基于大语言模型的SDN网络自动化配置研究

杨兴源,田 乐,姚 莹,潘 璠,胡宇翔*
(信息工程大学,河南郑州 450001)

摘 要: 传统网络依赖人工配置,在应对规模激增、需求复杂化及实时性要求提升的现代网络环境时,效率低下且成本高昂. 大语言模型(Large Language Model,LLM)凭借其出色的自然语言理解能力,在网络自动化配置中展现出巨大的潜力. 面向软件定义网络(Software Defined Networking,SDN),本文提出了一种基于LLM的轻量级自动化配置方法. 在数据平面,提出了一种基于检索增强生成(Retrieval-Augmented Generation,RAG)技术的代码自动生成方法RetroP4,支持基于用户意图生成P4代码;在控制平面,提出了一种基于任务分解的流表自动生成方法CtrlSynth,支持基于用户意图和数据平面P4代码生成流表配置. 实验结果表明:相较于通用大模型,RetroP4生成的P4代码的语法正确性提高了25%,语义正确性提高了87.5%;CtrlSynth能够准确生成与P4代码匹配的流表信息,在流量意图不超过300条时,准确率可达100%.

关键词: 大语言模型(LLM);网络配置;软件定义网络(SDN);可编程协议无关报文处理;检索增强生成(RAG)
基金项目: 国家重点研发计划(No.2022YFB2901501);中原科技创新领军人才项目(No.244200510038);河南省重大科技专项项目(No.221100210900-02)
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112(2025)09-3078-11
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250463

Research on SDN Network Automated Configuration Based on Large Language Models

YANG Xing-yuan, TIAN Le, YAO Ying, PAN Fan, HU Yu-xiang*
(*University of Information Engineering*, *Zhengzhou*, *Henan* 450001, *China*)

Abstract: Traditional networks, which depend on manual configuration, are inefficient and expensive in the face of today's rapidly expanding scales, increasingly complex demands, and the growing need for real-time responsiveness. Large language models (LLM), known for their exceptional ability to understand natural language, show immense promise for automating network configurations. This paper introduces a streamlined approach to automated configuration for software defined networking (SDN), leveraging LLM. For the data plane, we present RetroP4, a code generation method that uses retrieval-augmented generation (RAG) technology, enabling the creation of P4 code tailored to users' intentions. In the control plane, we propose CtrlSynth, a method for automatically generating flow tables by breaking down tasks, aligning the configurations with users' intentions and the P4 code from the data plane. Compared with general-purpose large models, the syntactic correctness of P4 code generated by RetroP4 is improved by 25%, and the semantic correctness is enhanced by 87.5%. CtrlSynth accurately produces flow table information that corresponds to the P4 code, achieving a 100% accuracy rate when dealing with up to 300 traffic-related intentions.

Key words: large language models (LLM); network configuration; software defined network (SDN); programming protocol-independent packet processors (P4); retrieval-augmented generation (RAG)
Foundation Item(s): National Key Research and Development Program (No.2022YFB2901501); Zhongyuan Science and Technology Innovation Leading Researcher Project (No.244200510038); Major Science and Technology Special Project of Henan Province (No.221100210900-02)

收稿日期:2025-05-31;录用日期:2025-09-11;责任编辑:覃怀银
*通讯作者:胡宇翔

1 引言

随着5G通信、物联网和云计算技术的发展,现代网络架构正朝着规模化、异构化和动态化方向演进. 面对日益激增的设备数量、复杂的服务需求以及实时性业务调度挑战,传统的网络配置模式(依赖于人工操作)已经难以应对.

在过去十年中,软件定义网络(Software Defined Networking,SDN)因其控制平面与数据平面分离的特性,被学术界和工业界广泛采纳,以简化网络配置[1]. SDN通过集中化的软件控制器管理数据平面设备的配置,显著提升了网络的可编程性和灵活性,其理念也被拓展至各类垂直行业网络[2]. 然而,尽管SDN架构提供了便利,其配置仍需要频繁的人工干预. 手动配置不仅成本高昂[例如,需要精通各种不同应用程序编程接口(Application Programming Interface,API)和协议的专家开发人员]、难度大,而且容易引入人为错误,导致配置冲突或性能下降.

为应对这一挑战,已有研究尝试将网络管理员指定的高级策略编译为每个网络设备的配置,减少人工直接操作,从而达到简化配置的效果[3,4]. 这些研究大多集中于基于模板、规则或形式方法的配置生成[5~7]. 例如,Contra[4]和NetComplete[7]通过策略编译技术和形式化方法,将高级策略转换为设备配置,显著减少了人工干预. 然而,上述方法通常依赖预定义的策略语言和编译规则,缺乏对自然语言意图的理解能力.

近年来,生成式人工智能,特别是大语言模型(Large Language Models,LLM)在自然语言理解与生成、编写代码、与用户对话以及跨任务泛化能力等方面显示出巨大潜力[8~11]. 值得注意的是,LLM能够深入理解用户意图,有望将网络管理员描述的高级策略转化为网络设备可执行的低级配置指令[12],从而大幅简化网络配置过程,降低管理门槛和成本.

已有研究表明:LLM能够生成网络配置代码,文献[13]首次系统性地评估了GPT-4在生成传统路由器(如Cisco、Juniper)配置方面的能力,但其工作聚焦于传统分布式网络的配置翻译和简单策略实现,并未涉及SDN架构. 在通用代码生成领域,文献[10]研究了LLM的代码生成能力,证明了其在编程任务上的潜力,但这些工作并非针对网络配置领域,缺乏对网络特定语法、语义和约束的深度优化.

检索增强生成(Retrieval-Augmented Generation,RAG)技术通过检索外部知识增强生成质量,已在代码生成和问答系统等领域得到广泛应用[14,15]. 然而,该技术在网络配置生成任务中的应用尚未得到充分探索. 另一方面,思维链(Chain-of-Thought,CoT)推理通过将复杂任务分解为多步子任务,有效提升了LLM在逻辑推理任务中的性能. 文献[16]提出的CoT提示方法虽然在复杂推理任务上的表现突出,但在网络配置生成领域的实际应用仍相对有限.

基于此背景,本文将RAG和CoT技术与SDN网络自动化配置的特定挑战相结合,致力于探索利用LLM实现SDN网络自动化配置的新方法. 本文提出一种基于LLM的轻量级自动化配置方案,主要聚焦于解决SDN中数据平面和控制平面的配置生成问题,主要贡献如下:

(1)提出基于RAG的数据平面代码生成方法RetroP4. 针对直接使用LLM生成P4代码准确率低的问题,创新性地将RAG技术引入数据平面配置.

(2)提出基于任务分解的控制平面配置生成方法CtrlSynth. 针对将复杂自然语言策略直接转换为低级配置的难题,设计了一种多阶段、任务分解的流程.

(3)提出并验证了端到端的SDN自动化配置可行性方案. 将RetroP4与CtrlSynth结合,在仿真环境中论证RetroP4结合CtrlSynth自动化配置SDN网络的有效性.

2 基于RAG的数据平面代码生成

本节介绍基于LLM和RAG的P4数据平面代码生成器RetroP4. RetroP4接收用户以自然语言指定编程需求,并将其转化为实际可执行的P4数据平面代码.

由于编写P4程序需要相关领域知识的支持,其复杂的语法和结构特性对于直接使用中小规模LLM进行准确生成构成了挑战. 同时考虑到实际情况中,用户不具备对LLM进行领域微调(Fine-tuning)的条件,本文提出采用RAG技术作为替代方案. 该方法旨在使LLM能够有效理解并遵循P4程序的结构规范和语法要求.

RAG从预设的P4领域知识库中动态检索与用户当前要求相关的信息片段. 这些检索到的信息随后被注入到LLM的上下文中,与用户原始的自然语言指令共同构成LLM的输入. 通过这种方式,RAG为LLM提供了必要的领域背景知识,显著提升了其生成P4代码的准确性和相关性.

2.1 RetroP4概述

RetroP4的工作流程概述如图1所示,用户首先通过自然语言描述需要生成的P4程序的功能需求,然后将需求传递给P4程序生成模块. P4程序生成模块利用RAG根据用户需求从预设的P4领域知识库中进行检索,并将检索到的相关知识与用户原始需求共同构成LLM的输入提示. LLM基于此增强的上下文信息生成对应的P4程序. 随后,将生成的P4程序输入到验证器中编译,以检验是否存在语法错误. 若编译失败,验证器产生的错误信息将被捕获并反馈给P4程序生成模

块,P4程序生成模块将错误信息与原始用户需求提交给LLM,LLM根据错误信息尝试修改代码并重新生成.上述生成-编译-反馈循环重复运行,直到P4程序顺利编译或达到最大尝试次数.

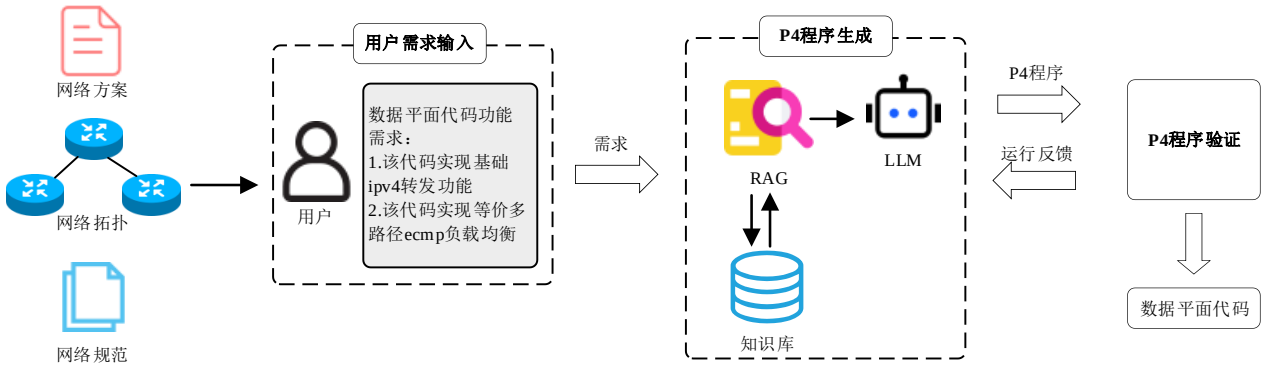


图1 RetroP4工作流程

2.2 用户需求输入

用户根据给定的网络方案、网络拓扑结构和网络规范,分析网络应具备的属性,并使用自然语言描述P4程序生成需求.该生成需求应聚焦于数据平面的功能,即对数据包的操作,例如基础ipv4转发、防火墙、泛洪等.

2.3 P4程序生成

考虑到P4程序固有的语法复杂性以及通用LLM直接生成正确P4程序可能面临的困难,本文采用RAG技术来提升RetroP4的P4程序生成能力.该方法的核心在于利用一个专门构建的P4领域知识库,该知识库包含了高质量的P4程序示例及其功能说明文档.通过RAG机制,RetroP4能够根据用户的自然语言输入请求,动态地从该知识库中检索并注入相关的P4语法结构、编程范式或功能实现示例等知识到LLM的上下文中.这种方式显著增强了LLM对P4编程规范的理解,从而有效提高了生成代码在语法正确性和语义准确性方面的质量.

2.3.1 知识库构建

为了克服开源通用LLM内部缺乏特定P4领域知识所导致的生成程序质量低、可用性差的问题,本文为RetroP4构建了一个专门用于辅助P4程序生成的知识库.该知识库包含两个核心组成部分:

(1)P4程序集:包含由P4官方编写和维护的、具有代表性的高质量P4程序示例.

(2)P4程序说明文档集:为了增强对程序功能的理解,本文利用通用LLM为知识库中的每个P4程序生成了对应的说明文档.这些文档的生成是基于P4官方提供的项目说明和程序代码本身.如图2所示,生成的说明文档清晰地描述了相应P4程序实现的总体功能及其内部关键模块的具体功能.这些文档可以帮助RetroP4理解相应程序实现的功能,从而进一步提高生成P4程序的质量.

功能描述:
1. 该P4程序实现了一个状态防火墙,主要功能是在交换机S1上过滤来自外部网络(例如主机h2)的连接请求,仅允许内部网络(例如主机h1)发起的连接,并允许外部网络对这些已建立的连接进行响应。
2. 程序使用Bloom Filter来跟踪和检查连接状态。当内部网络发起一个新的传输控制协议(Transmission Control Protocol,TCP)连接时(通过SYN标志位判断),Bloom Filter会被更新以记录这个连接。
3. 当外部网络尝试发起新的连接时,程序会检查Bloom Filter以确定是否已经存在对应的连接记录。如果不存在,则丢弃该数据包;如果存在,则允许数据包通过。
4. 对于有效的IPv4数据包,程序还会执行最长前缀匹配(LPM)查找,以确定转发端口,并设置相应的MAC地址和出端口信息。
5. 程序还处理了数据包的校验和计算与验证,确保数据包在传输过程中的完整性。
6. 在数据包处理过程中,程序会根据需要修改数据包的TTL值,并设置或更新数据包的源和目的MAC地址。
7. 最后,程序将处理后的数据包重新组装并发送出去。

图2 P4程序说明文档

为了支持高效的语义检索,本文将P4程序说明文档集按照文档进行分块处理,每个内容块随后使用text-embedding-3-small向量编码模型进行向量化.最终,这些向量化的文档块构成了RetroP4知识库的检索索引.

2.3.2 知识检索

RetroP4在处理具体的用户P4程序生成请求时,首先触发RAG的知识检索过程,如算法1所示,其通过比较欧氏距离来实现.

第一步,将用户的自然语言需求使用相同的text-embedding-3-small模型转化为查询向量.

第二步,在向量化的知识库索引中,计算查询向量与所有文档块向量之间的欧氏距离,其目的是找出与当前用户需求语义上相关的内容块.

第三步,返回相似度最高的前k个文档块(在本文中,k取3作为一个经验性取值)作为检验结果.同时关联并获取这些文档块所对应的P4程序代码.

第四步,检索到的文档块内容及其关联的P4程序代码被整合后传递给LLM,作为其生成P4代码的上下文信息.LLM综合用户的原始需求和这些增强的上下文知识,生成目标P4程序代码.

算法1　RAG检索过程

输入: P4程序生成需求r;P4说明文档df;返回检索结果个数k
输出: 检索结果s

```
cnt = 0; k = 0
s = []
r_vec = []
distances = []
r_vec = text-embedding-3-small(r);//向量化需求文本
FOR EACH doc IN df DO:
    distance = EUCLIDEAN_DISTANCE(r_vec, doc.vector)
    distances.append((distance, doc.content))
SORT_BY_ASCENDING(distances);// 按距离升序排序
WHILE cnt < min(k, LENGTH(distances)) DO:
    s.append(distances[cnt].content)
    cnt = cnt + 1
RETURN s
```

2.4　P4程序验证

在P4程序生成后,评估其质量是至关重要的,因为只有语法和基本语义正确的P4程序才能在目标平台上部署运行.本文使用P4官方编译器p4c作为验证工具,对RetroP4生成的P4程序进行编译检查.该过程可以检测代码中的语法错误和静态语义错误.对于RetroP4生成的每一个P4程序,如果能够被验证工具成功编译,那么其在语法和基础语义层面是正确的.否则,该程序存在语法或静态语义问题.

当编译失败时,p4c编译器会产生详细的错误报告,包括错误信息与错误位置.考虑到LLM的输入受到上下文长度限制,即无法无限制地提供所有历史信息,同时实验观察表明:仅提供最近一次的报错信息会导致LLM在同一个错误点反复修改而难以突破.因此,当验证器反馈错误时,会收集并返回最近三次(包含当前)编译失败的P4程序与相应的完整编译错误报告.RetroP4会整合这些历史错误信息,并将其与原始用户需求一起输入LLM,该步骤的提示词具体设计如图3所示.LLM基于这些更丰富的上下文信息生成修改后的P4程序.上述过程持续进行,直到P4程序成功编译或者达到最大编译次数.

```
需要修复最近P4代码编译错误，有往次历史代码和错误信息参考。请：
1. 保持原有功能
2. 修正编译错误
3. 保留必要注释
4. 不要出现往次历史代码的错误
原始需求：{description}
前两次P4代码：{previous_code3}
前两次错误信息：{error_log3}
前一次P4代码：{previous_code2}
前一次错误信息：{error_log2}
最近P4代码：{previous_code1}
最近错误信息：{error_log1}
请输出修正后的完整P4代码，用"P4"包裹
```

图3　P4程序错误反馈提示词设计

3　基于任务分解的控制平面配置生成

本节介绍本文提出的基于LLM的P4控制平面配置生成器CtrlSynth(Control plane configuration Synthesizer).CtrlSynth接收用户以自然语言描述的高级策略和要求,并将其转换为控制平面配置.

直接将复杂的自然语言策略一次性转换为精确的低级配置指令,对于未经优化的通用LLM来说极具挑战性.因此,为了提升LLM处理此类复杂任务的准确性和可控性,并使其能够更好地理解配置指令的生成逻辑,本文借鉴了CoT推理方法的优势[16].具体而言,本文设计了一个多阶段、任务分解的生成流程(如图4所示),将原本复杂的整体转换任务分解为一系列更简单、细粒度更高的子步骤,逐步完成从高级策略到低级配置的转换.

CtrlSynth通过四个步骤将高级策略和要求转化为控制平面配置,如图4的流程图所示,第一步与最后一步都由LLM执行,充分发挥其自然语义理解和生成能力;中间步骤使用确定性、规则驱动的程序处理,确保关键转换步骤的准确性和可解释性.

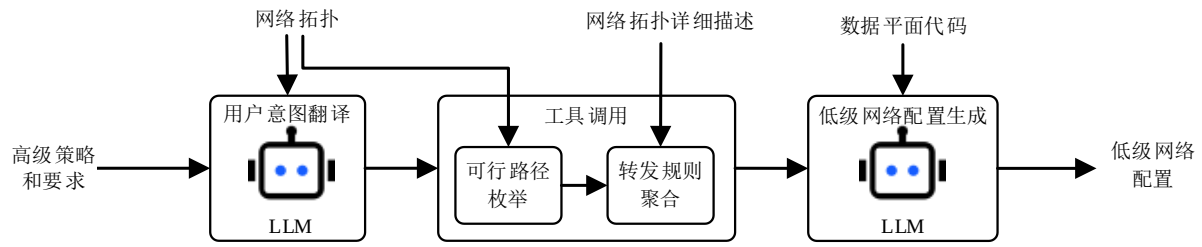


图4　CtrlSynth流程图

3.1 用户意图翻译

CtrlSynth 流程的第一步是理解用户的自然语言输入并将其转化为结构化的中间表示. 该步骤的目的是将用户描述的高级网络策略(如连接性、路径约束)精准捕获为一种预定义、规范化的数据结构,便于后续阶段进行确定性处理.

该中间表示是一种简洁的数据结构,主要包含通信主体与路径约束(必经节点和排除节点). 通信主体明确指定需要进行通信的源主机和目的主机,必经节点指定通信流必须经过的网络节点,排除节点指定通信流必须避开的网络节点. 该中间表示将作为后续阶段的基础输入.

将自由形式的自然语言策略无歧义地、可靠地映射到这种结构化表示是一项关键挑战,其准确性直接影响后续配置生成的质量. 为了提升翻译的鲁棒性和一致性,本文的实现参考了相关工作[17,18]的思路,将复杂的配置意图抽象和简化为核心的、可枚举的策略要素(如上述的通信对和路径约束).

图 5 展示了用户意图翻译的示例输入与输出. 由图 5 可知,用户输入包含自然语言策略描述与网络拓扑描述文件两部分. 自然语言策略描述中阐述网络的高级行为要求,包括指定哪些主机间可以通信(例如,h1 可以与 h3 通信),规定通信路径中必须经过或者避开哪些网络节点(例如,通信必须经过交换机 s1 且避开交换机 s2). 同时,策略也可以是模糊性的描述(例如,所有节点之间均可互通). 网络拓扑描述文件则是提供目标网络的连接信息,包括节点列表、链路关系. LLM 根据网络拓扑描述文件,验证用户意图中提及的主机和节点是否存在当前目标网络环境中,这是确保生成的意图表示具备物理可实现性的关键步骤.

```
用户输入：h1与h3可以通信经过s1不经过s2、s3
网络拓扑：{"host","switches","links"}
```

```
模型输出：
[
  {"interact":["h1","h3"],
  "waypoint":["s1"],
  "avoidance":["s2","s3"]}
]
```

图 5 用户意图翻译示例

3.2 可行路径枚举

在获得用户意图的结构化中间表示后(见 3.1 节),CtrlSynth 将基于给定的网络拓扑信息,枚举所有满足用户意图约束的、可行的端到端数据包传输路径. 此步骤的目标是让 CtrlSynth 对数据包的传递过程有一个整体认知. 在实际使用中发现,直接让 LLM 为交换机配置条目时,无法准确理解 IP 地址、转发端口与 MAC 地址的关系. 因此,需要让 LLM 建立一个关于 IP 地址、转发端口与 MAC 地址的整体认知.

本文采用预编写的确定性程序(而非 LLM)执行此步骤,原因是在实验中发现 LLM 在执行复杂的搜索任务时,执行效率低且容易出现错误. 此外,在大规模或复杂拓扑条件下,LLM 的推理速度难以满足实际应用要求. 而采用确定性程序,能够系统地搜索网络拓扑图,高效找出所有满足主机约束、必经节点和排除节点约束的可行性路径.

图 6 展示了可行路径枚举示例. 如图 6 所示,在该步骤输入包括用户意图的结构化中间表示(包含通信主机、路径约束)、网络拓扑描述(包含所有交换机、主机、链路连接关系). 随后,在拓扑图上执行搜索算法(本文使用深度搜索算法),找出所有从源主机到目的主机且满足必经节点和排除节点约束的有效路径. 该步骤的最终输出是一个结构化的路径集合,其中每条路径包含路径序列(经过的节点顺序).

```
输入：[
  {"interact":["h1","h3"],
  "waypoint":["s1"],
  "avoidance":["s2","s3"]}
]
网络拓扑：{"host","switches","links"}
```

```
输出：
从h1到h3的路径：
 - h1<->s1<->s4<->s5<->h3
 - h1<->s1<->s4<->s6<->h3
```

图 6 可行路径枚举示例

3.3 转发规则聚合

在该步骤中,CtrlSynth 根据 3.2 节生成的数据传输路径,提取目标交换机的规则需求集合. 由于交换机转发数据包是依据目标 IP 地址来决定下一跳的 MAC 地址和转发端口,因此需要明确在不同目标 IP 地址情况下,对应的下一跳 MAC 地址和转发端口. 该步骤的目的是对 3.2 节的路径枚举结果,逐路径提取转发规则的依赖关系,特别是 IP 地址、交换机出端口(Egress Port)与下一跳 MAC 地址之间的关键映射关系,并聚合生成面向交换机的配置视图.

在实验中发现,LLM 难以精确跟踪和管理所有可能的路径及其状态信息,并且在推导 IP->Port->MAC 映射关系的一致性和准确性方面表现不佳. 相比之下,采用确定性程序能够精确提取转发规则的依赖关系. 对于路径上的每一跳(即交换机),该程序能够根据拓扑信息(包括链路连接、端口以及邻接节点 MAC 地址)确定性地推导出该交换机转发此路径流量所需的精确匹配项(例如目标 IP 地址)和动作参数(例如出端口、下一跳 MAC 地址),并维护这些映射关系的一致性. 因此,本步骤采用确定性程序执行.

图7展示了规则需求集合示例. 该步骤的输入包括数据包传递路径以及详细的网络拓扑描述(涵盖所有交换机、主机、链路连接关系、端口信息以及相邻接口的MAC地址). 随后,对每条路径逐跳推导转发规则,并将分散的规则按交换机进行分组. 最终的输出结果是按交换机组织的规则需求集合,这些输出为下一阶段提供了精确的、机器可处理的输入基础.

```
交换机s1
从h1(10.0.1.1)到h3(10.0.7.3)的路径:
a.目标ip:10.0.7.3/32
  目标mac:1e:c1:83:84:84:0d
  出端口:4
b.目标ip:10.0.1.1/32
  目标mac:00:00:0a:00:01:01
  出端口:1
```

图7　规则需求集合示例

3.4　控制平面配置生成

作为最后一步,CtrlSynth生成最终的可部署控制平面配置. 该步骤根据3.3节的交换机规则需求集合以及已有的目标P4数据平面程序,生成符合目标P4数据平面程序接口规范的具体流表项配置命令. 图8展示了控制平面配置生成的示例.

```
table_set_default ipv4_lpm drop
table_add ipv4_lpm set_nhop 10.0.7.3/32 => 1e:c1:83:84:84:0d 4
table_add ipv4_lpm set_nhop 10.0.1.1/32 => 00:00:0a:00:01:01 1
```

图8　控制平面配置示例

在本步骤中,考虑到数据平面接口的多样性,即不同的P4程序定义了独特的流水线结构、匹配字段名称、动作名称及其参数列表,同时考虑到LLM在理解自然语言和结构化描述方面的优势,本步骤采用LLM生成流表命令. 后续的实验表明(见4.3.2节):LLM具有较好的生成控制平面配置的能力.

4　配置生成实验

本节进行了一些实验来评估RetroP4和CtrlSynth的有效性和性能.

4.1　实验环境

所有实验均在相同的硬件平台上运行,该平台配置为Windows11操作系统、Intel®Core™ i7-9700 CPU以及16 GB RAM. 在此硬件平台上,本文构建了虚拟机,虚拟机配置为8 GB内存,8个处理器内核,操作系统为64位Ubuntu. 本实验旨在评估本文提出的方法对Qwen2.5-72b和Qwen3-32b模型性能的影响. 这两个模型由阿里云推出,基于先进的LLM架构,专为复杂的自然语言理解和生成任务设计,在参数规模、技术优化、应用场景覆盖等方面均处于领先水平.

4.2　P4数据平面生成实验

本节将探讨RetroP4生成P4数据平面代码的有效性和性能. 本文选择了几种常见的P4数据平面实现的功能,对比RetroP4、带示例程序的LLM和LLM生成P4程序的性能,示例程序为P4程序结构样板. 实验结果如表1所示.

表1　RetroP4生成P4程序质量评估

实现功能	RetroP4	LLM+示例程序	LLM	RetroP4(Qwen3-32b)
	生成时间/尝试次数/是否通过编译/是否实现功能			
等价多路径负载均衡	62.06 s/1/是/是	39.69 s/6/是/否	30.12 s/10/否/否	50.87 s/1/是/是
基于流片段负载均衡	61.96 s/1/是/是	35.99 s/10/否/否	33.99 s/10/否/否	74.16 s/1/是/是
防火墙	59.16 s/1/是/是	25.81 s/1/是/否	36.86 s/10/否/否	87.80 s/1/是/是
大流量检测	52.74 s/1/是/是	31.47 s/2/是/否	26.73 s/10/否/否	87.22 s/1/是/是
基础ipv4转发	31.24 s/1/是/是	26.67 s/1/是/是	23.02 s/10/否/否	47.58 s/1/是/是
统计数据包	13.56 s/1/是/是	33.74 s/10/否/否	32.20 s/10/否/否	38.88 s/1/是/是
广播	35.16 s/1/是/是	25.34 s/1/是/否	19.00 s/10/否/否	36.51 s/1/是/是
速率控制	38.25 s/1/是/是	32.28 s/1/是/否	38.77 s/10/否/否	46.93 s/1/是/是

在表1中,当有多次尝试时,取每次生成时间的平均时间作为生成时间. 同时为了确保RetroP4遇到报错时会无休止地生成代码,本实验设置最大尝试次数为10次.

从实验结果可得,LLM生成P4程序结果不佳. 所有功能的P4代码生成都达到了最大尝试次数,同时生成的P4代码均未通过编译. LLM带示例程序的情况,生成P4代码的成功率有所提高,但仍有个别两个功能无法生成能够编译的代码. 经过检查,生成的能够编译的P4代码大多数没有实现用户要求功能,只有基础的ipv4转发功能得到实现. 其余功能在实现时出现定义该功能的动作框架,但未在其填充具体的实现;或者运行逻辑判断不准确,在实际运行时所定义的动作在错误的时机执行. RetroP4的编译成功率和功能实现率明显较高,相较于带示例程序的LLM在语法和语义正确率上分别提高了25%和87.5%. 对于上述所有的功能,RetroP4第一次生成便能够成功运行,同时生成代码符合预期要求.

对比不同条件下的代码生成时间可以发现，RetroP4生成时间较长．因为相较于不使用RAG的情况，使用RAG会提供更多的额外信息，RetroP4在处理更长的上下文时所用的时间会更多．同时发现，在实现统计数据包功能时，RetroP4反而用了更少的时间，经过检查发现，实现统计数据包功能的P4代码长度远小于其他功能的代码，使用RAG能够简化RetroP4生成P4代码的思考过程，从而导致生成代码的时间减少．而带示例程序的LLM和LLM需要思考如何实现该功能，因此生成时间较长．

对于Qwen3-32b模型，表1中的功能均能正确实现，但生成P4程序的时间普遍长于Qwen2.5-72b模型，除了在个别功能中，例如实现等价多路径（Equal Cost Multi Path，ECMP）负载均衡，短于Qwen2.5-72b模型．这可能因为其参数量较小，在某些复杂任务上需要更多的计算或推理步骤．

该实验表明：RetroP4具备根据用户需求生成实现相应功能的P4程序的能力，即使不使用RAG，仅提供部分参考信息也能够使LLM生成可编译的P4程序，同时LLM也具备纠正编译P4程序的能力，并且RetroP4在参数量更小的模型上也能正确生成P4程序．

4.3 P4控制平面配置生成实验

本节将探讨CtrlSynth配置P4控制平面的有效性和性能．本节分别探究CtrlSynth将用户需求转换为结构化中间表示的能力，以及将规则需求集合转化为控制平面配置的能力．在以下的两个实验中，每种需求数进行十次实验，取平均值作为最终结果．

4.3.1 生成规范化格式

本文首先生成20个网络拓扑，每个网络拓扑主机数不少于4个，交换机节点数不少于5个．其次，编写代码，针对每个网络拓扑随机生成了3~10个结构化中间表示，共计145个，每条中间表示包括通信主体与路径约束（必经节点和排除节点）．再次，使用LLM将其转换为自然语言要求，然后根据LLM生成的自然语言要求经过人工调整，修改表述错误的语句．最后，让CtrlSynth将这些要求从自然语言翻译成结构化中间表示，并通过将翻译后的中间表示与初始生成的中间表示进行比较，从而评估CtrlSynth的有效性和性能．实验结果如图9所示．

图中时间为翻译每条需求所用平均时间，不同线上的垂直线为标准差误差线．由实验结果中可知，在翻译需求数较少的情况下，Qwen2.5-72b模型翻译准确率为100%．当翻译需求数超过50条时，翻译准确率出现下降．通过查看翻译结果发现，当翻译需求量较大时，出现部分需求翻译漏翻的情况．尽管本文在提示词里明确要求了LLM对每条需求逐个翻译，从实验结果来看，较长的上下文可能会让LLM忽略之前的信息．Qwen3-32b模型在翻译需求数达到145条时，仍能保持接近100%的翻译准确率．这说明Qwen3-32b模型在效率和性能上可能有所优化，在特定任务上性能优于参数量更大的Qwen2.5-72b模型．

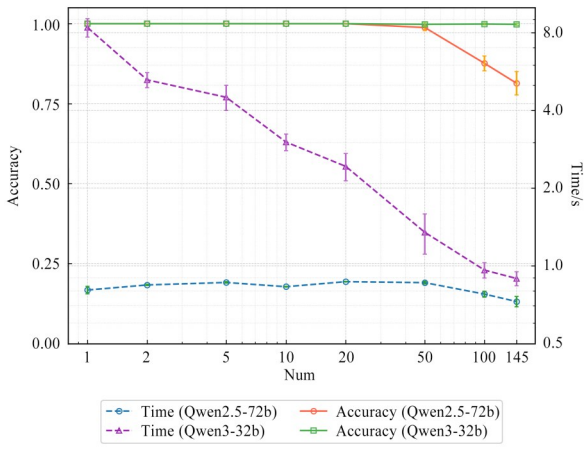


图9 翻译用户需求情况

同时，随着翻译需求数的增加，翻译每条需求所用时间逐渐减少，说明批量翻译可以降低翻译所用时间成本．因此，在实际翻译时，需要选择合适的翻译数量，以保证在准确性和时间成本之间进行正确的权衡．

4.3.2 生成控制平面配置

本文随机生成了500条规则需求，并保证生成的规则需求集合中的IP地址没有重复．同时，生成了对应规则需求集合的控制平面配置（由于在本实验中输入给CtrlSynth的P4程序是恒定的，因此可以根据该P4程序与规则需求集合生成对应的控制平面配置）．然后，要求CtrlSynth将这些规则需求集合转化为控制平面配置，通过将转化后的控制平面配置与预先设定的控制平面配置进行比较，从而评估CtrlSynth的有效性和性能．实验结果如图10所示．

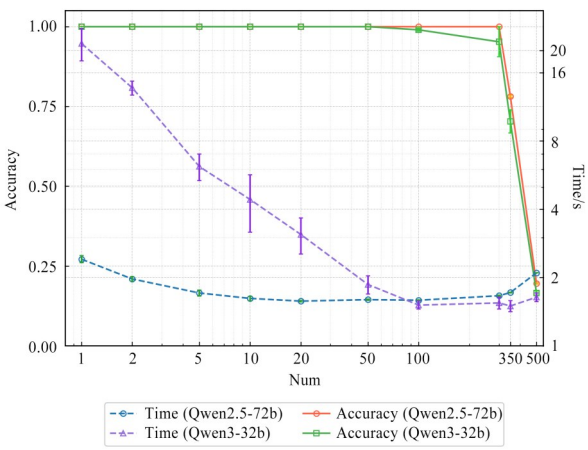


图10 生成控制平面配置的情况

对于Qwen2.5-72b模型,首先,随着规则需求数量增多,生成每条控制平面配置的用时更少. 其次,在规则需求不超过300条时,其生成准确率可达100%. 然而,当规则需求数量达到350条时,转化准确率出现下降. 在查看转化结果文件时发现,Qwen2.5-72b模型能稳定地生成前276条流表命令,并且该276条命令全部正确. 当规则需求数量达到500条时,Qwen2.5-72b模型正确地生成前100条流表命令. 说明此时已经达到模型Qwen2.5-72b的输出上限,剩下的流表命令被裁剪导致无法输出.

同样的,Qwen3-32b模型在规则需求数量较大的情况下转化准确率出现下降,在转化规则需求数量达到500条时,只能正确地生成前86条流表命令. 因此,在面对大规模拓扑时,CtrlSynth无法一次性生成全部的流表命令,需要多次生成.

5　仿真实验

本节通过实验验证RetroP4与CtrlSynth在网络自动化配置中的可行性,实验基于Qwen2.5-72B模型和Mininet构建的SDN网络环境. 本实验采用半自动化流程:首先,手动触发RetroP4生成所需的P4数据平面程序;其次,结合生成的P4程序,利用CtrlSynth生成对应的控制平面配置;最后,将上述程序和配置加载到Mininet的SDN网络中开展实验验证.

实验内容为在支持P4的交换机上实现ECMP负载均衡. 实验分为两个阶段. 首先,使用RetroP4生成实现该功能的P4程序,RetroP4输入为自然语言描述的ECMP功能需求,输出为可执行的P4数据平面代码. 其次,利用CtrlSynth和对应的网络拓扑文件为交换机进行网络配置,CtrlSynth输入包括网络拓扑信息、物理设置、RetroP4生成的P4程序以及自然语言描述的网络策略要求(参见图11). 输出为交换机的流表配置.

网络拓扑如图12所示,其中三个主机(h1,h2,h3)通过六个交换机(s1~s6)可以相互通信,所有主机可通过交换机网络相互通信.

第一步,用户向RetroP4提出数据平面代码功能要求,RetroP4结合用户输入需求和RAG检索结果生成完整的P4程序. 随后该程序经过p4c编译器进行编译验证. 验证通过后,RetroP4将可运行的P4程序输出给CtrlSynth.

第二步,CtrlSynth接收用户用自然语言提出的控制平面配置需求,并将其转化为规范化的中间表示格式. 该格式包括通信的双方主机以及路径约束(必须经过的交换机和必须排除的交换机节点).

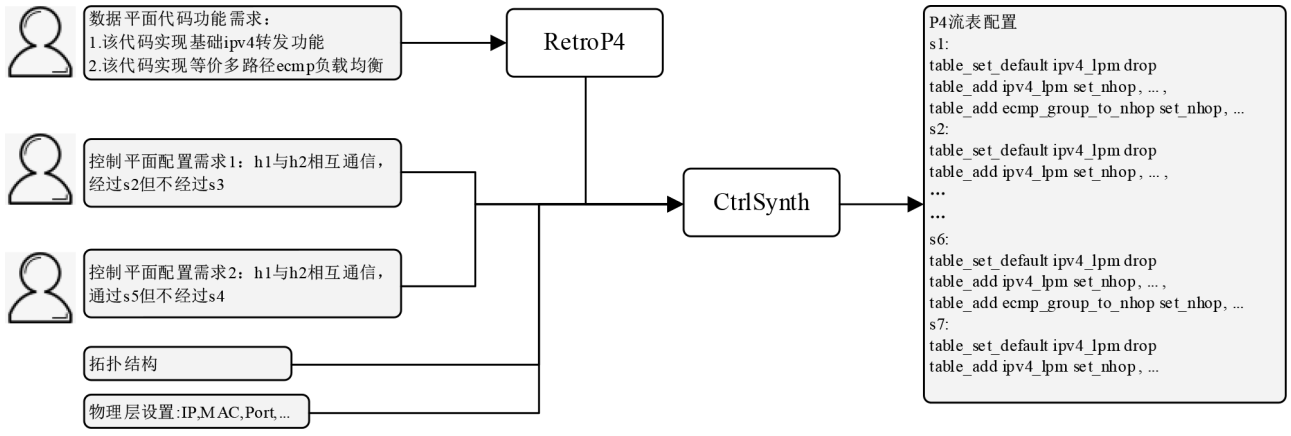


图11　工作流程概要

第三步,CtrlSynth基于上一步生成规范化格式与网络拓扑信息,枚举所有满足约束的可行性转发路径. 在本示例拓扑中,应该生成六个转发路径,其路径如图11所示.

第四步,CtrlSynth针对每台交换机,结合枚举出的所有转发路径,推导出该交换机需要安装的流表规则集合(中间表示).

第五步,CtrlSynth接收网络拓扑、交换机与主机配置(如IP地址、MAC地址、端口号)、RetroP4生成的P4程序以及上一步生成的交换机规则集合,为每台交换机生成具体的P4流表项配置.

在此示例中,CtrlSynth生成P4流表项配置,为主机之间的所有转发路径上的交换机配置控制指令. 例如,最终交换机执行以下任务:s1检查传入的数据包是否发往h2,如果是,则首先随机向s2、s4、s5发送数据包,其次,s2、s4、s5向s6发送数据包,最后,s6将数据包传递给h2(如图12中的蓝色路径);否则,首先随机向s2、s3、s5发送数据包,其次s2、s3、s5向s6发送数据包,最后s6将数据包传递给h3(如图12中的红色路径).

为了评估生成的配置的正确性和可行性,本文在Mininet网络模拟器中构建实验环境. 首先加载RetroP4生成的P4程序与CtrlSynth生成的流表项配置(如s1-

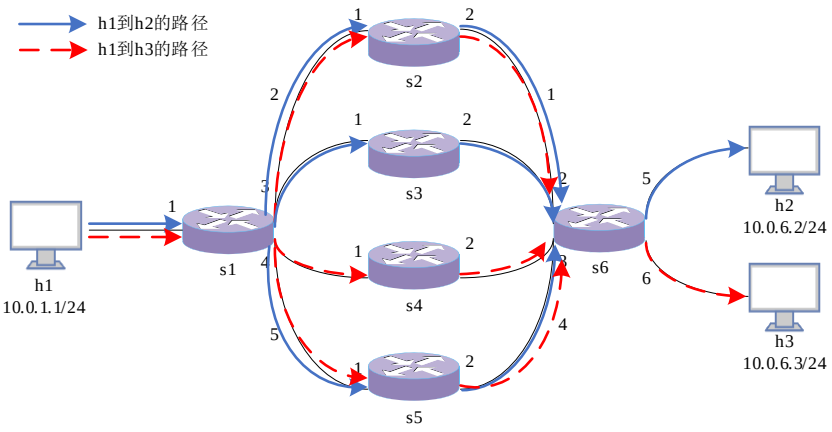


图 12 网络拓扑

command.txt). 其次,通过 p4utils 库启动网络实例,并使用 ping 命令验证基础连通性. 最后,抓取数据包分析转发行为. 实验结果表明:数据包被均匀分配至多条等价路径,成功验证了 ECMP 负载均衡的实现.

6 结论

本文介绍了 RetroP4 和 CtrlSynth,它们分别是用于生成数据平面代码和控制平面配置的特定领域 LLM. RetroP4 基于用户对 P4 代码功能的自然语言描述,自动生成 P4 代码. CtrlSynth 则基于用户对网络高级要求与策略的自然语言描述,自动生成控制平面配置. 本文验证了 RetroP4 生成实现不同功能的 P4 代码的能力,同时也验证了 CtrlSynth 将自然语言网络策略转化为控制平面配置的能力.

现有的自动化配置工具大多基于模板、脚本或图像界面进行配置,需要用户具备一定的网络专业知识. 本文通过 LLM 实现对用户意图的理解和转换,显著降低了配置门槛,使得非专业用户也能快速完成复杂配置任务. 同时,本文的方法提供了统一的自然语言接口,具备良好的可扩展性与人机交互友好性. 然而,当前方法仍存在一些局限性,例如依赖人工步骤,并且在面对超大规模网络时存在效率瓶颈,特别是 CtrlSynth 在批量处理大规模规则时容易出现输出截断问题. 未来,我们计划采用分批次生成的方式来应对这一问题.

由于资源的限制,本文没有对 LLM 本身进行更改. 一些研究表明:微调 LLM 可以获得更好的效果[19,20]. 因此,我们计划对 LLM 进行微调,以探索更好地生成数据平面代码的方法.

目前,本文的配置网络仍未实现自动化,部分操作仍需人工完成. 未来,我们计划将 RetroP4 和 CtrlSynth 两个步骤结合在一起,形成一个完整的 LLM 智能代理,可以根据用户需求同时生成数据平面代码与控制平面配置,实现网络智能化配置.

RetroP4 和 CtrlSynth 均以自然语言作为输入,并且 RetroP4 的输出结果是 CtrlSynth 输入的一部分,这使得它们具备具备良好的接口一致性. 此外,已有研究表明:LLM 在任务编排方面的能力已有显著提升[21],因此可以通过提示工程或智能体框架将两个模块串联执行. 然而,实现全自动化网络配置可能面临几个挑战. 首先,用户对网络的初始意图通常为高级策略描述,需要将其准确分解为数据平面的具体功能需求和控制平面的相应配置需求. 若意图分解出现偏差或遗漏,则将导致两个模块无法有效衔接,进而影响整体流程的正确性. 其次,在自动化流程中,任何一个环节出现问题都需要具备回退、重试或者提示用户的能力. 目前,单个模块具备一定的容错能力,但整合后的错误处理过程尚未设计.

尽管面临上述挑战,本文为基于 LLM 的网络自动配置提供了可行的实现路径与初步验证. RetroP4 与 CtrlSynth 的结合展示了统一自然语言接口简化网络配置的潜力,为构建智能网络管控系统奠定了基础. 未来的工作将围绕意图分解、分批次处理与模型微调等关键问题展开,以实现真正端到端的网络自主配置.

参考文献

[1] KHORSANDROO S, SÁNCHEZ A G, TOSUN A S, et al. Hybrid SDN evolution: A comprehensive survey of the state-of-the-art[J]. Computer Networks, 2021, 192: 107981.

[2] 王朝炜, 杜嘉楠, 王程, 等. 软件定义卫星网络中基于业务调度的混合路由算法[J]. 电子学报, 2024, 52(5): 1506-1515.
WANG C W, DU J N, WANG C, et al. A hybrid routing based on traffic scheduling in double-layer software de-

fined satellite networks[J]. Acta Electronica Sinica, 2024, 52(5): 1506-1515. (in Chinese)

[3] RAMANATHAN S, ZHANG Y, GAWISH M, et al. Practical intent-driven routing configuration synthesis[C]//Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation. California: USENIX Association, 2023: 629-644.

[4] HSU K F, BECKETT R, CHEN A, et al. Contra: A programmable system for performance-aware routin[C]//Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation. California: USENIX Association, 2020: 701-721.

[5] BECKETT R, MAHAJAN R, MILLSTEIN T, et al. Network configuration synthesis with abstract topologies[C]//Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM, 2017: 437-451.

[6] TIAN B C, ZHANG X Y, ZHAI E N, et al. Safely and automatically updating in-network ACL configurations with intent language[C]//Proceedings of the ACM Special Interest Group on Data Communication. New York: ACM, 2019: 214-226.

[7] EL-HASSANY A, TSANKOV P, VANBEVER L, et al. NetComplete: Practical network-wide configuration synthesis with autocompletion[C]//Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation. California: USENIX Association, 2018: 579-594.

[8] KOLIDES A, NAWAZ A, RATHOR A, et al. Artificial intelligence foundation and pre-trained models: Fundamentals, applications, opportunities, and social impacts[J]. Simulation Modelling Practice and Theory, 2023, 126: 102754.

[9] FLORIDI L, CHIRIATTI M. GPT-3: Its nature, scope, limits, and consequences[J]. Minds and Machines, 2020, 30(4): 681-694.

[10] LI J, TAO C Y, LI J, et al. Large language model-aware in-context learning for code generation[J]. ACM Transactions on Software Engineering and Methodology, 2025, 34(7): 1-33.

[11] GAO B, HE Z M, SHARMA P, et al. Cost-efficient large language model serving for multi-turn conversations with CachedAttention[C]//USENIX Annual Technical Conference. California: USENIX Association, 2024: 111-126.

[12] WANG C J, SCAZZARIELLO M, FARSHIN A, et al. Making network configuration human friendly[J/OL]. (2023-09-12)[2025-05-24]. https://arxiv.org/abs/2309.06342.

[13] MONDAL R, TANG A L, BECKETT R, et al. What do LLMs need to synthesize correct router configurations?[C]//Proceedings of the 22nd ACM Workshop on Hot Topics in Networks. New York: ACM, 2023: 189-195.

[14] ZHANG F J, CHEN B, ZHANG Y, et al. RepoCoder: Repository-level code completion through iterative retrieval and generation[C]//Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2023: 2471-2484.

[15] XU Z T, CRUZ M J, GUEVARA M, et al. Retrieval-augmented generation with knowledge graphs for customer service question answering[C]//Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, 2024: 2905-2909.

[16] WEI J, WANG X Z, SCHUURMANS D, et al. Chain-of-thought prompting elicits reasoning in large language models[C]//Proceedings of the 36th International Conference on Neural Information Processing Systems. New York: ACM, 2022: 24824-24837.

[17] BIRKNER R, DRACHSLER-COHEN D, VANBEVER L, et al. Config2Spec: Mining network specifications from network configurations[C]//Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation. California: USENIX Association, 2020: 969-984.

[18] SUBRAMANIAN K, D'ANTONI L, AKELLA A. Genesis: Synthesizing forwarding tables in multi-tenant networks[J]. ACM SIGPLAN Notices, 2017, 52(1): 572-585.

[19] DOERING N, GORLLA C, TUTTLE T, et al. Empirical analysis of efficient fine-tuning methods for large pre-trained language models[EB/OL]. (2024-01-08)[2025-05-21]. https://arXiv.org/abs/2401.04051.

[20] DING N, QIN Y J, YANG G, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models[J]. Nature Machine Intelligence, 2023, 5(3): 220-235.

[21] FAN S D, CONG X, FU Y P, et al. WorkflowLLM: Enhancing workflow orchestration capability of large language models[EB/OL]. (2024-11-08)[2025-05-21]. https://arXiv.org/abs/2411.05451.

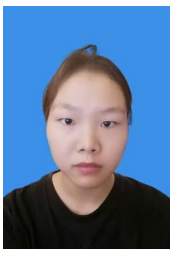
作者简介

杨兴源 男,2002 年 9 月生,江西萍乡人. 现为信息工程大学信息技术研究所先进网络系统与技术研究室硕士研究生. 主要研究方向为可编程网络、匿名通信、大语言模型.

E-mail: yxy32735452@163.com

田 乐 男,1987 年 11 月生,陕西咸阳人. 现为信息工程大学、国家数字交换系统工程技术研究中心副研究员、硕士生导师. 主要研究方向为新型网络体系架构、网络安全.

E-mail: xxgctianle@163.com



姚 莹 女,2001 年 8 月生,河南南阳人. 现为信息工程大学信息技术研究所先进网络系统与技术研究室硕士研究生. 主要研究方向为可编程网络、网络验证、大语言模型.

E-mail: yao.2023@foxmail.com

潘 璠 男,2001 年 8 月生,安徽安庆人. 现为信息工程大学信息技术研究所先进网络系统与技术研究室博士研究生. 主要研究方向为可编程数据平面、零信任、网络智能.

E-mail: panfan2024@126.com

胡宇翔 男,1982 年 11 月生,河南周口人. 现为信息工程大学教授、博士生导师. 主要研究方向为新型网络体系架构、路由与交换、网络系统安全等技术.

E-mail: chxachxa@126.com