A Network Extension for GameMaker HTML5

Teun Kokke

Undergraduate Dissertation
Software Engineering
School of Informatics
University of Edinburgh

2016

Abstract

summarising the report

COMPLETED

- planning the project
- general research on the topic, ie. answer the question: "where to start?"
- developing the implementation
- develop application using the working extension to test if it works (both locally and online)
- evaluation: assessing the limits of the implementation (given a specific server specification). Both locally and online (different distances to server, different cities? countries? continents?)
- write networking extension implementation
- execute experiments to evaluate the implementation

TODO

February

- give definition for server-reponse time
- in section "fairness depending on location", add world map with tested locations and their average mean
- Consider actual playability in a game OR add assumption how RTT affects atual usage of an application (find reference?)
- write literature review and development section
- transform details of experiment setup in text format (+ write additional details if details are missing)
- Write down details about implementation, design decisions must be well justified according to literature
- Possibly add new implementation features / clean up existing implementations.
- explain the evaluation results

March

- clarify thesis and write intro
- Record video of working implementation
- Create proper template for other developers to use + writing down a guide how to use it, where to start etc.
- Release the project to the community

- Finalize report

Acknowledgements

First of all, I would like to thank my supervisor Dr. Myungjin Lee for guiding me through the process of writing the report and giving feedback of my work.

I am also grateful to those people across the globe who have assisted me with testing and collecting data for the evalutation experiments.

My sincere gratitude to my family, friends, the Dutch Gamemaker community and the Yoyogames forums, for providing me with feedback and ideas without which this project would not have been the same.

List of Acronyms

- API - Application Program Interface
- IP - Internet Protocol
- TCP - Transmission Control Protocol
- UDP - User Datagram Protocol
- W3C - World Wide Web Consortium
- P2P - Peer-to-Peer
- RTT - Roundtrip Time
- RSS - Resident Set Size
- CPU - Central Processing Unit
- RAM - Random Access Memory
- LAN - Local Area Network
- GUI - Graphical User Interface
- IDE - Integrated Development Environment

Table of Contents

Chapter 1

Introduction

In the current day and age, we spend a large portion of our time using web applications. A vast amount of the internet consists of services supported by web applications, and browser games are as popular as ever.

There exist many good reasons for this. Browser applications and games do not require prior installation. They are therefore easy to start up, safe from viruses and don't require an admin-user account in order to be executed[1]. They are able to interact with other web applications. They are highly platform independent and potential users are generally easy to reach. Also their updates are seamless and can be implemented without requiring patch downloads to a harddrive.

The market in this area is therefore booming. Developers all across the world are trying to be the fastest at developing their games, in the easiest way possible. The easier the process, the faster the development. The faster the development, the sooner the game can be released.

One of many developer tools that aims for exactly these two ideals is GameMaker Studio. However due to the feature limitation of creating networked applications, developers may be forced to use less suited software instead.

This paper therefore investigates the quality of related software. It also demonstrates how an extension can be added to GameMaker, one that will add networking features to HTML5 games and applications. Additionally, the server's behaviour will be assessed based on the specified setup, and suggestions will be added based on other literature in order to improve this further.

Chapter 2

Background and Related Work

2.1 Background

2.1.1 Networks in Applications

Humans are naturally social beings, are found to be more drawn into games when this social aspect is being provided, and are therefore more likely to return in the future[2]. When allowed, we can share a sense of community, develop our own social identity within the application, and regularly seek social support from peer users, even about non-related and real-life topics[3]. Users of networked applications are **more likely to advertise** the application in their social circles than they are for non-networked applications.

2.1.2 Network Fairness

Having that said, one must consider the technical aspect of the network: if the application is in the form of a game, **playability and fairness are crucial for an enjoyable gameplay**. This is especially true for games containing elements where speed or response time is important[4].

In a typical networked game, game clients are described by a **limited set of parameters** received by a server. These parameters represent the "game state". **When due to delay between the clients and server the game state is desynchronised, fairness is reduced**[4].

2.1.3 TCP versus UDP

TODO: explain what is tcp and udp before mentioning which one is better when. because.. what are they? what do they do?

Choosing the right protocol to handle the data transmissions is therefore crucial. Transmission protocols are one of the contributors that will heavily characterise the network.

Generally speaking, TCP is a form of reliable data transmission but contains therefore extra overhead, making it therefore known to be slightly slower than UDP when the network has little packet loss.

2.1.3.1 Socket.io

Socket.io is an event-driven JavaScript library that can be used both on the client's browser, and a server[5]. It supports features that allow sending and receiving data using the WebSocket protocol built in 2011 (which is handled through TCP), without interruption of the code flow[6][7]. For this reason, it is **often used in combination with Node.js**.

2.1.3.2 WebRTC

As aforementioned, Websocket is a protocol built on TCP. However, it has a little sister: WebRTC. WebRTC built on UDP, but is still vaguely "under construction"[8]. Although certain web applications have already been created with WebRTC (mainly for P2P video streaming [9]), no proper standards are out yet[1].

This, combined with requirement of manually specifying the rules of communication, as well as the fact that WebRTC has to circumvent network security and privacy in order to allow web browsers to transmit data over UDP[9], makes UDP many times **more complicated** for developers to handle efficiently.

2.1.4 HTML5

2.1.4.0.1 HTML5 is a raising web standard released in 2014 by the W3C. It is designed to be **cross-platform** and runs on most modern web browsers such as Google Chrome, Mozilla Firefox, Apple Safari, and Opera [1]. Also mobile web browsers that come preinstalled on iPhones, iPads and Android phones support HTML5.

It superseeds its predecessors HTML4 and XHTML1.1 with the aim to reduce the dependence of functionality from third-party plugins such as Flash and Java applets, which are either deprecated or entirely unsupported by most devices [10].

Scripting is replaced in HTML5 by markup where possible, causing the world of browser-gaming to change rapidly. One of the the newly introduced features is the <canvas> element, which is defined as "a resolution-dependent bitmap canvas which can be used for rendering graphs, game graphics or other visual images on the fly"[11]. **The element can thus be used to draw graphics in JavaScript with the "Canvas API"**[12].

[1] HTML5 supported browsers are found at https://html5test.com/results/desktop.html

2.1.5 Node.js

Traditionally, servers create a separate thread for each client, therefore rapidly running out of RAM and keeping clients on hold until memory for a new thread is released[13].

Node.js is a JavaScript interface with the aim to create "real-time websites with push capability" allowing developers to work in the "non-blocking event-driven I/O paradigm" [13]. This means that developers can use it to create real-time web applications where a server and client can both initiate communication, and that both can exchange data freely without repeatedly having to refresh the webpage.

In short, new client connections get allocated to a heap in the memory and client events are handled on a single thread by the server's operating system without choking the (Node.js) event loop. **This therefore allows servers running Node.js to maintain thousands of concurrent connections without running out of RAM memory**[14][15], as opposed to the traditional, less scalable servers.

2.1.6 GameMaker

GameMaker by YoYoGames is a software creation tool with the aim to simplify and speed up game and app development. There have been several hits on the market for games developed with Gamemaker such as "Reflections", "Rick O'Shea" and "Simply Solitaire" [16].

Developing applications and games in GameMaker is cheap, simple to learn and flexible to use, making the software demanded by small teams, professionals and novice developers [17]. Sandy Duncan, the founder and former chief executive officer of YoYoGames stated in a phone interview that they have never lost money on a game that they developed with their technology[16].

During the rise of HTML5 and the growing popularity of Gamemaker, YoyoGames has provided the functionality to export any application to a JavaScript program that can be executed directly in the browser[18].

Some of the features are normally supported by GameMaker are however lost during the transition to a web application. One of these features is the networking functionality. Thus far since the update to export GameMaker applications to HTML5 in September 2011, YoYoGames has never included this feature[2]. Several attempts have been made by the YoYoGames community, however all known versions are considered to be in alpha stage and are regularly found to fail to be used by other developers[3].

The main GameMaker community forum is hosted directly by YoYoGames[4], serving 280.000 registered users recorded in January 2016, with a combined total of 220.000 topics and 3.480.000 posts. There also exists a notable GameMaker forum in the

[2]http://docs.yoyogames.com/source/dadiospice/002_reference/networking/index.html
[3]https://github.com/amorri40/39js
[4]http://gmc.yoyogames.com/

Netherlands[5] which hosts an additional 557.000 posts for 56.000 topics for almost 18.000 members.

2.2 Related Work

Being a cross-platform game development tool primarily for 2D graphically oriented games, GameMaker has many competitors. It is therefore sensible for developers to first consider using game developing tools that support networking features natively. Although, looking for "the best" development tool is unreasonable, as this is a matter of personal preference. Before being able to investigate pros and cons between gamemaker and related game development tools, appropriate competitors must first be identified.

Note that GameMaker does natively support networking functionality for non-browser games. However we must remind ourselves of the aim of this project: creating a networking extension to improve GameMaker's functionality **for developing 2D browser games**. For this reason, this paper will consider GameMaker to be a development tool for browser games.

In order to find the most related software for these purposes, assistance was found from online third-party instances. These instances collect user feedback for the browser-game development tools. Their ratings were established by allowing users to criticise tools based on their personal experience, and apply a score. The review system of html5gameengine.com[6] represents that of the Google Play Store[19] and the Apple App Store[20], and may therefore be a somewhat reasonable method for finding the more commonly used tools.

developer.mozilla.org[7] (world rank 198 by alexa.com[8]) hosts a list of tools titled "HTML5 game engines", and holds suggestions by 26 developers each with different backgrounds in 2D browser game development. gamepix.com[9] hosts a list of similar tools, but is less detailed about their sources.

Table 2.1 was carefully constructed by combining the best rated 2D browser-supported game developing tools as advertised by the communities for the instances mentioned above.

[5] http://www.game-maker.nl/forums/
[6] https://html5gameengine.com/
[7] https://developer.mozilla.org/en-US/docs/Games/Tools/Engines_and_tools
[8] http://www.alexa.com/
[9] http://www.gamepix.com/blog/the-big-list-of-2d-html5-games-engines/
[10] Advertised at developer.mozilla.org: developer.mozilla.orghttps://developer.mozilla.org/en-US/docs/Games/Tools/Engines_and_tools
[11] Advertised at http://www.gamepix.com/blog/the-big-list-of-2d-html5-games-engines/

Tool	Score	Voters	Mozilla[10]	Gamepix[11]	Native GUI
Pixi.js	5.0 / 5	50	yes	yes	no
Phaser	4.5 / 5	129	yes	yes	yes
EaselJS	4.5 / 5	63	no	yes	no
Crafty.js	4.5 / 5	18	yes	yes	no
Construct 2	4.0 / 5	136	yes	yes	yes
Quintus	4.5 / 5	29	no	yes	no
Unity	n/a	n/a	yes	yes	yes

Table 2.1: Some of the highest rated and most promoted 2D browser-game development tools

Tool	Topics	Community main page
Pixi.js	1455	http://www.html5gamedevs.com/forum/15-pixijs/
EaselJS	738	http://stackoverflow.com/questions/tagged/createjs[12]
Quintus	n/a	https://plus.google.com/communities/104292074755089084725
Crafty.js	1666	https://groups.google.com/forum/#!forum/craftyjs
Phaser	8793	http://www.html5gamedevs.com/forum/14-phaser/

Table 2.2: Displaying the community activeness for each of the mentioned tools, along with their corresponding location.

2.2.1 Pixi.js, EaselJS, Quintus, Crafty.js and Phaser

Despite Pixi.js, EaselJS, Crafty.js, Phaser and Quintus being legitimate tools for developing games, they are merely JavaScript frameworks and libraries to assist developers when creating raw JavaScript games. Therefore, if networking is expected to be part of a browser game, these tools will also require additional resources such as WebSocket for implementing this feature.

Their community is relatively small and scattered across the web. The forums for these tools are hosted on third-party websites and make it therefore difficult to measure their size and activity accurately.

Although the five tools assist in the development, they do not compare with GameMaker as their user base is many times smaller. A GUI is also not provided, forcing developers to use a standard programming IDE.

2.2.2 Construct 2

Construct 2 is one of the main contenders. It provides its own GUI and shares similar ideals as those of gamemaker: simplifying the development for novice programmers. This is done by supporting inbuilt D&D features. Code programming can also be used after installing a required extension. It is a generally well known tool with a community

[12] Statement of community moving to stackoverflow: http://community.createjs.com/

of over 204.000 registered users, responsible for more than 529.000 posts for 103.000 topics[13].

In April 2014, release r164 to r168 updated the engine with networking features by allowing users to create a said "network object"[14]. This object is pre-defined with "features to develop real-time online multiplayer games", provided using WebRTC DataChannels (UDP) for P2P connections.

It is a very well established update with optimisation support included, eg. by providing preliminary setups for "NAT traversal to connect through common router/network setups", "Interpolation and extrapolation modes to ensure smooth in-game motion", "Support for lag compensation" and more. Administrator at Scirra, Ashley G. states however that "even though Construct 2's Multiplayer object takes care of many of the complexities, ..., it is important to understand how multiplayer online games fundamentally work"[21].

2.2.3 Unity

In the world of browser-game development, Unity is by far the most acknowledged tool. It comes with a professional GUI, allowing developers to create games using D&D as well as with programming languages C# and JavaScript[15].

Networking features are supported as a D&D class[16], but can also be used by installing the "WebSocket-Sharp" plugin in C#[17].

Currently there are roughly 4.500.000 users registered to the Unity forum[18], however only a small percentage of this is active in 2D development as this only hosts 5600 topics[19].

This is most likely due to the fact that Unity is aimed towards experienced 3D game-developers and offers much overhead when only using its 2D capabilities. In the poorly populated 2D section of the forum, replies such as "have you considered using GameMaker?"[20] are therefore a common response.

https://unity3d.com/public-relations

4.500.000 registered users

http://forum.unity3d.com/forums/2d.53/

5,595 topics

licenses http://unity3d.com/unity/licenses cost https://store.unity3d.com/products/pricing

[13] https://www.scirra.com/forum/
[14] https://www.scirra.com/manual/174/multiplayer
[15] http://docs.unity3d.com/ScriptReference/index.html
[16] http://docs.unity3d.com/ScriptReference/Network.html
[17] https://github.com/sta/websocket-sharp
[18] http://forum.unity3d.com/forums
[19] http://forum.unity3d.com/forums/2d.53/
[20] http://forum.unity3d.com/threads/where-to-start-2d-toolkit-or-platformer-pro.380770/

Chapter 3

Literature Review

3.1 Getting Users to Advertise Your Application

literature related to networking applications in order to connect users such that they will want to invite their social circles to the game.

3.2 Fairnesss and Playability in Online Multiplayer Games

more detail on Network Fairness, causes and fixes to unfairness (see background section)

3.3 Multiplayer Networking in Modern Game Engines

game state maintained through a limited set of parameters

3.4 Details Why WebRTC is Complicated to Handle Efficiently

how does WebRTC work, why it is not suggested for the implementation and why it otherwise would

3.5 Drawing Graphics with the HTML5 Canvas API

basics to drawing graphics in html5 <canvas> element

3.6 Creating Extensions for GameMaker Studio

how to create an extension to GameMaker

Chapter 4

Development

4.1 Design

4.1.1 Prior Considerations

4.1.2 Server and Client

4.1.2.1 Good Coding Practices

4.1.2.2 Interaction

4.2 Implementation

4.2.1 Server

4.2.2 Client

4.2.3 The Extension

4.3 Applications Developed with the Extension

4.3.1 Benchmark Application

4.3.2 Real Game Application

4.3.3 Developer Template

Chapter 5

Network Extension Evaluation

5.1 Setup

5.1.1 Controlled Network Experiments

The controlled experiments were conducted in order to predict server behaviour when loaded under similar pressure in future occasions.

5.1.1.1 Concurrent Connections Specifics

5.1.1.1.1 Environment

- Variable number of concurrent connections cc, up to 15 instances with each simulating at most 500 clients.
- Clients do not contact the server after establishing a connection.
- The CPU usage, time and RSS (Resident set size) are recorded after each every time another group of 100 clients connect to the server.

5.1.1.1.2 Dependent variables

- CPU usage on the server
- CPU time on the server
- RSS on the server

5.1.1.2 Message Broadcasting Specifics

5.1.1.2.1 Environment

- 5000 concurrent connections (10 instances each simulating at most 500 clients).

- Each package that is sent has a size of 8 bytes.
- Each client sends packages at regular time intervals, causing the server to handle n messages every second.
- Each client measures the roundtrip time of its package to the server.

5.1.1.2.2 Dependent variables

- Mean roundtrip time between all clients
- Variance of the roundtrip time between all clients
- Standard Deviation of the roundtrip time between all clients

5.1.1.3 Server hardware and network specification

5.1.1.3.1 The controlled network experiments were executed on a Windows 7 64-bit platform with 12.6GB available physical memory and an Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz processor. The software included Node v0.12.3 and Socket.io v1.3.7 in order to handle the networking operations.

The network was controlled using Dummynet, using a below-average UK household network setup. This involves an upload speed of 5Mbit/s, and a download speed of 1Mbit/, although no packet loss was set in order to ensure consistency throughout the controlled network experiments.

5.1.2 Real Network Experiments

The network was consistently running with a 54.0Mb/s download and 3.0Mb/s upload speed.

5.1.2.0.1 Multiple tests were executed with clients being located at specified locations. In each case, the clients were sending 8-byte messages to the server at a regular interval of 1 second for 2 minutes. The 120 roundtrip times for each client were then averaged in order to blur occasional peaks. At this point the roundtrip times of the clients in each common location were averaged, and then the deviance of the roundtrip times at each of these locations were calculated.

All network experiments were executed using a single server located in Edinburgh and in each experiment all clients were connected and communicating to that server simultaneously.

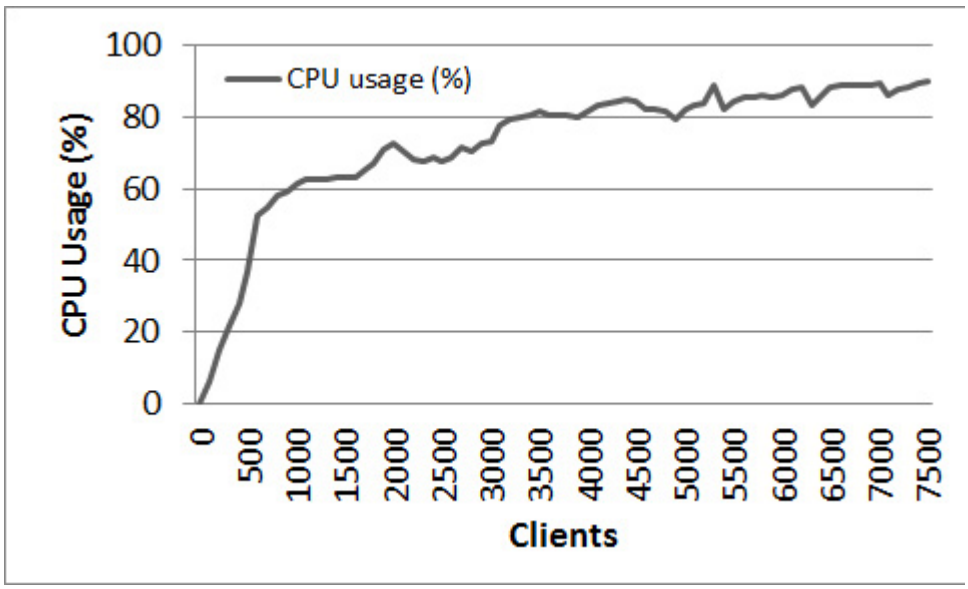

Figure 5.1: Displaying the CPU usage for the server process in percent.

When the CPU usage is above 70 percent, the user may experience lag. Such high CPU usage indicates insufficient processing power. Either the CPU needs to be upgraded, or the user experience reduced.

5.2 Results

5.2.1 Controlled Network Results

5.2.1.1 Concurrent Connections

The following experiment evaluates the server performance with regard to the number of clients.

5.2.1.2 Message Broadcasting Performance

The following experiment evaluates the number of messages that can be handled by the server simultaneously, and considers how this affects the fairness in response-time of the individual clients.

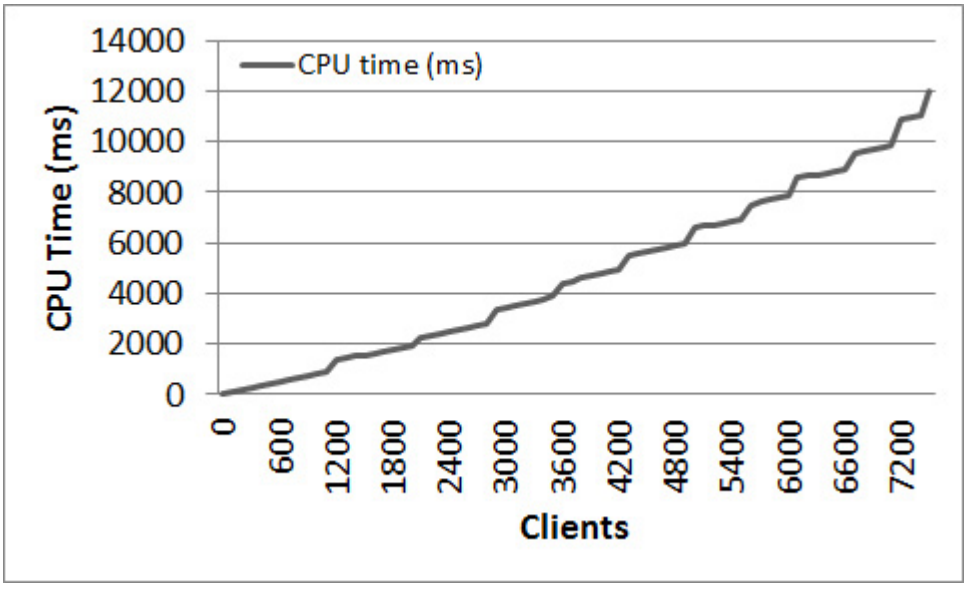

Figure 5.2: CPU time in milliseconds, displaying the amount of time required for the server to process the clients.

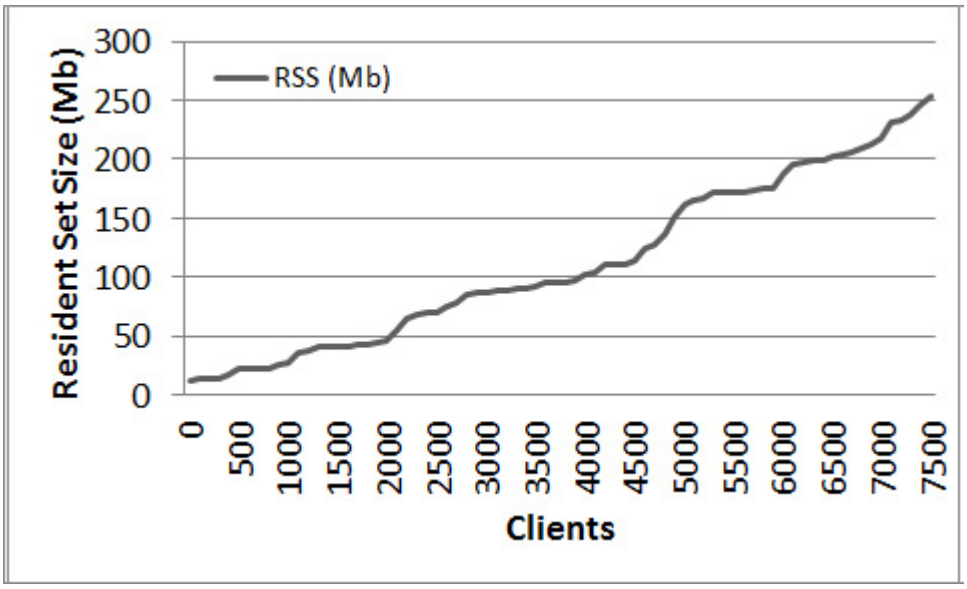

Figure 5.3: Resident set size in Megabit, showing the portion of RAM that is occupied by the server process.

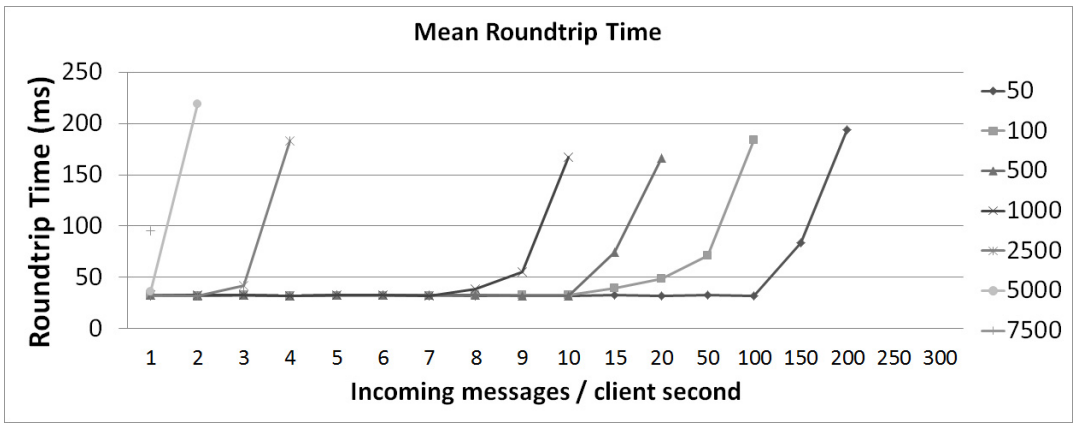

Figure 5.4: The mean roundtrip time of all the messages that pass through the server. As expected, with few concurrent clients connected to the server, the server manages to broadcast many more messages.

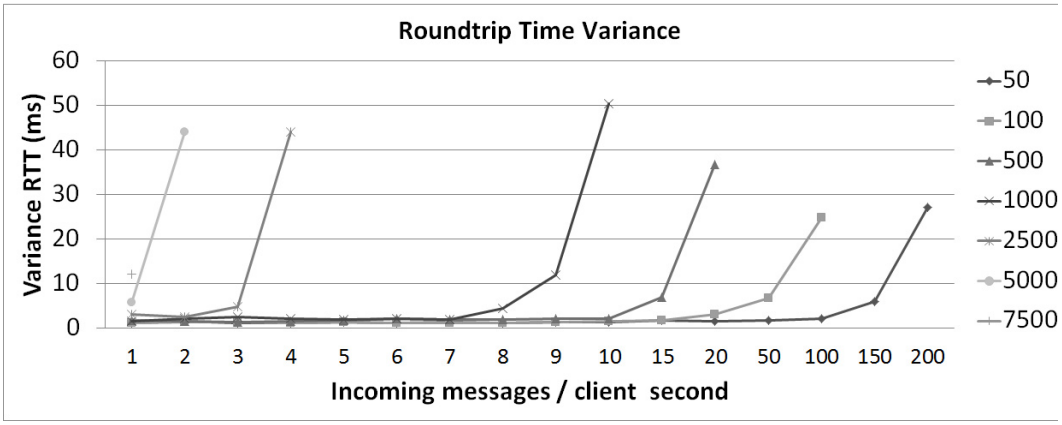

Figure 5.5: The variance of the roundtrip times, showing the fairness between clients decreases significantly when the server receives messages faster than it can broadcast.

5.2.2 Real Network Results

5.2.2.1 Location-wise Delay Fairness

The following experiment evaluates the effect of the geographical distance between groups of clients and the server with respect to fairness in response-time from the server to the clients.

5.2.2.1.1 Test cases:

1. Local network setting: Five clients physically located in the same local home network.
2. Same city: Five clients physically located in Edinburgh (LAN excluded).
3. Same country: Three clients physically located in Scotland: Edinburgh, Glasgow and Dundee.

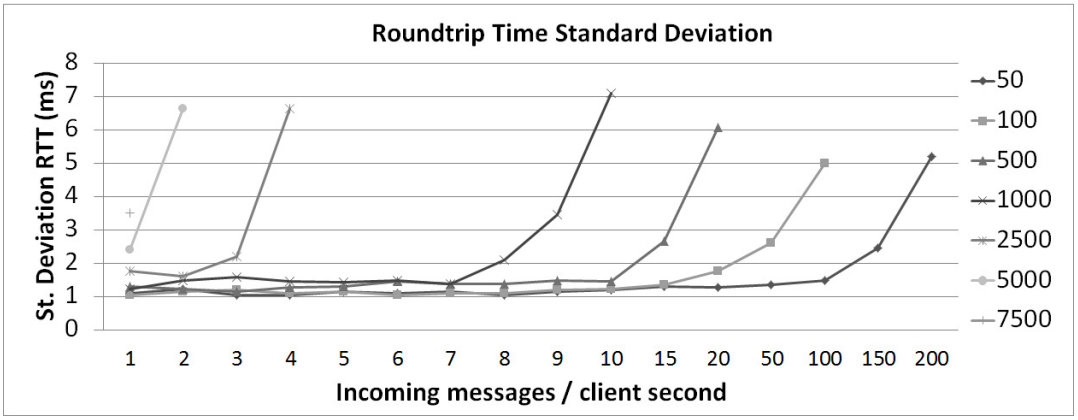

Figure 5.6: Standard deviation of the message roundtrip times.

4. Europe: Five clients physically located in the United Kingdom, Hungary, France, Germany and the Netherlands.

5. Inter-continental: Eight clients physically located in South Africa, California (USA), India, Thailand, Germany, Hungary, United Kingdom and the Netherlands.

5.2.2.1.2 Results: Average roundtrip times per location:

Location	Average RTT
LAN	32ms
Edinburgh	55ms
Dundee	65ms
Germany	67ms
Glasgow	68ms
France	69ms
the Netherlands	70ms
United Kingdom	71ms
Hungary	76ms
India	103ms
South Africa	144ms
California (USA)	158ms
Thailand	171ms

Average roundtrip times per test:

Test case	1	2	3	4	5
Mean RTT	32.1ms	54.8ms	62.7ms	70.6ms	107.5ms
Variance RTT	1.3ms	8.2ms	46.3ms	11.3ms	1900.9ms
Std RTT	0.4ms	2.9ms	6.8ms	3.36ms	43.6ms

Chapter 6

Conclusion and Future Work

6.1 Conclusion

6.1.1 Comparison of the Extended GameMaker Functionality with Related Work

6.1.2 Criticism on the Implementation and Design Decisions

6.2 Future Improvements

[22] [4] [23] [11] [24] [17] [10] [13] [14] [15] [6] [7] [5] [12] [1] [3] [25] [2] [8] [1] [9] [16] [18] [26] [27] [19] [20] [28] [29] [30] [21]

Bibliography

[1] Vinny Lingham. Top 20 reasons why web apps are superior to desktop apps. http://www.vinnylingham.com/top-20-reasons-why-web-apps-are-superior-to-desktop-apps.html, 2007. [Accessed: 2016-01-16].

[2] Christoph Klimmt, Hannah Schmid, and Julia Orthmann. Exploring the enjoyment of playing browser games. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 12(2):231–234, 2009.

[3] Erin L. O' Connor, Huon Longman, Katherine M. White, and Patricia L. Obst. Sense of community, social identity and social support among players of massively multiplayer online games (mmogs): A qualitative analysis. *Journal of Community & Applied Social Psychology*, 25(6):459–473, 2015.

[4] Jeremy Brun, Farzad Safaei, and Paul Boustead. *Fairness and playability in online multiplayer games*. PhD thesis, University of Wollongong, 2006.

[5] socket.io. http://socket.io/. Homepage, [Accessed: 2016-01-16].

[6] Drew Harry. Practical socket.io benchmarking. http://drewww.github.io/socket.io-benchmarking/, 2012. [Accessed: 2016-01-16].

[7] Mikito Takada. Performance benchmarking socket.io 0.8.7, 0.7.11 and 0.6.17 and node's native tcp. http://blog.mixu.net/2011/11/22/performance-benchmarking-socket-io-0-8-7-0-7-11-and-0-6-17-and-nodes-native-tcp/, 2011. [Accessed: 2016-01-16].

[8] Ilya Grigorik. *High Performance Browser Networking*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2013.

[9] Martin Kirkholt Melhus. *P2P Video Streaming with HTML5 and WebRTC*. PhD thesis, Norwegian University of Science and Technology, Trondheim, June 2015.

[10] Ian Elliot. Death of flash and java. http://i-programmer.info/news/86-browsers/8783-death-of-flash-and-java.html, 14 July 2015. [Accessed: 2016-01-16].

[11] Mark Pilgrim. *HTML5: Up and Running*. O'Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2010.

[12] Mozilla Developer Network. Canvas api. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API, 2015. [Accessed: 2016-01-16].

[13] Tomislav Capan. Why the hell would i use node.js. http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js, 2014. [Accessed: 2016-01-16].

[14] Mike Pennisi. Realtime node.js app: A stress testing story. https://bocoup.com/weblog/node-stress-test-analysis, 2012. [Accessed: 2016-01-16].

[15] Alejandro Hernandez. Init.js: A guide to the why and how of full-stack javascript. http://www.toptal.com/javascript/guide-to-full-stack-javascript-initjs. [Accessed: 2016-01-16].

[16] Thomas Claburn. Gamemaker promises drag-n-drop game creation. *Informationweek*, May 2012.

[17] Mark Overmars. Teaching computer science through game design. *Journal Computer*, 37(4):81–83, April 2004.

[18] Llopis Noel. Gamemaker studio v1.1.7. *Game Developer*, 20(1):40, January 2013.

[19] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh. Why people hate your app: making sense of user feedback in a mobile app store. *KDD '13: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 13:1276–1284, 2013.

[20] Jin Hyung Kim, Inchan Kim, and Ho Geun Lee. The success factors for app store-like platform businesses from the perspective of third-party developers: An empirical study based on a dual model framework. *PACIS Conference Proceedings*, 2010.

[21] Ashley Gullen. Multiplayer tutorial. https://www.scirra.com/tutorials/892/multiplayer-tutorial-1-concepts/page-1, March 2014. Scirra, creators of Construct 2.

[22] Alan Edwardes. Multiplayer networking in a modern game engine. Project Writeup, 2014.

[23] Peter Lubbers, Brian Albers, and Frank Salim. *Pro HTML5 Programming*. Paul Manning, Springer Science and Business Media, LLC., 233 Spring Street, New York, 2010.

[24] Nathaniel S. Borenstein. *Programming as if People Mattered: Friendly Programs, Software Engineering and Other Noble Delusions*. Princeton University Press, Princeton, New Jersey, 3rd edition, 1994.

[25] L Suarez, C Thio, and S Singh. Why people play massively multiplayer online games? *International Journal of e-Education, e-Business, e-Management and e-Learning*, 3(1), 2013.

[26] Alessandro Alinone. Optimizing multiplayer 3d game synchronization over the web. *Lightstreamer*, October 2013.

[27] Eric Li. Optimizing websockets bandwidth. *Multiplayer*, September 2012.

[28] Jason Lee Elliott. *HTML5 Game Development with GameMaker*. Packt, 35 Livery Street, Birmingham B3 2PB, UK, 2013.

[29] Barry W. Boehm. Seven basic principles of software engineering. *Journal of Systems and Software*, 3(1):3–24, March 1983.

[30] Klaus Pohl, Gunter Bockle, and Frank van der Frank. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, Springer-Verlag Berlin Heidelberg, 2005.