Zeemal Urooj

2018-UET-NML-CS-19

1802019

Semester 6 (Year)

12 June, 2021

Assignment:

Design a protocol for single player game

ABSTRACT:

We are going to develop a single player Snail game in which there will be human on one side and an AI agent on other side. We need a protocol to communicate between human player and AI player. This protocol will be responsible for the establishment of connection between human and server. The protocol will transfer human's move information to the server and the server will then do some computations for AI agent to move intelligently against human. There will be a main server, a grid sub-server, a heuristic sub-server and an AI agent in the process of communication of protocol.

Table of Contents

Functional Requirements of App:

Functional Requirements of application are:

- AI agent shall be able to play intelligently against the human player.
- AI agent shall be able to keep track of valid moves.
- AI agent shall be able to keep track of invalid moves.
- App should give the track that which player has won the match.
- App should allow AI agent to play against single player.
- Turn will be lost if a player does wrong move.
- Application must keep the record of both players that who is currently winning the game either AI player or human player.
- Whenever a player either AI player or human player do a turn than there must be an addition in the score of that player.
- AI agent should instantly response on its turn.

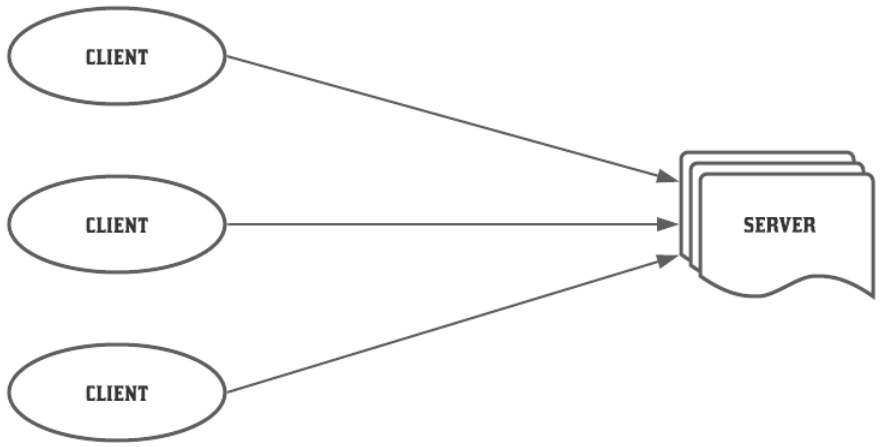
Non-Functional requirements of App:

Non-functional requirements of application are:

- Application will be android based and also web based.
- The game should be able to change its UI according to device.
- User should be able to choose snail of its own choice.
- Colors must be attractive for both snails.
- Snail should be able to slide on its own paints.
- Score board must be show with the table.

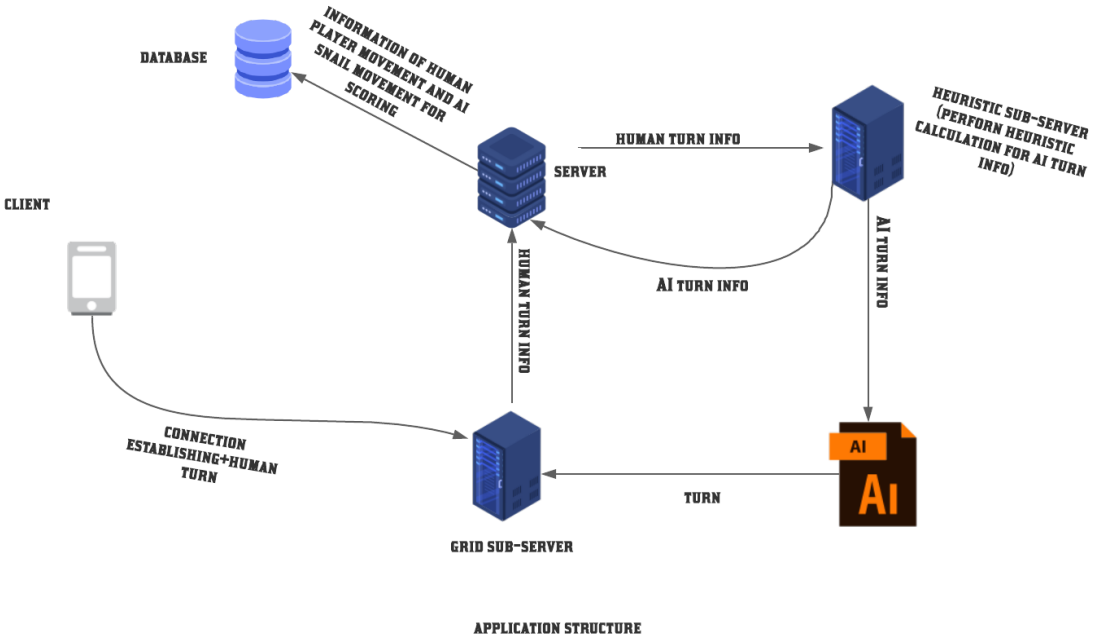
Application Structure:

The structure of this app is client-server based. Client is basically the human player and server is basically acting to perform calculations and move AI snail in the table according to that calculation. Grid (a layout system with columns and rows) is used in the game. Each block of row or column has some port number, IP address connected to it. When player turn on the game, client sent a connection request to server and then server accept the connection request. So whenever human player does his turn on grid, client send the information (port number and IP address of the block) to the server. Server then send the information to sub-server which is actually performing heuristic calculation and return the result to server. Server then according to this result send the information to AI snail. Then AI snail move on the grid with respect to this calculation.



CLIENT-SERVER
BASED PATTERN

Server save all the information which client send to the server. At start, both the snails are at zero position. Client request sub-server(grid) for connection and server accept the request. When human player does his turn then the snail of human player move to next grid and there is some port number and IP address associated to this grid. Grid (sub-server) send the port number and IP address of that grid to server and server send the information to sub-server which perform the heuristic calculation and return result to server then server send result to AI snail and AI then move on the grid according to that and at each step server is also keep track of scoring of both the snails to make sure who is currently winning the game. And for this server store the result in database after each move of both the player.



APPLICATION STRUCTURE

The Protocol:

Protocol is basically a complete set of rules for data communication. These rules are defined for the communication between two or more computers. Through these set of rules (protocol) computers communicate with each other very easily.

Main points of Protocol:

- Message Types
- Message Format
- Message Syntax
- Message Semantics

Brief description of these points is:

- ## Message type:

There are mainly three types of message types.

a) Command message
b) Data transfer
c) Control message

Two types of messages are used in our protocol. First one is command message which is used for connection and second one is data message which is used for all data transfer in our application.

At start, when human player turns on the game server and sub-servers are already connected. Command message is use for connection purpose. So, when client request for the connection to sub-server (grid) command message is used.

When grid send human turn info to the server data message is used because sub-server (grid) send the data of block. Server then send the data to sub-server for heuristic calculation using data message. Sub-server then sends the result to server and server store the result in database for scoring purpose using data message. Similarly, sub-server (heuristic) send the result to AI snail and AI send the result to sub-server(grid) using data message.

- ## Message Format:

There are two types of format:

a) Text oriented
b) Binary messages

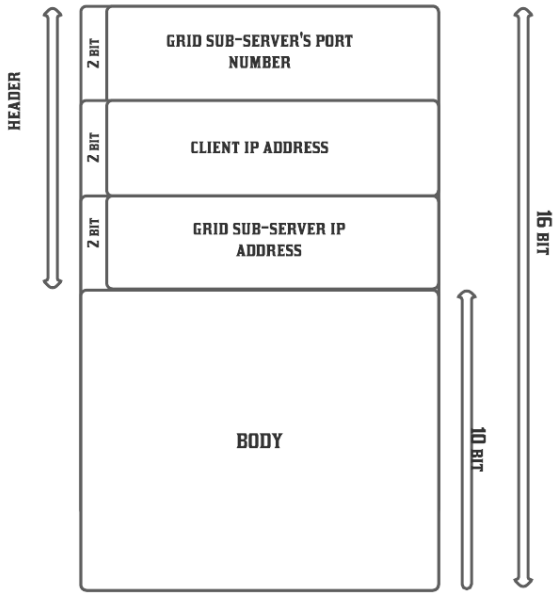
Our protocol will use only binary message format. Because there is no large data transfer in it. And in every data packet, header include the message types and IP addresses and body include the information about position of snails in grid sub-server. So, for this only binary message format is used because machine is easily dealing with the binary format.

And also, our protocol is just handling the connection and the movement of human player and the movement of AI player with respect to the human player after some calculation using IP address and ports so for binary messages approach is best.

- ## Message Syntax:

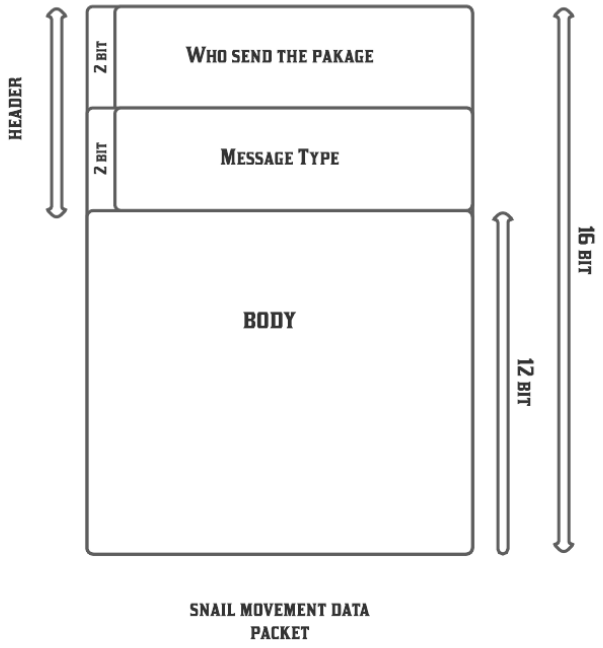
There will be total 3 packets used in this protocol. Header and body include in all 3 packets. First data packet is when client send grid sub-server a connection establishing

request. This data packet also includes header and body. First 2-bits of packets include grid sub-server's port number, next 2-bits include client IP address and next 2-bits include server IP address. Header use total of 6-bits. Body include next 10-bits.



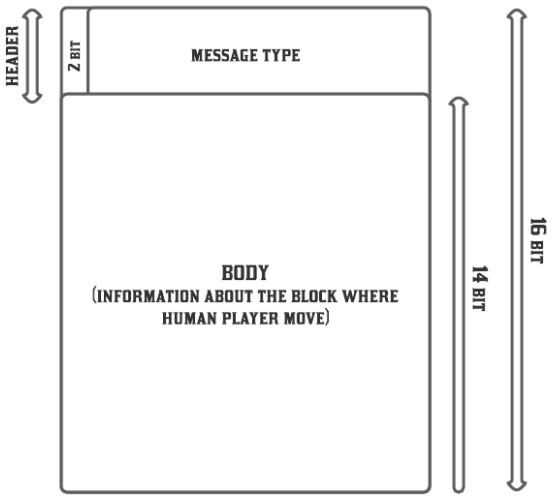
CONNECTION ESTABLISHING

Similarly, grid sub-server has another packet of 16-bits on which it moves the snails. Both human player and AI agent can send this packet. This packet also includes header and body. First 2-bits includes who send the packet human player or AI agent, next 2-bits include the message types. Header uses total of 4-bits. And body include where to move. Body uses total of 12 bits.



SNAIL MOVEMENT DATA PACKET

Another data packet when grid sub-server sends the main server, when it sends the information about the human player movement to main server. This packet also includes header and body. Header include the message type and body include the position where human player moves. Same type of data packet is used when main server send data to heuristic sub-server. And same type of data packet is used when heuristic sub-server send result to AI where AI move on grid.

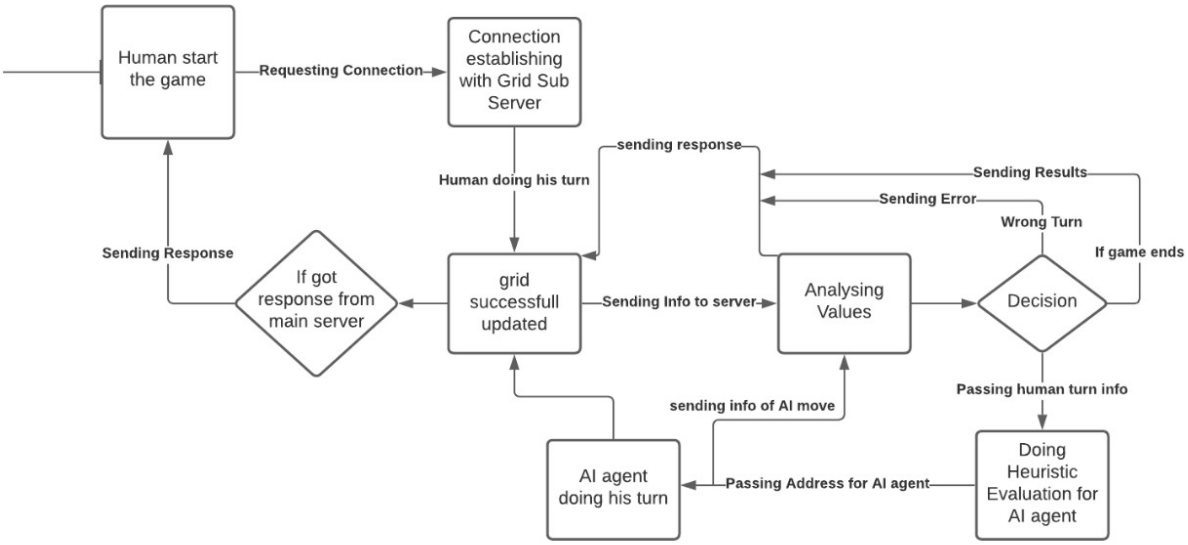


- **Message Semantics:**

Two sub severs are, 1 grid sub server second is heuristic sub server. Both of them will be connected to our main server. Two sub-servers are established a connection to main server and always remain in listening state. When human player turns on the game it sends the connection request to its sub-server (grid) and established a connection with sub-sever (grid).

When it accepts the request, human player play his turn on grid and then sub-server (grid) send the information of the block on which human player moves, to the server and main server store this in data base and increment in human player score. Main server also sends the information to sub-server (heuristic) for calculation then sub-server returns the result to AI and main server. AI snail on the bases of that move on the grid. Similarly, main server is also storing the information of AI in data base and also increment in AI score. On every turn main server will analyze the score of both players to keep track of both the player who is winning the game.

But when both the players either human player or AI snail does a wrong move, there turn will be lost and grid sub-server will send an error message to the back from where wrong turn has been done and main server will not increment in the score. If the grid is full and the score of both the snails are same the game will end and main server will send an encoded message to grid sub-server and grid sub-server will send a message of draw to the client.



If at the end human score is greater than AI score then the main server will send an encoded message to grid sub-server in which the winner name will be placed and the sub-server will show the name of the winner to the client.

Additional information for developer:

- Try to use simple language in coding so that other will also understand your code.
- Use simple comments on almost every line so that it become easier for understanding.
- Draw some paper prototypes for testing purpose so that it become easier for developing purpose.
- Always connect with the client or with some senior person while developing.
- Divide your work in small tasks and always work in small tasks because it helps in error detection.
- Try to develop protocol which is efficient, not costly and easy to handle.