FoodChain: A Decentralized Application for Restaurant Selection

Peter M. VanNostrand
University at Buffalo
pmvannos@buffalo.edu

Abstract

FoodChain is a decentralized application for selecting a restaurant to eat at among small groups of people. The application allows an organizer to create a time sensitive poll, invite people to dinner with the poll, and then view the results of attendee voting. The winner is calculated using STV. The application is implemented using Etherum blockchain and has a web UI front.

1 Introduction

Often, groups of friends or family struggle to decide on where to go to dinner, wasting time and often resulting in a restaurant choice that leaves many individuals dissatisfied. FoodChain is designed to solve this problem by providing a decentralized application for restaurant selection that uses blockchain technology to provide a trusted and open decision making process. FoodChain allows a group organizer to create a virtual poll which is placed on the Etherum blockchain and contains a list of potential restaurants and a time limit after which the poll closes. The organizer can then invite people they know to dinner, by authorizing them to participate in the poll. When a voter responds to the poll they can vote for their preferred choice.

This application uses the principle of Single Transferrable Voting (STV) to determine the winner. STV is a method of voting designed to provide optimal results in multi-party systems and works well for the FoodChain use case as there are very frequently multiple restaurant proposals within a group. STV works by performing a series of instant runoff votes. When voting, participants rank their preference of the available proposals from first being the most desired to last being the least desired. Then, all the voters' first choices are counted as in a traditional vote and the least voted for proposal is determined and eliminated as a potential option. Once an option has been dropped the votes are recounted, with the first non-dropped option from each voter's ballot being counted. This repeats for as long as there are more options then winners, in this case until all but one proposal is eliminated.

STV is beneficial as it allows two similar proposals to exist in the same vote without "stealing" votes from each other, the less popular option is simply eliminated, and voters' next choice in the ranking is counted instead. This maximizes voter satisfaction in a multi-option vote.

2 Decentralized Application

2.1 Use Case Diagram

The following is a use case diagram for FoodChain which shows the relationships between actors and use cases in the application

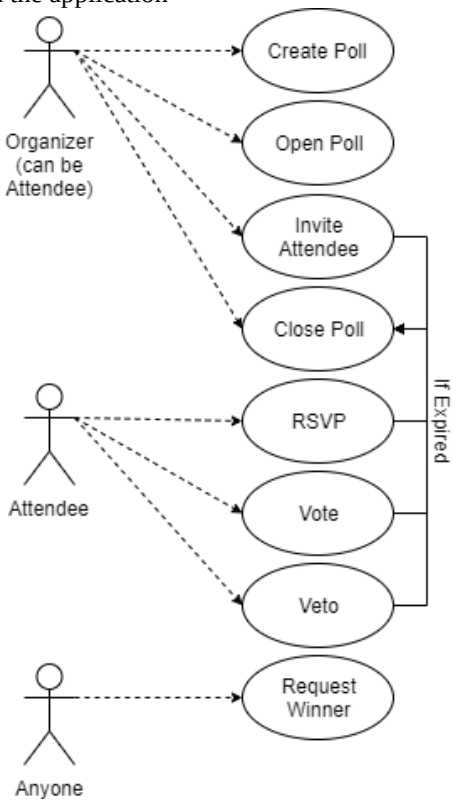

Figure 1: Use Case Diagram

2.2 Function Specifications

The following use cases are supported by this application, more may be added during the development process

Create Poll (constructor) – The event organizer creates a dinner poll with a defined number of proposals

Open Poll – The event organizer opens the poll for voting for a provided period of time, after which voting will no longer be allowed

Invite Attendee – The event organizer invites a guest to dinner by adding their blockchain address to the list of allowed voters

Vote – A registered voter votes by providing a ranked list of the proposals with first being their most preferred option and last being their least preferred option
Veto – An invited attendee indicates that they would be unable to eat at a given restaurant, flagging that option as unable to win the vote
Close Poll – The event organizer triggers the poll to "close" meaning that further voting is no longer allowed
Request Winner - Any actor requests that the current winner be computed by counting the current votes using the STV method
Check Time – A modifier that automatically closes the poll when a function call occurs the predefined poll duration has passed

2.3 Contract Diagram

Figure 2 shows a contract diagram for the FoodChain application. The diagram shows three cells: the data stored by the contract, the modifiers used by the contract, and the functions implemented in the contract.

The contract stores a list of proposals, a list of voters each with their votes, and information representing the current state of the diagram. A description of the functions can be found in Section 2.2. The only included modifier checks the relevant function calls to ensure that the poll has not expired.

FoodChain

```
enum State {init, voting, closed}
enum statusCode {success, notAllowed, incorrectState, invalidParameter}

struct Voter {}
struct Proposal {}

mapping(address => Voter) voters;
mapping(uint => address) voterAddresses;
uint numVoters;

State currentState;
address organizer;
Proposal[] proposals;

uint startTime = 0;
uint duration = 0;
```

```
modifier timed(){
       if(currentState==State.voting && now > (startTime + duration))
              currentState = State.closed;
      _;
}
```

```
constructor(uint8 numProposals) public {}
function invite(address voterAddress, bool allowVeto) public timed returns(statusCode code) {}
function rsvp(bool intent) public timed returns(statusCode code) {}
function vote(uint8[] memory votesArray) public timed returns(statusCode code) {}
function veto(uint8 proposalID) public timed returns(statusCode code) {}
function openPoll(uint durationSecs) public returns(statusCode code) {}
function closePoll() public returns (statusCode code) {}
function winningProposal() public view returns (uint8 winnerID) {}
```

Figure 2: Contract Diagram

2.4 Sequence Diagram

Figure 3 shows a sequence diagram for a simple use of the FoodChain application. First an organizer creates and opens the poll. Then they invite one or more voters and wait for the voters respond. Once invited the voters first RSVP to indicate their intended attendance, then they submit a ranked voting ballot and optionally veto a proposal. The poll then closes when called by the organizer or time has elapsed. Anyone can then determine the winner using STV by calling the provided function.

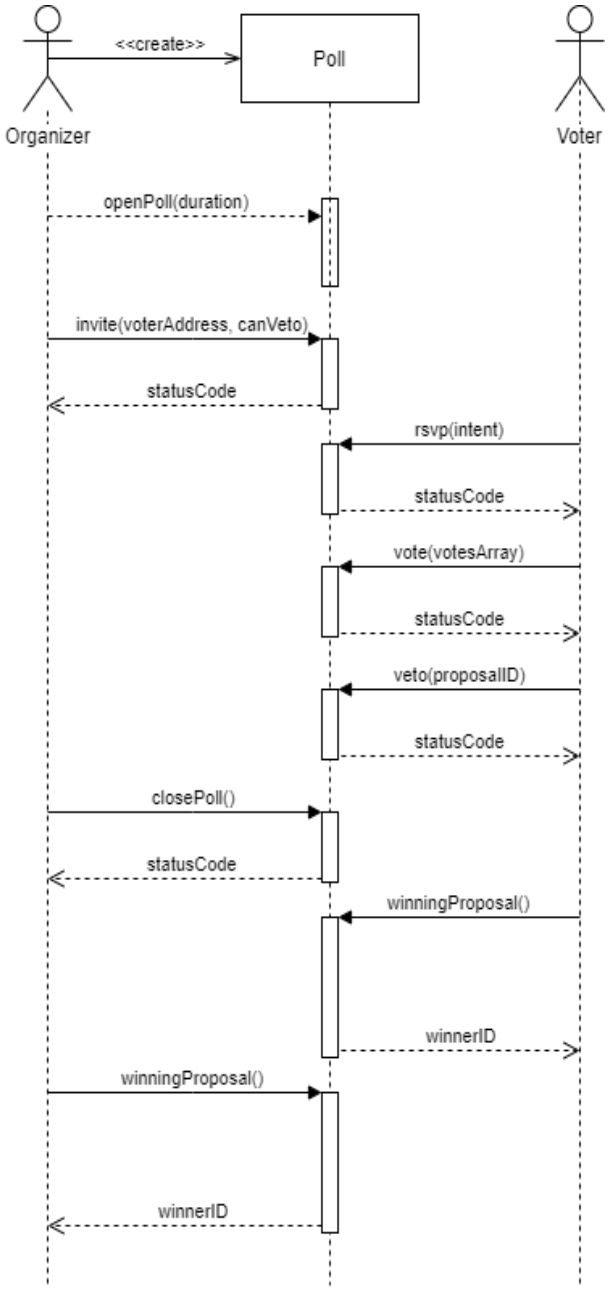

Figure 3: Sequence Diagram

2.5 Test Plan

The functionality of the FoodChain smart contract was tested using the Remix IDE both manually and through unit tests. Manual testing was used to check the functionality of the veto, openPoll, and closePoll functions as well as the timing modifier. Inviting, voting, RSVP, and winningProposal are covered by the Unit tests.

Most functions in the FoodChain application return a status code to indicate the success of the action requested. The potential values of these codes are shown below in Table I

Table I: Status Codes

Status Code	Value
Success	0
Not Allowed	1
Incorrect State	2
Invalid Parameter	3

2.5.1 Veto

1. Create and open a poll with two proposals
2. Invite three voters: two who cannot veto and one that can
3. Have all voters RSVP true to indicate they will attend
4. Have the two non-vetoing voters vote for proposal zero as their first choice and the third voter vote for proposal one as their first choice
5. Have the veto capable voter veto option zero. This should succeed and return zero
6. Close the poll and check that the winner is proposal one

2.5.2 Open Poll, Close Poll, and Timing

1. Create a poll with one proposal
2. Open the poll for a duration of 60 seconds
3. Invite two voters and have them RSVP true
4. Have the first voter vote within the 60 second duration, their vote should succeed and return a Success code
5. Have the second voter vote after the 60 second duration has elapsed, this should fail and return an Incorrect State code

2.5.3 Invite

This function is covered by the unit test checkInvite. In this test a new FoodChain contract is created and opened. Then a voter is invited and the status code is checked to be Success. A veto enabled voter is also invited and checked for a retuned Success code.

2.5.4 RSVP

This function is covered by the unit test checkRSVP. This test creates and opens new food chain, then invites a voter. This voter attempts to RSVP and the returned status code is checked to be Success. A second voter who is not invited also attempts to RSVP and the returned

status code is checked to be Not Allowed as only invited voters may RSVP.

2.5.5 Vote

This function is covered by the unit test checkVote. This test creates and opens a new FoodChain and then invites a voter who RSVPs true. Finally the voter attempts to vote by submitting an array of their ranked choices and the returned status code is checked to be Success.

2.5.6 Winning Proposal

This function is covered by the unit tests checkWinner and checkTransfer.

In the first test a poll is created, and opened, then three voters are invited and RSVP true. These voters then all select proposal two as their first choice. Winning Proposal is then called and the resulting winner ID is checked to be two.

The second test performs the same setup steps, but instead invites and RSVPs five voters. This test case verifies that the properties of STV are followed by implementing a three option poll with two popular options and one minority option that is similar to one of the popular options. This could be for example a poll containing a vegetarian location, a popular steakhouse, and a less well known barbeque location. In this case the steakhouse and barbeque restaurants are similar and between them hold a majority, but the presence of the barbeque restaurant splits the meat eating vote into two separate groups. In a traditional first-past-the-post election this would cause the vegetarian option to win, but instead using STV lets the barbeque proponents select the steakhouse as their second choice, such that when the minority barbeque option is eliminated their votes can be recounted for the steakhouse. (A parallel situation in US politics would be the Socialist Party siphoning votes from the Democratic Party due to their somewhat overlapping constituents)

Mechanically the test has two voters select option 0, two select option 1, and a third selects option 2 as their first choices. Critically the voter who selects option 2 as their first choice also selects option 1 as their second choice. When winningProposal is called it finds that the least selected first option is proposal 2, so this proposal is eliminated and the vote which picked this as their first choice is recounted for the second choice of option 1. Therefore proposal 1 is the winner and the returned winner ID is checked to match this value.

2.6 Remix Screenshot

Figure 4 shows a screenshot of the contract interface from the Remix IDE used for testing. The contract was deployed to a JavaScript virtual environment, by entering the number of proposals and clicking “Deploy” in the Remix “Deploy and Run Transactions” window.

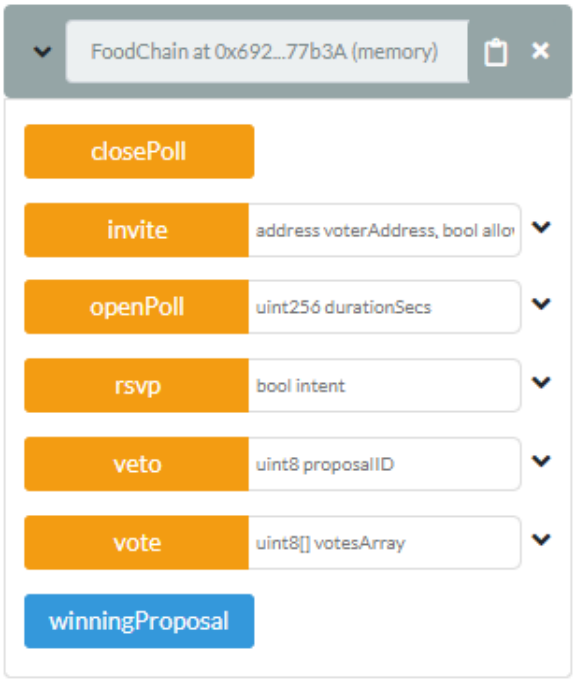

Figure 4: Remix IDE Contract Interface

3 Web UI

3.1 Setup

The provided web UI is configured to run on a local RPC network generated by Ganache. Then truffle is used to deploy the smart contract and npm is used to start the web server. Please use the following steps to setup the web UI for use

1. Launch Ganache
2. Connect MetaMask to your ganache blockchain accounts
3. Deploy the contract with the following commands
    a. cd FoodChain/FoodChain-contract
    b. truffle compile
    c. truffle migrate –reset
4. Start the Web UI server using the following commands
    a. cd FoodChain/FoodChain-app
    b. npm install
    c. npm start
5. Navigate to localhost:3010 in a web browser
6. For best experience open the development console (F12 in Chrome)

3.2 Using the Web UI

Once you've completed the setup steps above the system is ready to be tested and there are a few key details to note

3.2.1 Using Different Accounts

When using the FoodChain application, all function calls are performed as the currently selected account in MetaMask. To switch between accounts the user must open MetaMask and select a different account from the dropdown list. I recommend keeping the MetaMask dashboard open in a separate tab to allow quick switching between accounts.

3.2.2 Setting up the Poll

Once the contract is deployed the FoodChain organizer will need to open the poll and invite other users to vote. **This can only be done by Account1** as they are the organizer of the poll and deployed the contract originally. They are also the only account who can close the poll. All actions in the "Poll Controls" tile must be performed with Account1 selected in MetaMask to function as expected.

3.3 Recommended Testing Steps

The following is a recommended test procedure which should validate the majority of the FoodChain functionality. Please keep the development console open during this process (press F12 in Chrome) as it will provide positive and negative feedback for each action. Any red text in the console indicates that you've done something wrong.

1. Select Account1 in MetaMask
2. Enter a duration for the poll – for example 1000 seconds -- and press "Open Poll"
3. Select an account address from the dropdown and turn the "Veto Allowed" slider to on
4. Press "Invite" to invite that account to the poll and register them as an allowed voter
5. Switch to the account in MetaMask which corresponds to the invited address
6. With that account active in MetaMask toggle the "Will Attend" slider on and click "RSVP." This indicates the voter's intention to attend dinner
7. Enter your votes in the "Vote" tile. The text boxes should each contain only one number in the range 0-3 and no number should appear twice. For example, the four textboxes could contain [2, 1, 3, 0] but should not contain [2, 1, 3, 4] or [2, 2, 1, 1]
8. Press "Vote"
9. Scroll to the bottom of the page and press "Check Current Winner." You should see that proposal corresponding to your first vote entry has won.
10. Go to the "Veto" tile and enter the number of your first vote (the ID of the proposal that just won) and press the "Veto" button. This vetos the current winning option.
11. Press "Check Winning Option" again. You should observe that your first choice has been eliminated, and you're second choice is now the winner