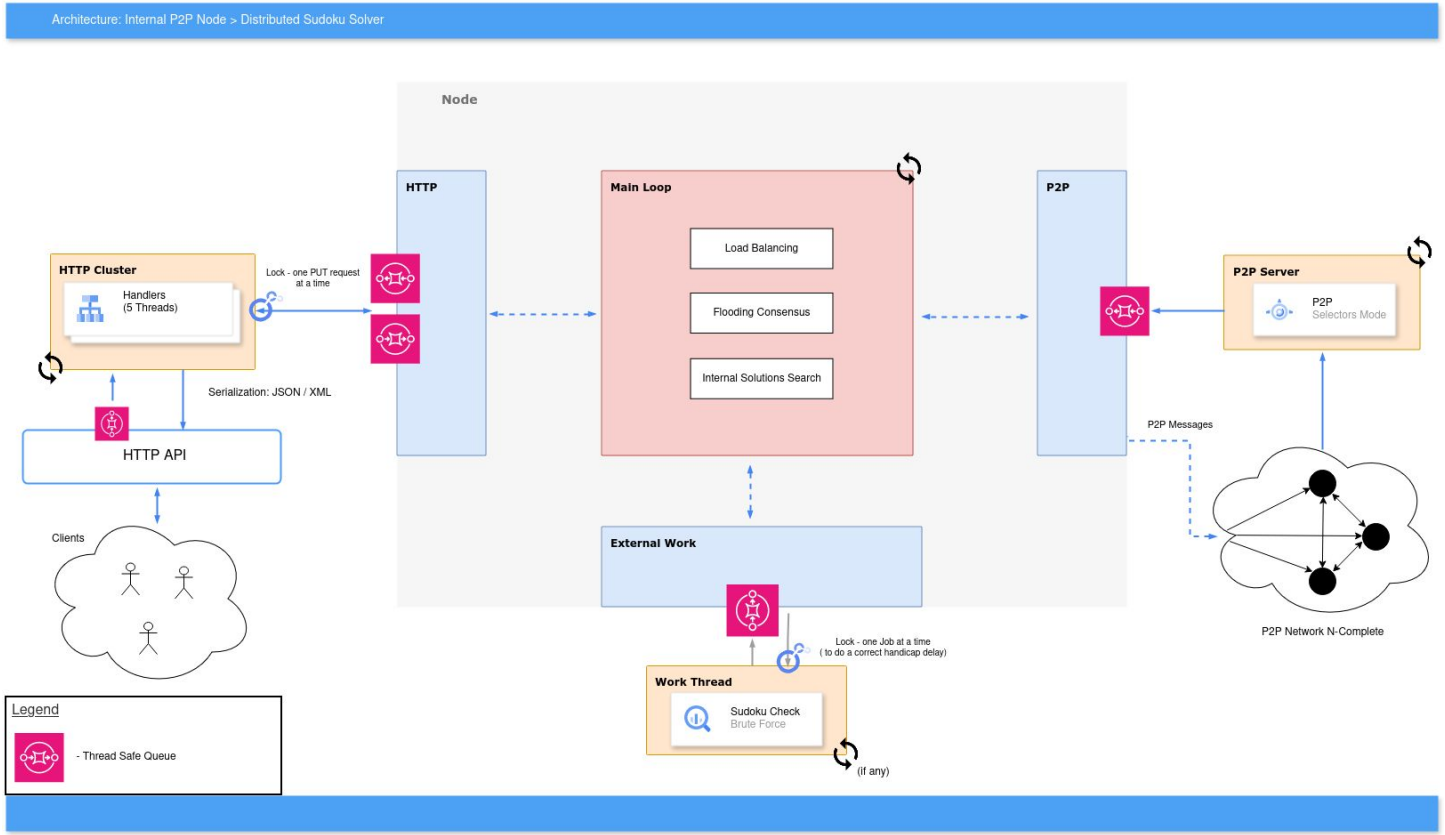
deti

CD – Final Project Report

Distributed Sudoku Solver

Group:
Pedro Pinto - 115304
João Pinto - 104384

Architecture



Network Protocol

`FLOODING_HELLO` -> Alive nodes

```
{
    "command": "FLOODING_HELLO",
    "replyAddress": "host:port",
    "args": {
        "aliveNodes": ["host:port"],
        "stats": {
            "all" : {
                "solved": 0, "internal_solved": 0,
                "invalid": 0,"internal_invalid": 0,
            },
            "nodes": [
                {
                    "address": "host:port",
                    "validations": 0,
                    "internal_validations": 0,
                },
                ...
            ]
        }

    }
}
```

`FLOODING_CONFIRMATION` -> Consensus nodes

```
{
    "command": "FLOODING_CONFIRMATION",
    "replyAddress": "host:port",
    "args": {
        "stats": {
            "all" : {
                "solved": 0,
                "invalid": 0,
            },
            "nodes": [
                {
                    "address": "host:port",
                    "validations": 0,
                },
                ...
            ]
        }
    }
}
```

`JOIN_REQUEST` -> Anchor node

```
{
    "command": "JOIN_REQUEST",
    "replyAddress": "host:port"
}
```

`JOIN_REPLY` -> Joining node

```
{
    "command": "JOIN_REPLY",
    "args": {
        "aliveNodes": ["host:port"]
    }
}
```

`SOLVE_REQUEST` -> Alive nodes

```
{
    "command": "SOLVE_REQUEST",
    "replyAddress": "host:port",
    "args": {
        "task_id": "sudoku_id[start-end]",
        "sudoku": sudoku
    }
}
```

`SOLVE_REPLY` -> Node that made the request

```
{
    "command": "SOLVE_REPLY",
    "replyAddress": "host:port",
    "args": {
        "task_id": task_id,
        "solution": solution
    }
}
```

`FLOODING_HELLO` - Message used by a node to introduce itself to the P2P network.

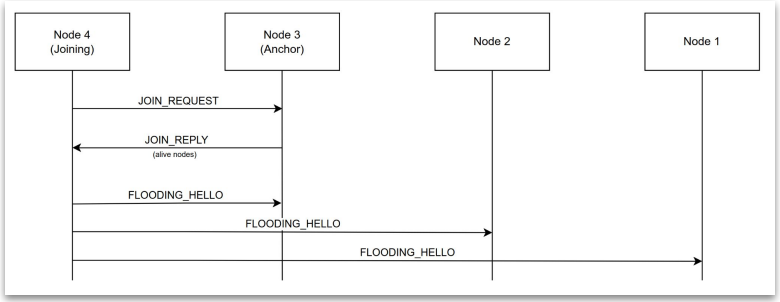
`FLOODING_CONFIRMATION` - Message used to announce that consensus has been reached.

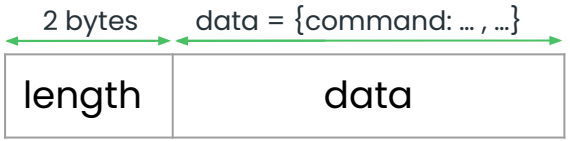
`JOIN_REQUEST` - Message sent by a node that wants to join the network.

`JOIN_REPLY` - Response to a join request, providing the list of known nodes.

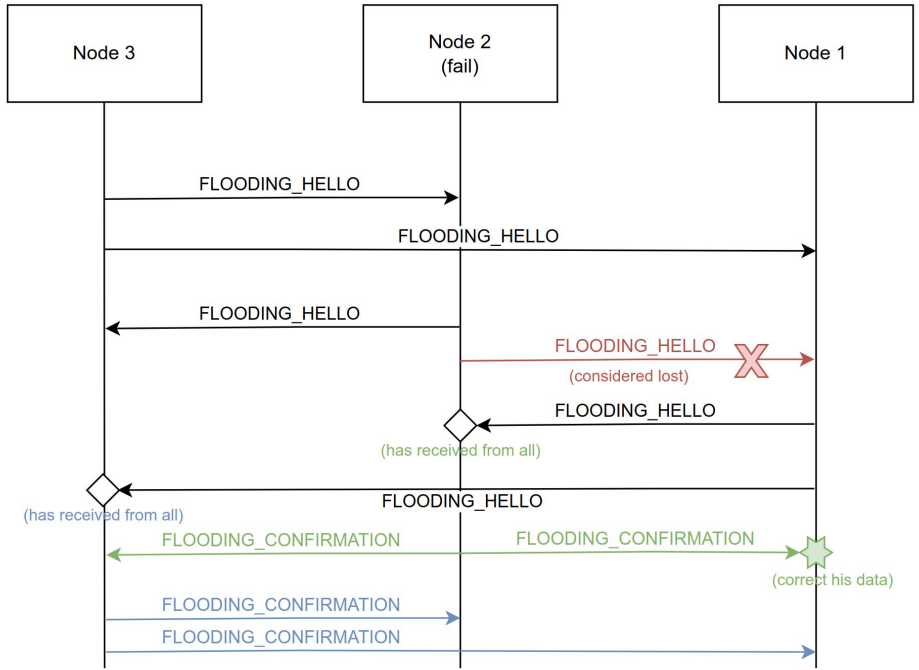
`SOLVE_REQUEST` - Request to solve a specific task.

`SOLVE_REPLY` - Response after completing a task.





Flooding Consensus Alg.



- Since the network is small and we are only **incrementing** values, such as the number of validations and sudokus solved, there are not many disadvantages to this approach.

Load Balancer

- Implementation of a load balancing system in a P2P network, including worker and task management at each node.
- Each task is associated with a worker, along with methods for managing task execution time, retry attempts, and task termination. We allow one retry attempt if a request times out. These mechanisms for handling worker failures, reassigning tasks, and verifying timeouts are performed at runtime, allowing a slow node to become a fast node.
- Selection of the best worker for a task based on the lowest response time (using Exponential Moving Average). Using EMA for response time allows the system to adapt to changes in worker performance over time. $$moving\_average = smooth\_factor \times elapsed\_time + (1 - smooth\_factor) \times moving\_average$$
- This smoothing factor is very important because it allows us to control rapid dynamic changes for specific reasons. For instance, in P2P response average time, we consider a highly dynamic approach because the number of nodes can fluctuate from 0 to 3-5 within seconds. Similarly, for controlling the current node's internal checks (when a node receives a request from a client), we apply this approach due to the same reason. In future work, we can adapt this smoothing factor in runtime to decrease for old and stable nodes and increase for less reliable nodes (only a suggestion).
- Efficiently splits Sudoku puzzles into dynamic sub-tasks, adapting task size based on worker capacity, ensuring optimal load balancing, and significantly enhancing overall resolution speed and scalability. This adjustment of load is sent to each worker according to their processing capacity, ensuring that each task takes approximately 0.75 seconds, according to the following equation: $$task\_size = int\left(task\_size \times \left(\frac{task\_size\_factor}{moving\_average}\right)\right) \qquad task\_size\_factor = 0.75$$
- Despite these benefits, it's important to consider the communication overhead involved in this system. Frequent communication between nodes for task assignment and flooding updates can increase network traffic, potentially leading to latency issues. However, by optimizing the frequency and size of these communications, the system can maintain high efficiency and minimize the impact of this overhead.
- Based on the P2P average response, we adjust the internal load of each node for solving Sudokus to ensure it remains available for both handling client requests and performing tasks. This time partitioning ensures that when a node is working alone, it maximizes its working time, but when other nodes are present, the reception of requests is not affected (it's dynamic). This approach optimizes efficiency and maintains a balanced load distribution across the network.

deti

CD – Final Project Report

Distributed Sudoku Solver

Group:
Pedro Pinto - 115304
João Pinto - 104384