B438 Network Game Paper

Team Members:

Matthew Branstetter

Branson Ferrell

Jason Henry

CSCI B438:19909

Paper

Dr. Doyle

December 2, 2020

Background:

In Networking B438, Dr. Doyle assigned a group project that was the duration of the whole semester, and the purpose of the project was to promote the learning of networking and how machines communicate with one another; while increasing knowledge about coding, networking, and familiarizing oneself with typical problems that are encountered while learning basic networking.

The group decided to create a game that was based around Battleship. Battleship is a notable game that consists mainly of two players. Each player get five ships, and would place them strategically around a grid (x,y) with one side being numbers, and one side being alphabetical letters. The numbers are on the left-hand side of the grid, while the letters are on the top.

Players take turns guessing where the opposing player's ships are, and if the guess is incorrect, it is known as a "miss." If a player guesses a space in which a ship resides, then it is a "hit." If the player fills the ship entirely up with "hits" then it sinks, and then four ships remain. Whoever can sink all five ships first is the winner.

In the game that the group created, it is very similar to Battleship, but slightly more simplified. The game resides on a blue grid, where the ships are grey squares. Red means a ship is hit, and white means a miss. (To reiterate: Blue means water, Grey means boat, Red means hit, White means miss.) However, the group made it so that each player can see what has been declared as "hit" and "miss" in real time, making it an online multiplayer game. The ships are only one size to keep simplicity.

Each person connects directly to one another, making one person the "server" role while the other is "client." This means that the connection is defined as "Server to Client" versus "Peer to Peer." This differs slightly because, the "Server" (typically player #1) will always go first and be able to choose the desired square to "attack." This is for simplicity sake, because if not, there would have to be additional resources added to aid in determining which person (Player #1 or Player #2) would go first.

Rationale:

The group decided that it would be the best route to take, using UDP. UDP stands for User Datagram Protocol. UDP is connectionless so it does not establish that a proper connection is made, it disregards error checking and correcting (contrary to TCP). UDP will erroneously drop packets so it is not as reliable as TCP, but the group decided to use UDP because it was deemed more efficient and wanted to accomplish a simple task with a fast-paced return. UDP had more documentation and was easier to navigate during the project versus TCP.

The rationale for using UDP at the time was favored by the group.

Originally, the group want to use Python, but Python was deemed inadequate and cumbersome to accomplish the project. The group decided on Lua script, specifically the LÖVE (Love) engine. Love is automatically included on the Linux operating system and Ubuntu. All group members were using some form of Linux or Ubuntu, so it seemed logical to use software already presented on the machines.

Some members of the group had to familiarize with Lua but was overall simple to learn. There were multiple sources on various platforms that were consulted while creating the game, which made it easier to understand the problems that were being encountered and how to navigate these problems while expanding

the group's experience networking. With all the information, wikis, videos, and previous examples by professional developers; it became clear to the group that this was a wise move to convert the Python code into Lua Script.

Resources:

Some of the resources that were consulted were things like YouTube videos, GitHub developments, Love2d wiki, and additional information on Lua sockets, which derived from a private user with a website dedicated to teaching people how to be fluent in Lua. Love2d wiki makes it very accessible to grasp, formulate, and implement programs on Love Engine, and makes it so programmers are able to access mountains of information, how to troubleshoot specific problems, contact professionals within the community, and how to successfully create an enjoyable game. GitHub was also increasingly helpful because professional and amateur programmers alike had a plethora of examples that the group could follow along to. The group used YouTube rarely, but found one video helpful, that helped understand the socket library in the Love Engine.

Results:

The group was able to achieve a game which connected through UDP ports, that was very similar to Battleship. The

connection is seamless and fast, which was one of the group's goals. The group was able to assign colors to enhance the gameplay. The group was also able to learn Lua Script and be confident in programming in Lua Script. Basic networking, and how machines communicate was also discovered. Game methods and ideology along with some basic development principles were learned and adopted graciously by the group. A dream among the group was to develop a working game of Battleship and create a unique experience that enhances replay-ability making it enjoyable content for possible future references to projects.

Problems:

There were two major problems the group discovered. Packet loss and packets being received incorrectly. The group sketched it up to be solely based on UDP, and how UDP behaves. Since UDP does not establish connection nor check that the packets have arrived and does not check the order that the packets arriving, sometimes making the game act belligerently. UDP 'assumes' that the packets arrive, and the game lacks a check system, making it incredibly difficult to backtrack. If packets are lost, the game is under the assumption that the packets are correctly sent, which causes volatility reactions from the game, resulting in an incorrect square being selected instead of the square selected by the player.

Solutions:

The group established some temporary solutions that help resolve one of the two major problems during gameplay. One solution was incredibly simple, but it works. A player would force restart the game, and it would create a new session. While the players would lose progress during the game, it would solve the issues with packets being received incorrectly. The group attempted to establish something more concrete but due to time constraints, was unable to completely quash that problem. However, the group attempted to use Lua ENet, which is an aid to gaming, which contains a protocol layered over UDP. It is supposed to aid UDP and customize it, but unfortunately, it seems like Lua ENet is temperamental and does not always prevent packet loss. The group was actively searching a viable method to prevent packet loss.

Conclusions:

Throughout the project, the group's skills and abilities were tested, along with knowledge about networking. The group learned large quantities of information that helped them develop the game that was envisioned from the beginning of the semester. Research through multiple avenues were sought after in attempts to turn out a quality game that would produce the desired outputs from not only the group, but the professor. It helped

promote communication and time management throughout the group, along with pushing members to learn a new programming language. It also taught members how to correctly make a working game over a network and gave hands on experience. Not all the members will go on to develop games or even program daily, but it was a refreshing task that was presented and gave a glimpse into the daily life of a game (or software) developer.

It can also be noted that members learned new problem-solving techniques, and how to approach certain situations in the industry, and what professionals must deal with, if not the generalized experience.

Members also expanded the knowledge of networking and programming, through different sources. The project was dissected into many parts, and then if an error occurred, trial and error ensued. It was decided that UDP would be the best fit and was also easier to understand. Multiple resources were pursued in attempts to gain understanding and predict how different code would (essentially)react with one another.

The group encountered only two major problems that presented themselves clearly through the development of the game. One problem was solved very easily, but the other presented a true challenge to the group.

Overall, the group developed a working rendition of Battleship that is enjoyable to play with a friend. It broadened the horizons of all members within the group through research, implementation, and execution. The results were what the group envisioned.

Future Work:

In the future, the group would like to figure out Lua ENet and how to create a more concrete protocol rather than UDP. The group would also like to be able to solve the packet loss and incorrect packets in a more "professional" manner rather than just restarting the game and losing progress. The developers hope to add more customization to the game; and develop Battleship further into a more aesthetically pleasing game and increase replay-ability within the game. Plus, add something unique that would draw more players to the game.

Commented Code:

*See Attached

Bibliography:

AtiByte. (2018, December 17). UDP Networking in Lua and Love2d using the Socket Library.

Love2D. (2020, November 15). *N/A*. Retrieved from https://github.com/love2d/love

Love2d. (2020, November 15). *Wiki Navigation: Love2D Wiki*. Retrieved from Love2D Wiki: https://love2d.org/wiki/Main_Page

Nehab, D. (2006, April 20). *Network Support for the Lau Language*. Retrieved from w3.impa: http://w3.impa.br/~diego/software/luasocket/udp.html