Large Language Model Driven Automated Network Protocol Testing

Yong Cui, **Yunze Wei**, Kaiwen Chi, Xiaohui Xie

Tsinghua University

Why Automate Protocol Testing?

Vendors rely on extensive testing to ensure device behavior

- **Goal:** Ensure protocol **conformance**, expected **function** and **performance**
- Protocol standards evolve rapidly, and **private protocols** are widely used.
- Traditional testing methods struggle to keep up with **new demands** in:
  - Industrial Internet, Satellite Communications, and Low-altitude economy, etc.

Traditional Network Protocol Testing is Labor-Intensive

- **Manual** test case writing dominates current practices
- Engineers need to **analyze** protocol specification, **write** test cases, **convert** test cases into executable scripts, **debug and refine** in multiple rounds
- Drawbacks:
  - Time-consuming and low efficiency
  - Limited coverage

Automated Protocol Testing Framework

- Goal
  - Automate the protocol testing pipeline from **specification** to **test case execution**
- Input
  - RFC docs, Human testing intent
- Output
  - Test cases, Tester script, DUT config (Intermediate artifacts, reusable)
  - Test Report (Final result)

```
+---------+                                        +----------+
|   RFC   |   +----------+   +-----------------+   | Test Env.|
|Documents|-->|          |-->|  Tester Script  |-->| +------+ |
+---------+   |Test Case |   |   Generation    |   | |Tester| |
              |          |   | --------------- |   | +-^----+ |
+---------+   |Generation|   |DUT Configuration|   |   |  |   |
|  Human  |-->|          |-->|   Generation    |-->| +----v-+ |
| Intent  |   +----------+   +-----------------+   | | DUT  | |
+---------+        ^                 ^             | +------+ |
                   |                 |             +----------+
              +-----------------------------+          |  Test
              |   Feedback and Refinement   |<---------+ Report
              +-----------------------------+
```

Typical Network Protocol Test Automation Framework (with **six levels** of protocol testing automation)

➢ **Draft: draft-cui-nmrg-auto-test**

A Structured Specification for Protocol Test Cases

- Define **6 key components** for protocol test cases
  - *Metadata*, *Topology*, *Setup*, *Parameters*, *Procedure*, *Evaluation Criteria*
- Classify test cases into **5 types**
  - Conformance, Functional, Performance, Interoperability, Robustness
- Provide **a machine-readable example** to enable automation

- **Test Case Structure Overview**

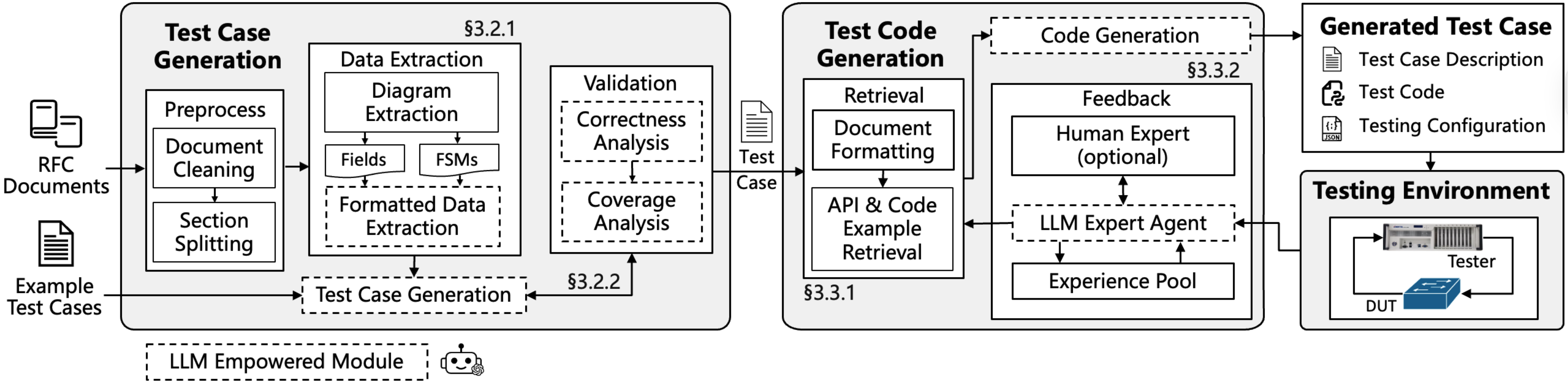
Component	Description
Metadata	Test ID, title, purpose, protocol, references
Topology	Node roles, links, addressing
Setup	Initial DUT/tester configurations
Parameters	Protocol knobs, traffic profiles
Procedure	Step-by-step actions and expected behavior
Evaluation	Metrics, pass/fail criteria, logs required

- **Application Scenarios**
  - Vendor internal testing
  - Pre-deployment acceptance testing

➢ **Draft: draft-cui-bmwg-testcase-spec**

NeTestLLM: L2->L3 Automation Example

- NeTestLLM automates network protocol testing via two stages:
  - ① Test Case Generation: Extract structured spec -> Generate test cases -> Validation
  - ② Test Code Generation: Retrieve doc -> Synthesize code -> Refine with feedback



➢ **Large Language Model Driven Automated Network Protocol Testing. ANRW 2025.**

Summary

- **NMRG:** draft-cui-nmrg-auto-test
- **BMWG:** draft-cui-bmwg-testcase-spec
- **ANRW:** Large Language Model Driven Automated Network Protocol Testing
- **Hackathon**

We welcome your feedback and comments!
We are also looking for collaboration!

Yong Cui, **Yunze Wei**, Kaiwen Chi, Xiaohui Xie

Tsinghua University