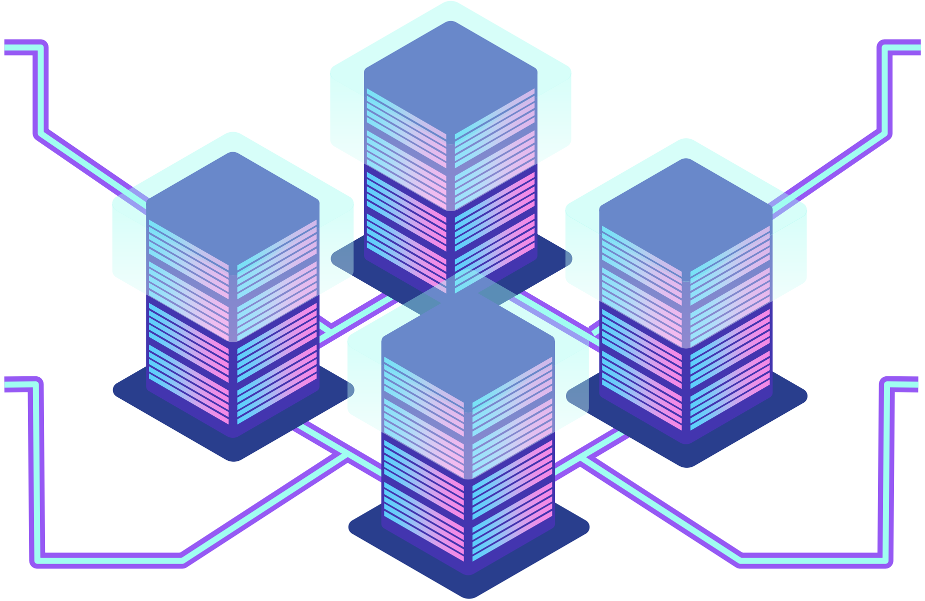
Distributed Sudoku Solver

Computação Distribuída 2023/2024

Henrique Oliveira 113585
Raquel Vinagre 113736

Introduction and Architecture



Sudoku solver

This project consists of a Distributed Sudoku Solver

HTTP

Each node has an http interface to handle client interaction and requests sent to the node

P2P

Nodes communicate with each other through a P2P-oriented implementation

Node | RequestHandler | Server state

The RequestHandler handles HTTP requests from clients, Nodes process Sudoku tasks and the Server State manages network topology and stores statistics

Protocol

01

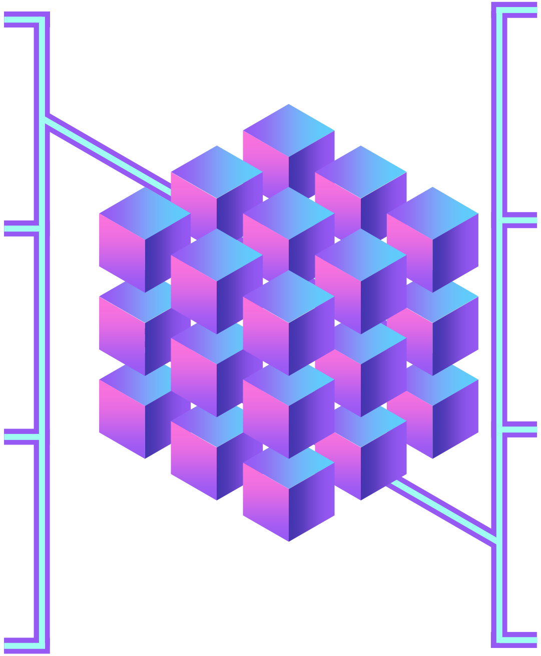
UDP

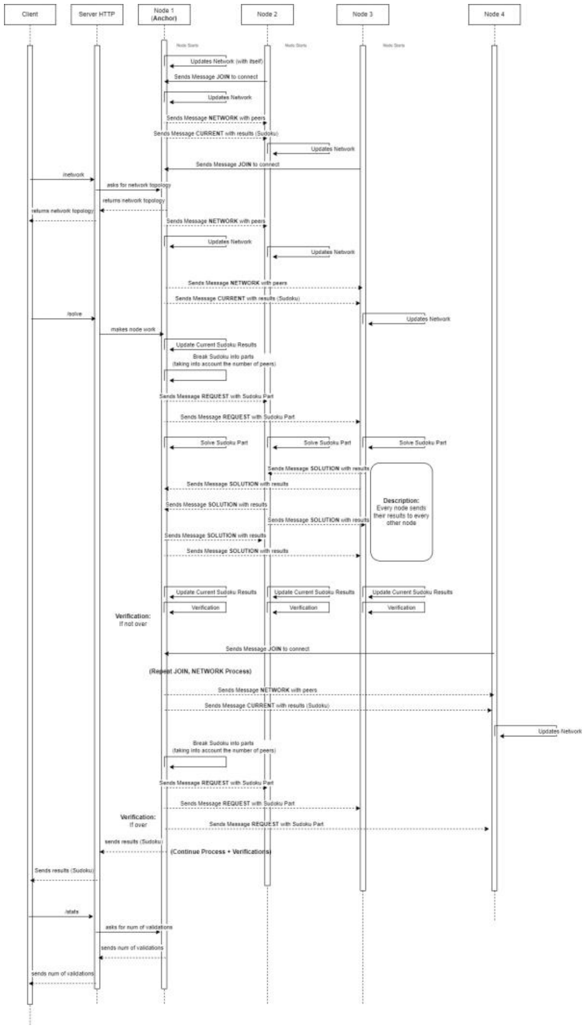
We chose **UDP** for the **P2P** network to achieve faster solving times, which we thought was more valuable over assuring a firmly established connection with TCP.

02

TCP

HTTP over **TCP** ensures reliable delivery of requests and responses, which is crucial for client interactions where data integrity is important.




Client
Server HTTP
Node 1 (Anchor)
Node 2
Node 3
Node 4
Node Starts
Node Starts
Node Starts
Updates Network (with itself)
Sends Message JOIN to connect
Updates Network
Sends Message NETWORK with peers
Sends Message CURRENT with results (Sudoku)
Updates Network
Sends Message JOIN to connect
/network
asks for network topology
returns network topology
returns network topology
Sends Message NETWORK with peers
Updates Network
Updates Network
Sends Message NETWORK with peers
Sends Message CURRENT with results (Sudoku)
/solve
Updates Network
makes node work
Update Current Sudoku Results
Break Sudoku into parts
(taking into account the number of peers)
Sends Message REQUEST with Sudoku Part
Sends Message REQUEST with Sudoku Part
Solve Sudoku Part
Solve Sudoku Part
Solve Sudoku Part
Sends Message SOLUTION with results
Sends Message SOLUTION with results
Sends Message SOLUTION with results
Sends Message SOLUTION with results
Sends Message SOLUTION with results
Sends Message SOLUTION with results
Description:
Every node sends their results to every other node
Update Current Sudoku Results
Update Current Sudoku Results
Update Current Sudoku Results
Verification
Verification
Verification
Verification:
If not over
Sends Message JOIN to connect
(Repeat JOIN, NETWORK Process)
Sends Message NETWORK with peers
Sends Message CURRENT with results (Sudoku)
Updates Network
Break Sudoku into parts
(taking into account the number of peers)
Sends Message REQUEST with Sudoku Part
Sends Message REQUEST with Sudoku Part
Verification:
If over
Sends Message REQUEST with Sudoku Part
sends results (Sudoku)
(Continue Process + Verifications)
Sends results (Sudoku)
/stats
asks for num of validations
sends num of validations
sends num of validations

Sudoku Distribution

Sudoku parts splitting logic

The "to-be-solved" sudoku is split in 3x3 grids, spread between existing nodes

Dynamic distribution

The workload distribution is done according to the number of existing nodes

Part solve logic

Each node performs a local solve on the part(s) it was given, according to the sudoku grid rule

Parallel solving

The solve logic mentioned above is done by all nodes concurrently

Local grid check

After trying to populate the 3x3 grid, the node then checks if the outcome could be a solution

Return part to anchor

After the part is solved, it is sent back to the anchor node

Sudoku assembly

After receiving all parts, the anchor node assembles the sudoku with all parts in their correct position

Sudoku final check

Finally, the anchor node’s HTTP counterpart does a final check of the re-assembled sudoku and sends the result to the client

Stats and Network

Stats

- Show total validations and solves
- Each node keeps a record of its validations
- Updated in real time

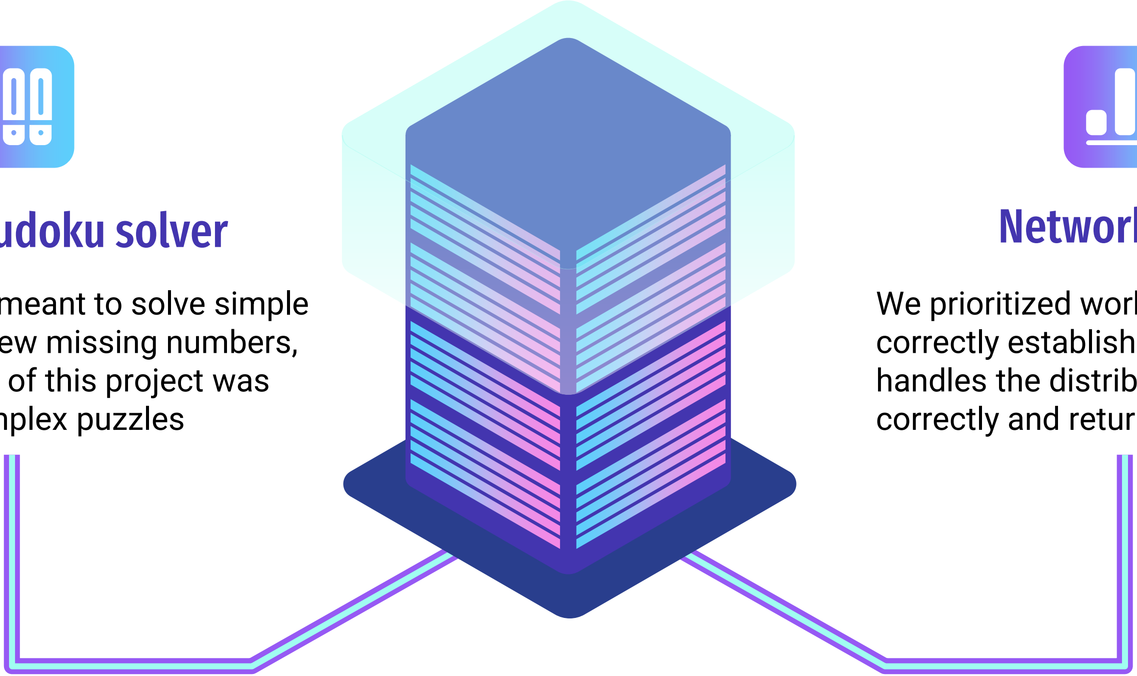
Network

- Information on all nodes' connections
- All nodes are informed when a new node joins
- New nodes on join learn about the network topology

Executive Decisions

Simple sudoku solver

Our solution is meant to solve simple sudokus, with few missing numbers, since the focus of this project was not solving complex puzzles



Network logic

We prioritized working towards a correctly established network that handles the distribution of workload correctly and returns a solved sudoku