TU060

Secure Systems Development

**Assignment 3**

**Bitcoin Explorer**

Ciaran Kieran

**D20125384**

School of Computer Science

Technological University Dublin

07/05/23



TABLE OF CONTENTS

[1.0 Project Overview 3](#_Toc134353882)

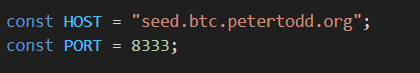
[2.0 Architecture 3](#_Toc134353883)

[3.0 Functionality 3](#_Toc134353884)

[4.0 Changes to be made 3](#_Toc134353885)

# 1.0 Project Overview

The project is the beginnings of a bitcoin explorer app. It is built in Node.js. The app works by accessing a bitcoin node which is currently hard coded in the app in the index.js file.



The aim of the project is to access the node and parse relevant data coming from the bitcoin blockchain.

The full plan of the app is yet to be executed. The app should establish the connection with the node and successfully communicate a series of messages to it in order to receive information coming from the about blockchain such as block and transaction information. This information should then be parsed by the app in a Node.js backend which can then be accessed by a React frontend to display the information in a browser.

However, at the moment, the app is not complete. It successfully establishes the connection to the bitcoin node and can receive messages such as *inv* payloads etc but it fails to successfully send *getblocks* and *getdata* messages to the node and parse the data appropriately.

The React frontend has also not been created. The app is only accessible by with working in the terminal with Node.js.

# 2.0 Architecture

The app is built in Node.js with a very simple file structure outlined as follows:

**index.js** – This is the part of the app that establishes the connection with the node and seeks to send it messages and parse its data.

**messageGen.js –** This file contains a series of message generator functions that generate messages to be sent to the bitcoin node. These functions are then imported into index.js and used within the *net.connect* function once the connection is established.

**messageParsers.js -** This file contains two functions:

* ***deserialiseMessage*** – Separates the command field of each message from the payload to help define which type of message is being sent.
* ***parseBlockHeader*** – This function is not yet operational because block messages have not been successfully accessed and parsed by the app from the node. The aim of the function is to log key data points about blocks added to the blockchain.

# 3.0 Functionality

The app successfully establishes the connection to the node. It also successfully sends through the version and *verack* messages. Once it does this, the node begins to send back messages. After being in operation for some time. After a wait of one or two minutes, the node sends through some *inv* payload messages as can be seen in the photos below.

# 

# As things stand the app cannot successfully send through a *getblocks* message to the node. Attempts were made to establish this functionality but something must be wrong with the *getblocks* message as things stand because once it is sent, the node immediately breaks communication so this functionality is currently commented out in the index.js file.

Upon receipt of the *inv* payloads, the app makes an attempt to gather the block hash from the payload and to send through a *getdata* message to the node based on the hashes of blocks included in the *inv* messages. However, this functionality at present has not been developed successfully yet and is incomplete in its approach. It is included in the app to provide the context for the intended implementation.

# 4.0 Changes to be made

The app needs a lot of work in order to successfully implement the project goals. Information points that need to be established include:

* Why is the *getblocks* message not being accepted by the node
* How can the *inv* messages be parsed successfully in order to gather block information such as the timestamp, nonce, hash and difficulty levels of the blocks.
* Once parsed, how can the information gathered about the blocks be successfully utilised in order to gather transaction data from the node

As well as gathering the information above and adapting the app to include the appropriate functionality, the following additional changes will ideally be made:

* Having parsed the relevant data, the app should establish a server which facilitates API calls from a React front end
* The React front end will need to be built and should be available across browsers
* The app should work in a loop which is updated with new information once it is sent through from the node.