

ONLINE VOTING SYSTEM USING SOCKET PROGRAMMING

*Report submitted to SASTRA Deemed to be University
as the requirement of the course*

CSE302: COMPUTER NETWORKS

Submitted by:

TARANI SRE S G
(Reg No: 124003337, B. Tech.CSE)

DECEMBER 2022



SCHOOL OF COMPUTING
THANJAVUR, TAMIL NADU, INDIA – 613 401



SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

Bonafide Certificate

This is to certify that the report titled “**Online Voting System Using Java Programming**” is submitted as a requirement for the course, **CSE302 : COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Tarani Sre S G (Reg.No : 124003334,III-B.Tech-CSE)** during the academic year 2022-23, in the School of Computing.

Project Based Work Viva voce held on 13th of December.

Examiner 1

Examiner 2

Table of Content

Bonafide Certificate	2
Acknowledgment	5
CHAPTER 1: About the Concepts	6
1.1 Introduction	6
1.2 Problem Statement	6
1.3 Significance of the Study	6
The main purposes of Online Voting System include:	6
Crucial points that this online voting system emphasizes on are listed below:	6
1.4 Problems with existing Voting System	7
The problem of the existing manual system of voting include among others the following:	7
1.5 User requirements for the Proposed System	7
1.6 What is BlockChain?	8
1.7 How the BlockChain Technology Work?	8
1.8 Protocol used: TCP	9
What is Transmission Control Protocol (TCP)?	9
How Transmission Control Protocol works	9
CHAPTER 2: About the Code	10
2.1 Flowchart	10
2.2 Uses of Methods used	11
Main	11
Vars	11
Block	11
ClientManager	11
ServerManager	11
NetworkManager	11
MessageStruct	11
ServerManager	12
2.3 How to Run	12
2.4 Source Code	13
Main.java	13
Block.java	18

Vars.java.....	20
ClientManager.java.....	20
ServerManager.java.....	26
MessageStruct.java.....	30
NetworkManager.java.....	30
ServerHandler.java.....	32
CHAPTER 3: The Output	33
3.1Snapshots.....	33
Casting Votes:.....	33
Displaying and Counting votes:.....	35
Votes being Invalid:.....	36
3.2 Shortcomings	37
3.3 Conclusion.....	38
3.4 Future plans	38
3.5 References	38

Acknowledgment

I would want to convey my heartfelt gratitude for Prof. Kannan Balasubramanian, my mentor, for his invaluable assistance in completing my project. I would also like to thank all my other supporting personnel who helped me by supplying the equipment that are essential and vital, without which I would not have been able to perform efficiently on this project.

I would also like to thank SASTRA University for accepting my project in my desired expertise. I am also grateful for my friends and family for their support and encouragement as I worked on this assignment

CHAPTER 1: About the Concepts

Introduction

Internet voting system have gained popularity and have been used for government elections in many countries. Generally, voting system can involve transmission of ballots and votes via private computer networks or the Internet. Electronic voting technology can speed up the counting of ballots and can provide improved accessibility for disabled voters. This online voting system is highly secure and very simple and reliable. It creates and manages voting and election details as all users must login with username and password then click on their interested candidate to register their vote. By applying high security, it will reduce false votes.

Problem Statement

This Project is meant to be a Proof of Concept (POC) model for Online Voting System using Blockchaining in Java Programming. The server in the model broadcasts the encrypted block of votes to all nodes in the blockchain.

Significance of the Study

The main purposes of Online Voting System include:

- Provision of improved voting services to the voters through fast timely and convenient voting.
- Check to ensure that members who are registered are the only ones to vote. Cases of dead people voting or also minimized.
- Online voting system will require being very precise or cost cutting to produce an effective election management system.

Crucial points that this online voting system emphasizes on are listed below:

- require less number of staff during election.
- This system is a lot easier to independently moderate the elections and subsequently reinforce its transparency and fairness.
- Less capital, less effort, and less labor intensive, as the primary cost and effort will focus primarily on creating, managing, and running a secure online portal.

- Increase number of voters as individual will find it easier and more convenient to vote especially those abroad.

Problems with existing Voting System

The problem of the existing manual system of voting include among others the following:

1. **Expensive and Time consuming:** The process of collecting data and entering this data into the database takes too much time and is expensive to conduct, for example, time and money is spent in printing data capture forms, in preparing registration stations together with human resources, and there after advertising the days set for registration process including sensitizing voters on the need for registration, as well as time spent on entering this data to the database.
2. **Too much paperwork:** The process involves too much paperwork and paper storage, which is difficult as papers become bulky with the population size.
3. **Errors during data entry:** Errors are part of all human beings; it is very unlikely for humans to be 100 percent efficient in data entry.
4. **Loss of registration forms:** Some times, registration forms get lost after being filled in with voter's details, in most cases these are difficult to follow-up and therefore many remain unregistered even though they are voting age nationals and interested in exercising their right to vote.
5. **Short time provided to view the voter register:** This is a very big problem since not all people have free time during the given short period of time to check and update the voter register.

Above all, a number of voters end up being locked out from voting. Hence there is great desire to reduce official procedure in the current voter registration process if the general electoral process is to improve.

User requirements for the Proposed System

The OVS should:

- Be able to display all the registered voters in the database to the SYSTEM ADMIN(s) as per their access rights and privileges.
- Have a user-friendly interface and user guides understandable by people of average computer skills.

- Be robust enough so that users do not corrupt in the event of voting.
- Be able to handle multiple users at the same time with the same efficiency, this will cater for the large and ever-growing population of the voters.

What is Blockchain?

In a few words, a block chain is a digital ever-growing list of data records. Such a list is comprised of many blocks of data, which are organized in chronological order and are linked and secured by cryptographic proofs. Blockchain being completely a new system does have a unique way to offer decentralization.

Blockchain will store any kind of data exchange on the platform. So, it's like a ledger system, where every data exchange has a spot in the log. More so, the data exchanges in the system are called transactions. Once the transaction is verified, it gets a place in the ledger system as a block. Once it gets on the ledger, no one can delete or alter it in any way. In reality, blockchain uses a peer-to-peer distributed network, which will ensure the decentralized nature of the technology.

Every device that connects to the network is considered a node. Also, to understand "how does blockchain work," you need to understand the concept of "key." This is the basis of technology. Furthermore, the keys offer security on the network. For this, a user on the network will generate key pairs known as private and public keys. Once you start using the keys, you end up with a unique credential that no one can get access to. Anyhow, you'll have to store the private key in a secure place because you'll use this key to sign or perform any action on the network. On the other hand, other users will use your public key to find you on the system.

How the Blockchain Technology Work?

Firstly, a user or a node will initiate a transaction signing it with its private key. Basically, the private key will generate a unique digital signature and make sure that no one can alter it. In reality, if anyone tries to modify the transaction information, the digital signature will change drastically, and no one will be able to verify it. Therefore, it will be dismissed.

After that, the transaction will get broadcasted to the verifying nodes. Basically, here, the blockchain platform can use different methods to verify whether the transaction is valid or not. These methods or algorithms are called consensus algorithm.

Anyhow, once the nodes verify that the transaction is authentic, it will get a place in the ledger. Also, it will contain a timestamp and a unique ID to secure it further from any alteration.

The block will then link up to the previous block, and then a new block will form a link with this block and so on. And this way, it creates a chain of blocks, thus the name blockchain.

Protocol used: TCP

What is Transmission Control Protocol (TCP)?

Transmission Control Protocol (TCP) is a standard that defines how to establish and maintain a network conversation by which applications can exchange data.

TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules that define the internet.

How Transmission Control Protocol works

TCP is a connection-oriented protocol, which means a connection is established and maintained until the applications at each end have finished exchanging messages.

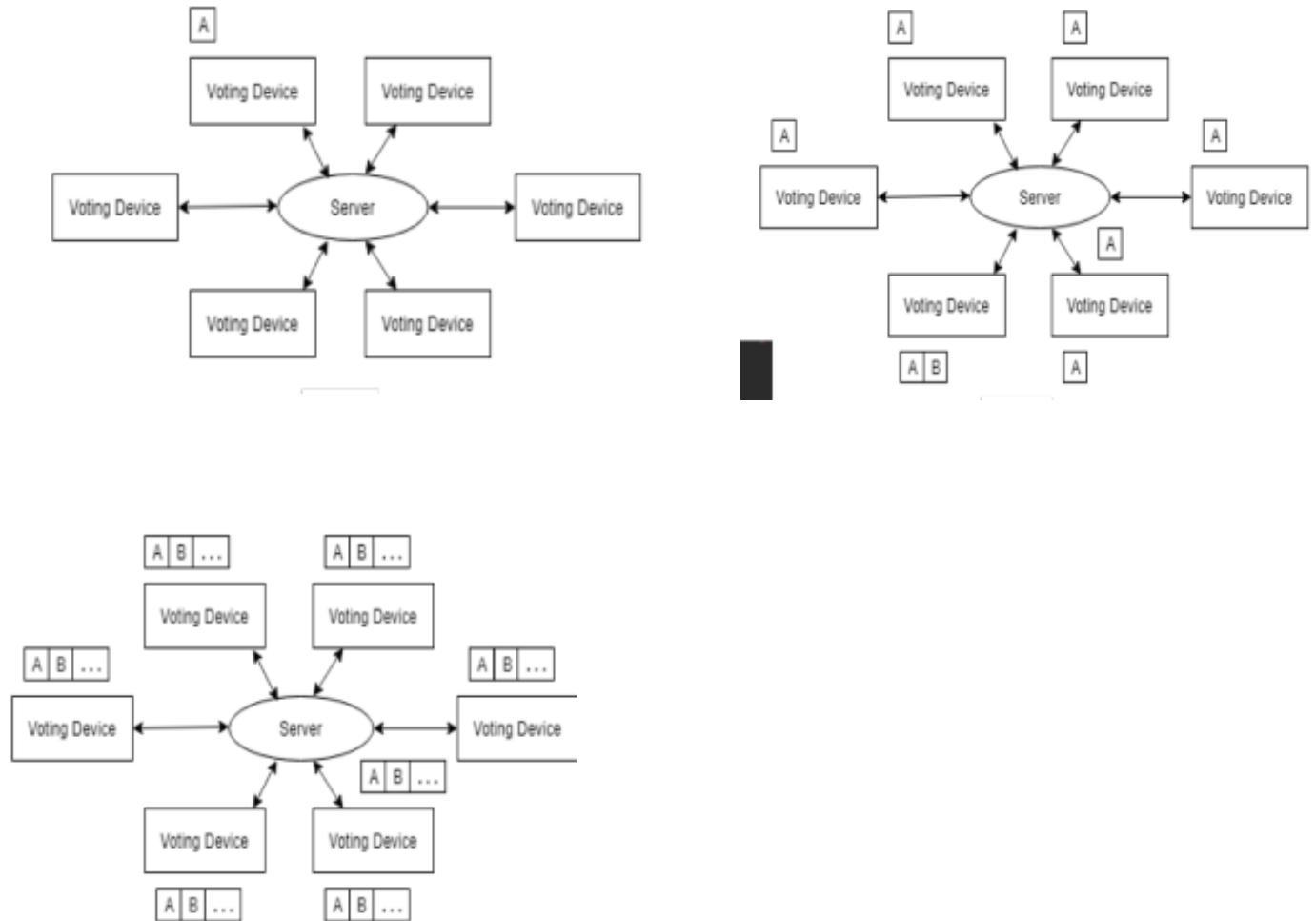
TCP performs the following actions:

- determines how to break application data into packets that networks can deliver.
- Sends packets to, and accepts packets from, the network layer.
- Manages flow control. Handles retransmission of dropped or garbled packets, as it's meant to provide error-free data transmission.
- Acknowledges all packets that arrive.

In the Open Systems Interconnection (OSI) communication model, TCP covers parts of Layer 4, the transport layer, and parts of Layer 5, the session layer.

CHAPTER 2: About the Code

Flowchart



Uses of Methods used

Main

Main class of a tiny framework to simulate voting using Blockchain via P2P network. A bi-directional migration between server and client is supported using JAVA Serialization/Reflection and Socket. Detailed system design, user case and limitations are elaborated in report.

Vars

Used only for directing the Data file blocks.

Block

Initializing and securing variables.

ClientManager

Responsible for all the network communications in client side.

In a new thread, it will run a loop receiving messages sent from server and dispatch it to the main thread to handle.

In the main thread, it provides a message handler handling all the incoming messages. Also, it has interfaces serving ProcessManager.

ServerManager

Responsible for all the network communications in server side.

In a new thread, it will run a loop accepting all the incoming clients and create a new instance of Server Handler in a new thread reading incoming message from connected clients.

In the main thread, it provides a message handler handling all the incoming messages. Also, it has interfaces serving Cluster Manager.

NetworkManager

Base class of Server Manager and Client Manager to provide network operations.

MessageStruct

A structure for communicating between server and client. Two fields indicate the message type(_code) and **No table of figures entries found**. Message types and description:

Type	description	direction
0	server broadcasts block	server -> client
1	client sends block	client -> server
2	server sends client id	server -> client

ServerManager

Server-side class to prepare and wait for messages from a client specified by _socket

How to Run

1. Open the file 'BC' meaning Block chain in the folders of your computer. Locate where the bc jar file is stored.
2. Type 'cmd' after selecting where the location of the folder on top of the screen is . Type "java -jar BC.jar" on the command prompt and give enter.
3. The Main menu is active now and you can just give any number between 1 to 3 and it will show more description that the server is connected to 6777th port.
4. As the instruction is given, type "Server" or "S".
5. Now, open another cmd prompt from the same location and enter "java -jar BC.jar" to get to the main menu.
6. Type any options you want, Cast, Count or View votes respectively 1,2 and 3.
7. It will be showing if the Respective terminal need to be a server or a client. "Client" should be entered.
8. After the connection is deemed successful, any action can be done/selected from the main menu.

Source Code

Main.java

```
package com.main;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.nio.file.FileSystems;
import java.nio.file.Path;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.*;

import com.blockchain.Block;
import com.network.ClientManager;
import com.network.ServerManager;

import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;

import static java.lang.System.exit;
import static java.lang.System.lineSeparator;

public class Main {

    private static final String DEFAULT_SERVER_ADDR = "localhost";
    private static final int DEFAULT_PORT = 6777;

    /*
     * Everything starts from here!
     */
    public static void main(String[] args) {
        //      int clientId=0;
        System.out.println(Vars.OWNERINFO);
        System.out.println("1. Cast Votes");
        System.out.println("2. View Votes on Blockchain");
        System.out.println("3. Count Votes");
        System.out.println("0. Exit\n");

        Scanner scanner = new Scanner(System.in);
```

```

System.out.println("Enter your choice: ");
int ch = scanner.nextInt();

if(ch == 1)
{
    System.out.println("\n ----- Casting Votes ----- \n");
    System.out.println("Please choose a role you want to be: server or
client.");
    System.out.println("server PORT - The port to listen to; \"6777\" is
default port.");
    System.out.println("client SERVER_ADDRESS PORT - The server address and
port to connect to; \"localhost:6777\" is default address-prt combination.");
    System.out.println("Make sure run the server first and then run client to
connect to it.");
    System.out.println("> ----- ");

    Scanner in = new Scanner(System.in);
    String line = in.nextLine();
    String[] cmd = line.split("\\s+");

    if (cmd[0].contains("s"))
    {
        // server selected

        /* work as server */
        int port = DEFAULT_PORT;
        if (cmd.length > 1) {
            try {
                port = Integer.parseInt(cmd[1]);
            } catch (NumberFormatException e) {
                System.out.println("Error: port is not a number!");
                in.close();
                return;
            }
        }

        ServerManager _svrMgr =new ServerManager(port);
        new Thread(_svrMgr).start();

    }
    else if (cmd[0].contains("c"))
    {
        //client selected

```

```

        /* work as client */
        String svrAddr = DEFAULT_SERVER_ADDR;
        int port = DEFAULT_PORT;
        if (cmd.length > 2) {
            try {
                svrAddr = cmd[1];
                port = Integer.parseInt(cmd[2]);
            } catch (NumberFormatException e) {
                System.out.println("Error: port is not a number!");
                in.close();
                return;
            }
        }

        ClientManager _cltMgr = new ClientManager(svrAddr, port);

        /* new thread to receive msg */
        new Thread(_cltMgr).start();

        _cltMgr.startClient();
    }
    else {
        showHelp();
        in.close();
        return;
    }
    in.close();
}

// VIEW VOTES
else if(ch == 2)
{
    System.out.println("\n ----- Displaying Votes ----- \n");

    String userHomePath = System.getProperty("user.home");
    String fileName;
    fileName=userHomePath+"/Desktop/blockchain_data";
    fileName=Vars.DATAFILE;
    File f=new File(fileName);

    try
    {
        if(!f.exists())
            System.out.println("Blockchain file not found");
    }
}

```

```

        ObjectInputStream in=new ObjectInputStream(new
FileInputStream(fileName));

        ArrayList<SealedObject> arr=(ArrayList<SealedObject>) in.readObject();
        for(int i=1;i<arr.size();i++) {
            System.out.println(decrypt(arr.get(i)));
        }
        in.close();

        System.out.println("-----\n");

    } catch (IOException e1) {
        e1.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (NoSuchPaddingException e) {
        e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (InvalidKeyException e) {
        e.printStackTrace();
    }
}

// COUNT VOTES
else if(ch == 3)
{
    String userHomePath = System.getProperty("user.home");
    String fileName;
    fileName=userHomePath+"/Desktop/blockchain_data";
    //fileName="c:/temp/sankar.txt";
    fileName=Vars.DATAFILE;
    File f=new File(fileName);

    try
    {
        if(!f.exists())
            System.out.println("Please cast your votes first !");

        else
        {
            System.out.println();
            System.out.println("-----");

```



```

        System.out.println("Vote count: ");
        ObjectInputStream in=new ObjectInputStream(new
FileInputStream(fileName));

        ArrayList<SealedObject> arr=(ArrayList<SealedObject>)
in.readObject();

        HashMap<String,Integer> voteMap = new HashMap<>();

        for(int i=1; i<arr.size(); i++)
        {
            Block blk = (Block) decrypt(arr.get(i));
            String key = blk.getVoteObj().getVoteParty();

            voteMap.put(key,0);
        }

        for(int i=1;i<arr.size();i++) {
            Block blk = (Block) decrypt(arr.get(i));
            String key = blk.getVoteObj().getVoteParty();

            voteMap.put(key, voteMap.get(key)+1);
        }
        in.close();

        for(Map.Entry<String, Integer> entry : voteMap.entrySet()) {
            System.out.println(entry.getKey() + " : " + entry.getValue());
        }

        System.out.println("-----\n");
    }

} catch (IOException e1) {
    e1.printStackTrace();
} catch (ClassNotFoundException e) {
    e.printStackTrace();
} catch (NoSuchPaddingException e) {
    e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
} catch (InvalidKeyException e) {
    e.printStackTrace();
}
}

```

```

        else if(ch == 0)
            exit(0);
    }

    public static void showHelp() {
        System.out.println("Restart and select role as server or client.");
        exit(0);
    }

    public static Object decrypt(SealedObject sealedObject) throws IOException,
    NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException
    {
        SecretKeySpec sks = new SecretKeySpec("MyDifficultPassw".getBytes(), "AES");

        // Create cipher
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, sks);

        try {
            // System.out.println(sealedObject.getObject(cipher));
            return sealedObject.getObject(cipher);
        } catch (ClassNotFoundException | IllegalBlockSizeException |
        BadPaddingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            return null;
        }
    }
}

```

Block.java

```

package com.blockchain;

import java.io.Serializable;

public class Block implements Serializable
{
    //inner class
    public class Vote implements Serializable
    {
        private String voterId;
        private String voterName;
        private String voteParty;

        public Vote(String voterId, String voterName, String voteParty)

```

```

    {
        this.voterName=voterName;
        this.voterId=voterId;
        this.voteParty=voteParty;
    }

    public String getVoterId() {
        return voterId;
    }

    public void setVoterId(String voterId) {
        this.voterId = voterId;
    }

    public String getVoterName() {
        return voterName;
    }

    public void setVoterName(String voterName) {
        this.voterName = voterName;
    }

    public String getVoteParty() {
        return voteParty;
    }

    public void setVoteParty(String voteParty) {
        this.voteParty = voteParty;
    }
}

private Vote voteObj;

private int previousHash;
private int blockHash;

public Block(int previousHash, String voterId, String voterName, String voteParty)
{
    this.previousHash=previousHash;
    voteObj=new Vote(voterId, voterName, voteParty);

    Object[] contents={voteObj.hashCode(), previousHash};
    this.blockHash=contents.hashCode();
}

```

```

    public Vote getVoteObj() {
        return voteObj;
    }

    public void setVoteObj(Vote voteObj) {
        this.voteObj = voteObj;
    }

    public int getPreviousHash() {
        return previousHash;
    }

    public void setPreviousHash(int previousHash) {
        this.previousHash = previousHash;
    }

    public int getBlockHash() {
        return blockHash;
    }

    public void setBlockHash(int blockHash) {
        this.blockHash = blockHash;
    }

    @Override
    public String toString() {
        return "Voter Id:" + this.voteObj.voterId + "\nVoter Name: " +
this.voteObj.voterName + "\nVoted for party: " + this.voteObj.voteParty;
    }
}

```

Vars.java

```
package com.main;
```

```

public class Vars {
    public static String DATAFILE = System.getProperty("user.dir") + "\\\" +
"BlockchainData";
    public static String OWNERINFO = "\n\n---[          Sastra B.Tech          ]---\n\n---[
Blockchain Proof of Concept ]---\n\n";
}

```

ClientManager.java

```
package com.network;
```

```

import java.io.BufferedReader;
import java.io.File;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.nio.file.Files;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.EnumSet;
import java.util.HashSet;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SealedObject;
import javax.crypto.spec.SecretKeySpec;

import com.blockchain.Block;
import com.main.Vars;

import static com.main.Main.decrypt;
import static java.nio.file.attribute.PosixFilePermission.*;

public class ClientManager extends NetworkManager {

    /* the socket communicating with server */
    private Socket _socket = null;
    private Block genesisBlock;
    private ArrayList<SealedObject> blockList;
    private ArrayList<String> parties;
    private HashSet<String> hashVotes;
    private int prevHash=0;

    private int clientId;

    public ClientManager(String addr, int port) {
        try {
            _socket = new Socket(addr, port);

```

```

        System.out.println("Connected to server: " + addr + ":" + port);
        genesisBlock=new Block(0, "", "", "");
        hashVotes=new HashSet<>();
        parties = new ArrayList<>();
        parties.add("BJP");
        parties.add("INC");
        parties.add("BSP");

        blockList=new ArrayList<>();
        blockList.add(encrypt(genesisBlock));
    } catch (IOException e) {
        System.out.println("Cannot connect to server " + addr + ":" + port);
        e.printStackTrace(e.getStackTrace());
        System.exit(0);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public void startClient() {

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Welcome to Voting Machine ! ");
    String choice ="y";
    do{
        Block blockObj=null;

        String voterId= null;
        String voterName =null;
        String voteParty=null;

        try {
            System.out.print("Enter Voter ID : ");
            voterId = br.readLine();
            System.out.print("Enter Voter Name : ");
            voterName = br.readLine();

            System.out.println("Vote for parties:");
            int voteChoice;

            do {
                for (int i=0 ;i<parties.size() ;i++) {
                    System.out.println((i+1)+" . "+ parties.get(i));

```

```

    }

    System.out.println("Enter your Vote : ");
    voteParty=br.readLine();
    voteChoice=Integer.parseInt(voteParty);
//    System.out.println("vote choice : "+ voteChoice);
    if(voteChoice>parties.size()||voteChoice<1)
        System.out.println("Please enter correct index .");
    else
        break;
}while(true);

voteParty = parties.get(voteChoice-1);
blockObj=new Block(prevHash, voterId, voterName, voteParty);

if(checkValidity(blockObj)) {
    hashVotes.add(voterId);
    sendMsg(new MessageStruct( 1,encrypt(blockObj) ));

    prevHash=blockObj.getBlockHash();
    blockList.add(encrypt(blockObj));
    //add
}
else
{
    System.out.println("Vote Invalid.");
}
System.out.println("Cast Another Vote (y/n) ? ");
choice=br.readLine();

} catch (IOException e) {
    System.out.println("ERROR: read line failed!");
    return;
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}while(choice.equals("y")||choice.equals("Y"));
close();
}

public SealedObject encrypt(Block b) throws Exception
{
    SecretKeySpec sks = new SecretKeySpec("MyDifficultPassw".getBytes(), "AES");

```

```

        // Create cipher
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");

        //Code to write your object to file
        cipher.init( Cipher.ENCRYPT_MODE, sks );

        return new SealedObject( b, cipher);
    }

    private boolean checkValidity(Block blockObj) {
        // TODO Auto-generated method stub
        if( hashVotes.contains((String)blockObj.getVoteObj().getVoterId() ))
            return false;
        else
            return true;
    }

    public void sendMsg(MessageStruct msg) throws IOException {
        sendMsg(_socket, msg);
    }

    // Close the socket to exit.
    public void close() {

        String userHomePath = System.getProperty("user.home");
        String fileName;
        fileName=userHomePath+"/Desktop/blockchain_data";
        // fileName="c:/temp/sankar.txt";
        fileName=Vars.DATAFILE;
        System.out.println("Data file at : " + fileName);
        File f=new File(fileName);

        try
        {
            if(!f.exists())
                f.createNewFile();
            else {
                f.delete();
                f.createNewFile();
            }

            // Files.setPosixFilePermissions(f.toPath(), EnumSet.of(OWNER_READ,
OWNER_WRITE, OWNER_EXECUTE, GROUP_READ, GROUP_EXECUTE));

```



```

        System.out.print("Thanks for using!\nClient Session Ended\nVote information
stored at ");
        System.out.println(fileName);

        ObjectOutputStream o=new ObjectOutputStream(new
FileOutputStream(fileName,true));
        o.writeObject(blockList);

        o.close();

        _socket.close();

    } catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    System.exit(0);
}

@Override
public void msgHandler(MessageStruct msg, Socket src) {
    switch (msg._code) {
        case 0:
            /* message type sent from server to client */
//            System.out.println((String)msg._content.toString()) ;
            try {

                blockList.add((SealedObject)msg._content);

                Block decryptedBlock=(Block) decrypt((SealedObject)msg._content);
                hashVotes.add(decryptedBlock.getVoteObj().getVoterId());

            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            break;
        case 1:
            /* message type sent from broadcast to all clients */
            //server manages this
            break;
        case 2:
            clientId=(int)(msg._content);

```

```

        default:
            break;
    }
}

/*
 * Running a loop to receive messages from server. If it fails when receiving, the
 * connections is broken. Close the socket and exit with -1.
 */
@Override
public void run() {
    while(true) {
        try {
            receiveMsg(_socket);

        } catch (ClassNotFoundException | IOException e) {
            if(_socket.isClosed())
            {
                System.out.println("Bye.");
                System.exit(0);
            }

            System.out.println("Connection to server is broken. Please restart
client.");

            close(_socket);
            System.exit(-1);
        }
    }
}
}

```

ServerManager.java

```

package com.network;

import java.io.IOException;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.Map;
import java.util.concurrent.ConcurrentSkipListMap;
import java.util.concurrent.atomic.AtomicInteger;

```

```

import javax.crypto.SealedObject;

import com.blockchain.Block;

public class ServerManager extends NetworkManager {

    private ServerSocket _svrSocket = null;

    /* manage the increasing client id to assign a new client an id */
    private volatile AtomicInteger _cid = null;

    /* maintain the map between client id and socket of a client */
    private volatile Map<Integer, Socket> _clients = null;

    public ServerManager(int svrPort) {
        try {
            _clients = new ConcurrentSkipListMap<Integer, Socket>();
            _cid = new AtomicInteger(0);

            _svrSocket = new ServerSocket(svrPort);

            System.out.println("Waiting for clients...");
            System.out.println("Please connect to " + InetAddress.getLocalHost() + ":"
+ svrPort + ".");
        } catch (IOException e) {
            System.out.println("ERROR: failed to listen on port " + svrPort);
            e.printStackTrace();
        }
    }

    /*
     * Run a loop to accept incoming clients. Once a connection is established,
     * create a new instance of ServerHandler in a new thread to receive
     * incoming messages by running a loop.
     */
    @Override
    public void run() {
        while (true) {
            try {
                /* accepting new clients */
                Socket socket = _svrSocket.accept();
            }
        }
    }
}

```

```

        addClient(socket);
        System.out.println("New client(cid is " + getCid(socket) + ")
connected!");

        /* create a new instance of ServerHandler to receive messages */
        new ServerHandler(this, socket).start();
        /* send the client id to the new client */
        sendMsg(socket, new MessageStruct(2, Integer.valueOf(getCid(socket))));
    } catch (IOException e) {
        /* ServerSocket is closed */
        break;
    }
}

}

public void clientDisconnected(Socket client) {
    int cid = getCid(client);
    System.out.println("Client " + cid + " has disconnected.");

    deleteClient(cid);
}

/* ===== Message handlers begin =====*/

@Override
public void msgHandler(MessageStruct msg, Socket src) {
    switch (msg._code) {
        case 0:
            /* message type sent from server to client */
            //client manages this.
            break;
        case 1:
            /* message type sent from broadcast to all clients */
            System.out.println("Broadcasting : " + (String)msg._content.toString());

            broadcast((SealedObject)msg._content,src );
            break;
        default:
            break;
    }
}

private void broadcast(SealedObject o, Socket src) {

```

```

        //broadcast to all except src
        int srcCid=getCid(src);
//        System.out.println("Source id : "+ srcCid);
        for(int i = _cid.get()-1 ;i>=0;i--) {
            if(i!=srcCid) {
                try {
                    sendMsg(getClient(i), new MessageStruct(0, o));
                } catch (IOException e) {
                    System.out.println("ERROR: Connection with " + srcCid + " is
broken, message cannot be sent!");
                    e.printStackTrace();
                } catch (NullPointerException e) {
                    continue;
                }
            }
        }
    }
}

/* ===== Message handlers end =====*/

/* ===== Client info manage methods begin =====*/
private void addClient(Socket socket) {
    _clients.put(Integer.valueOf(_cid.getAndIncrement()), socket);
}

private boolean deleteClient(int idx) {
    if (_clients.remove(Integer.valueOf(idx)) == null) {
        System.out.println("delete failed!");
        return false;
    }
    return true;
}

private Socket getClient(int cid) {
    return (Socket)_clients.get(Integer.valueOf(cid));
}

private int getCid(Socket socket) {
    for (Map.Entry<Integer, Socket> entry : _clients.entrySet()) {
        if (entry.getValue() == socket) {
            return entry.getKey().intValue();
        }
    }
}

```

```

    }
    return -1;
}

/* ===== Client info manage methods end =====*/

    public void close() {
        System.out.println("Server is about to close. All connected clients will
exit.");
        try {
            _svrSocket.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println("Bye~");
    }
}

```

MessageStruct.java

```

package com.network;

import java.io.Serializable;
public class MessageStruct extends Object implements Serializable {
    private static final long serialVersionUID = 3532734764930998421L;
    public int _code;
    public Object _content;

    public MessageStruct() {
        this._code = 0;
        this._content = null;
    }

    public MessageStruct(int code, Object content) {
        this._code = code;
        this._content = content;
    }
}

```

NetworkManager.java

```

package com.network;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

public abstract class NetworkManager implements Runnable{

    /*
     * Send a message to socket
     * Send a message to socket
     */
    public void sendMsg(Socket socket, MessageStruct msg)
        throws IOException {
        ObjectOutputStream out;

        out = new ObjectOutputStream(socket.getOutputStream());
        out.writeObject(msg);
    }

    /*
     * Try to receive a message from socket.
     */
    public void receiveMsg(Socket socket)
        throws ClassNotFoundException, IOException {
        ObjectInputStream inStream = new ObjectInputStream(socket.getInputStream());
        Object inObj = inStream.readObject();

        if (inObj instanceof MessageStruct) {
            MessageStruct msg = (MessageStruct) inObj;
            msgHandler(msg, socket);
        }
    }

    /*
     * Close socket
     */
    public void close(Socket socket) {
        try {
            socket.close();
        } catch (IOException e) {

```

```

        e.printStackTrace();
    }
}

/*
 * An interface for ServerManager and ClientManager to implement, handling all
 * the incoming messages.
 */
public abstract void msgHandler(MessageStruct msg, Socket src);
}

```

ServerHandler.java

```

package com.network;

import java.io.IOException;
import java.net.Socket;

public class ServerHandler extends Thread{
    /* the socket to receive messages from */
    private Socket _socket = null;
    /* used for callback */
    public ServerManager _svrMgr = null;

    public ServerHandler(ServerManager svrMgr, Socket socket) {
        _svrMgr = svrMgr;
        _socket = socket;
    }

    /*
     * Keep running a loop to receive messages from a client specified by
     * _socket. Once the connection is broken, call ServerManager to
     * remove this client.
     */
    @Override
    public void run() {
        while (true)
            try {
                _svrMgr.receiveMsg(_socket);
            } catch (IOException | ClassNotFoundException e) {
                _svrMgr.clientDisconnected(_socket);
                break;
            }
    }
}

```


CHAPTER 3: The Output

Snapshots

Casting Votes:

```
----- Casting Votes -----  
Please choose a role you want to be: server or client.  
server PORT - The port to listen to; "6777" is default port.  
client SERVER_ADDRESS PORT - The server address and port to connect to; "localhost:6777" is default address-prt combinat  
ion.  
Make sure run the server first and then run client to connect to it.  
> -----  
server  
Waiting for clients...  
Please connect to Kailash/192.168.29.164:6777.
```

```
---[      Sastra B.Tech      ]---  
---[ Blockchain Proof of Concept ]---  
  
1. Cast Votes  
2. View Votes on Blockchain  
3. Count Votes  
0. Exit  
  
Enter your choice:  
1  
  
----- Casting Votes -----  
Please choose a role you want to be: server or client.  
server PORT - The port to listen to; "6777" is default port.  
client SERVER_ADDRESS PORT - The server address and port to connect to; "localhost:6777" is default address-prt combinat  
ion.  
Make sure run the server first and then run client to connect to it.  
> -----  
client  
Connected to server: localhost:6777  
Welcome to Voting Machine !  
Enter Voter ID : 100  
Enter Voter Name : Tarani  
Vote for parties:  
1. BJP  
2. INC
```

```

Vote for parties:
1. BJP
2. INC
3. BSP
Enter your Vote :
1
Cast Another Vote (y/n) ?
y
Enter Voter ID : 102
Enter Voter Name : Akash
Vote for parties:
1. BJP
2. INC
3. BSP
Enter your Vote :
2
Cast Another Vote (y/n) ?
n
Data file at : D:\BC\BC\classes\BlockChainData
Thanks for using!
Client Session Ended
Vote information stored at D:\BC\BC\classes\BlockChainData


D:\BC\BC\classes>

```

```

server
Waiting for clients...
Please connect to Kailash/192.168.29.164:6777.
New client(cid is 0) connected!
Broadcasting : javax.crypto.SealedObject@22e9d3ec
Broadcasting : javax.crypto.SealedObject@1543404e
Client 0 has disconnected.

```



The data is being stored into a file named BlockChainData in the format of SealedObject. After the client stops casting votes, it disconnects but saves the data. The data is fully erased only if we delete the BlockChainData File. Then it starts freshly as a new voting machine.

Displaying and Counting votes:

```
1. Cast Votes
2. View Votes on Blockchain
3. Count Votes
0. Exit

Enter your choice:
2

----- Displaying Votes -----

Voter Id:100
Voter Name: asdf
Voted for party: BJP
-----

D:\BC\BC\classes>java -jar bc.jar

---[          Sastra B.Tech          ]---
---[ Blockchain Proof of Concept ]---

1. Cast Votes
2. View Votes on Blockchain
3. Count Votes
0. Exit

Enter your choice:
3

-----

Vote count:
BJP : 1
-----

D:\BC\BC\classes>
```

When the client selects the display option, the data is reviewed again and displayed on the command prompt, and when the client selects 'Count Votes', the votes are counted and displayed.

Votes being Invalid:

```
1. Cast Votes
2. View Votes on Blockchain
3. Count Votes
0. Exit

Enter your choice:
1

----- Casting Votes -----

Please choose a role you want to be: server or client.
server PORT - The port to listen to; "6777" is default port.
client SERVER_ADDRESS PORT - The server address and port to connect to; "localhost:6777" is default address-port combination.
Make sure run the server first and then run client to connect to it.
> -----
client
Connected to server: localhost:6777
Welcome to Voting Machine !
Enter Voter ID : 100
Enter Voter Name : asdf
Vote for parties:
1. BJP
2. INC
3. BSP
Enter your Vote :
1
Cast Another Vote (y/n) ?
y
Enter Voter ID : 100
Enter Voter Name : asd
Vote for parties:
1. BJP
2. INC
3. BSP
Enter your Vote :
1
Vote Invalid.
Cast Another Vote (y/n) ?
```

Whenever the client enters the same Voter ID, no matter what the name is the vote is considered invalid and the details of it is not stored, but discarded. It will also not be displayed when "Display Votes" command is given. This is to make sure that there are no duplication.

Shortcomings

```
PROBLEMS 15 OUTPUT DEBUG CONSOLE TERMINAL

Please choose a role you want to be: server or client.
server PORT - The port to listen to; "6777" is default port.
client SERVER_ADDRESS PORT - The server address and port to connect to; "localhost:6777" is default address-prt combination.
Make sure run the server first and then run client to connect to it.
> -----
client
Cannot connect to server localhost:6777
PS C:\Users\taran>
```

When we enter client at first without opening a terminal server, or run both client and server in the same terminal,

then we would get an error that it “Cannot connect to server localhost:6777”.

To overcome this issue, separate terminals are to be given to each other server and clients.

```
ntManager.java 2 MessageStruct.java 1 NetworkManager.java 1 ServerHandler.java 1 ServerManager.java 1 Block.java 1
C:\Users\taran> cd Downloads\Blockchain-based_E-Voting-master\Blockchain-based_E-Voting-master\src\com\blockchain > Block.java > Block
46 private Vote voteObj;
PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\taran> & 'C:\Users\taran\AppData\Local\Programs\Eclipse Foundation\jdk-11.0.12.7-hotspot\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:50436' '-cp' 'C:\Users\taran\AppData\Local\Temp\vscodesws_e2b40\jdt_ws\jdt.ls' '-java-project\bin' 'com.main.Main'
----- MAIN MENU -----
1. Cast Votes
2. View Votes on Blockchain
3. Count Votes
0. Exit

Enter your choice:
2

----- Displaying Votes -----

Blockchain file not found
java.io.FileNotFoundException: C:\Users\taran\Desktop\blockchain_data (The system cannot find the path specified)
    at java.base/java.io.FileInputStream.open0(Native Method)
    at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:112)
    at com.main.Main.main(Main.java:133)
PS C:\Users\taran>
```

When the Directory isn't pointing to the desired path, there appears an Exception, “java.io.FileNotFoundException”. The directories are the most important part of this particular code.

When the code is run not using the said jar file and using only the codes, the blockchain might not do the job. This may lead to not saving the data in blocks and hence losing it as soon as the client disconnects.

Conclusion

The web application is designed to serve the purpose of implementing blockchain and ease the difficulty of general public in terms of transportation during the Election times. This serves as a secure yet easy way of Applying vote which is compulsory for all the citizen in our country.

Future plans

This is a reliable Proof of Concept which is implemented and yet to be developed to its full extent. This POC needs Graphical User Interface and also little more development in the security part (logging in with a password,etc). This POC mainly concentrates on the blockchain technology involved and thereby ceasing duplication.

References

Anderson C.(2006). How to Rig a Democracy: A Timeline of Electronic Voting in the United States. The Independent.

Bellis, M. (2007). The History of Voting Machines.

Cranor, L.F, & Cytron, R.K. (1996). Design and Implementation of a Security-Conscious Electronic Polling System.

Electronic Voting and Counting – Development of the System. (2005). Elections ACT.

<http://www.elections.act.gov.au/EVACS.html>

<https://academy.binance.com/academy/blockchain>

How Does Blockchain Work: Simply Explained - 101 Blockchains