International Research Journal in Global Engineering and Sciences. (IRJGES)
ISSN: 2456-172X | Vol. 1, Issue. 3, September - November, 2016 | Pages 90-96

DEPLOYING SMART CONTRACT IN ETHEREUM NETWORK AS DECENTRALIZED APPLICATION

M. Revathi[1], S.Jayanthi[2]

Assistant Professor, Department of CSE, Agni College of Technology, Chennai.

Abstract: This Project evolve around Ethereum, an open Blockchain platform which enables usage of Decentralized Application(DApp). We have created a DApp for a Car Rental Agency as a Smart contract and deployed it in the Ethereum test network. Every Online Application involves a centralized server for processing user request. Only a single server is responsible for managing data and processing user's request. Every user forms a distributed network with every other user. Every transaction is called as a block and all the blocks are maintained by all users. Solidity is the language used for contracts.

I. INTRODUCTION

Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology. No one controls or owns Ethereum – it is an open-source project built by many people around the world. Ethereum was designed to be adaptable and flexible. Blockchain technology is the technological basis of Bitcoin. A blockchain is a distributed computing architecture where every network node executes and records the same transactions, which are grouped into blocks. Only one block can be added at a time, and every block contains a mathematical proof that verifies that it follows in sequence from the previous block. In this way, the blockchain's "distributed database" is kept in consensus across the whole network. Individual user interactions with the ledger (transactions) are secured by strong cryptography.

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 node of the network runs the EVM and executes the same instructions. For this reason, Ethereum is sometimes described evocatively as a "world computer". Specifically, ethereum is suited for applications that automate direct interaction between peers or facilitate

coordinated group action across a network. All action on the Ethereum blockchain is set in motion by transactions fired. Every time a contract account receives a transaction, its code is executed as instructed by the input parameters sent as part of the transaction. The contract code is executed by the Ethereum Virtual Machine on each node participating in the network as part of their verification of new blocks.

The term "transaction" is used in Ethereum to refer to the signed data package that stores a message to be sent from an externally owned account to another account on the blockchain .Transactions contain the recipient of the message, a signature identifying the sender and proving their intention to send the message via the blockchain to the recipient, value field - The amount of wei to transfer from the sender to the recipient, an optional data field, which can contain the message sent to a contract, a start gas value, representing the maximum number of computational steps the transaction execution is allowed to take, a gas price value, representing the fee the sender is willing to pay for gas. One unit of gas corresponds to the execution of one atomic instruction, i.e., a computational step.

Smart contracts are code functions and can interact with other contracts, make decisions, store data, and send ether to others. Contracts are defined by their creators, but their execution, and by extension the services they offer, is provided by the ethereum network itself. Contracts can send "messages" to other contracts. Messages are virtual objects that are never serialized and exist only in the Ethereum execution environment. They can be conceived of as function calls. A message contains the sender of the message (implicit), the recipient of the message**,** value field - The amount of wei to transfer alongside the message to the contract address, an optional data field, that is the actual input data to the contract, a start gas value, which limits the maximum amount of gas the code execution triggered by the message can incur. The fact that contract executions are redundantly replicated across nodes, naturally makes them expensive, which generally creates an incentive not to use the blockchain for computation that can be done off chain.

When you are running a decentralized application (DApp), it interacts with the blockchain to read and modify its state, but DApp will typically only put the business logic and state that are crucial for consensus on the blockchain. When a contract is executed because of being triggered by a message or transaction, every instruction is executed on every node of the network. This has a cost: for every executed operation, there is a specified cost, expressed in

several gas units. Gas is the name for the execution fee that senders of transactions need to pay for every operation made on an Ethereum blockchain. The name gas is inspired by the view that this fee acts as crypto fuel, driving the motion of smart contracts. Gas is purchased for ether from the miners that execute the code. Contracts generally serve four purposes Maintain a data store representing something which is useful to either other contracts or to the outside world, Serve as a sort of externally-owned account with a more complicated access policy; this is called a "forwarding contract" and typically involves simply resending incoming messages to some desired destination only if certain conditions are met, Manage an ongoing contract or relationship between multiple users, Provide functions to other contracts, essentially serving as a software library.

Solidity is a contract-oriented, high-level language whose syntax is similar to that of JavaScript and it is designed to target the Ethereum Virtual Machine. Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

II .EXISTING SYSTEM

Centralized computing is computing done at a central location, using terminals that are attached to a central computer. The computer itself may control all the peripheral directly (if they are physically connected to the central computer), or they may be attached via a terminal. Alternatively, if the terminals have the capability, they may be able to connect to the central computer over the network. The terminals may be text terminals or thin clients for example.

This type of arrangement does have some disadvantages. The central computer performs the computing functions and controls the remote terminals. This type of system relies totally on the central computer. Should the central computer crash, the entire system will "go down" (i.e. will be unavailable).

Another disadvantage is that central computing relies heavily on the quality of administration and resources provided to its users. Should the central computer be inadequately supported by any means (e.g. size of home directories, problems regarding administration), then your usage will suffer greatly. The reverse situation, however, (i.e., a system supported better than your needs) is one of the key advantages to centralized computing.

The client–server model is a distributed application structure that partitions tasks or

workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

Example:

When a bank customer accesses online banking services with a web browser (the client), the client initiates a request to the bank's web server. The customer's login credentials may be stored in a database, and the web server accesses the database server as a client. An application server interprets the returned data by applying the bank's business logic, and provides the output to the web server. Finally, the web server returns the result to the client web browser for display.

In each step of this sequence of client–server message exchanges, a computer processes a request an returns data. This is the request-response messaging pattern. When all the requests are met, the sequence is complete and the web browser presents the data to the customer.

III.PROPOSED SYSTEM

A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

Decentralized autonomous organizations have been seen by some as difficult to describe. Nevertheless, the conceptual essence of a decentralized autonomous organization has been typified as the ability of blockchain technology to provide a secure digital ledger that tracks financial interactions across the internet, hardened against forgery by trusted time stamping and by dissemination of a distributed database. This approach eliminates the need to involve a

bilaterally accepted trusted third party in a financial transaction, thus simplifying the sequence. The costs of a blockchain enabled transaction and of making available the associated data may be substantially lessened by the elimination of both the trusted third party and of the need for repetitious recording of contract exchanges in different records: for example, the blockchain data could in principle, if regulatory structures permitted, replace public documents such as deeds and titles. In theory, a blockchain approach allows multiple cloud computing users to enter a loosely coupled peer-to-peer smart contract collaboration.

Blockchain is a distributed database that maintains a continuously growing list of ordered records called blocks. Each block contains a timestamp and a link to a previous block. Blockchains are "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way. The ledger itself can also be programmed to trigger transactions automatically.

A Decentralized Application (or 'DApp') is a piece of software consisting of a user interface (UI) and a decentralized backend; typically making use of a blockchain and smart contracts. Most of the projects listed on this page were built using Ethereum - a popular development platform for creating DApp.

IV.CONCLUSION

The Ethereum protocol was originally conceived as an upgraded version of a cryptocurrency, providing advanced features such as on-blockchain escrow, withdrawal limits, financial contracts, gambling markets and the like via a highly generalized programming language. The Ethereum protocol would not "support" any of the applications directly, but the existence of a Turing-complete programming language means that arbitrary contracts can theoretically be created for any transaction type or application. What is more interesting about Ethereum, however, is that the Ethereum protocol moves far beyond just currency. Protocols around decentralized file storage, decentralized computation and decentralized prediction markets, among dozens of other such concepts, have the potential to substantially increase the efficiency of the computational industry, and provide a massive boost to other peer-to-peer protocols by adding for the first time an economic layer. Finally, there is also a substantial array

of applications that have nothing to do with money at all.

The concept of an arbitrary state transition function as implemented by the Ethereum protocol provides for a platform with unique potential; rather than being a closed-ended, single-purpose protocol intended for a specific array of applications in data storage, gambling or finance, Ethereum is open-ended by design, and we believe that it is extremely well-suited to serving as a foundational layer for a very large number of both financial and non-financial protocols in the years to come.

References:

[1]. M. Carli, M. Farais, E. D. Gelasca, R. Tedesco, and A. Neri, "Quality assessment using data hiding on perceptually important areas," in Proc. IEEE Int. Conf. Image Processing, ICIP, Sep. 2005, pp. III-1200-3–III-1200-3.
[2]. A. Yilmaz and A. Aydin, "Error detection and concealment for video transmission using information hiding," Signal Processing: Image Communication, vol. 23, no. 4, pp. 298–312, Apr. 2008.
[3]. S. Kapotas and A. Skodras, "A new data hiding scheme for scene change detection in H.264 encoded video sequences," in Proc. IEEE Int. Conf. Multimedia Expo ICME, Jun. 2008, pp. 277–280.
[4]. K. Nakajima, K. Tanaka, T. Matsuoka, and Y. Nakajima, "Rewritable data embedding on MPEG coded data domain," in Proc. IEEE Int. Conf. Multimedia and Expo, ICME, Jul. 2005, pp. 682685.
[5]. Y. Li, H.-X. Chen, and Y. Zhao, "A new method of data hiding based on H.264 encoded video sequences," in Proc. IEEE Int. Conf. Signal Processing, ICSP, Oct. 2010, pp. 1833–1836.
[6]. D.-Y. Fang and L.-W.Chang, "Data hiding for digital video with phase of motion vector," in Proc. IEEE Int. Symp. Circuits Systems, ISCAS, Sep. 2006.
[7]. C. Xu, X. Ping, and T. Zhang, "Steganography in compressed video stream," in Proc. Int. Conf. Innovative Computing, Information and Control, ICICIC"06, 2006, vol. II, pp. 803–806.
[8]. K. Wong, K. Tanaka, K. Takagi, and Y.Nakajima, "Complete video quality-preserving data hiding," IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 10, Oct. 2009.
[9]. K. Solanki, U. Madhow, B. S. Manjunath, S. Chandrasekaran, and I. El-Khalil, ""Print and Scan" resilient data hiding in images," IEEE Trans. Inform. Forensics Security, vol. 1, no. 4, pp. 464–478, Dec.2006.
[10]. X.-P. Zhang, K. Li, and X. Wang, "A novel look-up table design method for data hiding with reduced distortion," IEEE Trans. Circuits Syst. Video Technol., vol. 8, no. 6, pp. 769–776, Jun. 2008.
[11] W. J. Lu, A. Varna, and M. Wu, "Secure video processing: Problems and challenges," in Proc. IEEE Int. Conf. Acoust.,Speech, Signal Processing,Prague, Czech Republic, May 2011, pp. 5856–5859.
[12] B. Zhao, W. D. Kou, and H. Li, "Effective watermarking scheme in the encrypted domain for buyer-seller watermarking protocol," Inf. Sci.,vol. 180, no. 23, pp. 4672–4684, 2010.

[13] P. J. Zheng and J. W. Huang, "Walsh-Hadamard transform in the homomorphic encrypted domain and its application in image watermarking,"in Proc. 14th Inf. Hiding Conf., Berkeley, CA,USA, 2012, pp. 115.
[14] W. Puech, M. Chaumont, and O. Strauss, "A reversible data hiding method for encrypted images," Proc. SPIE, vol. 6819,pp. 68191E-1–68191E-9, Jan. 2008.
[15]. X. P. Zhang, "Reversible data hiding in encrypted image," IEEE Signal Process. Lett., vol.18, no. 4, pp. 255–258, Apr. 2011.
[16]. W. Hong, T. S. Chen, and H. Y. Wu, "An improved reversible data hiding in encrypted images using side match," IEEE Signal Process.Lett., vol.19,no. 4, pp. 199–202, Apr. 2012.
[17] .X. P. Zhang, "Separable reversible data hiding in encrypted image," IEEE Trans. Inf. Forensics Security, vol. 7, no. 2,pp. 826–832, Apr. 2012.
[18] . K. D. Ma, W. M. Zhang, X. F. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," IEEE Trans. Inf. Forensics Security, vol. 8, no. 3, pp. 553–562, Mar. 2013.