Rupert Gobber,   207554

Project Course Final Report

Smart contract on Ethereum

1. Introduction on Ethereum

Ethereum is a decentralized software platform that enables Smart Contracts and Distributed Applications (ĐApps) to be built and run without any downtime, control or interference from a third party. The platform is also the basis for its own virtual currency, Ether.

At the heart of it there is the Ethereum Virtual Machine ("EVM"), which can execute code of arbitrary algorithmic complexity. In computer science terms, Ethereum is "Turing complete".

Developers can create applications that run on the EVM using friendly programming languages modelled on existing languages like JavaScript and Python.

Like any blockchain, Ethereum also includes a peer-to-peer network protocol. The Ethereum blockchain database is maintained and updated by many nodes connected to the network. Each and every node of the network runs the EVM and executes the same instructions.
The Ethereum network has two types of accounts, namely:

- External Accounts
- Contract Accounts

These accounts, both External and Contract are referred to as "state objects" and comprise the "state" of the ethereum network. Every state object has a well-defined state. For external accounts, the state comprises of the account balance while for contract accounts the state is defined by the memory storage and balance.

Architecture and architectural decisions
The ethereum network is a public blockchain network. It forms the basis of all decentralized peer-to-peer applications and organizations run on the network. Currently the main ethereum network uses Proof of Work as consensus protocol.
Ethereum's main components:

- P2P network:
  Ethereum runs on the Ethereum main network, which is addressable on TCP port 30303, and runs a protocol called ÐΞVp2p.
- State machine:
  Ethereum state transitions are processed by the Ethereum Virtual Machine (EVM), a stack-based virtual machine that executes bytecode (machine-language instructions). EVM programs, called "smart contracts," are written in high-level languages (e.g., Solidity) and compiled to bytecode for execution on the EVM.

- Data structures:
  Ethereum's state is stored locally on each node as a database(usually Google's LevelDB), which contains the transactions and system state in a serialized hashed data structure called a Merkle Patricia Tree.
- Consensus algorithm:
  Ethereum uses Bitcoin's consensus model, Nakamoto Consensus, which uses sequential single-signature blocks, weighted in importance by PoW to determine the longest chain and therefore the current state. However, there are plans to move to a PoS weighted voting system, codenamed Casper, in the near future.
- Transactions:
  Ethereum transactions are network messages that include (among other things) a sender, recipient, value, and data payload.

The ethereum network is comprised of two types of nodes namely, full nodes and light-weight nodes.

Full nodes contain the entire history of transactions since the genesis block. They are a full-fledged proof of the integrity of the blockchain network. Full nodes have to contain each and every transaction that has been verified according to the rules set up by Ethereum's specifications.

Light-weight nodes on the other hand only contain a subset of the entire blockchain. These types of nodes are mostly used in e-wallets which have to be light-weight in nature and hence the entire blockchain cannot be stored on them. These nodes, in contrast, do not verify every block or transaction and may not have a copy of the current blockchain state. They rely on full nodes to provide them with missing details (or simply lack particular functionality). The advantage of light nodes is that they can get up and running much more quickly, can run on more computationally/memory constrained devices, and don't eat up nearly as much storage.

In addition to the main ethereum network, also public testing network (Rospben, Rinkybit) or permissioned network (Quorum, Phanteon) are available. These types of network, mainly differ from the main network by consensus algorithm and network governance.

Underlying storage
The Ethereum Virtual Machine has three areas where it can store items.
- The first is "storage", where all the contract state variables reside. Every contract has its own storage and it is persistent between function calls and quite expensive to use.
- The second is "memory", this is used to hold temporary values. It is erased between (external) function calls and is cheaper to use.
- The third one is the stack, which is used to hold small local variables. It is almost free to use, but can only hold a limited amount of values.

For almost all types is not possible to specify where they should be stored, because they are copied every time they are used. The types where the so-called storage location is important are structs and arrays. If you e.g. pass such variables in function calls, their data is not copied if it can stay in memory or stay in storage. This means that you can modify their content in the called function and these modifications will still be visible in the caller.

There are defaults for the storage location depending on which type of variable it concerns:
- state variables are always in storage
- function arguments are in memory by default
- local variables of struct, array or mapping type reference storage by default

- local variables of value type (i.e. neither array, nor struct nor mapping) are stored in the stack

Implementation languages
Smart contracts are high-level programming abstractions that are compiled down to EVM bytecode and deployed to the Ethereum blockchain for execution.
They can be written in:Solidity (a language library with similarities to C and JavaScript), the most supported and used.

- Serpent (similar to Python),
- LLL (a low-level Lisp-like language),
- Mutan (Go-based, but deprecated).

There is also an experimental secure smart contract language under development called Vyper (a strongly-typed Python-derived decidable language).

2. Scenario and tools

Scenario description
In this course project, the voting scenario has been implemented. In particular, there are some candidates and a voters can vote them once a time. Blockchain is used to prevent double-voting and the smart contract is used to prevent this situation.

Testing environment
Ganache - Is a local virtual blockchain which sets up 10 default Etheruem addresses, complete with private keys and all, and pre-loads them with 100 simulated Ether each. There is no "mining" per-se with Ganache - instead, it immediately confirms any transaction coming its way. This makes iterative development possible - you can write unit tests for your code which execute on this simulated blockchain, deploy smart contracts, play around, call functions, and then tear it all down for further simulation or new tests, returning all addresses to their initial state of 100 Ether.

Supporting tools
Ethereum provides useful tools for development and testing. The main ones are:

- Command Line Development Management Tools, for creating a basic structure of an DAPP project:
  - Truffle
  - Embark
  - Dapple
- Testnode with RPC Interace, for deploying contracts on a virtual node and make transactions without the need to be mined:
  - Javascript testrpc
  - Python testnode
- Browser based IDE, to get instant feedback for solidity code:
- ReMix
- EthFiddle
- Superblocks Lab

Deployment process
Truffle is a development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM).

Deployment process step by step:

- Install Truffle. *$ npm install -g truffle*
- Create a bare Truffle project with no smart contracts included, use *$ truffle init.*
- Once this operation is completed, you'll now have a project structure with the following items:
    - contracts/: Directory for Solidity contracts
    - migrations/: Directory for scriptable deployment files
    - test/: Directory for test files for testing your application and contracts
    - truffle-config.js: Truffle configuration file
- Write a Smart Contract with .sol extension in *contracts/* directory
- Create a deployment script in the *migrations/* directory. Migrations are JavaScript files that help to deploy contracts to the Ethereum network.
- Configure the file truffle-config.js with a custom network i.e. "development".
- A local (Ganache), test (Rospben, Rynkybit) or the main Ethereum Network can be used.
- Run the deployment on the network: *$ truffle migrate --network "development"*

MetaMask configuration
MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today. It allows you to run Ethereum dApps right in your browser without running a full Ethereum node. In this scenario, let's create a new network within Metamask at the localhost: 7545 port. In this way, Metamask will search for the node with private key within Ganache blockchain and connect to it.

3. Contract testing

Regarding contract stress testing, the speed of contract execution depends mainly on factors such as network congestion, gas price and the size of the transaction. Moreover, speed cannot be compared between measurements, it can vary significantly based on network status.

Furthermore, several test networks are not the same as in the main network. Even if we managed to complete a sort of stress test in Testnet, the results would probably be very different in the Mainnet. The parameters and the use of the networks are different.

The major delays in executing smart contracts are not caused by the quality of the code but by the network itself. Code optimization makes no difference to speed, but obviously it could make a big difference to the cost of execution (gas). The higher the cost, the greater the incentive to validate the transaction for other peers. In the end, assuming the contract does what it needs to do and are safe, gas is the only measure that turns out to be interesting.

Deploying contract on Testnet
Testnet is a place to test your smart contracts solutions. Basically, it's a clone of the Ethereum network that allows you to deploy and test your smart contracts without paying real fees.

There are many testnets currently in use, and each behaves similarly to the production blockchain (where your real Ether and tokens reside). Developers may have a personal preference or favorite testnet, and projects typically develop on only one of them.

- Ganache: formerly known as the TestRPC, Ganache CLI is a fast and flexible blockchain emulator for Ethereum. It is both a desktop application and a command line tool, and is available on different major operating systems, such as Mac, Windows, and Linux. If you are

interested in using this simulator, you can install it via NPM, a package manager for Node.js modules. Ganache CLI has some notable characteristics:
– Users can instantly mine their transactions.
– Transactions are free.
– It does not need a faucet or mining because users can recycle or reset their accounts instantly with a fixed amount of Ether.
– Users can modify the gas price and mining speed.
– It has a GUI where users can monitor their test chain events.
– It also allows users to specify which hardfork should be used. Supported hardforks are byzantium, constantinople, and petersburg.

- Ropsten: A proof-of-work blockchain that most closely resembles Ethereum; you can easily mine faux-Ether
- Kovan: A proof-of-authority blockchain, started by the Parity team. Ether can't be mined; it has to be requested
- Rinkeby: A proof-of-authority blockchain, started by the Geth team. Ether can't be mined; it has to be requested

To test smart contracts, Ethereum Ropstens is the best network, but Ropsten Eths are needed: they can be extracted "undermining" with our computer. Or you can ask someone for the computer, or ask someone else for money. For this reason this network was not chosen for testing.

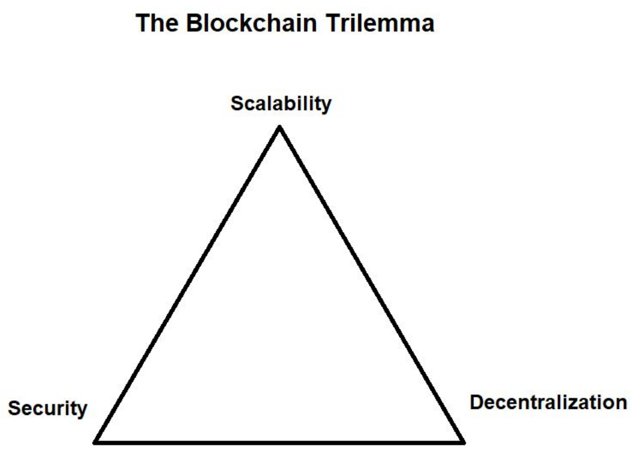
In Rinkeby test network, user can't mine Ether but only receive it by means of faucets. It is considered more stable, and the formation of a block takes 15 seconds.

4. Analysis of blockchain implementations

There are 3 main attributes of any blockchains:

- Security: a distributed network is resistant to a wide variety of attacks or hacks such as 51% attacks, Distributed Denial-of-Service (DOS) attacks and Sybil attacks.
- Decentralization: open source decentralization allows for a censorship-resistant, inclusive network that enables anyone to participate without prejudice.
- Scalability: capacity of blockchains in processing transactions in the network. A scalable system is able to cater to more transaction and activity in the network without suffering from network stress.

These 3 attributes describe the Blockchain Trilemma



Any blockchain implementation can only choose 2 attributes sacrificing the third. For instance, Bitcoin and Ethereum were designed to focus on decentralization and network security. As a result, the attribute of scalability wasn't a core feature since both blockchains have slow processing speed. This is because full decentralization and high levels of security require distributed consensus on the state of the blockchain, and this process can take significant amounts of time. The security is guaranteed by the distributed consensus in the blockchain. This is allowed using consensus algorithms.

Since the decentralized ledger constituting a blockchain continues to increase with each block added to the chain, scalability is an inherent problem of blockchain technology. If a cryptocurrency wants to become mainstream, it is necessary to process hundreds of thousands of transactions per second to ensure that the economy can move without major delays.
To avoid the previous problems, many solutions have been developed.

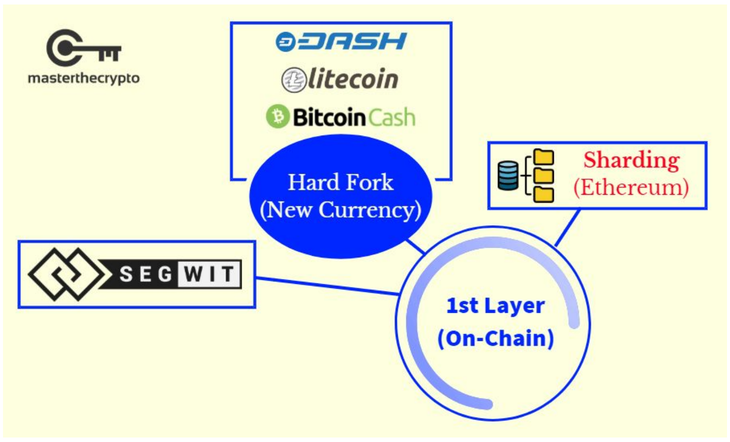
Blockchain implementation can be divided into 2 main categories:

- First Layer solution (on-chain)
- Second Layer solution (off-chain)

1st Layer on-chain

This is closely linked to the expansion of the network. There are two ways to think about level 1 scalability:

- Horizontal scalability: adding nodes to a network to handle growing tasks. It is a bad idea since every transaction has to be processed by every single node then increasing the load.



- Vertical scalability: adding more power to the machines that are in the network to handle the increasing activities. It is important to note that with the increasing of the block size, also increase the work to validate it. In this way, only "powerful" nodes can undermine causing less decentralization (less participant nodes) so less security.

Consequences of vertical / horizontal scalability:

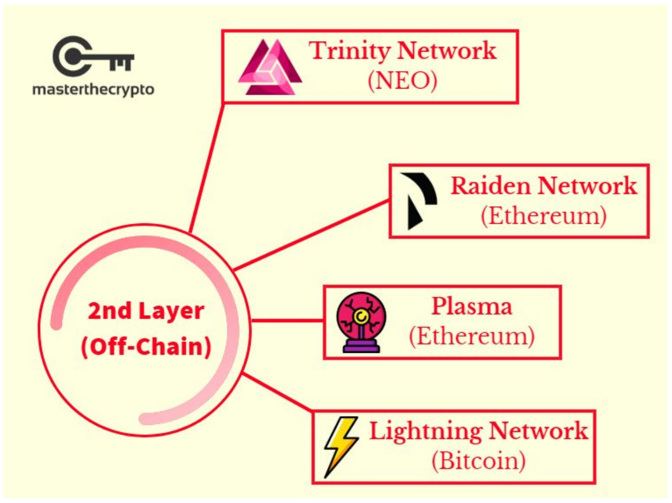
- Bitcoin is secure and decentralized, but not fast.
- Ethereum is secure and decentralized but not fast.
- Ripple XRP currency is fast but less secure and has limited decentralization.

Some projects have chosen to sacrifice security or decentralization in favor of the high volume of transactions:

- Bitcoin Cash (BCH) is a bitcoin fork, in which the protocol was changed bringing the block size from 8 to 32 MB. The consequence was that mining has become a more expensive operation and more complete operational nodes have been required. This implies less decentralization of the network in favor of a higher transaction throughput.
- SegWit (Segregated Witness) is a soft-fork of Bitcoin. Similar to BCH but with 4 MB blocks. For now it is a good compromise, but in the future it will have the same problems of the current Bitcoin network.
- EOS project has renounced decentralization. The protocol has been changed, using a DPoS with 21 validators nodes. The consequence is that it is less secure, but quick to verify transactions.

2nd Layer off-chain
Layer 1 solutions are not valid as they congest the network and need more powerful nodes to validate the blocks. Scalability problems persist. For this reason the Layer 2 solutions have been designed with these characteristics:

- They leave the basic level (for example Bitcoin and Ethereum network) and create the protocols on it.
- Little less safe but provide higher throughput. The second level of mechanisms can extend the usefulness of public blockchains, allowing interactions to occur in chains that still refer to the reliable base level when necessary.



- Applications that require high throughput can use Level 2 environments while still benefiting from a secure base layer.

Several “2nd Layer” solutions are already tested and working. The most important are the following.

- Lightning Network
  Protocol P2P over SegWit (Bitcoin) that allow to process transactions in a separate channel and add blocks later. The peers open an “off-chain” payment channel between them. When the payment channel is open, any number of transactions can directly occur without ever touching the main blockchain. In this way, Bitcoin can be transferred as quickly as the user wallets can communicate over the Internet. At the end, they conduct a “closing transaction” on the main blockchain, that save all of their previous transactions.

- Plasma Cash
  Protocol that allow users to focus only on the blocks containing the coins they take into account. This help data optimization. The workflow is similar to the Bitcoin Lightning Network. Plasma allows a series of contracts that run on top of a root chain (for the moment it is integrated only on Ethereum and OmiseGo) and consists of a network of “child chains”, called sidechains, connected to a root chain. In this way, the root chain process only a small number of commitments from the child chains. Each child chain functions has its own blockchain with its own consensus. The process of utilizing Plasma conceptually works as follows:
  - Smart contracts are created on the root chain and act as the child chain’s anchor to the root chain.
  - A child chain is created as a new blockchain with its own consensus (typically something like PoS).
  - All states within the child chain are enforced with fraud proofs that ensure all state transitions are valid.
  - Smart contracts specific to that dapp or child chain can then be deployed to the child chain.
  - Necessary assets can be transferred from the root chain to the child chain.

    Users of a dapp running on a child chain do not ever actually have to interact with the root chain. Further, the Ethereum blockchain can handle more extensive and more concurrent datasets. The burden that is removed on the root chain also translates to the Ethereum nodes, who do are rewarded with lower processing and storage requirements.

- Casper
  Attempt to implement the PoS algorithm on Ethereum. There are two main project by Ethereum team:
  - Casper FFG: hybrid PoW / PoS, every 50 block in PoS there is a PoW validation “on-chain”
  - Casper CBC: his focus is to create a comprehensive proof of stake consensus mechanism with sharding.

  The principals PoS advantage are:
  - Only designated nodes validate the transaction, and not the entire network;
  - Since there’s no mining, expensive special hardware is not needed and the energy requirements are lower;
  - It’s easy to identify validators with high loyalty, simply pick the ones who have staked a higher number of crypto tokens and a longer duration.

- Raiden Network
  It is Ethereum’s version of Bitcoin Lightning Network. It is an off-chain scaling solution that transfers in bidirectional payment channels.
  The interaction with Raiden only requires developers to interact with an API to build scalable applications on top of it. Raiden bypasses the need for a global consensus by leveraging hash-locked transfers called balance proofs. Balance proofs are collateralized by on-chain deposits that are made before setting up bidirectional payment channels. Bidirectional payment channels allow for nearly unlimited token transfers between two participants as long as their net sum does not exceed the initial deposit amount.

- Loom Network
  It is a software development kit that teaches and guides developers through the process of creating their own blockchains over Ethereum. It is a “build your own blockchain” generator

5. References

References of development environment and smart contract implementation

- Ethereum
  https://www.ethereum.org
- Solidity
  https://solidity.readthedocs.io/en/v0.5.3/
- Remix
  https://remix.ethereum.org
- Truffle
  https://truffleframework.com
- Ganache
  https://truffleframework.com/docs/ganache/overview
- MetaMask
  https://www.metamask.io

References of blockchain analysis section

- Ethereum test networks comparison
  https://medium.com/coinmonks/ethereum-test-networks-69a5463789be
- Comparison cryptocurrency speed (3 May 2018 article)
  https://www.fool.com/investing/2018/05/23/ranking-the-average-transaction-speeds-of-the-15-l.aspx
- Comparison cryptocurrency speed (5 June 2018 article)
  https://medium.com/coinmonks/understanding-cryptocurrency-transaction-speeds-f9731fd93cb3
- Comparative Analysis of Blockchain Consensus Algorithms
  http://docs.mipro-proceedings.com/sp/sp_09_4999.pdf
- Lightning Network (Bitcoin)
  https://lightning.network/
  https://99bitcoins.com/bitcoin-lightning-network/
- Plasma & Raiden (Ethereum)
  https://blockonomi.com/plasma-raiden-ethereum-scaling/
- Casper (Ethereum)
  https://github.com/ethereum/research/blob/master/papers/casper-basics/casper_basics.pdf
  https://medium.com/@jonchoi/ethereum-casper-101-7a851a4f1eb0
- Loom Network (Ethereum)
  https://www.investinblockchain.com/loom-network/
- Blockchain scalability project
  https://www.investinblockchain.com/solving-blockchain-scalability-problem/