A Report on Assignment 3
of
Design and Implementation of a Decentralized News-Checking DApp

Introduction of Blockchains, Cryptocurrencies, and Smart Contracts

CS 765

Created By

Santanu Sahoo (23M0777)
Arnab Bhakta (23M0835)

IIT Bombay
Mumbai

Instructor

Prof. Vinay Ribeiro

Contents

1 Introduction

The proliferation of fake news in the digital age poses a significant threat to the integrity of information dissemination and public discourse. Traditional centralized fact-checking mechanisms may be susceptible to biases and lack transparency, raising concerns about their credibility. In response to these challenges, decentralized applications (DApps) built on permissionless blockchains offer a promising alternative.

This report presents the design and implementation of a DApp aimed at combating fake news. Leveraging blockchain technology, the DApp provides a decentralized platform for fact-checking news articles and items. Users can request fact-checking, while registered fact-checkers independently assess the truthfulness of news items through transparent and decentralized voting mechanisms.

Key Features of the DApp:

1. **Permissionless Fact-Checking:** Any user can request the DApp to fact-check a news article or item, ensuring inclusivity and accessibility.
2. **Decentralized Fact-Checking:** Registered fact-checkers, distributed across the network, can independently assess the truthfulness of news items, reducing reliance on centralized authorities.
3. **Transparent Voting Mechanism:** Fact-checkers can vote on the veracity of news items using a binary or numerical scale, with the DApp considering all votes to determine the overall truthfulness score.
4. **Mitigation of Sybil Attacks:** The DApp employs strategies to mitigate Sybil attacks, where malicious users create multiple identities to skew voting results.
5. **Trustworthiness Evaluation:** Mechanisms are in place to evaluate the trustworthiness of fact-checkers based on their voting history, ensuring that the opinions of more trustworthy users are given greater weight.
6. **Incentivization of Rational Voting:** Rational voters are incentivized to participate and vote truthfully through mechanisms such as reputation systems and token rewards.

By harnessing blockchain's transparency and decentralization, this DApp aims to foster a more informed and trustworthy information ecosystem, promoting the integrity of news reporting.

Approach to Addressing Challenges

2 Sybil Attack Handling Approach

The strategy for mitigating **Sybil attacks** involves implementing stringent measures to ensure the integrity of our decentralized application (DApp). Here's how we address this challenge:

1. **Account Creation Fee:** We demand payment of a joining fee from users upon registration on our platform to discourage dishonest actors from establishing numerous fraudulent identities. This cost serves as a barrier to entrance and prevents the formation of many false identities.

2. **One Account, One Vote Policy:** Every user account is limited to one vote on any one news article, as per our stringent policy. This keeps people from utilizing numerous accounts to cast multiple ballots and therefore having disproportionate influence on the results of the election.

3. **Voting Deposit Requirement:** Before casting their vote, users are required to deposit a certain amount of cryptocurrency(Bitcoin) as a voting fee. This deposit serves as a form of collateral and incentivizes users to vote responsibly. Additionally, it imposes a cost on malicious actors who seek to manipulate the system by creating numerous accounts to skew the voting results.

By implementing these measures, we ensure the integrity and fairness of our fact-checking process, safeguarding against the threat of Sybil attacks and maintaining the trust of our user community.

3 Method to evaluate or re-evaluate the trustworthiness of voters

We evaluate how trustworthy each voter is by keeping track of their voting history. This involves recording two things: the total number of votes they've cast *total_votes_casted*) and how many of those votes were correct (*correct_votes_casted*). We then calculate their trustworthiness based on the ratio of correct votes to total votes cast(*total_votes_casted*) and the number of correct votes they've made (*correct_votes_casted*). The trustworthiness of a voter is determined by the ratio of correct votes to total votes cast:

$$\text{Trustworthiness} = \frac{\text{correct\_votes\_casted}}{\text{total\_votes\_casted}}$$

We update the voter rating for the last n voters using the following aproach:

Algorithm 1 Re-evaluation of trustworthiness of Voters

```
for each (topic, rating, total_votes_Casted, correct_votes_casted) in rating do
    if news.accuracy = user.vote then
        Increment correct_votes_casted by 1
        Increment total_votes_Casted by 1
    end if
    rating[topic] = correct_votes_casted / total_votes_Casted
end for
```

4 Weighted Voting System and Topic-Specific Trust Evaluation

In this section, we discuss our approach to implementing a weighted voting system and evaluating the trustworthiness of voters based on specific topics.

To achieve this, we store the votes and ratings for each topic in the format <**topic, rating, vote**>. These records are crucial for calculating the trustworthiness of news items.
When a news item covers multiple topics, we employ the following algorithm to address this issue:

Algorithm 2 Calculation of News Accuracy for Multi-topic News

```
news_accuracies ← {}
for each topic in news.topics do
    news_accuracies[topic] ← 0
    for each vote in news.votes do
        news_accuracies[topic] ← news_accuracies[topic] + vote.value × vote.voter.rating[topic]
    end for
    news_accuracies[topic] ← news_accuracies[topic]/len(news.votes)
end for
news_accuracy ← avg(news_accuracy_per_topic)
news.accuracy ← news_accuracy
```

This algorithm iterates over each topic covered by the news item and calculates the accuracy score based on the weighted average of votes from trustworthy voters for that specific topic. Finally, the overall accuracy of the news item is determined by averaging the accuracy scores across all topics.

5 Incentivizing Truthful Participation of Rational Voters

After a voter submits a deposit to cast their vote on a news item, upon the declaration of the vote result, rational voters will receive back their deposited amount. Additionally, the deposited amounts from fake voters will be redistributed among the rational voters as an incentive. This incentivizes voters to provide truthful and accurate votes, thereby promoting a culture of honesty and integrity in the fact-checking process.

Suppose a decentralized fact-checking platform requires users to submit a small amount of Bitcoin as a deposit to cast their vote on a news item. **Let's say the deposit amount is 0.001 BTC.**

After the vote result is declared, rational voters who provided accurate and truthful votes will receive back their deposited amount of **0.001 BTC.**

Now, let's assume that there were some fake voters who deliberately provided incorrect votes. Their deposited amounts, totaling **0.005 BTC (Asume 5 fake voters)**, will be redistributed among the rational voters who casted truthful votes.

For instance, if there were **10 rational** voters who casted accurate votes, each of them would receive an additional **0.0005 BTC** as an incentive for their honesty.

6 News Uploading Approach

Authentication Requirement: The news publisher must possess a valid ID to upload news articles onto the platform.

News Structure: When publisher upload a new news , the belwo requirement must be satisfied:

- **Body:** The content of the news article.
- **Title:** The title or headline of the news article.
- **Publisher and News Id:** The name of the publisher along with their unique ID.
- **Topic:** The category or type of news article (e.g., Politics, Sports, Technology).
- **Timestamp:** The time at which the news article was generated or uploaded.

News Uploading In solidity Function :

```
pragma solidity ^0.8.0;
    function uploadNews(string memory _title, string memory _body, string[]
        memory _topics, bool _truthValue) public {
        require(msg.sender == author, "Only the author can upload news");

        // Set news details
        title = _title;
        body = _body;
        topics = _topics;
        truthValue = _truthValue;
        timestamp = block.timestamp;
        newsId = keccak256(abi.encodePacked(author, _title, _body, _topics,
            _truthValue, timestamp));
    }
}
```

The uploadNews function enables the contract deployer, acting as the author, to submit a fresh news article to the contract, specifying its title, body, topics, and truth value. Initially, the function verifies if the caller holds the authorization to upload news. Upon validation, it proceeds to set the news details and generates a unique newsId through cryptographic hashing. These details are then securely stored within the contract's variables, thereby allowing accessibility to other functions and users across the blockchain network.

7 Bootstrapping

Bootstrapping is the initial setup process in a system, often establishing the foundational conditions for its operation. In the context of our simulation, we commence by assigning a 100% rating to the most trustworthy voters. This ensures that there are reliable participants in the voting process who can accurately assess the truthfulness of news items. For other voters, we initialize their ratings to zero. As the simulation progresses and voters cast their votes, their ratings will be adjusted based on the accuracy of their voting behavior. This approach ensures that the voting system starts with a foundation of trustworthiness and adapts dynamically as the simulation unfolds.

8 Pseducode Code in Solidity Style

8.1 Smart Contract Pseducode Overview :

```
pragma solidity ^0.8.0;

contract NewsDetection {

    struct News{    }
    struct Vote {    }
    struct VoterRating {    }
    struct Voter {    }

    News immutable news;

    uint constant deposit = 10 ether;
    uint constant NUM_VOTE_THRESHOLD = 100;

    bool detectedAccuracy;
    bool isLocked;
    uint accuracy;
    Vote[] votes;
    uint private totalDeposit;

    constructor(address _author, string memory _title, string memory _body,
       string[] memory _topics) {    }

    function hasVoted(address _voter) private view returns (bool) {    }
    function castVote(Voter _voter, bool _voteValue) public {    }
    function calculateNewsAccuracyTopic(string memory _topic) private view
       returns (uint[2] memory) {    }
    function calculateAccuracy() private {    }
    function recalculateRating() private {    }
    function increaseVoterRating(address _voter) private {    }
    function decreaseVoterRating(address _voter) private {    }
    function distributeRewards() private {    }
    function rewardVoter(address _voter, uint rewardAmount) private {    }
    function lockVotes() private {    }
    function unlockNews() public {    }
}
```

This Solidity smart contract, **NewsDetection,** facilitates a voting system for detecting the accuracy of news. It allows users to cast votes on news articles' accuracy, calculates the overall accuracy based on the votes received, rates voters based on their participation, and distributes rewards accordingly. The contract includes functions for checking if a user has already voted, casting votes, calculating news accuracy, recalculating ratings, increasing and decreasing voter ratings, distributing rewards, and managing the locking and unlocking of news for voting.

8.2 Structures present in Smart contract

```
struct News{
    address immutable author;
    string immutable title;
    string immutable body;
    string[] immutable topics;
    bytes32 immutable newsId;
    uint immutable timestamp;
}

struct Vote {
    Voter voter;
    bool voteValue;
}

struct VoterRating {
    uint numTotalVotes;
    uint numCorrectVotes;
}

struct Voter {
    address voter;
    map(string => VoterRating) ratings;
}
```

These structs encapsulate key components of a decentralized news verification system. The ‘News‘ struct holds immutable details of a news article such as author’s address, title, body, topics, unique identifier, and timestamp. ‘Vote‘ records a voter’s decision on news truthfulness, containing their identity and a boolean value indicating the vote. ‘VoterRating‘ tracks a voter’s performance, storing total and correct votes across topics. ‘Voter‘ represents a participant, storing their address and maintaining topic-wise ratings for news verification.

8.3 Pseducode for casting vote

```
function hasVoted(address _voter) private view returns (bool) {
    for (uint i = 0; i < votes.length; i++) {
        if (votes[i].voter == _voter) {
            return true;
        }
    }
    return false;
}
function castVote(Voter _voter, bool _voteValue) public {
    require(!isLocked, "Voting is locked");
    require(!hasVoted(_voter.voter), "Already voted");
    require(msg.value == deposit, "Incorrect deposit amount");

    votes.push(Vote(_voter, _voteValue));
    totalDeposit += deposit;
   if (votes.length >= NUM_VOTE_THRESHOLD) {
        calculateAccuracy();
    }
}
```

8.4 News Item accuracy calculation :

```
function calculateNewsAccuracyTopic(string memory _topic) private view
    returns (uint[2] memory) {
     uint[2] memory newsAccuraciesTopic;
     uint[2] memory numVotes;
     for (uint i = 0; i < votes.length; i++) {
         if (votes[i].voteValue == true) {
             numVotes[0]++;
             newsAccuraciesTopic[0] +=
                votes[i].voter.ratings[_topic].numCorrectVotes *100 /
                votes[i].voter.ratings[_topic].numTotalVotes;
         } else {
             numVotes[1]++;
             newsAccuraciesTopic[1] +=
                votes[i].voter.ratings[_topic].numCorrectVotes *100 /
                votes[i].voter.ratings[_topic].numTotalVotes;
         }
     }
     newsAccuraciesTopic[0] /= numVotes[0]*100;
     newsAccuraciesTopic[1] /= numvotes[1]*100;
     return newsAccuraciesTopic;
}

function calculateAccuracy() private {
     uint[2] memory newsAccuracies;
     for (uint i = 0; i < topics.length; i++) {
         uint[2] memory newsAccuraciesTopic =
            calculateNewsAccuracyTopic(topics[i]);
         newsAccuracies[0] += newsAccuraciesTopic[0];
         newsAccuracies[1] += newsAccuraciesTopic[1];
     }

     uint totalTrueVotes = newsAccuracies[0]*100/length(topics);
     uint totalFalseVotes = newsAccuracies[1]*100/length(topics);
     uint totalVotes = totalTrueVotes + totalFalseVotes;

     require(totalVotes > 0, "No votes cast");

     if (totalTrueVotes > totalFalseVotes ) {
         detectedAccuracy = true;
         accuracy = (totalTrueVotes * 100) / totalVotes;
     } else if (totalFalseVotes > totalTrueVotes ){
         detectedAccuracy = false;
         accuracy = (totalFalseVotes * 100) / totalVotes;
     } else {
         revert("Draw");
     \
     distributeRewards();
     lockVotes();
     recalculateRating();
}
```

8.5 Recalculate Voter Ratings :

```
function recalculateRating() private {
    for (uint i = 0; i < votes.length; i++) {
        if (votes[i].voteValue == detectedAccuracy) {
            increaseVoterRating(votes[i].voter);
        } else {
            decreaseVoterRating(votes[i].voter);
        }
    }
}

function increaseVoterRating(address _voter) private {
    for (uint i = 0; i < news.topics.length; i++) {
        voters[_voter].ratings[news.topics[i]].numTotalVotes++;
        voters[_voter].ratings[news.topics[i]].numCorrectVotes++;
    }
}

function decreaseVoterRating(address _voter) private {
    for (uint i = 0; i < news.topics.length; i++) {
        voters[_voter].ratings[news.topics[i]].numTotalVotes++;
        voters[_voter].ratings[news.topics[i]].numCorrectVotes--;
    }
}
```

8.6 Distribute Rewards :

```
function distributeRewards() private {
    uint correctVotes = 0;

    for (uint i = 0; i < votes.length; i++) {
        if (votes[i].voteValue == detectedAccuracy)
            correctVotes++;
    }

    uint rewardAmount = totalDeposit / correctVotes;

    for (uint i = 0; i < votes.length; i++) {
        if (votes[i].voteValue == detectedAccuracy) {
            rewardVoter(votes[i].voter, rewardAmount);
        }
    }
}

function rewardVoter(address _voter, uint rewardAmount) private {
    payable(_voter).transfer(rewardAmount);
}
```

9 Simulation Analysis

9.1 Input Parameters

- **N**: Represents the total number of voters participating in the system.
- **Q**: Denotes the percentage of voters being malicious.
- **P**: Indicates the percentage of voters being trustworthy.
- **R** (INITIAL_RATING_MALICIOUS): Defines the initial rating assigned to malicious voters. This rating could influence their impact on the system initially.
- **TOPICS**: Represents a list of topics relevant to the news articles. Each voter may have ratings associated with these topics for evaluating news accuracy.
- **NUM_TRUE_NEWS**: Specifies the number of true news articles generated for the simulation.
- **NUM_FALSE_NEWS**: Specifies the number of false news articles generated for the simulation.

9.2 Results Summary

Input parameters (N=100 for all)			**Average Trustworthiness** for all voter types			**Detection Accuracy**
Q	P	R	Honest	Trusted	Malicious	percentage(%)
0.05	0.05	0	0.7	0.9	0	100
0.05	0.05	0	0.7	0.9	0	100
0.05	0.05	0	0.7	0.9	0	100
0.05	0.05	0	0.7	0.9	0	99.9
0.05	0.02	0.1	0.7	0.9	0.05	100
0.05	0.02	0.2	0.3	0.1	1	0
0.05	0.05	0.2	0.7	0.9	0.05	100
0.05	0.05	0.4	0.7	0.9	0.05	99.9
0.05	0.05	0.5	0.3	0.1	1	0
0.05	0.06	0.5	0.7	0.9	0.05	100
0.05	0.06	0.6	0.3	0.11	0.99	0.7
0.05	0.07	0.6	0.3	0.1	1	0
0.05	0.08	0.6	0.7	0.9	0	100

Table 1: Simulation Results Summary

9.3 Observation when R=0

After simulating with multiple values of Q and P while keeping R = 0, We find that When R, representing the weight assigned to voters' ratings, is set to 0, it means that the system does not consider the ratings of voters at all. Consequently, regardless of the initial rating assigned to malicious voters, their votes will not contribute to the determination of the accuracy of news articles. Since their votes are effectively disregarded, the system essentially relies solely on the votes of trustworthy voters. As a result, the accuracy of the system becomes 100%, as it is solely determined by the votes of trustworthy voters who are presumed to provide accurate assessments of news articles. Therefore, in this scenario, the accuracy of the system remains consistently high, unaffected by the presence or actions of malicious voters.

9.4 Observation when R>0 and Q>P

When the probability of malicious voters (Q) exceeds that of trustworthy voters (P), even with low values of R (weight assigned to voters' ratings), malicious voters can significantly influence the outcome. Their higher likelihood of being present and voting with malicious intent allows them to sway the overall decision, potentially leading to inaccurate determinations.

9.5 Observation when R>0 and Q=P

When Q is close to P, both honest and potentially malicious voters are equally represented. For R values up to 0.4, the system relies more on the ratings of honest voters, mitigating the influence of malicious actors. However, as R exceeds 0.5, all voters' ratings, including those from potentially malicious individuals, gain prominence. Consequently, malicious voters dominate the outcome, leading to potential inaccuracies in news article evaluations.

9.6 Observation when R>0 and Q<P

When Q is lower than P, trustworthy voters outnumber potentially malicious ones. In such cases, a considerably high R value is necessary for malicious voters to exert any significant influence on the outcome. With lower R values, the system prioritizes the ratings of trustworthy voters, thereby minimizing the impact of potentially malicious actors. As a result, achieving a balance between Q, P, and R is crucial to maintaining the integrity of the news verification system.

9.7 Conclusions

When R is 0, Trusted Voters solely determine the outcome, resulting in 100% accurate news detection, regardless of the Q value.

For non-zero R values:

- If Q exceeds P, even with low R values, malicious voters can sway the outcome.
- When Q is approximately equal to P, honest voters prevail when R is 0.4 or less, but malicious voters dominate when R surpasses 0.5.
- If Q is less than P, a significantly high R value is required for malicious voters to gain any influence on the outcome.