Creator's Algorithm - Build your own Solution now

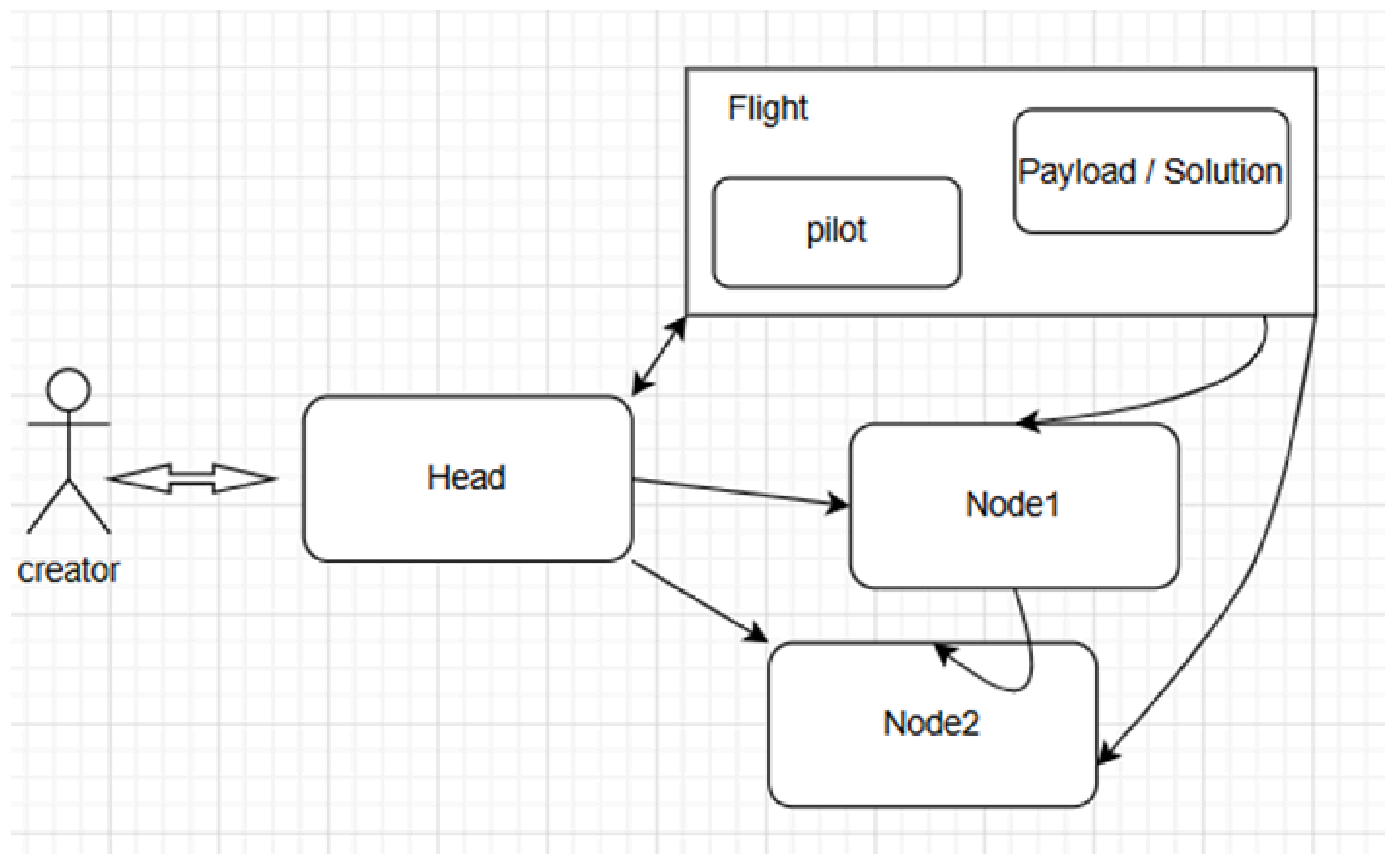
Bhagavan Bejjipurapu
Computer Science, School of Science and Engineering

Context and Motivation

It is a design pattern that enables the creation of new algorithms by leveraging the structure and principles of this algorithm, allowing for recursive algorithm design and adaptation. We have numerous solutions for every problem, each being intelligent in its own way. Yet, we still seek a perfect artificial intelligence—at least, most of the world is trying to build one nowadays. I am presenting an algorithm that aims to create a perfect intelligence with a single objective: determining the most ideal solution for a given problem.
At first, you may feel like you've encountered this idea before and wonder whether it truly solves anything, but trust me, it will. This solution applies to any problem in the world, but for now, our focus is AI. We all know that, despite advancements in AI, we are not yet achieving 100% accuracy in our results. However, with a systematic approach, we can create a truly effective solution.

Theory

If a program can solve a specific problem, then a collection of programs can solve any problem by executing them sequentially. For example, if I want to create a humanoid robot that talks like a human and responds with human-like emotions, I need to build a robust CNN model to process images. The model would first analyze the image, send the details to an emotion detector, which identifies the emotions, and finally, the system would respond to the user with appropriate emotional expressions.
This approach is not limited to robotics—it can also be applied to breaking down complex problems into smaller, manageable parts. For instance, understanding an image involves multiple steps, such as segmenting the image, identifying the speaker, and detecting relevant features.
To solve problems systematically, this algorithm provides a structured solution. It consists of four key components: the Head Node, a Set of Nodes, a Flight, and an Actor.



Architecture of creators Algorithm

Steps and Methodology

1. Preprocessing
First, we must define the problem we aim to solve and establish the criteria for an acceptable solution. This helps determine when to stop the process.

2. Nodes
Each node in the system is classified as either a Good Node or a Learning Node:
- A Good Node can complete its assigned task and return results to the Flight. Based on the results, the Flight can call another node with modified parameters.
- A Learning Node is not yet capable of performing the assigned task independently.

For robustness, each node performs a specific type of task. If a node requires additional processing, it can delegate work to another specialized node. This allows the system to scale by integrating new sub-algorithms and dynamically creating solution trees.

3. Head Node & Flight
The Head Node receives input from the user, processes it, and transforms it into a Flight, which carries information about the problem statement. The Head Node also attaches a Pilot Condition, which guides the Flight toward a solution. The Pilot Condition functions like a landing criterion, ensuring the Flight continues until an acceptable solution is found.
Each Flight has five key properties:
1. Response Flag – Indicates whether a solution has been found.
2. Payload – Stores the current solution (which may be temporary at different stages).
3. Solution Type – Defines how to handle responses from previous nodes (e.g., additive, distributed, etc.).
4. Solution Interpretation Method – Specifies how to compare the new solution with existing solutions.
5. Landing Validation Method – Ensures the Flight stops at a valid solution.

4. Actor
The Actor initiates the process by providing input to the Head Node. The Actor also has control over the Flight, allowing them to:
- Monitor progress at any time.
- Stop the Flight if an adequate solution is found.
- Perform emergency stops if needed.
- With minimal human intervention, resource allocation and debugging become significantly easier.

Each node communicates with other nodes, continuously seeking solutions until the first Flight finds a valid answer. Essentially, you become the creator, designing solutions, defining Flight interactions, and modifying them based on your specific problem. You can enhance the system by restricting or expanding node and Flight capabilities, making it more powerful with your creativity.

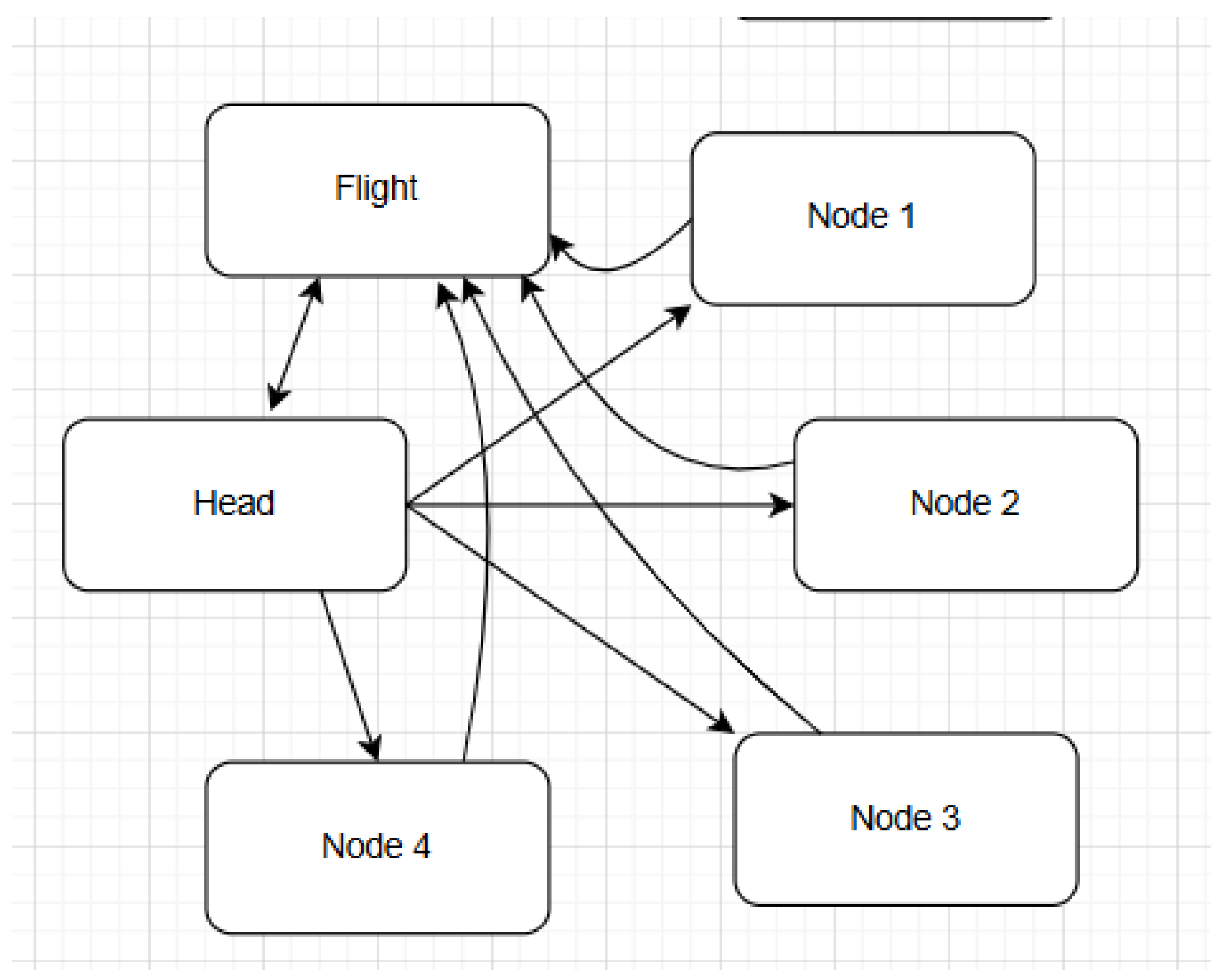
Pros & Cons of Algorithm

Pros:
- Its an effective way for handling computer resources, single point of execution, taking decision in flight.
- Can allow asynchronous executions with a heartbeat check in flights.
- This algorithm has the capability to create new intelligence. When a task is assigned to train a model one step at a time, in theory, the system should eventually be able to understand and solve a wide range of problems.
- It allows for a more detailed analysis of problems, enabling us to break them down further and strategize the next steps or possible solutions.
- The system is highly robust, as solutions can be adapted based on the limitations and capabilities of individual nodes. This ensures flexibility and scalability when solving complex issues.

Cons:
- Due to its limitless capabilities, the system requires strong security measures for each node and Flight to prevent vulnerabilities or malicious interference.
- A deep understanding of the problem is essential for effective analysis and defining the Pilot Condition, which guides the Flight towards a valid solution.
- While the solution is divided into multiple parts for efficiency, the system still has a single point of failure:
  - If the Flight is destroyed or the current node fails, the entire process is put on hold.
  - If the user is lost or disconnected from the process, it can lead to incomplete execution.
- Creating a validation condition to determine when the Flight should stop can be highly complex, especially for intricate or undefined problems.

Applications

It can solve any distributed problems like,
All the artificial intelligences like a humanoid bot or dog.
Consesnses, Atomic commit and so on in distributed environment.

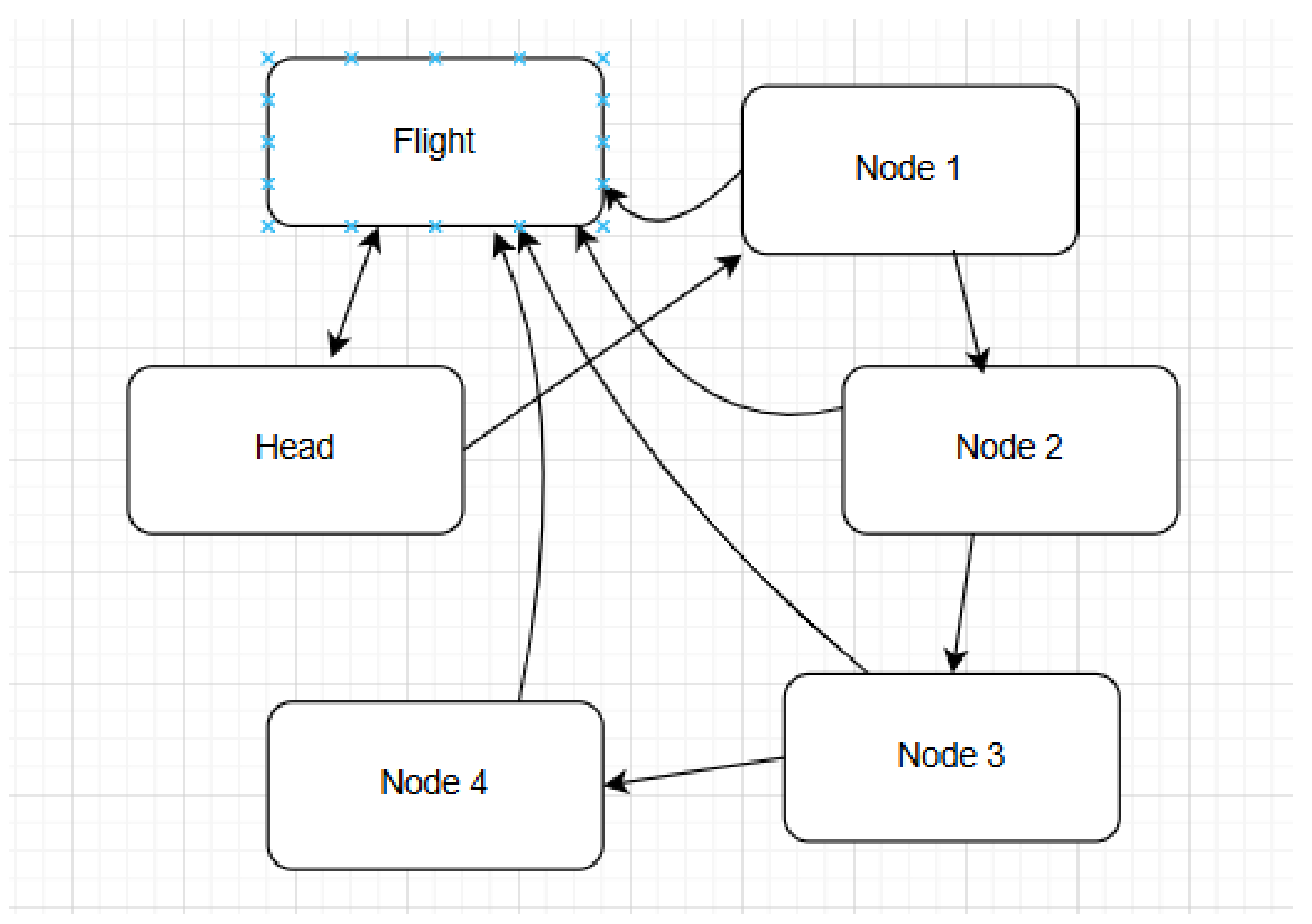


1. You are well aware of Global Consensus, a distributed problem where in a distributed system, the goal is for all nodes to agree on a single value (e.g., choosing a leader or agreeing on a decision).
Here we will modify the flight to our convinience where it is collective type. So flight waits for all the nodes to respond back to flight and then it aggregates the results back to user.

2. Its also same with Atomic commit problem in distributed computing, where it commits based on the responses from nodes

3. Additionally, it effectively solves additive-nature solutions, such as reducing the context window of execution. For example, the Head Node first receives a query from the user and sends a Flight to the next node. That node processes the query, generates a solution, and then forwards the same Flight to its child node. The child node takes the Flight, completes its assigned task, and passes it further down the chain. This process continues until the Pilot Condition is met.

4. In this way, it also addresses the recursive nature of problem-solving—if we update the Flight's landing condition after each node, ensuring progressive refinement at every step.



Conclusion

A node can be a basic program or an large LLM model this methodology achieves consensus based on the individual decisions.

These applications mean that it can achieve a common solution from a set of opinions like an intelligence. So, this algorithm provides an innovative and structured approach to finding optimal solutions even for some of the biggest challenges in the world, while requiring minimal resources (ie, because of flights). Its modular and systematic nature allows for problem-solving at various levels, from developing humanoid robots to tackling highly complex AI-driven tasks.