ResilM-IBN：一种自愈型多智能体网络管理框架

董黎刚，苟敬文，蒋献，张子天，诸葛斌
（浙江工商大学信息与电子工程学院，浙江 杭州 310018）

摘　要：针对意图驱动网络在多业务环境下意图解析不准确、执行链路不可控及缺乏自愈机制等问题，提出一种面向软件定义网络的自愈型多智能体管理框架ResilM-IBN，以提升网络管理的智能化与鲁棒性。该框架通过结构化通信与多智能体协作机制，实现意图执行过程的语义一致与任务可追溯；构建基于“意图–验证–执行–再验证”链路的闭环执行流程，以增强过程可控性；设计协作式自愈机制，利用“用户意图约束+运行时反馈”双条件触发逻辑实现跨智能体的异常检测与自动修复。基于Mininet与Ryu的原型验证，结果表明，ResilM-IBN能够在复杂场景下实现稳定、可靠的意图执行与自愈修复。所提框架在可行性、稳定性与可迁移性方面表现优异，可为意图驱动网络的工程化实现提供有效参考。

关键词：意图驱动网络；软件定义网络；大语言模型；多智能体
中图分类号：TP393.0
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2026008

ResilM-IBN: multi-agent framework for self-healing network management

Dong Ligang, Gou Jingwen, Jiang Xian, Zhang Zitian, Zhuge Bin
School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Abstract: To address the challenges of inaccurate intent parsing, uncontrollable execution chains, and the lack of self-healing mechanisms in intent-based networking under multi-service environments, ResilM-IBN, a self-healing multi-agent management framework for software-defined network, aiming to enhance network intelligence and robustness, was proposed. Through structured communication and multi-agent collaboration, the framework ensures semantic consistency and task traceability throughout the intent execution process. A serial closed-loop execution process following the “intent-verification-execution-re-verification” workflow was constructed to improve controllability and reliability during intent realization. Furthermore, a collaborative self-healing mechanism was designed, where cross-agent anomaly detection and automated recovery were triggered by a dual-condition logic combining user-intent constraints and runtime feedback. Experimental results obtained from a Mininet-Ryu prototype demonstrate that ResilM-IBN can achieve stable and reliable intent execution and self-healing recovery in complex scenarios. The proposed framework exhibits strong feasibility, stability, and portability, providing a practical reference for the engineering implementation of intent-based networking.

Keywords: intent-based networking, software-defined network, large language model, multi-agent

---

收稿日期：2025–11–27；**修回日期：**2026–01–08
通信作者：蒋献，jiangxian@zjgsu.edu.cn
基金项目：国家自然科学基金资助项目（No.W2421086, No.61871468）；浙江省科技创新重点基金资助项目（No.2023R5211）；浙江省自然科学基金资助项目（No.LZ23F010003）；浙江省重点研发计划基金资助项目（No.2026C02A1244）；桐乡通用人工智能研究院基金资助项目（No.TAGI2-B-2024-0014）；浙江省新型网络标准及应用技术重点实验室基金资助项目（No.2013E10012）
Foundation Items: The National Natural Science Foundation of China (No.W2421086, No.61871468), Key Scientific and Technological Innovation Project of Zhejiang Province (No.2023R5211), The Natural Science Foundation of Zhejiang Province (No.LZ23F010003), Key Research and Development Program of Zhejiang Province (No.2026C02A1244), Tongxiang General Artificial Intelligence Research Institute Project (No.TAGI2-B-2024-0014), Key Laboratory of Emerging Network Standards and Application Technologies of Zhejiang Province (No.2013E10012)

0　引言

随着网络规模和业务复杂度的持续提升，依赖人工配置或静态脚本的传统网络管理方式在多业务、多策略环境下效率低、易出错且缺乏自愈能力，难以满足云数据中心、5G/6G 核心网及工业互联网对自动化与鲁棒性的需求。以 5G 网络切片[1]为例，云数据中心、5G 核心网及工业互联网等场景普遍呈现业务多样、动态性强和运行环境复杂的特征，使人工或半自动化管理难以保证高效性与稳定性。近年来，多篇综述与研究[2-3]从体系结构层面对意图驱动网络的发展现状进行了系统梳理，指出在复杂应用场景下引入更高层次抽象以降低运维复杂度已成为网络管理的重要趋势，但同时也暴露出执行可控性与运行时保障不足等共性问题。

意图驱动网络（intent-based networking，IBN）通过将高层业务意图自动映射为底层配置，被认为是实现网络智能化与零接触运维的重要途径。它能够以“用户意图”为中心驱动网络状态调整，从而显著减少人工干预。在此背景下，IBN 已被逐步引入云原生网络、5G 核心网与边缘计算等复杂场景[4]，用于支撑跨资源、跨域的自动化管理与编排。然而，现有 IBN 系统仍存在多方面限制：一是意图解析准确性不足，自然语言的歧义导致配置不一致；二是执行链可控性差，解析结果与控制层间缺乏可验证反馈；三是缺少自愈机制，运行时异常仍依赖人工修复。这些问题使 IBN 在复杂动态环境中难以实现闭环自动化管理。

早期 IBN 研究主要依赖模板化语言或形式化规则实现意图解析。例如，NILE[5]和 SNIL 通过“主体–动作–对象”三元组结构提升了解析可解释性，SyNET[6]和 Jinjing[7]利用 SMT 求解器保障配置逻辑正确性，但这类方法依赖人工设计规则，难以支持跨域协同且缺乏自然语言适配能力。随着生成式人工智能的发展，研究者开始借助大语言模型（large language model，LLM）来增强意图理解与配置生成的灵活性，形成了更自然的人机交互方式。在此类方向上，已有研究[8]开始尝试将 LLM 能力用于多域基础设施的意图驱动管理与编排，并通过少样本学习降低跨域迁移门槛；也有工作[9]围绕“意图–策略生成”链路，探讨如何利用 LLM 自动生成可执行的管理策略以支持应用级的意图驱动运维。代表性工作如 VPP[10]与 LLM-NetCFG[11]通过验证反馈修正配置错误，实现了较高的解析精度；NetBuddy[12]与 Emergence[13]采用分层策略树与有限状态机结构，对复杂意图进行分解与修复；GIA 框架[14]进一步结合知识增强提示与意图冲突检测机制，在灵活性和泛化性上取得突破。此外，近期研究开始关注小模型或受限模型在 IBN 场景下的可落地性，以缓解通用大模型在时延和资源消耗方面的限制[15]。然而，这些研究多停留在“意图到配置”的单向映射阶段，尚未形成可验证的闭环执行链，也缺乏运行时的异常检测与修复能力。相关研究表明，即便引入意图保障或意图漂移检测机制，执行失败后的恢复过程仍缺乏统一建模与系统化支撑[16-17]。

为解决执行链可控性与鲁棒性不足的问题，近期研究开始探索系统化与多智能体协同框架[18-21]。KlonetAI[20]采用单智能体范式，将自然语言意图直接转化为虚拟网络编排指令，虽然降低了用户门槛，但在扩展性与闭环验证方面存在局限。MNC 框架[21]首次引入多智能体协作思想，将意图处理划分为需求分析、配置生成与质量优化等环节，并结合链式思维（chain of thought，CoT）与多智能体辩论机制提升结果一致性。与此同时，部分研究[22-23]尝试以分角色或 Agentic-AI 的方式组织 IBN 功能模块，通过多智能体协作与结构化通信机制提升整体任务分解与协同决策能力，其中以 MetaGPT[24]为代表的工作展示了 LLM 驱动的多角色智能体协作范式。然而，这些系统普遍仍局限于配置生成层，缺乏与实际控制器的联动与运行时闭环验证，无法实现端到端的可控自愈网络控制。

综合来看，现有 IBN 框架大多聚焦于语义解析或策略生成环节，缺少贯穿“解析–执行–验证–修复”的系统性设计。尽管已有研究分别在意图生命周期管理、执行保障或多智能体协作等方面取得进展[16,22,25]，但相关能力往往以模块化形式分散存在，尚未形成面向实际网络运行环境的统一自愈管理框架。在动态环境中，一旦策略偏离预期，往往需要人工介入才能恢复，阻碍了 IBN 从研究走向工程化应用。

针对上述问题，本文提出一种面向软件定义网络（software defined network，SDN）[26]的自愈型多智能体意图驱动网络（resilient multi-agent intent-

based networking，ResilM-IBN）管理框架。该框架以结构化通信、闭环执行与协作式自愈为核心思想，旨在实现从意图解析到执行验证的全过程自治与可控。通过多智能体协同与统一语义接口，ResilM-IBN能够在动态网络环境中保持语义一致、执行可追溯，并具备运行时异常检测与自动修复能力，从而显著提升系统的鲁棒性与可维护性。

本文的主要工作如下。

1) 提出ResilM-IBN框架：构建一种面向SDN管理的多智能体意图驱动网络框架，首次从意图执行可验证性视角，将意图执行过程形式化为“意图–解析–验证–执行–再验证”的可验证闭环链路，实现从静态映射到动态自治的范式转变，从体系结构层面提升了IBN的可控性与语义一致性。

2) 建立统一语义空间下的多智能体协作机制：首次将多智能体协作领域的消息池理念与结构化通信思想引入IBN，实现跨智能体的语义一致协同与任务链可追溯执行，为意图生命周期管理提供标准化通信基础。

3) 提出一种意图约束的事件触发式自愈策略：设计“用户意图约束+运行时验证反馈”联合驱动的多智能体自愈机制，实现跨智能体的自动检测、修复与历史意图回放，提升系统的可维护性与鲁棒性。

1 框架设计原则与分层结构

为构建具有统一语义、可追溯与可控自愈特性的体系，ResilM-IBN在设计过程中遵循4类约束原则（独立性、一致性、可验证性与鲁棒性），并据此形成分层架构与配套实现机制。具体而言，本节首先介绍框架的设计约束与总体架构，阐明其核心理念与目标属性；随后给出系统的3层架构设计及各层功能定位，为下一节关键机制的实现提供基础。

1.1 设计约束

复杂动态的网络环境要求IBN架构同时具备可扩展性、可追溯性和鲁棒性。然而，传统单体式控制框架在多模块协同时往往难以满足这些需求，常表现为功能耦合度高、通信方式缺乏统一以及执行过程缺乏验证，难以适应快速演化的网络需求。为确保后续框架设计具备明确的结构边界与可验证的运行属性，本文在提出ResilM-IBN之前，首先定义以下4类形式化约束。

设系统由多个功能模块（智能体）组成，其集合表示为

$$\mathcal{A}=\{a_1,a_2,\cdots,a_N\} \tag{1}$$

其中，每个a_i表示一个具备独立功能的智能体。系统中所有跨模块交互均通过结构化消息完成，定义统一的消息集合为

$$\mathcal{M}=\{m_1,m_2,\cdots,m_K\} \tag{2}$$

任一消息$m\in\mathcal{M}$均遵循统一的结构化格式，用于描述操作类型、参数集合以及任务标识等信息。系统执行结果的集合记为

$$\mathcal{R}=\{r_1,r_2,\cdots,r_L\} \tag{3}$$

用于表示意图执行后的观测反馈。

1) 独立性约束：独立性约束用于保证系统中各智能体之间的功能解耦，避免模块间形成紧耦合依赖。形式化地，任意两个不同的智能体$a_i,a_j\in\mathcal{A}(i\neq j)$，其状态不允许直接共享，二者之间的交互仅能通过统一消息集合$\mathcal{M}$完成。记智能体之间的直接依赖关系为$D(a_i,a_j)$，则独立性约束可表示为

$$D(a_i,a_j)=\varnothing,\quad \forall i\neq j \tag{4}$$

2) 一致性约束：一致性约束用于保证不同智能体对任务参数和操作语义的解释保持一致。为此，系统定义统一的结构化消息模式$\mathcal{S}$，所有消息$m\in\mathcal{M}$均需满足$m\in\mathcal{S}$。任一智能体a_i对消息m的解析函数记为$\phi_i(m)$，一致性约束要求对于任意消息m，不同智能体对其关键字段的语义解释结果保持一致，即

$$\phi_i(m)=\phi_j(m),\quad \forall a_i,a_j\in\mathcal{A} \tag{5}$$

该约束从接口层面消除了语义歧义，为多智能体协作提供统一的语义空间基础。

3) 可验证性约束：可验证性约束用于确保意图执行过程具备可观测与可判定能力。系统中定义验证函数为

$$V:\mathcal{M}\times\mathcal{R}\to\{0,1\} \tag{6}$$

其中，$V(m,r)=1$表示执行结果r满足消息m所对应的预期条件，反之表示验证失败。对于任一被执行的结构化消息$m\in\mathcal{M}$，系统必须能够获得对应的执行反馈$r\in\mathcal{R}$，并通过验证函数判断其执行状态。该约束保证了意图从解析到执行的全过程可验证性，是构建闭环执行机制的必要前提。

4) 鲁棒性约束：鲁棒性约束用于描述系统在执行异常或环境变化情况下的恢复能力。设系统运行过程中可能出现的异常状态集合为$\mathcal{E}$，当执行结果满足相应异常条件时，系统应允许触发后续的恢复或修复操作。为此，系统需具备一种由异常状态生成修复消息的触发机制（如$\mathcal{E} \rightarrow \mathcal{M}$），用于引导系统进入自愈流程。该约束并不限定具体修复策略的实现方式，而是从机制层面规定系统在异常条件下能够安全地进入自愈流程，其具体检测指标与触发逻辑将在2.3节中进一步说明。

通过上述约束，ResilM-IBN在体系结构层面明确了多智能体协作的边界条件与运行规范。这些约束共同构成后续统一消息机制、闭环执行机制以及协作式自愈策略得以成立的前提。

1.2　框架分层结构与功能划分

在前述设计约束的指导下，本文构建的ResilM-IBN系统总体架构如图1所示。该架构以模块解耦、结构化通信与闭环执行为核心设计理念，可在复杂动态网络环境下实现高效管理与持续优化。

架构整体采用分层协作式设计，由上至下依次包括用户交互层、智能体协作层与执行控制层。用户交互层负责接收自然语言意图输入并提供可视化反馈；智能体协作层承担意图解析、任务分解、策略生成及异常处理等核心功能；执行控制层面向底层SDN控制器和网络基础设施，负责流表下发、拓扑维护与状态验证。各层通过共享的结构化消息池进行信息交互，实现跨模块的语义一致与任务协同，从而构成一个可追溯、可验证、可自愈的意图驱动网络管理闭环。

通过上述设计约束与分层架构，ResilM-IBN在体系层面缓解了传统IBN中的高耦合、通信不统一和缺乏验证等问题。这些约束条件不仅支撑了本文框架的运行逻辑，也为其他意图驱动网络架构的设计提供了可借鉴的思路。还为后续的统一消息机制、闭环执行机制及协作式自愈奠定了统一的运行基础。

1.3　多智能体角色划分与协作流程概述

为避免单体智能体在复杂意图场景下面临职责耦合、状态膨胀及验证困难等问题，ResilM-IBN从意图生命周期视角对网络管理功能进行解耦，将意图解析、执行控制、运行验证与异常修复等关键能力划分为若干职责单一、状态独立的功能智能体，从而构建一种面向意图驱动网络管理的多智能体协作框架。

在该框架中，多智能体并不是多个同构模型实例的简单并行，而是在统一语义空间下，由多个功能异构、职责解耦的智能体通过结构化消息进行协作的体系结构设计。具体而言，框架主要包括以下几类功能智能体。

1) 意图解析智能体，负责将用户输入的自然语言意图解析为具有顺序关系的计划级任务链。

2) 指令生成智能体，依据统一的语义模板将任务链的每一个节点渲染为结构化JSON指令。

3) 执行类智能体，负责与底层SDN控制器交

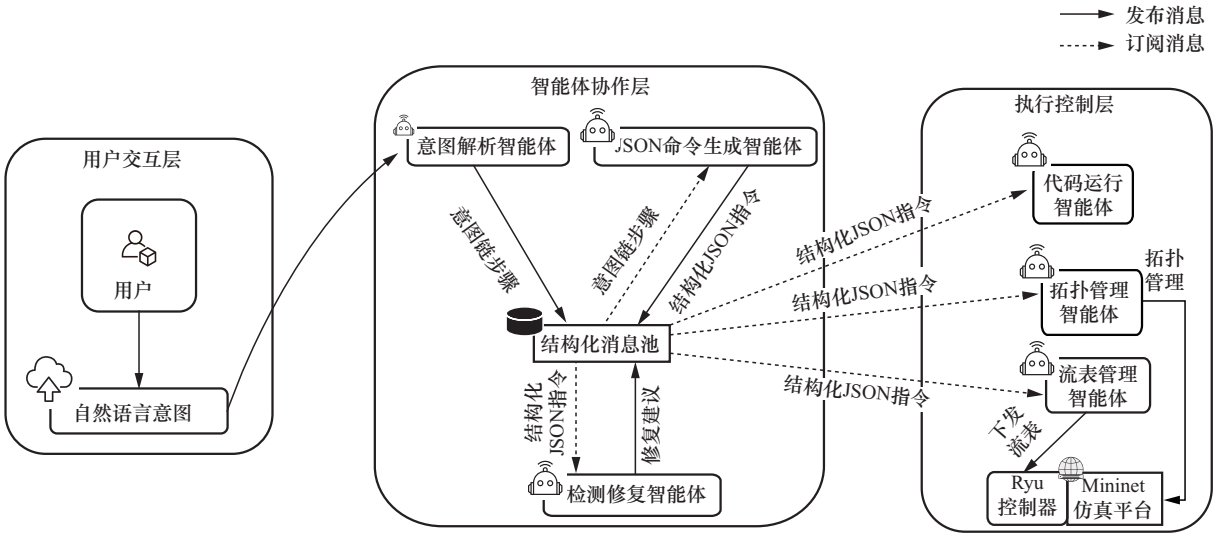

图1　ResilM-IBN系统总体架构

互或执行系统级指令，完成具体网络操作。

4) 验证与自愈类智能体，用于对执行结果进行状态验证，并在满足用户意图授权条件时协调异常修复流程。

各智能体之间不直接共享内部状态，而是通过共享的消息池进行解耦通信。所有交互消息均遵循统一的结构化格式，并通过唯一的trace_id标识进行关联，从而在多智能体协作过程中保持语义一致性与任务可追溯性。该设计使不同智能体能够在不引入紧耦合依赖的前提下协同工作，并支持功能模块的独立演化与替换。

在运行流程上，用户提交的自然语言意图首先由意图解析智能体转化为计划级任务链并发布至消息池；随后，相关任务经由验证智能体进行合法性与一致性检查后，被执行类智能体依次订阅并执行；执行结果再次进入验证流程，用于判断网络状态是否满足用户意图中声明的期望条件。当检测到执行异常且意图中显式授权修复操作时，自愈类智能体被触发并协调相应的修复行为。通过上述基于消息驱动的协作流程，ResilM-IBN在保持控制逻辑解耦的同时，实现了意图执行、验证与自愈恢复的闭环运行。

2 关键机制

为克服现有IBN系统在语义一致性不足、执行过程不可控及缺乏自愈能力等方面的局限，本文在1.3节所提出的系统架构基础上设计3项关键机制，以实现意图解析、任务执行与异常修复的闭环协同。具体而言：结构化通信与任务协作机制用于解决多智能体之间语义不一致与交互耦合问题，实现统一语义空间下的任务协同与可追溯通信；串行化闭环执行机制旨在保障意图从解析到执行的全过程可验证与可调节，通过“意图–验证–执行–验证”流程实现链路级控制；协作式自愈机制针对运行时异常与执行偏差，引入“用户意图约束+运行时反馈”的双条件触发逻辑，实现跨智能体的自动检测、修复与回放。三者共同构成ResilM-IBN的核心运行框架，为IBN系统提供了语义一致、可控且具备自愈能力的管理基础。

2.1 统一消息机制与意图解析

在IBN框架中，语义一致性与可追溯性是保障跨模块协作可靠性与系统可控性的关键性质。Kou等[27]将语义一致性定义为用户意图与底层配置之间保持语义对齐的能力，并通过建立端点组集合与意图集合之间的双射关系来验证解析过程中的一致性，从而避免意图语义漂移与策略歧义问题。本文在此基础上将该概念扩展至多智能体通信层面：通过统一结构化消息格式与共享消息池机制，使不同智能体能够在同一语义空间下解释任务参数与动作语义，从而维持系统全局语义一致性。与此同时，可追溯性通常用于描述系统对操作与数据流转的全程记录与回溯能力。Risdianto等[28]指出，在网络可见性解决方案中，通过在各处理阶段持续存储生成的数据与决策，可确保系统操作的可追溯性。借鉴这一思想，本文将可追溯性概念扩展至多智能体意图执行场景：通过为每条结构化消息分配唯一标识符与时间戳，实现跨模块交互的因果链追踪与历史回溯，从而支持可验证的闭环执行。

为了形式化描述这一机制，本文将自然语言意图到结构化消息的生成过程抽象为如下映射函数，即

$$M{:}I \rightarrow (a,P,t,\tau) \tag{7}$$

其中，M表示自然语言意图到结构化消息的映射函数；I表示用户输入的自然语言意图，输出为符合统一规范的结构化消息；a表示动作类别（action）；P表示参数集合（parameters）；t表示唯一任务标识符（trace_id）；τ表示时间戳（timestamp）。所有消息均遵循这一映射关系，从而在语义上保持一致性，并确保任务链路的可追溯性。

基于这一抽象模型，ResilM-IBN采用JSON格式作为统一结构化消息载体，并结合发布/订阅模式实现消息的分发与接收。

此外，由于非网络管理专业用户的自然语言意图难免存在表述上的不确定性，在生成最终配置命令之前，需要先将其转化为标准化的任务链，以便进入后续执行流程，从而避免直接映射配置可能导致的歧义与冲突。ResilM-IBN首先将高层需求解析为由多个任务节点组成的结构化任务链，为了形式化描述自然语言意图的任务链分解过程，定义任务分解函数$D(\cdot)$将意图I映射为任务集合T，其中包含步骤数组plan_steps；随后，逐步渲染为可执行级消息序列M。

$$T = D(I) = (\text{intent},\text{plan\_steps}) = \{[s_1,\cdots,s_n]\} \tag{8}$$

其中，T为输出计划级消息，plan_steps为长度为n的自然语言步骤数组。与传统的直接意图到配置的映射相比，任务链能够明确显示任务之间的顺序依赖，避免了直接映射时可能出现的执行冲突；同时为每个任务节点划定明确的语义边界，便于后续进行参数校验、异常检测与结果回溯。

随后，再通过$R(\cdot)$渲染函数将任务链中的每一个任务节点逐一渲染为符合统一规范的结构化消息，即

$$M = R(T) = [M_1,\cdots,M_n] \tag{9}$$

为体现LLM不直接掌握网络拓扑、由后台补全上下文的工程现实，本文将$R(\cdot)$明确为两阶段映射。对任一任务节点M_i有

$$M_i = \underbrace{\text{Complete}(M_i^*,K)}_{\text{backend completion}} \quad \text{with} \quad M_i^* = \underbrace{R_{\text{LLM}}(o_i,\Theta_i;\Sigma(o_i))}_{\text{LLM renders partial JSON}} \tag{10}$$

其中，任务节点三元组(o_i,Θ_i)分别表示操作类型（如ping_test、limit_bandwidth）以及参数集合（如主机、具体数值）；$\Sigma(o_i)$为动作o_i的统一消息格式模板（规定所需字段/可选项/默认值）；R_{LLM}依据模板仅生成可由意图直接确定的字段（如主机名、期望结果、动作类别），得到部分JSON数据M_i^*；后台函数$\text{Complete}(\cdot,K)$使用系统知识K（如拓扑、主机表、别名、单位换算）解析并填充占位符/缺失项（如将自然语言中的主机h_2解析为10.0.0.2），产出可执行消息M_i。为简洁起见，后台补全可用一行式规则刻画，即

$$\text{Complete}(M^*,K):\forall k \in \text{keys}(M^*), v_k \leftarrow \begin{cases} M^*[k], & \text{已给定} \\ \text{resolve}_k(K), & \text{待补全占位符} \\ \text{default}_k, & \text{缺乏上下文} \end{cases} \tag{11}$$

其中，k表示结构化消息中的字段键，即待补全的具体配置项；resolve_k表示针对该键的详细配置信息解析函数；default_k为模板中该键对应的缺省策略。通过统一消息机制与任务链解析，ResilM-IBN在体系层面满足了一致性约束（保证语义统一）和可验证性约束（支持执行跟踪与回溯）。该机制不仅提升了系统在跨模块通信和任务管理上的可扩展性，也为后续的闭环执行和协作式自愈提供了统一的数据基础。

上述统一消息机制从结构层面保证了意图在系统内的一致表示，但仍需进一步说明意图解析过程在实际生成步骤序列时所遵循的约束条件。为此，下文将对意图解析模块的具体建模方式与设计取舍进行说明。具体而言，本文将自然语言意图到结构化消息的生成过程抽象为统一语义空间下的映射关系。该映射并非自由生成，而是在受限动作空间与解析规则约束下完成的，以确保意图解析结果的确定性与执行安全性。

本文的意图解析模块面向单一用户意图实例进行设计，其核心目标是将输入意图映射为具有明确顺序关系的计划级步骤序列，而非直接生成可执行的底层网络配置。形式化地，意图解析的输出表示为$\langle s_1,s_2,\cdots,s_n\rangle$，其中每个步骤$s_i$描述一项原子化网络管理操作，并隐含其在整体意图中的执行顺序依赖关系，为后续动作级解析与闭环执行提供了中间抽象层，从而避免解析误差直接传播至执行平面。

为约束解析结果的语义范围，本文定义有限动作集合为

$$\mathcal{O}=\{\text{create\_topology,install\_flowtable,ping\_test,wait,}\cdots\} \tag{12}$$

意图解析模块要求每个步骤s_i在后续解析过程中仅能映射至上述动作集合中的单一元素，从而将解析结果限定在系统当前支持的操作语义内。

本文以执行可控性与系统安全性为首要目标。基于网络管理场景中配置错误可能引发的连锁影响，选择通过系统层的规则化约束对解析结果进行显式收敛，而非依赖对LLM进行参数级领域微调。该设计将领域知识与安全策略外置于系统逻辑中，使解析模块在不同LLM实例之间具备良好的可迁移性，同时降低了模型输出不确定性对执行平面的影响。

框架按照步骤序列依次推进执行流程，确保每一步操作在完成并获得反馈后才进入下一步。当某一步骤的解析结果无法映射至系统当前支持的动作集合，或违反对应动作的字段约束时，该步骤将被标记为不可执行状态（如action = false），并作为结构化消息发布至消息池。由于系统未为该类动作注册任何执行订阅者，该消息不会触发后续执行逻辑，从而保证网络状态不受影响。该不可执行安全约束策略为意图解析过程提供了确定的系统行为，使系统在面对不支持或不明确操作时保持保守且可预测的响应。

2.2 控制解耦与闭环执行机制

在网络与服务管理领域，可控性通常用于描述用户或系统对网络行为进行期望设定与动态调控的能力。Meijer等[29]指出，可控性体现为用户能够明确表达其对网络和数据处理方式的期望，并通过策略制定或路径控制等机制影响网络行为，从而增强系统的透明度与自治性。

为同时提升系统的可控性与可维护性，ResilM-IBN在体系结构中引入控制与执行解耦机制和闭环执行机制。前者通过统一的结构化消息接口分离控制逻辑与底层操作，使各智能体在共享消息池中独立运行、协同决策，从而提升系统的独立性与可演化性；后者在执行链路中增加状态验证与反馈环节，实现意图解析、执行与验证的闭环过程，确保执行结果的可验证性与稳定性。

这种解耦与闭环设计直接支撑了系统的可维护性。根据系统质量模型ISO/IEC 25010的定义，可维护性指系统在其运行环境中被定位、修复与调整的难易程度[30]。在本文场景中，可维护性主要体现如下。

1) 体系结构的模块化与解耦性，使控制逻辑与执行逻辑可独立更新。

2) 接口与消息的结构化定义，减少人工干预与修改错误。

3) 基于反馈的局部修复机制，不需要重启即可恢复功能。

由此，控制解耦与闭环执行机制共同构成了ResilM-IBN的可维护性基础，使系统在多业务、多意图场景下保持长期稳定运行。

2.2.1 控制与执行解耦的形式化描述

控制/执行解耦的核心思想是通过消息池引入统一的交互接口，使控制平面和执行平面之间不再直接依赖。控制平面仅负责意图解析与决策生成，而执行平面专注于具体网络操作的落实，二者通过结构化消息进行间接交互。

在体系结构层面，该解耦关系可抽象为控制平面与执行平面之间的接口映射，即

$$\mathcal{M} \xrightarrow{\mathcal{G}} \mathcal{R} \tag{13}$$

其中，$\mathcal{M}$表示统一消息集合，$\mathcal{R}$表示执行结果集合。控制平面依据式(8)所定义的语义约束，将用户意图解析为结构化消息并发布至共享消息池；执行平面通过订阅相应操作类型，调用映射函数$\mathcal{G}$生成对应的执行结果反馈。

上述抽象明确界定了系统的接口边界，使控制逻辑与执行逻辑能够独立演化。当新增或替换智能体时，仅需遵循统一的消息接口规范，而不需要修改系统核心控制流程，从体系结构层面避免了侵入式修改，增强了框架的可扩展性与可维护性。

2.2.2 预执行验证模块

在结构化消息下发至执行平面之前，ResilM-IBN在控制与执行平面之间引入预执行验证模块，用于对生成的消息进行合法性与一致性检查，以避免因意图歧义或LLM幻觉导致的异常配置下发。该过程在逻辑上位于控制映射与执行映射之间，是闭环执行流程的前置操作。

形式化地，定义预执行验证函数为V_{pre}: $\mathcal{M} \to \{0,1\}$，其中，$V_{\text{pre}}(m)=1$表示消息m通过预执行验证，$V_{\text{pre}}(m)=0$表示验证失败。预执行验证规则主要包括但不限于以下几类。

1) 结构合法性验证：检查消息是否符合统一的结构化模式，关键字段是否完整。

2) 参数合法性验证：验证参数类型与取值范围是否满足预定义约束。

3) 上下文一致性验证：结合当前网络拓扑与资源状态，检查消息中引用的对象是否存在且可达。

4) 意图约束一致性验证：判断生成的操作是否违反用户意图中显式声明的限制条件。

为了具体说明上述验证机制的工程实现逻辑，以上下文一致性验证为例，其核心在于检查意图操作对象在当前网络拓扑中的存在性与状态可用性。定义当前网络拓扑状态为图$G=(V,E)$，其中，V为网络节点集合，E为链路集合。对于一条待下发的结构化消息m，设其操作的目标对象集合为O_m（如源主机m.src、目的主机m.dst或目标交换机m.sw）。上下文一致性验证函数$V_{\text{ctx}}(m)$的判别逻辑为

$$V_{\text{ctx}}(m)=\begin{cases}1, & \forall o \in O_m, o \in V \wedge \text{State}(o)=\text{ACTIVE}\\ 0, & \text{其他}\end{cases} \tag{14}$$

其中，$\text{State}(o)$表示从SDN控制器（如Ryu）获取的节点实时状态。该模块通过调用北向接口获取实时网络拓扑数据；若$V_{\text{ctx}}(m)=0$，系统将抛出异常并阻断消息下发，从而防止向不存在或离线的设备

下发空指针指令，确保执行链的安全性。

当且仅当消息通过预执行验证时，系统才允许其进入执行阶段，即

$$V_{\text{pre}}(m) = 1 \Rightarrow \text{Dispatch}(m) \tag{15}$$

当验证失败时，系统将阻断该消息的下发过程，并将验证结果反馈至控制平面，以触发参数修正或重新规划。

需要说明的是，本文主要关注意图执行闭环的整体建模与关键模块实现，其中预执行验证模块作为独立阶段被明确定义，其具体实现留作后续工作。

2.2.3　验证增强的闭环执行机制

为提升意图执行过程的可观测性与可追溯性，ResilM-IBN 在控制与执行流程中引入一种验证增强的闭环执行机制。该机制的核心思想是通过在执行链路中显式引入验证与反馈步骤，使意图从解析到执行的全过程形成信息层面的闭合回路，从而避免“仅下发、不反馈”的一次性操作模式。

在具体实现中，当用户意图涉及状态修改或策略更新等操作时，控制平面在生成结构化消息序列的过程中，自动在相应修改操作之后补全一条验证或观测步骤，用于获取执行后的网络状态或运行结果，并将反馈信息返回给用户呈现。该过程通过提示词约束引导 LLM 在任务规划阶段生成包含“执行–验证”关系的消息序列，从流程层面保证关键操作具备可观测的执行结果。

形式化地，设一条结构化消息 $m \in \mathcal{M}$ 被下发执行后产生相应的观测结果 $r \in \mathcal{R}$，该过程可表示为 $r = \Gamma(m)$，其中 $\Gamma(\cdot)$ 表示系统的反馈获取过程，通过测试、状态查询或运行反馈等方式获得并返回执行结果。

对于包含修改操作的消息 m^{chg}，控制平面在任务链中追加一条验证消息 m^{ver}，构成最小的验证增强执行对为

$$m^{\text{chg}} \rightarrow m^{\text{ver}}, \quad m^{\text{ver}} \in \mathcal{M} \tag{16}$$

其中，m^{ver} 用于触发对修改操作结果的观测，其输出结果主要用于支撑用户对执行效果的确认与分析，而不直接参与系统内部的自动判定或纠偏决策。

结合预执行验证模块（见 2.2.2 节），上述设计使用户意图被解析为结构化消息后，首先经过合理性与一致性检查；随后在意图执行完成后，通过追加的验证步骤获取对应的运行反馈，并将结果返回给用户呈现。由此，本文在体系结构与执行流程层面对意图执行过程进行了统一刻画，将其形式化为一条“意图–解析–验证–执行–再验证”的逻辑闭环执行链路。该闭环执行机制侧重于流程与信息层面的闭合，其目标在于增强意图执行过程的可观测性与可追溯性，而非实现基于验证结果的自动决策或纠偏控制。相关能力可在此机制基础上作为进一步扩展方向展开。

对于由复杂意图分解得到的结构化消息序列 $\{m_1, m_2, \cdots, m_L\}$，ResilM-IBN 采用消息驱动的事件推进方式组织执行顺序：系统以消息的发布、执行反馈与验证结果作为推进事件，在同一意图链路内按照“上一步完成→触发下一步”的规则逐步推进。具体而言，当第 i 步消息 m_i 被执行后，系统获取其反馈 $r_i = g(m_i)$，并在需要时触发相应的验证/观测步骤；仅当该步骤的反馈结果返回并被记录后，才允许发布或执行下一步消息 m_{i+1}。因此，意图执行顺序可抽象为如下的事件门控关系，即

$$\text{Enable}(m_{i+1}) \Leftarrow \text{Observed}(m_i), \quad i = 1, \cdots, L-1 \tag{17}$$

其中，$\text{Observed}(m_i)$ 表示与 m_i 相关的执行反馈（及必要的验证观测）已经获得并回传。通过这种以事件为触发条件的顺序推进机制，系统在异步环境中不需要依赖线程锁也能保持同一意图链路的顺序一致性。ResilM-IBN 的闭环执行时序如图 2 所示。

2.3　协作式自愈与鲁棒性设计

在系统工程中，鲁棒性通常被定义为系统在存在扰动或不确定性条件下仍能维持预期性能的能力。传统鲁棒设计主要依赖冗余与过度配置以降低敏感性，而近年来的研究进一步指出，鲁棒性应与弹性区分：前者强调抗扰动能力，后者强调在受扰后恢复与适应的能力[31]。本文采用扩展意义下的鲁棒性定义，即鲁棒性不仅体现为对异常的抵抗能力，还包括通过自愈过程实现性能恢复与意图重建的能力。

基于这一定义，ResilM-IBN 的协作式自愈机制在运行层面体现恢复性鲁棒性。系统通过运行时验证与反馈检测执行偏差，并在检测到异常后触发跨智能体的协作修复流程。不同于传统被动修复模式，本文意图约束的事件触发式自愈策略仅在用户

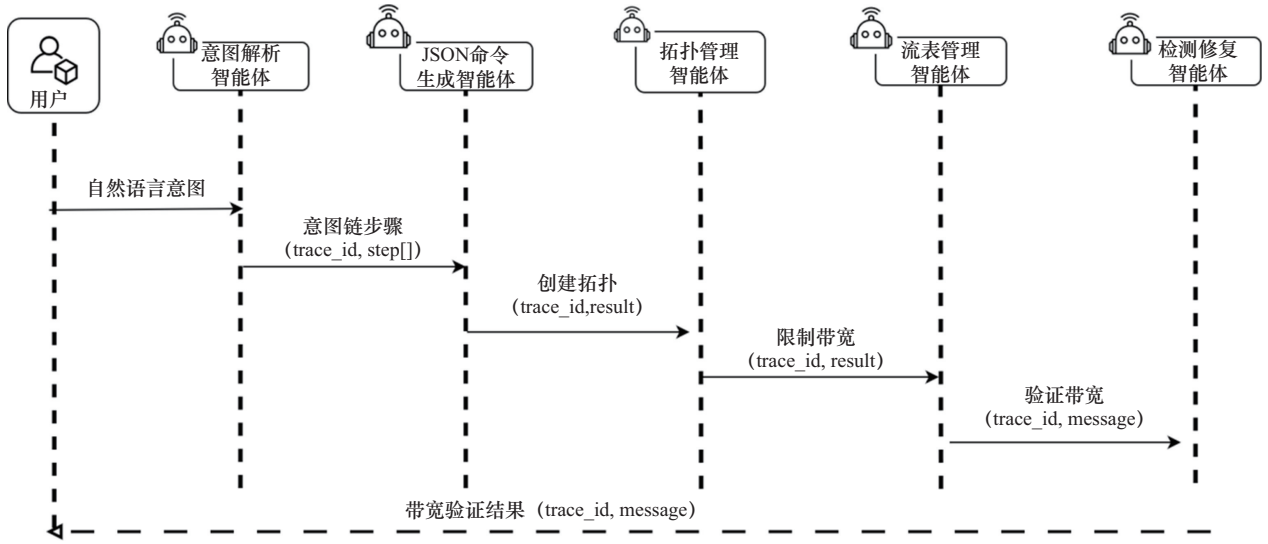

图2 ResilM-IBN的闭环执行时序

意图中显式授权修复行为时执行自愈链路，从而在自动化效率与用户可控性之间取得平衡。该机制建立在闭环执行模型之上，将修复过程划分为异常检测、修复决策、执行与验证4个阶段，不同智能体通过结构化消息池协同工作，以确保修复动作的可追溯性与语义一致性。

2.3.1 异常类型与自愈触发条件的形式化描述

主机连通异常自愈算法如算法1所示。为增强自愈机制对不同网络异常场景的适配能力，本文首先对异常类型进行统一抽象。定义网络异常类型集合为

$$\mathcal{E} = \{ e^{\text{conn}}, e^{\text{bw}}, e^{\text{sw}}, e^{\text{cfg}}, \cdots \} \tag{18}$$

其中，e^{conn}表示端到端连通性异常，e^{bw}表示链路带宽或时延异常，e^{sw}表示交换机或节点故障，e^{cfg}表示策略或配置冲突，其他异常类型可按需扩展。在此基础上，定义异常检测函数为

$$d: \mathcal{M} \to \mathcal{E} \cup \{ \varnothing \} \tag{19}$$

其中，$d(m) = e$表示针对结构化消息m的执行过程检测到类型为e的异常，若未检测到异常则返回$\varnothing$。该抽象使算法1中的“探测”操作不再局限于连通性检测，而可统一表示多种异常检测接口。

自愈流程的触发需同时满足“检测到异常”与“用户授权修复”两个条件。为此，定义修复触发判定为

$$\text{Trigger}(m,e) = \begin{cases} 1, & e \neq \varnothing \wedge \text{CheckRepairAuth}(m) = 1 \\ 0, & \text{其他} \end{cases} \tag{20}$$

其中，$\text{CheckRepairAuth}(m)$表示结构化意图中是否允许针对消息$m$执行修复操作。仅当触发条件成立时，系统才会进入协作式自愈流程。

2.3.2 协作式自愈流程与算法说明

在上述触发条件下，ResilM-IBN将自愈过程划分为异常检测、修复决策、执行与再验证4个阶段，不同智能体通过结构化消息池进行协作，以保证修复过程的可追溯性与语义一致性。为便于说明，算法1给出主机连通性异常场景下的协作式自愈流程示例，其他类型异常可在相同框架下通过扩展检测指标与修复策略实现。

其核心触发条件由“异常检测结果”与“用户意图中的修复授权参数”共同决定：仅当异常检测函数返回非空结果且用户意图中允许修复时，系统才会进入修复链路。该策略的引入使ResilM-IBN在面对动态拓扑变化或策略冲突时，能够根据用户意图恢复网络状态并保持整体性能稳定。

算法1 主机连通异常自愈算法

输入 结构化消息m（字段：src,dst, expectresult），超时时间T

输出 修复结果RepairOutcome $\in$ { Success,Fail, Skip }

1) 发送探测请求：$r \leftarrow \text{PING}(m.\text{src}, m.\text{dst})$

2) if r = pass then

3) return Success

4) end if

5) 检查期望条件：allow $\leftarrow \text{CheckRepairAuth}(m)$

6) if allow = false then
7)　　return Skip
8) end if
9) 向消息池发布修复建议：Publish(qa.repair_suggest, { trace_id,*m*.src,*m*.dst })
10)等待修复完成信号：ack ← Await(flow.repairdone,trace_id,*T*)
11) if ack = timeout then
12)　　return Fail
13) end if
14) 再次执行检测：r' ← PING(*m*.src,*m*.dst)
15) if r' = pass then
16)　　return Success
17) else
18)　　return Fail
19) end if

为便于理解，算法 1 中的关键字定义如下：PING(src,dst) 用于向目标主机对发送探测报文，以验证端到端连通性；CheckRepairAuth(*m*)（对应式(20)及意图修复授权参数）用于检查结构化消息 *m* 中是否包含修复授权或期望结果字段，以判断该任务是否允许进入自愈流程；Publish(topic, message) 表示向消息池发布修复建议，供其他智能体订阅并执行修复动作；Await(event,trace_id,*T*) 用于在超时时间 *T* 内监听指定事件（如flow.repairdone）的反馈信号，若超时未响应则判定修复失败；RepairOutcome 为修复结果的枚举集合，取值包括（Success,Fail,Skip），分别对应修复成功、失败或跳过。上述关键字共同构成 ResilM-IBN 自愈机制的模块化接口，实现跨智能体的协作式检测与修复流程。需要说明的是，这些关键字描述的是自愈流程中的逻辑接口，其具体检测方式与修复动作可根据异常类型进行扩展与替换，而不影响整体协作流程结构。

在上述算法中，自愈流程是否被触发由用户意图中的修复授权参数显式控制。用户在意图层面通过包含“异常发生时允许修复”的语义约束，指示系统在检测到执行偏差后是否可以进入协作式自愈流程。该语义约束在意图解析阶段被映射为结构化消息中的授权字段，例如期望执行结果标志（如 expect_result）与修复建议标识（如 auto_fix）。当系统检测到异常且授权字段表明允许修复时，算法 1 所示的自愈流程才会被触发；否则，系统仅记录异常并反馈执行结果，而不执行自动修复操作。

通过上述意图约束的事件触发机制，ResilM-IBN 在避免越权修复风险的同时，实现了与用户意图一致的高效恢复。以用户意图“若 h_1 与 h_2 无法通信则修复”为例，系统仅在检测到连通性异常且意图明确授权的情况下执行修复操作。该流程验证了意图约束自愈机制的可行性，并为不同异常场景下的协作式修复提供了可复用的流程模板。

2.3.3　自愈机制的通用性与扩展性讨论

尽管算法 1 以连通性异常为示例进行说明，但其设计并不局限于单一异常类型。通过前述异常集合 $\mathcal{E}$ 与检测函数 $d(\cdot)$ 的抽象，不同类型异常均可在统一自愈流程下进行处理。异常类型、检测指标、修复策略的对应关系如表 1 所示，总结了协作式自愈机制在不同异常场景下可采用的检测指标与修复策略示例，用于说明框架的适配性与扩展能力。本文原型系统以主机连通性异常为代表场景进行实现与验证，其他异常类型可在保持整体流程结构不变的前提下按需扩展。

表 1　异常类型、检测指标、修复策略的对应关系

异常类型	检测指标示例	修复策略示例
主机 / 端到端连通性异常	Ping 失败	重新下发流表
链路带宽 / 时延异常	吞吐下降 / 时延超阈	QoS 参数调整 / 路径切换
交换机 / 节点故障	心跳丢失 / 端口失活	节点隔离 / 流表迁移
配置冲突	策略不一致	回滚配置 / 重新部署策略
链路抖动或不稳定	丢包率波动 / 时延方差升高	路径平滑 / 备用链路切换
控制信道异常	控制消息丢失 / 响应超时	控制通道重建 / 主备切换
资源过载（CPU/队列）	队列积压 / 端口利用率过高	负载分担 / 限流调度

在结构化意图中，修复授权通过显式参数进行描述，具体表达形式与触发逻辑已在算法说明中给出。该设计确保算法 1 所示的自愈流程始终受用户意图约束，从而避免未经授权的越权修复行为。

2.3.4　状态回溯与鲁棒性增强

在即时修复机制之外，ResilM-IBN 引入一种基于任务链重放的状态回溯机制，以增强系统在长

期运行过程中的鲁棒性。该机制以成功执行的结构化操作记录作为恢复依据，通过对历史操作序列的可控重放实现网络状态的确定性恢复。

形式化地，记一次意图闭环执行对应的任务链为有序JSON操作序列，即

$$\mathcal{J}=\langle J_1,J_2,\cdots,J_n\rangle \tag{21}$$

其中，J_i表示第i个已成功执行的结构化操作（例如流表安装、带宽限制、链路控制等）。系统在每次任务链执行完成后，将$\mathcal{J}$按时间顺序持久化记录到日志中，形成可重放的执行轨迹。

当系统检测到网络运行异常时，触发回溯过程。为避免在恢复阶段执行冗余或仅用于观测的步骤，系统引入可配置的跳过规则集合Π，并对任务链进行裁剪，得到待重放序列为

$$\mathcal{J}^{\star}=\{J_i\in\mathcal{J}\,|\,\pi(J_i)=1,\pi\in\Pi\} \tag{22}$$

其中，判定函数$\pi(\cdot)\in\{0,1\}$用于表示某一操作在恢复阶段是否需要执行（例如连通性测试类操作可配置为$\pi(J_i)=0$）。随后系统按照原始顺序对$\mathcal{J}^{\star}$进行重放，即

$$\text{Replay}(\mathcal{J}^{\star})=\langle J_{k_1},J_{k_2},\cdots,J_{k_m}\rangle,\quad k_1<k_2<\cdots<k_m \tag{23}$$

该过程以“仅重放曾经成功执行过的操作”作为安全约束，从而避免在故障恢复阶段引入未验证的配置变更，并在机制层面保证恢复行为的确定性与可控性。

从时间尺度看，该任务链重放机制与协作式自愈机制形成互补。协作式自愈主要面向短时运行区间内的局部异常，强调快速检测与即时修复；而任务链重放机制作用于跨多个意图周期的异常场景，通过对历史配置动作的顺序重放提供长期稳定性保障。二者共同构成覆盖不同时间尺度的鲁棒性增强手段，使系统既具备即时响应能力，又能够在复杂故障条件下恢复至一致的运行状态。

综上所述，协作式自愈、状态回溯与扩展机制协同作用，使ResilM-IBN在动态网络环境下兼具即时修复能力、长期鲁棒性与演进灵活性，为多意图、多异常场景下的网络稳定运行提供了体系化支撑。

3 实验结果与分析

本节旨在通过系统化实验验证ResilM-IBN框架的核心机制与性能表现。首先介绍实验环境与测试方法，确保实验具有可复现性。随后围绕4个方面展开分析：常见意图的执行性能、自愈功能的正确性、不同LLM后端对系统性能的影响以及与现有代表性系统的横向对比。通过定量与定性相结合的方式，本节系统地回答前文提出的关键研究问题。

3.1 实验环境与参数设置

为验证本文提出的ResilM-IBN在执行复杂意图时的性能，进行仿真实验。本文实验在一台搭载Intel Xeon Gold 6348H处理器（16核，2.60 GHz）、64 GB内存的服务器上进行，操作系统为Ubuntu 18.04LTS。意图解析模块依赖于外部语言模型提供的语义映射能力；在本文原型中采用了DeepSeek-V3作为实例验证，该选择体现了本方法对强语义解析模块的依赖与潜在局限。

在实验规模方面，本文最大测试拓扑包含40个主机节点与8台交换机。该规模的设计参考了代表性框架KlonetAI[20]（其实验拓扑包含36台主机），以验证ResilM-IBN在多业务场景下的可扩展性与稳定性。该拓扑已涵盖复杂的多链路和多业务交互场景，能够充分验证本文框架在多意图解析、闭环执行与自愈机制下的表现。需要说明的是，本文原型系统尚未实现预执行验证模块，实验重点主要聚焦于意图执行、自愈能力与系统扩展性。

3.2 常见意图执行性能验证

为了验证多智能体架构在常见网络配置任务下的执行性能与稳定性，本文重点评估平均执行时间与执行成功率，并分析任务拆解复杂度对性能的影响。目前IBN研究尚无统一的公共意图数据集，不同框架的意图表达依赖具体控制器与网络场景设置。参考KlonetAI、MNC、GIA[14]等代表性工作，本文采用自定义的典型意图集进行实验，以验证ResilM-IBN在多类型任务下的执行性能与鲁棒性。实验选择15条涵盖常见网络管理操作的意图，包括拓扑创建、下发阻断流表、带宽限制等。每条意图的首个操作均为创建全新拓扑，且拓扑规模不同，确保测试覆盖多样化场景。意图执行完成后清理拓扑，避免残留状态对后续测试的影响。为保证结果的统计稳定性并减小随机波动影响，本文对每条意图重复执行10次取平均值。重复次数的设定参考了GIA中采用的5次重复实验设计，并在此基础上适当增加以提高结果的置信度，同时控制实验成

本。记录执行时间与执行结果，计算平均执行时间与执行成功率，同时统计任务拆解的步骤数和每条意图所需的 Token 数量，以分析其对执行时间的影响。

常见意图性能执行结果如图 3 所示，其中，横轴为意图编号，左纵轴为平均执行时间（柱状图顶部数值表示），右纵轴为执行成功率（空心圆点表示）。意图分解步骤数如图 4 所示。

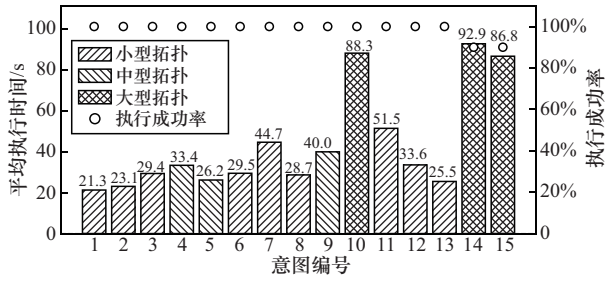

图 3　常见意图性能执行结果

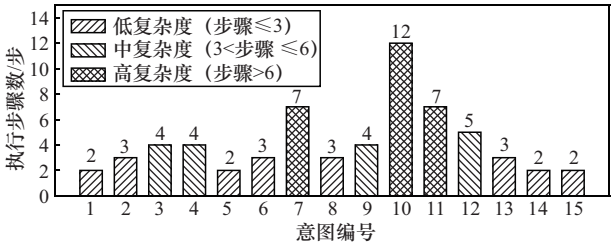

图 4　意图分解步骤数

从图 3 可以看出，系统在大多数场景下均能保持较高的意图执行成功率。这体现了 ResilM-IBN 的统一结构化消息机制和任务链解析流程能够有效保障多智能体间的语义一致性与执行可追溯性，确保复杂意图在动态环境下的正确落地。

在执行时间方面，大多数意图的平均执行时间控制在 60 s 以内，符合交互式配置的性能要求。然而，10 号、14 号和 15 号的意图执行时间显著较长（分别为 88.3 s、92.9 s 和 86.8 s），较整体平均值高出约 40%~60%。结合图 4 可见，这些意图要么涉及较大的拓扑规模（14 号、15 号），要么包含更多的任务拆解步骤（10 号意图共 12 步），因此执行时间主要受任务复杂度与拓扑初始化的双重影响。

整体来看，多智能体架构在任务拆解与执行调度中的协同能力，使其即便在高复杂度任务和大型拓扑下，仍能维持稳定表现。这不仅验证了 ResilM-IBN 的鲁棒性，也为后续支持更复杂的网络管理意图提供了架构基础。

意图执行消耗 Token 数量如图 5 所示。由图 5 可以看出，各类意图的 token 开销整体保持在可控范围内，未随意图复杂度或拓扑规模显著放大。其中，意图解析阶段的 token 消耗较为稳定，而随意图复杂度变化的开销主要集中在 JSON 指令生成阶段。

结合图 3 和图 4 的实验结果可知，尽管部分复杂意图在执行时间上有所增加，但对应的 token 消耗并未出现指数级增长，表明 ResilM-IBN 在引入多智能体闭环执行机制的同时，能够有效约束语言模型的推理开销。

3.3　协作式自愈机制验证

本节验证本文协作式自愈策略在动态网络环境中的正确性与有效性，并评估其在不同通信状态下的性能表现。多智能体解耦架构使自愈功能能够独立集成并与核心控制逻辑隔离，这在传统单智能体系统中难以实现，因此该实验不仅验证功能正确

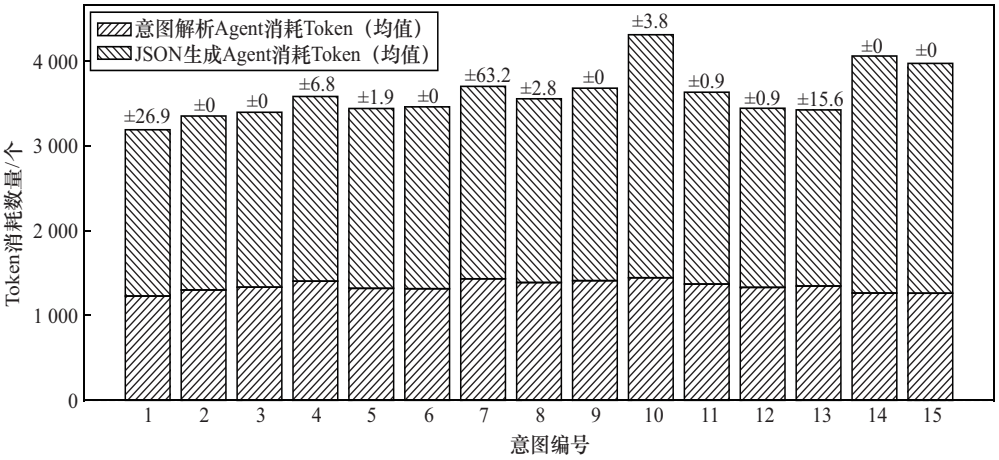

图 5　意图执行消耗 Token 数量

性，也验证架构在功能扩展与协作上的优势。

实验采用8台交换机组成的环状拓扑，每台交换机连接8台主机。首先向拓扑中下发10对阻断流表，使部分主机对无法通信。随后构造20条意图，前10条为“修复”类（目标主机初始不可达），后10条为“不需要修复”类（目标主机初始可达），执行过程中动态判断是否触发修复流程。每轮执行全部20条意图后，清理拓扑并重新下发阻断流表，同样重复10轮以获取平均值。

修复动作触发成功率如图6所示。从图6可以看出，在修复意图中，触发成功率为100%；在不需要修复意图中，未发生误触发，保持0触发率，验证了策略判断的准确性。

自愈意图执行成功率如图7所示。从图7可以看出，无论初始连通状态如何，20条意图的最终通信可达率均为100%，表明系统在自愈场景下能稳定实现用户目标。

自愈意图执行时间如图8所示。从图8可以看出，修复意图整体平均执行时间为24.3 s，不需要修复意图整体平均执行时间为22.2 s，额外耗时约2.1 s，主要来自流表清理与允许规则下发，符合预期且处于可接受范围。

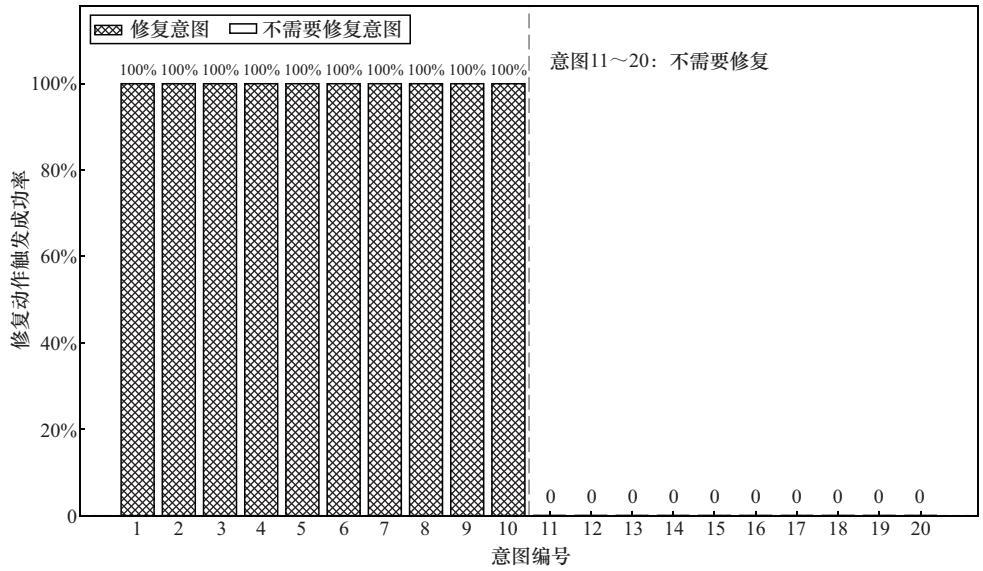

图6　修复动作触发成功率

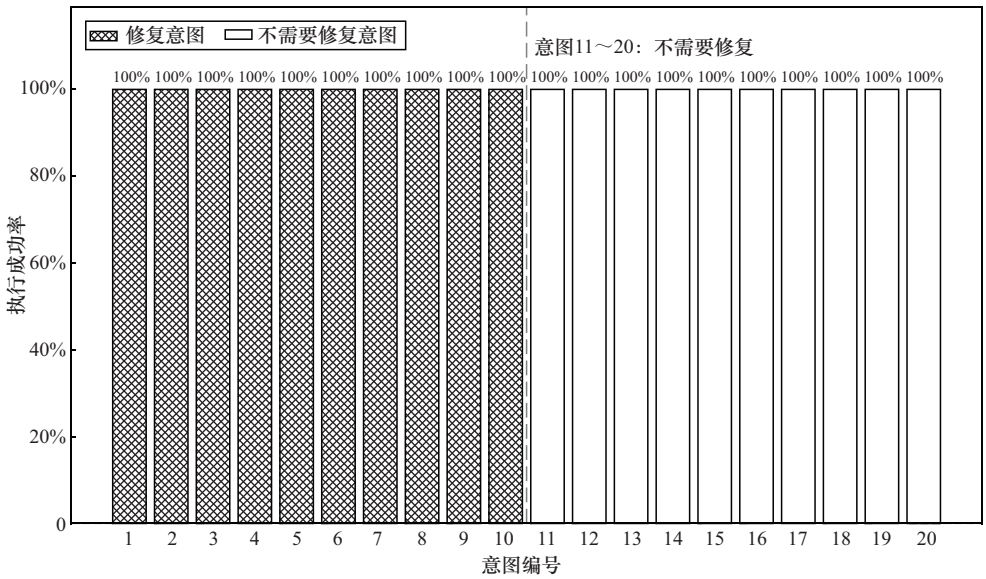

图7　自愈意图执行成功率

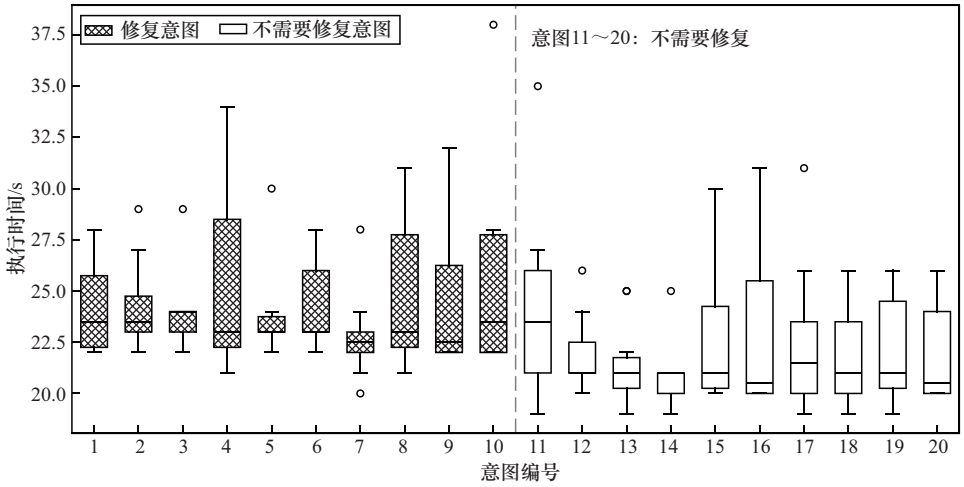

图8　自愈意图执行时间

实验结果验证了协作式自愈策略在准确性与运行层鲁棒性方面的优势，系统能够在不同初始通信状态下正确触发或跳过修复流程，并确保最终通信恢复目标的实现。这种恢复能力体现了系统在遭遇异常或链路中断时的自适应稳定性，是 ResilM-IBN所强调的运行时鲁棒性。

3.4　不同LLM后端对系统性能的影响

为分析不同LLM后端对系统整体性能的影响，本节在保持3.2节实验中网络拓扑、意图集合与执行流程不变的前提下，仅替换系统中用于意图解析与指令生成的LLM后端实例。通过对比不同模型配置下的意图执行成功率、执行时间及token消耗情况，评估本文系统在多种LLM支持下的适配性与运行开销差异。

3.4.1　意图执行成功率与时间分析

不同LLM后端下系统意图执行成功率与执行时间对比如图9所示。由图9可以观察到，在统一的意图输入与执行流程条件下，不同LLM后端对系统端到端执行性能产生了明显影响。不同模型均能够在意图解析与指令生成阶段保持较高的执行成功率，表明本文系统对不同 LLM后端具有良好的适配性。

在执行时间方面，不同模型之间存在较为显著的差异。部分LLM后端在保持较高成功率的同时，能够有效降低意图处理与执行阶段的整体耗时；引入显式推理机制的模型虽然在成功率上表现稳定，但其端到端执行时间相对较高。这一结果表明，LLM后端的推理策略与调用方式会对系统的实时性产生直接影响。

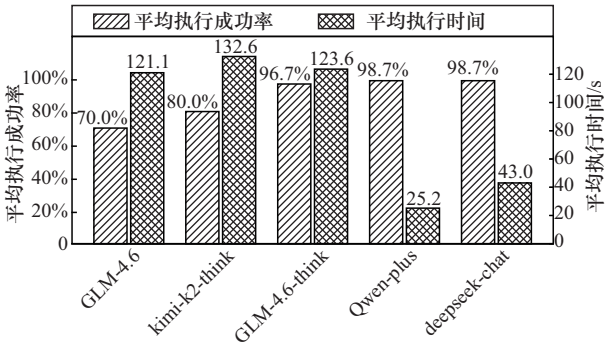

图9　不同LLM后端下系统意图执行成功率与执行时间对比

综合来看，该实验结果说明本文框架能够在多种LLM后端支持下稳定运行，同时在执行成功率与时间之间呈现出不同的性能权衡特征，为后续LLM后端的选择与配置提供了实验依据。

3.4.2　Token消耗与系统开销分析

不同LLM后端下系统Token消耗对比如表2所示。由表2可以看出，在保持统一意图输入与执行流程的前提下，不同LLM后端在Token使用规模上存在一定差异，反映了其在意图理解与指令生成策略上的实现差别。

表 2　不同LLM后端下系统Token消耗对比

LLM实例	平均意图解析Token消耗/个	平均JSON生成Token消耗/个	平均总Token消耗/个
GLM4.6-nothink	2 834	3 388	6 223
GLM4.6-think	2 093	3 844	5 936
kimi-k2-think	2 807	3 292	6 099
Qwen-plus	1 428	2 475	3 902
deepseek-chat	1 344	2 262	3 606

总体而言，引入更复杂推理机制的模型在意图执行过程中通常伴随更高的Token消耗，而部分非推理配置在保证功能正确性的同时，能够有效降低系统调用开销。上述结果表明，LLM后端的选择会对系统的资源消耗产生直接影响，但并未改变本文框架在意图执行层面的可用性与稳定性。

该实验进一步验证了本文意图驱动网络管理框架在多种LLM后端支持下的良好适配能力，为后续根据实际部署需求在性能与资源开销之间进行权衡提供了实验依据。

3.5 IBN系统横向对比

为进一步凸显ResilM-IBN在体系结构与运行性能上的优势，本节将其与现有代表性系统进行横向比较。选择的对比对象包括：MNC（multi-agent network configurator，强调多智能体协作）和KlonetAI（LLM驱动的单智能体系统，具备自然语言到配置的映射能力）。

需要指出的是，由于KlonetAI和MNC尚未公开其完整实现源码，且其实验分别基于特定的仿真平台（Klonet与Cisco Packet Tracer）开展，本文难以在完全一致的物理拓扑与意图集合下对其进行复现实验。这一差异在一定程度上限制了跨系统性能指标在统一基准下的直接对比。本文采用原文文献数据进行间接对比，结合定性功能分析，以全面评估不同框架在时间、准确率及功能完备性上的差异。代表性IBN系统在相近任务场景下的性能数据参考对比如表3所示。

基于表3的数据与架构特性具体分析如下。

1) 准确率维度的分层对比与优势分析：现有工作与ResilM-IBN在准确率评估的侧重点上存在显著差异。

意图解析层：KlonetAI聚焦于将自然语言转化为API调用的映射质量。根据文献[20]中Table 1报告的实验结果，其在Klonet容器化仿真平台上可实现98.1%~100%的翻译准确率，该指标定义为意图文本被正确转化为目标API调用的比例。ResilM-IBN在此阶段通过受限动作集约束（详见2.1节），同样实现了精准的意图映射。

配置生成层：MNC的核心评估指标是生成CLI配置代码的正确性，文献[21]在Table 1中给出了基于GPT-4o mini后端的实验结果，其配置生成准确率在不同复杂度任务下呈现26.2%~69.2%的波动。该指标反映的是静态生成代码与目标配置的一致性，易受LLM幻觉与上下文复杂度影响。

端到端达成层（本文关注点）：与现有工作主要关注“解析”或“生成”的中间状态不同，ResilM-IBN关注的是最终网络状态的真实交付。得益于“意图–验证–执行–再验证”的闭环机制，系统确保了98.6%的端到端意图达成率。此外，ResilM-IBN独有的协作式自愈能力作为一种额外的鲁棒性保障功能，使系统不仅能精准执行正常任务，更具备了应对运行时突发异常（如链路中断）的韧性，这是单纯依赖静态配置生成的系统所无法具备的特性。

2) 执行时间与开销分析：在执行效率方面，ResilM-IBN处理复杂意图的执行时间为21.3~92.9 s，该时间统计包含意图验证、网络状态探测以及必要的自愈修复过程。对比而言，MNC在文献[21]的Table 1中报告的执行时间为35.36~85.46 s，该指标主要反映多智能体推理与配置代码生成所需时间；

表3 代表性IBN系统在相近任务场景下的性能数据参考对比

对比维度	KlonetAI[20]	MNC[21]	ResilM-IBN（本文）
测试基准环境	Klonet（基于容器的仿真平台）	Cisco Packet Tracer（网络仿真模拟器）	Mininet + Ryu（真实SDN控制器原型）
拓扑规模	9~30个节点（Fat-Tree）	7~10个网络设备	3~48个节点（环状拓扑）
意图执行准确率	98.1%~100%（意图解析准确率）	26.2%~69.2%（CLI命令生成准确率）	98.6%（端到端意图达成率）
意图执行时间/s	40~60（仅意图翻译与下发）	35.36~85.46（含多智能体推理与指令生成耗时）	21.3~92.9（含验证、探测与自愈闭环）
是否支持闭环验证	无（单向意图下发）	部分（基于LLM的静态代码审查）	完整（执行–检测–再验证）
是否支持运行时自愈	不支持	不支持	支持（基于反馈的自动修复）

KlonetAI的执行时间则主要来源于意图翻译与下发过程，文献[20]中基于图 5 的实验结果显示，其在30个节点规模下的耗时为40~60 s。

需要注意的是，三者在时间上的差异并非单纯由算法效率决定，而是源于架构设计目标的不同：KlonetAI采用“即发即弃”的单向执行模式，时间最低但缺乏执行结果保障；ResilM-IBN 虽然引入了完整的闭环验证与自愈流程，在时间上有所增加，但这种以时间换取可靠性的设计更契合生产网络对稳定性的实际需求。

局限性讨论：需要指出的是，由于底层仿真器（Cisco Packet Tracer vs Mininet）与硬件环境的差异，上述量化数据主要作为性能量级的参考，而非绝对性能的直接对标。然而，这种对比依然验证了ResilM-IBN 在引入复杂的闭环自愈机制后，并未导致系统开销呈指数级增长，仍在可接受的时间范围内实现了超出同类方法的执行可靠性。

4　结束语

本文面向 SDN 环境，研究了 IBN 在多业务、多策略场景下的自愈与闭环管理问题。首先，设计了一种自愈型多智能体意图驱动网络管理框架 ResilM-IBN，实现了从意图解析到执行验证的全过程自治。随后，提出了结构化通信与任务协作、串行化闭环执行以及协作式自愈 3 项核心机制，分别用于解决语义一致性、执行可控性和运行时异常修复等关键问题。最后，基于 Mininet 与 Ryu 的原型进行实验，结果表明，ResilM-IBN 在复杂意图场景下保持 98.6% 的执行成功率和 100% 的修复成功率，显著提升了系统的鲁棒性与自适应性。未来的工作将进一步探索多智能体间的动态协同优化与跨域意图解析，以增强系统的扩展性和智能演化能力。

参考文献：

[1] 田晨景, 谢钧, 曹浩彤, 等. 5G 网络切片研究进展[J]. 计算机科学, 2023, 50(11): 282-295.
Tian C J, Xie J, Cao H T, et al. Research developments of 5G network slicing[J]. Computer Science, 2023, 50(11): 282-295.

[2] Leivadeas A, Falkner M. A survey on intent-based networking[J]. IEEE Communications Surveys & Tutorials, 2023, 25(1): 625-655.

[3] Liu X L, Li T, Yang C G, et al. Generic intent-driven networking paradigm with different levels of policy abstraction[J]. IEEE Communications Magazine, 2025, 63(4): 162-168.

[4] Boutouchent A, Meridja A N, Kardjadja Y, et al. AMANOS: an intent-driven management and orchestration system for next-generation cloud-native networks[J]. IEEE Communications Magazine, 2024, 62(6): 42-49.

[5] Jacobs A S, Pfitscher R J, Ferreira R A, et al. Refining network intents for self-driving networks[J]. ACM SIGCOMM Computer Communication Review, 2019, 48(5): 55-63.

[6] Wang C J, Scazzariello M, Farshin A, et al. Making network configuration human friendly[J]. arXiv Preprint, arXiv: 2309.06342, 2023.

[7] 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 Press, 2019: 214-226.

[8] Lin J Y, Dzeparoska K, Tizghadam A, et al. AppleSeed: intent-based multi-domain infrastructure management via few-shot learning[C]//Proceedings of the 2023 IEEE 9th International Conference on Network Softwarization (NetSoft). Piscataway: IEEE Press, 2023: 539-544.

[9] Dzeparoska K, Lin J Y, Tizghadam A, et al. LLM-based policy generation for intent-based management of applications[C]//Proceedings of the 2023 19th International Conference on Network and Service Management (CNSM). Piscataway: IEEE Press, 2023: 1-7.

[10] 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 Press, 2023: 189-195.

[11] Lira O G, Caicedo O M, Fonseca N L S D. Large language models for zero touch network configuration management[J]. IEEE Communications Magazine, 2025, 63(7): 146-153.

[12] El-Hassany A, Tsankov P, Vanbever L, et al. Network-wide configuration synthesis[C]//Computer Aided Verification. Berlin: Springer, 2017: 261-281.

[13] Dzeparoska K, Tizghadam A, Leon-Garcia A. Emergence: an intent fulfillment system[J]. IEEE Communications Magazine, 2024, 62(6): 36-41.

[14] Kou S W, Yang C G, Gurusamy M. GIA: LLM-enabled generative intent abstraction to enhance adaptability for intent-driven networks[J]. IEEE Transactions on Cognitive Communications and Networking, 2025, 11(2): 999-1012.

[15] Angi A, Sacco A, Marchetto G. LLNet: an intent-driven approach to instructing softwarized network devices using a small language model[J]. IEEE Transactions on Network and Service Management, 2025, 22(4): 3403-3418.

[16] Mehmood K, Kralevska K, Palma D. Knowledge-driven intent lifecycle management for cellular networks[J]. IEEE Transactions on Network and Service Management, 2025, 22(5): 4806-4826.

[17] Dzeparoska K, Tizghadam A, Leon-Garcia A. Intent assurance using LLMs guided by intent drift[C]//Proceedings of the NOMS 2024-2024 IEEE Network Operations and Management Symposium. Piscataway: IEEE Press, 2024: 1-7.

[18] Manias D M, Chouman A, Shami A. Towards intent-based network management: large language models for intent extraction in 5G core networks[C]//Proceedings of the 2024 20th International Conference on the Design of Reliable Communication Networks (DRCN). Piscataway: IEEE Press, 2024: 1-6.

[19] Yin S K, Fu C Y, Zhao S R, et al. A survey on multimodal large lan-

guage models[J]. National Science Review, 2024, 11(12): nwae403.
[20] Li Q, Xiong Y X, Li Z H, et al. KlonetAI: automating $(com)_2$Nets management with human language intents[J]. IEEE Network, 2025, 39(3): 12-19.
[21] Wang C, Li H. MNC: a multi-agent framework for complex network configuration[J]. Intelligent Systems with Applications, 2025, 26: 200531.
[22] Brodimas D, Birbas A, Kapolos D, et al. Intent-based infrastructure and service orchestration using agentic-AI[J]. IEEE Open Journal of the Communications Society, 2025, 6: 7150-7168.
[23] Dehghan Biyar E, D'Angelo M, Cisneros J C, et al. Autonomous conflict handling in intent-based management[J]. Computer Networks, 2025, 271: 111561.
[24] HONG S R ZHUGE M C, CHEN J, et al. MetaGPT: meta programming for a multi-agent collaborative framework[C]//Proceedings of the 12th International Conference on Learning Representations. Vienna: OpenReview, 2024.
[25] Lingga P, Jeong J, Dunbar L. ICSC: intent-based closed-loop security control system for cloud-based security services[J]. IEEE Communications Magazine, 2025, 63(4): 169-175.
[26] 付永红, 毕军, 张克尧, 等. 软件定义网络可扩展性研究综述[J]. 通信学报, 2017, 38(7): 141-154.
Fu Y H, Bi J, Zhang K Y, et al. Scalability of software defined network[J]. Journal on Communications, 2017, 38(7): 141-154.
[27] Kou S W, Yang C G, Gurusamy M. SAFLA: semantic-aware full lifecycle assurance for intent-driven networks[J]. IEEE Transactions on Cognitive Communications and Networking, 2026, 12: 658-672.
[28] Risdianto A C, Usman M, Ahmad Rathore M. Transforming network management: intent-based flexible control empowered by efficient flow-centric visibility[J]. Future Internet, 2024, 16(7): 223.
[29] Meijer A R, Boldrini L, Koning R, et al. User-driven path control through intent-based networking[C]//Proceedings of the 2022 IEEE/ACM International Workshop on Innovating the Network for Data-Intensive Science (INDIS). Piscataway: IEEE Press, 2023: 9-19.
[30] Tran H T, Domerçant J C, Mavris D N. A network-based cost comparison of resilient and robust system-of-systems[J]. Procedia Computer Science, 2016, 95: 126-133.
[31] Singh B, Kannojia S P. A review on software quality models[C]//Proceedings of the 2013 International Conference on Communication Systems and Network Technologies. Piscataway: IEEE Press, 2013: 801-806.

[作者简介]

董黎刚（1973– ），男，浙江上虞人，博士，浙江工商大学教授，主要研究方向为智慧网络与智慧教育。

苟敬文（2000– ），男，四川巴中人，浙江工商大学硕士生，主要研究方向为智慧网络。

蒋献（1988– ），男，浙江兰溪人，浙江工商大学讲师，主要研究方向为智慧网络与智慧教育。

张子天（1988– ），男，河北石家庄人，博士，浙江工商大学副研究员，主要研究方向为基于机器学习的网络流量预测与资源管理。

诸葛斌（1976– ），男，浙江兰溪人，博士，浙江工商大学教授，主要研究方向为网络和通信技术、互联网技术和网络安全。