AGENT

THIRD WEEK

Gülşen Sabak & Süleyman Emir Taşan
Bogazici University Computer Engineering

LANGSMITH - INTRODUCTION

- What is LangSmith?
  - Development platform for LLM applications
  - Created by LangChain team
  - End-to-end solution for building, testing, and monitoring

- Key Value Proposition
  - Transforms LLM development from art to engineering
  - Provides visibility into "black box" LLM operations
  - Accelerates development cycles by 40-60%

LANGSMITH - CORE COMPONENTS

- Development Environment
  - Interactive debugging tools
  - Visual execution graphs
  - Step-by-step inspection capabilities
- Testing Framework
  - Systematic prompt evaluation
  - Performance benchmarking
  - Comparative analysis tools
- Monitoring System
  - Production application tracking
  - Cost and usage analytics
  - Performance monitoring

LANGSMITH - DEVELOPMENT PROCESS

- Prompt Engineering
  - Version control for prompts
  - Template management
  - A/B testing capabilities
- Chain & Agent Development
  - Component-level debugging
  - Visual tracing of execution paths
  - Runtime metrics analysis
- Iteration Process
  - Identify bottlenecks
  - Optimize underperforming components
  - Validate improvements

LANGSMITH - TESTING

- Dataset Management
  - Create test datasets from production data
  - Maintain evaluation benchmarks
  - Share datasets across teams
- Evaluation Methods
  - Model-based automated scoring
  - Human feedback collection
  - Custom evaluation metrics
- Performance Metrics
  - Response quality scoring
  - Latency & throughput analysis
  - Cost efficiency measurements

LANGSMITH - MONITORING

- Real-time Analytics
  - Application health dashboard
  - Token usage tracking
  - Error rate monitoring
- Cost Optimization
  - Identify expensive operations
  - Track spending by component
  - Optimize for efficiency
- Quality Assurance
  - Detect performance degradation
  - Monitor for hallucinations
  - Track user satisfaction metrics

LANGSMITH - ADAPTABILITY

- Ecosystem Compatibility
  - Seamless LangChain integration
  - Support for major LLM providers
  - API access for custom tooling
- Implementation Roadmap
  - Pilot project identification
  - Incremental adoption path
  - Team training resources
- Business Impact
  - Faster time-to-market
  - Reduced operational costs
  - Enhanced application reliability

Paper 1

WHAT IS AN AGENT?

- Agent is an entity which is placed in an environment and senses different parameters that are used to make a decision based on the goal of the entity. the entity performs the necessary action on the environment based on this decision.
- Entity: Entity refers to the type of the agent.
- Environment: This refers to the place where the agent is located.
- Parameters: The different types of data that an agent can sense from the environment are referred to as parameters.
- Action: Each agent can perform an action that results in some changes in the environment.

ENVIRONMENT FEATURES THAT AFFECT THE COMPLEXITY OF AN AGENT BASED SYSTEM

- Accessibility: An accessible environment allows agents to sense and retrieve accurate data. Inaccessible environments introduce uncertainty, requiring agents to handle incomplete or noisy data.
- Example: A network security agent can access all traffic logs in an accessible environment, whereas in a restricted setting, it must infer missing data.

- Determinism: A deterministic environment ensures that every action leads to a predictable outcome. A non-deterministic environment introduces randomness, requiring agents to adapt to unpredictable changes.
- Example: A chess-playing agent operates in a deterministic environment, while a stock-trading agent faces non-determinism.

ENVIRONMENT FEATURES THAT AFFECT THE COMPLEXITY OF AN AGENT BASED SYSTEM

- Dynamism: A static environment remains unchanged unless altered by the agent's actions. A dynamic environment evolves independently of the agent, requiring continuous adaptation.
- Example: Traffic monitoring agents operate in a highly dynamic environment due to fluctuating vehicle movement.

- Continuity: A discrete environment consists of predefined states, making transitions structured and predictable. A continuous environment involves fluid changes, requiring sophisticated models for decision-making.
- Example: A factory robot navigating a grid follows a discrete environment, whereas a drone flying in open space operates in a continuous one.

KEY FEATURES OF AGENTS

- Sociability: Agents communicate and collaborate with peers to enhance efficiency.
- Example: In smart grids, agents share power usage data to optimize energy distribution.

- Autonomy: Agents independently make decisions without centralized control.
- Example: A self-driving car decides routes and speed based on traffic conditions.

- Proactivity: Agents anticipate future conditions and take preemptive actions.
- Example: An e-commerce recommendation agent predicts user preferences and suggests products.

- Reactivity: Agents respond dynamically to environmental changes in real-time.
- Example: A weather monitoring agent updates forecasts based on new sensor data.

- Learning: Agents improve performance through experience and interaction.
- Example: AI-powered chatbots refine responses based on user interactions.

MAS FEATURES

Leadership:

- MAS can be either leader-follow, where a designated agent guides others, or leaderless, where all agents function autonomously.

Heterogeneity:

- MAS can be homogeneous (identical agents) or heterogeneous (diverse agents with varying functionalities).

Topology:

- The structure can be static (fixed connections) or dynamic (changing relationships).

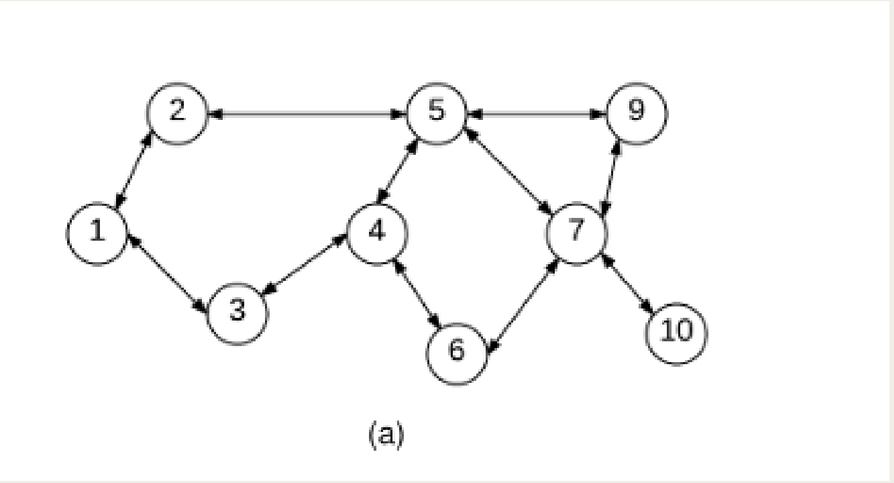
MAS CHALLENGES

- Coordination Control: Ensuring agents work together effectively, including reaching consensus and synchronization.
- Learning: Implementing machine learning in MAS involves high processing and communication overhead, dynamic adaptability, and scalability concerns.
- Fault Detection: Identifying and isolating faulty agents in a decentralized manner.
- Task Allocation: Efficiently distributing tasks among agents while considering computational constraints.
- Localization: Determining agent positions in large-scale MAS with dynamic mobility.
- Security: Addressing threats related to decentralized architectures, trust, and malicious attacks.

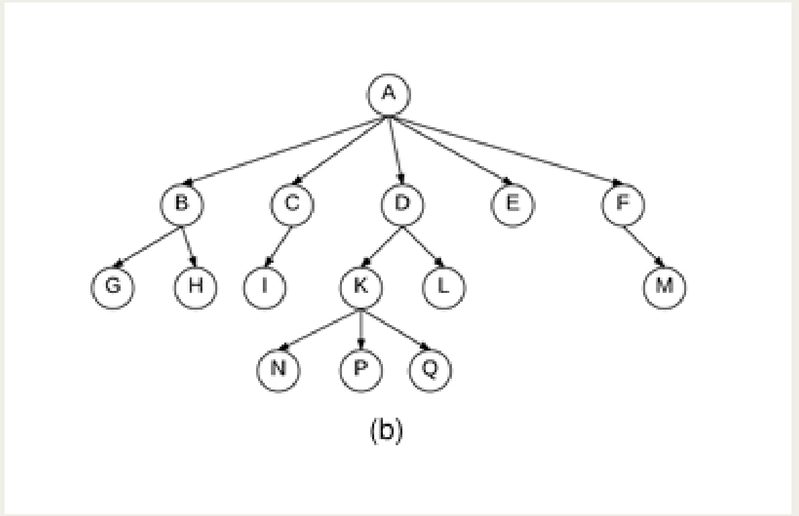
AGENT ORGANIZATION

- Flat: All agents are equal with no central authority, directly communicating with their peers.
- Hierarchical: A tree-like structure where parent agents control subordinate agents.
- Holonic: Agents are grouped into holons with layered communication structures.
- Coalition: Temporary groupings of agents with similar goals.
- Team: Agents collaborate toward a team goal, which may change dynamically.
- Matrix: Agents report to multiple managers based on functions and projects.
- Congregation: Agents form groups based on shared resource needs or goals.

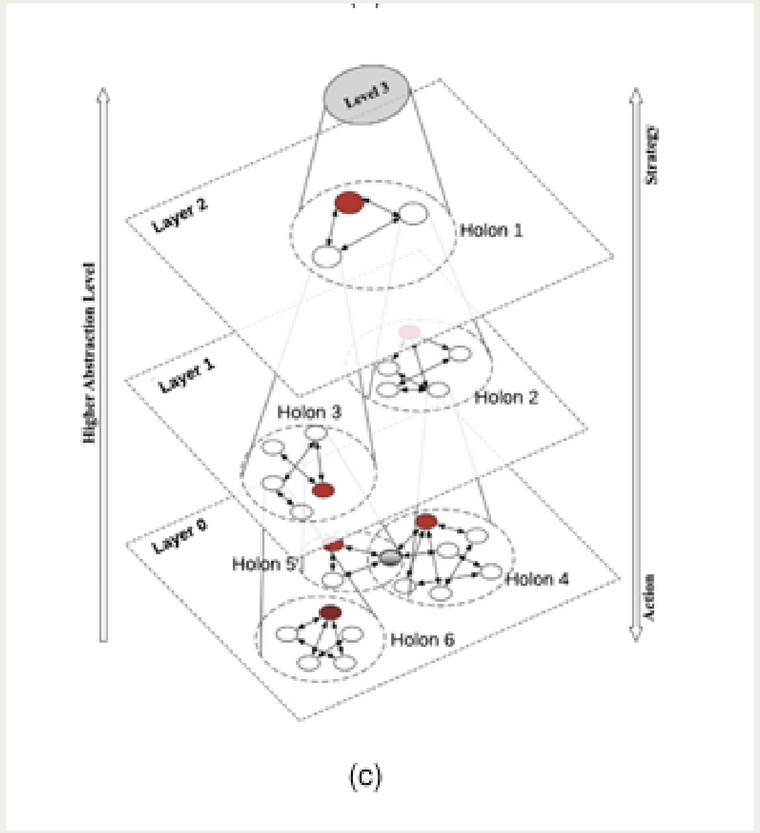
AGENT ORGANIZATION



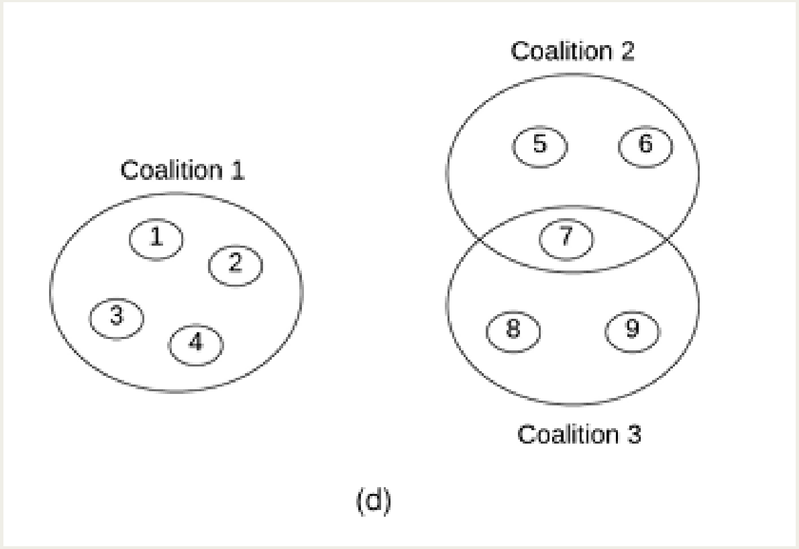
(a)



(b)

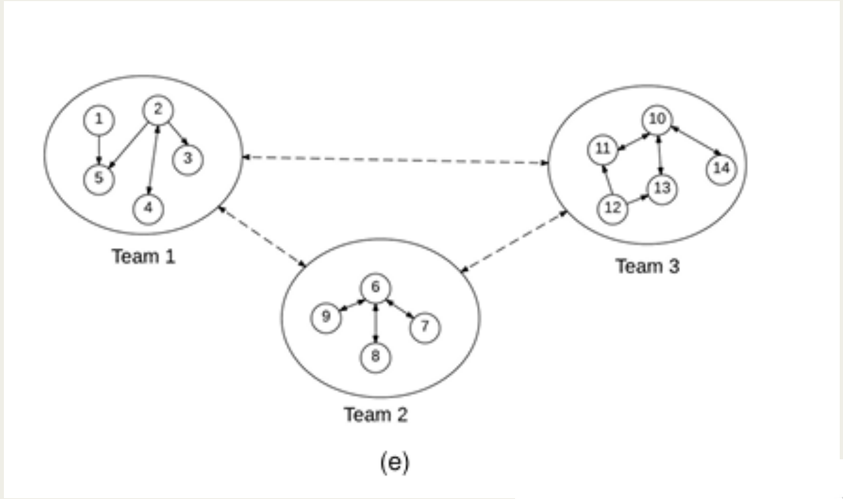


(c)

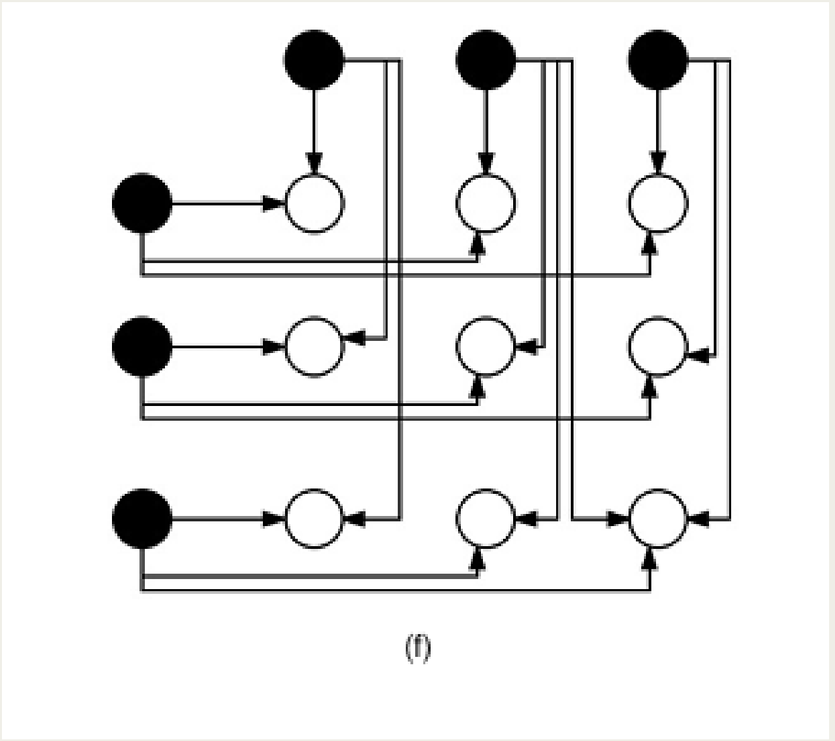


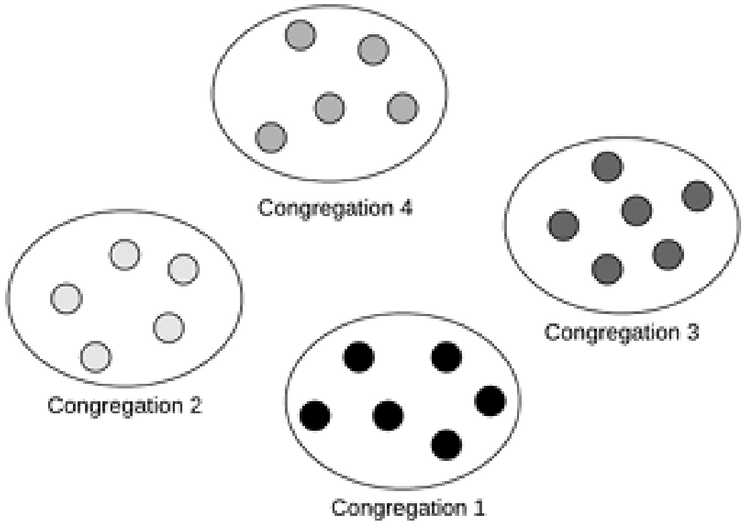
(d)

AGENT ORGANIZATION



(e)

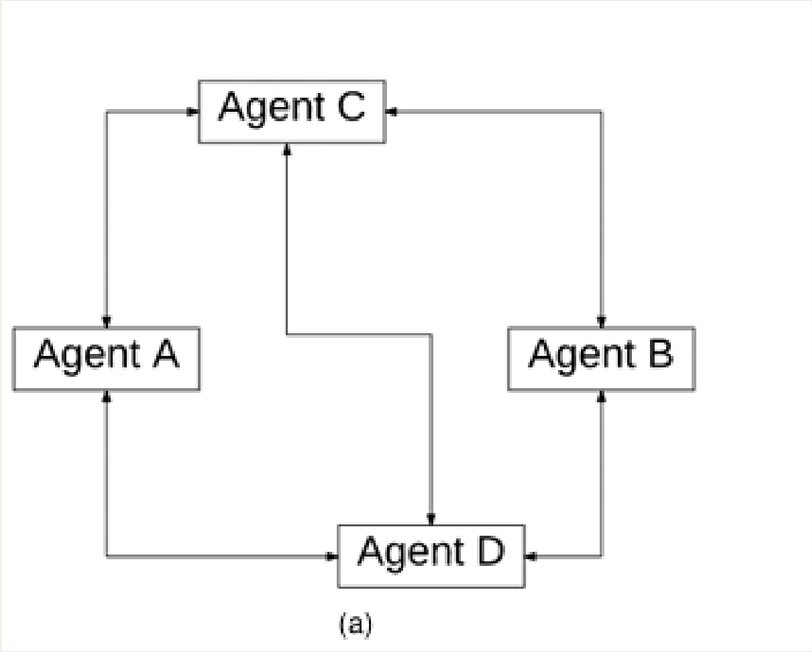

(f)

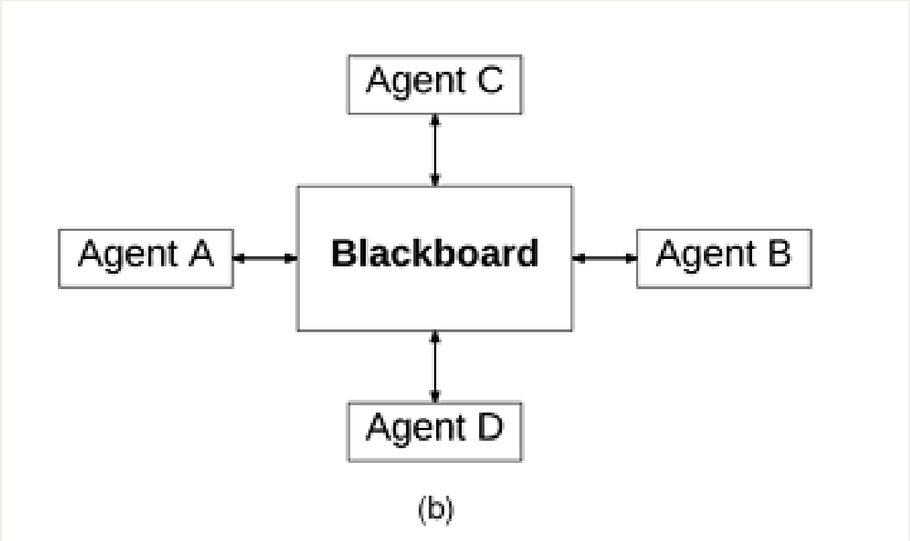


(g)

AGENT COMMUNICATION

- Speech Act: Agents use structured linguistic expressions to influence each other's beliefs and actions.
- Message Passing: Agents directly exchange messages via point-to-point or broadcast communication.
- **Blackboard System:** Agents store and retrieve shared data on a common repository controlled by a knowledge-based access system.





SINGLE AGENT SYSTEMS POWERED LLMS

- Single-agent systems powered by Large Language Models (LLMs) demonstrate autonomous decision-making, tool-use capabilities, and memory retention for improved problem-solving. These agents utilize in-context learning and external knowledge bases to enhance their decision-making process. They can break down complex tasks into smaller subgoals, utilize external resources efficiently, and maintain coherence across interactions.

- **Single -Agent VS. Multi-Agent Systems**
- Single-agent LLMs operate independently, focusing on internal mechanisms and environment interaction. In contrast, Multi-Agent Systems (MAS) leverage multiple autonomous LLM-based agents that specialize in different capabilities and engage in collaborative decision-making. This cooperative structure allows for dynamic problem-solving, where agents communicate, exchange knowledge, and specialize in distinct tasks.

AGENT PROFILING

- Agents in LLM-MAS are characterized based on their roles, traits, and expertise. Profiling methods include:
- Pre-defined: Agents have fixed roles determined by designers.
- Model-Generated: LLMs autonomously define agent roles based on system objectives.
- Data-Derived: Agent roles and behaviors are generated from datasets or prior interactions.
- For instance, in a software development environment, agents may assume roles such as product managers, engineers, or testers, each with defined expertise and responsibilities.

AGENTS COMMUNICATION

- LLM-MAS rely on structured communication paradigms to facilitate interactions among agents. Common paradigms include:
- Cooperative: Agents work together toward a shared objective.
- Debate: Agents engage in argumentation to refine solutions.
- Competitive: Agents operate with conflicting goals, fostering adversarial interactions.

- Communication structures vary:
- Layered: Hierarchical communication between different levels of agents.
- Decentralized: Peer-to-peer interactions among autonomous agents.
- Centralized: A central agent or group coordinates system-wide communication.
- Shared Message Pool: A repository where agents publish and retrieve relevant messages.

COMMUNICATION STRUCTURES VARY

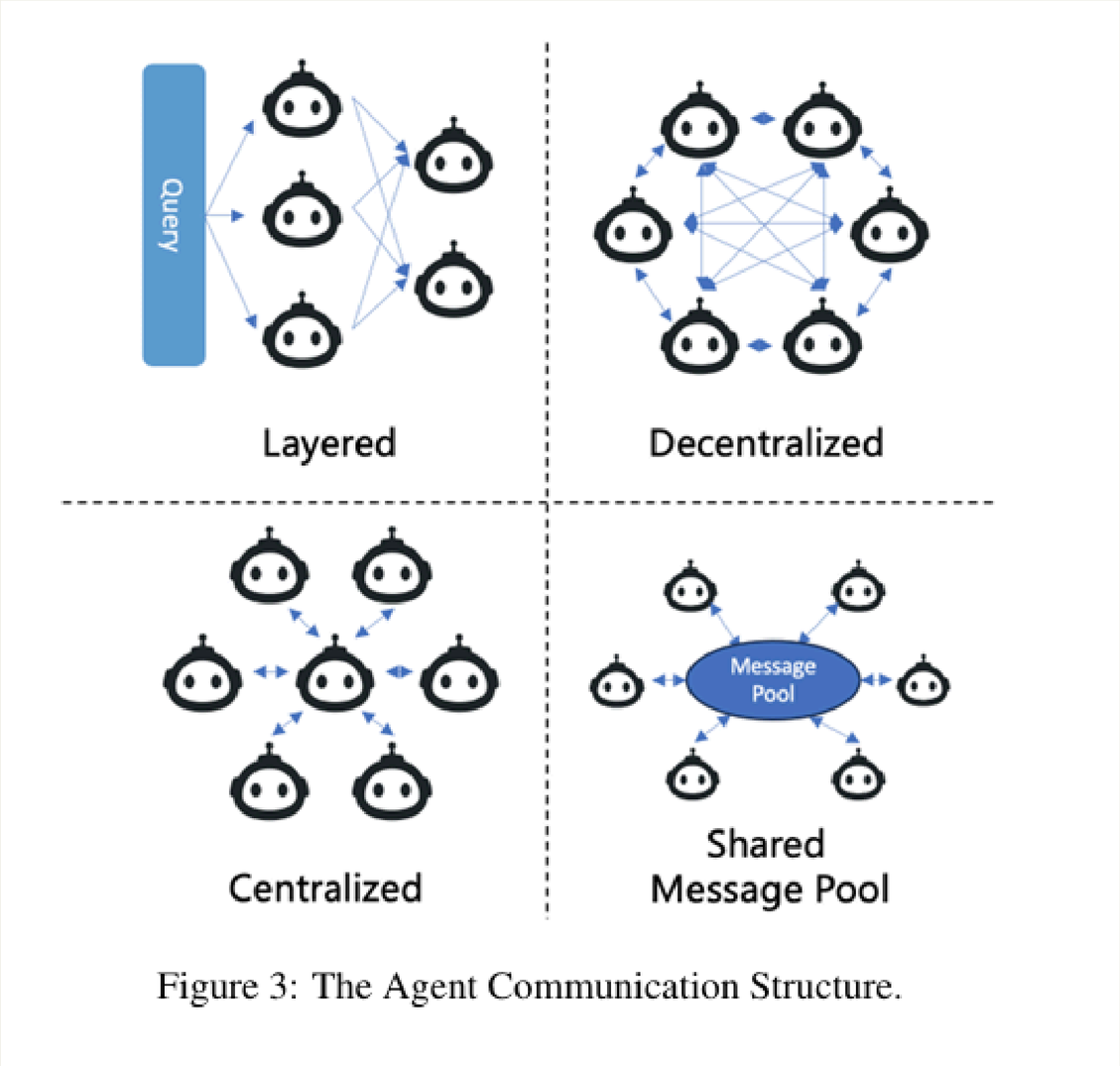


Figure 3: The Agent Communication Structure.

AGENTS CAPABILITIES ACQUISITION

- LLM-MAS agents acquire capabilities through various feedback mechanisms:
- Environment Feedback: Real or simulated environments provide responses to agent actions.
- Agent Interaction Feedback: Agents evaluate and refine each other’s contributions.
- Human Feedback: External input from human users ensures alignment with desired objectives.

- Adjustments to complex problems are managed through:
- Memory: Retaining past interactions for improved decision-making.
- Self-Evolution: Agents dynamically refine their strategies based on performance.
- Dynamic Generation: New agents are introduced to adapt to evolving challenges.

MULTI-AGENTS FRAMEWORK

- Several frameworks support LLM-based MAS development:
- MetaGPT: Implements human workflow methodologies into agent interactions.
- CAMEL: Focuses on autonomous agent cooperation using inception prompting.
- AutoGen: Customizable framework allowing developers to program interactions in natural language and code.

- **Scaling Up LLM-MA Systems**
- The scalability of LLM-MAS faces challenges in computational complexity, coordination, and resource management. Efficient agent orchestration is required to optimize workflows, assign tasks, and manage communication patterns. As systems expand, advanced methodologies are needed to ensure efficiency, resource optimization, and coordinated task execution.

LANGGRAPH FEATURE SET AND PRIMITIVES

- Nodes (Execution Units): Nodes process information and perform computations.
- They can represent functions, agents, or external tools.

- Edges (Connections): Edges direct data flow between nodes.
- They can be conditional (decision-based) or unconditional (direct transitions).

- State (Memory & Context): Tracks and stores information across executions.
- Can be global (shared across agents) or local (specific to one node/agent).

- Cyclic Execution (Loops & Feedback): Enables iterative workflows where agents evaluate, refine, and retry steps.

- Entry Points: Define where execution begins in a graph.
- Multiple entry points allow dynamic flow control.

TOOL CALLING AND MEMORY

- Function Calling (with OpenAI or Custom LLMs): The LLM manages the process by calling the appropriate functions.
- Self-Reflection / Self-Ask: The agent asks itself questions to refine its responses and improve accuracy.
- Structured Tool Calling (with LangGraph support): Tools are organized within a structured workflow for better coordination.

- Agent-Specific Memory: Each agent maintains its own independent memory.
- Shared Memory: All agents can access the same memory, preventing context loss.
- Retrieval-Augmented Memory: Past conversations and information are retrieved dynamically to improve responses.

Paper 2

AGENTIC AI: AUTONOMOUS INTELLIGENCE FOR COMPLEX GOALS — A COMPREHENSIVE SURVEY

- **What is Agentic AI?**
    - Unlike traditional AI, they are artificial intelligence systems that can make independent decisions, set and achieve goals.
    - They can perform complex tasks without human intervention and adapt to changing conditions.
- **Why is it important?**
    - It enables AI systems to transform from passive and task-based to active and strategic decision-making systems.
    - It offers more efficient solutions by reducing human error in sectors such as health, finance and manufacturing.
    - It provides great advantages in areas that require rapid decision-making, such as crisis management.

AGENTIC AI: AUTONOMOUS INTELLIGENCE FOR COMPLEX GOALS — A COMPREHENSIVE SURVEY

- **Key Features of Agentic AI**

    - Autonomy: It can make decisions on its own and act towards a goal.
    - Adaptability: It can change its strategy according to environmental conditions and new data.
    - Decision-Making: It can analyze complex situations without human guidance and determine the best solution.
    - Multi-Tasking: It can follow and prioritize multiple goals at the same time.

- **Application Areas**
  - Healthcare
    - Disease diagnosis and patient monitoring
    - Personalized treatment plans
  - Finance
    - Algorithmic trading and risk management
    - Fraud detection
  - Manufacturing
    - Smart maintenance and production line optimization
  - Education
    - Personalized learning experiences and intelligent teacher assistants

AGENTIC AI: AUTONOMOUS INTELLIGENCE FOR COMPLEX GOALS — A COMPREHENSIVE SURVEY

Feature	Traditional AI	Agentic AI
Rule-based	Yes (works with rules)	No
Decision making	Human guided	Can make autonomous decisions
Adaptability	Low	High (can be adaptive to real time)

- **Challenges and Ethical Issues**
  - Technical Challenges
    - Goal alignment problems
    - Resource constraints (computational power, energy efficiency)
    - Scalability issues (applicability to large systems)
  - Ethical and Social Issues
    - Lack of transparency (difficult to understand how decisions are made)
    - Bias and fairness issues
    - Privacy and data security concerns

- **Conclusion and Future Perspective**
    - Agentic AI differs from traditional AI systems with its autonomous decision-making and adaptation capabilities.
    - Human-AI collaboration will increase, and AI will play a supporting role.
    - Ethics and regulations will improve, and transparency and security will be increased.
    - With smarter learning models, AI systems will work more efficiently with less data.
    - Important Research Areas:
        - Independent decision-making
        - Scalability
        - Data privacy and cybersecurity

Thank you!

Gülşen Sabak & Süleyman Emir Taşan
Bogazici University Computer Engineering