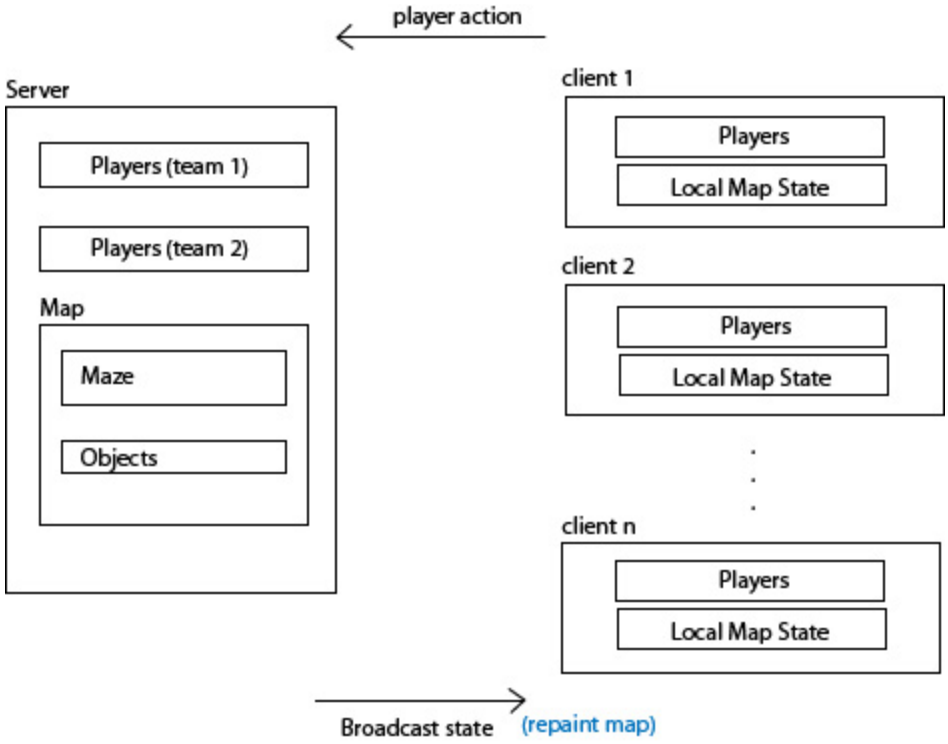
Structures



Various game semantics and how it's implemented by the data structures:

Game initialization (server side) :
Read in the map file and parse it to store it as cells in the 2d array 'map'. Also store all jail and home-base cells for each side in separate arrays, these arrays will hold the positions of these cells (for ease of access this data later on). Store each player on both teams into the Players array (according to their teams). Each Player type consists of its position (x, y), its team and whether it's holding a flag/jack-hammer and whether it can move(because of jail). The objects are also stored in an 'object' array, where we store for each object it's position (x,y) and whether or not it's being held by a player. Set the Gamestatus (enum type) from 'NOT_STARTED' to 'IN_PROGRESS' once the host has decided to start the game and there is an even amount of players on each side. Randomly assign each player a position in the homebase on their side of the map. Broadcast the game state to all players.

Game Synchronization:
To make sure the map never reaches an inconsistent state since the state of the game is being updated very frequently, we will implement a lock on the game state data structure on the server side; before each update to the game state, we will lock this mutex and unlock it when the state update it done. This prevents multiple threads from modifying the game state at the same time.

Player moving to different tiles :
Player sends the server it's desired new coordinates to server via RPC, based on the

coordinates the server decides if it's a valid value then goes into the players array and sets the player's coordinates to the new moved coordinates.

If the player is also holding an object, update the object's position to the player's new position.

If the player's new coordinates are that of the enemy's jail cell, look through all player's team members that are in jail by searching through the player array coordinates and checking if they match the enemy team's jail array coordinates to find those players on the same team that are in jail cells. For each of these players and the player itself, set their new position to a random, unoccupied space in their home base and make all of their 'canMove's to true.

If the player's new coordinates places it on its team's side and there are enemies around the player (look at Players tagging other players):

Then it broadcasts the new (updated) player arrays to all subscribed player/clients.

Players tagging other players:
When a player's new coordinates (from a move action) are on its own side, check if there are enemy players on the four adjacent tiles next to it (Up, Down, Left, Right). If they are, all these players are sent to jail (look at Going to jail).

Going to jail:
If a player is being sent to jail, mark the random tile it's assigned to in the jail area to the player's position and also mark the player's canMove to false.

Shoveling/Jack-Hammering:
Player sends 'shovel' action to server (which includes which direction the player wants to shovel: Up, Down, Left, Right). The server first checks if the player is holding a shovel (in the player type). If player holds a shovel, depending on the direction the player wanted to shovel we check if that tile is an interior wall we can destroy. If yes, we mark that coordinate in the map array as a floor. Then mark the user as no longer holding the shovel (one charge) and set the object as no longer being held and set its new position as the designated shovel position for that team. We then broadcast the new map, object and player arrays to all subscribed players.

Picking up objects:
Player sends 'pick up' action to server. The server checks each of the four adjacent tiles to the player and if there is an item in any of them (for each tile compare the tiles location with the location of the objects). If there is a match, mark that player as carrying a shovel/flag (whichever object was matched). And mark the object as 'held'. Then broadcast the updated players and object arrays to all players.

Dropping objects:
Player sends 'drop' action to server. The server gets the player's coordinates and assigns the

object to that position. Mark the player as no longer holding an object and the object as no longer being held. Broadcast updated player/object arrays.

Player Disconnection:
If the host of the game disconnects, the game will end and broadcast to all the other players the game is over. If a player who is not the host disconnects, the player will be removed from the map, and the game will continue.

Winning/Losing:
After each ‘move’ action - check the position of the two flags, if they are both on the same side of the board check to see the position of all players on that team. If all players’ positions are on their own side change the gamestate gamestatus to ‘PLAYER1_WIN’ or ‘PLAYER2_WIN’ depending on which side it was. Include this in the game state broadcasted to all players. Once players receive the gamestate from the msg received it will check the game status to each time to see if the game is “IN_PROGRESS” once it’s changed to a win, the game ends, declaring one side the winner.