Ethereum: A Next-Generation Smart Contract and Decentralized Application platform by Buterin

1. What did the paper do well

The authors provide a good survery of blockchains and observe their limitations to justify the proposal of a new protocol, Ethereum, that increases the computational powers of the underlying blockchain. They did a good observation on different approaches for implementing alternative blockchains.

2. Where did the paper fall short

My observation is that Bitcoin's computational power is equivalent to that of pushdown automata, the scripts have access to a stack and the input is traversed only once. The PDA can recognize (and calculate) context-free languages, meaning it's powerless in performing any scripts that require more than O(n) operations. The paper doesn't explore these facts from a theory of computation standpoint, and instead focuses on other shortcomings of the system, they list lack of Turing completeness, value blindness, lack of state and blockchain blindness as the limitations for Bitcoin's scripting language.
In my opinion, the paper poorly formalized the EVM instance, to further clarify it for myself I had to look up different sources to understand it better. What it failed to mention was that accessing data on the blockchain is extremely costly, with read operations costing about 700 gas and writing operations costing above 1000 gas. This expense causes the users to be cautious about storing data on the blockchain, reducing the blockchain's size. Furthermore, to reduce the size of the blockchain the protocol stores the states in a data structure called Patricia Tree that allows efficiently inserting, deleting, and updating nodes in a Merkle Tree. Aside from the transaction lists and the most recent states, an Ethereum block also contains block number and difficulty stored on itself.

3. What I learned

From what I understand, the first approach of creating altcoin is implementing an independent network, either creating new protocols and software from scratch such as Namecoin or slightly modifying Bitcoin's software and running it with a new genesis block, such as Litecoin. The second approach is implementing custom behavior on top of the Bitcoin's blockchain, like colored coins or meta-coins. Colored coins are issued by publicly assigning a color to an existing coin, whereas meta coins have a different transition function aside from that of Bitcoin.
The authors introduce Ethereum, which aims to solve these problems by providing a Turing Complete system called the EVM, I argue that EVM is close to Von-Neumann architecture, where the system can dynamically change the underlying code, as opposed to a Turing Machine where the configuration of the system can't change, but the Turing Machine can emulate the execution of its input. This power comes from two opcodes, the CALLCODE or DELEGATECALL, which essentially updates the code that the system is running, an unsupported feature in Turing Machines. But in the end, there is no difference between a Turing Machine and a Von-Neumann, since they are computationally equivalent. EVM fetches the code pointed to by the PC (program counter), performs it if there are enough gas left, increment PC it, and repeat until either a stop instruction is reached or PC becomes larger than the code.
Ethereum is similar to Bitcoin's blockchain, but there are some key differences. First of all, an account in Ethereum could be either an external account or a contract account. The main difference is that contract accounts are essentially programs that will be executed upon receiving a message. An account has an address, a nonce to ensure transactions can only be processed once, a current Ether balance, a contract code (if it's a contract account), and finally a storage

that is empty by default. The transactions in this protocol consist of a recipient, signature to verify the transaction, amount of Ether to transfer, optional data field and STARTGAS value, and GASPRIC value, the two components to calculate their computation cost. A code could spawn multiple messages, each includes an implicit sender, a recipient, amount of ether, data fields, and a STARTGAS value that is calculated. These messages only exist in the execution and are not stored on the blockchain. While executing the code, the script has access to three types of space to store data, a stack, a memory of infinitely expandable byte arrays, and a long term storage, key-value that is persistence on the blockchain. The script can access data from the previously mentioned source, as well as the value, sender, and data of the message, and block header data.

The paper then considers different applications and their implementation on EVM, I was impressed by Decentralized Autonomous Organizations that give X percent of the members a majority right. The system must accept at least three types of inputs, a proposal for change, a vote on a proposal, and a vote count order, as well as any other organization-specific input that they might have. This system allows a group of entities to run an organization without explicitly trusting each other. Another interesting application was the saving wallets, which would solve the problem of losing keys in Bitcoin. Recall that if a key in Bitcoin is lost, all access to its funds is lost forever, one potential solution is to use multisigs, but they will give full ownership to the funds. Etherum allows the creation of saving wallets where the user can access up to 1% of the funds daily on her own, and the banker can access up to 1% of the funds daily on his own, but the user can limit their access. With both of their keys, they can access all of the funds. This way in the event of a key loss the banker can eventually access all of the funds, and if the banker starts abusing their access the user can cut off their access.

4. What Questions Do I Have

As mentioned above, I didn't fully understand what EVM does untill I looked it up. One thing that was also unclear was how the contract code is set.