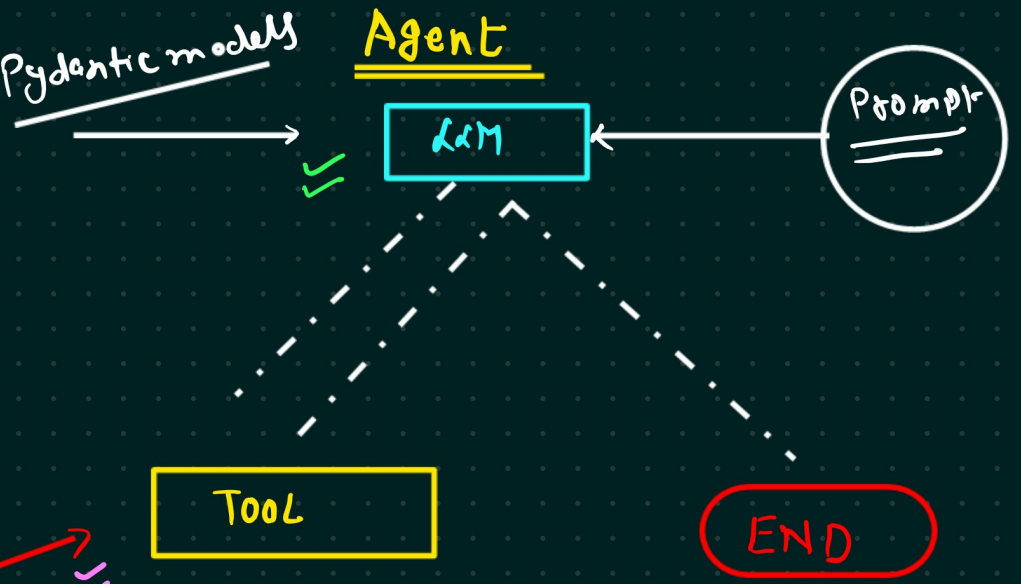
Agent ⇒ LLM + tools

LLM → brain

tools → Action



MultiAgent

Multi + Agent

many

→ More than one Agent

→ Multiple Agent is talking to each other

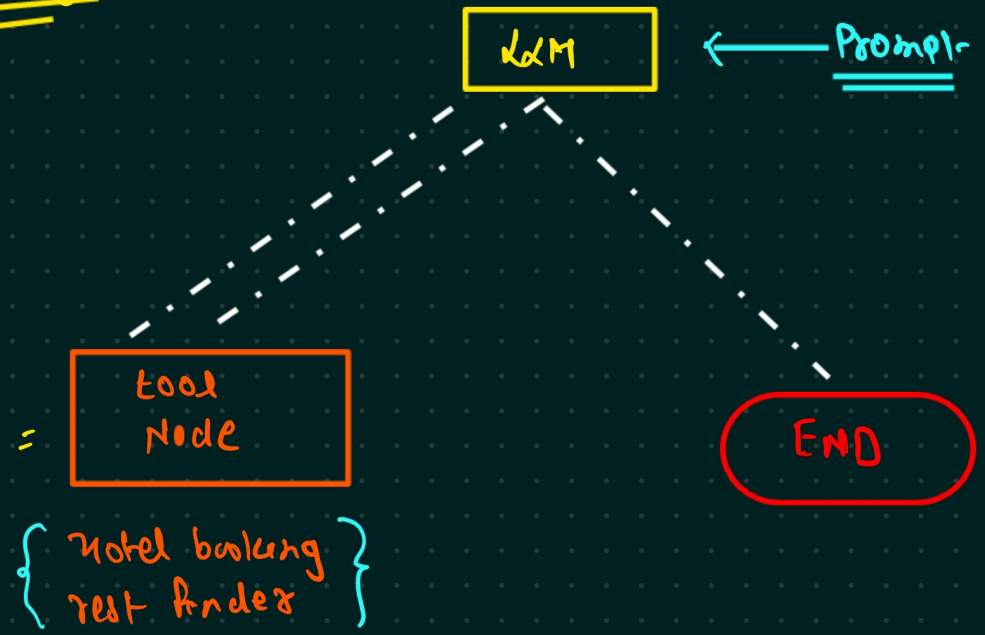
1. Network
2. Supervisor → Hierarchical Pattern

= Travel planner ⟶ = Hotel booking ✓

= Best restaurant ✓

Simul tool calling

(Agent)



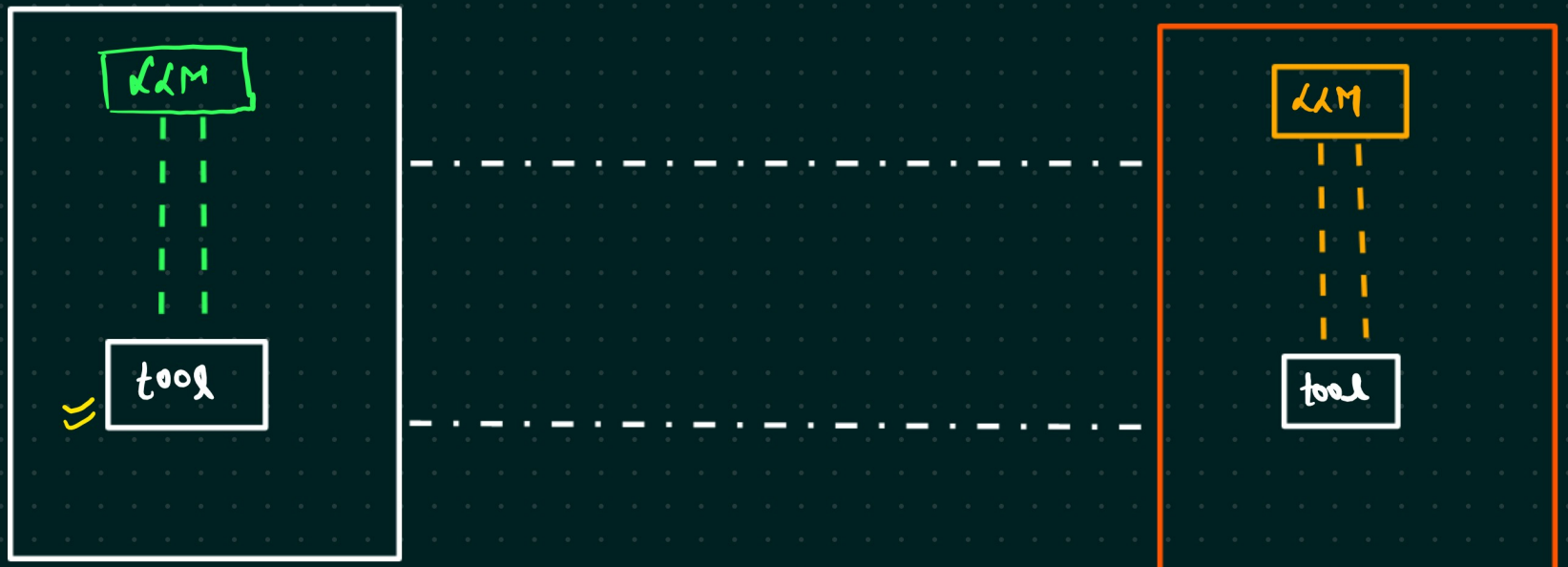
MultiAgent ⇒ travel Planner

1. Hotel booking
2. restaurant finder

2 Agent ⇒ N no. of Agent

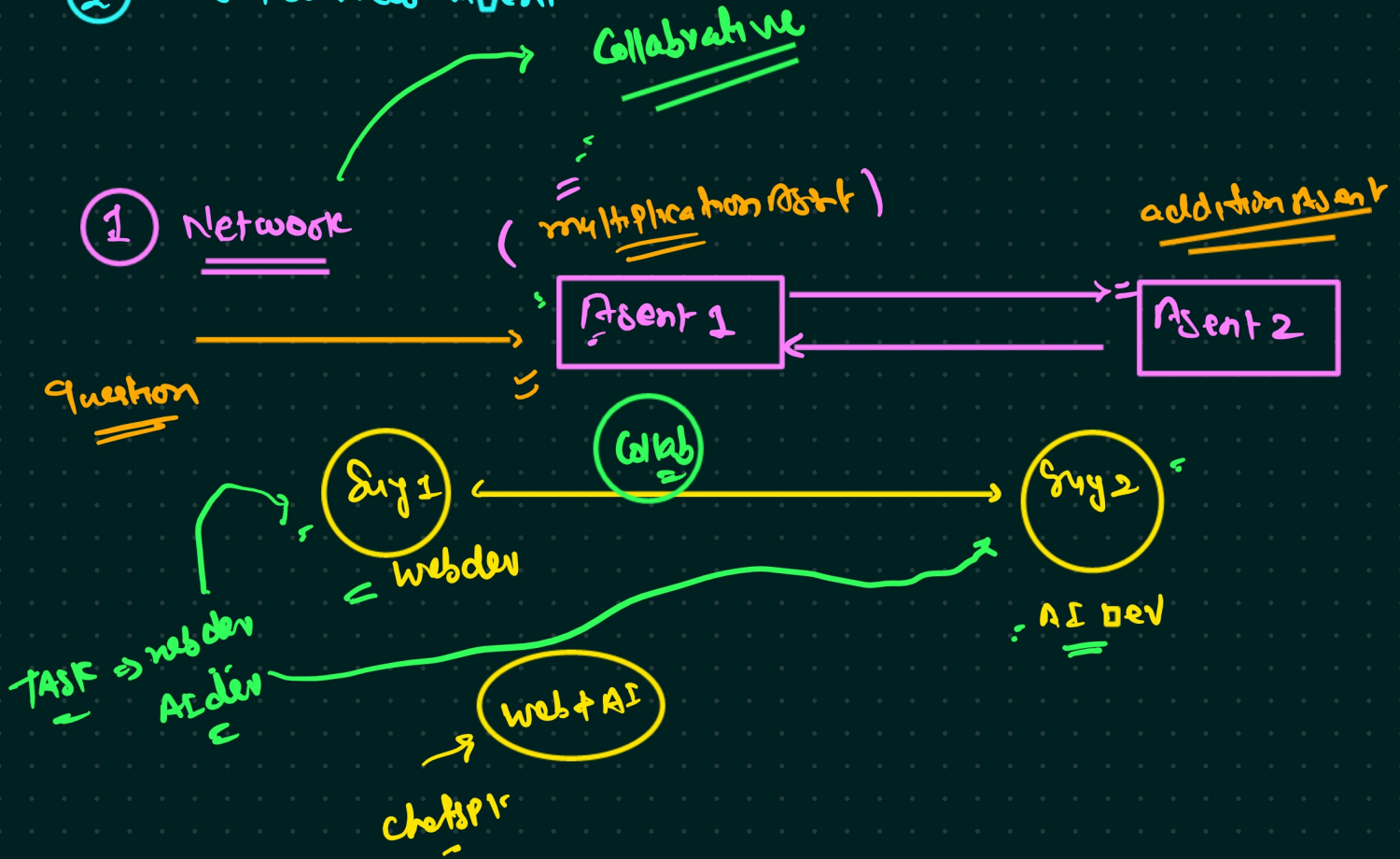
Hotel booking Agent

Restaurant Finder Agent

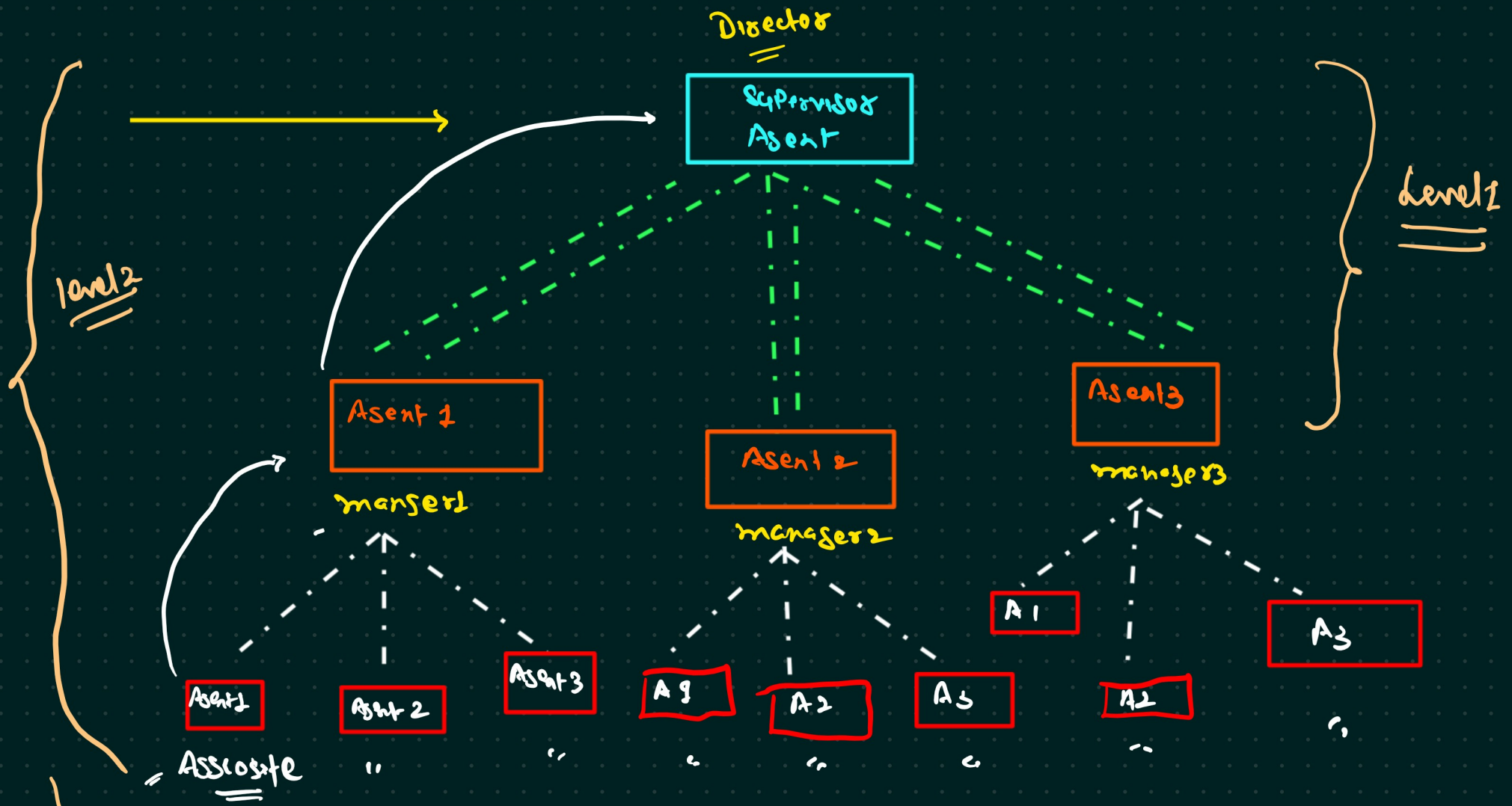


MultiAgent (2 Agent)

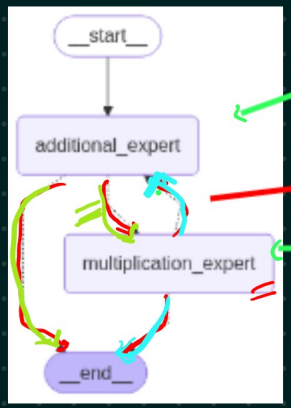
1. Network multi agent ✓
2. Supervisor agent ✓



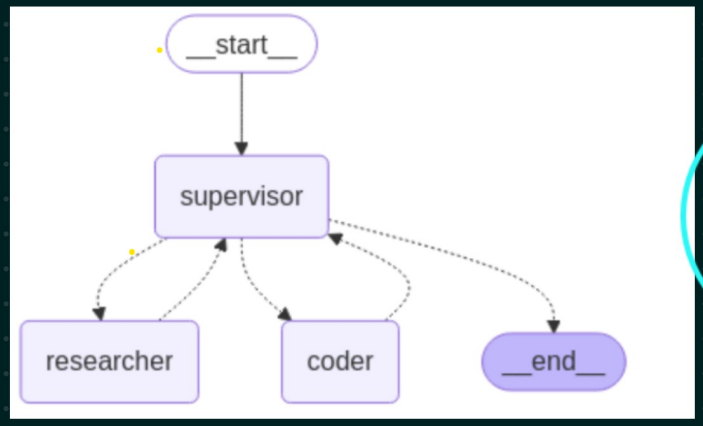
(2) Supervisor multi agent

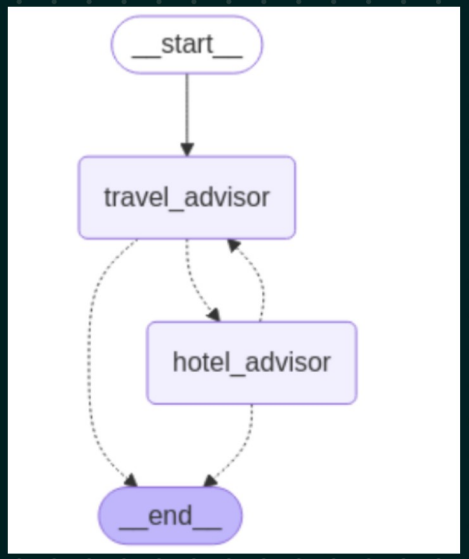


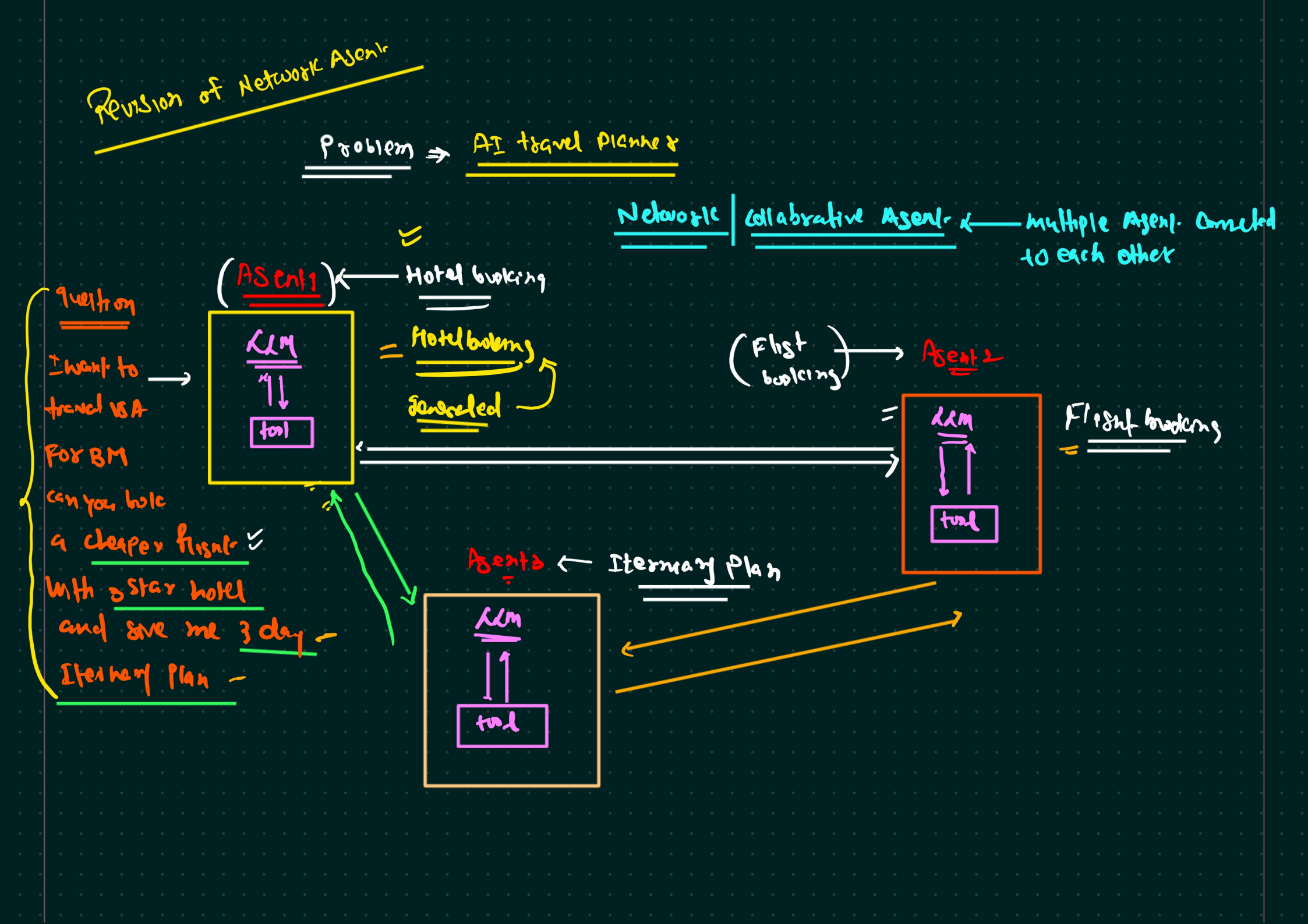
(3) Hierarchical Agent ← extention of supervisor Agent




```python
def additional_expert(state:MessagesState)-> Command[Literal["multiplication_expert", "__end__"]]:
    system_prompt = (
        "You are an addition expert, you can ask the multiplication expert for help with multiplication."
        "Always do your portion of calculation before the handoff."
    )

    messages = [{"role": "system", "content": system_prompt}] + state["messages"]

    ai_msg = llm.bind_tools([transfer_to_multiplication_expert]).invoke(messages)

    if len(ai_msg.tool_calls) > 0:
        tool_call_id = ai_msg.tool_calls[-1]["id"]
        tool_msg = {
            "role": "tool",
            "content": "Successfully transferred",
            "tool_call_id": tool_call_id,
        }

        return Command(
            goto="multiplication_expert", update={"messages": [ai_msg, tool_msg]}
        )
    return {"messages": [ai_msg]}
```






Revision of Network Agent
Problem ⇒ AI travel planner
Network | Collabrative Agent ← multiple agent connected to each other
(Agent1) ← Hotel booking
question
I want to travel USA
For BM
can you book
a cheaper flight
with 3 star hotel
and save me 3 day -
Iternary plan -
LLM
tool
Hotel booking
Cancelled
(Flight booking) → Agent2
LLM
tool
Flight booking
Agent3 ← Iternary Plan
LLM
tool

```python
##Agent2
def multiplication_expert(state:MessagesState)-> Command[Literal["additional_expert", "__end__"]]:

    system_prompt = (
        "You are a multiplication expert, you can ask an addition expert for help with addition. "
        "Always do your portion of calculation before the handoff."
    )

    messages = [{"role": "system", "content": system_prompt}] + state["messages"]

    ai_msg = llm.bind_tools([transfer_to_addition_expert]).invoke(messages)

    if len(ai_msg.tool_calls) > 0:
        tool_call_id = ai_msg.tool_calls[-1]["id"]
        tool_msg = {
            "role": "tool",
            "content": "Successfully transferred",
            "tool_call_id": tool_call_id,
        }
        return Command(goto="additional_expert", update={"messages": [ai_msg, tool_msg]})
    return {"messages": [ai_msg]}
```

addition_tool → description
multiply_tool → description

Real time

tool → Calculation

metadata