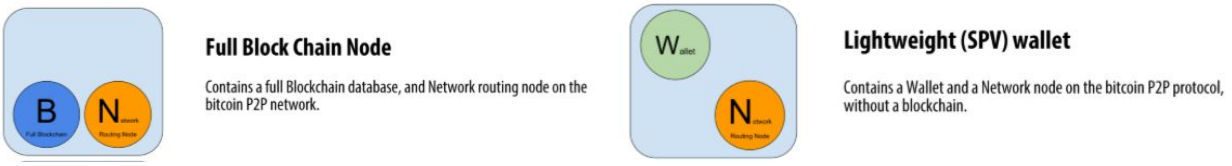
Bitcoin Specification Document

# Title

## Bitcoin Lightweight Node Senior Project (Working Title)

# Summary

This project's goal is to create a lightweight, thin node at Rowan University that will be able to access the blockchain and verify transactions through the Single Payment Verification (SPV) system. As this is a Lightweight Node, it will be missing one key feature that distinguishes it from a Full Node. Full Nodes consist of the entire blockchain starting with the Genesis Node, which is the first block that was created in 2009. A Lightweight Node contains a wallet to keep track of Bitcoins and record of block headers from the blockchain. The list of headers is 1000 times smaller than the blockchain and allows access to transactions on the blockchain through *Merkle Paths*. A Lightweight Node also contains a Network Node that is on the Peer-to-Peer (P2P) network of Bitcoin Nodes. This allows a Lightweight Node to send a neighboring Node an address and determine the depth of a block in the overall blockchain. If the other Full Nodes on the network have accepted this transaction as part of the blockchain; the Lightweight Node believes it does exist. The Lightweight Node sends this information through the use of *Bloom Filter* for privacy protection.




Full blockchain vs Lightweight Nodes, Photo Credit: *Mastering Bitcoin, Antonopoulos*

# Goals

The goal of this project is to successfully implement a Lightweight Node at Rowan University that resides on the Bitcoin P2P Network and allows for safe, secure transaction look-up. The group also wishes to create a display at the Rowan University STEM Student Research Symposium on April 21st related to Bitcoin.

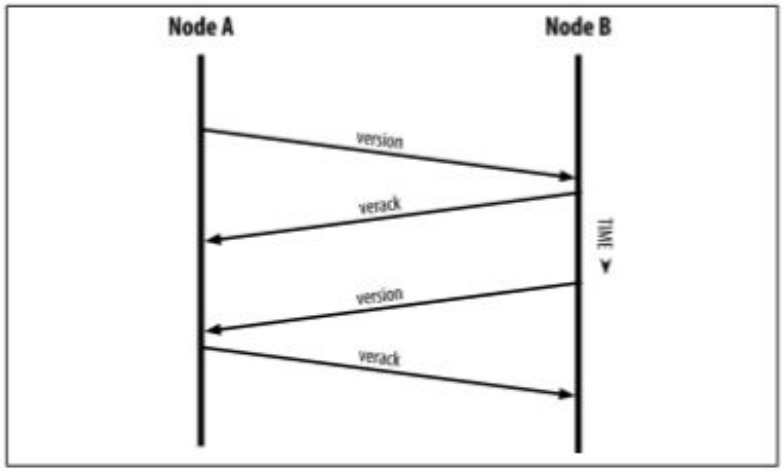
If time and resources permitted, the group would look into implementing a Full Node as a stretch goal.

# Features

## Connecting to the Bitcoin P2P Network

When first connecting to the P2P Network a Node is selected at random to connect with and a TCP connection is established, generally to port 8333. There are long running list of trusted Nodes known as Seed Nodes which can be used to quickly discover other nodes. The Nodes perform what is known as a “handshake” by transmitting various data back and forth to verify the Node. Information includes:

- *PROTOCOL_VERSION* - A constant that defines the Bitcoin P2P protocol version the client speaks
- *nLocalServices* - A list of local services supported by the node, currently just NODE_NETWORK
- *nTime* - The current time
- *addrYou* -The IP address of the remote node as seen from this node
- *addrMe* - The IP address of the local node, as discovered by the local node subver A sub-version showing the type of software running on this node (e.g., “/Satoshi: 0.9.2.1/”)
- *BestHeight* - The block height of this node’s blockchain




Lightweight Node Handshake, Photo Credit: *Mastering Bitcoin, Antonopoulos*

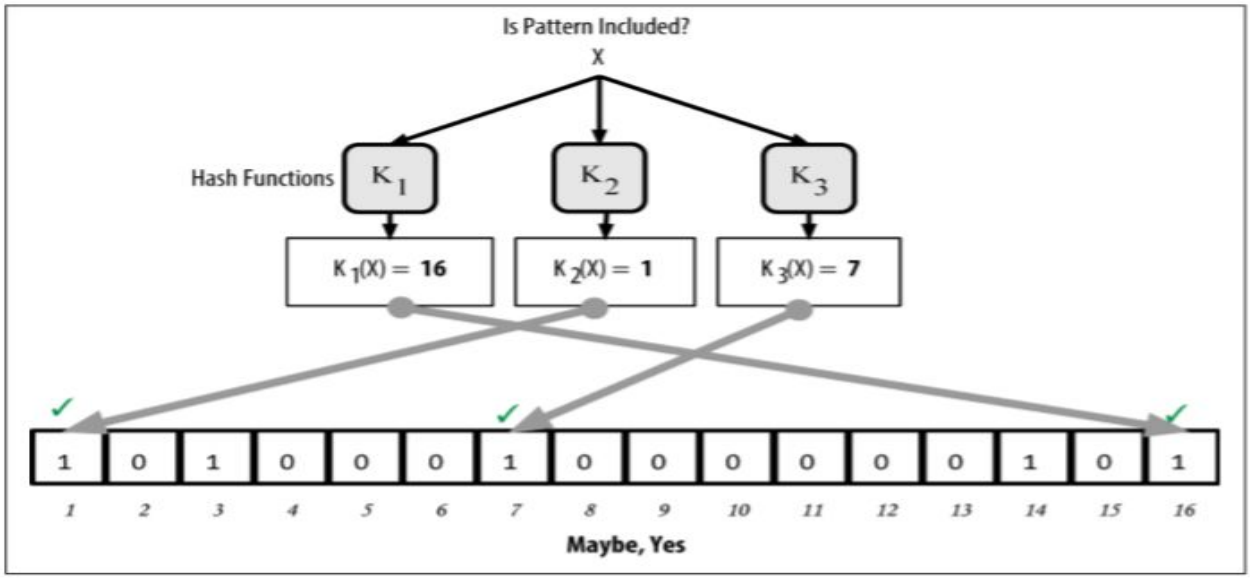
The new Node then uses the Node it is connected to, to find other Nodes in the Network and expand its neighboring Nodes.The Lightweight Node then sends a message asking for the block headers and receives up to 2000 headers at once. This continues until the Lightweight Node has a list of all block headers.

## Transaction Look-up

Normally on a Full Node a record of the blockchain is kept to look up transactions; however with the Lightweight Node this is not possible because it contains a list of block headers. The Lightweight Node has the ability to send information to a Full Node and verify that a transaction is in the blockchain.

This is performed using two advanced programming techniques; *Merkle Trees* and *Bloom Filters*. *Bloom Filters* are essentially a hash function performed M times. The bloom filter creates a bit array of size N binary digits. It contains a variable number of hashes, M. After the hash takes place the bit array is indexed and changes a bit from 0 to 1. After performing the hashes, the Node can check if the indexed Nodes are set to 1; if they are set to 1, then the transaction is apart of the *bloom filter*.

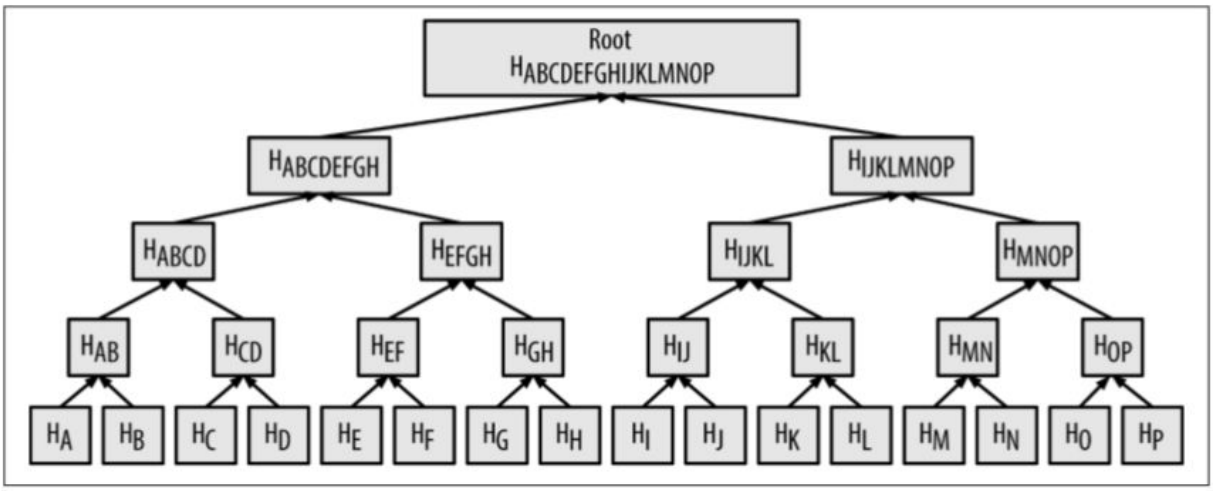
*Bloom filters* allow Nodes to send less information over the network and protect the privacy of users. *Bloom filters* with more hashes provide stronger privacy, but less information is transmitted over the network which could lead to the transaction now being found by the receiving node.




Bloom Filter Testing, Photo Credit: *Mastering Bitcoin, Antonopoulos*

*Merkle Trees* are trees where the parent nodes data is the hash of its left child concatenated with the hash of its right child, and then hashed. This continues all the way up the tree until only the root node remains. The hashing algorithm used is the Standard Hash Algorithm 256 (SHA256) applied twice.

$$H(x) = SHA256(SHA256(X))$$




Merkle Tree containing 16 transactions, Photo Credit: *Mastering Bitcoin, Antonopoulos*

# Limitations

The main limitation is the ability to obtain and use a dedicated computer provided by Rowan University or Computer Science department. The reason behind this is because of the size of which Bitcoin has grown too. Currently at time of writing, there are 451,597 blocks on the blockchain. Collecting this information would involve downloading 98.6GB of data to implement our stretch goal of a Full Node client.

The first step to solve this limitation would be to check viable options to obtain a computer and the second step to make sure Rowan University's IRT would permit our client connecting through the Rowan network and firewall. Our group will proceed the project with the intent of having a dedicated computer provided and no firewall limitations. If there are problems, the group intends to use the client running from our personal machines with a potential storage of block headers on a cloud-hosted database to keep up-to-date information.

## Dedicated Computer Minimum Requirements

| | |
|---|---|
| Operation System: | (Preferred Windows) |
| Hard Drive: | 125GB |
| Memory: | 4GB |
| Additional: | Network Connection, Remote Desktop enabled, Sleep mode disabled |

# GitHub

The GitHub repository can be found here:
https://github.com/FrankieF/FJSTSeniorProjectSpring2017