Assignment 4

FIT3140

Semester 2  
2018

Jake McNee  
+  
Hugh Dickeson

## Introduction

**JSON**

JSON information via socket.io sends simple objects that are received via ports given in Socket.IO, within our application, the servers are defined by a port and socket ID that get handed to a larger umbrella of service that is listening on the port8000, all of the other servers are listening on other ports that are randomised to connect with clients. The information that is handed between sockets is always in JSON, the point of JSON is that it is human readable and separated by pairs of relevant tags for usage.

**Protobuf**

Protocol buffering (Protobuf) is a different format that is used to pass data between clients or servers. Its binary nature allows the data to be serialised such that long chunks of information can be passed between servers at a more efficient rate. Protobuf is not language dependent and can be implemented to most common languages. In this case Node.JS was used for its implementation. In large cases the smaller size of the serialised format can greatly reduce space needed in messages to server which can grant some improvements in speed. The messages are coded/decoded using a .proto file, which gives the user set encryption key to the server for processing.

# Testing

**Simple Working Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Function** | **Average of 10 (time)** | **Average of 10 (size)** |
| Json | Add new block (small) | 7.268ms | 22 |
| Protocol | 6.260ms | 186 |
| Json | Add new block(large) | 8.234ms | 22 |
| Protocol | 11.234ms | 2124 |
| Json | Add New Server (after Minimal Transactions) | 7.650ms | 22 |
| Protocol | 9.231ms | 184 |
| Json | Add New Server (after large number of transactions eg:100) | 12.282ms | 22 |
| Protocol | 17.826ms | 232 |

The testing aims to highlight the differences between sending information through sockets in JSON and proto-buffering. The functions that we chose were supposed to show that the size of the inputs and structures would make a drastic effect on the performance of the application implementations, although the results were not what were were anticipating

The time was measured in js console using a comparison of Date.now() in an average of 10 instances per function

The size was measured as JSON.stringify(data).toString().length and recorded as an integer from the console.

## Design

The major idea of decentralised applications is that the decisions aren’t being made by one server. In this case there is a ‘server’, on the same port number, that directs the decision making to a different server which will relay a result back that may or may not be broadcast to each server (and then subsequently to the client whom sent the request).

Each server listens on a port between 3000 and 3999 for clients wishing to make a transaction, upon such a request the message is sent out, and directed to a server that has a proportional weight to the other servers that are connected, the transaction is verified via traversing each node in the block chain that is on the weighted server for 2 things, firstly that there have been no breaks in the hashing of each block (for example a broken chain) and that the funds are there for the new transaction, you cannot spend coins that you do not own. (the server is paid for this transaction).

In the event that the transaction can occur, a new block is added to each chain (each server) that contains the information of the transaction, and the client is notified of their successful endeavor. Or the failure if that is the case.

If a new server joins while there already exists a chain, a proportionally chosen server will transmit the entire chain to the new member.

## Data structure

Block chain structures were introduced for this application, the concept is simple that each block validates the block in either direction (if one exists) so that the entire chain cannot be broken.

In this version of the code there is only one chain that holds all of the transactions but I can imagine that the concept is infinitely scalable to have more chains based on different transaction types seeing as the data within the chain could even be a block chain of its own if it was necessary.

Within this spike the JSON packets are being sent via sockets (in one spike) that contains the data of the next block.

The protocol buffer data structure is implemented in a way which aims to minimise the size of data being sent between socket functions. Complimenting the source code is a messages.proto file which is the key to encoding and decoding data. The user customises this as needed such that the data structures used within the code match the ones given in the .proto file. This allows the data to be encoded into a serialised format which can be processed quicker once transactions are big enough.

The remainder of the data structures used were simple lists, dictionaries, and integers.

## Foreseen Issues

There is a correlation between the amount of blocks and the time that it takes to validate, obviously, should the chain become long enough so that the transactions take longer than they are worth to validate at the same time that another request is made we might have issues regarding the validity of the chain with multiple transactions occurring while they are still being validated.

With the randomisation of the ports within secondServer.js there is the potential of clashing ports (1/1000) or if there are a thousand servers up then there will be nowhere to listen to. We have not allowed for this possibility within the code.

Js can only deal with a certain buffer amount unless specified, there may be a limit on how much can passed through Sockets via either protocol or JSON emitting. This means that there could/ should be a limit in place regarding description size and client names.

The starting amount of each client is specified within the BlockChain (in this version) having some way to determine if a transaction is valid would have to occur differently should clients be bringing their wealth from other locations.

With our testing comparisons we have not thought about the largest cases of chains being sent, without the proper time implementation regarding testing loads in the real world we do not know how this may affect the timing and size of each transaction in the long run.