Distributed Sudoku Solver Protocol

Computação Distribuída
2023/2024

Henrique Oliveira 113585
Raquel Vinagre 113736

Protocol

Overview

The distributed Sudoku solver is implemented as a peer-to-peer (P2P) network where each node can act as both a client and a server. The nodes collaborate to solve Sudoku puzzles by distributing parts of the puzzle among themselves. The key components include SudokuNode, RequestHandler, Sudoku, and ServerState.

Components

1. SudokuNode Class

The SudokuNode class represents a node in the P2P network. Each node handles P2P communication, task distribution, and coordination of solving Sudoku puzzles.

- **Initialization**:
  - Sets up host, ports, peer connections, task storage, and server state.
  - Determines the host IP and binds the P2P server socket to the specified port.
  - If the node is an anchor node, it initializes the peer list with itself.
  - Non-anchor nodes attempt to connect to the anchor node and join the network.
- **Start Methods**:
  - Binds the P2P server socket and starts the HTTP server.
  - Connects to the anchor node and begins accepting connections.
- **Connection Handling**:
  - Accepts and handles incoming P2P connections, processes messages, and manages network state.
- **Message Processing**:
  - Handles different message types (JOIN, UPDATE, LEAVE, REQUEST, PARTIAL_SOLUTION, SOLUTION, NETWORK), updating peer lists, solving Sudoku parts, and coordinating the solution process.

- **Task Distribution**:

    - Splits the Sudoku puzzle and distributes parts to peers. Each part is represented by a 3x3 grid of the sudoku puzzle.

- **Sudoku Solving**:
    - Solves Sudoku parts and checks for solution completeness.

2. RequestHandler Class

The RequestHandler class handles HTTP requests for solving Sudoku puzzles and managing the network.

- **HTTP POST Requests**:
    - /solve: Receives a Sudoku puzzle, assigns an ID, distributes parts, waits for the solution, and returns the solved puzzle or a status message.
    - /check: Receives a Sudoku puzzle and checks if it is solved.
- **HTTP GET Requests**:
    - /network: Returns the current peer list.
    - /stats: Returns server statistics.

3. Sudoku Class

The Sudoku class represents a Sudoku puzzle and contains methods for solving and validating the puzzle.

4. ServerState Class

The ServerState class manages the state of the server, including statistics and other relevant data.

- **Initialization**:
    - Initializes statistics for solved puzzles and validations.
    - Manages network state and flags for solving status.
- **Updating Statistics**:
    - Updates statistics for nodes based on solved puzzles and validations.
- **Retrieving Statistics**:
    - Retrieves current statistics for all nodes.

Protocol Flow

Node Initialization and Connection

1. Each node is initialized with HTTP and P2P ports, a handicap time, and an anchor node address.
2. Nodes determine their own host IP and bind the P2P server socket to the specified port.
3. If the node is an anchor node, it initializes the peer list with itself.
4. Non-anchor nodes attempt to connect to the anchor node and join the network.

HTTP Server

1. Each node runs an HTTP server that handles requests via the RequestHandler.
2. The HTTP server processes POST requests to /solve and /check, and GET requests to /network and /stats.

P2P Communication

1. Nodes accept incoming connections on the P2P port and process messages using the process_message method.
2. Messages can be of various types: JOIN, UPDATE, LEAVE, REQUEST, PARTIAL_SOLUTION, SOLUTION, NETWORK.

Joining the Network (**JOIN**)

1. When a node wants to join the network, it sends a JOIN message to the anchor node.
2. The anchor node updates its peer list and informs all existing peers about the new node.
3. The new node receives the current network topology and updates its own peer list.

Updating the Network (**UPDATE**)

1. When a new node joins, existing nodes are informed via UPDATE messages, ensuring all nodes have a consistent view of the network.

Leaving the Network (LEAVE)

1. When a node leaves, it sends a LEAVE message, and other nodes remove it from their peer lists.

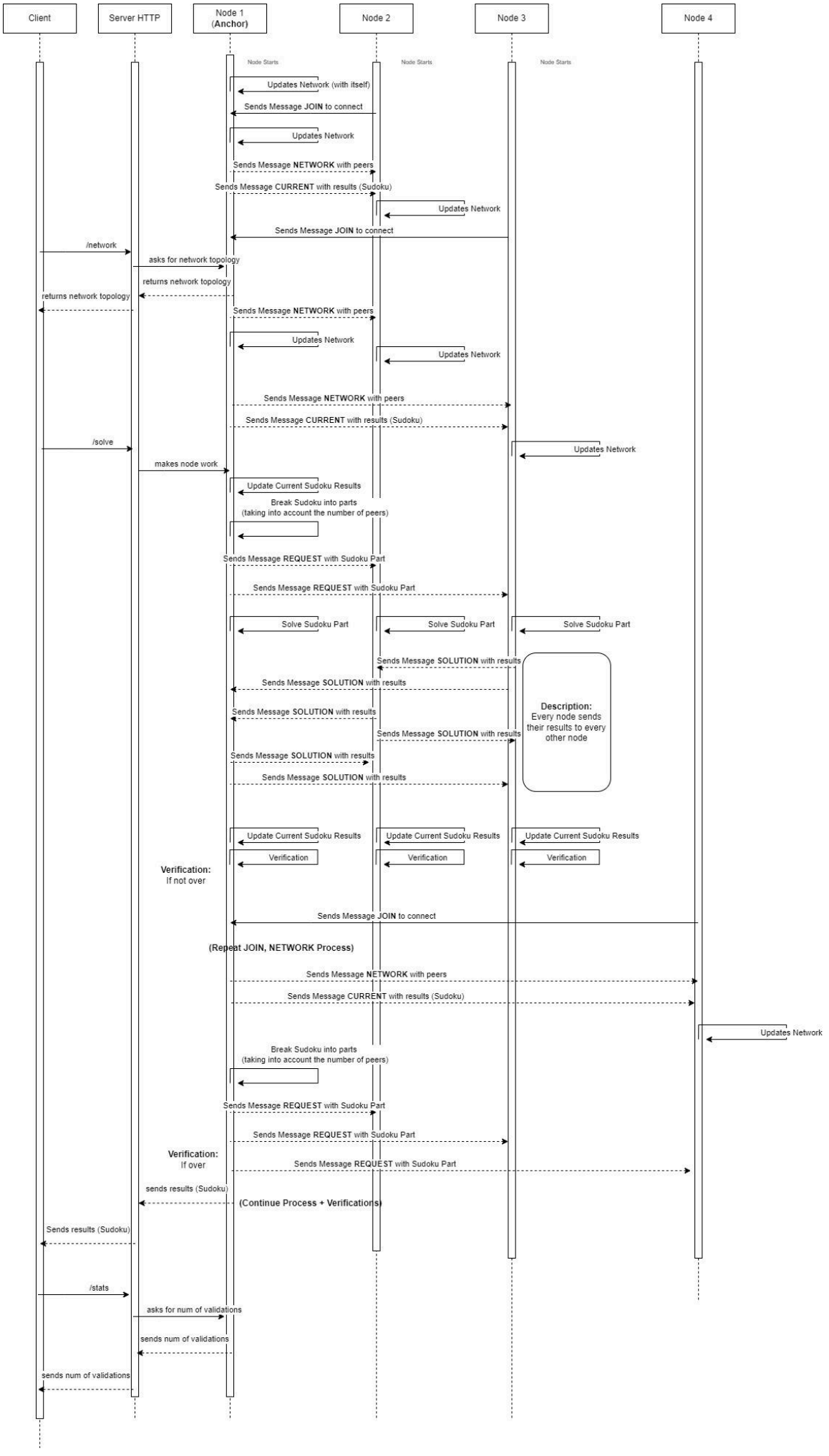
Solving Sudoku Puzzles

1. **Distributing Tasks**: When a node receives a REQUEST message to solve a Sudoku puzzle, it splits the puzzle into 3x3 parts and distributes them among peers using distribute_sudoku_task.
2. **Solving Parts**: Each node that receives a part attempts to solve it and sends back a PARTIAL_SOLUTION.
3. **Collecting Solutions**: The main node collects all partial solutions and attempts to solve the entire puzzle. If successful, it broadcasts a SOLUTION message.

Example Protocol Flow

1. **Node Initialization**:
    - A node starts and checks if it's an anchor node.
    - If not an anchor, it sends a JOIN message to the anchor node.
2. **Joining the Network**:
    - Anchor node receives JOIN, updates peer list, and sends an UPDATE message to all peers.
    - New node receives network topology from the anchor and updates its peers.
3. **Solving a Sudoku Puzzle**:
    - A client sends a puzzle to a node via /solve.
    - The node splits the puzzle, assigns parts to peers using REQUEST messages.
    - Peers solve their parts and send PARTIAL_SOLUTION messages back.
    - The main node collects parts, assembles them, checks for completeness, and returns the solved puzzle.

(Protocol diagram in the next page)


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