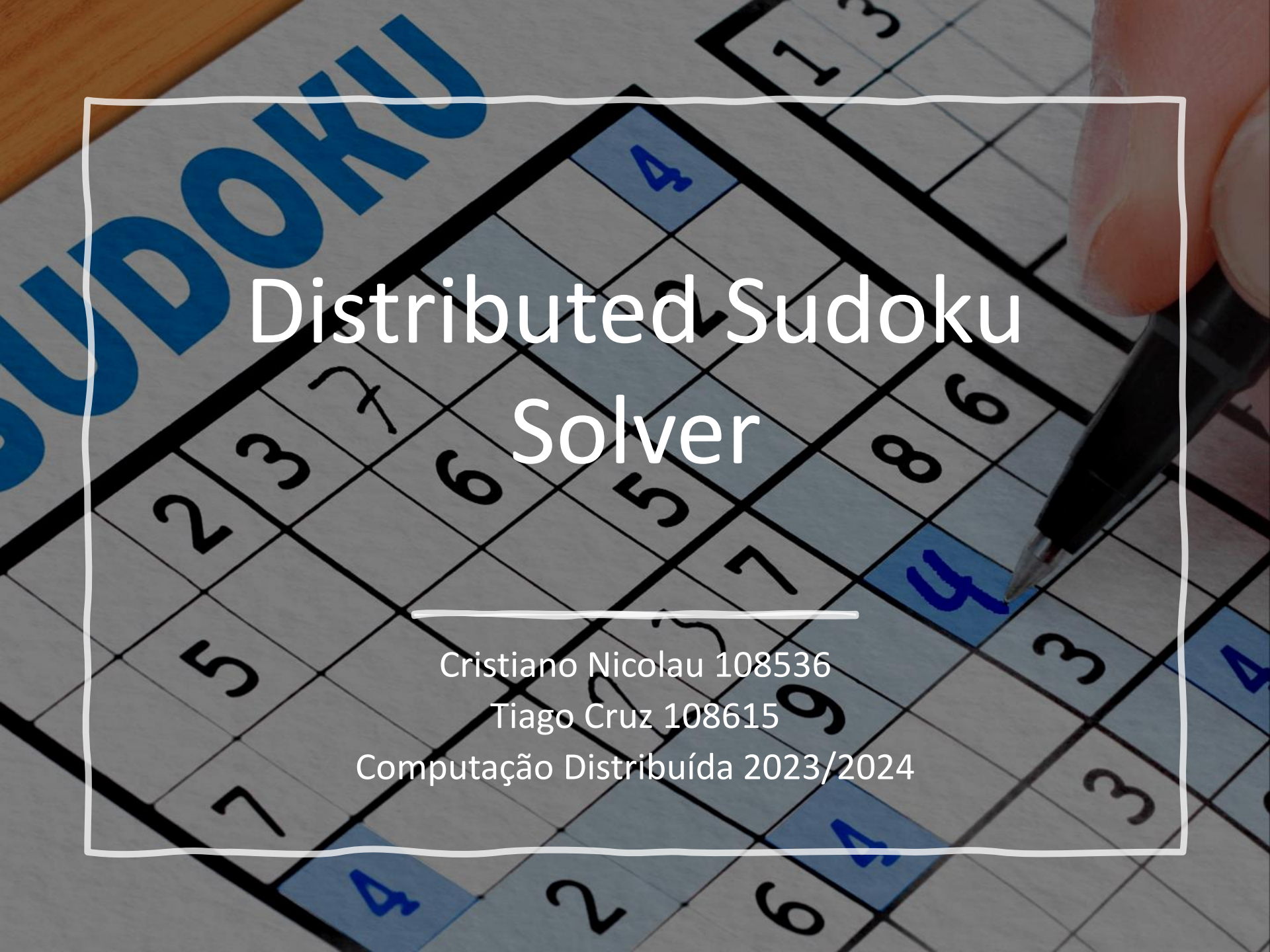

Distributed Sudoku Solver

Cristiano Nicolau 108536
Tiago Cruz 108615
Computação Distribuída 2023/2024

P2P Decentralized System

- This project is based on a decentralized peer-to-peer (P2P) architecture.
- In a P2P system, nodes communicate directly with each other without relying on a central server.
- Each node can perform tasks and share information with other nodes, making the system more resilient and scalable.
- Communication between nodes is performed using the UDP (User Datagram Protocol) to ensure low latency and efficiency in message exchange.
- When multiple peers are present, one peer takes on the role task distribution. This coordinating peer sends and receives messages from other peers while also contributing to the Sudoku solving process.

System Architecture

Our system divides the Sudoku solving task among multiple peers.

Each peer node can accept requests to solve Sudoku puzzles via its HTTP interface.

The system dynamically distributes parts of the puzzle to different peers to find the solution efficiently.

Peers communicate using a custom P2P protocol to exchange information and updates.

Node.py

The node.py script is responsible for the main functionality of each peer in the network

Key functions include:

- Initializing the node and connecting to the network.
- Handling incoming messages and requests from other nodes.
- Distributing tasks among available peers and processing solutions.
- Broadcasting updates and statistics to other peers.
- Managing the task queue and ensuring each part of the puzzle is solved.
- Handling node failures and dynamically adjusting the network.

Protocol

The custom P2P protocol in our project includes the following steps:

Connect: Nodes establish connections with each other.

Task Distribution: Sudoku solving tasks are distributed among peers.

Solution Propagation: Peers send solutions back to the originating node.

Stats Broadcast: Nodes periodically broadcast their statistics to all peers.

Failure Handling: Nodes detect and handle peer failures, redistributing tasks as needed.

Objective	Destination	Message	Response
Connect to a Node	Any Node	{"type": "connect", "address": f"{host}:{port}"}	{"type": "connected", "address": f"{host}:{port}"}
Acknowledge Connection	Any Node	{"type": "connected", "address": f"{host}:{port}"}	n.a.
Broadcast All Peers	All Peers	{"type": "all_peers", "all_peers": self.all_peers}	n.a.
Disconnect from a Node	Any Node	{"type": "disconnect", "address": f"{host}:{port}"}	n.a.
Request Sudoku Solution	Any Node	{"type": "solve", "sudoku": self.sudoku, "row": i, "col": j, "address": self.id}	{"type": "solution", "sudoku": self.resolving_sudoku, "col": self.resolving_peer[1], "row": self.resolving_peer[0], "solution": num_solution, "address": self.id}
Provide Sudoku Solution	Any Node	{"type": "solution", "sudoku": self.resolving_sudoku, "col": self.resolving_peer[1], "row": self.resolving_peer[0], "solution": num_solution, "address": self.id}	n.a.
Broadcast Stats	All Peers	{"type": "stats", "origin": self.id, "solved": self.solver.solved_puzzles, "stats": {"address": self.id, "validations": self.solver.validations}, "all_stats": self.all_stats}	n.a.
Fetch Stats via HTTP	HTTP Server	GET /stats	Aggregated Stats in JSON format
Fetch Network Info via HTTP	HTTP Server	GET /network	Network Info in JSON format

Message sequence chart