Blockchain cw

# Project Setup

In this the initial form of the application is set up. To do this a button and textbox are added to the from. The button is contains an event handler which so when it is clicked, it will change the rich textbox’s text to the text contained within the text box. This allows for a user to write whatever they want in there and see it appear in the textbox. As seen below you can see how when typing hello world in to the textbox will get it to appear in the rich textbox.

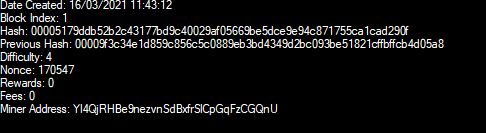
# Blocks and Blockchain

The next step to extend the application is to implement the blockchain class and the block class. This is done so when the application start it will create a new blockchain object called blockchain. This is blockchain object is a global variable within the form and will persist as long as the form. Within the a blockchain it will contain the a list of block objects. This represents all the blocks in a block chain and having them in a list allows for new blocks to be append to the list. Once a blockchain object it created, it will created the genesis block which is the first block in the block chain.

In creating blocks within the blockchain, there are multiple method which need to be set up. These are the: creation time, position index, previous hash, current hash, transactions, difficulty, nonce ,rewards and fees. The previous hash is used as a way of connection the blockchain and allowing or verification of the blockchain to occurs, as if it can be checked by seeing is each block’s previous hash is the same and the block before it. The difficulty value represents the amount of zeros needed in the hash to be accepted into the block chain. Transaction contains a list of all the transaction which the block contains will all of their values which is calculated and stored in fees and rewards with the minerAddress being stored to know who owns this block. Within current hash is where the hash of the block is contained. This is calculated by hashing the a lot of the data which is contained within the block itself as it uses the position index, creation time, pervious hash nonce, difficulty and rewards to create the hash. This will create a value wholly unique to the block itself allowing for it to be used to verify is it is the correct block.

For creating the Genesis block it uses a unique constructor within block which contains the basic information of the block but does not contain the hash of the pervious block as it is the initial block. Along with that it does not contain any allocation of transaction or being assigned to any user of the blockchain.

With the block chain being initialised buttons were added to the form which would allow the user to add new blocks to the blockchain. To see the blocks of the blockchain, a textbox and button were added were the user can print the number of the block which they desired and it would print the information of the block into the richTextBox part of the from. Below can be seen the basic information which is show on each block:



Along with this another button was implemented which would print all the information about the entire blockchain which is based of the same principles as printing the single block

# Transaction and digital signatures

The next necessary function added to the blockchain is the creation of wallets. To create a wallet, the wallet class is used to be able to create wallet objects. When a wallet object is created it produces both a private and public key for a wallet. The uses of asynchronous keys is necessary to be able to give some encryption and security to the blockchain. By having a public key it allows for it to be known to the blockchain and is used to assign blocks to specific wallets and therefore users. The private key is kept the owner of the wallet and is unknown to anyone else in the blockchain. It needs to be kept private as is works as the verification for a wallet to ensure that the user is in fact the owner which is done through both the private key and public keys. This is since only the private key create with the public key can be used together to produce a valid result.

To create a wallet in the form a new button is create which will create a wallet and will produce two keys. Each of these keys have their own respective textbox in the form which is where they will be after the creation of the wallet.

Transaction in a wallet is when a value of coin are sent from one wallet in the block chain to another. These transaction are stored within the block of the blockchain. To create a transaction, the recipients wallet public key and the sender public have to be filled in the from. For the recipient they have their own dedicated textbox and the sender is set from the public key text box. along with this to verify that the sender can create this transaction from their wallet, their private key is also required to create a transaction. To decide what is to be sent there two more textboxes created. Each of which contains the amount to be sent and the fee for the transaction respectively. Once all the information has been filled out the user can push the create transaction button which will create a new transaction object.

When a transaction object is create it passes all the information mention above. It then stores all the information about the public keys for both the sender and recipient along with the amount being sent in the transaction, fee of the transaction and the time of the transactions creation. With all of this information, it creates a hash from it. like with the block this is done to definitive identification and verification to the block. Once done it will also create the digital signature to sign the transaction which is created from the wallet of the sender. This means if the private or public key is wrong from the sender then it will not verified and therefore be considered invalid. In create signature, it takes the hash produced by the transaction and then signs the data it using both the sender’s private and public key.

With the transaction created it is then added to a transaction pool within the block chain object. This contains all of the transaction which have been created and not processed by a block. Every new transaction is put into this pool. Also, no transaction are valid until they have been processed by a new block in the blockchain.

# Consensus algorithm(Proof of Work)

With the transaction and wallets set up within the blockchain, it now means they have to be added to the block in the blockchain. The way a block is created is modified to accommodate this. When a new block is created, it will check the current transaction pool and will take as many transaction that are there up to it’s limit. The default for this block chain is 5 transaction per block. Once taken from the transaction pool, they are permanently saved within the block.

Proof of work is used with in a blockchain to ensure that the get block created does the required amount of work. This is decided by ensuring that the value the hash produces meets the conditions in the proof of work. To implement proof of work, difficulty is used to define the amount of work needed to be calculated to be considered a valid block in the blockchain. Difficulty in this implimnetation represents the amount of 0’s which need to be at the beginning of the hash to be considered valid for the blockchain. For example a difficulty of 3 would accept 00043… but reject 00625… into the blockchain. As hashes are created using attributes of the block, a changing variable needs to be introduced to allow the hash to change on repeated hashes. To solve this a NONCE is used, which is a number only used once. This value starts off at 0 and increases by increments of 1 for each time a hash is created. The process is called mining. In a traditional block chain, it gives the block to the user who can fulfil the proof of work requirements the fastest.

Once a user has the proof of work on a select block, that block in the blockchain become theirs. This means that all the transaction have to be processed by that user. The reward for mining the box are the fee and reward for the transaction within the block.

# Validation

To ensure the integrity of the blockchain, it needs to be able to verify if the structure is correct. This is done by checking every block to ensure that the previous block from it has the same has as the valued it stored from it on initialisation. If these values match it means the blockchain should correct. This is important as if the blockchain was change In any way there would be discrepancy in the pervious hash of the last block and the hash stored in said block. It also makes it hard to change any values if you change something a block, you also have to change the entire blockchain to represent that change. To check the validity of the current blockchain there is a button in the form which does what is described above. It will display the result of the verification check in the richTextbox.

To calculate the value in a wallet, the blockchain checks all the blocks and the transactions they hold. Form the sum of the transaction with that wallet, the balance is found. This is a secure way of taking account balance as a blockchain is immutable, so it is unlikely to have been tampered with. To access this in the form the user needs to put the desired account in the recipient textbox where if valid will display the current balance if the wallet.

# Extending proof of work

As proof of work calculation can become quite demanding, multi-threading can be used to improve this process. This is because with multi-threading multiple mining process can be running in parallel meaning that they should be able to obtain a hash of the correct difficult in less time. this takes advantage of the multi-core process which allows this kind of parallelism to occur.

To try and prevent the same work from being done be the different threads, multiple nonces. Which each nonce being separate for each thread it means that other threads won’t effect the value. Along with this the amount the nonce iterates is now the size of the amount of threads which means that each thread is mining the block at a different nonce value at all times. Once one of the threads has found a solution, it’s current nonce value is used for the block and all other threads which may still be running are then aborted. This prevents other threads from changing the values of the nonce and prevents them from continuing to process the block.

Below is a table of results obtained from testing the amount of time it takes for non-threaded mining and threaded mining:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Adjusting difficulty level for proof of work

Adjusting difficuluty levels can be used within blockchain as way to ensure that the amount of time taken to mine block stay closer to what the blockchain aims for. It does this through finding the block time which is the amount of time it took for the newest block to be created from the last block which was created. If the block time is shorted than the desire block time for the block chain, the difficult can be increased forcing the next block to take longer and visa versa for when the block time if longer. This allows for dynamic difficult to be implemented which helps to counteract from mining to go much faster than wanted and also preventing form difficult to become exponential hard for the proof of work needed. This helps prevent a monopoly from forming within the block chain.

# Implementing alternative mining preference settings

The goal of is to create multiple settings in how the blockchain chooses to send transaction to new blocks when created. the 4 method of preference settings implemented are:

* Greedy
* Altruistic
* Random
* Address Preference

All of these setting can be access in the form with a click of the respective button. This will change the setting inside the blockchain to that respective mining preference.

For greedy, the goal is to create a block with the transaction in the transaction pool which contain the highest fee meaning the greatest profit for the user mining them. This can be done by working through all the transactions to find the current highest and add it to the list of transaction being sent over to the new block. This would be done until all the transaction have been taken from the pool or when the transaction limit has been exceeded for the block.

For altruistic, this will add the transaction which have been in the transaction pool the longest with the goal of working through the transaction from oldest to newest. This is done by comparing the times of creation between each transaction and selecting the oldest ones to be sent to the new block.

For random, it will choose a transaction from the transaction pool at random. This is done through creating a random number between 0 and the amount of transactions in the transaction pool. The method is useful as it theoretically creates an unbiased approach to choosing transaction meaning that every transaction has an equal chance to be chosen to be processed.

For Address preference, it will only take the transactions from the pool where a specific address is included within the transaction. This allows for specific mining to be done which can benefit the targeted miner as it ensure their transaction are processed. This is implemented by sending the address of the user who has been selected and then scans through the transaction list to seen if any fit that criteria. Any which do fit it are then added to the block. Otherwise the block will contain no transaction and be worth no value.