A decidir

Tiago da Silva Guerreiro and Lucas Costa dos Prazeres

Abstract

Keywords

I. Introduction

II. Technical Background

This article's multiplayer game is designed in Javascript, using Node.js in backend, and Socket.io to establish Websocket communication.

A. *Websocket*

Websocket provides bidirectional full-duplex communication over a single TCP connection. Unlike HTTP, the websocket connection remains open until terminated by the server or client, which reduces the number of handshakes. Since the connection is already open, data transmission is faster.

The websocket technology is crucial to applications that require low latency, such as multiplayer browser-based games.

B. *Node.js*

Node.js is a server environment that enables Javascript on the server side, without a browser. It uses an event-driven, asynchronous I/O, which makes Node.js capable of processing various requests simultaneously, and consequently, highly scalable.

Node.js provides a low cost, highly scalable environment for server-side applications. This technology is very useful to browser-based multiplayer games, since the server has to handle multiple requests from clients.

Special thanks to professor Eduardo Cerqueira, ITEC, UFPA, Belém-PA, e-mail: cerqueira@ufpa.br

C. *Socket.io*

III. GAME CHANGES

IV. METHOD

One essential network requirement of a multiplayer game is the ability to handle many connections (from many players) as gracefully as possible, while providing a good real-time game experience. This is why it is important to observe how time-related parameters behave on such applications, like delay or latency. This article manages to study how this temporal characteristic responds to linear variations on the number of active connections in a small experiment.

The experiment consists of a series of matches played with different numbers of clients and a single server hosting the game. The server is a common laptop running *Ubuntu Linux* with the backend game instance listening on localhost on port 3000. Since localhost applications usually are not acessible to hosts on different LANs, we used a port forwarding tool called *ngrok*, which generated a tunnel using a USA located server to generate a public url to access the game server. Then, a few regular matches were played by clients located in different LANs - unlike the previous study [], in which the matches were played in a single LAN - while a native node.js function called *process.hrtime()* collected temporal benchmarks in background showing the average time it took for the server to respond to every player move. The results are presented in the next section.

V. RESULTS

VI. CONCLUSION

[1]

REFERENCES

[1] B. Chen and Z. Xu, "A framework for browser-based multiplayer online games using webgl and websocket," in *2011 International Conference on Multimedia Technology*. IEEE, 2011, pp. 471–474.