Chris Dannen - Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners - 2017

Tom Rochette <tom.rochette@coreteks.org>

November 2, 2024 — 36c8eb68

0.1 Context

0.2 Learned in this study

0.3 Things to explore

- Having every transactions stored on every node of the network seems like a huge waste of space
- Is the cost of using the Ethereum network based on the number of instructions of your contract? In other words, how much CPU time you need to execute it
- How does the network decide which node will execute a smart contract?
  - How do you prevent duplicate execution?
  - Are all nodes supposed to execute the smart contract?
- The ability to have decade-old smart contracts still execute seems to rely on the programming language used to describe the smart contract... How can we guarantee said language will still exist in a few decades?
- Where is ether coming from?
- It seems wasteful to execute contracts which have too low transaction fee, but is it possible to determine a contract cost in order to avoid this execution?
- Given the Ethereum blockchain contains the contracts it can execute, it seems its biggest weakness/attack vector point would be to spam it with a ton of extremely large EVM program so that the blockchain becomes too big to store on a computer (requires too much storage space)
- What makes it possible to prevent many "clients" from all agreeing on some random PoW?
- Is it possible to develop a mechanism to make mining attractive even to solo miners?

1 Overview

- Ethereum could be considered as an OSI layer (encapsulating higher levels)

2 Notes

2.1 Chapter 1 - Bridging the Blockchain Knowledge Gap

2.1.1 What Ethereum Does

- In Ethereum, smart contracts are written in the programming language Solidity

2.1.2 Three Parts of a Blockchain

- A blockchain can be thought of as a database that is distributed, or duplicated, across many computers

- The innovation represented by the word blockchain is the specific ability of this network database to reconcile the order of transactions, even when a few nodes on the network receive transactions in various order
- What is widely called a blockchain is really the combination of three technologies
  - Peer-to-peer networking
  - Asymmetric cryptography: In Bitcoin and Ethereum, asymmetric cryptography is used to create a set of credentials for your account, to ensure that only you can transfer your tokens
  - Cryptographic hashing
- Adam Back released Hashcash in 2002, which pioneered the use of mining to send transactions
- Satoshi Nakamoto added distributed consensus to this innovation with the creation of Bitcoin in 2009

2.1.2.1 Ethereum Assumes Many Chains

- Ethereum was built with the assumption that copycats are a foregone conclusion, and that there may be many blockchains, and thus there should be a set of protocols in place by which they can communicate

2.1.3 Ether as a Currency and Commodity

2.1.3.1 Gresham's Law

- In an economy, "bad" money drives out "good." In other words, people save and hoard currencies they expect to appreciate in value, while spending currencies they expect to depreciate in value

2.1.3.2 The Path to Better Money

- Ether can be used to pay to run programs on Ethereum's network
- Ether can be considered a commodity

2.1.3.3 Cryptoeconomics and Security

- What makes systems like Ethereum and Bitcoin so secure is that they are not based on any hack-proof technology but rather rely on powerful financial incentives and disincentives to keep malefactors at bay

2.1.3.4 Back to the Good Old Days

- The Ethereum network functions as one large computer which executes programs in lockstep; it is a machine which is "virtualized" by a network of other machines

2.1.4 What Smart Contracts (Really) Do

- Smart contract: some business logic that runs on the network, semi-autonomously moving value and enforcing payment agreements between parties

2.1.5 Where's the Data

- How transactions are recorded in Ethereum: it's all stored on every node of the network

2.1.6 Deciding Where You Fit In

- The Ethereum network is a fully redundant distributed database, with copies on every node. That means you can trust your application to fire off a call when a certain condition is met, even if that condition happens decades into the future - and even if the nodes have all changed

2.1.7 What You Can Build Today

2.1.7.1 The Promise of Decentralized Databases

- Like all databases, a blockchain has a schema: rules define, constrain, and enforce relationships between entities
- In all databases, shared read/write access creates enormous complexity

2.2 Chapter 2 - The Mist Browser

- Two types of client applications: wallets and full nodes
  - Wallet: A lightweight node that connects to a blockchain to perform basic functions, such as sending and receiving cryptocurrency
  - Full node: Command-line interface that can perform the full gamut of operations allowed by the network

2.2.1 Wallets as a Computing Metaphor

2.2.1.1 Your Address is What?

- An account is a data object: an entry in the blockchain ledger, indexed by its address, containing data about the state of that account, such as its balance
- An address is a public key belonging to a particular user

2.2.2 Breaking with Banking History

- No individual has the power to create more ether

2.2.3 Working with Messages and Transactions

- Transactions are used to refer to state changes in the distributed database
- Messages are data objects passed back and forth across the network between smart contracts, and do not necessarily result in any changes being made on the chain

2.2.4 Transactions Change State

- A transaction in Ethereum refers to a piece of data bearing a cryptographic signature, which goes in the blockchain, and is thus recorded on every node in the network
- Every transaction triggers a message to accomplish this state change, but messages are also sent by EVM code

2.2.5 So, What Is a Blockchain?

- A block is a unit of time that encompasses a certain number of transactions

2.2.6 Paying for Transactions

- The EVM requires a tiny fee to process the transaction
- By forcing users to pay for transactions on the EVM, the likelihood of wasteful never-ending programs being executed is theoretically reduced
- These costs are priced in a unit called gas
  - You can think of gas as a metric indicating the number of steps the EVM will have to take to complete the instructions in the transaction
- Whether or not a transaction executes is determined by the amount of gas the sender is willing to pay. If the total number of steps exceeds the gas budgeted for a transaction, all steps are rolled back, and no part of the transaction is executed. If a user sends a transaction with too low a transaction fee, it will be processed only after some time, or not at all

2.3 Chapter 3 - The EVM

- The EVM is a single, global 256-bit "computer" in which all transactions are local on each node of the network, and executed in relative synchrony

2.3.1 What are Virtual Machines, Exactly?

- A virtual machine is an emulation of a computer system by another computer system

2.3.2 What the EVM Does

- The EVM can run arbitrary computer programs written in the Solidity language

2.3.3 Blocks The History of State Changes

- Transactions and state changes in the Ethereum network are segmented into blocks, and then hashed
- Each block is verified and validated before the next canonical block can be placed on "top" of it

2.3.3.1 Understanding Block Time

- In Bitcoin, a block is 10 minutes
- In Ethereum, block time is not a function of the issuance schedule of ether. Instead, block time is a variable that is kept as low as possible, for the sake of speedy transaction confirmation

2.3.4 Renting Time on the EVM

- For every instruction the EVM executes, there must be a cost associated, to ensure the system isn't jammed up by useless spam contracts

2.3.4.1 Why Is Gas So Important?

- Gas cost ensure that computation time on the network is appropriately priced
- Fees in the EVM are based on the amount of work being done, not on the size of the transaction

2.3.5 Working with Gas

2.3.5.1 How Gas Relates to Scaling the System

- There is no way to jam up the EVM without paying a lot, in the form of transactions fees, to do it
- Scaling is handled in a de facto way through the gas fee system
- Miners are free to choose the transactions that pay the highest fee rates, and can also choose the block gas limit collectively
- The gas limit determines how much computation can happen (and how much storage can be allocated) per block

2.3.6 Accounts, Transactions, and Messages

2.3.6.1 Externally Owned Accounts

- Contains a balance of ether
- Capable of sending transactions
- Controlled by the account's private keys
- Has no code associated with it
- A key/value database contained in each account, where keys and values are both 32-byte strings

2.3.6.2 Contract Accounts

- Have an ether balance
- Hold some contract code in memory
- Can be triggered by humans (sending a transaction) or other contracts sending a message
- When executed, can perform complex operations
- Have their own persistent state and can call other contracts
- Have no owner after being released to the EVM
- A key/value database contained in each account, where keys and avlues are both 32-byte strings

2.3.7 Transactions and Messages

2.3.7.1 Characteristics of Transactions

- A recipient address; specifying no recipient is the method for uploading new smart contracts
- A signature identifying the sender
- A value field showing the amount being sent
- An optional data field, for a message (if this is being sent to a contract address)
- A STARTGAS value
- A GASPRICE value

2.3.7.2 Characteristics of Messages

- A chunk of data sent by a contract to another contract (never to or from a human)
- Messages are virtual objects that are never serialized and exist only in the EVM
- A message contains the following
  - The sender address of the message
  - The recipient address of the message
  - The value field (indicating how much ether, if any, is being sent)
  - An optional data field (containing input data for the contract)
  - A STARTGAS value limiting the amount of gas the message can use

2.4 Chapter 4 - Solidity Programming

2.4.1 Primer

- In general, a computing environment is an infinite loop that repeatedly carries out whatever operation is current in the system's program counter
- The program counter iterates one by one until the end of that particular program is reached
- The machine exists the execution loop only if it encounters (throw) an error, or hits an instruction designating the machine to STOP or RETURN a result or value
- Three types of space in which to store data
  - The stack
  - Dynamic memory (heap)
  - Key/value store

2.5 Chapter 5 - Smart Contracts and Tokens

2.5.1 Assets Backed by Anything

- In financial parlance, an asset is a valuable resource that you expect will produce a benefit or value in the future

2.5.2 Money, Tokens, Reputation... So What?

- Longevity is an asset's killer feature: the longer an asset will grow in value, and the more impossible to counterfeit it, the more desirable it is

2.6 Chapter 6 - Mining Ether

2.6.1 Defining Mining

- The proof-of-work algorithm for the Ethereum protocol is Ethash
  - Also known as Ethereum's consensus algorithm or consensus engine
- The block that is selected as canonical is the one with the greatest amount of proof of work behind it
- Hashpower is the amount of computation a miner can apply to the network

2.6.2 Difficulty, Self-Regulation, and the Race for Profit

2.6.2.1 Factors Required for Block Validation

- Four pieces of data
  - Hash of the transaction ledger for this block
  - Root hash of the entire blockchain
  - Block number since the chain started
  - Difficulty of this block

2.6.3 How Proof of Work Helps Regulate Block Time

- In Ethereum, uncle blocks are required to bolster the winning block
  - As uncles lag more, it becomes harder for the network to find a true block, being that valid uncles are a requirement
- Ethash: The Ethereum protocol's defense against mining hardware optimization
  - A memory-hard algorithm that can't be brute-forced with custom application-specific integrated circuit (ASIC)
  - Key to this algorithm memory-hardness is its reliance on a directed acyclic graph (DAG) file, which is essentially a 1 GB dataset created anew every 125 hours, or 30k blocks. This period of 30k blocks is also known as an epoch

2.6.4 How Ethereum Uses Stale Blocks

2.6.4.1 Uncle Rules and Rewards

- In Ethereum's implementation of GHOST, uncles that are validated along with a block receive 7/8 of the static block reward, or 4.375 ether
- A maximum of two uncles are allowed per block
- These two places are won on a first-come, first served basis
- No transactions fees are collected or paid out for uncle blocks, because users are paying these costs once already in the valid block, which actually executes their commands
- In order to be worthy of a reward, an uncle block must have an ancestor in common with the true block within the last seven generations

2.6.5 The Difficulty Bomb

- One reason why cryptocurrencies have value in the marketplace is that they are limited in issuance

2.6.6 How Ethereum and Bitcoin Use Trees

2.6.6.1 Merkle-Patricia Trees

- In Bitcoin, the block header is an 80-byte chunk of data that includes the Merkle root as well as five other things
  - A hash of the previous block header
  - A timestamp
  - A mining difficulty value
  - A proof of work nonce

  - A root hash for the Merkle tree containing the transactions for that block
- From the perspective of the EVM, one limitation of the Merkle tree is that although it can prove or disprove the inclusion of transactions in the root hash, it can't prove or query the current state of the network, such as a given user's account holdings

2.6.6.2 Contents of an Ethereum Block Header

- Every block in Ethereum contains not just one Merkle tree, but three trees for three kinds of objects
  - Transaction tree
  - Receipts tree (data showing the outcome of each transaction)
  - State tree

2.6.7 Forking

- A fork occurs when two valid blocks point to the same parent, but some of the miners see one, and the rest see the other
- Anyone with more than 50 percent of the hashpower can engender a "hostile" deliberate fork

2.7 Chapter 7 - Cryptoeconomics Survey

2.7.1 How We Got Here

2.7.1.1 New Technologies Create New Economies

- The emerging field of cryptoeconomics is the study of economic activity conducted across network protocols in an adversarial environment

2.7.1.2 Rules of the Game

- Beware of centralization
- Most people are rational
- Large networks have people who churn in and out
- Censorship is not possible
- Nodes can talk freely
- Debt and negative reputation claims are unenforceable

2.7.1.3 Hashing

- Hashing is more secure than encryption, at least in the sense that there exists no private key that can "reverse" a hash back into its original, readable form
- If a machine doesn't need to know the content of a dataset, it should be given the hash of the dataset instead

2.8 Chapter 8 - Dapp Deployment

2.8.1 Web 3 Is Here (Almost)

- Web 3 is very much a vision that centers on the Ethereum protocol in particular. It is generally considered to have three components
  - Peer-to-peer identity and messaging system
  - Shared state (a blockchain)
  - Decentralized file storage

2.9 Chapter 9 - Creating Private Chains

2.9.1 Private and Permissioned Chains

- A private chain is just a cloud database achieved by way of the peer-to-peer Ethereum protocol: it's a silo that you control and that you can grant access to
- This should be contrasted with a permissioned blockchain, which like an enterprise software application has defined roles with permissions that can be set by a central administrator

2.9.2 Setting Up a Local Private Chain

- You need only three things to create a private chain
  - Custom genesis JSON file
  - Custom network ID (a number)
  - A directory where the network ID file is stored

3 See also

4 References

- https://dev.to/5chdn/the-ethereum-blockchain-size-will-not-exceed-1tb-anytime-soon-58a