Song Sa

Beijing, China 15083129061 shemol106@gmail.com https://shemol.tech

Profiles

SherlockShemol

Education

Beijing University of Posts and Telecommunications **Sep 2020 - Jul 2024**
Communication Engineering Bachelor's Degree
85.01/100 (Top 29.34%)

Beijing University of Posts and Telecommunications **Sep 2024 - Present**
Information and Communication Engineering Master's Degree

Skills

- Passionate about exploring various AI products, including Z-Code, Alma, Cursor, Antigravity, v0, Lovable, Cline, Readever, etc.
- Follow technical blogs from Claude, OpenAI, and tweets/blog posts from Andrej Karpathy, Lee Robinson, etc.
- Familiar with HTTP/HTTPS protocols and common data structures and algorithms
- Proficient in HTML5, CSS3, and ES6+ syntax
- Experienced in TypeScript for strongly-typed programming to enhance code robustness
- Familiar with React framework and Hooks programming pattern, understanding common state management solutions
- Familiar with basic configuration of frontend build tools like Webpack/Vite
- Experienced in Go for open source project development, familiar with related data types and basic concurrent programming
- Proficient in Go and concurrent programming, with Kubernetes Operator development experience
- Proficient in Docker containerization and Linux system configuration, capable of independently completing application containerization deployment and environment setup
- Proficient in Git distributed version control system, with open source community collaboration experience, capable of efficiently designing branch strategies, resolving code conflicts, and submitting Pull Requests

Projects

Agora: Distributed Protocol Agent Testing Platform **Mar 2025 - Present**

Tech Stack: Python, asyncio, gRPC, Event-Driven Architecture, Prompt Engineering

Project Overview: Designed and implemented a distributed system testing platform. The core innovation is using LLM as the protocol decision engine, replacing traditional hardcoded state machines. The system adopts a two-layer architecture: the upper Orion layer provides intelligent clients, fault injection, and behavior verification; the lower Constellation layer implements LLM-driven protocol Agents (Raft/PBFT).

- **LLM-Native Protocol Decision Engine:** Delegated the decision logic of Raft/PBFT protocols entirely to LLM. Agents construct structured prompts with STATE (role/term/log status) + TRACE (recent event history), letting LLM output JSON decisions (action + params), executed by deterministic Handlers. Achieved a complete "Perceive→Reason→Execute" AI Agent loop.
- **Constellation Unified Framework:** Designed BaseProtocolAgent abstract base class, encapsulating common components including EventSystem, StateManager, NetworkLayer, and TimerSystem. New protocols only need to implement abstract methods like get_protocol_rules() and make_fallback_decision(), significantly reducing protocol development costs.
- **Safe Fallback and Explainability:** When LLM outputs invalid JSON or violates protocol safety, automatically switches to pure-rule Fallback strategy, ensuring consistency safety takes priority over LLM expression. The STATE/TRACE mechanism preserves complete decision chains for issue tracing and debugging.
- **Intelligent Testing Orion Layer:** Client Agent is LLM-driven, intelligently selecting send/retry/success/fail actions based on response status (ok/redirect/error); Injector supports various fault scenarios including network partition, delay injection, and state tampering; Checker validates system behavior against protocol invariants in real-time.

Consen: Multi-Agent Distributed Protocol Auto-Generation and Verification System **Nov 2025 - Present**

Tech Stack: Python, Multi-Agent, asyncio, Prompt Engineering

Project Overview: Built a Multi-Agent collaboration system that achieves automated generation, testing, and repair of distributed consensus protocol (Raft/EPaxos) code through LLM-driven red-blue adversarial mechanism. The system contains three core Agents: Orchestrator (orchestration agent), Coder (code generation), and Checker (adversarial testing), forming a complete PLAN→BUILD→TEST→FIX→STABLE development loop.

- **Multi-Agent Collaboration Architecture:** Orchestrator Agent serves as the coordinator, coordinating sub-agents (Coder Agent and Checker Agent) through JSON decision protocols to achieve fully automated lifecycle transitions of PLAN→BUILD→TEST→FIX. Uses stateless Prompt design, constructing complete context for each decision round to ensure Agent decision consistency.
- **LLM-Driven Code Generation:** Coder Agent supports three modes: Plan/Build/Fix. Plan mode parses protocol specifications to auto-generate implementation plans; Build mode incrementally constructs code step-by-step; Fix mode combines Failure Log and protocol specifications to locate and fix bugs. Achieves THINK→CODE structured output through Chain-of-Thought prompt engineering.
- **Red Team Adversarial Testing System:** Checker Agent acts as an LLM-driven red team attacker, automatically generating attack plans based on protocol specifications and source code, performing fault injection through DropRule/DelayRule/MutateRule to detect Safety (consistency violation) and Liveness (liveness failure) bugs. Supports both CFT/BFT fault models.
- **Experience-Driven Testing Optimization:** Implemented Tests Memory module for persistent storage and similarity-based retrieval of successful attack patterns; implemented Bug Pattern Loader to retrieve relevant cases from historical bug pattern library for Prompt injection, improving test coverage and bug discovery efficiency.

Awards

University Second-Class Scholarship (2020-2021)	**Sep 2021**
University Third-Class Scholarship (2022-2023)	**Sep 2023**
Second Prize, 'Challenge Cup' Beijing College Students' Academic Science and Technology Competition	**Jul 2023**
2024 Open Source Promotion Plan (OSPP) Completed Successfully	**Nov 2024**
University First-Class Scholarship (2024)	**Nov 2024**
2nd Place, 2025 CloudWeGo Hackathon Finals	**Apr 2025**
University First-Class Scholarship (2025)	**Nov 2025**

Open Source Contributions

KubeEdge-Sedna: Joint Inference and Federated Learning Controller Optimization

PR Link 1: https://github.com/kubeedge/sedna/pull/446
PR Link 2: https://github.com/kubeedge/sedna/pull/445
PR Link 3: https://github.com/kubeedge/sedna/pull/438
PR Link 4: https://github.com/kubeedge/sedna/pull/437

minionS: Added Docker containerization support; Windows support for PDF processing; DeepSeek API support for remote clients

PR Link: https://github.com/HazyResearch/minions/pull/54
PR Link: https://github.com/HazyResearch/minions/pull/47
PR Link 1: https://github.com/HazyResearch/minions/pull/16
PR Link 2: https://github.com/HazyResearch/minions/pull/40

lmp: Added Fedora dependency installation support for eBPF scripts; Implemented automatic KVM BTF detection with vmlinux fallback

PR Link: https://github.com/linuxkerneltravel/lmp/pull/976

Dify: Fixed frontend Chain-of-Thought rendering bug; Fixed memory leak under high load; Added unit tests for Avatar, Chip, Badge components

PR Link 1: https://github.com/langgenius/dify/pull/27776
PR Link 2: https://github.com/langgenius/dify/pull/30236
PR Link 3: https://github.com/langgenius/dify/pull/30201
PR Link 4: https://github.com/langgenius/dify/pull/30119
PR Link 5: https://github.com/langgenius/dify/pull/30096

Cherry Studio: Fixed API Key whitespace truncation; Optimized model state lookup with Map; Fixed global memory settings submission failure; Fixed custom endpoint suffix issue

PR Link 1: https://github.com/CherryHQ/cherry-studio/pull/10751
PR Link 2: https://github.com/CherryHQ/cherry-studio/pull/12161
PR Link 3: https://github.com/CherryHQ/cherry-studio/pull/12147
PR Link 4: https://github.com/CherryHQ/cherry-studio/pull/12163