

Report Overview

This report covers my project from inception to completion. The abstract section provides the proposed project and why it is considered a proper project for the class. The introduction section is next and provides the motivation and goals for the project. It details if any changes were made to the project in order to complete my goals but still successfully demonstrate my motivation behind the project. Finally, the introduction provides a description of the implemented project. The background details any methods used to complete my implemented project. The proposed method/ system description/ implementation section explains what I did. It includes flowcharts and diagrams of my program. The discussion and conclusion section discusses and analyzes the results of my project. Finally the appendices present my code and anything else that should be considered necessary.

Abstract

My project's intent was to design a game of pong that would be played between two TS-7250 boards. This was done to demonstrate real time communication between two machines. Pong is a simple game, similar to tennis, where two paddles try to hit the ball. If a paddle cannot reach the ball, or if the paddle wasn't near the ball, the ball moves past the paddle. The user who kept the ball in play gets a point. One of the paddles would be controlled by a controller for the hardware element for the project.

Introduction

The motivation behind my project was to create a game that successfully demonstrated inner-task communication between two machines. My goal was to create Pong because it provided ample opportunity for two machines to communicate with one another. When the player presses a button the paddle should change in location. If the other player presses their button, their paddle should change in location. In either case the boards need to communicate their user's button presses so the game knows where the paddles are at.

When it came time to implement this project I realized that a lot of my time was going to have to be spent creating graphics. When I first proposed the idea I was not concerned about this because I assumed there were plenty of good libraries available to me. I did not want to reinvent the wheel in terms of recreating pong. That wasn't my motivation. I was using pong as my task so that two machines could communicate with one another. When I started coding I realized that a lot of libraries suggested to me were not commonly found in compilers. This also did not concern me because I have installed libraries in eclipse before. Yet when it came time to install a library, like SDL (Simple DirectMedia Layer), I found out it was not easy to include. Just to include that library I had to find other libraries to include. I then had to find the Linux version of these other libraries but sometimes these libraries could not be found or were redacted. In the end, in the interest of finishing my project on time I changed my game so that it would still require communication between two boards but wouldn't rely heavily on graphics.

The implemented project is a simple math game. When the user runs the program, it asks the user if it wants to host or join a game. If a user chooses host a game, then the board broadcasts a message requesting to play a game. If the user chooses join a game, then the board listens for broadcasts requesting to play a game. Once a host board finds a join board, the boards play the game. More information about the game being played and how they connect are provided in the background section.

Background

The game implemented for this project was various math questions. A math question comes up that has five different choices. The users have to answer the problems as quickly as possible. The board chosen as host creates the problems. After two boards connect with one another the host board sends the question and the choices to the join board. Once the join board receives the questions it sends back to the host board the message “play”. The boards then display the question and waits for feedback from the user. The boards time how long it takes for the user to answer the question. The boards wait for both users to answer the question before presenting a new question. After all the questions are answered the user’s answers are compared to the actual correct answers. It also finds the total time it took for the user to answer the question. Finally it displays the number of right and wrong answers and the total time took for each user.

To communicate with the boards I implemented the UDP and TCP communication protocols that were discussed heavily in class. The UDP protocol was used to find boards that wanted to play. UDP was chosen because it was connectionless. A user first and foremost wants to find another board to play. It doesn’t matter which board the other user is on. After the boards find one another the communication switches to the TCP protocol. This is because the user’s found one another and they now wish to play against one another. You do not want another board to interfere with the game thus handshaking between the boards is now needed.

Proposed method/System Description/Implementation

Below is the various structures and their data variable contents used to implement the game.

question	Int num1 Int num2 Int op Char op1 Int answer Char qs[SIZE]
choices	Int num1 Int num2 Int num3 Int num4 Int num5 Char strnum[SIZE]
problem	Question q Choices c Struct timeval start, end Unsigned long time[2] Int right[2] Char response1[SIZE] Char response[SIZE]
player	Int sock,n,host,boardnum,opponentbrd,p,play,osock,host_r,host_w,clinet_r,clie nt_w

```

Struct sockaddr_in client, server, p, b
Char buffer[SIZE], cmp[SIZE],
send[SIZE],cpy[SIZE],strnprob[QUESTION_SIZE]
Problem problem[NUMPROBLEMS]
Unsigned long host_t, client_t

```

A few remarks should be mentioned about these structures. For the player structure the sockaddr_in variables p and b were used for the UDP connection protocol. These names were originally chosen to stand for player and broadcast. Player UDP contained the IP of the board that user is on. While broadcast contained the IP to connect with all the other boards. When these names were chosen no TCP connection was implemented. I later added the TCP connections because it was an oversight not to include TCP connections. I forgot that the messages being sent could be lost using only UDP connection. The sockaddr_in variables client and server variables are the TCP connection protocols used by the host board. The UDP sockaddr_in variables are used to find the two IPs of the boards wanting to play. The TCP sockaddr_in variables used the information found from the UDP protocols to play the game between the two boards. The int variable sock is used by both join and host boards for UDP connection while the osock int variable is used only by the host board. Any other questions about these structures are provided in the comment section of my code.

A table of the functions I created are shown below along with their uses and what file they are in.

Function	Use	File
generate_problems	Used to create the problems for the game	generate_problems.c
choices generate_choices	Sends back the choices for each question	generate_problems.c
generate_results	Creates the response strings that will be sent	generate_problems.c
error	Error message when sending and receiving messages using UDP and TCP protocols	game.c
displayResults	Prints the results of the game	game.c

	locally to the host board and sends the results of the game to the join board	
playGame	Used to play the game	game.c
setLobby	Establishes the TCP connection for the game before playGame function can being. It calls playGame upon completion	game.c
setUsr	Starts the UDP connection for the game. It is used by both hostGame and joinGame	game.c
hostGame	Calls generate_problems to create the problems to be play. It establishes the UDP connection between the two boards by making the host board the master. It broadcasts a message requesting to play a game with one of the join boards. It calls setLobby upon completion	game.c
joinGame	It finds a game to join by listening for broadcast to play a game. It establishes the UDP connection between the two boards by making the join board the client. It calls setLobby upon completion	game.c
menu	Displays the menu for main	game.c
main	Decides if the board is the host or join board. C	game.c

Experiments and Results

When the user first runs the program. The following message pops up.

Welcome to Math Game

Do you want to

1) Host Multiplayer Game?

2) Join Multiplayer Game?

4) Exit?

If the user selects exit the game ends. If the user enters 1 or 2 for host game or Join Game the following prompt is shown.

<Hosting a Game or Joining a Game>

User Information: board 19 ip 10.3.52.19

User is Set

Main first calls setUsr to create the UDP connections needed to find the game. It gives the second and third line above. After that if the user enters Host Game main calls hostGame, if the user enters Join Game main calls joinGame. The following prompt will be shown to the user.

Problems generated (if the user selected Host Game)

Playing against board 19 on IP 10.3.52.19 (both)

The two lines shown above are shown once the boards establish UDP connection between one another. The UDP connections are used to find the boards' IP. Once both these functions are finished setLobby is called. Below is the output shown for setLobby.

Creating Lobby (if Host Game) Joining Lobby (if Join Game)

Ready to play the game

The line Ready to play the game is shown once the users establish TCP connection in the setLobby function. It calls the playGame function once it finishes. Below is a snippet from a game played. <Any lines in these brackets are the user input.>

Question 3)

What is the value of $11 * 20$?

1) 218

2) 220

3) 212

4) 211

5) 223

<Client chose 2> <Host chose 3>

After the users enter selections the playGame calls getResult. GetResults goes through each question for the user, Below is the output given by GetResults at questions 3.

Question 3)

What is the value of $11 * 20$?

1) 218

2) 220

3) 212

4) 211

5) 223

Client:211 is wrong.

The correct answer is 220

Host:212 is wrong.

The correct answer is 220

At the end of getResult it gives the final results for the program. Here is the output from this

particular game.

Totals

Client Right: 2	Client Wrong: 8	Total Time: 1867797
Host Right: 1	Host Wrong: 9	Total Time: 31329137

A complete output file for this game is provided. The file is called output. The debugging process is discussed in the Discussion and Conclusions part of this report. My program was able to play the game. If one user enters a number before the other user, it would wait for that user who has not selected an answer before proceeding to the next question. To make sure the program was working properly I had a lot of printf statements (that were later commented out) that printed the strings that were to be sent between the two boards.

Discussion and Conclusions

The procedures used to figure if the results were right worked fine. A few problems came about during the connection part of my code. I didn't give myself enough time to test my code. I never tried running my project with multiple users joining or hosting.

In total I ran the program around 50 times upon completion of the code. The program would run around 50% of the time. The reason for the poor results was because of some faulty code I could not fix within my setLobby function for my joining board. This was due to the TCP connection and the port used for the connecting. I tried debugging this part but was unable to find a proper solution. If a failed connection occurred I would switch the port number being used for connection.

Overall I am disappointed with my project. I spent too much time originally trying to make the pong game. I also was never able to implement a controller. I was too late to realize how superfluous designing the graphics were in regards to what this project was going for, which were real time concepts from this course. The point of using pong was that it was such a dynamic game that could really show real time inner-task communication between the two boards I could have not done a game and instead chose something else. If I could redo this project I would have tried to learn the concepts from lab five and six way ahead of time. At the time of proposal we had yet to start these labs. They were instrumental to completing my project and if I would have taught this material to myself ahead of time I would have not had been in such a predicament of completing this project on time. I would have come up with a better game than the one implemented, as this game does not show the concepts from those labs as well as different game could have. Finally, I should have gone for something more hardware based than what I chose.