A Framework to Evaluate AI Agents for Network Configuration

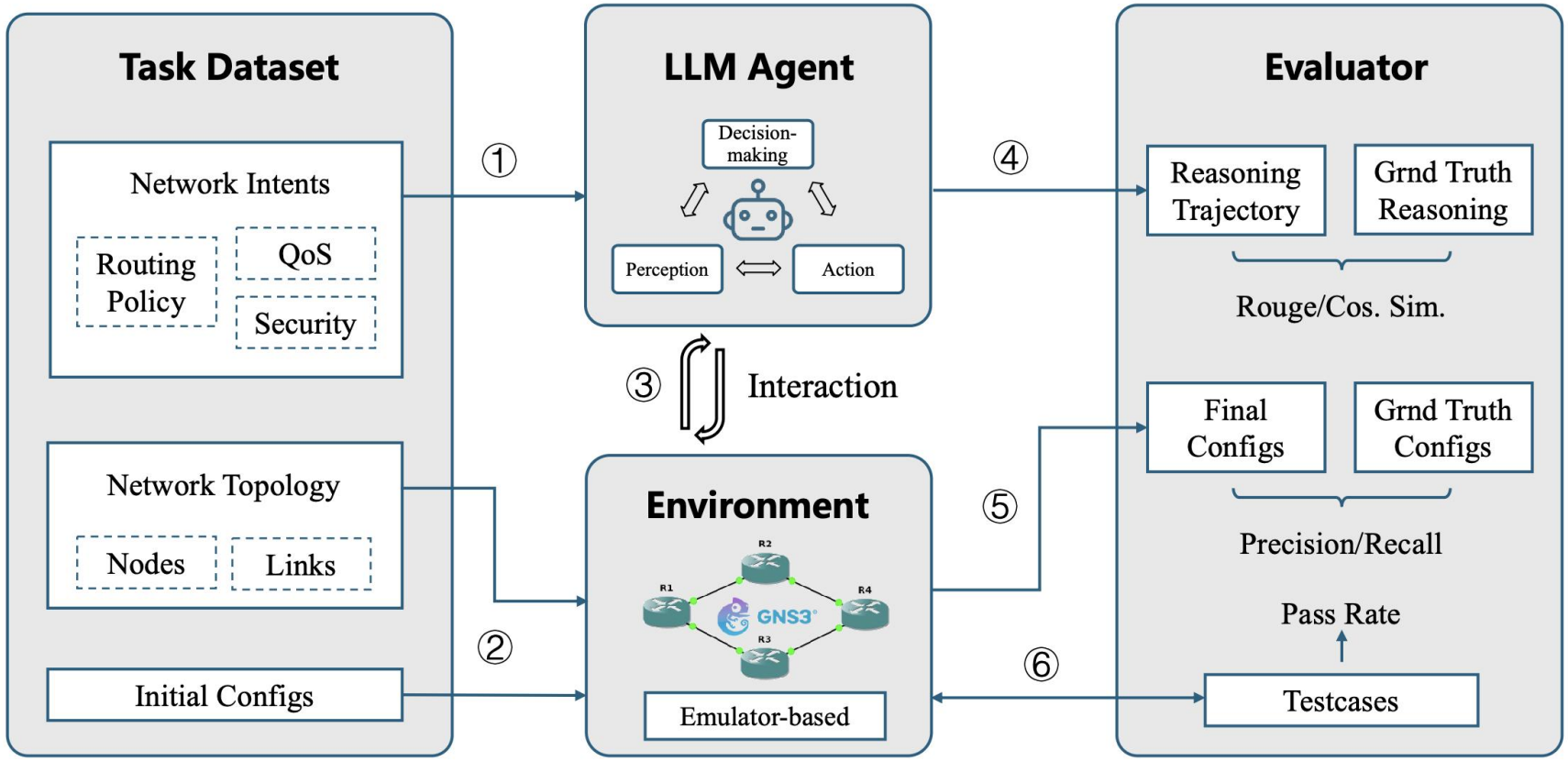
Yong Cui, Chang Liu, Xiaohui Xie, Chenguang Du
Tsinghua University, Beijing Zhongguancun Laboratory

March, 2026

Motivation

- Challenges
  - Existing evaluation methods are too simplistic to reflect the complexity and diversity of real network environments
  - Lack of standardized evaluation benchmarks for Agent network configuration capabilities
- Propose a evaluation framework for LLM network configuration agents
  - ✓ Define task datasets covering various network configuration tasks
  - ✓ Integrate a simulation environment with real vendor device image
  - ✓ Offer triple evaluation metrics: reasoning quality, command accuracy, functional correctness

Framework Overview



Key Components:

- **Task Dataset**
- **Emulator Environment**
- **LLM Agent**
- **Evaluator**

Task Dataset & Emulator Environment

- Task Dataset
  - 40 configuration tasks in JSON format
  - Covers routing, QoS, and security domains
  - Each task includes six parts: **intent, topology, initial config, ground-truth config, reasoning trace, and testcases**
- Emulator Environment
  - Built on GNS3 with real vendor device images
  - Agent-Network Interface (ANI) provides 4 APIs to the agent:
    - ✓ get_topology, get_running_config, update_config, execute_validation
  - Task Evaluation Interface provides 2 APIs to the evaluator:
    - ✓ Export final configs of all devices; Run predefined testcases

LLM Agent & Evaluator

- AI Agent (Under Evaluation)
  - Interacts with emulator via ANI (get topology, get config, update config, validate)
  - Optional: external knowledge retrieval (vendor manuals)
  - Two interaction strategies:
    - ✓ Single-Turn: complete reasoning and commands in one pass
    - ✓ Multi-Turn (ReAct-style): interleaved reasoning and actions with feedback
- Evaluator
  - Computes three scores for each task:
    - ✓ Reasoning Score: cosine similarity between agent and ground-truth reasoning
    - ✓ Command Score: F1 of precision and recall on extracted config commands
    - ✓ Testcase Score: pass rate of verification testcases in the emulator
  - (See details in Refined Evaluation Metrics slide)

Evaluation Workflow

Step 1: Task Dataset gives a task definition to the Agent， Environment and Evaluator .

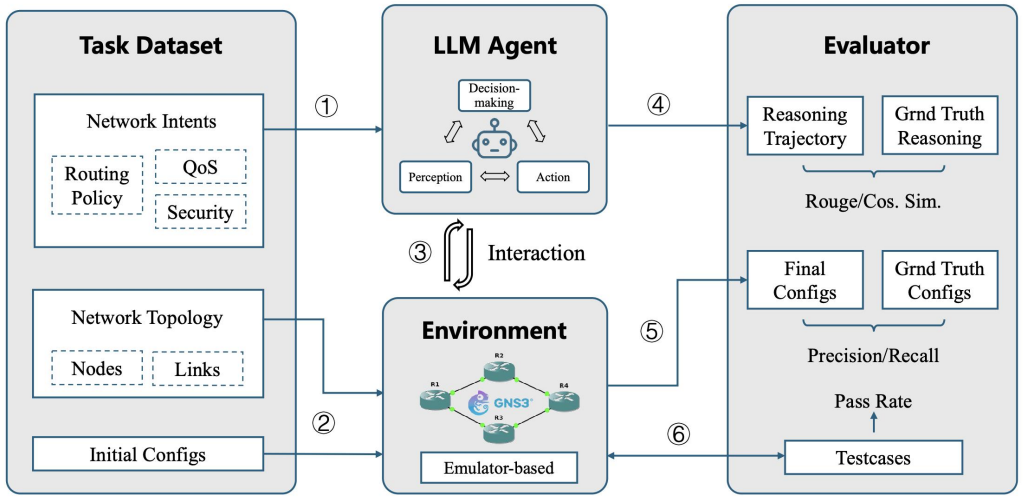
Step 2: The Environment sets up the network topology and initial configs

Step 3: The Agent interact with Environment.

Step 4: The Agent provides its reasoning process to the Evaluator.

Step 5: The Environment gives final configs to the Evaluator.

Step 6: The Evaluator calculates three scores by running tests.



Changes in draft-01

- New Section 5: MCP-Based Implementation
  - Add MCP (Model Context Protocol) as a standard way to implement ANI
  - Define MCP tools for all core network operations
  - Add tools for advanced use: batch_configure_devices, check_device_status
- More Details on Evaluation Metrics
  - Add formulas for S_reasoning, S_command, S_testcase
  - Add step-by-step explanation of how each score is computed
- Terminology and Description Updates
  - Add MCP definition in Terminology section
  - Rename execute-cmd to execute_validation
  - Specify task domains: routing, QoS, and security
  - Add new fields: annotated reasoning trace, expert-crafted testcases

New Section: MCP-Based Implementation

- MCP as a standardized interaction interface.
  - Agent acts as the MCP Client, and the Network Environment acts as the MCP Server
- Why use MCP?
  - ✓ Standardization: a unified interface that works across different Agents and environments
  - ✓ Extensibility: Easy to add new tools without changing the Agent
  - ✓ Traceability: All tool calls and results are logged
- MCP Tool Definitions
  - Core tools from ANI: get_topology, get_running_config, update_config
  - Validation tool: execute_validation (run show/ping to verify results)
  - Advanced tools: batch_configure_devices, check_device_status

Refined Evaluation Metrics

- Reasoning Score (S_reasoning)
  - Measures if agent reasoning aligns with the expert-annotated ground truth
  - One-shot: prompt asks agent to output reasoning before commands
  - Multi-turn: a helper LLM summarizes steps into one reasoning trace
  - Formula: S_reasoning = cosine_similarity(r_agent, r_gt)
- Command Score (S_command)
  - Measures if agent generates correct configuration commands
  - Compare the applied commands against the ground truth
  - Use ciscoconfparse + wildcard matching for fair comparison
  - Formula: S_command = 2 * Precision * Recall / (Precision + Recall)
- Testcase Score (S_testcase)
  - Measures if configuration actually works in the network
  - Run test cases in the emulator and take the pass rate as the score
  - Examples: routing (next hop), ACL (allow/deny), QoS (policy)
  - Formula: S_testcase = Passed / Total Testcases

Metrics cover reasoning quality, command accuracy, and functional correctness

Summary

➢ Main Contributions

- Evaluation framework for network configuration agents with end-to-end evaluation
- Established standardized task dataset and evaluation metrics
- Designed interactive environment based on real device simulation

➢ Future Work

- Support multi-agent collaboration via A2A protocol to handle complex configuration tasks
- Increase task complexity with larger topologies and multi-domain configurations
- Use Agent Skills to define evaluation workflows for simpler implementation and more flexible benchmarking

➢ Goal: An automated, fair, and comprehensive AI Agent evaluation framework

Thanks!
We welcome collaborators!

Q&A

Chenguang Du
ducg@zgclab.edu.cn