Minimising Lag in Game Networking Models Without a Central Server

Implementation and Analysis of Client-Hosted and Peer-to-Peer
Networking Models in the Context of Games.

Szymon Jackiewicz
Student Number: 150249751
Supervisor: Dr Graham Morgan

Word Count: ???

MComp Computer Science w Games Engineering
Stage 3

Department of Computer Science
Newcastle University
18 February 2019

Declaration

"I declare that this dissertation represents my own work except, where otherwise stated."

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Minimising Lag in Game Networking Models Without a Central Server

Implementation and Analysis of Client-Hosted and Peer-to-Peer Networking Models in the Context of Games

Szymon Jackiewicz

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contents

Chapter 1:   Introduction

Throughout this project, I will be exploring the different methods of providing a synchronised, multiplayer gaming experiance on two or more computers on a network. I will be covering the networking basics of how two game clients running on seperate machines can communicate with each other as well as exploring the relevant details of how and why the networking systems in games are designed the way they are.

I will explore the existing methods of writing an online multiplayer game from scratch and aim to provide an implementation of a networking library for online games to communicate without a central server. I also aim to provide analysis of efficieny of different methods of achieving this goal.

1.1   Basic concepts and Terminolory

First of all, I will define some basic concepts and terminolory that will be used throughout this document.

Client: Within the context of this document, a client can be defined as a piece of software responsible to running the game (i.e. game client). A client can also be defined as a computer interacting with a server, however it is possible to run two different instances of a game client on a single machine.

Online Multiplayer Game: A video game can be defined as a simulation of a certain scenario that can be manipulated by the player of the game. When talking about online multiplayer games, it can be thought of as a simulation that runs on several clients connected by a network (e.g. LAN or the Internet) that is to be synchronised. When one player performs an action that effects the state of the simulation, this action should also be seen by all participants of this perticular simulation instance and therefore the simulation should remain in the same state across all participating clients.

Ping: In network connections, the ping between different clients refers to the shortest amount of time that is needed for one client to send information to another and receive a response from this client. One client sends a "ICMP echo request" to another networked client (e.g. a game server). The receiving client, then responds with an "ICMP echo reply" back to the original device. The time between sending the request and receiving the reply, is the ping between the two clients.

Lag: The grater the ping between two connected clients, the bigger the difference in the state of each clients' simulation once an action to be synchronised is performed. When a change is made by one client, this change should be seen by other clients participating in the same simulation and lag occurs when this change does not appear instentaneous to the user.

Jitter: TODO

1.2 Networking principles in games

TODO: talk a bit about UDP, TCP/IP and routing...

1.2.1 The Centralised Server Model

Most online multiplayer games that are played today, make use of the "centralised server model" for sinchronising the simulation state between several clients participating in the same simulation. This means that in an example of a First Person Shooter (FPS), if one player presses the "jump" key, their character will jump and this information is would also be sent over to the game server. The server would then send the information that this player has jumped, to all other clients. There is a potential problem here however. Given that the ping between the server and client A is α and between the server and client B is β, the time between client A pressing an input and client B being notified of this input can not be less than $\alpha + \beta$ and due to the limitations of physics $\alpha > 0$ and $\beta > 0$. This means that at any given time the lag experianced between clients A and B will be more than $\alpha + \beta$ when processing times are factored in too. This leads to a problem of a poor experience for a client with a high ping to the server, as they will receive the updates from the server later than every other player and therefore be at a disadvantage if the game requires real time reactions. Unfortunately under some implementations, this also results in a poor experiance for every other player, who despite having resonable ping to the server, can be shot from behind cover by a laggy player who fired a shot before the cover was reached.

Possible solutions to the variable ping problem

TODO: talk about region locks, and disconnecting players with high ping but ping can be high even in normal conditions

Battelfield 1® implementation

An interesting approach to this issue has been implemented by DICE in the game Battelfield 1. Given 2 players; player A with a low ping to the server and player B with a ping of <150ms to the server.

When player B fires at a moving player A, player B's client will perform the check concluding that player A has been hit and this information is sent to the server. The server will then perform it's own checks and if the server agrees that this hit is possible, then it sends the hit confirmation to player B and damage information to player A. This approach is called Clientside-Server Authoritative as while the hit registration is calculated on clientside, the server must still confirm that this is valid.

Concidering another scenario, suppose that player A still has a low ping to the server but player B, now has the ping of >150ms. An icon will appear on player B's UI showing an "aim-lead" indicator. Now when the shot is fired in the same scenario, the hit will not register anymore as the hit registration has switched from Clientside-Server Authoritative to Fully-Server Authoritative, meaning that the check is performed only once the shot information is received by the server.

Whilst this implementation makes the game feel less responsive for players with high ping, it provides a lot more fairness for everyone else and allow for players with different pings play in a more fair way.

Chapter 2: Research

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2.1   Implementation

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2.2   Security

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Chapter 3:   Design

3.1   Code Architecture

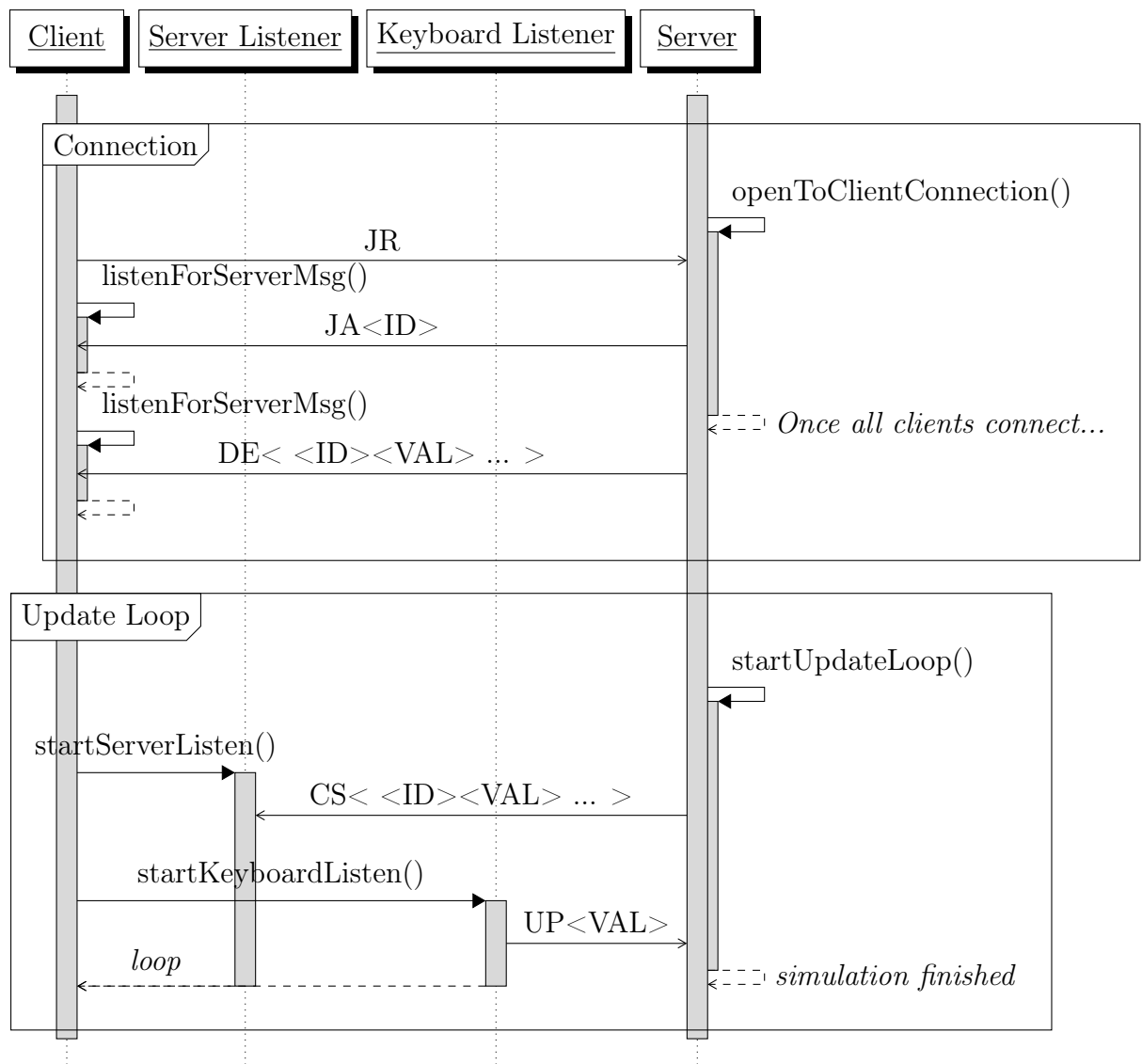Test for inline: `test Hello;`

3.2   Protocols

Figure 3.1: Graph showing the protocol of joining a session hosted on the server as well as sending and receiving updates.

3.3   Message Codes

Message Type	Message Code	Description	Example Payload
Join Request	JR	Allows a client to send a join request to the server.	JR
Join Acknowledgement	JA	Allows the server to confirm that the client's information has been saved. Is followed by 1 byte indicating the client's ID	JA1
Ping Request	PQ	Message instructing the recipiant to reply with PS. Can be used to time the delay in this connection.	PQ
Ping Response	RS	This should be sent whenever a PQ message is received.	RS
Update	UP	Used by a client to update it's value on the server. Is followed by 1 byte representing the new value.	UP9
Define	DF	Used by the server to define the initial values for each of the clients connected to this instance. It is followed by a non-zero, even amount of bytes representing the client ID and it's value pair.	DF1020304050
Current State	CS	Used by the server to broadcast it's real state to all clients. When this is received, clients are expected to update their local state to this. It is followed by a non-zero, even amount of bytes representing the client ID and it's value pair.	CS1927344157

Table 3.1: Table showing the message codes for distinguishing messages from each other and how each one is to be used

Appendix A: Appendix Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.