4p missing the last point of plan as listed in the project description!

Distributed systems 2023, HY

BingoGame - Design plan

Team members: Mila Katajisto, Laura Nordlund

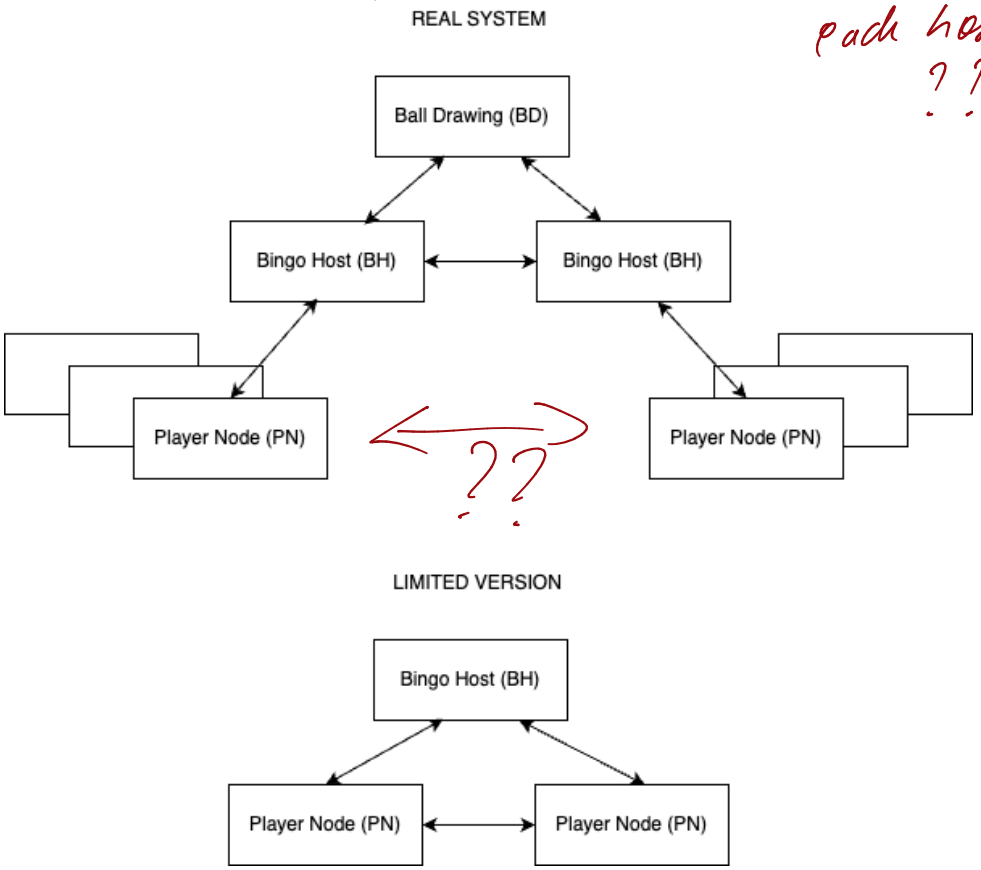
Selected topic

The aim of this project is to create a service, where users can play bingo against each other. The bingo host is in charge of drawing the 'balls', keeping the master state of the current game and broadcasting the drawn numbers to the player nodes. The player nodes keep track of their own bingo cards and a copy of all the drawn numbers. The players can communicate with each other during the game.

Description

In this project we focus on creating a small network with a limited number of functions. In the limited version there are one bingo host and two players. In a real system there could be more players, and it could be possible to have many games hosted by different nodes running at the same time. Also there could be a separate node that would be in charge of drawing the balls.

← same sequence to all hosts or different to each host separately ??



The system will have a shared distributed state, allowing players to see the current state of the game (i.e. numbers drawn) and syncrhonise their moves.

The project will be written in Python, and communication between the different nodes happens through sockets.

Description of node roles and functionalities

Bingo Host (BH)

- responsible for managing overall game state
- controls the generation of bingo numbers and broadcasts them to the players
- responsible for node discovery and initialisation of the player nodes

Player Nodes (PN)

- represent individual players playing the game
- receive bingo numbers from the BH and mark them on their bingo cards
- communicate with the BH and other player nodes for synchronisation (sync checks, communicating about card hits and bingos)

Node Discovery

- BH manages a centralised naming service where PNs register when connecting to the game
- The naming service allows the player nodes to discover and identify each other during the game

Before the game starts

- BH initializes and starts listening for incoming connections from PNs
- Each PN establishes a connection with the BH
- PNs register themselves with the centralized naming service facilitated by the BH
- Upon registration player also receives a bingo card
- Registartion for a round is open for a certain period of time
- At least two players need to register in order to start the game
- When registration time ends, BH sends a signal to all players to start the game - the message could also include the connection information of all player nodes
- Players acknowledge to have received the information and that they are ready

During the game

- BH broadcasts numbers to players one by one
- When receiving a number, PN checks if the number is in the player's bingo card
- If it's a match, PN stores the match and sends message to other nodes
- Player nodes keep track of all the drawn numbers

- Players can send synchronisation requests to other players to make sure their state is up to date - basically the response to a request should contain the numbers that have been drawn so far
- Communication among PNs occurs through the connections facilitated by the BH's naming service

directly or via BH?

must be!!

Ending the game

- When a PN recognises a bingo, it sends a message to the BH
- BH checks if the bingo is accurate ← so also BH knows the cards!
  - if yes, the winner is announced to all players
  - if not, BH announces that the round continues
- Consensus round: When a bingo message is received, BH sends a verification message to other nodes with the numbers forming the bingo. Other PNs check if they agree given numbers are right and send a message to BH.
- BH handles the cleanup process

ok

alternatively the PNs could agree on each received number separately.

Communication Between Nodes and Messages

The limited system version uses socket-based communication. While sockets are perhaps not the optimal solution for node discovery, it should be sufficient in this simplified, smaller-scale system. When building the real system it would make sense to explore additional tools and libraries with more sophisticated functionalities.

Messages

Start Game Message

- Syntax: { "type": "start_message", content: "Game starts now!", connections: { "player1": player1.address, "player2": player2.address }}
- The message that the BH sends to all players when the game begins
- Contains the connection information to other player nodes
- PNs should acknowledge the message

Bingo Number Message

↓ Bingo numbers could also be like H47 ← for letter of column and number in row

- Syntax: { "type": "bingo_number", "number": 23 }
- The message that the BH sends to all players for each new drawn number

Number Marked Message

- Syntax: { "type": "number_marked", "player": "player1", "number": 42 }
- The message that PN sends to all other players when a number is marked in their card

Synchronisation Request Message

- Syntax: { "type": "sync_request", "requester": "player1" }

- The message that a PN can send to other player nodes to ensure that their local game state is up to date. The response to this message should contain the numbers that are drawn so far

Bingo Message

- Syntax { “type”: “bingo”, “numbers”: [2, 10, 21, 33, 46], “player”: “player1”, “timestamp”: “2023-11-20-13:23:44”}
- The message that a PN sends when they identify a bingo in their card
- Timestamp to ensure that if there are multiple bingos at approximately the same time, the first one wins

Voting Round Message:

- Syntax { “type”: “winner_confirmation”, “player”: “player1” }
- The message sent by BH when the winner for a round is found
- Response from player nodes: yes / no

Round End Message

- Syntax { “type”: “round_end”, “winner”: “player1” }
- The message sent by BH when the winner for a round is found

Bingo card from host to player missing

Nice plan. Simple game but the description of implementation idea is complex enough.

A bit unclear what features the prototype will contain
also there was not yet any mention about scalability or fault tolerance

4p