CPSC 416: Project 2

Description

A distributed game where the game state is hosted on the server, and each car connects to the server to play the game. A car will only connect to the server to join the race when it reaches the minimum threshold of players. The server will then start the game when it has enough cars in the race. In order for the cars to move start moving, players inside the cars must answer problems/questions that arises in their screen. The car will then answer with the majority decision of the players (if it is correct then the car will move a specific distance and is asked a new question, otherwise freeze for 1 second and then is asked a new question).

Nodes

The topology of the game application will be composed of three types of nodes: single server, car, and player.

Server
Keeps track of the number of cars connected and works to connect each cars with all the other cars on the network. The server will generate a random race track with obstacles in the map as well as trivia questions to asa. Starts the race when there is enough cars connected to the Server. List of trivia questions will be sent to each car in the beginning.

Car
A car node is a supernode which will disseminate information from the players and its own heartbeat using smart query flooding (similar to what Kazaa uses). Each car is connected to the server to retrieve information about list of questions, random race track, and other cars connected in the network. A car is also connected to set of players that makes up the “team”. A car is also connected to rest of the cars in the game they’re competing against. The car node will have a disk storage to store the race track and the questions from the server (given at the beginning of the race). The car node will then distribute these questions to the players inside the car.

Player

Each player node is connected to a unique car node (only 1 node at a time). The player node will also be sending a heartbeat to the Car node to notify whether the player has disconnected from the car or not.

Objective

A game similar to mario kart where players race to the finish line but other players can affect the gameplay of the other players in the network with various interruptions. To move in the race and to interrupt other players, each player will receive a question (trivia type questions) from the server, the first player to answer correctly will be able to select either to move forward by x-amount of spaces or to use an interruption to affect the other players gameplay on the network (i.e. time-out the players).

Challenges

To maintain fairness when sending questions to the players we need to ensure that each player receives the question at the same time. Also, when answers are received we need to make sure that the player who answered first and correctly should receive the reward.

Distributed Systems Aspects and Challenges:

Synchronization

Each car must store the most up to date game state, such that when one car moves forward in the map, the other cars in the game should also update their game state. To ensure fairness, each car should track location of other cars and be able to confirm valid moves of other cars (i.e. valid move is when a car moves specific distance, no more, no less).

Concurrency

When two cars try to get into the same position at the same time the system will choose a winner of the two and then decide which one of the cars to advance in position. This will depend the other players in the other cars majority decision. Every time two cars advance to the same position, other players will be able to vote for a car in which they think should be the “winner” and the car will take the majority decision and will then send it to the server. The server will then tally up the score and choose a winner

Scalability

The System has the ability to scale with the numbers of cars or players participating inside the race and should be able to handle large numbers of cars/players.

Integrity
When a malicious car tries to send a location that is not consistent with other cars(players), there will be a validation process in which the other cars will deem it unfair. At that point the malicious car will have its position frozen for a certain amount of time to prevent it from doing so again

Failure Handling (parts of it)
In our game, players can be knocked out of the car from impeding obstacles on the tracks (means that they have been disconnected from the car). The player have an option of returning back into the race and reconnecting into their car node by solving a Proof of Work. There will be a restriction such that the player can only go back into a car in which they have been kicked out of.

<u>SWOT ANALYSIS</u>

<u>INTERNALS</u>

Strengths:

- Group members are familiar with one others coding practices and also have communication experience with one another making it easier to coordinate
- Team members are hard working, intelligent, and well versed in the systems we intend to design.
- Members have experience programming a peer-to-peer system with error handling from Project 1.
- Able to identify the areas that need improvement and then collaborate and collaborate together to implement solutions.
- Members in the group all have knowledge on the proposed project and have an idea of what is going on in every aspect of the project.

Weaknesses:

- Not being able to schedule meetings could become an issue with people having other courses to work on.
- Complexities that may hinder the robustness of the program should be avoided while still maintaining a level of complexity as to satisfy the end-user.
- Having to use the golang testing framework which is not familiar to any of the group members.

<u>EXTERNALS</u>

Opportunities:
- Project idea is neat solution to a game which involves many clients in a peer-to-peer system
- Being able to explore systems that define what it means to be a distributed one
- Using a new tool for everyone that is Azure (although we already used in project 1 it is still a learning process).

Threats:
- Time constraints on project deadlines not being met.
- Adding features to the system to best showcase aspects of distributed systems without being too simple.
- Being able to fully test the system and debug most of the key bugs affecting the system before rolling out on the final deadline

The proposal draft being due on Mar 2nd is not included in the deadlines as this report encompasses that date

Date	Event
March 9	Final Proposal for report is due
March 13	Finalize all implementation as well as divide up the tasks and being the initial coding of system
March 18	Implement the car, server so that the car can act in a peer-to-peer system
March 23	Schedule a meeting with the TA
March 24	Finish implementation of the person, the coordinate system for the track, and features involving concurrency between cars
March 31	Finalize all the details, debug the system and start the testing process
April 6	Project code freeze and all testing finished
April 15	Demos and peer review