Networking Extension

For GameMaker Studio HTML5 Export

Recall

GameMaker simplifies computer game development,

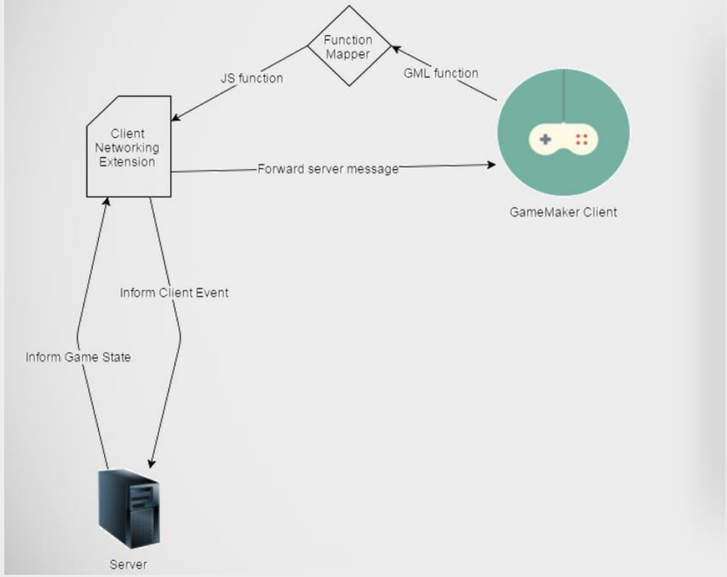
but does not provide networking support for browser export

Recall

GameMaker simplifies computer game development,

but does not provide networking support for browser export

Solution: Networking Extension



Technologies:

- NodeJS
- Socket.io

Related Tools

Developer “Favourite” Poll:

[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

https://html5gameengine.com/

Related tools

Pixi.js, EaselJS, Quintus, Crafty.js and Phaser:
- Commonly used developer tools and frameworks for JavaScript browser games
- No GUI provided
- Dependent on additional libraries for networking functionality

Construct 2:
- Strong competitor overall, supporting similar features as GameMaker
- Networking feature supported, but requires proper understanding of networks

Unity:
- Generally designed for 3D games
- Complicated to learn
- Large overhead when creating 2D games

Related work: 39js

Previous attempt for creating a networked GameMaker browser application

However:

- Scripted implementation, thus lacking maintainability
- Does not represent game state, but only mediates messages back and forward
- No use of design patterns
- Complicated to update once a project becomes large
- Untested “alpha stage”
- Performance metrics not included

Own Development

Socket.io details are abstracted away from: developers do not have to think of “how do networks work?”.

- A **singleton** central controller maintains the logic, assigning tasks to client instances depending on the newly received message.

- A **client** class holds public methods for updating client states and handling messages. (ia. *connect()*, *disconnect()*, *send_message()*, *update()* and *return_ping()*)

- Use of the **module pattern**, keeps the data for the client instances private. This data is synchronized with that of the real client. *(ia. username, x, y, mouse_x, mouse_y, hspeed, vspeed, room_name)*

Evaluation: Server Specification

Hardware

Windows 7 64-bit platform

12.6GB available physical memory

Intel(R) Core(TM) i7-2600K CPU@3.40GHz

Software

Node v0.12.3

Socket.io v1.3.7

Network

- Local: 54.0Mb/s download, 3.0Mb/s upload
- Dummynet: 5Mb/s download, 1Mb/s upload

Evaluation: Concurrent Connections

Impact on server memory (RSS), CPU, and its affect on the latency

- How many concurrent connections does the server support?
- How many messages per second per client can the server handle? ..and for how many clients?

Evaluation: Concurrent Connections

Impact on server memory (RSS), CPU, and its affect on the latency

- How many concurrent connections does the server support?
- How many messages per second per client can the server handle? ..and for how many clients?

Setup

- 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.
- Data is recorded after each every increment of 100 connected clients

Evaluation: Concurrent Connections

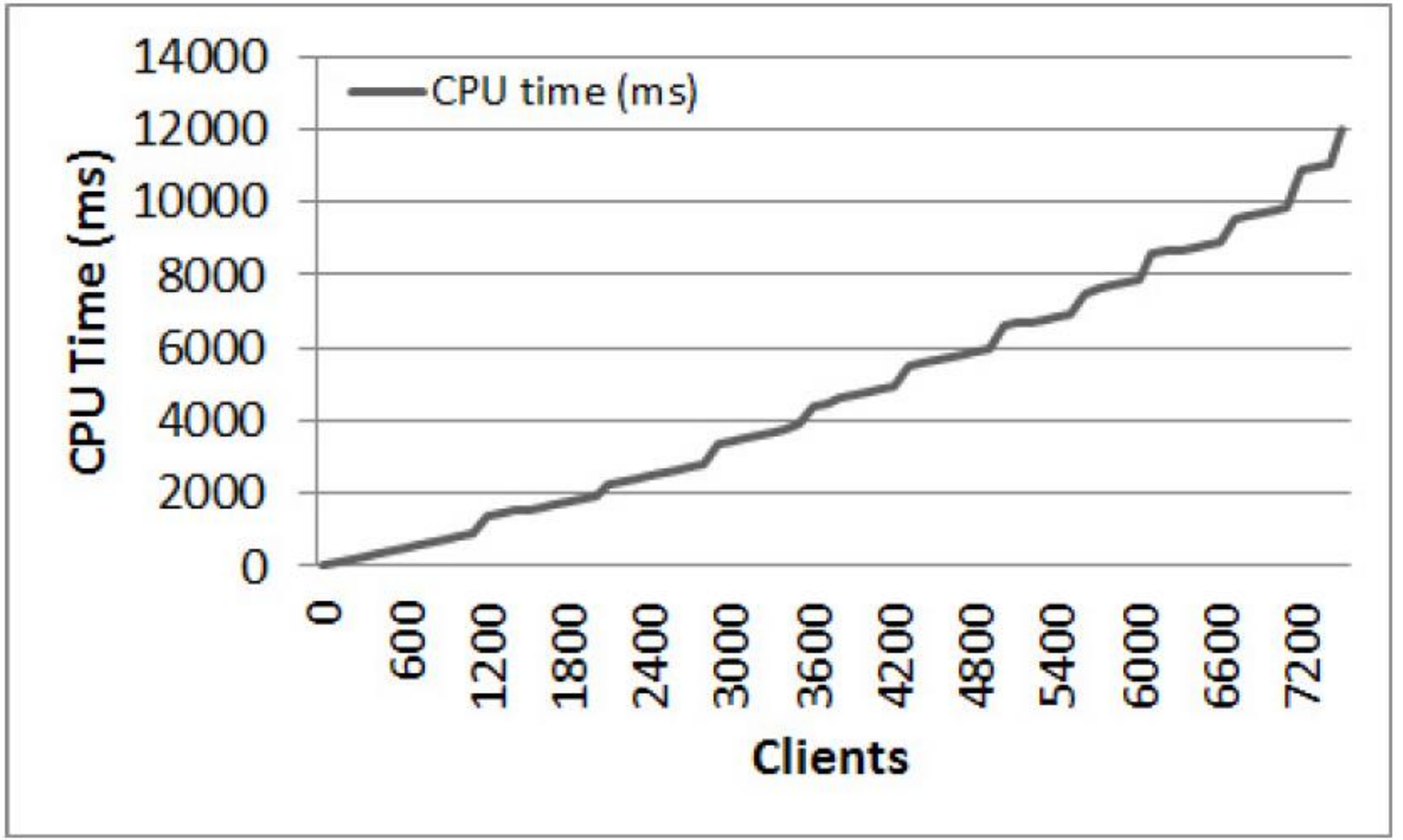


Figure 5.2: CPU time in milliseconds, displaying the amount of time required for the server to process the clients.

Evaluation: Concurrent Connections

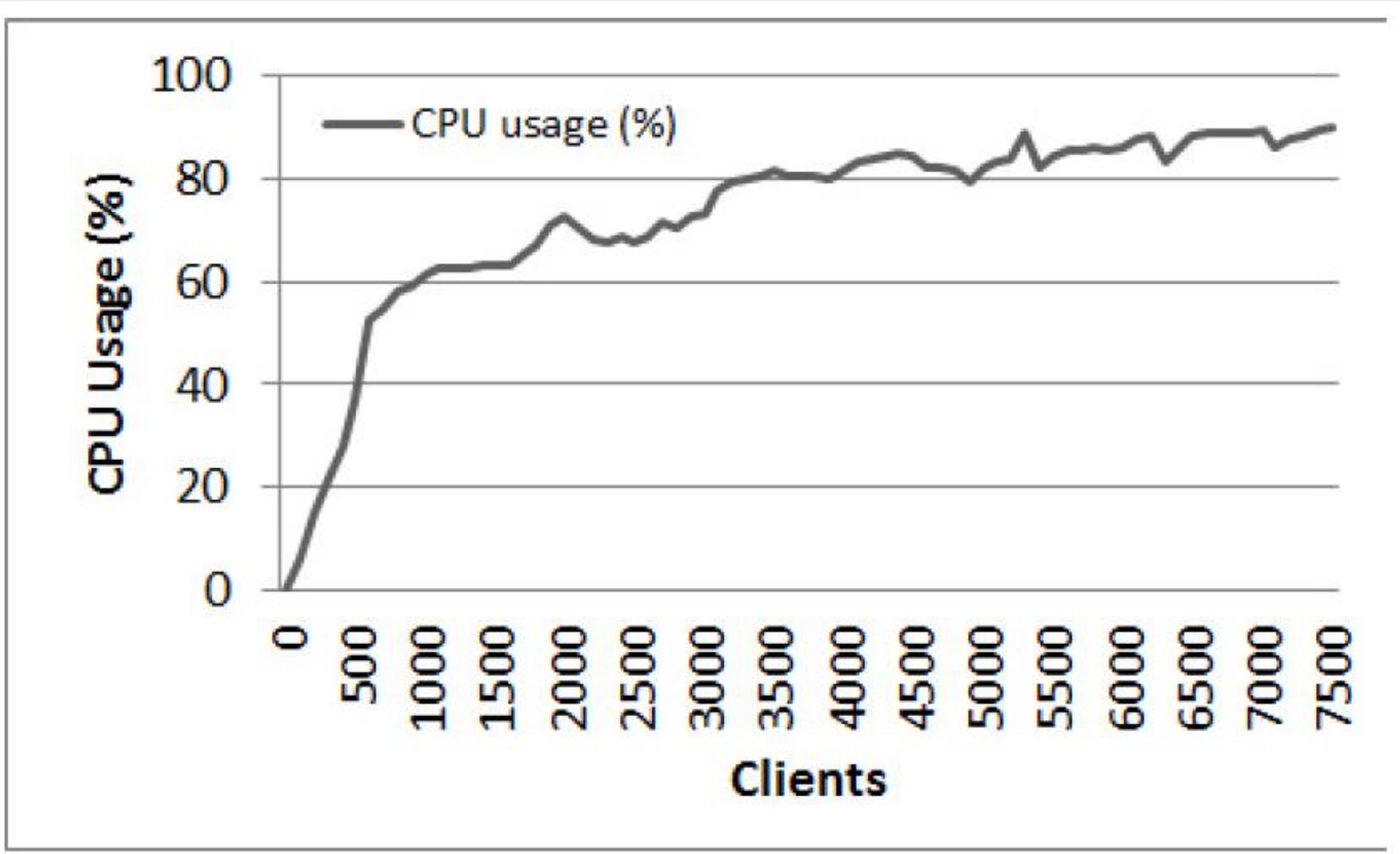


Figure 5.1: Displaying the CPU usage for the server process in percent.

Evaluation: Concurrent Connections

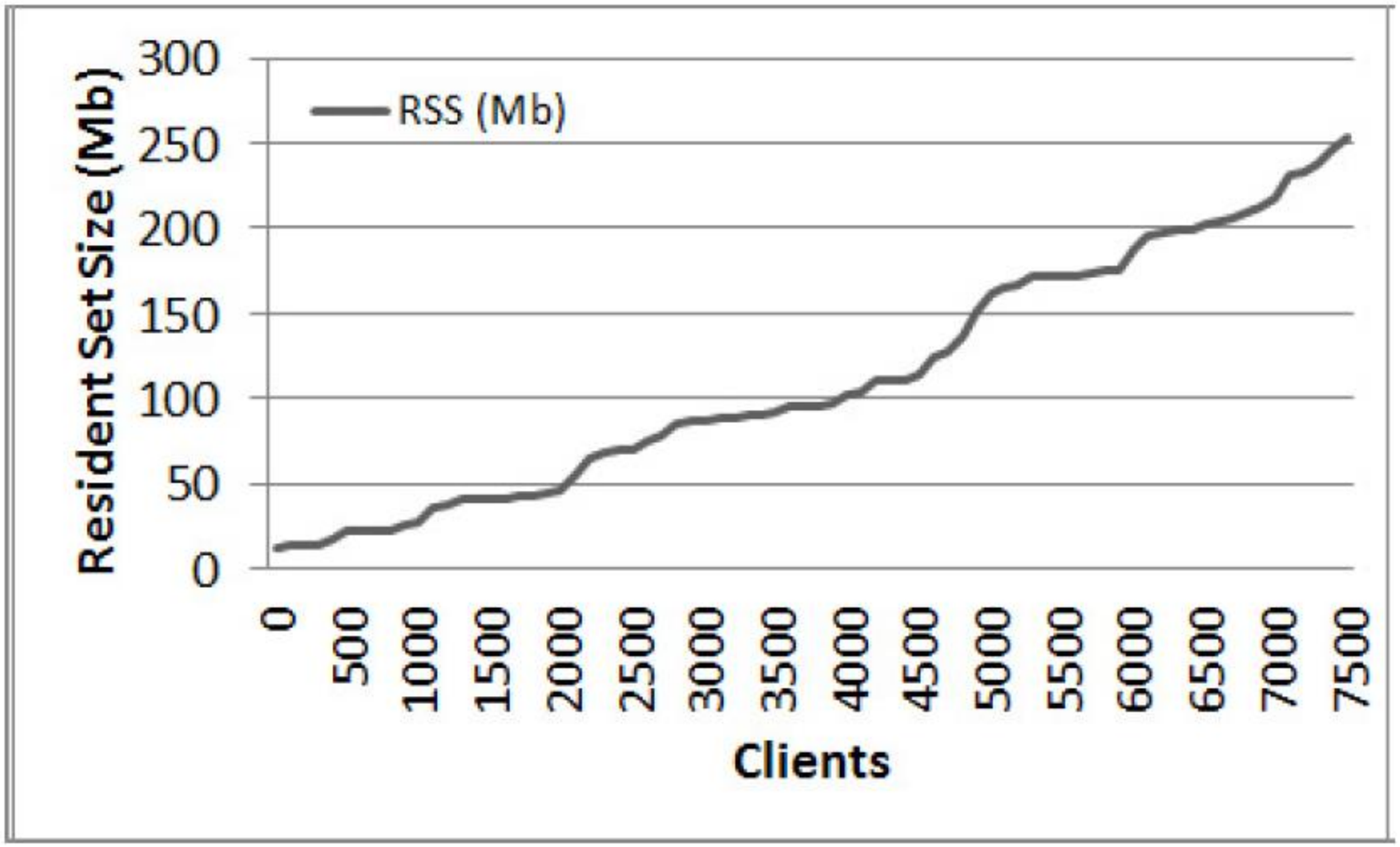


Figure 5.3: Resident set size in Megabit, showing the portion of RAM that is occupied by the server process.

Evaluation: Message Broadcasting

Impact on server memory (RSS), CPU, and its affect on the latency

- How many concurrent connections does the server support?
- How many messages per second per client can the server handle? ..and for how many clients?

Setup

- N clients simultaneously connected to the server
- Each client attempts to broadcast x messages per second
- experiment terminates when RTT extends the 200ms threshold

Evaluation: Message Broadcasting

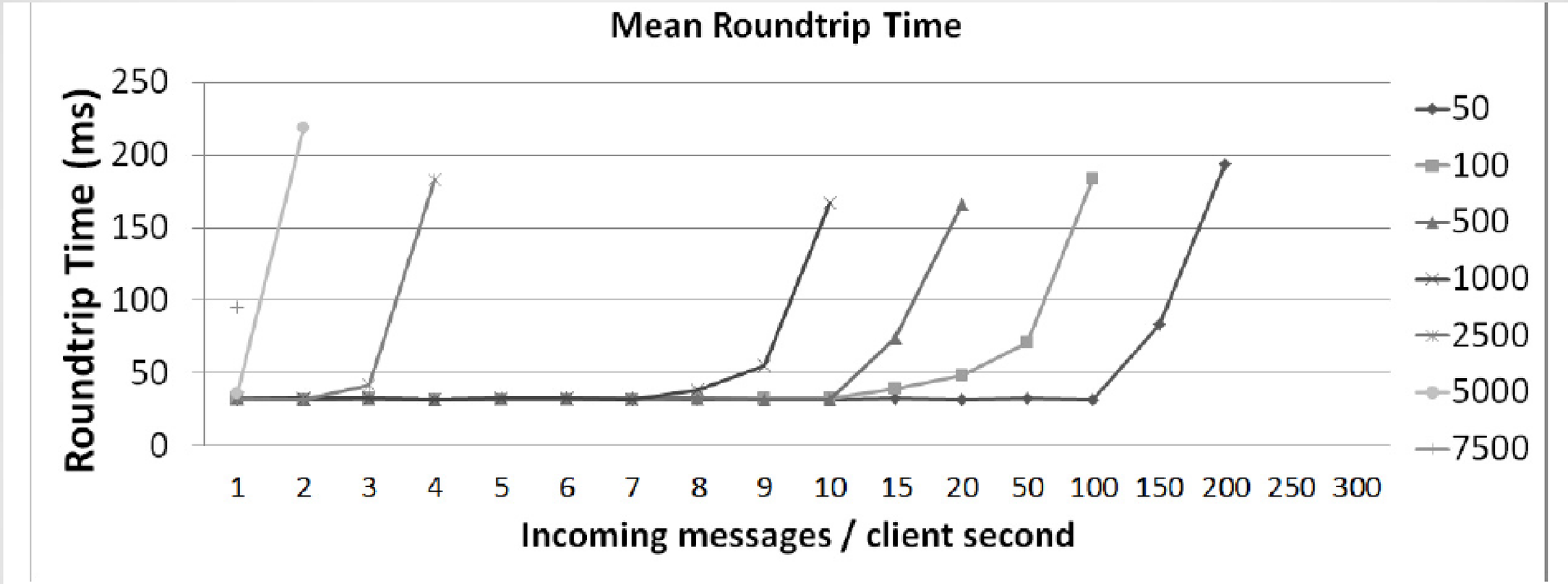


Figure 5.4: The mean roundtrip time of all the messages that pass through the server. As expected, with few concurrent clients connected to the server, the server manages to broadcast many more messages.

Evaluation: Message Broadcasting

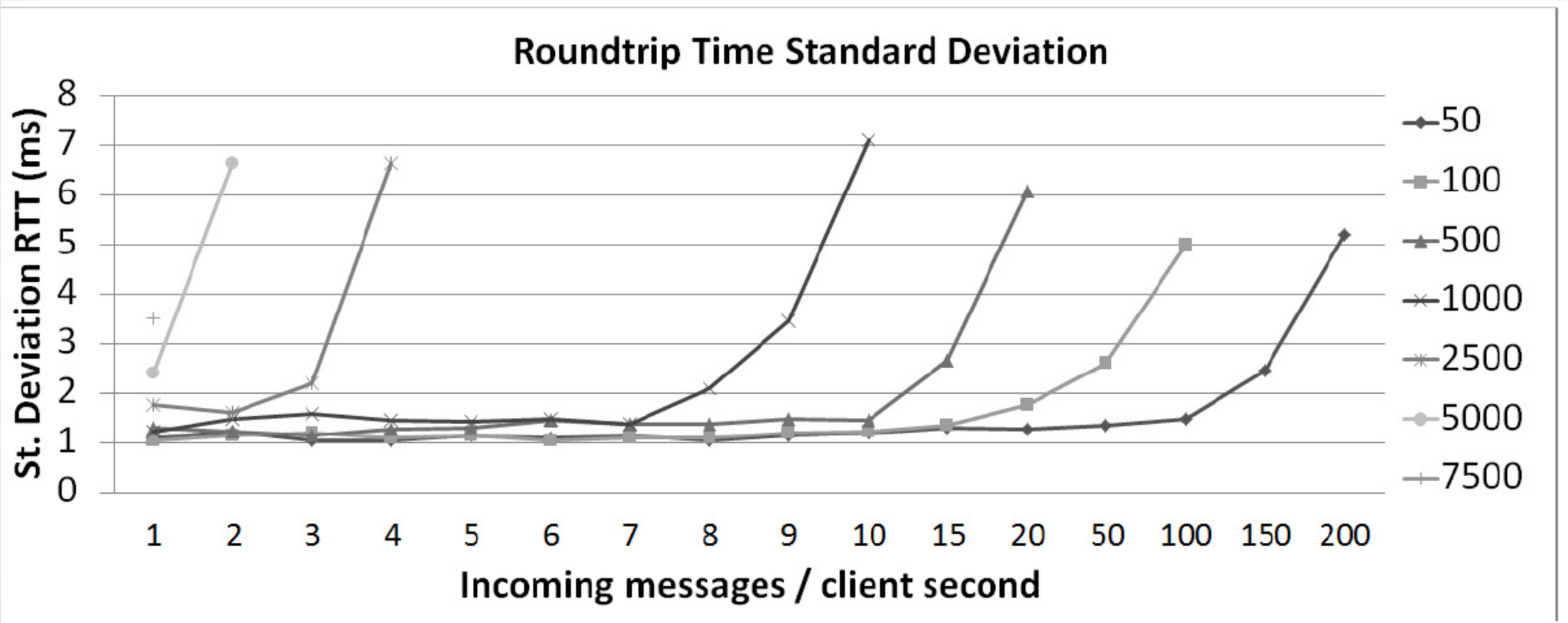


Figure 5.6: Standard deviation of the message roundtrip times.

Evaluation: Location-wise fairness

Latency difference depending on client-server location variability

- How does distance affect the latency between the server and its clients?

Evaluation: Location-wise fairness

Latency difference depending on client-server location variability

- How does distance affect the latency between the server and its clients?

Setup

- Clients send 8-byte messages every second for 2 minutes
- The message latency is averaged per 2 minute intervals
- The server was located in Edinburgh (UK)

Evaluation: Location-wise fairness

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

Evaluation: Location-based fairness

Latency difference depending on client-server location variability

- How does distance affect the latency between the server and its clients?

Fairness considering clients at different locations:

1) LAN
2) Edinburgh
3) UK
4) Europe
5) World

Evaluation: Location-based fairness

Latency difference depending on client-server location variability

- How does distance affect the latency between the server and its clients?

Fairness considering clients at different locations:

1) LAN
2) Edinburgh
3) UK
4) Europe
5) World

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

Conclusion

- Evaluations return **latency** values within **commonly accepted range** for gaming purposes within most areas of the world.

- Each client takes up a relatively **small, constant amount of memory** in the server. (roughly 20MB for 5000 clients)

- The server **processor usage** follows a **logarithmic trend** wrt. the number of clients

- The server has a **clear threshold** at which it can **broadcast** messages reliably. Beyond this point, it needs to buffer messages to the next round, recursively making the situation worse.

- **Distance** strongly **affects** the **network quality**

Solutions

- **Distance**: spread servers locations, let clients act as server (peer-to-peer).
  - Next problem: there won't be a central "game state" anymore

- **Scalability**: The server reliably supports up to 1000 clients, each broadcasting 10 messages per second. 2D browser games do not usually expect this load. Install more servers, assign clients to a specific server (common solution)

Timeline

Complete:

- Planning
- Realisation
- Research
- Implementation
- Develop Application
- Benchmarking
- Evaluation
- Dummynet
- Comparison

To-do (NOW):

- Code Optimisation
- Create example templates for other developers
- Insert extension in large scale project (?)
- Finalising the report

?

Thank You For Lıstenıng