A Decentralized Web Hosting System

A project submitted in partial fulfillment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY
in
Computer Science and Engineering by

Abhishek Tripathi (1709010003)
Divya Darshan (1709010032)
Aditya Nath (1709010007)

Under the Supervision of

Prof. Mr. Murari Kr Singh

IEC Group of Institutions, Greater Noida

to the

Faculty of Computer Science Department

DR APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

March 2021

CERTIFICATE

Certified that Abhishek Tripathi (1709010003) , Divya Darshan (1709010032), Aditya Nath(1709010007) has carried out the research work presented in this thesis entitled “A Decentralized Web Hosting System” for the award of/ Bachelor of Technology from Dr APJ Abdul Kalam Technical University, Lucknow under my supervision. The thesis embodies results of original work, and studies are carried out by the student himself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Signature

Prof. Mr. Murari Kr Singh

IEC Group of Institution, Greater Noida .

ABSTRACT

This paper proposes a decentralized solution for web hosting based on interplanetary file system (IPFS) and Ethereum blockchain. Particularly, we use Ethereum smart contracts to manage the IPFS network and the web hosting service. IPFS platform is used to store data and to host websites. All storage miner nodes on the IPFS network offer the pinning service to ensure that source codes of the websites and users' data are retained long-term. Moreover, these nodes also enable the interplanetary name space (IPNS) service for creating and updating mutable links to IPFS contents. TXT record is also used in the domain name system (DNS) to map domain names to IPNS addresses for hosted websites. For privacy-preserving data storage, websites need to be deployed an encryption algorithm. The proposed model that combines between the IPFS and blockchain networks to form a platform providing the decentralized web hosting service. Experiment illustrates building and hosting a web application on the IPFS network. Experimental results show that, compared to the traditional web hosting model, the hosted web application on the proposed platform ensures the confidentiality, integrity, and availability.

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the final year report of the B.Tech. We owe special debt of gratitude to our Guide **Prof. Mr. Murari Kr Singh**, Greater Noida for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his continuous effort that our endeavours have seen light of the day.

We also do not like to miss the opportunity to acknowledge the contribution of other teachers for their kind assistance and cooperation during the development of my project. Last but not the least, we acknowledge our friends for their contribution in the completion of project.

Abhishek Tripathi (1709010003).

Divya Darshan (1709010032).

Aditya Nath (1709010007).

TABLE OF CONTENT

TOPICS	PAGE NO.
Certificate	ii
Abstract	iii
Acknowledgement	iv

CHAPTERS

1	INTRODUCTION	6
2	LITERATURE SURVEY	10
3	SYSTEM REQUIREMENT SPECIFICATION	12
4	SYSTEM FEATURES	13
5	SYSTEM IMPLEMENTATION	17
6	SYSTEM CODING	20
7	RESULT AND ANALYSIS	25

CONCLUSION	26
FUTURE ENHANCEMENT	27
REFERENCE	28

INTRODUCTION

The internet as we know is reliant on centralized operators (servers). All the user data present on the internet is in the hands of a few which makes it vulnerable to cyber-attacks and hacks. It also makes easier for governments and corporates to track us and conduct surveillance.
It sounds creepy that how much they know about your behavior, your likes, and everything personal. Decentralized Internet, on the other hand, gives users a choice to enjoy the same services, decentralized and not creepy.

1.1 What is the Decentralized Internet?

Decentralized Internet or DWeb is like the internet that does not rely on servers to work. A decentralized web relies on a peer-to-peer network of interconnected devices that would host the internet, not a group of servers owned by big companies.
It promises a high level of privacy and control over the data that is available on the internet. On the decentralized version, it would be harder for the government and companies to block a website or display the ads on our browsers.

1.2 How Does It Work?

In the decentralized internet, each web page would be spread out across millions of nodes on different devices. Your computers and mobile phones would not only requests for the services but provide them. Moreover, it will also allow the information to be shared amongst devices in numerous ways rather than using servers for exchanging information. Nobody would be able to track your information, and thus the power would always remain in the hands of the users.

1.3 Existing system

WEB2.0(Centralized Web)

Web 2.0 is a term that was introduced in 2004 and refers to the second generation of the World Wide Web. The term "2.0" comes from the software industry, where new versions of software programs are labeled with an incremental version number. Like software, the new generation of the Web includes new features and functionality that was not available in the past. However, Web 2.0 does not refer to a specific version of the Web, but rather a series of technological improvements.

1.4 Problem Statement

Few large and physical servers are responsible for providing us with hosting and essential elements to use the internet. These servers keep our websites, emails, social media accounts, and all our sensitive details. The reliance of the internet on these servers open to major vulnerabilities that might get worse in the future.

A.Servers Can Go Down

A small malfunction in the servers could pose a severe impact on our internet. All our websites, emails, online bank accounts can stop working making us paralyzed to do anything.

B.Servers Can be Hacked

Cyber hacks have led millions of internet users to lose their money and private data over the past few years. Internet being dependent on a few servers make it more vulnerable to hacks and cyber-attacks.

C.Privacy Issues

in the lights of Facebook scandal and Cambridge Analytica, public concerns around privacy and spying have grown significantly. The personal data of the people aren't secure with centralized internet.

1.5 Proposed System

In A Decentralized Web Hosting System we create a Voting Web App which works on Decentralized web instead of Centralized web.
In this voting app we takes some initial candidates who stand for election as standard and the vote given to them is displayed in front of their name.

Advantages:-

- The vote given can never be changed.
- The web application created can never be hacked.
- The data does not remain centralized.
- Even if one node is down it does not affects the whole system.
- When accessing the dta is taken from the nearest node.

1.6 Objective

The main objectives of the proposed research work are as follows:

• Conducting a systematic summarization of various works pertaining to Decentralized Web Hosting to A Simplified Voting System Web App.
• To provide substantial basis for us to understand the impact and directions of future improvement of Voting System Web App using Decentralized Web Hosting.
• Short time lag. .
• Help voter to better discover their voting candidates.
• The given vote can be considered perfectly safe with no room of alteration.

1.5 Organization of report

This project report has been broadly divided into six chapters:

• **CHAPTER 1** gives purpose of system, problem statement and proposed work.

• **CHAPTER 2** discusses about literature survey.

• **CHAPTER 3** gives system requirement specification of the project.

• **CHAPTER 4** gives the SYSTEM FEATURES of the project.

• **CHAPTER 5** gives the system implementation details of the project.

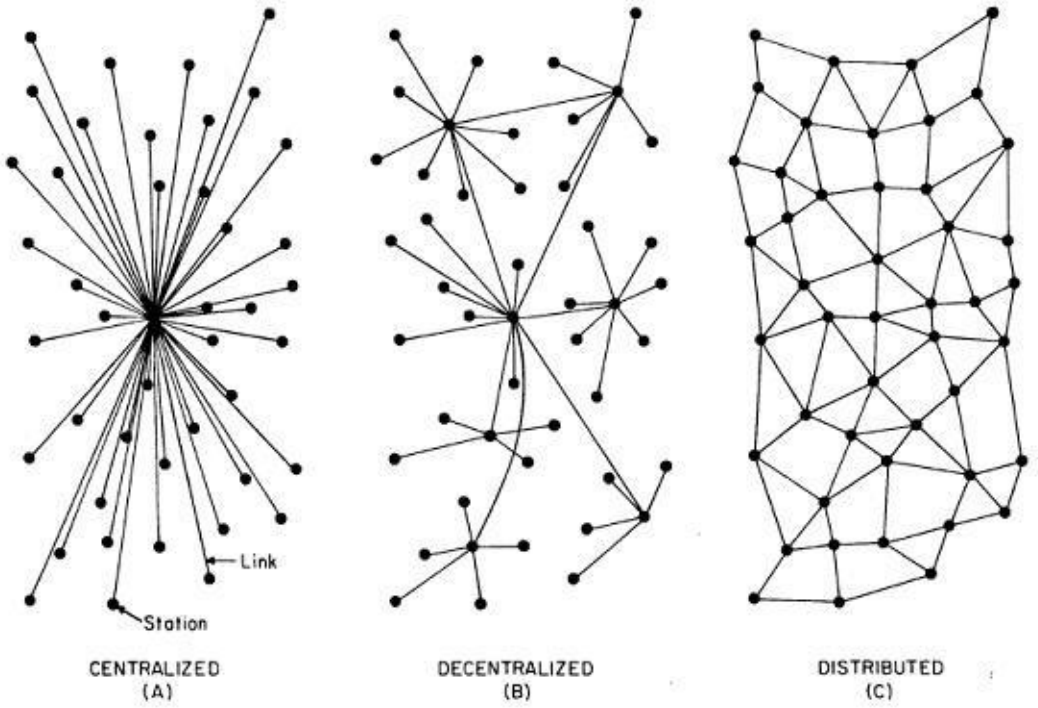
• **CHAPTER 6** gives the SYSTEM CODING

• **CHAPTER 7** gives the results of the project

CHAPTER 2

LITERATURE SURVEY

Decentralized vs Centralized vs Distributed. What's the difference?



Centralized

Centralized systems directly control the operation of the individual units and flow of information from a single center. All individuals are directly dependent on the central power to send and receive information, and to be commanded.

- single server
- easy to publish
- difficult to scale
- single point of failure

Examples (Google, Facebook, Amazon, etc.)

A Decentralized Web Hosting System

Distributed

Distributed systems spread computation across multiple nodes instead of just one. Google for example has adopted a distributed architecture internally to speed up computing and data latency. This means that a system can be both centralized and distributed.
Examples (Google, Facebook, Amazon, etc.)

Decentralized

Decentralized systems are ones where no node is telling any other node what to do. Bitcoin is both distributed because its timestamped public ledger, the blockchain, resides on multiple computer and decentralized because if one node goes down, the network is still able to operate.

- Multiple Servers
- Demand and Failures better handled

Examples (Bitcoin, Ethereum, Steemit)

So does a profitable decentralized app look like?

	Web 2.0	Web 3.0 (dApps)	Status
Scalable computation	Amazon EC2	Ethereum, Truebit	In progress
File storage	Amazon S3	IPFS/Filecoin, Storj	In progress
External data	3rd party APIs	Oracles (Augur)	In progress
Monetization	Ads, selling goods	Token model	Ready
Payments	Credit Cards, Paypal	Ethereum, Bitcoin, state channels, 0x	Ready

Chapter 3

SYSTEM REQUIREMENT SPECIFICATION

Scalable Computation	Ethereum
File Storage	IPFS
External Data	Oracles(Augur)
Monetization	Token Model
Payements	Bitcoin,Ethereum

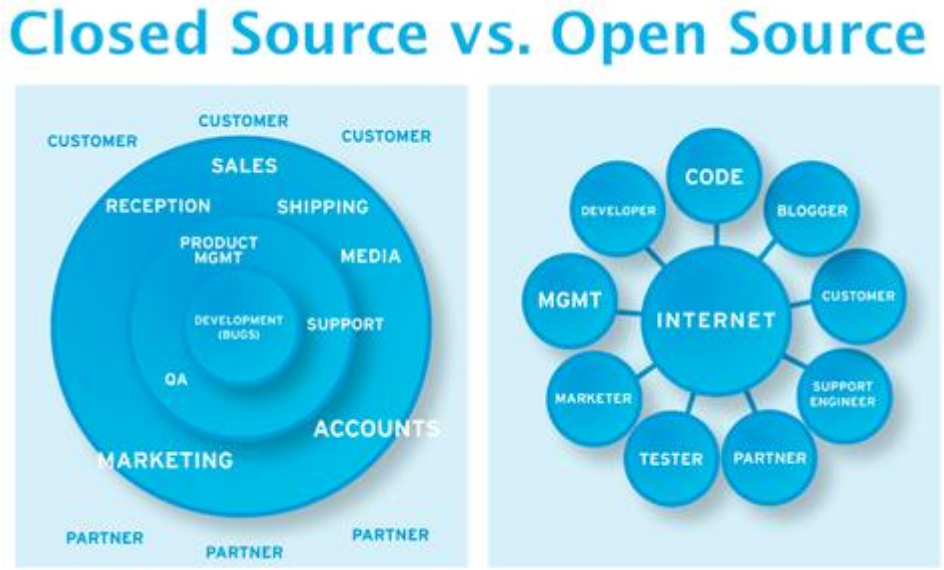
3.1 Programing languages used

- HTML
- Nodejs
- Solidity
- Javascript

CHAPTER 4

SYSTEM FEATURES

Feature
4.1 - its open source



If we delve into the traditional business models, all of them require the product or service for sale to be better than that of the competitor. Open sourcing your product would mean that any competitor could take all of your work, white label it, and sell it as their own.

Having the app be open source allowed the network the transparency it needed to improve itself with developer contributions and grow trust amongst its users to give its coins real world value. Open sourcing your dapp will gain the trust of potential users. Anyone can fork your dapp, but they can't fork your development team. Users want to get behind the people best suited to maintain the dapp, and those tend to be the original developers.

4.2: Use of Cryptocurrency

Traditional modes of monetization

- transaction fees
- advertising revenues
- referral commissions
- access rights to user data
- subscription services.

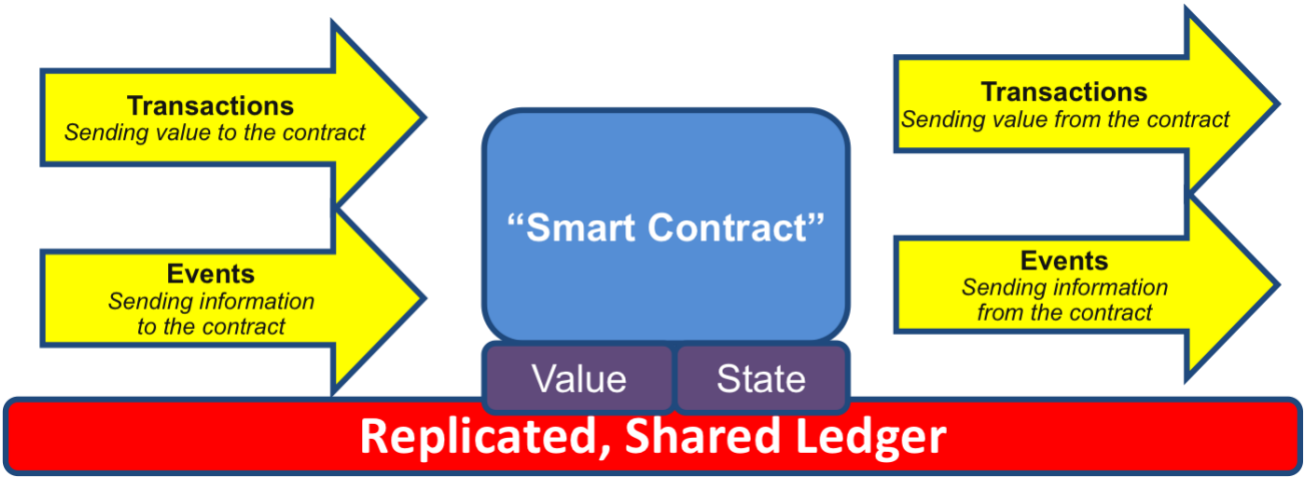
Cryptocurrency way

- Allocate scarce resources in the network using a scarce token
- Users need this appcoin to use the network.
- Owners of scarce resources get paid in appcoins.

Example 1 (Bitcoin) - The owners (miners) of the scarce resources (computing power) get paid with transaction fees directly from the users so that they can use the service. Because the network grew to include more users and there were a fixed amount of coins from the output, the values of the coins grew as well.

Example 2 (FileCoin) - Miners of the scarce resource (storage space) get paid in filecoin. More people using the service to store files, more the value of filecoin rises.
We can apply this model to any kind of dapps. Scarce resources could be storage space, trades, images, videos, texts, ads, etc.

4.3: Decentralized Consensus

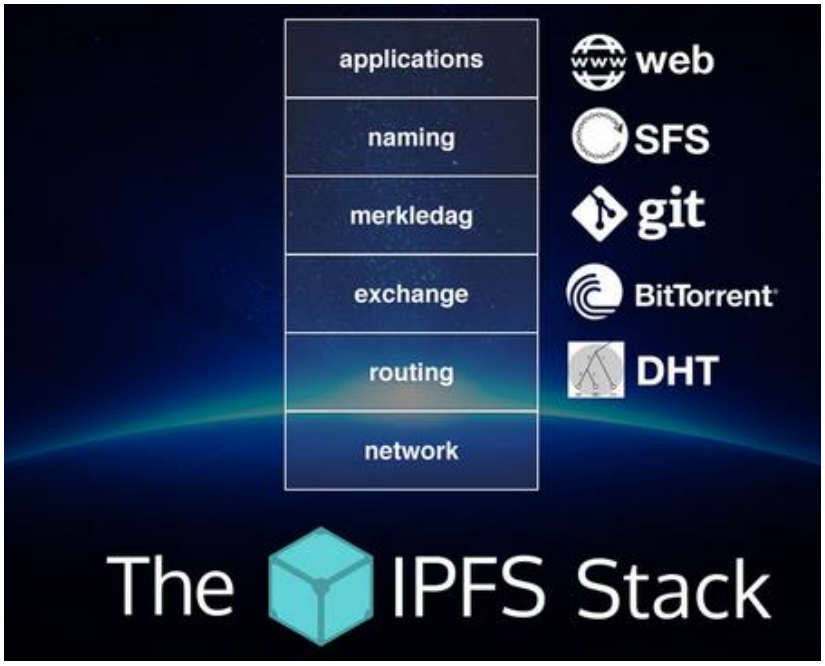


- The Use of a proof algorithm to maintain consensus (like proof of work)
- Distributed Hash Table to store data
- Blockchain for app level constructs (usernames, status updates, high scores, timestamps)
- DHT + Blockchain since blockchain solves the major security issue of DHTs (not forcing nodes to trust each other on the validity of data
- Smart Contracts (cryptoeconomically secured code)

```
if (user.sendsMoney(customerID))
{
runContract();
}

func runContract()
{
 println('hello world');
}
```

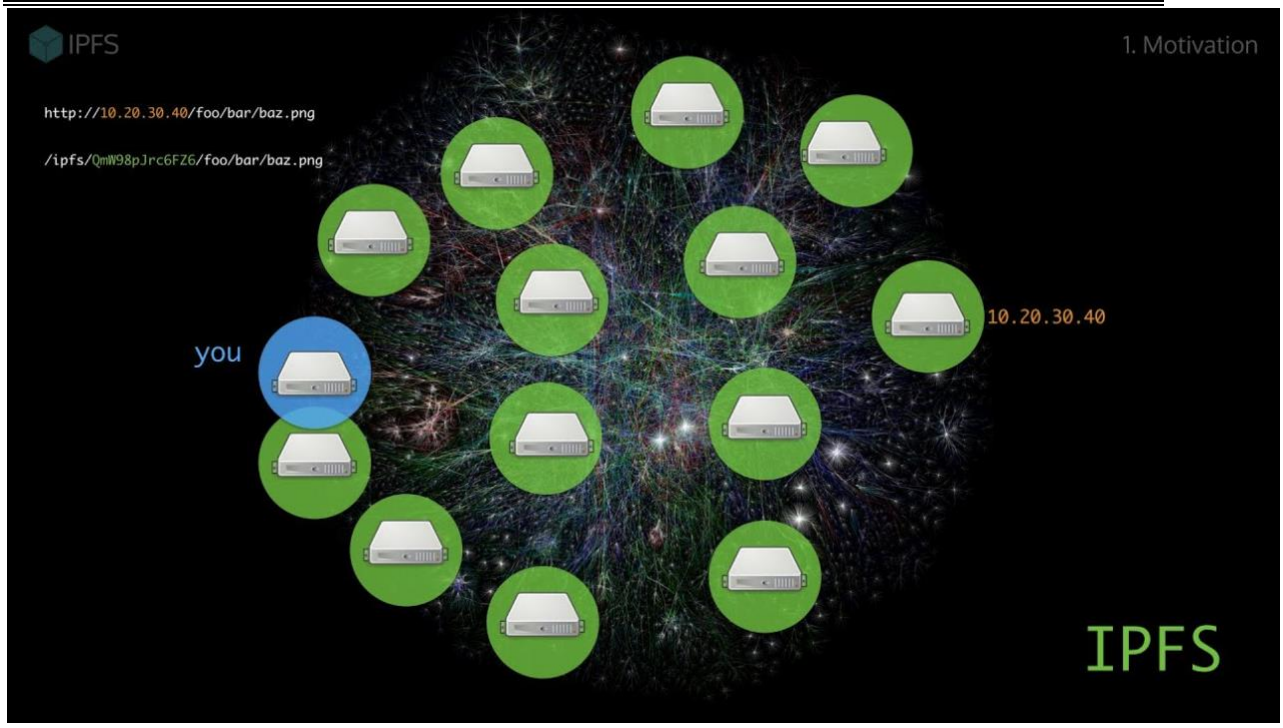
4.4: No central point of failure

Use IPFS

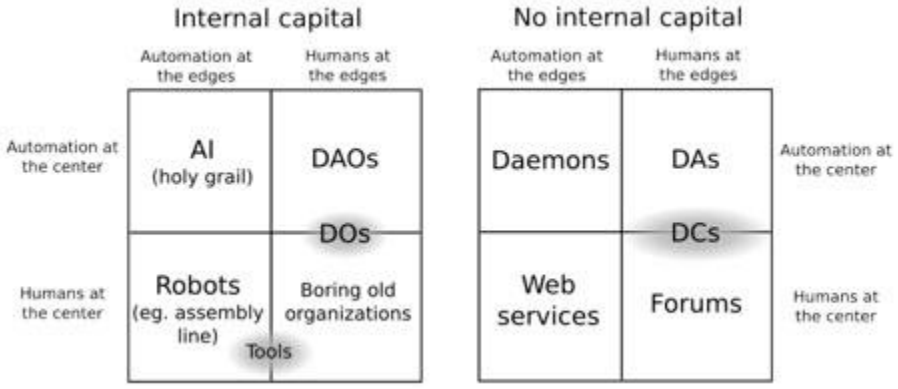


IPFS	HTTP
Websites/apps have no centralized origin server	Based on a centralized origin server
Utilizes content-addressing	Location based addressing
DDoS attacks seem impossible	Susceptible to DDoS attacks

A Decentralized Web Hosting System



These technologies are ways of building more autonomous software
Where value generation and capital generation are more closely aligned

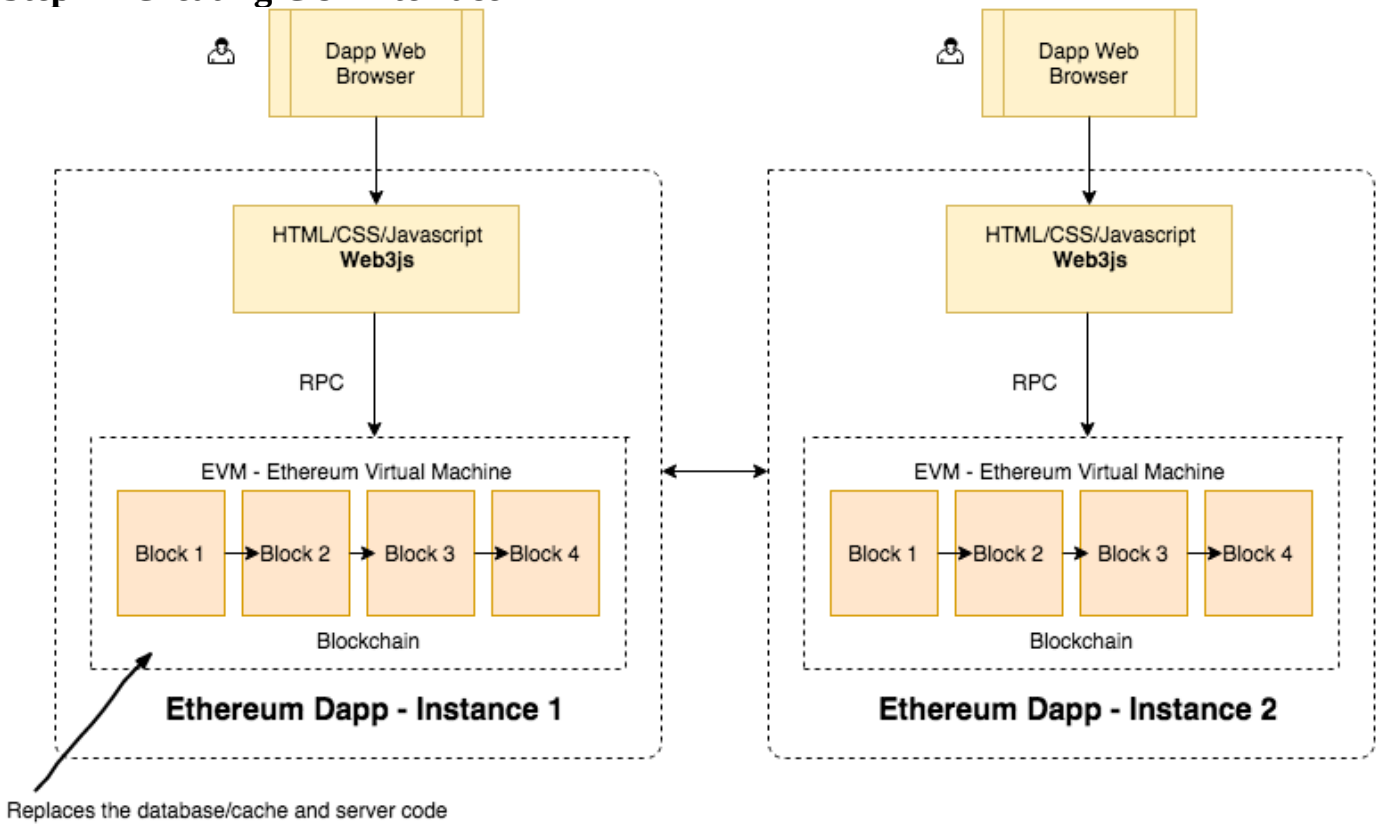


CHAPTER 5

SYSTEM IMPLEMENTATION

Alright lets get to our voting dapp demo...

Step 1 - Setting up Environment
Step 2 - Creating Voting Smart Contract
Step 3 - Interacting with the Contract via the Nodejs Console
Step 4 - Creating GUI interface



Steps

5.1 - Setting up Environment

Instead of developing the app against the live Ethereum blockchain, we will use an in-memory blockchain (think of it as a blockchain simulator) called testrpc.
npm install ethereumjs-testrpc web3
testrpc creates 10 test accounts to play with automatically. These accounts come preloaded with 100 (fake) ethers.

5.2 - Creating Voting Smart Contract

- We'll use Ethereum's solidity programming language to write our contract
- Our contract (think of contract as a class) is called Voting with a constructor which initializes an array of candidates
- 2 functions, one to return the total votes a candidate has received & another to increment vote count for a candidate.
- Deployed contracts are immutable. If any changes, we just make a new one.

Let's install the solidity compiler via npm
npm install solc

After writing our smart contract, we'll use Web3js to deploy our app and interact with it

```
siraj:~/hello_world_voting$ node
> Web3 = require('web3')
> web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"
));
```

Then ensure Web3js is initalized and can query all accounts on the blockchain

```
> web3.eth.accounts
```

Lastly, compile the contract by loading the code from Voting.sol in to a string variable and compiling it

```
> code = fs.readFileSync('Voting.sol').toString()
> solc = require('solc')
> compiledCode = solc.compile(code)
```

Deploy the contract!

- dCode.contracts[':Voting'].bytecode: bytecode which will be deployed to the blockchain.
- compiledCode.contracts[':Voting'].interface: interface of the contract (called abi) which tells the contract user what methods are available in the contract.

```
> abiDefinition = JSON.parse(compiledCode.contracts[':Voting'].interface)
> VotingContract = web3.eth.contract(abiDefinition)
> byteCode = compiledCode.contracts[':Voting'].bytecode
```

```
> deployedContract = VotingContract.new(['Rama','Nick','Jose'],{data: byteCo
de, from: web3.eth.accounts[0], gas: 4700000})
> deployedContract.address
> contractInstance = VotingContract.at(deployedContract.address)
```

deployedContract.address. When you have to interact with your contract, you need this deployed address and abi definition we talked about earlier.

5.3 - Interacting with the Contract via the Nodejs Console

```
> contractInstance.totalVotesFor.call('Rama')
{ [String: '0'] s: 1, e: 0, c: [ 0 ] }
> contractInstance.voteForCandidate('Rama', {from: web3.eth.accounts[0]})
'0xdedc7ae544c3dde74ab5a0b07422c5a51b5240603d31074f5b75c0ebc786bf5
3'
> contractInstance.voteForCandidate('Rama', {from: web3.eth.accounts[0]})
'0x02c054d238038d68b65d55770fabfca592a5cf6590229ab91bbe7cd72da46de
9'
> contractInstance.voteForCandidate('Rama', {from: web3.eth.accounts[0]})
'0x3da069a09577514f2baaa11bc3015a16edf26aad28dffbcd126bde2e71f2b76f'
> contractInstance.totalVotesFor.call('Rama').toLocaleString()
'3'
```

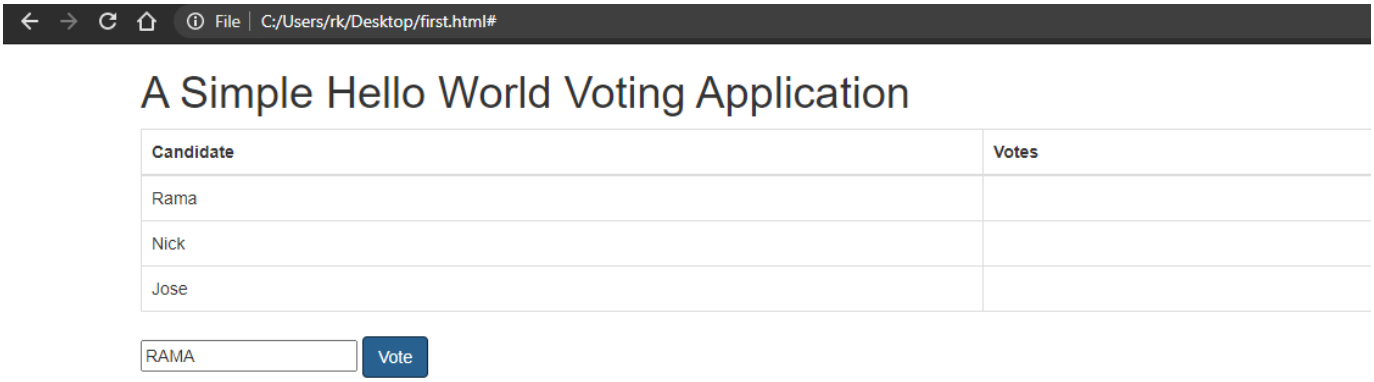
5.4 - Creating GUI interface

Let's use our HTML + JS client side skills w00t!

CHAPTER 6

SYSTEM CODING

6.1 Code for html file (displayed to the client)



Index.html

```
<!DOCTYP
E html>
                <html>
                <head>
                  <title>Hello World DApp</title>
                  <link
                href='https://fonts.googleapis.com/css?family=Open+Sans:400,700'
                rel='stylesheet' type='text/css'>
                  <link
                href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.m
                in.css' rel='stylesheet' type='text/css'>
                </head>
                <body class="container">
                  <h1>A Simple Hello World Voting Application</h1>
                  <div class="table-responsive">
                    <table class="table table-bordered">
                      <thead>
                        <tr>
                          <th>Candidate</th>
                          <th>Votes</th>
```

```
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Rama</td>
        <td id="candidate-1"></td>
      </tr>
      <tr>
        <td>Nick</td>
        <td id="candidate-2"></td>
      </tr>
      <tr>
        <td>Jose</td>
        <td id="candidate-3"></td>
      </tr>
    </tbody>
  </table>
 </div>
 <input type="text" id="candidate" />
 <a href="#" onclick="voteForCandidate()" class="btn btn-
primary">Vote</a>
</body>
<script
src="https://cdn.rawgit.com/ethereum/web3.js/develop/dist/web3.js">
</script>
<script src="https://code.jquery.com/jquery-
3.1.1.slim.min.js"></script>
<script src="./index.js"></script>
</html>
```

6.2 Code for creating smart contract in Ethereum blockchain using Solidity

Voting.sol

```
pragma solidity ^0.4.11;
                    // We have to specify what version of compiler this code
                    will compile with

                    contract Voting {
```

```
  /* mapping field below is equivalent to an associative
array or hash.
  The key of the mapping is candidate name stored as type
bytes32 and value is
  an unsigned integer to store the vote count
  */

  mapping (bytes32 => uint8) public votesReceived;

  /* Solidity doesn't let you pass in an array of strings in
the constructor (yet).
  We will use an array of bytes32 instead to store the list of
candidates
  */

  bytes32[] public candidateList;

  /* This is the constructor which will be called once when
you
  deploy the contract to the blockchain. When we deploy
the contract,
  we will pass an array of candidates who will be
contesting in the election
  */
  function Voting(bytes32[] candidateNames) {
    candidateList = candidateNames;
  }

  // This function returns the total votes a candidate has
received so far
  function totalVotesFor(bytes32 candidate) returns (uint8)
{
    if (validCandidate(candidate) == false) throw;
    return votesReceived[candidate];
  }

  // This function increments the vote count for the
specified candidate. This
  // is equivalent to casting a vote
  function voteForCandidate(bytes32 candidate) {
    if (validCandidate(candidate) == false) throw;
    votesReceived[candidate] += 1;
  }
```

```
  function validCandidate(bytes32 candidate) returns
(bool) {
    for(uint i = 0; i < candidateList.length; i++) {
      if (candidateList[i] == candidate) {
        return true;
      }
    }
    return false;
  }
}
```

6.3 Code for Java Script (It is used to connect the client html file to the ethereum smart contract)

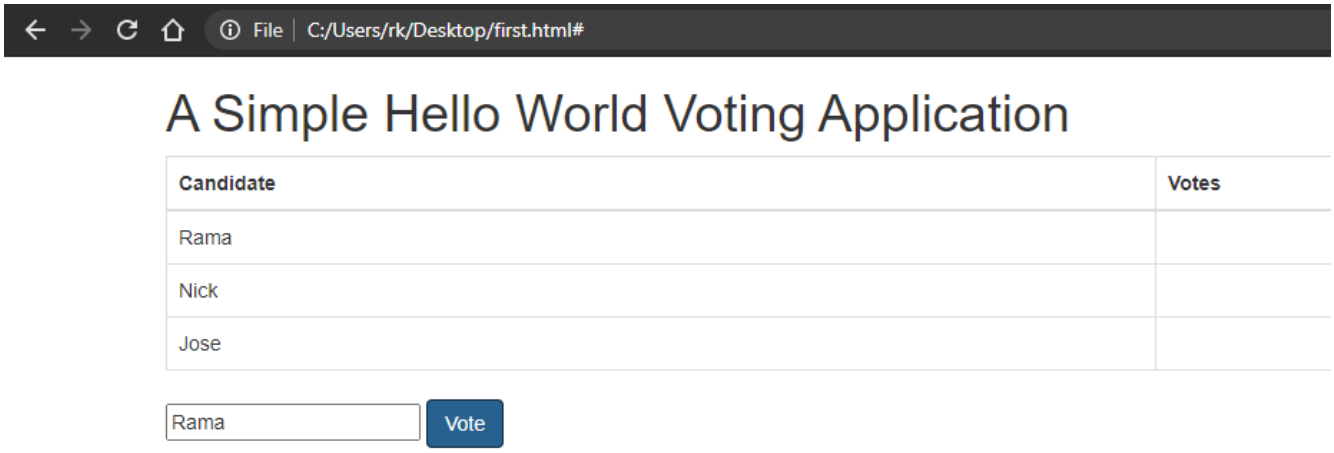
Index.js

```
we
b3
=
ne
w
We
b3(
ne
w
We
b3.
pro
vid
ers.
Htt
pPr
ovi
der(
"htt
p://l
oca
lho
st:8
545
"));
        abi = JSON.parse('[{"constant":false,"inputs":[{"name":"candidate","type":"bytes32"}],
```

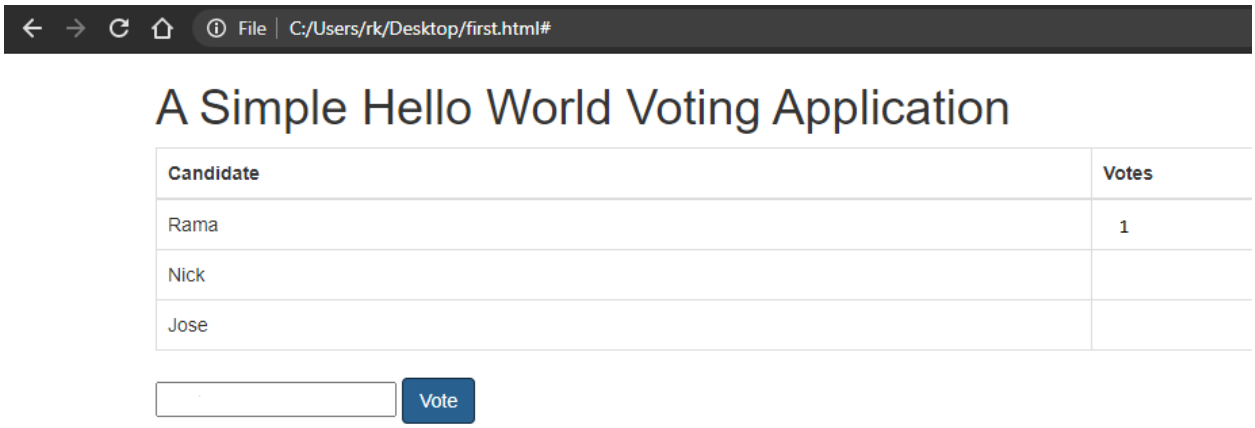
```
"name":"totalVotesFor","outputs":[{"name":"","type":"uint8"}],
"payable":false,"type":"function"},
{"constant":false,"inputs":[{"name":"candidate","type":"bytes32"}],
"name":"validCandidate",
"outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},
{"constant":true,
"inputs":[{"name":"","type":"bytes32"}],"name":"votesReceived",
"outputs":[{"name":"","type":"uint8"}],"payable":false,"type":"function"},
{"constant":true,"inputs":[{"name":"x","type":"bytes32"}],
"name":"bytes32ToString","outputs":[{"name":"","type":"string"}],"payable":false,
"type":"function"},{"constant":true,"inputs":[{"name":"","type":"uint256"}],
"name":"candidateList","outputs":[{"name":"","type":"bytes32"}],
"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"candidate",
"type":"bytes32"}],"name":"voteForCandidate","outputs":[],
"payable":false,"type":"function"},{"constant":true,"inputs":[],
"name":"contractOwner","outputs":[{"name":"","type":"address"}],
"payable":false,"type":"function"},{"inputs":[{"name":"candidateNames"
,"type":"bytes32[]"}],"payable":false,"type":"constructor"}]')
VotingContract = web3.eth.contract(abi);
// In your nodejs console, execute contractInstance.address to get
the address at which the contract is deployed and change the line
 below to use your deployed address
contractInstance = VotingContract.at('0x2a9c1d265d06d47e8f7b00ffa987c9185aecf672');
candidates = {"Rama": "candidate-1", "Nick": "candidate-2", "Jose": "candidate-3"};

function voteForCandidate() {
  candidateName = $("#candidate").val();
  contractInstance.voteForCandidate(candidateName,
 {from: web3.eth.accounts[0]}, function() {
    let div_id = candidates[candidateName];
    $("#" + div_id).
html(contractInstance.totalVotesFor.call(candidateName).toString());
  });
}

$(document).ready(function() {
  candidateNames = Object.keys(candidates);
  for (var i = 0; i < candidateNames.length; i++) {
    let name = candidateNames[i];
    let val = contractInstance.totalVotesFor.call(name).toString();
    $("#" + candidates[name]).html(val);
  }
});
```

CHAPTER 7

RESULT AND ANALYSIS



7.1 After Vote



CONCLUSION

In this Project, we have tried to implement the Decentralized web hosting concept on a voting web app. For the implementation of the Decentralized web-hosting system Ethereum computation is done using Solidity Language. For Client to access the voting web app HTML language is used. Furthermore, to connect the smart contract generated using Solidity language and HTML page JavaScript is used. Because of these processes the app created can not hacked or would crash even if a particular node stops working. Which make the developed web app more reliable compared to centralized web hosting.

FUTURE ENHANCEMENT

In this Project, we have tried to implement the Decentralized web hosting concept on a voting web app. For the implementation of the Decentralized web-hosting system Ethereum computation is done using Solidity Language. For Client to access the voting web app HTML language is used. Furthermore, to connect the smart contract generated using Solidity language and HTML page JavaScript is used. Because of these processes the app created can not hacked or would crash even if a particular node stops working. Which make the developed web app more reliable compared to centralized web hosting. Which makes the future scope of this project very expected.

REFERENCES

- https://towardsdatascience.com/decentralizing-your-website-f5bca765f9ed
- https://peergos.org/posts/p2p-web-hosting
- https://github.com/gdamdam/awesome-decentralized-web
- https://www.colocationamerica.com/blog/is-decentralized-web-hosting-a-fad
- https://www.cse.wustl.edu/~jain/cse570-19/ftp/decentrl/index.html