Introduction to Blockchain Technology: Meme Economy Blockchain

Rahul Agrawal, Akshay Katyal, Mehmed Mustafa
Anant Sujatanagarjuna, Steffen Tunkel, Chris Warin

July 18, 2020

Abstract

Here could go our kick-ass abstract.

Contents

1 Introduction

2 Foundations

2.1 RESTful Applications

2.2 Flask Framework

2.3 Postman

3 Related Work

4 Approach

4.1 Block Design

Each block inside the chain has the following fields:

- index - the index of the block.
- minerID - the id of the miner. This identification is used when distributing mining rewards.
- previous_hash - the hash of the previous block inside the chain
- proof_of_work - a nonce value (magic number) which makes the hash of the block to match the difficulty pattern
- timestamp - an instance of time showing the creation time of the block
- transaction_counter - the number of transactions inside the block
- transactions - a list which holds all transactions inside the block

The genesis block has "0" value on all fields and an empty list for the transactions. The hash value of the genesis block does not match any hash patterns and when validating the chain the genesis block is skipped. It should be noted that even if the block is not skipped, since all values inside the genesis block are predefined, the hash value will always be the same value and could be hard coded.

All consecutive blocks, after the genesis block, have their proper field values. The minerID field is a special field because it's value is different for each node. Each node assigns its ID to the minerID field and tries to find a nonce value with this assigned specific minerID. Also each time a new transaction is added to the memory pool of transactions, the finding nonce value process is restarted, since the transactions list is updated.

4.2 Peer to Peer Network

4.3 Transactions

In the Block Design section it was discussed that each block has a list of transactions. There are 5 different kind of possible transactions:

- add_memeFormat - Used to add a new meme format to the network
- add_meme - Used to add a new meme to the network
- add_upvote - Used to upvote a meme inside the network
- sell_ownership - Used to sell the ownership of a meme format
- purchase_ownership - Used to purchase the ownership of a meme format

4.3.1 Raw-JSON body input formats of transactions

- add_memeFormat - {"imagePath" : "imagePathValue",
  "name" : "nameValue"}
- add_meme - {"imagePath" : "imagePathValue",
  "name" : "nameValue",
  "memeFormat" : "memeFormatID"}
- add_upvote - {"imageVoteId":"memeID",
  "upvoteID" : "upvoteID"}
- sell_ownership - {"ownershipSaleOfferID" : "ownershipSaleOfferID",
  "memeFormat" : "memeFormatID",
  "saleAmount" : "saleAmount"}
- purchase_ownership - {"ownershipPurchaseID" : "ownershipPurchaseID",
  "ownershipSaleOfferID" : "ownershipSaleOfferID"}

4.3.2 Transaction requirements and validation

This subsection discusses the requirements of each transaction. Once all of the requirements for a specific transaction are met, then the transaction is validated and ready to be mined in the next block.

- add_memeFormat - There are not any requirements for this transaction.
- add_meme - Each meme to be uploaded must belong to a specific meme format. This meme format must be already mined inside a block and appended to the chain.
- add_upvote - The meme to be up voted must be already mined inside a block and appended to the chain. The node which up votes must have enough credits available in its wallet.

- sell_ownership - The meme format to be sold must be already mined inside a block and appended to the chain. The node which creates this transaction must have the ownership of the meme format to be sold.
- purchase_ownership - The meme format to be purchased must be already offered for a sale, mined and appended to the chain. The purchasing node must have enough credits in order to buy the meme format. The worth of the meme format is specified inside the sell_ownership transaction.

4.4 Mining and Consensus

4.5 Wallet and Rewards

4.6 Visualization

5 Outlook

6 Conclusion

7 Template Section

This template should give us a first version we can start of with. This last section should support us in writing a more coherent paper together. Therefore I put some guidelines for the writing in section 7.1. Also there are some example for the use of functionalities in Section 7.2. You can copy them and adapt them the way you need them. Just leave this section here for now.

7.1 Guidelines

- **pushing to the repo**
  When you push the newest version to the repo, please leave out the files created by the compiler (besides the pdf). The report on the repo just needs the tex-file, the bib-file and maybe the most current pdf-file. Of course push the changes to the sub folders if you added images or sources.
- **references**
  For every section, subsection, figure, or table you include give it a label. Therefore everyone can refer to it later. It can be done by *\label{marker}*. The marker should declare the type of the object and a short (one word in the best case) name for the object. The types are "sec:" for a section, "fig:" for a figure and "tab:" for a table. You can refer to them then by *\ref{fig:example}*. Also you should use a $\sim$ symbol before instead of normal space to avoid line breaks there.
- **citations**
  For cititions the BibTex code for the source needs to be in the 'literature_list.bib'

(a) Meme Format (b) Meme

Figure 1: Example for 2 subfigures.

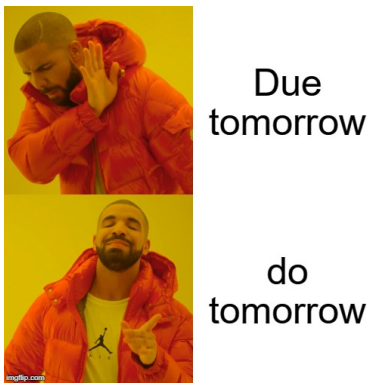

Figure 2: Example for a single figure.

file. You can usually get them pretty easily from *Google Scholar*. Please safe all papers/sources you used in the "sources" folder in addition. The citation can then be made by *\cite{antonopoulos2017mastering}* for example. Please use the ~ here too. The result then looks like this [**?**].

- **abbreviations**
  Please use the \ac{...} command to handle abbreviations. You can define them at the end of the document. Here is one example... When used the first time it automatically defines the abbreviation: *Artificial Intelligence* (AI). For all furhter times it just prints: AI. Also the plural is possible AIs

7.2 Examples

```
1       # Computes the hash of the Block
2       def compute_hash(self):
3           # self.__dict__ -> all variables inside the Block class
4           encoded_block = json.dumps(self.__dict__, sort_keys=True).encode()
5           return hashlib.sha256(encoded_block).hexdigest()
```

	SVM	**Neural Network**
MR-1	Permutation of training & test features	Permutation of input channels (RGB channels) for training & test data
MR-2	Permutation of order of training instances	Permutation of the convolution operation order for training & test data
MR-3	Shifting of training & test features by a constant (only for RBF kernel)	Normalizing the test data
MR-4	Linear scaling of the test features (only for linear kernel)	Scaling the test data by a constant

Table 1: Content totally out of context, literally just as an example for a table.

Abbreviations and Acronyms

AI *Artificial Intelligence*

ML *Machine Learning*

List of Tables

List of Figures