Multi-agent System Part-1

Prepared by Ahnaf Hossain Rafi

Definition

Multi-agent system is a system of specialized agents powered by LLMs to collaborate on complex tasks, improving modularity, scalability, and robustness. Each agent can have its own prompt, LLM, tools, and other custom code to best collaborate with the other agents.

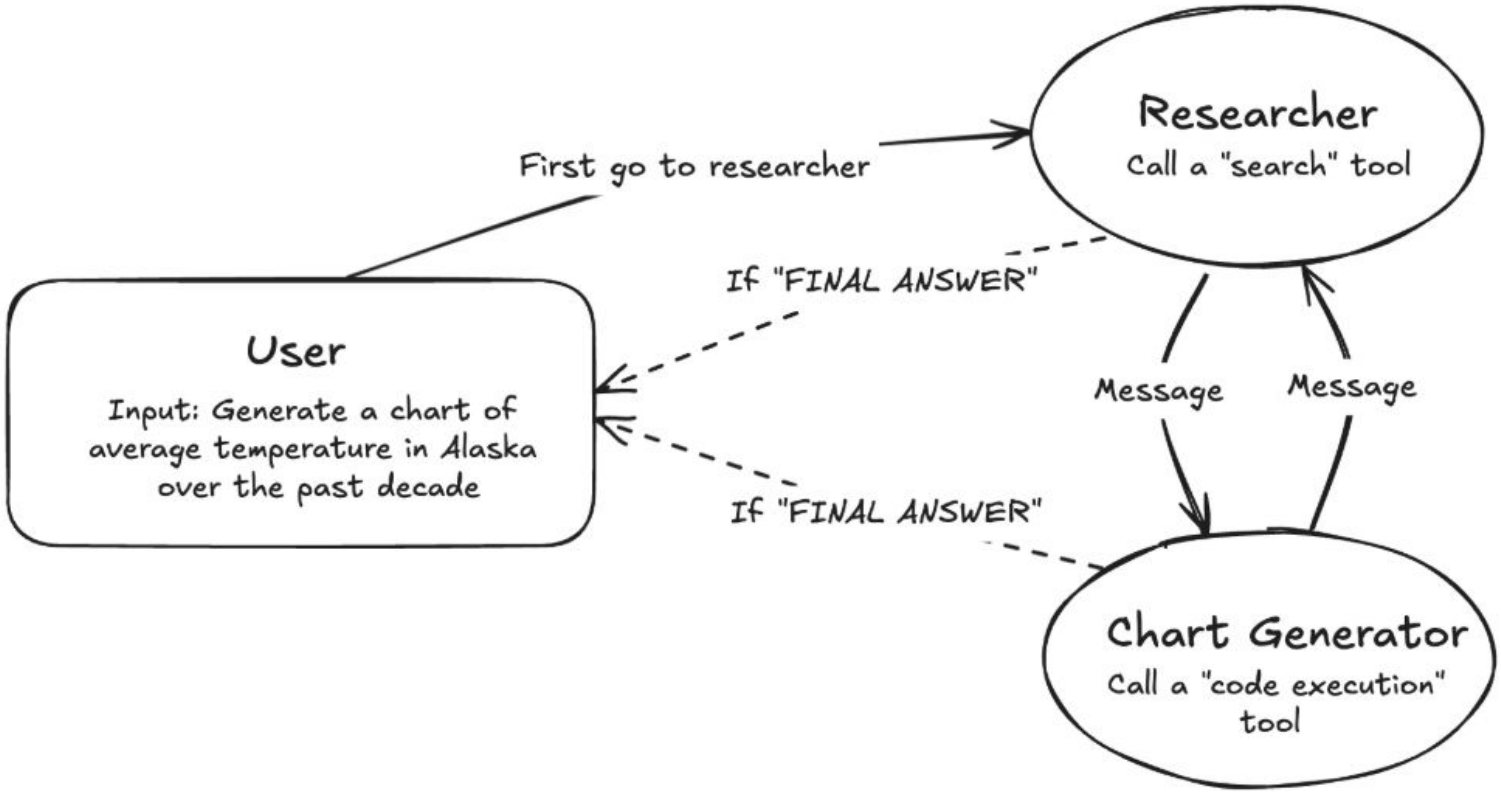
Benefits

- Grouping tools/responsibilities can give better results. An agent is more likely to succeed on a focused task than if it has to select from dozens of tools.
- Separate prompts can give better results. Each prompt can have its own instructions and few-shot examples. Each agent could even be powered by a separate fine-tuned LLM!
- Helpful conceptual model to develop. You can evaluate and improve each agent individually without breaking the larger application.

Multi-agent Collaboration

Agents work together using a shared scratchpad of messages. Everything one agent writes is visible to the others. The advantage is that each agent can see every step taken. The drawback is that this can become too detailed and noisy, especially when only the final answer is needed. We call this collaboration because of the shared nature the scratchpad.

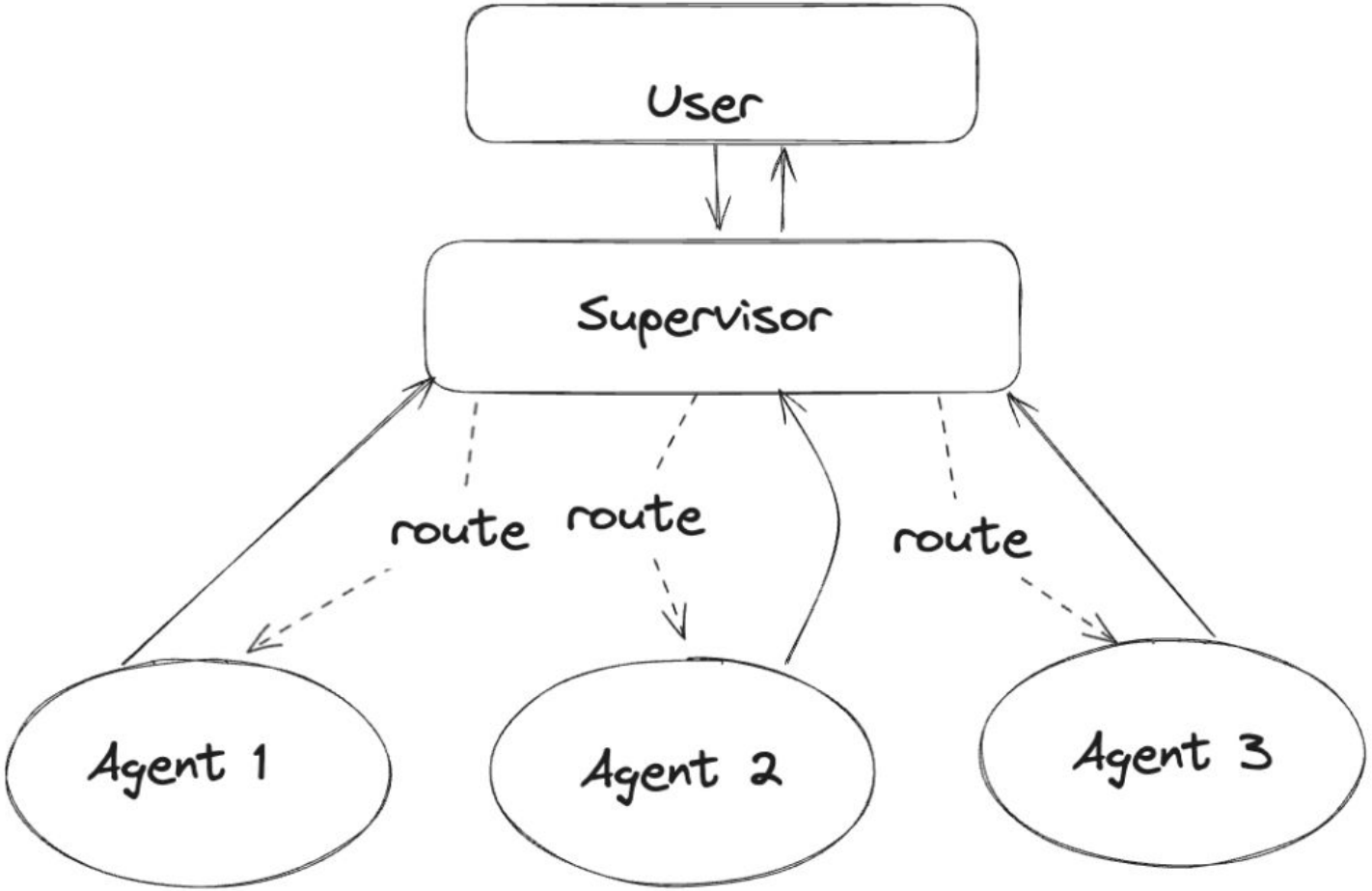
One specific prompt template and a single LLM call.


Researcher
Call a "search" tool
First go to researcher
If "FINAL ANSWER"
User
Input: Generate a chart of average temperature in Alaska over the past decade
Message
Message
If "FINAL ANSWER"
Chart Generator
Call a "code execution" tool

Agent Supervisor

Multiple agents are connected, but do NOT share a shared scratchpad. Rather, they have their own independent scratchpads, and then their final responses are appended to a global scratchpad.

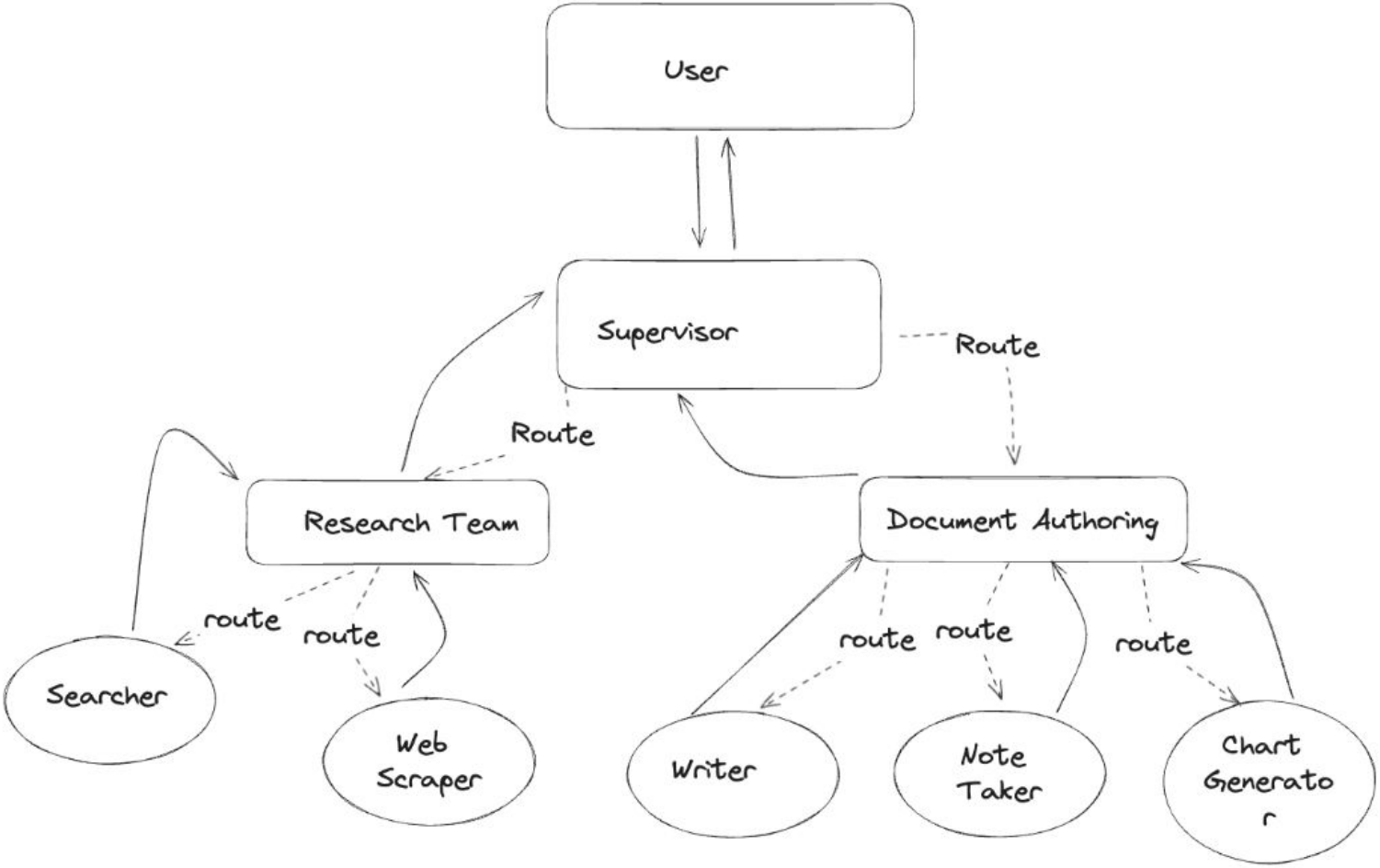
The independent agents are LangChain agents. This means they have their own individual prompt, LLM, and tools.


User
Supervisor
route
route
route
Agent 1
Agent 2
Agent 3

Hierarchical Agent Teams

Same as before but the agents in the nodes are actually other langgraph objects themselves. This provides even more flexibility than using LangChain AgentExecutor as the agent runtime. We call this hierarchical teams because the subagents can in a way be thought of as teams. A supervisor agent connects them.

Think of it like a company with a manager, team leads, and team members.


User
Supervisor
Route
Route
Research Team
Document Authoring
route
route
route
route
route
Searcher
Web Scraper
Writer
Note Taker
Chart Generato r

Example

User’s request:
“Give me a summary of AI agents and include a chart showing their growth over time.”

Step-by-Step Message Passing
1. User → Supervisor

User sends the request.

User → Supervisor:
“Give me a summary of AI agents and a chart of their growth.”

2. Supervisor → Research Team

Supervisor analyzes the request:

Needs information → send to Research Team

Needs a chart → handled later by Document Authoring

Supervisor → Research Team:
“Find information about AI agent history and growth.”

3. Research Team → Searcher/Web Scraper
Research Team decides it needs external data → routes internally.

Research Team → Searcher:
“Search for articles on AI agent growth.”

Searcher calls a search tool, gets results.
Searcher → Research Team:
“Here are links and summaries.”

Research Team might then tell Web Scraper:
Research Team → Web Scraper:
“Scrape key numbers from this site.”

Web Scraper returns structured data.

4. Research Team → Supervisor

Now the team has all raw research.
Research Team → Supervisor:
“Here’s the collected information and growth data.”

5. Supervisor takes this output and routes it to Document Authoring

This is the step you asked about.

Supervisor → Document Authoring:
“Turn this research into a summary and create a growth chart.”

Supervisor also sends the research output as an attachment/message.

6. Document Authoring → Internal Sub-agents
6.1 Writer for the summary

Document Authoring → Writer:
“Write a concise summary using this research.”

Writer returns a draft.

6.2 Chart Generator for the chart

Document Authoring → Chart Generator:
“Generate a chart using these growth numbers.”

Chart Generator calls a plotting tool → returns PNG/SVG.

6.3 Note Taker if needed
(Only if Supervisor asked for structured notes, not in this example.)

7. Document Authoring → Supervisor

Document Authoring collects all sub-results:

Final written summary
Generated chart

Then sends them upward.

Document Authoring → Supervisor:
“Here is the final content + chart.”

8. Supervisor → User

Supervisor assembles the final response.

Supervisor → User:
“Simplified summary + chart here you go!”

Done!