证, 可以使用图算法快速完成验证. 例如, 验证任意 k 条链路失效情况下的可达性, 只需验证源节点和目的节点之间的边不相交路径大于 k 即可. 经过实验, 对于任意链路失效环境下的可达性验证, ARC 相比 Batfish 快出了 3–5 个数量级. ARC 能够完成高效验证的原因关键在于, 对于任意链路失效环境, ARC 可以直接使用图算法进行验证, 而 Batfish 则需要对每个环境生成数据平面再进行验证. ARC 为达到高验证效率, 只对常见的路由协议完成了建模, 存在表达性不足的问题.

(3) 基于 BDD

Fayaz 等人于 2016 年提出了 ERA[34], 旨在通过直接分析控制平面实现高效且具有高表达性的验证, 以解决 Batfish[12]验证速度慢和 ARC[33]表达性不足的问题. 为实现高表达性, ERA 使用统一的数据格式建模不同协议的路由信息, 为实现高验证速度, ERA 抽象掉与验证属性无关的协议路由信息, 然后基于高效的 BDD 数据结构建模控制平面的协议行为, 并使用卡诺图 (Karnaugh map) 和等价类方法进行了优化. ERA 首先找到源和目的节点之间的各个路径, 然后根据控制平面和环境信息, 使用 BDD 分别对各个路径的路径通告和选择等协议行为进行建模, 最后结合生成的路由信息、路径上的静态路由信息和 ACL 信息验证网络属性. 实际上, ERA 也是通过模拟控制平面协议行为生成数据平面信息再进行分析. 与 Batfish 的区别在于, ERA 只生成必要的数据平面信息, 并基于 BDD 等技术实现了更高效的验证. 经过实验, ERA 在同一网络属性验证上相比 Batfish 快出了 23 倍. ERA 相比 Batfish 达到了更高的验证速度, 相比 ARC 具有更高的表达性. 但是, ERA 实际上验证速度不及 ARC 且表达性不及 Batfish.

(4) 开创性研究总结

上述工具为控制平面验证方向中开创性研究, 通过模拟数据平面或直接分析控制平面实现了网络属性验证. Batfish[12]通过模拟生成数据平面, 实现了多协议行为的建模. 由于 Batfish 需要完整生成数据平面, 存在扩展性不足且难以处理多环境情况下的网络属性验证. ARC[33]使用加权有向图建模控制平面, 通过定制的图算法直接分析控制平面完成了高效验证, 但只能建模有限的网络协议行为, 表达性较低. ERA[34]使用统一的数据结构建模控制平面实现了相对 ARC 更高的表达性, 使用 BDD 数据结构与高级别的抽象实现了相对 Batfish 更高的验证速度. 但是, ERA 的表达性和扩展性都未达到较高的水平, 且其基于路径的分析方法存在准确性问题. 总的来看, 上述研究在表达性和扩展性之间做出了取舍, 难以同时达到高表达性和高扩展性.

3.2 高扩展性验证研究

与数据平面验证一样, 控制平面验证面临着验证大型网络的挑战. 为完成包含上千甚至上万台网络设备的大型网络下的验证任务, 陆续出现了高扩展性的控制平面验证研究, 致力于实现高验证速度和低算法复杂度.

(1) 基于控制平面等价性的网络压缩

Beckett 等人于 2018 年提出了 Bonsai[36], 可以在保证不影响验证结果的前提下, 将大型网络压缩成规模更小的抽象网络. Bonsai[36]基于稳定路径问题[95]和路由代数理论[96]提出了一种建模网络控制平面行为的代数方法 SRP (stable routing problem). 针对单个子网, Bonsai 使用 SRP 对网络进行建模, 若抽象前后网络的 SRP 满足指定的等价性约束条件, 则能保证特定网络属性的验证结果在抽象前后的网络中保持一致. Bonsai 基于网络结构的对称性进行抽象, 在满足等价性条件的前提下, 合并结构对称的网络节点以缩小网络规模. 此外, Bonsai 使用等价类技术和 BDD 结构优化了网络抽象的过程. 实验表明, Bonsai 可以帮助控制平面验证工具提升数个数量级的验证速度. Bonsai 的抽象思路与数据平面验证方向中 Plotkin 等人[26]提出的方法非常相似, 都是利用网络的对称结构对网络进行压缩. 区别在于, Bonsai 基于控制平面行为等价性进行抽象, 而 Plotkin 等人的方法是基于数据平面行为等价性进行抽象.

由于 Bonsai[36]会改变网络拓扑结构, 抽象后网络的单条链路可以表示原网络的多条链路, 难以进行任意 k 条链路失效情况下的可达性验证. 为解决该问题, Giannarakis 等人基于 Bonsai 进行了改进, 于 2019 年提出了 Origami[39]. Origami 面临的关键挑战在于, 需要对网络进行合适的抽象, 既要抽象后的网络结构足够小以加快验证速度, 又要

抽象后的网络结构足够大以提供足够的链路用于链路环境分析. 与 Bonsai 一致, Origami 使用代数方法建模网络控制平面行为以及抽象前后的网络等价性判断, 区别在于 Origami 额外考虑了链路失效数量. Origami 采取了抽象查找与属性验证相结合的方式, 以找到合适大小的网络抽象. Origami 首先使用 Bonsai 找到最小的网络抽象, 然后使用 SMT 求解方法验证链路失效情况下的可达性, 若可达性成立则表示验证完成, 否则根据反例对网络抽象进行调整并重新进行验证, 直到验证成功或不能再进行抽象. 实验表明, Origami 可以将网络链路数量缩小 1–3 个数量级, 帮助已有验证工具完成之前无法完成的验证任务.

(2) 基于抽象解释

Beckett 等人于 2018 年提出了 ShapeShifter[37], 基于抽象解释技术, 实现了接近线性的复杂度. 文中指出, 扩展性是控制平面验证的关键挑战, Batfish 等控制平面验证工具[12,13,33,34]的验证时间和内存消耗, 随着网络规模的增加呈高度的非线性增长, Bonsai[36]能够被验证工具用于提高扩展性, 但当网络结构不对称时难以发挥作用. ShapeShifter 基于路由代数理论[96]对控制平面的协议行为进行建模, 然后分析路由代数表示的控制平面收敛结果验证可达性. 相比于完全建模协议属性, ShapeShifter 将复杂的协议属性抽象成为简单的属性, 或不建模特定属性, 通过牺牲一定的验证精确度, 换取更少的内存消耗与更高效的协议计算过程, 从而实现复杂度的优化. 例如, ShapeShifter 使用简单的可达性标记替换掉 BGP 协议的 AS path 属性, 可以在简化建模的同时不影响可达性的验证. 实验表明, ShapeShifter 相比 Batfish 在可达性验证上快出了 1–2 个数量级, 其优化效果随着网络规模的增加得到明显提升. 并且能够在 95% 的情况下准确验证. ShapeShifter 的抽象不受到网络结构限制, 但是相比 Bonsai 不损失验证结果准确度的抽象, ShapeShifter 只能保证抽象后验证结果的可靠性, 即可以保证不会漏报网络错误, 但可能会出现误报情况. 此外, ShapeShifter 只能验证可达性, 且不能处理控制平面有多个收敛数据平面的情况.

(3) 基于网络结构特征

Lopes 等人于 2019 年提出了 FastPlane[38], 能够快速完成 BGP 协议环境下的数据平面收敛. FastPlane 利用了 BGP 路由信息在传播过程中优先级单调递减的特点, 基于 Dijkstra 最短路径算法, 在模拟路由通告过程时, 每次选择优先级最高的路由进行通告, 避免了低优先级路由的无效通告过程, 从而实现快速模拟. FastPlane 的缺点在于只能应用于 BGP 协议, 不能处理控制平面多收敛数据平面的情况, 且只适用于具有单调特征的网络, 即路由信息的优先级需要在传播过程中不断递减. 经过实验, FastPlane 计算路由表和转发表的过程相比 Batfish[12]快出了两个数量级, 可处理包含上千台网络设备的大型网络.

(4) 基于模型检测

在控制平面验证工具 Plankton[35]的基础上, Prabhu 等人于 2020 年提出了新版本的 Plankton[40]. 两个版本的 Plankton 实现思路基本一致, 都是基于模型检测技术对等价类的控制平面行为建模和验证. 区别在于, 新版本的 Plankton 更加注重于扩展性, 只对收敛的数据平面进行验证, 且提出了更多的优化技术用于提高扩展性与验证结果准确性, 如使用 Bonsai[36]减少搜索空间范围, 考虑等价类之间的依赖关系等. 经过实验, 相比同样可以验证所有收敛数据平面状态的 MineSweeper[13], Plankton 的验证速度快出了 4 个数量级. 文中指出, Plankton 取得高扩展性的原因在于采用了相比 SMT 求解技术速度更快的模型检测方法, 且使用等价类等技术进行了优化.

(5) 基于增量计算

Zhang 等人于 2020 年提出了控制平面增量验证工具 RealConfig[42]. 文中指出, 现代网络配置更新频繁, 已有控制平面验证工具在每次配置更新都需要重新进行建模验证, 即使网络变化的程度非常小. 为解决该挑战, RealConfig 基于增量技术实现了增量控制平面验证, 主要通过增量数据平面生成和增量数据平面验证实现. 为实现增量数据平面生成, RealConfig 基于 DDlog (differential Datalog)[97]建模控制平面的协议行为. DDlog 是逻辑编程语言 Datalog[91]的改进版本, 在 Datalog 的基础上支持了增量计算, 使得 RealConfig 可以根据控制平面的更新, 增量计算数据平面, 实现高效的数据平面模拟. 为实现增量数据平面验证, RealConfig 基于已有的数据平面增量验证工具 APKeep[16]完成验证. 为能够验证多种网络属性, RealConfig 采取了 Batfish[12]模拟生成数据平面然后进行数据平面验证的思路, 并使用增量技术对模拟和计算过程进行了改进. 实验表明, 相比使用定制化算法版本的 Batfish[94], RealConfig 在配置更新情况下的数据平面生成过程上快出了 1–2 个数量级.

Li 等人于 2020 年提出了 NUV[15], 旨在找到受配置更新影响的验证策略, 以帮助已有控制平面验证工具增量验证网络策略以提高验证速度. NUV 使用自定义的代数方法对各个流量的控制平面协议行为进行建模, 并将配置的更新情况转换成对应的代数条件, 然后根据代数条件判断受到更新影响的流量, 最后根据流量找到受到更新影响的网络策略. 此外, NUV 使用 Bonsai[36]的等价性判断方法判断更新前后流量转发行为的等价性, 对受到更新影响的流量进一步的筛选, 实现了更精确的网络策略查找. 经过实验, NUV 可以将 MineSweeper[13]的验证速度提高 2 个数量级.

(6) 高扩展性验证研究总结

上述研究基于不同技术实现了较高的扩展性. Bonsai[36]利用网络结构的对称性, 将具有相同控制平面行为的节点合并, 实现网络结构的简化. Origami[39]在 Bonsai 的基础上进行了改进, 结合抽象查找和属性验证过程, 使得抽象后的网络既能加速网络验证, 又能支持链路失效环境的假设分析. ShapeShifter[37]使用抽象解释技术简化了控制平面协议行为的建模, 通过牺牲验证结果的准确性实现了验证算法的复杂度优化. Bonsai、Origami、ShapeShifter 都是基于路由代数理论[96]建模控制平面以及证明抽象后验证结果的准确性. 其中, Bonsai 和 Origami 能够保证抽象后的验证结果完全准确, 而 ShapeShifter 只能保证验证结果的可靠性. FastPlane[38]利用单调网络中 BGP 路由信息在传播过程中优先级递减的特点, 优化了路由通告过程, 实现了数据平面的快速模拟, 但其只能应用于具有单调特征的 BGP 网络. Plankton[40]通过结合模型检测和等价类技术, 实现了显著高于当时最先进验证工具 MineSweeper[13]的验证速度. RealConfig[42]基于已有技术 DDlog[97]和 APKeep[16], 实现了控制平面增量验证. NUV[15]基于自定义的代数方法查找受到配置更新影响的验证策略, 可被其他控制平面验证工具用于提升验证速度. RealConfig 和 NUV 都针对于解决网络配置频繁更新的问题, 通过增量验证避免了不必要的重复建模或验证过程.

3.3 高表达性验证研究

控制平面验证除了需要较高的扩展性以处理大型网络的验证任务, 还需要较高的表达性以处理实际的复杂网络情况. 其中, 控制平面验证的表达性与控制平面和环境信息的建模能力相关. 控制平面信息是指由配置文件实现的各种网络协议, 如 BGP、OSPF、RIP 等. 控制平面建模能力越高, 表示越能准确建模越多的网络协议行为, 越能实现准确且覆盖范围高的验证. 由于不同厂商的网络设备在实现相同的网络协议时, 其配置语言甚至实现形式都各不相同, 需要考虑不同厂商设备的这种实现差异. 此外, 控制平面可能存在多个收敛数据平面, 且在收敛过程中会产生多个数据平面过渡状态, 需要对这些收敛和过渡数据平面情况准确建模. 环境信息是指链路状态、外部路由通告等影响数据平面生成结果的网络信息. 环境信息建模能力越高, 表示越多的网络情况能够得到分析, 例如可以保证任意 k 条链路失效情况下可达性成立, 或任意外部路由信息不会引起转发环路.

(1) 基于模型检测

Prabhu 等人于 2017 年提出了 Plankton[35], 旨在验证控制平面所有可能生成的数据平面状态, 包括所有的收敛状态以及收敛过程中的过渡状态. 由于在某些情况下控制平面可能存在多个收敛数据平面, 且路由传播等协议行为以任意顺序的方式执行, 同一数据平面状态存在多个不同的收敛路径. 为全面保证网络的正确性, 需要对收敛路径上所有过渡状态的数据平面进行验证. 文中指出, Batfish 等控制平面验证工具[12,33,34]都未考虑过渡状态的验证, 且不能处理控制平面存在多个收敛数据平面的情况. 为解决上述挑战, Plankton 基于模型检测技术详细地建模了所有的数据平面过渡与收敛状态. Plankton 首先将网络划分为多个等价类, 然后使用状态机建模等价类的控制平面协议行为. 其中, Plankton 借鉴了数据平面验证中的等价类划分方法, 区别在于 Plankton 中的等价类是指具有相同控制平面行为的控制平面消息. Plankton 使用状态机建模时, 状态机的状态表示每次协议行为执行后生成的数据平面信息, 状态之间的转移表示选择执行的协议行为. 建模完成后, Plankton 使用显式模型检测工具对每个状态进行遍历, 每当状态的数据平面信息发生变化时, 使用数据平面验证工具验证网络属性. 此外, Plankton 使用偏序规约、哈希等处理模型检测状态爆炸的技术提高验证速度, 能够在可接受时间内完成中型网络的验证.

(2) 基于 SMT 求解器技术

Beckett 等人于 2017 年提出了 Minesweeper[13], 旨在支持建模多种网络协议与环境信息, 同时能够验证所有可能生成的数据平面情况. 文中指出, Batfish[12]只能分析单个数据平面情况, ARC[33]和 ERA[34]存在建模路由协议不足的问题. 此外, 由于路径之间的协议行为会相互影响, ERA 基于路径的分析方法在某些情况下的验证结果不准

确. Minesweeper 根据控制平面等信息, 将控制平面协议行为和网络属性建模成逻辑公式, 使用 SMT 求解器进行求解验证. 其中, Minesweeper 旨在使用求解器找到一个违反网络属性的行为 (例如一条不被允许出现的转发路径), 只有当求解结果不满足时网络属性才能够成立. 使用逻辑公式建模的好处在于, 既能对多种网络协议行为进行建模, 也能够验证所有可能生成的数据平面情况 (只有所有数据平面情况都满足属性时, 才不会找到反例). 为保证结果的准确性, Minesweeper 采用了基于图的协议行为建模方法. 为提高验证速度, Minesweeper 不对收敛过程中的过渡状态进行验证, 且提出了一些优化建模和验证过程的方法, 如忽略对不影响验证结果的协议属性建模. 实验表明, Minesweeper 可以在真实网络中实现快速验证.

Ye 等人于 2020 年提出了 Hoyan[17], 是首个考虑不同厂商设备协议行为实现差异的控制平面验证工具, 且有足够的扩展性与表达性快速完成大型广域网中不确定情况 (例如多收敛数据平面, 任意链路失效) 下的网络属性验证. 文中指出, 不同厂商对网络协议的理解不同, 导致不同厂商的网络设备使用不同的方法实现相同的网络协议. 为了准确建模控制平面的协议行为, 需要对各个厂商设备的协议行为进行准确建模. 然而, 由于开发者无法获取厂商设备的协议行为实现方法, 准确建模变得非常困难. 此外, 已有的验证技术存在一定的扩展性与表达性问题. 例如, Batfish[12]需要完整生成所有情况下的数据平面以验证不确定情况下的网络属性, 验证速度非常慢; MineSweeper[13]使用逻辑公式建模控制平面协议行为, 在建模大型网络时需要大量的 SMT 公式, 存在扩展性问题. 为准确建模不同厂商的协议行为, Hoyan 使用定制化算法模拟生成数据平面信息, 然后与实际网络设备的数据平面进行对比, 根据差异修正模型, 直到模拟生成的数据平面与实际网络设备完全一致. 为实现高扩展性与高表达性, Hoyan 整体上模拟生成数据平面信息, 在生成过程中局部使用逻辑公式建模与验证属性相关的信息, 最后使用 SMT 求解器完成验证. Hoyan 实现高扩展性的关键点在于, 基于模拟技术实现了轻量级的控制平面协议行为建模, 且只使用逻辑公式建模必要的信息, 而不是像 MineSweeper 一样全局使用逻辑公式建模. Hoyan 实现高表达性的关键点在于, 使用逻辑公式有效建模了不确定情况. 经过实验, Hoyan 建模厂商设备协议行为的准确度接近 100%. 对于错误链路情况下的可达性验证, Hoyan 的验证速度相比 MineSweeper、Plankton[40]等最前沿的验证工具提高了数个数量级, 能够快速完成大型广域网的验证任务. Hoyan 已经被部署在阿里巴巴公司的广域网中运行, 且阻止了许多由于配置错误而导致的潜在服务故障.

(3) 基于图算法

Abhashkumar 等人于 2020 年提出了 Tiramisu[41], 通过改进 ARC[33], 在保持高扩展性的同时支持建模更多的协议行为. 文中指出, 已有控制平面验证工具都在扩展性和表达性之间做出了取舍, 难以同时实现高扩展性与高表达性. Tiramisu 使用与 ARC 相同的图模型和定制化图算法进行建模和验证, 以达到高扩展性, 然后通过图模型分层等建模方式, 达到了高表达性. Tiramisu 使用两层的图模型建模各个子网的控制平面协议行为, 第 1 层图模型用于建模参与路由信息传播的路由进程, 第 2 层图模型在第 1 层的基础上进行更加细粒度的建模, 以建模各种协议交互行为. Tiramisu 通过分类验证策略, 对每种策略类型使用定制化的建模和验证方法, 实现了高效的建模和验证. 例如, 对于验证路由优先级的策略, Tiramisu 基于稳定路径问题[95]和路由代数理论[96]建模路由协议行为以生成数据平面信息, 然后基于数据平面信息完成验证; 对于验证区域流量隔离等通信路径存在性相关的策略, Tiramisu 直接基于图模型, 使用深度优先算法判断路径的存在性完成验证. 经过实验, Tiramisu 相比 MineSweeper[13]和 Plankton[40]等最先进的验证工具, 实现了 2–600 倍的验证速度提升. 在保持高扩展性的同时, Tiramisu 通过细粒度的图建模实现了高表达性, 除了可以建模 OSPF、BGP 等网络层路由协议, 还能建模数据链路层的 VLAN 协议.

(4) 高表达性验证研究总结

上述研究基于不同技术, 实现了不同方面的表达性提升. Plankton[35]基于模型检测技术进行详细地建模, 能够验证控制平面所有可能生成的数据平面状态, 以全面保证网络的正确性. Minesweeper[13]基于逻辑公式建模, 能够建模多种网络协议与环境信息, 且可以验证所有的收敛数据平面状态. Hoyan[17]是首个考虑不同厂商设备协议行为实现差别的验证工具, 通过不断对比模拟生成与实际设备中的数据平面信息修正模型, 实现了接近 100% 的建模准确性. Hoyan 还通过结合模拟与 SMT 求解技术, 同时达到了高表达性与高扩展性, 既能建模多种网络协议与处理不确定性情况, 又能快速完成大型广域网下的网络属性验证. Tiramisu[41] 基于 ARC[33]进行了改进, 通过划分

策略类型并使用定制化的建模和验证, 在保持高扩展性的同时实现了多种网络协议的建模.

3.4 总 结

本节根据研究主要解决的问题, 将控制平面验证研究分成了开创性研究、高扩展性验证和高表达性验证 3 个研究部分, 并结合时间顺序和验证技术具体介绍了各个部分的已有研究. 开创性研究是最早出现的控制平面验证研究, 提供了控制平面验证的不同实现方案. 高扩展性研究旨在实现大型网络下的快速验证, 使用了不同技术用于提升验证速度或优化算法复杂度. 高表达性验证旨在实现准确且大范围地建模控制平面与环境信息, 以应对实际网络中复杂的网络情况. 与数据平面验证研究的划分方法一致, 本章根据研究侧重解决的问题以及创新点对控制平面验证研究进行划分. 例如, Hoyan[17]同时实现了高扩展性与表达性, 但其创新之处在于首次实现了不同厂商设备协议行为差异的准确建模, 因此被划分到高表达性验证研究.

下面对控制平面验证的实现和优化方法进行总结. 实现方面, 如图 5 所示, 目前的控制平面验证研究主要有模拟、图算法和 SMT 求解 3 种实现思路. 基于模拟的实现思路是通过模拟控制平面协议行为 (例如路由通告、选择、过滤等), 生成数据平面以及验证所需信息, 然后基于生成信息完成建模验证, 例如使用已有的数据平面验证技术[12,35,42]或使用定制的算法[13,17,34,37]建模验证; 基于图算法的实现思路则是使用图模型建模控制平面信息, 然后基于图算法完成验证[33,41], 例如使用最小割算法验证任意链路错误情况下的可达性; 基于 SMT 求解的实现思路是使用 SMT 公式建模控制平面行为, 然后使用求解器求解违反网络属性的行为, 网络属性在没有求解结果的情况下满足[13]. 其中, 已有研究大多采用基于模拟的实现思路, 且主要使用逻辑公式、路由代数或定制化的算法模拟控制平面协议行为. 对于控制平面的多收敛数据平面问题, 已有研究主要基于稳定路径问题[95]和路由代数理论[96]解决. 对于不同厂商设备的配置语言或协议实现的差异, 已有研究大多使用 Batfish[12]的配置解析器解析不同厂商的配置语言, 但只有 Hoyan[17] 通过对比模型与实际设备的转发信息准确建模了不同厂商设备的协议实现. 优化方面, 与数据平面验证一样, 控制平面验证可以使用等价类和增量技术提升验证速度. 通过等价类划分, 控制平面验证工具可以只对与验证策略相关的等价类进行建模, 且可以并行计算不同的等价类验证任务以提高验证速度. 增量计算技术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, 可以用于有效处理网络配置更新情况. 此外, 提高扩展性的技术还有使用网络变换技术缩小验证网络规模, 使用抽象解释技术简化建模验证过程等. 对于表达性提升, 已有研究主要通过细粒度地建模实现, 例如 Plankton[35]详尽地建模了所有可能执行顺序的控制平面协议行为, 能够验证控制平面所有可能生成的数据平面状态, Tiramisu[41]则使用细粒度的图模型, 对多种网络协议行为进行了建模.

表 3 对控制平面验证研究进行了总结, 分为了 5 个方面进行说明, 分别是实现思路、基于技术、扩展性、表达性和简要总结. 其中, 实现思路分为模拟和分析两种; 基于技术是指工具主要使用的建模和验证方法; 扩展性是指工具处理大型网络验证任务的能力, 与验证速度和算法复杂度相关; 表达性是指工具建模分析控制平面和环境信息的能力; 简要总结部分对各个研究的关键内容或创新处进行了简要说明.

4 有状态网络验证

上述介绍的数据平面验证与控制平面验证研究已经实现了较高的表达性与扩展性, 但是这些研究都针对验证无状态网络, 没有考虑网络中的中间件, 或直接将中间件当作无状态设备处理, 不能正确完整地验证中间件. 调查显示, 中间件在网络中的数量已经与路由器的数量相当[98], 且引发了超过 40% 的网络高危错误[99]. 因此, 实现有状态网络验证以保证中间件按照意图正确运行是非常必要的. 有状态网络验证实际上属于数据平面验证或控制平面验证中的一个子研究方向, 通过分析数据平面或控制平面以验证网络属性. 有状态网络验证与无状态网络验证的区别在于, 需要额外考虑中间件的建模验证, 特别是有状态中间件的建模验证. 有状态网络验证的难点在于, 有状态中间件不遵循路由器的简单转发模型, 其转发决策由保存的状态信息决定, 与中间件先前接收到的数据包相关. 此外, 由于中间件接收到的数据包数量不受限制, 且以任意顺序接收, 准确建模有状态中间件非常困难. 有状态网络验证除了验证可达性等基本的网络属性, 还需要验证中间件状态相关的网络属性, 以保证中间件的运行正确性. 例如, 验证 NAT 在修改某个数据包的端口后, 后续属于同一数据流的数据包都被转换为同一端口.

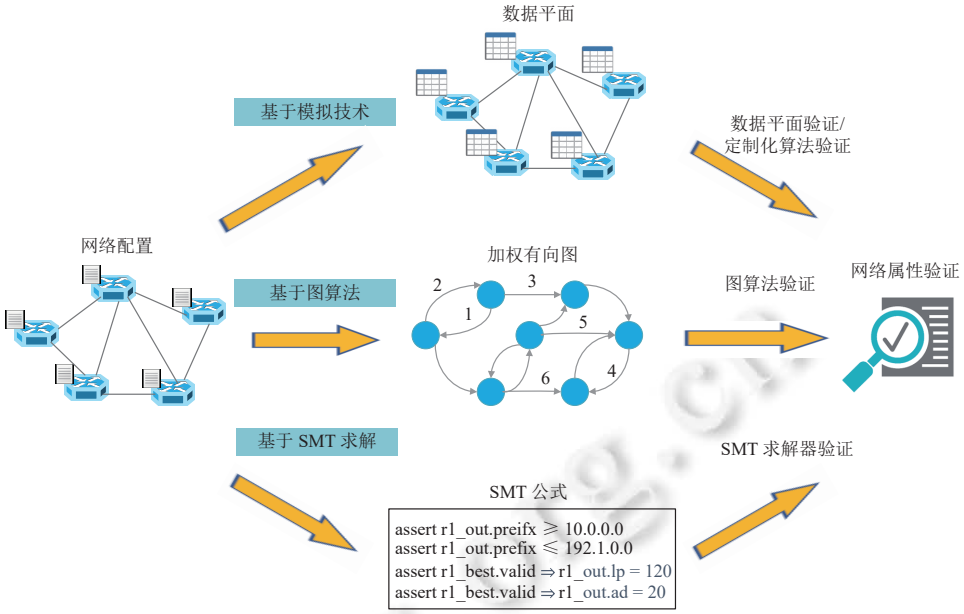


图 5 控制平面验证主要实现思路

表 3 控制平面验证技术总结

工具	实现思路	基于技术	扩展性	表达性	简要总结
Batfish[12]	模拟	Datalog	低	高	完整模拟生成数据平面, 然后使用数据平面验证工具完成验证
ARC[33]	图算法	图算法	高	低	使用定制化的图算法直接分析控制平面验证网络属性, 实现了很高的验证速度, 但表达性较低
ERA[34]	模拟	BDD	中	中	基于路径模拟控制平面协议行为, 然后根据生成信息验证网络属性, 兼顾了扩展性和表达性
Bonsai[36]	—	稳定路径问题路由代数理论	—	—	一种网络变换方法, 基于网络结构对称性与控制平面行为等价性简化网络结构, 并保证验证结果在变换前后的网络中相同
Origami[39]	—	稳定路径问题路由代数理论	—	—	基于Bonsai进行改进, 结合网络抽象过程与验证过程, 根据验证结果调整网络抽象, 使抽象后的网络能够提供足够的链路用于链路环境分析
ShapeShifter[37]	模拟	抽象解释	高	低	对控制平面协议行为进行抽象, 牺牲了一定的验证精确度, 实现了接近线性的时间复杂度
FastPlane[38]	模拟	图算法	高	低	通过优化路由选择、传播等协议行为, 实现了数据平面的快速收敛, 但只适用于具有单调特征的BGP网络
Plankton[40]	模拟	模型检测	高	高	使用相对SMT求解技术更高效的模型检测技术, 并通过等价类等技术进行优化, 实现了高扩展性
RealConfig[42]	模拟	DDlog	高	高	使用DDlog增量生成数据平面以应对控制平面的变化情况, 使用APKeep[16]增量验证数据平面
NUV[15]	—	路由代数理论	—	—	可以找到受控制平面更新影响的网络策略, 帮助其他工具实现增量验证
Plankton[35]	模拟	模型检测	中	高	详尽地建模了所有可能的数据平面情况, 能够验证控制平面所有可能生成的数据平面状态
Minesweeper[13]	SMT求解	SMT求解	中	高	可以建模所有可能生成的数据平面情况, 但使用SMT公式建模存在一定的扩展性问题
Tiramisu[41]	模拟/图算法	图算法	高	高	基于ARC[33]进行改进, 使用分层图模型建模提升了表达性, 并通过划分策略类型进行定制化地建模验证实现了高扩展性
Hoyan[17]	模拟	SMT求解	高	高	结合了模拟技术建模速度快和SMT求解技术可有效处理不确定性情况的优点, 且可以准确建模不同厂商的控制平面协议行为

目前, 有状态网络验证属于起步阶段, 且主要采用数据平面验证方法实现. 由于已有的有状态网络验证研究数量较少, 本节不对有状态网络验证研究进行划分, 直接根据研究的出现时间进行顺序介绍.

4.1 开创性研究

目前的有状态网络验证研究仍处于开创性阶段, 使用不同的技术实现了有状态网络下的网络属性验证.

(1) 基于符号执行

Stoenescu 等人于 2013 年最早开始了有状态网络验证研究, 基于符号执行技术提出了 SymNet[43], 并于 2016 年提出了建模语言 SEFL 对 SymNet 进行了优化[46]. SymNet 采用了数据平面验证工具 HSA[10]将网络设备建模为转移函数, 然后输入符号数据包进行分析的验证思路. 区别在于, SymNet 通过在数据包头部添加状态信息, 实现了中间件的状态建模. SymNet 假设数据流之间不会相互影响, 且忽略中间件的全局变量建模. 为同时兼顾到扩展性和表达性, SymNet 使用 SEFL 语言建模网络设备, 相比基于 C 语言的建模有更少的符号执行路径, 相比基于 HSA 的建模能够支持更多的网络功能. 此外, SymNet 基于测试技术, 通过发送测试数据包对比模型与实际设备的转发行为, 以保证建模准确性. 实验表明, SymNet 使用 SEFL 能够准确建模网络设备.

(2) 基于 SMT 求解技术

Panda 等人于 2015 年提出了中间件建模和有状态网络验证的思路[44], 并基于该思路于 2017 年提出了验证工具 VMN[47]. 文中指出, 中间件的正确性验证非常困难, 一方面是因为中间件的代码是厂商私有的, 另一方面是因为中间件的正确性难以描述, 例如入侵检测系统旨在检测出可疑数据包, 但可疑数据包的判断并没有标准统一的定义. 此外, 网络管理人员主要关注于验证数据包转发行为的正确性. 因此, VMN 选择只建模中间件的转发功能, 抽象掉中间件的其他功能部分. 例如, VMN 只建模入侵检测系统如何转发可疑数据包, 而不关心可疑数据包的检测过程. 这种建模的好处在于, VMN 只需要了解中间件的转发行为即可建模, 且使用一个模型即可表达所有相同类型的中间件. VMN 使用逻辑公式建模中间件以及所要验证的网络属性, 然后使用 SMT 求解器进行求解验证. 为提高扩展性, VMN 利用了中间件的转发行为具有空间局部性的特点, 根据转发行为对中间件进行了分类. 当网络中只包含特定类型的中间件时, 可以在保证不影响验证结果的前提下, 缩小建模和验证的网络规模. 此外, VMN 还根据网络结构对称性与验证策略之间的联系, 减少了需要验证的策略数量, 进一步提升了验证速度.

(3) 基于抽象解释技术

Velner 等人于 2016 年对有状态网络中的区域流量隔离属性验证进行了复杂度分析, 并提出了一种原型验证工具[45]. 为实现复杂度分析, 文中假设中间件状态有限且数据包在网络中不会丢失. 在该假设前提下, 文中证明即使在网络中不存在转发循环的情况下, 区域流量隔离属性的验证也是不可解的. 为使得验证问题可判定, Velner 等人[45]提出了一种保证验证结果可靠性的抽象方法, 将网络设备处理数据包的先进先出顺序抽象为任意顺序. 在经过任意顺序抽象使得问题可判定之后, Velner 等人根据转发行为将中间件分为了无状态型 (stateless)、增长型 (increasing)、渐进型 (progressing) 和任意型 (arbitrary) 这 4 种类型, 并分别对各种类型的有状态网络进一步地分析了计算复杂度. 其中, 区域流量隔离属性验证的计算复杂度在无状态型和增长型网络下是多项式时间, 在渐进型网络下是 coNP-complete, 在任意型网络下是 EXPSPACE-complete. Velner 等人[45]根据上述分析提出了有状态网络验证原型工具, 基于 Datalog 实现了无状态型和增长型网络下的验证, 基于 Petri 网络实现了渐进型和任意型网络下的验证.

基于 Velner 等人的复杂度分析与抽象方法[45], Alpernas 等人于 2018 年提出了一种具有更高扩展性的抽象方法[48], 可以用于高效验证有状态网络中的区域流量隔离属性. 该抽象方法除了将网络设备处理数据包行为抽象为任意顺序, 还将中间件状态抽象为每次处理完数据包都返回到初始状态. 文中证明, 该抽象方法不仅可以保证验证结果的可靠性, 还能够将验证问题的计算复杂度从 EXPSPACE-complete 优化到 coNP-hard. Alpernas 等人使用自定义语言 AMDL 建模中间件, 然后结合网络拓扑和初始状态等信息转换成 Datalog 程序, 最后通过执行 Datalog 程序判断是否会到达错误状态以验证网络属性. 实验表明, 该验证工具实现了与 VMN[47]相当的验证速度.

(4) 基于符号模型检测技术

Yuan 等人于 2020 年提出了 NetSMC[49], 相较于 VMN[47] 提升了数个数量级的验证速度. 基于 Velner 等人的

复杂度分析[45], 文中指出任何有状态网络验证工具都需要进行一定的抽象以达到确定性的网络属性验证, 如减少支持的网络功能与验证策略, 或者不保证验证结果准确性. NetSMC 将网络抽象为单个数据包转发的模型, 即任意时间上网络中只存在一个数据包. 这种抽象方法的好处在于, 对于如区域流量隔离等多种网络属性的验证, 能够保证不损失验证结果的准确性. 但是, 其同时带来了一定的局限性, 如不能验证数据包交互带来的错误. NetSMC 使用线性时态逻辑建模网络策略, 并进一步转换成计算树逻辑以适用于符号模型检测框架. 相较于直接使用通用的模型检测工具进行验证, NetSMC 提出了定制化的符号模型检测算法以提高扩展性. 例如, NetSMC 使用一阶逻辑而不是通用的 BDD 数据结构建模网络状态, 以有效处理中间件保存的大量状态信息. 此外, NetSMC 借鉴数据库查询理论, 提出了判断状态集合包含关系的算法, 相比使用通用的 SMT 求解方法判断更加高效. 经过实验, 在包含 8 个中间件的网络中验证区域流量隔离属性时, NetSMC 的验证速度相比 VMN 快出了 200 倍. 除了实现较高的验证速度, NetSMC 通过使用线性时态逻辑能够建模丰富的网络策略.

(5) 开创性研究总结

上述研究分别基于符号执行、SMT 求解、抽象解释和符号模型检测技术, 实现了有状态网络验证. SymNet[43,46]基于符号执行技术, 通过在数据包头部添加状态信息建模有状态中间件, 并提出了建模语言 SEFL 以加快符号执行过程. VMN[47]基于 SMT 求解技术实现, 提出了一种适用于验证可达性的中间件抽象建模方法, 并通过缩小验证网络规模提升了验证速度. Velner 等人[45]证明了区域流量隔离属性验证在有状态网络中是不可判定的, 并通过抽象网络设备处理数据包的顺序实现了可判定. Alpernas 等人[48]在 Velner 等人抽象方法上进一步地抽象了中间件的状态变化, 实现了复杂度的优化. NetSMC[49]基于符号模型检测技术实现, 将网络抽象为单个数据包转发的模型, 并通过定制的符号模型检测算法实现了较高的扩展性. 相比数据平面验证研究, 上述有状态网络验证研究的扩展性都处于较低的水平, 只能完成离线验证任务. 例如, SymNet 的验证速度与数据平面离线验证工具 HSA[10]相当, NetSMC 需要半个小时完成包含 147 个中间件的网络的区域流量隔离属性验证.

4.2 总 结

由于有状态网络验证处于起步阶段, 研究数量较少, 本节未对有状态网络验证研究进行分类, 直接按照研究的出现时间进行顺序介绍. 目前, 有状态网络验证主要解决了实现问题, 提出了有状态网络验证的不同实现方案, 但是其扩展性与表达性仍处于较低水平, 不能完成大型网络下的快速验证, 或大范围且准确地建模各种网络功能. 其中, 已有研究主要基于数据平面验证技术实现, 即通过分析数据包的转发信息验证可达性等网络属性. 与无状态网络中的数据平面验证的区别在于, 有状态网络验证还需要考虑影响数据包转发行为的中间件状态信息. 由于中间件的状态信息与中间件先前接收到的数据包相关, 且数据包的接收数量与接收顺序不受限制, 建模有状态网络非常困难, 区域流量隔离等基本的验证问题在这种情况下也变得不可解. 为实现验证问题的可判定, 不能完整地建模有状态网络, 都需要进行一定的抽象, 如假定中间件状态有限、网络设备以任意顺序处理数据包、网络中只存在一个数据包等. 这种抽象方法带来的问题是, 不能完整地建模所有网络行为, 或不能保证验证结果的准确性.

表 4 对有状态网络验证研究进行了总结, 分为了 5 个方面进行说明, 分别是基于技术、准确性、扩展性、表达性和简要总结. 其中, 基于技术是指工具主要使用的建模和验证方法; 准确性包含准确、可靠、完备 3 种类型, 准确是指既不会误报也不会漏报错误, 可靠是指不会漏报但是会误报错误, 完备是指不会误报但是会漏报错误; 扩展性是指工具处理大型网络验证任务的能力, 与验证速度和算法复杂度相关; 表达性是指工具建模网络功能的能力, 由于目前的有状态网络验证都基于数据平面验证实现, 在此主要体现为数据平面行为与中间件功能的建模分析能力; 简要总结部分对各个研究的关键内容或创新处进行了简要说明.

5 相关研究方向

(1) SDN 控制平面验证

本文只介绍了传统网络中的控制平面验证研究. 对于 SDN 控制平面验证, 可以分为 SDN 程序验证和控制器验证两种实现方案[60]. 其中, SDN 程序验证用于保证 SDN 应用得到正确实现, 控制器验证用于保证错误规则不会

被下发到交换机. 目前, SDN 控制平面验证主要基于形式化验证技术实现, 例如基于定理证明的 VeriCon[100]与 NetKAT[101], 基于模型检测的 Verificare[102]和 FlowLog[103], 基于符号执行的 MSAID[104]等.

表 4 有状态网络验证技术总结

工具	基于技术	准确性	扩展性	表达性	简要总结
SymNet[46]	符号执行	可靠	低	低	通过在数据包头部添加状态信息建模有状态中间件, 提出了建模语言SEFL以加快符号执行过程
VMN[47]	SMT求解	准确	低	中	提出了一种中间件抽象建模方法, 使用网络压缩技术提高了扩展性
Alpernas等人[48]	抽象解释	可靠	中	中	通过抽象数据包处理顺序与中间件状态, 实现了较低复杂度的验证
NetSMC[49]	模型检测	可靠	中	中	将网络抽象为单个数据包转发的模型, 使用定制的符号模型检测算法实现了较高的验证速度

(2) 网络测试

网络验证可以通过分析数据平面或控制平面以验证网络的正确性, 但其仍存在一定的局限性, 难以发现网络中存在的全部错误, 例如路由器不正确按照转发表转发数据包, 网络链路拥塞等. 网络测试是对网络验证的补充, 能够发现网络设备硬件错误等实现错误. 但是, 网络测试并不能证明网络中不存在错误[73]. 与网络验证一样, 网络测试也可以分为数据平面测试和控制平面测试[60]. 数据平面测试通过发送测试数据包, 然后监测网络行为以发现网络错误; 控制平面测试研究目前集中于 SDN 网络, 基于白盒测试或黑盒测试方法检测 SDN 程序中的设计或实现错误. 目前, 数据平面测试研究有基于符号执行的 ATPG [105]、BUZZ[106], 基于模型检测的 NOT NICE[107]等; 控制平面测试研究有基于符号执行和模型检测的 NICE[108], 基于模糊测试的 STS[109]等.

(3) 网络配置综合

由于不同的网络管理人员对网络运行有着不同的理解, 且网络意图和配置实现之间存在巨大的语义鸿沟, 非常容易出现配置错误. 网络配置综合旨在根据网络意图自动生成正确的网络配置. 网络配置综合与软件验证领域中的程序综合 (program synthesis) 的实现思想比较相似, 都试图使用形式化方法自动生成符合规约的配置或代码. 其中, 程序综合是指使用指定的编程语言自动生成符合程序规约的技术[71]. 目前, 网络配置综合已有研究有基于 SAT/SMT 求解技术的 ConfigAssure[64]、CPR[110]、NetComplete[111]、JINJING [112]、AED[113], 基于 Datalog 的 SyNET[114], 基于图算法的 Propane [115,116].

(4) 网络仿真

网络仿真技术使用虚拟机技术构建出虚拟环境, 通过运行与真实网络相同的网络设备固件与配置, 生成对应的转发表信息进行验证. 网络仿真通过仿真实际网络运行情况, 能够发现网络验证所不能发现的设备实现错误, 可以被看作是网络验证的增强版本[17]. 但是, 网络仿真需要网络设备厂商提供包含设备固件的虚拟机或容器, 且需要大量计算资源用于运行虚拟环境, 难以应用于包含多种厂商网络设备的大型网络. 目前, 网络仿真已有研究有 CrystalNet[117]、MiniNet[118]、MaxiNet[119].

(5) 基于意图的网络 (IBN)

IBN 是一种新兴的网络范式, 可以自动地实现用户意图, 并通过持续监控网络状态信息, 判断用户意图是否得到正确实现[65]. 在 IBN 中, 用户只需要描述想要的结果, 而不用描述如何实现. IBN 只是一种新的理念, 并不是一种技术, 通过结合多种现有技术进行实现, 如网络验证、网络综合、自然语言处理等. 目前, IBN 已经得到了广泛的关注, 学术界和产业界都已经开始了 IBN 相关研究[55,120].

6 未来研究方向

(1) 定量属性验证

目前, 网络验证关注的网络属性主要为可达性、无转发循环、区域流量隔离等布尔值属性. 尽管其可以对网

络安全意图提供保证, 但其无法对网络性能意图提供保证, 如延迟, 丢包率等. 因此, 需要实现对表达性能意图的定量属性的验证. 目前, 已有的定量属性验证研究有 SLA-Verifier[121]、Netter[122].

(2) 错误定位与修复

已有的网络验证研究在发现违反网络属性的错误之后, 依然需要根据输出信息手动定位错误配置并完成修复. 因此, 需要自动化的错误定位与修复功能, 以保证在发现网络错误之后能够及时修复. 目前, 已经有相关研究实现了数据平面验证或控制平面验证后的错误定位[123−125], 但存在准确性不足或应用场景有限的缺陷. 错误自动修复方面, 可以利用网络配置综合工具进行实现, 例如 CPR[110]、NetComplete[111]、AED[113].

(3) 更方便地使用网络验证工具

网络验证研究已经取得了显著的成果, 可以为大型复杂的网络提供安全性和可用性方面的保证. 然而, 由于网络验证工具大多使用定制化的语言表达网络意图, 且缺少容易操作的图形界面, 使用起来比较困难. 因此, 下一步的目标是提高网络验证工具的易用性, 使其能够得到广泛应用, 像 Ping 和 Traceroute 等工具一样流行[126]. 目前, 已经出现相关研究用于帮助工程师轻松地使用或开发网络验证工具. 例如, 计算出策略的网络覆盖率以帮助用户更好地制定验证策略[127,128]、提出高级的编程语言以便于简单表达网络意图或开发网络验证工具[127,129]、自动推断生成网络策略以实现更高效的验证[130−132]等.

(4) 检测网络验证工具的正确性

与其他软件工具一样, 网络验证工具也可能存在漏洞, 导致输出错误的验证结果. 因此, 在网络验证工具投入运行之前, 有必要进行正确性检测. 最近, Birkner 等人提出了 Metha[133], 基于黑盒差异测试技术对已有网络验证工具进行了测试, 成功发现了验证工具存在的漏洞.

7 总 结

网络验证是近年来的热点研究领域, 学术界和工业界对该领域做出了大量的研究, 并将部分的工具进行了实际应用以保证网络的安全可靠性. 本文分为数据平面验证、控制平面验证和有状态网络验证 3 个研究方向, 对网络验证领域中的已有研究做出了综述. 由于目前传统网络架构仍占据主流地位, 本文只对传统网络中的控制平面验证研究进行了介绍. 无状态数据平面验证作为网络验证领域最早的研究工作, 目前已经得到了很好的解决, 可以在大型网络中完成实时验证; 控制平面验证也取得了显著的进展, 可以在运行有多种协议的大型网络中完成快速验证; 有状态网络验证目前研究相对较少, 仍处于起步阶段. 网络验证领域的研究成果及时地满足了目前日益复杂庞大的网络的可用性和安全性需求, 且仍有非常大的发展空间. 本文希望通过提供网络验证领域发展以来的热点研究内容与解决方法, 帮助读者了解网络领域研究的发展脉络, 以思考该领域现有的研究技术并找到创新的研究思路.

References:

[1] McKeown N. How SDN will shape networking. 2011. http://www.youtube.com/watch?v=c9-K5O_qYgA

[2] McKeown N. Mind the gap. 2012. https://www.youtube.com/watch?v=Ho239zpKMwQ

[3] Intentionet. Three ways to break a network (and one to save it). 2020. https://www.intentionet.com/blog/three-ways-to-break-a-network-and-one-to-save-it/

[4] Tung L. Azure global outage: Our DNS update mangled domain records, says Microsoft. 2019. https://www.zdnet.com/article/azure-global-outage-our-dns-update-mangled-domain-records-says-microsoft/

[5] Jander M. Centurylink is in the hot seat. 2020. https://www.futuriom.com/articles/news/centurylink-peers-blame-wobbly-router-for-massive-outage/2020/09

[6] Opengear. Measuring the true cost of network outages. 2020. https://opengear.com/white-paper/measuring-the-true-cost-of-network-outages

[7] Russinoff DM. A mechanically checked proof of correctness of the AMD K5 floating point square root microcode. Formal Methods in System Design, 1999, 14(1): 75–125. [doi: 10.1023/A:1008669628911]

[8] Brat G, Drusinsky D, Giannakopoulou D, Goldberg A, Havelund K, Lowry M, Pasareanu C, Venet A, Visser W, Washington R.

Experimental evaluation of verification and validation tools on martian rover software. Formal Methods in System Design, 2004, 25(2): 167–198. [doi: 10.1023/B:FORM.0000040027.28662.a4]

[9] Mai HH, Khurshid A, Agarwal R, Caesar M, Godfrey PB, King ST. Debugging the data plane with Anteater. In: Proc. of the ACM SIGCOMM 2011 Conf. Toronto: Association for Computing Machinery, 2011. 290–301. [doi: 10.1145/2018436.2018470]

[10] Kazemian P, Varghese G, McKeown N. Header space analysis: Static checking for networks. In: Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation. San Jose: USENIX Association, 2012. 9.

[11] Lopes NP, Bjørner N, Godefroid P, Jayaraman K, Varghese G. Checking beliefs in dynamic networks. In: Proc. of the 12th USENIX Conf. on Networked Systems Design and Implementation. Oakland: USENIX Association, 2015. 499–512.

[12] Fogel A, Fung S, Pedrosa L, Walraed-Sullivan M, Govindan R, Mahajan R, Millstein T. A general approach to network configuration analysis. In: Proc. of the 12th USENIX Conf. on Networked Systems Design and Implementation. Oakland: USENIX Association, 2015. 469–483.

[13] Beckett R, Gupta A, Mahajan R, Walker D. A general approach to network configuration verification. In: Proc. of the Conf. of the ACM Special Interest Group on Data Communication. Los Angeles: Association for Computing Machinery, 2017. 155–168. [doi: 10.1145/3098822.3098834]

[14] Steffen S, Gehr T, Tsankov P, Vanbever L, Vechev M. Probabilistic verification of network configurations. In: Proc. of the Annual Conf. of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: Association for Computing Machinery, 2020. 750–764. [doi: 10.1145/3387514.3405900]

[15] Li YH, Wang ZL, Yin X, Shi XG, Wu JP, Ye FD, Yao JY, Zhang H. Assisting reachability verification of network configurations updates with NUV. Computer Networks, 2020, 177: 107326. [doi: 10.1016/j.comnet.2020.107326]

[16] Zhang P, Liu X, Yang HK, Kang N, Gu ZC, Li H. APKeep: Realtime verification for real networks. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2020. 241–255.

[17] Ye FD, Yu D, *et al*. Accuracy, scalability, coverage: A practical configuration verifier on a global WAN. In: Proc. of the Annual Conf. of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: Association for Computing Machinery, 2020. 599–614. [doi: 10.1145/3387514.3406217]

[18] Beckett R, Mahajan R. Capturing the state of research on network verification. 2020. https://netverify.fun/2-current-state-of-research/

[19] Al-Shaer E, Marrero W, El-Atawy A, ElBadawi K. Network configuration in a box: Towards end-to-end verification of network reachability and security. In: Proc. of the 17th IEEE Int'l Conf. on Network Protocols. Plainsboro: IEEE, 2009. 123–132. [doi: 10.1109/ICNP.2009.5339690]

[20] Al-Shaer E, Al-Haj S. FlowChecker: Configuration analysis and verification of federated openflow infrastructures. In: Proc. of the 3rd ACM Workshop on Assurable and Usable Security Configuration. Chicago: Association for Computing Machinery, 2010. 37–44. [doi: 10.1145/1866898.1866905]

[21] Kazemian P, Chang M, Zeng HY, Varghese G, McKeown N, Whyte S. Real time network policy checking using header space analysis. In: Proc. of the 10th USENIX Conf. on Networked Systems Design and Implementation. Lombard: USENIX Association, 2013. 99–112.

[22] Khurshid A, Zhou WX, Caesar M, Godfrey PB. VeriFlow: Verifying network-wide invariants in real time. In: Proc. of the 1st Workshop on Hot Topics in Software Defined Networks. Helsinki: ACM, 2012. 49–54. [doi: 10.1145/2342441.2342452]

[23] Yang HK, Lam SS. Real-time verification of network properties using Atomic Predicates. In: Proc. of the 21st IEEE Int'l Conf. on Network Protocols (ICNP). Goettingen: IEEE, 2013. 1–11. [doi: 10.1109/ICNP.2013.6733614]

[24] Zeng HY, Zhang SD, Ye F, Jeyakumar V, Ju M, Liu JD, McKeown N, Vahdat A. Libra: Divide and conquer to verify forwarding tables in huge networks. In: Proc. of the 11th USENIX Conf. on Networked Systems Design and Implementation. Seattle: USENIX Association, 2014. 87–99.

[25] Wang HZ, Qian C, Yu Y, Yang HK, Lam SS. Practical network-wide packet behavior identification by AP classifier. In: Proc. of the 11th ACM Conf. on Emerging Networking Experiments and Technologies. Heidelberg: Association for Computing Machinery, 2015. 10. [doi: 10.1145/2716281.2836095]

[26] Plotkin GD, Bjørner N, Lopes NP, Rybalchenko A, Varghese G. Scaling network verification using symmetry and surgery. In: Proc. of the 43rd Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. St. Petersburg: Association for Computing Machinery, 2016. 69–83. [doi: 10.1145/2837614.2837657]

[27] Bjørner N, Juniwal G, Mahajan R, Seshia SA, Varghese G. ddNF: An efficient data structure for header spaces. In: Proc. of the 12th Int'l Haifa Verification Conf. Haifa: Springer, 2016. 49–64. [doi: 10.1007/978-3-319-49052-6_4]

[28] Horn A, Kheradmand A, Prasad MR. Delta-net: Real-time network verification using atoms. In: Proc. of the 14th USENIX Conf. on

Networked Systems Design and Implementation. Boston: USENIX Association, 2017. 735–749.

[29] Yang HK, Lam SS. Scalable verification of networks with packet transformers using atomic predicates. IEEE/ACM Trans. on Networking, 2017, 25(5): 2900–2915. [doi: 10.1109/TNET.2017.2720172]

[30] Jensen JS, Krøgh TB, Madsen JS, Schmid S, Srba J, Thorgersen MT. P-Rex: Fast verification of MPLS networks with multiple link failures. In: Proc. of the 14th Int'l Conf. on Emerging Networking Experiments and Technologies. Heraklion: Association for Computing Machinery, 2018. 217–227. [doi: 10.1145/3281411.3281432]

[31] Jayaraman K, Bjørner N, *et al*. Validating datacenters at scale. In: Proc. of the ACM Special Interest Group on Data Communication. Beijing: Association for Computing Machinery, 2019. 200–213. [doi: 10.1145/3341302.3342094]

[32] Horn A, Kheradmand A, Prasad MR. A precise and expressive lattice-theoretical framework for efficient network verification. In: Proc. of the 27th Int'l Conf. on Network Protocols (ICNP). Chicago: IEEE, 2019. 1–12. [doi: 10.1109/ICNP.2019.8888144]

[33] Gember-Jacobson A, Viswanathan R, Akella A, Mahajan R. Fast control plane analysis using an abstract representation. In: Proc. of the 2016 ACM SIGCOMM Conf. Florianopolis: Association for Computing Machinery, 2016. 300–313. [doi: 10.1145/2934872.2934876]

[34] Fayaz SK, Sharma T, Fogel A, Mahajan R, Millstein T, Sekar V, Varghese G. Efficient network reachability analysis using a succinct control plane representation. In: Proc. of the 12th USENIX Conf. on Operating Systems Design and Implementation. Savannah: USENIX Association, 2016. 217–232.

[35] Prabhu S, Kheradmand A, Godfrey B, Caesar M. Predicting network futures with plankton. In: Proc. of the 1st Asia-Pacific Workshop on Networking. Hong Kong: Association for Computing Machinery, 2017. 92–98. [doi: 10.1145/3106989.3106991]

[36] Beckett R, Gupta A, Mahajan R, Walker D. Control plane compression. In: Proc. of the 2018 Conf. of the ACM Special Interest Group on Data Communication. Budapest: Association for Computing Machinery, 2018. 476–489. [doi: 10.1145/3230543.3230583]

[37] Beckett R, Gupta A, Mahajan R, Walker D. Abstract interpretation of distributed network control planes. Proc. of the ACM on Programming Languages, 2020, 4(POPL): 42. [doi: 10.1145/3371110]

[38] Lopes NP, Rybalchenko A. Fast BGP simulation of large datacenters. In: Proc. of the 20th Int'l Conf. on Verification, Model Checking, and Abstract Interpretation. Cascais: Springer, 2019. 386–408. [doi: 10.1007/978-3-030-11245-5_18]

[39] Giannarakis N, Beckett R, Mahajan R, Walker D. Efficient verification of network fault tolerance via counterexample-guided refinement. In: Proc. of the 31st Int'l Conf. on Computer Aided Verification. New York: Springer, 2019. 305–323. [doi: 10.1007/978-3-030-25543-5_18]

[40] Prabhu S, Chou KY, Kheradmand A, Godfrey B, Caesar M. Plankton: Scalable network configuration verification through model checking. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2020. 953–967.

[41] Abhashkumar A, Gember-Jacobson A, Akella A. Tiramisu: Fast multilayer network verification. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2020. 201–219.

[42] Zhang P, Huang YH, Gember-Jacobson A, Shi WB, Liu X, Yang HK, Zuo ZQ. Incremental network configuration verification. In: Proc. of the 19th ACM Workshop on Hot Topics in Networks. Association for Computing Machinery, 2020. 81–87. [doi: 10.1145/3422604.3425936]

[43] Stoenescu R, Popovici M, Negreanu L, Raiciu C. SymNet: Static checking for stateful networks. In: Proc. of the 2013 Workshop on Hot Topics in Middleboxes and Network Function Virtualization. Santa Barbara: Association for Computing Machinery, 2013. 31–36. [doi: 10.1145/2535828.2535835]

[44] Panda A, Argyraki K, Sagiv M, Schapira M, Shenker S. New directions for network verification. In: Proc. of the 1st Summit on Advances in Programming Languages, SNAPL 2015. Asilomar: Schloss Dagstuhl-Leibniz-Zentrum fur Informatik, 2015. 209–220.

[45] Velner Y, Alpernas K, Panda A, Rabinovich A, Sagiv M, Shenker S, Shoham S. Some complexity results for stateful network verification. In: Proc. of the 22nd Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Eindhoven: Springer, 2016. 811–830. [doi: 10.1007/978-3-662-49674-9_51]

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

[47] Panda A, Lahav O, Argyraki K, Sagiv M, Shenker S. Verifying reachability in networks with mutable datapaths. In: Proc. of the 14th USENIX Conf. on Networked Systems Design and Implementation. Boston: USENIX Association, 2017. 699–718.

[48] Alpernas K, Manevich R, Panda A, Sagiv M, Shenker S, Shoham S, Velner Y. Abstract interpretation of stateful networks. In: Proc. of the Int'l Static Analysis Symp. Freiburg: Springer Int'l Publishing, 2018. 86–106. [doi: 10.1007/978-3-319-99725-4_8]

[49] Yuan YF, Moon SJ, Uppal S, Jia LM, Sekar V. NetSMC: A custom symbolic model checker for stateful network verification. In: Proc. of the 17th USENIX Conf. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2020. 181–200.

[50] Holzmann GJ. Design and Validation of Computer Protocols. Prentice-Hall, 1991.
[51] Intentionet. Move fast, don't break the network. https://www.intentionet.com/
[52] Forwardnetworks. Meet forward enterprise. https://forwardnetworks.com/
[53] Apstra. Data center networks. https://apstra.com/
[54] Handigol N. Intent-based verification leading a new wave of network automation. 2019. https://www.networkcomputing.com/networking/intent-based-verification-leading-new-wave-network-automation
[55] Cisco. Cisco intent-based networking (IBN). https://www.cisco.com/c/en/us/solutions/intent-based-networking.html
[56] Junipernetwork. What is intent-based networking? https://www.juniper.net/us/en/research-topics/what-is-intent-based-networking.html
[57] Huawei. Autonomous driving network (ADN). https://carrier.huawei.com/en/adn
[58] Zhang SY, Malik S, McGeer R. Verification of computer switching networks: An overview. In: Proc. of the 10th Int'l Conf. on Automated Technology for Verification and Analysis. Thiruvananthapuram: Springer, 2012. 1–16. [doi: 10.1007/978-3-642-33386-6_1]
[59] Qadir J, Hasan O. Applying formal methods to networking: Theory, techniques, and applications. IEEE Communications Surveys & Tutorials, 2015, 17(1): 256–291. [doi: 10.1109/COMST.2014.2345792]
[60] Li YH, Yin X, Wang ZL, Yao JY, Shi XG, Wu JP, Zhang H, Wang Q. A survey on network verification and testing with formal methods: Approaches and challenges. IEEE Communications Surveys & Tutorials, 2019, 21(1): 940–969. [doi: 10.1109/COMST.2018.2868050]
[61] Zhang XL, Wang C, Li Q, Wu JP. Toward comprehensive network verification: Practices, challenges and beyond. IEEE Network, 2020, 34(1): 108–115. [doi: 10.1109/MNET.001.1800330]
[62] Kreutz D, Ramos FMV, Veríssimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: A comprehensive survey. Proc. of the IEEE, 2015, 103(1): 14–76. [doi: 10.1109/JPROC.2014.2371999]
[63] Network verification—When Hoare meets cerf. https://www2.eecs.berkeley.edu/Colloquium/Archives/15-16/Fall2015/varghese.shtml
[64] Narain S, Levin G, Malik S, Kaul V. Declarative infrastructure configuration synthesis and debugging. Journal of Network and Systems Management, 2008, 16(3): 235–258. [doi: 10.1007/s10922-008-9108-y]
[65] 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]
[66] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69–74. [doi: 10.1145/1355734.1355746]
[67] Shenker S. Software-defined networking at the crossroads. 2013. https://www.youtube.com/watch?app=desktop&v=WabdXYzCAOU
[68] Fonseca PC, Mota ES. A survey on fault management in software-defined networks. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2284–2321. [doi: 10.1109/COMST.2017.2719862]
[69] Yu YB, Li X, Leng X, Song LB, Bu K, Chen Y, Yang JF, Zhang L, Cheng K, Xiao X. Fault management in software-defined networking: A survey. IEEE Communications Surveys & Tutorials, 2019, 21(1): 349–392. [doi: 10.1109/COMST.2018.2868922]
[70] Brim SW, Carpenter BE. Middleboxes: Taxonomy and issues. RFC 3234, 2002. https://rfc-editor.org/rfc/rfc3234.txt
[71] Wang J, Zhan NJ, Feng XY, Liu ZM. Overview of formal methods. Ruan Jian Xue Bao/Journal of Software, 2019, 30(1): 33−61 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5652.htm [doi: 10.13328/j.cnki.jos.005652]
[72] Wing JM. A specifier's introduction to formal methods. Computer, 1990, 23(9): 8–22. [doi: 10.1109/2.58215]
[73] Dahl OJ, Dijkstra EW, Hoare CAR. Structured Programming. London: Academic Press, 1972.
[74] Hall A. Seven myths of formal methods. IEEE Software, 1990, 7(5): 11–19. [doi: 10.1109/52.57887]
[75] CCF. CCF China computer science and technology development report 2017–2018. China Machine Press, 2018. 1–68 (in Chinese).
[76] Clarke EM, Emerson EA, Sistla AP. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. on Programming Languages and Systems, 1986, 8(2): 244–263. [doi: 10.1145/5397.5399]
[77] King JC. Symbolic execution and program testing. Communications of the ACM, 1976, 19(7): 385–394. [doi: 10.1145/360248.360252]
[78] Chang CL, Lee RCT. Symbolic Logic and Mechanical Theorem Proving. Sea Harbor Drive Orlando: Academic Press, 1997.
[79] Biere A, Heule M, van Maaren H, Walsh T. Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications. Washington: IOS Press, 2009.
[80] Cousot P, Cousot R. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proc. of the 4th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages. Los Angeles: Association for Computing Machinery, 1977. 238–252. [doi: 10.1145/512950.512973]
[81] Burch JR, Clarke EM, McMillan KL, Dill DL, Hwang LJ. Symbolic model checking: 10^{20} States and beyond. Information and

Computation, 1992, 98(2): 142–170. [doi: 10.1016/0890-5401(92)90017-A]

[82] Clarke E, Biere A, Raimi R, Zhu YS. Bounded model checking using satisfiability solving. Formal Methods in System Design, 2001, 19(1): 7–34. [doi: 10.1023/A:1011276507260]

[83] De Moura L, Bjørner N. Satisfiability modulo theories: Introduction and applications. Communications of the ACM, 2011, 54(9): 69–77. [doi: 10.1145/1995376.1995394]

[84] Beyer D, Keremoglu ME. CPAchecker: A tool for configurable software verification. In: Proc. of the 23rd Int'l Conf. on Computer Aided Verification. Snowbird: Springer, 2011. 184–190. [doi: 10.1007/978-3-642-22110-1_16]

[85] Griffin TG, Wilfong G. On the correctness of IBGP configuration. In: Proc. of the 2002 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications. Pittsburgh: Association for Computing Machinery, 2002. 17–29. [doi: 10.1145/633025.633028]

[86] Al-Shaer ES, Hamed HH. Discovery of policy anomalies in distributed firewalls. In: Proc. of the 2004 IEEE INFOCOM. Hong Kong: IEEE, 2004. 2605–2616. [doi: 10.1109/INFCOM.2004.1354680]

[87] Yuan LH, Chen H, Mai JN, Chuah CN, Su ZD, Mohapatra P. FIREMAN: A toolkit for firewall modeling and analysis. In: Proc. of 2006 IEEE Symp. on Security and Privacy. Berkeley/Oakland: IEEE, 2006. 199–213. [doi: 10.1109/SP.2006.16]

[88] Feamster N, Balakrishnan H. Detecting BGP configuration faults with static analysis. In: Proc. of the 2nd Conf. on Symp. on Networked Systems Design & Implementation. Berkeley: USENIX Association, 2005. 43–56.

[89] Xie GG, Zhan JB, Maltz DA, Zhang H, Greenberg A, Hjalmtysson G, Rexford J. On static reachability analysis of IP networks. In: Proc. of the 24th IEEE Annual Joint Conf. of the IEEE Computer and Communications Societies. Miami: IEEE, 2005. 2170–2183. [doi: 10.1109/INFCOM.2005.1498492]

[90] Network verification in the light of program verification. https://www.microsoft.com/en-us/research/publication/network-verification-in-the-light-of-program-verification/

[91] Huang SS, Green TJ, Loo BT. Datalog and emerging applications: An interactive tutorial. In: Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data. Athens: Association for Computing Machinery, 2011. 1213–1216. [doi: 10.1145/1989323.1989456]

[92] Griffin TA, Huston G. BGP wedgies. RFC 4264. 2005. https://rfc-editor.org/rfc/rfc4264.txt

[93] Parr TJ, Quong RW. ANTLR: A predicated-*LL*(*k*) parser generator. Software: Practice and Experience, 1995, 25(7): 789–810. [doi: 10.1002/spe.4380250705]

[94] Batfish. Batfish. https://github.com/batfish/batfish

[95] Griffin TG, Shepherd FB, Wilfong G. The stable paths problem and interdomain routing. IEEE/ACM Trans. on Networking, 2002, 10(2): 232–243. [doi: 10.1109/90.993304]

[96] Griffin TG, Sobrinho JL. Metarouting. In: Proc. of the 2005 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications. Philadelphia: Association for Computing Machinery, 2005. 1–12. [doi: 10.1145/1080091.1080094]

[97] VMware. Differential-Datalog. https://github.com/vmware/differential-datalog

[98] Sherry J, Hasan S, Scott C, Krishnamurthy A, Ratnasamy S, Sekar V. Making middleboxes someone else's problem: Network processing as a cloud service. In: Proc. of the 2012 ACM SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication. Helsinki: Association for Computing Machinery, 2012. 13–24. [doi: 10.1145/2342356.2342359]

[99] Potharaju R, Jain N. Demystifying the dark side of the middle: A field study of middlebox failures in datacenters. In: Proc. of the 2013 Conf. on Internet Measurement Conf. Barcelona: Association for Computing Machinery, 2013. 9–22. [doi: 10.1145/2504730.2504737]

[100] Ball T, Bjørner N, Gember A, Itzhaky S, Karbyshev A, Sagiv M, Schapira M, Valadarsky A. VeriCon: Towards verifying controller programs in software-defined networks. In: Proc. of the 35th ACM SIGPLAN Conf. on Programming Language Design and Implementation. Edinburgh: Association for Computing Machinery, 2014. 282–293. [doi: 10.1145/2594291.2594317]

[101] Anderson CJ, Foster N, Guha A, Jeannin JB, Kozen D, Schlesinger C, Walker D. NetKAT: Semantic foundations for networks. In: Proc. of the 41st ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. San Diego: Association for Computing Machinery, 2014. 113–126. [doi: 10.1145/2535838.2535862]

[102] Skowyra R, Lapets A, Bestavros A, Kfoury A. A verification platform for SDN-enabled applications. In: Proc. of the 2014 IEEE Int'l Conf. on Cloud Engineering. Boston: IEEE, 2014. 337–342. [doi: 10.1109/IC2E.2014.72]

[103] Nelson T, Ferguson AD, Scheer MJG, Krishnamurthi S. Tierless programming and reasoning for software-defined networks. In: Proc. of the 11th USENIX Conf. on Networked Systems Design and Implementation. Seattle: USENIX Association, 2014. 519–531.

[104] Li YH, Wang ZL, Yao JY, Yin X, Shi XG, Wu JP, Zhang H. MSAID: Automated detection of interference in multiple SDN applications. Computer Networks, 2019, 153: 49–62. [doi: 10.1016/j.comnet.2019.01.042]

[105] Zeng HY, Kazemian P, Varghese G, McKeown N. Automatic test packet generation. In: Proc. of the 8th Int'l Conf. on Emerging

Networking Experiments and Technologies. Nice: Association for Computing Machinery, 2012. 241–252. [doi: 10.1145/2413176.2413205]

[106] Fayaz SK, Yu TL, Tobioka Y, Chaki S, Sekar V. BUZZ: Testing context-dependent policies in stateful networks. In: Proc. of the 13th USENIX Conf. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2016. 275–289.

[107] Ruchansky N, Proserpio D. A (not) NICE way to verify the openflow switch specification: Formal modelling of the openflow switch using alloy. In: Proc. of the 2013 ACM SIGCOMM Conf. on SIGCOMM. Hong Kong: Association for Computing Machinery, 2013. 527–528. [doi: 10.1145/2486001.2491711]

[108] Canini M, Venzano D, Perešíni P, Kostić D, Rexford J. A NICE way to test openflow applications. In: Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation. San Jose: USENIX Association, 2012. 127–140.

[109] Scott C, Wundsam A, Raghavan B, *et al*. Troubleshooting blackbox SDN control software with minimal causal sequences. In: Proc. of the 2014 ACM Conf. on SIGCOMM. Chicago: Association for Computing Machinery, 2014. 395–406. [doi: 10.1145/2619239.2626304]

[110] Gember-Jacobson A, Akella A, Mahajan R, Liu HH. Automatically repairing network control planes using an abstract representation. In: Proc. of the 26th Symp. on Operating Systems Principles. Shanghai: Association for Computing Machinery, 2017. 359–373. [doi: 10.1145/3132747.3132753]

[111] El-Hassany A, Tsankov P, Vanbever L, Vechev M. Netcomplete: Practical network-wide configuration synthesis with autocompletion. In: Proc. of the 15th USENIX Conf. on Networked Systems Design and Implementation. Renton: USENIX Association, 2018. 579–594.

[112] 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 ACM Special Interest Group on Data Communication. Beijing: Association for Computing Machinery, 2019. 214–226. [doi: 10.1145/3341302.3342088]

[113] Abhashkumar A, Gember-Jacobson A, Akella A. AED: Incrementally synthesizing policy-compliant and manageable configurations. In: Proc. of the 16th Int'l Conf. on Emerging Networking Experiments and Technologies. Barcelona: Association for Computing Machinery, 2020. 482–495. [doi: 10.1145/3386367.3431304]

[114] El-Hassany A, Tsankov P, Vanbever L, Vechev M. Network-wide configuration synthesis. In: Proc. of the 29th Int'l Conf. on Computer Aided Verification. Heidelberg: Springer, 2017. 261–281. [doi: 10.1007/978-3-319-63390-9_14]

[115] Beckett R, Mahajan R, Millstein T, Padhye J, Walker D. Network configuration synthesis with abstract topologies. In: Proc. of the 38th ACM SIGPLAN Conf. on Programming Language Design and Implementation. Barcelona: Association for Computing Machinery, 2017. 437–451. [doi: 10.1145/3062341.3062367]

[116] Beckett R, Mahajan R, Millstein T, Padhye J, Walker D. Don't mind the gap: Bridging network-wide objectives and device-level configurations. In: Proc. of the 2016 ACM SIGCOMM Conf. Florianopolis: Association for Computing Machinery, 2016. 328–341. [doi: 10.1145/2934872.2934909]

[117] Liu HH, Zhu YB, Padhye J, Cao JX, Tallapragada S, Lopes NP, Rybalchenko A, Lu GH, Yuan LH. CrystalNet: Faithfully emulating large production networks. In: Proc. of the 26th Symp. on Operating Systems Principles. Shanghai: Association for Computing Machinery, 2017. 599–613. [doi: 10.1145/3132747.3132759]

[118] Handigol N, Heller B, Jeyakumar V, Lantz B, McKeown N. Reproducible network experiments using container-based emulation. In: Proc. of the 8th Int'l Conf. on Emerging Networking Experiments and Technologies. Nice: Association for Computing Machinery, 2012. 253–264. [doi: 10.1145/2413176.2413206]

[119] Wette P, Dräxler M, Schwabe A, Wallaschek F, Zahraee MH, Karl H. MaxiNet: Distributed emulation of software-defined networks. In: Proc. of the 2014 IFIP Networking Conf. Trondheim: IEEE, 2014. 1–9. [doi: 10.1109/IFIPNetworking.2014.6857078]

[120] Singh A, Aujla GS, Bali RS. Intent-based network for data dissemination in software-defined vehicular edge computing. IEEE Trans. on Intelligent Transportation Systems, 2021, 22(8): 5310–5318. [doi: 10.1109/TITS.2020.3002349]

[121] Zhang Y, Wu WF, Banerjee S, Kang JM, Sanchez MA. SLA-verifier: Stateful and quantitative verification for service chaining. In: Proc. of the 2017 IEEE Conf. on Computer Communications. Atlanta: IEEE, 2017. 1–9. [doi: 10.1109/INFOCOM.2017.8057041]

[122] Zhang H, Zhang C, de Amorim AA, Agarwal Y, Fredrikson M, Jia LM. Netter: Probabilistic, stateful network models. In: Proc. of the 22nd Int'l Conf. on Verification, Model Checking, and Abstract Interpretation. Copenhagen: Springer, 2021. 486–508. [doi: 10.1007/978-3-030-67067-2_22]

[123] Gember-Jacobson A, Raiciu C, Vanbever L. Integrating verification and repair into the control plane. In: Proc. of the 16th ACM Workshop on Hot Topics in Networks. Palo Alto: Association for Computing Machinery, 2017. 129–135. [doi: 10.1145/3152434.3152439]

[124] Shrestha R, Sun X, Gember-Jacobson A. Localizing router configuration errors using unsatisfiable cores. In: Proc. of the 2019 USENIX Symp. on Networked Systems Design and Implementation (NSDI). Boston: USENIX Association, 2019.

[125] Tang A, Kakarla SKR, Beckett R, Zhai EN, Brown M, Millstein T, Tamir Y, Varghese G. Campion: Debugging router configuration differences. In: Proc. of the 2021 ACM SIGCOMM Conf. New York: ACM, 2021. 748–761. [doi: 10.1145/3452296.3472925]

[126] Beckett R, Mahajan R. Network verification 2.0. 2020. https://netverify.fun/network-verification-2-0/

[127] Beckett R, Mahajan R. Putting network verification to good use. In: Proc. of the 18th ACM Workshop on Hot Topics in Networks. Princeton: Association for Computing Machinery, 2019. 77–84. [doi: 10.1145/3365609.3365866]

[128] Xu XY, Beckett R, Jayaraman K, Mahajan R, Walker D. Test coverage metrics for the network. In: Proc. of the 2021 ACM SIGCOMM Conf. Association for Computing Machinery, 2021. 775–787. [doi: 10.1145/3452296.3472941]

[129] Beckett R, Mahajan R. A general framework for compositional network modeling. In: Proc. of the 19th ACM Workshop on Hot Topics in Networks. New York: Association for Computing Machinery, 2020. 8–15. [doi: 10.1145/3422604.3425930]

[130] Birkner R, Drachsler-Cohen D, Vanbever L, Vechev MT. Config2Spec: Mining network specifications from network configurations. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2020. 969–984.

[131] Kakarla SKR, Tang AL, Beckett R, Jayaraman K, Millstein T, Tamir Y, Varghese G. Finding network misconfigurations by automatic template inference. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation (NSDI 2020). Santa Clara: USENIX Association, 2020. 999–1013.

[132] Kheradmand A. Automatic inference of high-level network intents by mining forwarding patterns. In: Proc. of the 2020 Symp. on SDN Research. San Jose: Association for Computing Machinery, 2020. 27–33. [doi: 10.1145/3373360.3380831]

[133] Birkner R, Brodmann T, Tsankov P, Vanbever L, Vechev MT. Metha: Network verifiers need to be correct too! In: Proc. of the 18th USENIX Symp. on Networked Systems Design and Implementation (NSDI 2021). USENIX Association, 2021. 99–113.

附中文参考文献:

[65] 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]

[71] 王戟, 詹乃军, 冯新宇, 刘志明. 形式化方法概貌. 软件学报, 2019, 30(1): 33–61. http://www.jos.org.cn/1000-9825/5652.htm [doi: 10.13328/j.cnki.jos.005652]

[75] 中国计算机学会. CCF2017–2018中国计算机科学技术发展报告. 北京: 机械工业出版社, 2018. 1–68.

方星(1997－), 男, 硕士生, 主要研究领域为网络验证, 网络安全, 计算机网络.

马超(1991－), 男, 工程师, 主要研究领域为网络风险评估, 网络验证.

胡波(1981－), 男, 硕士, 高级工程师, 主要研究领域为网络安全, 网络验证.

黄伟庆(1972－), 男, 博士, 正高级工程师, 博士生导师, CCF 高级会员, 主要研究领域为网络安全, 物联网安全, 云计算安全, 信号处理理论与技术, 电磁声光检测与防护.