Unit 5

CHAPTER 5

Blockchain Ethereum Platform using Solidity

University Prescribed Syllabus

What is Ethereum, Types of Ethereum Networks, EVM (Ethereum Virtual Machine), Introduction to smart contracts, Purpose and types of Smart Contracts, Implementing and deploying smart contracts using Solidity, Swarm (Decentralized Storage Platform), Whisper (Decentralized Messaging Platform).

5.1 WHAT IS ETHEREUM?

GQ. What is Ethereum. What the major features of it? **(4 Marks)**

Ethereum is a blockchain-based computing platform that gives programmers the ability to create and deploy decentralised apps, which are those that are not controlled by a single entity. You can design a decentralised application where the decision-making authority resides with the system's users.

Features of Ethereum

(1) **Ether :** This is Ethereum's cryptocurrency.

(2) **Smart contracts :** Ethereum supports the creation and implementation of such a contract.

(3) **Ethereum Virtual Machine :** Ethereum offers the underlying technology the software and architecture that recognises smart contracts and enables you to communicate with them.

(4) **Decentralized applications (Dapps) :** A Dapp (also spelt DAPP, App, or DApp) is a short form for a decentralised application. Decentralized apps, which are consolidated applications, are possible using Ethereum.

(5) **Decentralized autonomous organisations (DAOs) :** You may build these for democratic decision-making using Ethereum.

5.1.1 Ether

- Ether (ETH) is Ethereum's cryptocurrency. It serves as the network's fuel. It is used to cover the transaction fees and computational costs associated with every transaction carried out on the Ethereum network. Ether is a peer-to-peer currency, similar to Bitcoins.
- Ether may be used to purchase gas, which is required to process every transaction completed on the Ethereum network, in addition to paying for transactions.
- Additionally, gas must be purchased using ether if you wish to deploy a contract on Ethereum. Therefore, the execution cost a user pays to perform a transaction in Ethereum is called gas.
- Decentralized apps, smart contracts, and routine peer-to-peer payments may all be implemented with ether.

5.1.2 Smart Contracts

GQ. Explain the concept of smart contracts. How it differs from traditional systems? **(4 Marks)**

- The concept of smart contracts was first proposed by Nick Szabo in 1994. Szabo is a legal scholar and cryptographer known for laying the groundwork for digital currency.
- A smart contract is a simple computer programme that makes it easier for two parties to exchange any asset. You may wish to exchange money, stocks, real estate, or any other kind of digital asset. These contracts can be created by any user on the Ethereum network.
- The terms and conditions that were mutually agreed upon by the parties make up the majority of the contract (peers).

- A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible.
- The main advantage of smart contracts is that they cannot be changed once they have been performed, and every transaction carried out on top of one is forever recorded—it is immutable.
- Therefore, the transactions associated with the original contract will not change if the smart contract is modified in the future; you cannot edit those transactions.
- Any smart contract execution on Ethereum is decentralised since the smart contract verification is carried out by anonymous network participants without the requirement for a centralised authority.
- The identities of the two parties are secure on the Ethereum network, and any asset or currency may be transferred in a trustworthy and transparent manner.
- When a transaction is completed correctly, the sender's and the receiver's accounts are updated appropriately, which builds confidence between the parties.

5.1.3 Ethereum Virtual Machine

GQ. Write a short note on EVM. **(4 Marks)**
GQ. How smart contracts are compiled on EVM? **(4 Marks)**
GQ. Elaborate the interaction of Smart contract and EVM with DApps. **(4 Marks)**
GQ. Describe cycle of smart contract execution on EVM. **(4 Marks)**
GQ. What is Gas in Ethereum? **(2 Marks)**
GQ. What are consensus mechanism used in Ethereum? **(4 Marks)**
GQ. Differentiate the mining process of Ethereum from Bitcoin. **(4 Marks)**

- EVM is intended to function as a runtime environment for compiling and deploying Ethereum-based smart contracts. The smart contract language for Ethereum i.e Solidity programming language, is understood by EVM.

- In essence, you may install your stand-alone environment, which can serve as a testing and development environment, as EVM is run in a sandbox environment.
- In other words the EVM doesn't interact with the OS so it has no access to disk, RAM or networking it's essentially self-contained.
- Once you are satisfied with the smart contract's performance and usefulness, you may deploy it on the Ethereum main network after testing it (using it) "n" times and confirming it.EVM is quasi-Turing complete machine.
- Theoretically, Turing machine can run any computation irrespective of complexity, recursion, depth, length and complication no matter how much time and storage is required to complete it. Executing instructions on EVM incurs some cost known as gas.
- Ethereum platform has built in token known as ether.
- Gas in this context means some fraction of ether which needs to be supplied along with other inputs while executing the transaction or smart contract.
- This is similar to the fuel(gas/petrol/diesel) that is required to run a motor vehicle.
- A vehicle runs as long as fuel is available. In the same manner code execution on Ethereum continues till the availability of the gas.
- Hence, EVM is known as quasi-Turing complete machine due to dependency on gas availability for execution.
- Also, gas offers protection against infinite loop scenario in smart contract programming.
- The significance of EVM is also evident in the effectiveness of preventing Denial-of-Service or DOS attacks.
- In addition, EVM is also responsible for ensuring that a specific program does not have access to the states of each other.

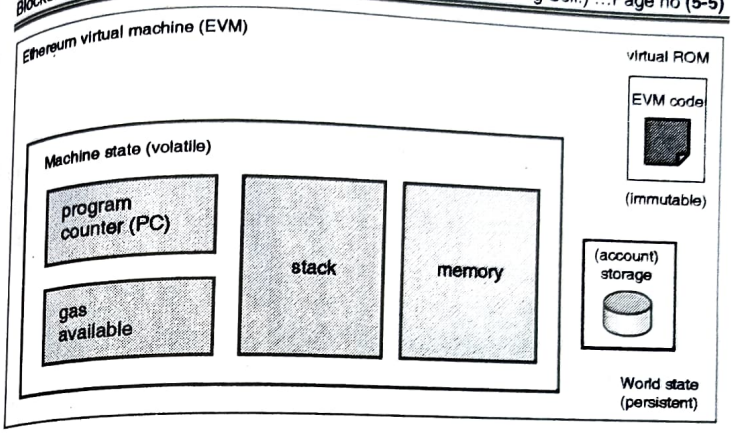


Fig. 5.1.1 : Ethereum Virtual machine

Interaction of Smart Contract and EVM with DApp

- When a external application such as Dapp wants to interact with the smart contract it needs some details about contract such as the function signatures and properties exposed by it.
- This is similar to how a distributed application interacts with an external web service using service description language.
- A smart contract program, written say solidity gets compiled to a bytecode.

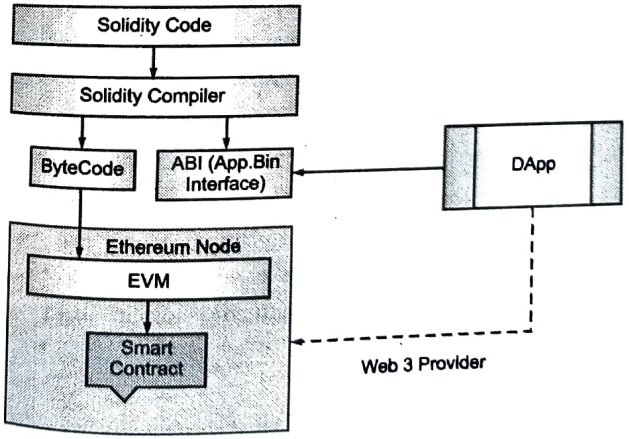


Fig. 5.1.2 : Smart contract and EVM

- The Ethereum language compiler generates a human readable interface know as Application Binary Interface (ABI).
- It contain all the necessary details needed by a Dapp to interact with smart contract using an external helper library such as Web3. Any programming language used in the smart contract is translated into bytecode that the EVM can understand.
- The EVM has the ability to read and run this bytecode. As soon as your smart contract is created in Solidity, it is translated into bytecode and deployed on the EVM, providing security against hacker attacks.

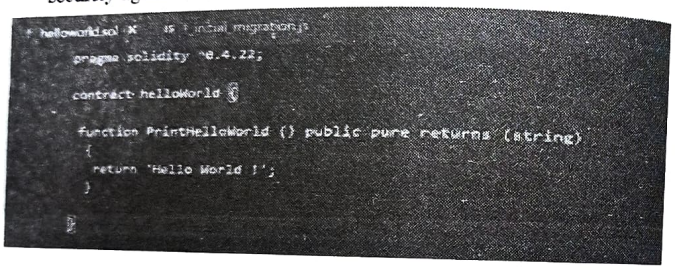


Fig. 5.1.3 : Sample Smart contract in solidity

- When a external application such as Dapp wants to interact with the smart contract it needs some details about contract such as the function signatures and properties exposed by it.
- This is similar to how a distributed application interacts with an external web service using service description language.
- The Ethereum language compiler generates a human readable interface know as Application Binary Interface (ABI). It contain all the necessary details needed by a Dapp to interact with smart contract using an external helper library such as Web3.

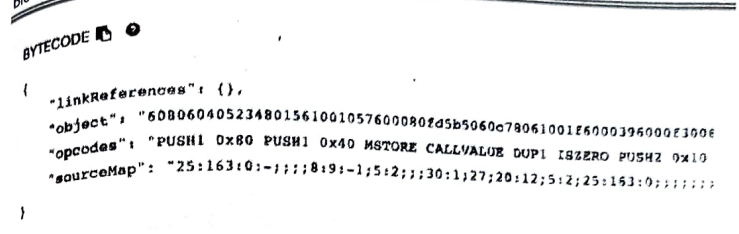


Fig. 5.1.4 : Conversion of smart contract to bytecode

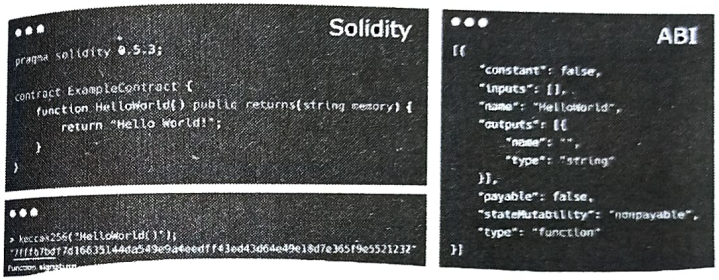


Fig. 5.1.5 : Conversion of smart contract ABI

☞ How Does EVM Work?

- Consider the case when person A wishes to give person B 10 ethers. A smart contract will be used to transfer funds from A to B and send the transaction to the EVM. The Ethereum network will use the proof-of-work consensus algorithm to validate the transaction.
- The Ethereum miner nodes will validate this transaction, determining if A is who he claims to be and whether he has the required amount to transmit. The miners will charge a fee to validate this transaction and will get a reward throughout this process. Once the transaction is validated, the ether will be deducted from A's wallet and deposited to B's wallet.
- Using their respective EVMs, each node on the Ethereum network runs smart contracts.

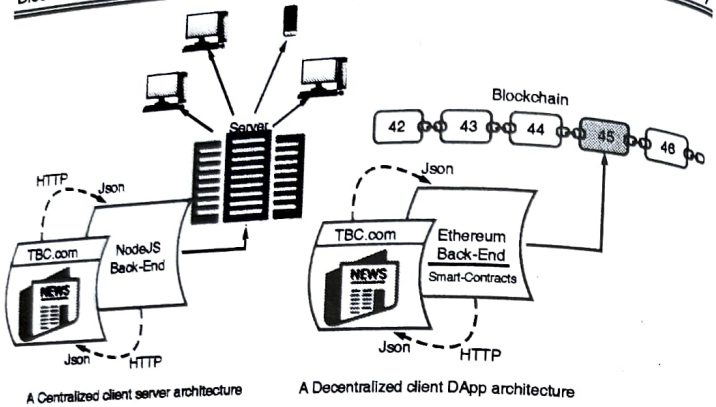


Fig. 5.1.6 : Total cycle of smart contract execution over Ethereum blockchain

☞ Proof of Work

Every node in the Ethereum network has :

- The full chain the complete history of all transactions
- The transactions linked to the smart contract as well as the history of the smart contract, which includes the address where the smart contract is deployed.
- The handle to the smart contract's current state.
- Validating the blocks is the aim of the miners on the Ethereum network. Miners employ their computing power and resources to generate the correct hash value by altering the nonce for each block of a transaction.
- The nonce will be changed by the miners and run through the Ethash algorithm, which is the hashing technique used for Ethereum. This produces a hash value that should be less than the predefined target as per the proof-of-work consensus.
- If the hash value generated is less than the target value, then the block is considered to be verified, and the miner gets rewarded. When the proof of work is completed, the result is broadcast and shared with every other node so that they may update their ledger.

- The block is uploaded to the Ethereum main blockchain and the miner is rewarded with a reward, which as of right now is three ethers, if other nodes acknowledge the hashed block as valid. Additionally, the transaction fees that were produced for validating the block are given to the miner.
- The cumulative transaction fees associated with all the transactions that are included into the block are also paid to the miner as a reward.

☞ Proof of Stake

- Proof of stake is another procedure being developed for Ethereum. It serves as an alternative for proof of work and is intended to reduce the use of costly resources used for mining with proof of work.
- In proof of stake, the validator the miner can validate transactions based on quantity of cryptocoins he or she currently possesses before initiating the mining process.
- Therefore, the miner has a better chance of mining the block based on the amount of crypto currency they have collected thus far. In contrast to proof of work, proof of stake is now less common.

☞ Gas

- Applications running on the Ethereum network require gas, much like a car needs fuel to run. A user must pay a fee to complete any transaction on the Ethereum network.
- In this situation, the fee is made up of ethers, and the intermediary monetary value is known as gas. Gas is a unit used on the Ethereum network to represent the amount of processing power needed to execute a smart contract or a transaction. Therefore, you would have to pay gas, which is measured in ethers, if you needed to complete a transaction that updated the network.
- The transaction fees in Ethereum are computed using a formula. The transaction fees equal the amount of gas required to execute a transaction multiplied by the gas price. "Gas limit" refers to the amount of gas used for the computation and the amount of ether a user is required to pay for the gas.

- The transaction fees in Ethereum are computed using a formula.
- The transaction fees equal the amount of gas required to execute a transaction multiplied by the gas price. "Gas limit" refers to the amount of gas used for the computation and the amount of ether a user is required to pay for the gas.

How is Ethereum Mining Different from Bitcoin Mining ?

- By market capitalization, it is the second-largest cryptocurrency after Bitcoin. But unlike Bitcoin, it wasn't designed to be a kind of digital money.
- The goal of Ethereum's creators was to create a new type of global, decentralised computing platform that would take the openness and security of blockchains and apply them to a wide range of applications.
- Key distinctions between bitcoin and ethereum
- The most popular cryptocurrency is Bitcoin, while Ethereum comes in second.
- The cryptocurrency itself and the blockchain network are both referred to as Bitcoin. The terms "Ethereum" and "Ether" refer to the network and respective cryptocurrency, respectively.

Some other differences between the Bitcoin and Ethereum networks are the following :

(1) **Abbreviation** : *BTC* refers to Bitcoin currency, and *ETH* refers to Ether.

(2) **Inception :** Bitcoin is the world's first cryptocurrency, created in 2009. Ethereum came next in 2015. Ether was originally intended to complement Bitcoin, but the two coins ended up competing.

(3) **Trades enabled :** Bitcoin only transacts in digital currencies, but Ethereum provides a variety of trading options, including smart contracts.

(4) **Block your time :** Compared to 10 minutes for Bitcoin, the average block time for the confirmation of an Ether transaction is roughly 12 seconds.

(5) **Consensus System** : Both the Ethereum and Bitcoin networks have relied on proof-of-work (PoW) mechanisms to

verify transactions. As part of the Ethereum 2.0 upgrade, proof of stake (PoS) will replace the current Ethereum system in 2022.

(6) **Executable code :** Contrary to transactions on the Bitcoin network, Ethereum transactions can be attached with executable code. This enables decentralized applications and conditional transactions, which are transactions that take place only when certain conditions are met.

(7) **Encryption type :** The two blockchain networks run different encryption Ethereum runs Ethash, and Bitcoin runs Secure Hash Algorithm 256 (SHA-256) encryption.

Table 5.1.1 : Bitcoin Vs Ethererum

	Bitcoin(BTC)	Ethereum(ETH)
Founded	2009	2013
Hashing Algorithm	SHA-256	Ethash
Consensus mechanism	POW	POW now;moving to POS with Ethereum 2.0
Time is taken to mine a block	An average of 10 minutes	An average of 12-15 seconds
Executable code	Doesn't use	Does use
Reward	12.5 BTC	3 ETH

5.1.4 Decentralized Applications (DApps)

GQ. Write a short note on DApps. **(4 Marks)**

- Let's evaluate the differences between decentralised and traditional apps. For instance, when you sign in to TBC.com, an HTML-rendered online application is displayed. To access your data (your information), which is centrally hosted, the page will request an API.
- The procedure is straightforward: your front end calls the backend API, which then pulls your data from a centralised database.
- When you log in, the same web application is displayed if we make this application decentralised, but it uses a smart contract-based API to retrieve the data from the blockchain network.

- The API is therefore replaced with a smart contract interface, and the smart contract will pull its data from the blockchain network, which serves as its back end.
- The blockchain network is a decentralised network rather than a centralised database, and all transactions made using the blockchain network's smart contract are validated (verified) by the network's users (the miners).
- Therefore, every action or transaction carried out on a TBC.com-like application after it was transformed will be a decentralised action or transaction.
- A backend code that executes on a distributed peer-to-peer network makes up a Dapp. The main distinction is that it enables direct communication between end users and the decentralised application providers since it is software created to operate on the Ethereum network without being governed by a centralised system.
- An application qualifies as a Dapp when it is open-source (its code is on Github), and it uses a public blockchain-based token to run its applications.
- A token acts as fuel for the decentralized application to run. Dapp allows the back end code and data to be decentralized, and that is the primary architecture of any Dapp.

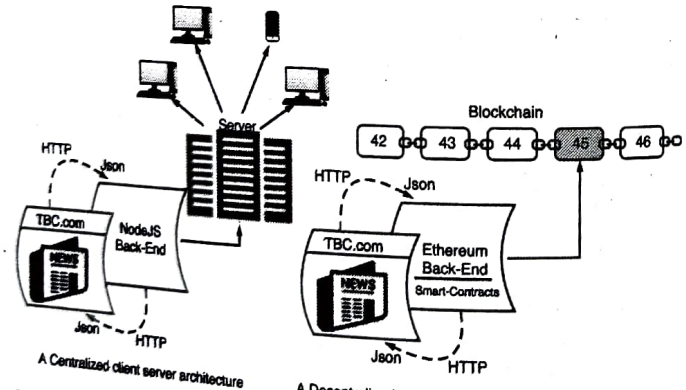


Fig. 5.1.6 : A comparison between Centralized Client-server architecture and Decentralized DApp architecture.

5.1.5 Decentralized Autonomous Organizations (DAOs)

GQ. Write a short note on Decentralized Autonomous Organization (DAO). **(4 Marks)**

- A DAO is a digital organisation that functions in a decentralised, democratic manner without the use of hierarchical administration.
- In summary, a DAO is an organisation where decision-making is preferably delegated to certain specified authorities or a group of designated individuals as a part of an authority rather than being centralised.
- DAOs rely on smart contracts for decision-making or, in other words, decentralised voting systems within the organisation since it lives on a blockchain network and is regulated by the protocols included in a smart contract.
- The voting mechanism, which is controlled by a decentralised application, must thus be used before any organisational decision can be taken.
- This is how it works. People contribute money through the DAO because it needs it to function and make decisions.
- Based on it, a token that represents each member's share in the DAO is issued to them. With those tokens, users may cast votes in the DAO, and the proposal's status is determined by which users cast the most votes.
- This voting procedure must be used for every decision made inside the organisation.

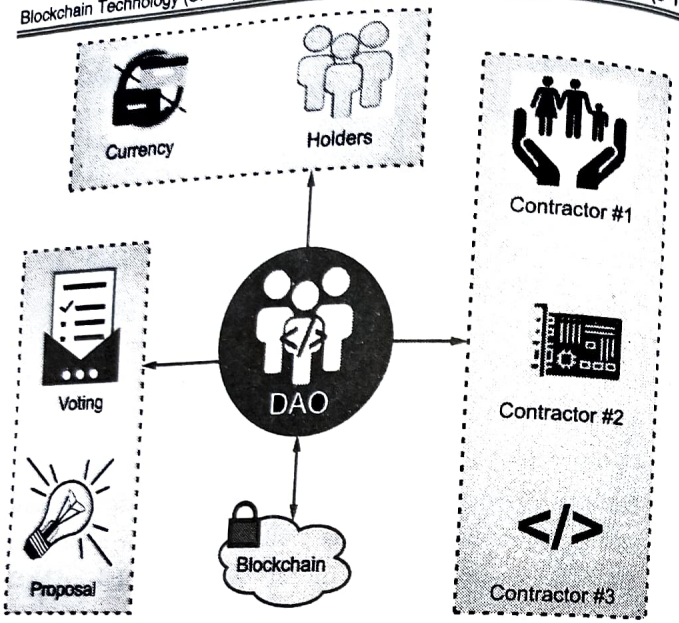


Fig. 5.1.8 : DAO's Process

5.2 THE ETHEREUM NETWORK

GQ. Enlist and explain three types of Ethereum network.

5.2.1 Types of Ethereum Network

The Ethereum network is a peer-to-peer network where nodes participate in order to maintain the blockchain and contribute to the consensus mechanism. Networks can be divided into three types, based on requirements and usage.

Mainnet

Mainnet is the current live network of Ethereum. The current version of main net is Byzantium (Metropolis) and its chain ID is 1. Chain ID is used to identify the network.

Testnet

- The commonly used test network for the Ethereum blockchain is known as Testnet. Before being implemented on the live production blockchain, smart contracts and DApps are tested on this test blockchain. Additionally, since it's a test network, study and experimentation are allowed.
- The primary testnet, known as Ropsten, has all the characteristics of various smaller and specialized testnets that were made for certain versions. Kovan and Rinkeby, for example, were created as additional testnets to evaluate Byzantium versions.

Private net

- As the name suggests, this is the private network that can be created by generating a new genesis block.
- This is usually the case in private blockchain distributed ledger networks, where a private group of entities start their blockchain and use it as a permissioned blockchain.

5.2.2 Ethereum Fork

GQ. What is Ethereum Classic? (2 Marks)

Ethereum Classic

- Ethereum Classic is the name of the original Ethereum blockchain.
- In 2016, a set of smart contracts on a platform known as The DAO, a decentralized autonomous organization, raised a record $150 million in an online crowdsale, the name of the crowdfunding method used to help support Ethereum.
- Shortly after that money was raised, an anonymous hacker stole $50 million DAO tokens.
- This resulted in the crypto community's decision to fork the network and to reappropriate the stolen funds. Forking is when the source code of an old open source program is used to create a new one.

- The network fork split the Ethereum blockchain into two: the original Ethereum Classic and Ethereum, the new one. This created competition between the two networks and became known as a hard fork. A hard fork is when nodes on the newest version of a blockchain no longer accept older versions.

5.3 ETHEREUM TOOL STACK

GQ. With the help of neat diagram, explain the Ethereum tool stack. **(6 Marks)**

GQ. What is testnet? Write down about Ethereum testnets. **(4 Marks)**

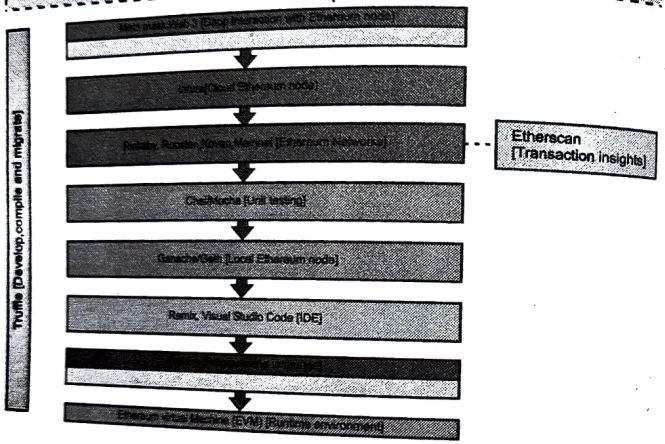


Fig. 5.3.1 : Ethereum tool stack

(1) **Truffle :** A world class development environment, testing framework and deployment framework for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier. The Truffle suite includes Truffle, Ganache, and Drizzle.

(2) **Ethereum Virtual machine :** EVM is intended to function as a runtime environment for compiling and deploying Ethereum-based smart contracts.

(3) **Solidity :** The smart contract language for Ethereum is Solidity programming language,

(4) **Remix :** Remix IDE allows developing, deploying and administering smart contracts for Ethereum like blockchains. It can also be used as a learning platform.

(5) **Ganache/Geth :** A personal blockchain for Ethereum development you can use to deploy contracts, develop your applications, and run tests. It is available as both a desktop application as well as a command-line tool (formerly known as the TestRPC). Ganache is available for Windows, Mac, and Linux.

(6) **Chai/Mocha :** Chai and Mocha are used for unit testing. Mochais a light-weight Node.js test framework, and Chaiis a Test Driven Development (TDD) assertion library for node.Both Mocha and Chai run in NodeJs and the browser and allow asynchronous testing. Although Mocha can be paired with any of the assertion libraries, it is delightfully paired with Chai most of the time. Chai provides us with several APIs like Assert, Expect/Should, and more. Mocha provides all the functionality required for automated testing in simpler ways.

(7) **Ethereum Test Networks :** A version of a project is released to an Ethereum Test Network ("testnet"), which simulates Ethereum Mainnet (the primary public Ethereum blockchain network) before to its release on the blockchain (or before modifications are made to the blockchain itself), giving developers, the community, and you an opportunity to try it out before real money is involved. It can be exciting to own 10,000 Ether or a trillion tokens on a testnet, even though they are simple to get and have no real-world worth. Currently, three testnets are operational, and each of them functions identically to the production blockchain (where your real Ether and tokens reside).

(i) **Ropsten :** The most comparable proof-of-work blockchain to Ethereum; it is simple to mine fake Ethereum.

(ii) **Kovan :** A proof-of-authority blockchain created by the Parity team is called Kovan. Ether must be requested rather than mined.

(iii) **Rinkeby :** A proof-of-authority blockchain created by the Geth team is called Rinkeby. Ether must be requested rather than mined.

(8) **Infura :** Infura is a Web3 backend and Infrastructure-as-a-Service (IaaS) provider that offers a range of services and tools for blockchain developers. This includes the Infura API (Application Programming Interface) suite. The flagship Infura Ethereum API is at the heart of the Infura Web3 service.

(9) **Metamask :** MetaMask is a cryptocurrency wallet that enables users to store Ether and other ERC-20 tokens. The wallet can also be used to interact with decentralized applications, or dapps.

5.4 ETHEREUM APPLICATIONS AND USE CASES

GQ. Elaborate on various applications of Ethereum. **(4 Marks)**

- According to the Ethereum Foundation, Ethereum can be applied to codifying, decentralizing, securing and trading almost everything. Its uses include the following :
  - (i) crowdfunding
  - (ii) financial exchanges
  - (iii) company governance
  - (iv) domain names
  - (v) intellectual property
  - (vi) smart property with hardware integration
  - (vii) voting
  - (viii) contracts and agreements
- Ethereum can be used for the following purposes :
  - (i) buy and sell cryptocurrencies like Ether and other fungible, Ethereum Request for Comments 20-validated tokens;
  - (ii) host smart contracts and decentralized apps (dApps);
  - (iii) carry out decentralized finance activities;
  - (iv) exchange nonfungible tokens;

(v) power Ethereum-based enterprise software independently from the public Ethereum chain; and

(vi) play Ethereum-based video games, where users can earn real cryptocurrency from playing the game.

5.5 BENEFITS OF ETHEREUM

GQ. What are the pros of Ethereum? **(4 Marks)**

Many benefits of blockchain technology apply to Ethereum, including the following :

(1) **Decentralization :** Due to the decentralised nature of Ethereum, there is no influence from outside cloud service providers. Peer-to-peer transactions are made possible through the usage of blockchain. In contrast to certain other software systems, which frequently need faith in a central authority, users can exchange value or store data without the requirement for an intermediary.

(2) **Availability :** Ethereum is decentralised, so if a node goes down there is no downtime. Other computer architectures rely on centralised servers, and interruptions can have an impact on performance.

(3) **Privacy :** Users have the option to remain anonymous when utilising the network for exchanges. To utilise an Ethereum application, they don't have to input their personal information.

(4) **Security :** Ethereum is made to be impossible to hack, much like any other decentralised blockchain-based network. To exploit the network, hackers would need to gain control over majority of the nodes.

(5) **Permissionless :** Because Ethereum is a permissionless blockchain, anybody may take part. In contrast, permissioned blockchains are only accessible to certain individuals.

(6) **Less ambiguity :** Stronger contracts are guaranteed by the hardcoded smart contracts that are the basis of trading and agreement on Ethereum. A freelancer who receives work through a dApp on Ethereum, for instance, may sign a smart

contract with the hardcoded clause "X remuneration will be delivered when Y job is performed." This is distinct from regular contracts, which call for interpretation and execution.

5.6 DRAWBACKS OF ETHEREUM

GQ. What are the pros and cons of Ethereum? **(4 Marks)**

The Ethereum platform has been criticised in two ways :

(1) **Resource-intensive :** The PoW consensus technique is being used by Ethereum as a resource-intensive method of ensuring that all network nodes agree on the status of all data stored on the blockchain. Every blockchain node stores every smart contract, and each node concurrently performs every smart contract's computations.

(2) **Security :** Additionally, the PoW approach creates a security concern. Smart contract flaws on the open blockchain are readily apparent to everyone and can be harder to fix than to exploit.

- When Ethereum switches from a PoW consensus algorithm to a PoS consensus algorithm in 2022, these criticisms will be addressed. By providing greater mining or block validation power to miners with more coins, PoS is anticipated to increase the Ethereum blockchain's energy efficiency. Fewer nodes are required to perform more work as a consequence. Additionally, no specialised equipment is needed; all that is needed are the currencies needed for mining and an internet connection.
- The fact that greater mining power is concentrated within a smaller group of miners is a drawback of this approach. More manipulation and collaboration are made possible by this on the network.
- Other drawbacks of Ethereum include its high entrance barrier, high cost of development, and difficulty of usage for those who are not familiar with the technology.

5.7 ETHEREUM 2.0

- The Ethereum network will undergo a significant update in 2022 known as Ethereum 2.0, also referred to as ETH 2.0 or Serenity.
- The objective is to boost the network's transaction throughput to tens of thousands of transactions per second from 15 transactions per second currently.
- It will do this by distributing workloads across a large number of parallel blockchains that all use a single consensus PoS blockchain. Any threat actor would find it too expensive to carry out the act of maliciously altering any one chain, which would need altering the common consensus.

5.8 SMART CONTRACTS

GQ. Write a short note on Smart contracts. **(6 Marks)**

GQ. How smart contracts works? **(4 Marks)**

- Smart contracts are digital transaction protocols that, if certain conditions are met by all parties, verify, control, and self-execute an agreement contained in digital codes on a blockchain.
- These contracts take place between anonymous parties and are automatically enforced without the assistance of any third party, in contrast to traditional (physical) ones.
- Signatories (parties), the subject of the agreement, and the conditions of the agreement are its three primary components. For the transaction to be successful, all parties involved must adhere to the terms of the agreement (a set of guidelines and penalties).
- Executing agreements using digital contracts is said to be safe, cost-effective, and eliminates the need for a middleman.
- Furthermore, the decentralised nature of the blockchain network guarantees the transparency, traceability, and irreversibility of all transactions.

Smart Contracts

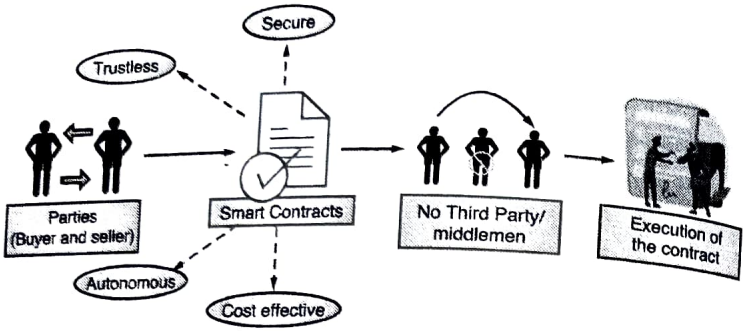


Fig. 5.8.1 : Smart contracts

- Two or more individuals, organisations, or governments are parties to a traditional (physical) contract. Therein, you sign an agreement, then you employ a third party to carry it out because you have faith in them.
- This third party might be a government agency, a lawyer, or any other type of organisation. Its purpose is to handle the procedures and carry out the contract.
- This raises the expense of auditing and enforcing laws as well as the risk of financial loss brought on by fraud and data manipulation. With smart contracts, the contract is written in code.
- The outcome is validated by the users of the Ethereum blockchain-based network rather than a centralised authority.
- The threat of any data tampering or alteration is eliminated once a contract is executed since the transaction is registered and cannot be changed or interfered with.
- Consider the situation where Bob hired Alice to create the website for his business and paid her $500 for the job.
- The smart contract's agreement is built by the developers using the Ethereum programming language.
- The smart contract has all the conditions (requirements) for building the website. Once the code is written, it is uploaded and deployed on the Ethereum Virtual Machine (EVM).

- A smart contract may be executed using EVM, a runtime compiler. Every member of the network owns a copy of the contract once the code is installed on the EVM.
- Each node on the Ethereum network will review and certify that Alice's work has been completed in accordance with the coding specifications when Alice submits it for evaluation.
- Once the work is authorised and validated, the $500 contract will execute automatically, and Alice will be paid in ether. Alice will be paid $500 in ether, and Bob's account will be promptly debited.
- Nick Szabo, an American computer scientist and cryptographer, first used the phrase "smart contracts" in 1994 while attempting to carry out the conditions of a contract utilising distributed ledger technology and automated transaction protocols.
- Smart or self-executing contracts are computer programs created on a blockchain that facilitate transactions when parties satisfy a predetermined set of conditions. Also, there is no need for the parties to rely on an intermediary for the agreement's validation and execution.

Smart Contracts Functioning

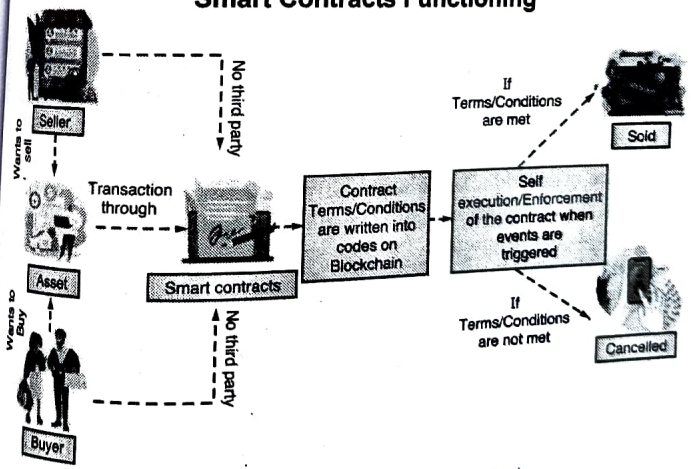


Fig. 5.8.2 : Working of smart contracts

The process

- There are two parties (a buyer and a seller) interested in purchasing and selling an item.
- These two parties establish a smart contract, which is a fully digital and self-executing agreement, with its terms or conditions written in codes on a decentralised blockchain network. When all parties agree to these conditions, the transaction is completed.
- The smart contracts platform provides top-notch security and total transparency. Additionally, it prevents data manipulation and enables the two parties to follow the transaction. The parties involved's identities, however, are kept private.

Examples

- In fields including property rights, intellectual property, banking and insurance, legal services, e-government, crowdfunding, etc., smart contracts instances are common. Let's think about the following instances to help us better understand the concept:
- A group of investors proposes to fund a business project idea from the ABC team. Both parties agree to the conditions of a smart contract, which is codified and contains a list of guidelines and penalties. The blockchain will send the funds to ABC if the project proposal appears to be legitimate according to the protocols.
- On the other side, the blockchain will return the funds to the group if the project concept doesn't look suitable in view of the contract requirements. In this example, the contract stores and validates transaction information and self-executes the contract only if the relevant event triggers.
- A customer agreeing to pay a seller for delivering particular products on a certain date is another real-world example. The terms of the contract specify the payment amount and the deadline for receiving the products. If either participant doesn't follow through, the transaction will be held in the blockchain.

- Ethereum wallets are popular blockchain-based cryptocurrency apps that need an Ethereum account from the user. They are able to conduct financial transactions without using a bank or any third party.
- In order to provide high-end safety to wallet users, open-source blockchain Ontology has announced a partnership with blockchain distribution network bloXroute Labs, Inc. Ethereum smart contracts will be more secure and safe for users because to its integrated design with the Ethereum Virtual Machine.
- Smart contracts in finance can help streamline and accelerate a variety of financial services.
- They can be used, for instance, by insurance firms to establish official agreements and resolve disputes.
- Similar to how bond issues can issue bonds for trading that complies with regulations, stock markets can set securities trading rules in these contracts. Banks may use these contracts in the same way to handle syndicated loans more quickly and with fewer operational risks.

5.8.1 Types of Smart Contracts

GQ. What are the types of smart contracts. **(2 Marks)**

(1) **Smart Legal Contract :** The most traditional type of smart contract is a smart legal contract, which has the same legal requirements as its traditional counterparts (i.e., mutual assent, expressed by a valid offer and acceptance; adequate consideration; capacity; and legality) and is used to hold parties responsible for upholding their end of an agreement. When correctly configured, a smart contract is legally binding and requires the parties to uphold their duties; failing to do so may result in immediate legal action being taken against the party in violation by the smart contract.

(2) **Decentralized Autonomous Organizations :** Communities on the blockchain are known as Decentralized Autonomous Organizations, or DAO. A set of established guidelines that are defined using smart contracts can serve to define these

communities. Every person is bound by the community's rules, and it is up to each individual to uphold those laws. These regulations, which are composed of several smart contracts, collaborate to keep an eye on community actions.

(3) **Application Logic Contracts :** Application Logic Contracts, or ALCs, are blockchain contracts that incorporate application-based code that keeps up with other contracts on the network. They allow communication between various devices, such as when blockchain technology and the Internet of Things (IoT) are combined. ALCs are an essential component of multi-function smart contracts and are often managed by a programme.

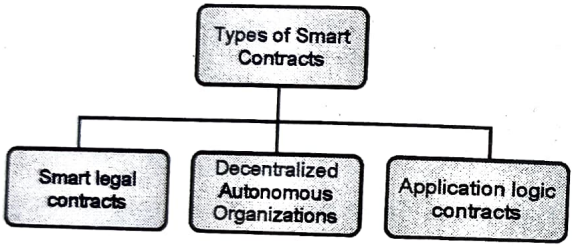


Fig. 5.8.3 : Types of Smart contracts

5.8.2 The Need for Smart Contracts

GQ. What is the need of smart contracts? **(4 Marks)**

- As we all know, smart contracts fall into one of these three types and offer a wide range of capabilities.
- The inherent ability of smart contracts to bring transparency, Time Efficient, precision, safety, cost effectiveness, and trust to transactions, even in sectors that have historically lacked these qualities, drives the need for them.
- A smart contract may be created by anybody and released across the network.

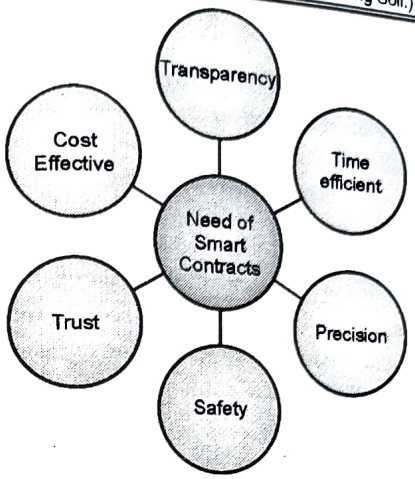


Fig. 5.8.4 : Need of Smart contcats

Transparency

The parties' understanding of the terms and conditions is complete. Furthermore, users have a very straightforward means of confirming the elements that would affect them and the contract beneficiaries since the execution of the programme or the smart contract requires a few specific inputs.

Time Efficient

- As previously noted, whenever a control variable or a user call activates a smart contract, it starts working right away.
- The blockchain and other sources in the network make data instantly available to the system, so it doesn't take very long for the execution to validate, process, and settle the transaction.
- For instance, transferring property title deeds, which often requires weeks of human verification of mountains of paperwork, may be completed in a matter of minutes or even seconds with the use of smart contract software that verifies the parties and the papers.

Precision

Since the platform is essentially just predefined computer code, there can be no subjective errors and all the findings will be accurate and faultless.

Safety

- Every block of data on the blockchain is cryptographically encrypted, which is a fundamental aspect of the technology. In other words, even while the data is redundantly stored on a large number of network nodes, only the owner of the data has access to see and utilise the data.
- Similar to this, every process will be 100 percent safe and impenetrable when it uses the blockchain to store crucial inputs and results. By giving auditors a native, unaltered, and non-repudiable version of the data chronologically, the same also makes auditing and regulatory affairs simpler.

Trust

- The fact that smart contracts will never display subjectivity or bias in carrying out the agreement implies that all parties involved are totally committed by the outcomes and can completely rely on the system.
- This also implies that the "trusted third-party" necessary in traditional transactions of vital significance is not necessary in this case.

Cost effective

- As seen in the example, using a smart contract has very little expenses. Businesses typically employ administrative personnel whose only responsibility it is to ensure that the transactions they do are legal and comply with laws.
- Duplication of effort was inevitable if there were numerous parties participating in the transaction. Due diligence may be completed concurrently by both parties in smart contracts, effectively eliminating redundancy.

5.8.3 The Drawbacks

GQ. What are the drawbacks of smart contracts. (2 Marks)

- This is not to claim that there aren't any issues with the use of smart contracts. Development has also been slowed down in this area by similar issues.
- It is practically hard for the parties concerned to amend or add new terms to current clauses without considerable reworking or legal consequences due to the tamper-proof nature of everything in blockchain.
- Second, even though everyone may view and watch activities on a public blockchain. It is not always known who the parties are that are engaged in a transaction.
- This anonymity raises concerns about legal impunity in the event that either side breaches the agreement, particularly in view of the fact that present laws and politicians are not very technologically tolerant.

5.9 INTRODUCTION TO SOLIDITY

- The programming language Solidity is used to create a smart contract on the Ethereum network. We will study the fundamentals of Solidity and create our first smart contract, "Hello World," as well.
- Remember that Solidity is not an object-oriented programming language; rather, it is contract-oriented. Each state of a smart contract is unique.
- You may store the data on the blockchain by utilising a state. However, it costs some Gas to store data; it is not free. As a result, anytime you write code on the public blockchain, that each block should use a minimum quantity of gas.
- Initially, there is no setup required to start with Solidity. Ethereum provides an online IDE called Remix, on which you can develop and deploy your smart contract.

5.9.1 Remix Introduction

- When you open the Remix in your browser, you will see the screen as below.

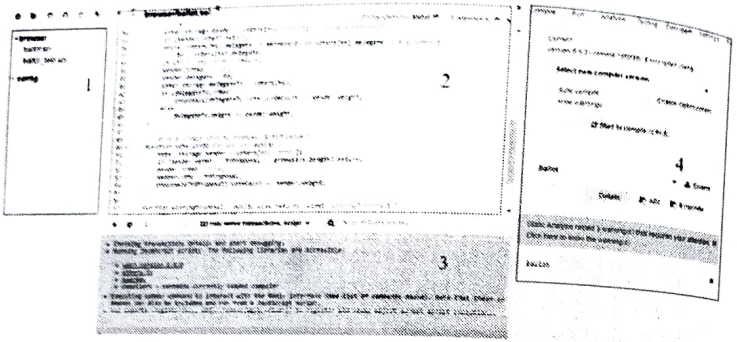


Fig. 5.9.1 : Remix IDE

There are four different sections

(1) **File explore :** We can handle our contract files with file explorer. B y default, all files are saved in your browser; deleting the browser's storage will thus result in the permanent deletion of all the Solidity files you added. Other options available in Explorer include transferring all files to a different instance and exporting all contract files to GitHub.

(2) **Code Editor :** Your primary portion will be the Code Editor, where we will write the Solidity code. Every time the current file is modified, the Editor recompiles the code. Additionally, there are options for font size and syntax highlighting.

(3) **Terminal :** As you work with Remix, the log and debugging information is displayed. You may look up the details of a transaction you made or an event there.

(4) **Deploy And Test :** Options for analysing, compiling, deploying, and testing your smart contract are provided in the fourth part. The auto-compile option is available here. It will compile the smart contract every 5 seconds if you enable that option.

Create a Contract

First, we'll delete the ballot.sol and ballot test.sol default contract files and then, using the code below, add a new file with the name myContract.sol.

```
pragma solidity ^0.5.0;

contract helloWorld
{
  function printHelloWorld() public pure returns (string
memory)
  {
    return 'helloWorld';
  }
}
```

Fig. 5.9.2 : Sample smart contract

Here, we have created one smart contract named helloWorld. Let's understand the contract structure in brief.

Version Pragma

- We should annotate the source file with the compiler version before doing any coding.
- The system will be informed that the code you are about to write should be compatible with the specified compiler, ensuring that your code will not malfunction even in the event that later versions introduce incompatible updates.
- The following uses the pragma version.

```
pragma solidity ^0.4.0;
```

- Such source files cannot be compiled with versions prior to 0.4.0. The following code can be compiled on 0.4.0 or 0.4.x but should not be compiled on 0.5.0 or later since the (^) symbol here denotes the higher version.
- The pragma version can be written in a variety of ways. You may alternatively write as follows for greater clarity; it functions the same as explained.

```
pragma solidity >=0.4.0 <0.5.0;
```

Contract structure

```
contract [contract name]
{
   function [method name]()
   [visibility specifiers] [modifiers] [returns] ([return type(s)]){
  //code here
}
...
..
}
```

Deploy Contract on Remix

- To deploy the smart contract, go to the "Run tab" right upper corner, and click on the deploy button. Make sure right contract is selected if there are multiple contracts present.

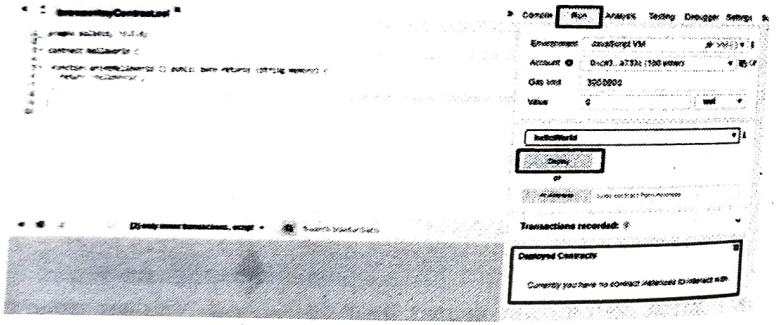


Fig. 5.9.3 : Deployment of smart contract

- As there is no deployed contract at this point of time, "Deployed Contract" section has no information.
- Also note we're going to execute this contract on JavaScript VM, which is the sandbox blockchain environment and it won't be persisted in any kind of state information. Page refresh will start a new blockchain again from the first.

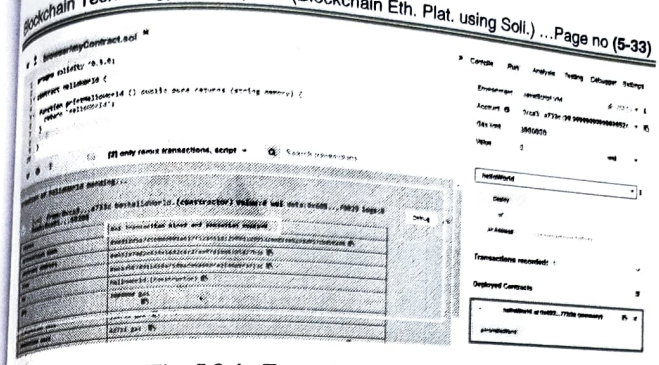


Fig. 5.9.4 : Execution of smart contract

- That section will be updated with contract information after our smart contract has been deployed, and the method should be visible there.
- The information about the transaction we performed is currently updated on the terminal. From the terminal, you can also monitor gas use, contract addresses, etc.

Cost of deployment

- In our case, before deployment, we had 100 ether in my account, and some ether was consumed during deployment. Anything that modifies the state of the blockchain will cost money.

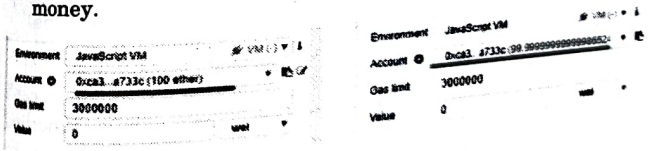


Fig. 5.9.5 : Cost of deployment

Run Contract on Remix

Next to run our method, click on printHelloWorld text which appears as a button.

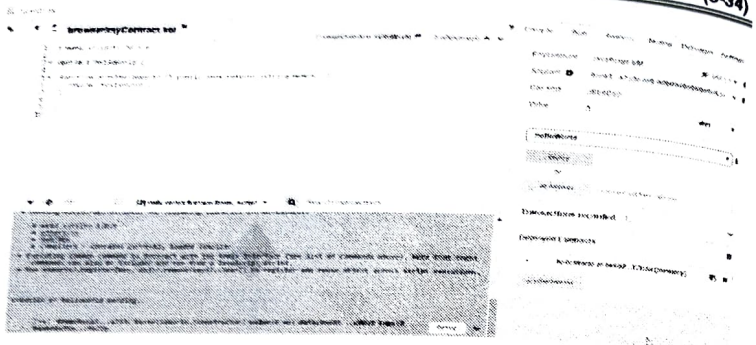


Fig. 5.9.6 : Outcome

5.9.2 Variables in Solidity

GQ. What are different types of access modifiers in solidity. **(2 Marks)**

GQ. What do you mean by State and local variable. Give example. **(2 Marks)**

Access Modifier

Access Modifiers are the keywords used to specify the declared accessibility of a function or a type. There are four access modifiers available in Solidity.

Public	Private	Internal	External
The Public element can be inherited and can be accessed by external elements. All can access a public element.	The Private element doesn't get inherited and can't be accessed by external elements. It can be accessed from the current	The Internal element can be inherited but can't be accessed by external elements. Only the base contract and derived contract can	The External element can't be inherited but it can be accessed by external elements. Current contract instance can't access external

contract instance only.	access internal element.	element, it can be accessed externally only.

Table 5.9.1 : Access modifiers in solidity

Variable Declaration

Variable declaration in Solidity is a bit different; you have to specify the type (data type) first, followed with an access modifier, and the variable name. If you would not specify any access modifier, it will be considered as a private.

Structure

<type> <access modifier> <variable name> ;

Example

uint **public** a;

In Solidity, there are primarily two types of variables: State variables and Local variables. Like any other language, we consider them as global and local variables. There are some differences, though.

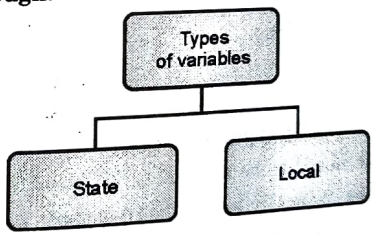


Fig. 5.9.7 : Variable types

State variable

- State variables permanently store the value in contract storage. What would you do if you were using C# or another language and wanted to keep user information for a long time? After establishing a connection with a database server, save the data to the database. Instead of creating a connection, Solidity allows you to simply utilise state variables to store data forever.

- Each method has its own scope, and the State variables should define outside of the scope of any defined functions.

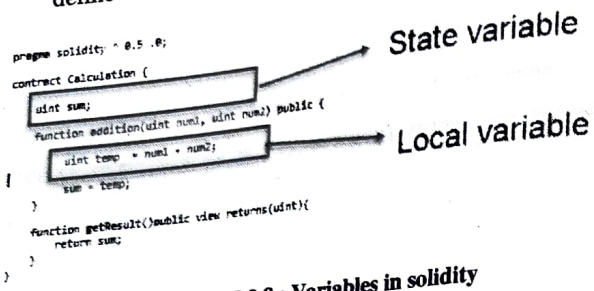


Fig. 5.9.8 : Variables in solidity

☞ Local Variable

- A local variable can only be accessed from inside the function due to its internal context. These variables are typically used to store temporary values while processing or performing calculations.
- The local variable "temp" in the top screen is only used inside the "addition" function.

5.9.3 Types

- Solidity is a statically typed language, which means the type of each variable needs to be specified. Declared types have some default values, typically called "zero-state". For instance, the default value for bool is false.
- There is no concept of null or undefined in Solidity! So, no need to handle null reference exception .
- There are two kinds of types in Solidity: *value types* and *reference types*. The Solidity data types can be classified according to the data location. If variable store its own data; it is a value type. If it holds a location of the data; it is a reference type.

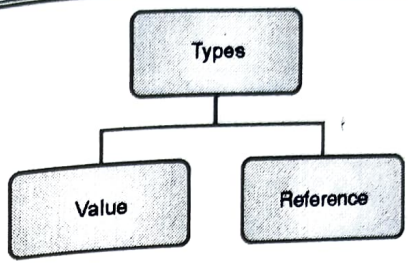


Fig. 5.9.9 : Types

Value Type

These variables are passed by value, it means that they are copied when they are used either in assignment or in a function argument. Following are basic value types.

Types	Description
Booleans	Booleans have possible values as true or false, and can be used with a different operator for conditional checking.
Integers	It stores the values in a rage of 8 bits to 256 bits. Un-signed integer holds positive values and signed integer holds positive as well as negative values. int and uint are the aliases for int256 and uint 256, respectively. Best practices for integers is to specify the bits value while you declare them, so, you would use minimum space and the cost of storing would be lesser. Thus, use uint8 or uint16 instead of using int (uint 256) always.
Fixed Point Numbers	*Fixed point numbers are not fully supported by Solidity yet. They can be declared, but cannot be assigned to or from.* However, you can use floating point number for calculation, but the value coming out of from the calculation should be an integer value.

Types	Description
Bytes and Strings	Bytes are used to store a fixed size character set. Basically, the length of bytes is from 1 to 32. If you want to store more than that, you can use string, which has dynamic length. An advantage of bytes is, bytes1 to bytes32 use less gas, so they are better to use when you know how long data you have to store.
Enums	Enum is a way to create user-defined types, it used to assign names to integral constants, which makes the contract more readable and maintainable. Options in enum are represented by subsequent unsigned integer values starting from 0.

Table 5.9.2 : Solidity variable Types

Reference type

- Reference types do not store the data directly to the variable, instead, it stores the location of the data. With a reference type, two different variables can reference the same location, in such a case; any change in one variable will affect to another variable.
- Since the version 0.5.0 of Solidity, for all complex types, you need to define data location explicitly with a variable.

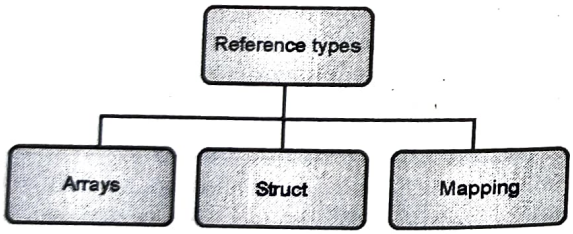


Fig. 5.9.10 : Reference types

Reference types consist of,

(1) **Arrays :** Arrays are the groups of variables of the same type in which each individual variable has a particular location called

index. By using that index location, you can access that variable. The size of arrays can be fixed size or dynamic.

```
uint[5] arrayName;
uint[] arrayName;
```

(2) **Structs :** Structure is the group of different types although it is not possible to contain a member of its own type. Structure is a reference type variable and can contain both - value types and reference types.

Declaration	Example

struct <name of structure> {
<type> variable1;
<type> variable2;
}
	struct User {
	string name;
	uint age;
	bool isValid;
	}

(3) **Mapping :** Mapping types are the most used reference type; they are used to store data in a key-value pair; where the key can be any built-in value types or byte and string. You can think of it as a hash table or dictionary as in any other language, in which a user can store data in a key-value format and data can be retrieved by key.

Declaration	Example
mapping(_KeyType => _ValueType) <access specifier> <name>;	mapping (address => uint) account;

- The main difference between the value type and reference type is Data location. Arrays and Structs have additional data location which specifies where data (value of the variable) should be stored.

5.9.4 Function And Address in Solidity

GQ. List down different types of solidity functions. Give example. **(4 Marks)**

GQ. What are the different kinds of address types in solidity? Explain with example. **(4 Marks)**

In Ethereum smart contract, address and function are broadly used value types.

Function Type

Solidity functions are methods used to perform some specific task. They get invoked during the execution of a smart contract.

Function Structure

function (<parameter types>) {internal | external | public | private} [pure | view | payable] [returns (<return types>)]

- Function parameter types are the input variables. It can be used as any other local variables with an assignment. Return variables are used to return something from the function.
- We can pass return variables with "returns" keyword. In Solidity function, you can also return two or more values.

```
pragma solidity ^0.5.0;

contract Types
{
  uint sum;
  function result(uint _a, uint _b) public returns(uint)
  {
    sum = _a + _b;
    return sum;
  }
}
```

Fig. 5.9.11 : Functions in solidity

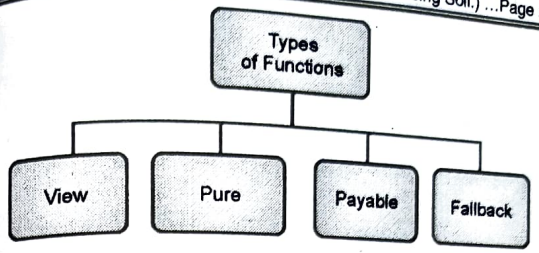


Fig. 5.9.12 : Function type

View Function

```
pragma solidity ^0.5.0;

contract Types
{
   function result(uint a, uint b) public view returns (uint)
   {
      return a + b + now;      //"now"
is current block timestamp in

//seconds in unix epoch format
   }
}
```

Fig. 5.9.13 : View functions

Functions which will not alter the storage state can be marked as a view function. In simple terms, it is used for viewing the state.

Pure Function

A function that does not modify or read the state, is called a pure function.

```
pragma solidity ^0.5.0;

contract Types
{
   function result(uint a, uint b) public pure returns (uint)
   {
      return a * (b + 42);
   }
}
```

Fig. 5.9.14 : Pure functions

Payable Function

- Payable Functions allows to receive Ethers while it being executed, means that, if someone sends some Ethers to the smart contract, and it doesn't have a payable function, then smart contract won't accept ether and transaction will get failed.
- To catch that transfer amount, the smart contract needs to have a payable function.

```
pragma solidity ^0.5.0;
contract Types
{
   uint receivedAmount;

   function receiveEther() payable public
   {
      receivedAmount = msg.value;
   }
   function getTotalAmount() public view returns (uint)
   {
      return receivedAmount;
   }
}
```

Fig. 5.9.15 : Payable functions

Fallback Function

- Solidity supports a parameterless anonymous function called Fallback function. One contract can have only one fallback function, and it must be defined with external visibility.
- Generally, a Fallback Function is used to receive Ether with a simple transfer, when someone called it without providing any data.

5.9.5 Address Type

- On the Ethereum blockchain, every account and smart contract has an address and it's stored as 20-byte values.
- It is used to send and receive Ether from one account to another account. You can consider it as your public identity on the Blockchain.
- In Solidity, address type comes with two flavors, address and address payable.
- Both address and address payable stores the 20-byte values, but address payable has additional members, transfer and send.

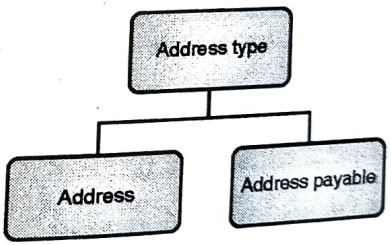


Fig. 5.9.16 : Address type

Address

Address type defines with *address* keyword.

address myAddress;

An address is used to store the value of any address. In below example, "caller" is an address type.

```
pragma solidity ^0.5.0;

contract Types
{
    address public caller;

    function getCallerAddress() public returns (address)
    {
        caller = msg.sender;
        return caller;
    }
}
```

Fig. 5.9.17 : Address type

Address payable

Address payable has an additional keyword "payable"

(1) address payable caller

When we want to transfer some funds to the address, we need to use address payable. There are two members to perform a transfer, send and transfer. Both are doing the same thing, but the difference is; when the transaction gets failed, "send" will return a false flag, whereas, "transfer" throws an exception and stop the execution.

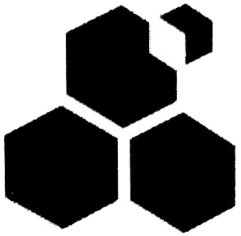
5.10 SWARM

GQ. What is SWARM? Which elements are used in it. **(4 Marks)**

GQ. Enlist and brief about different DApps employing SWARM. **(4 Marks)**

GQ. Write a shot note on SWARM. **(6 Marks)**

- Swarm is a decentralised storage, service, and communication platform created to provide dApp developers with a permissionless, censorship-resistant architecture.
- Swarm, which is based on the Ethereum web3 stack, seeks to offer a variety of Web 3.0 services, including as chat, streaming audio and video, and database hosting.



- By giving consumers back control over their data, Swarm aspires to become "the operating system of the re-decentralised internet."
- Gavin Wood, a co-founder of Ethereum, developed Swarm after starting to work on the platform's software architecture in 2015. With the help of Vitalik Buterin, a fellow Ethereum pioneer, Wood set out to create a Web 3.0 storage and service solution that would be censorship resistant, DDOS resistance, and provide zero downtime. Swarm, which is built on Ethereum, makes advantage of the blockchain's security and smart-contract features as well as its developer community. Viktor Trón has provided extensive documentation on the ideas and specifics of Swarm in his so-called Book of Swarm.
- Swarm is a representation of a communication and storage network that intends to someday provide the fundamental infrastructure for a completely decentralised internet, with digital services distributed throughout a wide global network of nodes.
- Swarm's front-end user interface is similar to that of the World Wide Web, but the network's back-end is different from traditional internet usage since data is hosted on a peer-to-peer architecture rather than on centralised servers.
- Due to its incentive system, the decentralised infrastructure is intended to be self-sustaining. Users may exchange resources for network services like data storage and distribution, with payments handled by Ethereum smart contracts and powered by the native BZZ currency.

- These elements make up the decentralised storage mechanism used by Swarm:
  (1) **Chunks** : Data saved on Swarm is divided into chunks, which are 4KB or smaller blocks. A 32-byte hash of the content in each chunk identifies it.
  (2) **Reference :** A unique file identifier that facilitates the retrieval of data stored in chunks for clients.
  (3) **Manifest :** A data structure that allows for URL-based content retrieval.
- The Manifest uses the unique reference to identify the appropriate data chunks when a client requests content on Swarm, allowing the nodes containing those chunks to be contacted for retrieval.
- Similar to this, data that is uploaded to Swarm is divided into chunks that are distributed between nodes and given a timestamp for identification.
- Smart contracts control the built-in BZZ incentives, which are distributed to nodes who make their resources accessible for file storage.
- Swarm features built-in redundancy to provide ongoing data accessibility, to guard against nodes leaving the network, and to defend against DDOS attacks.
- The native token of Swarm is called BZZ. It powers network transactions and rewards nodes for their resource contributions. Greater BZZ holdings will influence Swarm governance voting more, similar to other stake-based blockchain governance systems. Swarm airdropped 1 million BZZ, called "The Rise of the Bee," to early testnet members in June 2021.

☞ Uses

- Swarm enables dApp developers to safely and effectively store and deliver data and content to blockchain customers.
- The node-to-node message capability, scalable state-channel infrastructure, database services, and media streaming services are all features of the Swarm base-layer architecture.
- In 2020, Swarm started distributing Swarm Grant Waves to

promote network adoption and broaden the ecosystem. The Grant Waves provide developers with mentoring as well as cash assistance for Swarm initiatives.

- Swarm has been included into a variety of dApps, supported by the distribution of funding, including:
  (1) **Etherna :** A decentralised open source video platform that prioritises content durability, creator rewards, and censorship resistance.
  (2) **Zetaseek :** It is a search engine built on the blockchain for individual users that is intended to organise "files, links, and references" in information that has been posted to the Swarm network.
  (3) **Scaleout :** A platform for data storage that focuses on end-to-end privacy, the use of DevOps tools, and advanced machine learning.
  (4) **Boma :** A communication and engagement platform with a privacy-focused approach that offers event organisers a variety of features like engagement data, CMS capabilities, galleries, and audio and video streaming.
  (5) **Giveth :** It is a decentralised platform for non-profit organisations to raise money that offers complete accountability and transparency while facilitating donor communities.

☞ Outlook

- Swarm is looking at the possibility of using the blockchain's storage and communication capabilities to build the foundation of the decentralised internet, commonly known as a "world computer."
- The Swarm roadmap outlines a variety of short-term objectives in order to achieve that aim, including features and functionality including node splitting, browser compatibility, large-scale network simulations, and support for lightnodes.

5.11 WHISPER (DECENTRALIZED MESSAGING PLATFORM)

GQ. What the three types of resources foundational technologies Ethereum need in order to deliver services? **(4 Marks)**

GQ. What is Whisper? List down the use cases of it. **(4 Marks)**

GQ. Explain any four fundamental features of Whisper. **(4 Marks)**

GQ. Write a shot note on Whisper. **(6 Marks)**

Whisper is an Ethereum's inter-application communication protocol.

- One of the three fundamental requirements for decentralised applications is messaging.
- Compute, Storage, and Messaging are the three types of resources that non-trivial programs usually need in order to deliver services.
- Created with the grand vision of building a global decentralised computing platform, Ethereum serves these basic needs with three pieces of foundational technologies: the EVM (Ethereum Virtual Machine) provides compute, Swarm handles storage of large files, and Whisper is the answer to messaging.

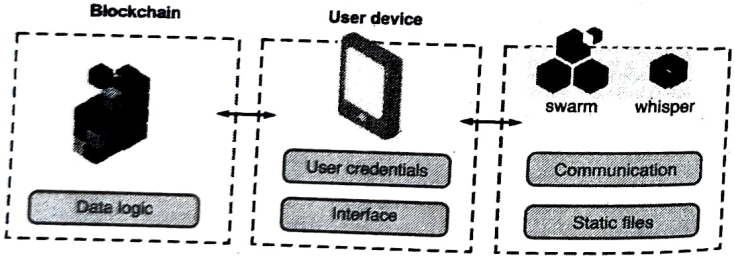


Fig. 5.11.1 : Foundational technologies

- In a nutshell, Whisper is a peer-to-peer (P2P) messaging protocol for decentralised applications (Dapps) to provide Dapp developers a simple API to send and receive messages in almost complete secrecy.

- Due to this concern on privacy, Whisper's creators had to make some unusual trade-offs between performance and privacy. Whisper is therefore more suited to a certain group of use scenarios.

Use Cases for Whisper

What Whisper is good for

- **Publish-subscribe coordination signaling.** Dapps could collaborate with one another by implementing the pubsub pattern with Whisper.
- **Secure, untraceable communication.** Whisper is designed from the ground up to support highly private and secure communication with plausible deniability.

What Whisper isn't good for

- Real-time, extremely low latency communication. It is challenging to ensure latency for time-sensitive applications since Whisper messages may be routed in a probabilistic manner.
- sending massive data sets. Whisper works best for messages that are under 64 KB in size. Another channel built for content distribution, like swarm, would be a better option for bigger messages.

Whisper Fundamentals

Whisper is a programmable messaging protocol that offers dapp creators a great deal of freedom in managing the security and privacy constraints on their communications. Whisper's functionality must be understood, at least at a high level, in order to properly benefit from it.

A network of equal peers

- The Whisper network is made up of decentralised systems connected together as nodes. A node creates this network by utilising the ĐΞVp2p protocol to identify its peers.
- As the basis upon which the protocols in the Ethereum stack are constructed, ĐΞVp2p is a crucial bit of technology in the Ethereum ecosystem. Whisper and its sister technologies

Ethereum and Swarm do not interact despite using the same wire protocol.

- The common use of the same implementation by Whisper and Ethereum nodes is only a coincidence brought on by practicality.

Identity-driven communication

- A Dapp instance can begin receiving messages from a node after it has connected to Whisper by creating an identity there. Although it is required to create two-way communications, an identity is not absolutely necessary in order to send messages. This raises interesting use cases and challenges.
- In general, a Whisper identity is an entity (an individual or a group) that consumes messages. An identity can be viewed practically as the owner of an encryption key.
- Therefore, one has to generate an encryption key in order to receive Whisper messages. For various use cases, symmetric (AES-256) and asymmetric (secp256k1) keys are both supported.
- Encryption ensures that only the intended recipient(s) can access the content of a piece of message. If a node can decrypt a piece of message, then the message is intended for a recipient using that node.

Delivering Messages in Darkness

- Whisper makes a big deal out of its ability to communicate in the dark. This means that two nodes in a Whisper network can communicate without leaving any traceable evidence to traffic analyzers and other peers, even if those peers participated in the message routing. Performance is traded for privacy in order to achieve this.
- Two fundamental requirements must be met in order to achieve complete communication darkness: the content of the message must be unavailable to those who intercept the messages, and communicating nodes must be difficult to identify.

- Whisper uses encryption as its primary method of message transmission; it is not feasible to send unencrypted communications using Whisper. Routing information must also be concealed in order to conceal the fact that two nodes are communicating with one another.

Messages are addressed to no one

- The sender Dapp makes an API call to its Whisper node to encrypt the message with a shared symmetric key or the recipient's public key and then enclose the encrypted message in an envelope. A Whisper envelope provides metadata to aid with routing and processing, just like its physical equivalent in the real world.
- Whisper envelopes do not have any recipient information, in contrast to regular envelopes. A typical Whisper envelope looks like this:

```
{Version, Expiry, TTL, Topic, AESNonce, Data, EnvNonce }
```

- The content of the data field, which contains the encrypted data, would be the sole piece of information that would be interesting to an attack.
- Being unable to easily identify the recipient of a message makes sense as part of the dark communication approach. Sending the message to every node is the only way to have any chance of it getting to its intended destination without this vital piece of information.

Routing in the blind

- Whisper prevents traffic analysis by routing every communication it sends to every network node.
- Whisper functions similarly to the user datagram protocol (UDP) when it is in broadcast mode. It is hard to determine which receiver the sender is attempting to reach because each node gets a copy of the same message.
- A strong adversary may still be able to know if two nodes are interacting even though it is difficult to tell which recipient a message is directed to if they are able to control all except the two conversing nodes.

- In this case, the adversary will be able to determine that communication has happened between nodes A and B if node A delivers a message to node B.
- Such an attack can be defeated by introducing noise into the network by having well-behaved nodes sending junk messages encrypted with a random key into the network.

Probabilistic message filtering

- A node must be able to decode the message in order to determine whether a message is meant for an identity using its service.
- A node must utilise every key it has in its possession before it can decide if a piece of message is transmitted to its users since the envelope carries no information about the intended receiver. Decryption is a costly task! In most real applications, it is impossible to decode all incoming messages because to the fact that each node will get a copy of every message sent across the network.
- Each communication must be linked to a subject in order to avoid this issue. An identity registers the topics it's interested in by using its encryption key to create a message filter on a node.
- This efficient probabilistic filter, known as a bloom filter, can tell to a very high degree of certainty (false-positives are possible) if a piece of message belongs to a topic of interest.
- A node will only attempt decryption if a filter signals a possible match.

Quality of Service Assurance

- It is simple to assume that Whisper is vulnerable to denial of service (DoS) attacks since every Whisper message is sent to every node that can receive it.
- At least two methods can be used to initiate an attack :
  - (i) **Flood Attack :** Repeatedly send messages into the network
  - (ii) **Expiry Attack :** Make messages hang around for a long time by setting a long TTL (Time-to-Live) on the envelope

- The Whisper node and message filters prevent flood attacks by demanding proof-of-work (PoW) calculation from the sender and providing the result as the EnvNonce field in the message envelope. If the PoW is too low, a node could decline to accept a message. The node may not require the same level of PoW that message filters choose.
- A message rating system that computes a message rating by factoring in PoW, message size, and TTL prevents expiry attack.
- The rating's requirements are rather simple :
  - (i) Smaller messages have higher ratings
  - (ii) Messages with higher PoW have higher ratings
  - (iii) Messages with lower TTL have higher ratings
- A message's rating determines both its forwarding priority and the amount of time it will be kept in the system.
- Therefore, it is in the sender's best interests to ensure that the messaging rating is as high as is required to accomplish the intended goal. For instance, a node may delete up messages that it deems to be of low rating when its message pool is about to reach its memory limit. By using this grading system, a DoS attack's impact is reduced.

Barriers to Wider Adoption

- Despite how fantastic Whisper is, hardly as many people really use it.
- This is partially because of Whisper's design decisions and partly because there are other communications options available inside the Ethereum ecosystem.

Key Management

- Currently, Whisper relies on Whisper nodes preserving users' secret keys in order to decode communications addressed to them. Because of this manner of operation, it is impossible to employ "zero client" providers like Infura, one of the most well-known Ethereum infrastructure services, which operate under the tenet that wallet programmes like MetaMask should be in charge of key management.

- This restriction can be lessened by operating Whisper nodes concurrently with the wallet application on the user's device. For instance, a trading application built in Electron may use Whisper to implement order signalling and communicate with other Whisper nodes.
- The basis for alternative Whisper implementations that use protocols other than TCP and UDP, which ĐΞVp2p requires to communicate with standard Whisper nodes supported by Go Ethereum (Geth), or Parity, is being laid by some intriguing advances in this area, headed by Guillaume Ballet.

☞ Disabled by Default

- Whisper is currently an opt-in service. It must be enabled at the command line in order to be used. Whisper is therefore not active on the vast majority of Ethereum nodes. Whisper is still an experimental protocol, thus this deliberate choice to keep it off by default is a smart one.
- Whisper is still only a proof-of-concept protocol, but it has already been adopted in real-world settings. The largest production Whisper user at the present is arguably Status, an intriguing Ethereum client project that leverages Whisper to provide conversation.

Chapter Ends...

❑❑❑