Practical session for blockchain

(5) Prediction Market

Jungyoon Song[1] Giyeong Lee[1]

[1]Financial Risk Engineering Lab.
Seoul National University

May 2, 2022

DeFi: Decentralized Finance

- **DeFi** offers a financial instrument without intermediary, brokerage, or bank by using smart contracts on a blockchain.
- There are many categories of DeFi: Dexes, lending, bridge, liquid staking, etc.
  cf. `https://defillama.com/categories`
- The most common blockchain for DeFi is Ethereum. (approximately 70%)

DeFi Market Size

- Total Value Locked(TVL)
  : The sum of all crypto staked, loaned, or used for other financial actions across all of DeFi.

- TVL across DeFi blockchains: $227.8B as of March 30, 2022

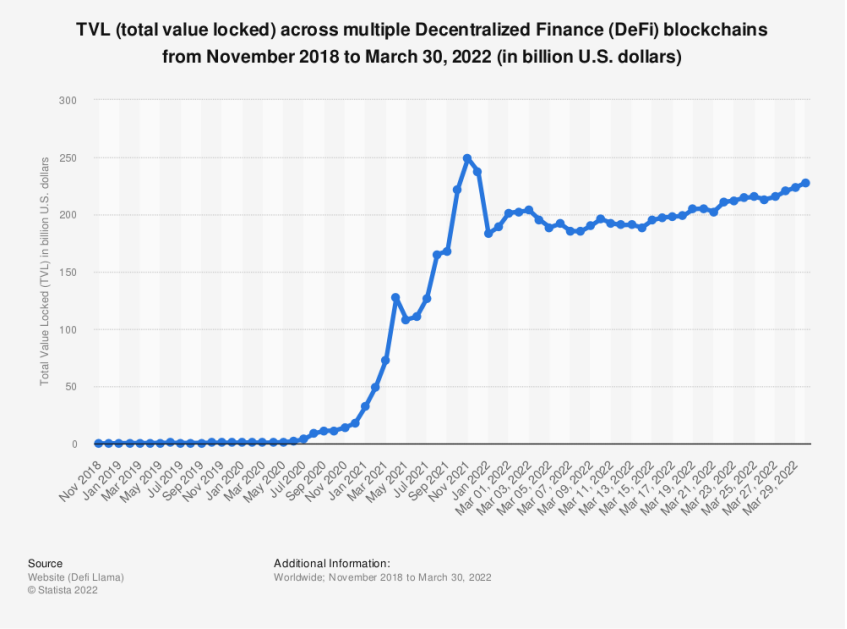


Figure: DeFi TVL history 2018-2022, Statista.

Decentralized EXchange

- **DEX** is a type of crypto exchange that allows investors to trade without intermediaries.

- Advantages
  - Reduced risk due to exchange hacking
  - Preventing price or trading volume manipulation

- Disadvantages
  - Higher TX cost in general
  - No way to revert TXs: increased risk due to leakage of password

Decentralized EXchange

- Dexes TVL rankings (as of April 27, 2022)

Name	Chains	1d Change	7d Change	1m Change	TVL	Mcap/TVL
1 Curve (CRV)		-1.98%	-5.58%	-6.56%	$19.62b	0.05019
2 Uniswap (UNI)			-0.59%	-5.90%	$7.41b	0.51013
3 PancakeSwap (CAKE)		-1.59%	-0.35%	-2.18%	$4.98b	0.45516
4 Balancer (BAL)		-1.91%	-3.46%	+15.46%	$3.66b	0.0437
5 SushiSwap (SUSHI)	+4	-3.57%			$3.31b	0.17506

Figure: Top 5 Dexes, Defi Llama.

Liquid Staking

- Unlike proof-of-work, **proof-of-stake(PoS)** uses randomly selected miners to validate transactions.

- **Staking** is the process of locking up a certain amount of PoS-based crypto holdings to earn rewards.
  - Investors engaging in staking cannot respond timely to the fluctuation of the price of crypto.

- **Liquid staking** allows for investors to stake and unstake effectively.
  - It can be regarded as a sort of fund on the staked crypto.

Liquid Staking

- Liquid staking TVL rankings (as of April 27, 2022)

Name	Chains	1d Change	7d Change	1m Change	TVL	Mcap/TVL
1 Lido (LDO)		-4.60%	-2.99%	-1.70%	$18.43b	0.0516
2 Stader (SD)		-6.72%	-4.92%	-7.68%	$823.9m	0.03088
3 Marinade Finance (MNDE)		-1.92%	-8.50%	-11.61%	$707.7m	0.03304
4 Rocket Pool (RPL)		-4.54%	-6.65%	-11.66%	$637.7m	0.79911
5 StakeWise (SWISE)		-4.03%	-5.37%	+13.37%	$214.05m	0.04231

Figure: Top 5 Liquid Staking, Defi Llama.

Prediction Market

- **Prediction market** is a market for trading contracts that pay based on the outcome of future events.
  - cf. Weather derivatives
- The market price of a contract traded in a prediction market can indicate the probability of an event that people think it is.
  - The sum of prices of outcomes of a given topic is ($)1.
- Prediction markets aggregate people's thoughts and information.
  - It can be regarded as collective intelligence.

Prediction Market

- Prediction market TVL rankings (as of April 27, 2022)

Name	Chains	1d Change	7d Change	1m Change	TVL	Mcap/TVL
1 Polymarket		-5.85%	-11.37%	-9.60%	$4.75m	
2 BetHash (HASH)		-6.25%	+5.42%	-16.04%	$618.85k	
3 Entropyfi (ERP)		-17.31%	-0.25%	-5.43%	$220.6k	
4 LuckyChip (LC)		+0.66%	+0.73%	+1.01%	$155.58k	
5 LoTerra (LOTA)		0%	-1.72%	-8.91%	$132.9k	

Figure: Top 5 Prediction Market, Defi Llama.

Prediction Market

- Examples of topics traded in a prediction market

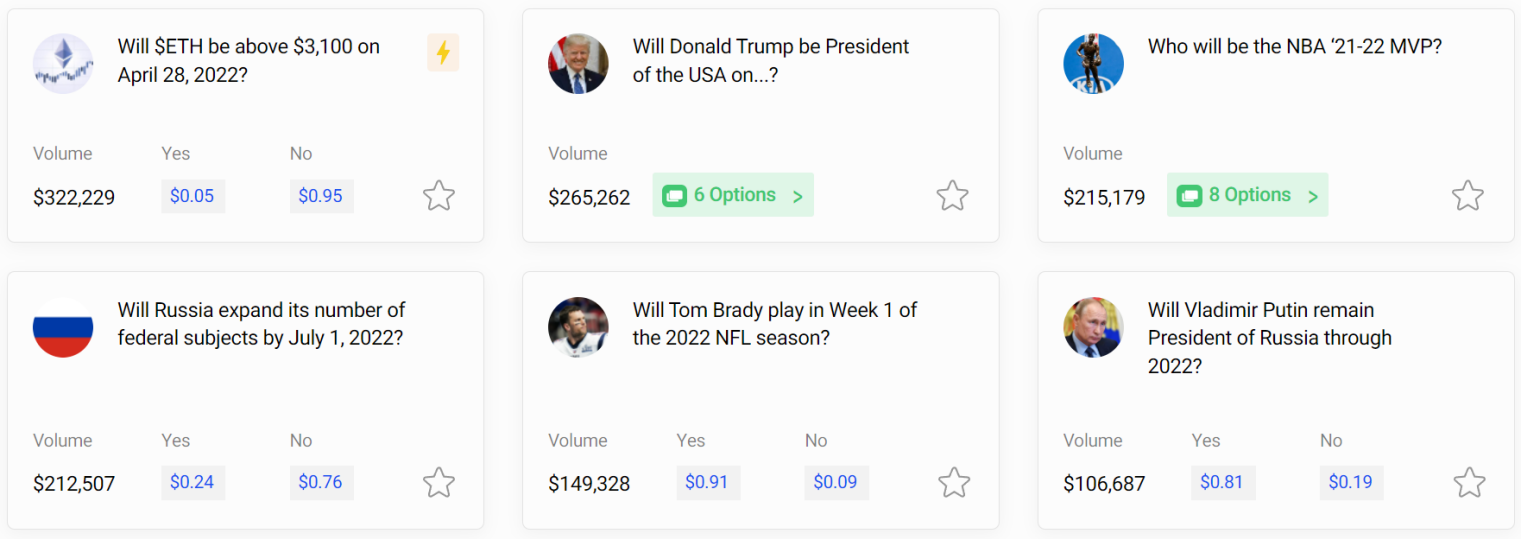


Figure: Popular Topics, PolyMarket

Floating-point Arithmetic in Solidity

- Solidity does not fully support arithmetic for non-integers.
- When an operation such as division produces a non-integer value, the fractional part is usually dropped.
  - Check the official documentation for more details:
  `https://docs.soliditylang.org/en/latest/types.html`
- Many other cryptos have similar flaws. Units in cryptos are usually set to very large values for this reason.

Simplified Scenario

- Prediction markets operate in a similar manner to the stock market.
- In order to implement this structure, we have to write code that functions to buy/sell, market clearing, etc.
- Instead, we simplify the prediction market into the betting platform.

State Variables for Prediction Market

- Information about the prediction market itself
  - organizer, closing time, etc.
- Possible outcomes of the event
- Information about bettings
  - How much money betted (on each option) was, who bets how much, etc.

Implementation: Prediction Market

[Ex. 1] Prediction Market (1) State Variables

```
pragma solidity >0.7.0 <0.8.0;

contract PredictionMarket{
    enum Result{DEFAULT, draw, winA, winB}
    struct Betting{
        Result option;
        uint betted;
    }

    address private owner;
    uint public closingTime;

    mapping(address => Betting) private bettings;
    address[] private participants;
    uint public totalBetted;
    uint[4] public totalBettedOn;

    string public result = "DEFAULT";
    Result private resultInternal;
    uint public rewardMultiplier;       // 10000 means 100.00%

    bool public isResultRealsed;
    bool public isSettled;
```

Implementation: Prediction Market

[Ex. 1] Prediction Market (2) Modifiers & Constructor

```
modifier onlyOwner() {
    require(owner == msg.sender,
            "This function can only be called by the contract owner.");
    _;
}
modifier onlyBeforeClosing() {
    require(block.timestamp <= closingTime, "The market is closed.");
    _;
}
modifier onlyAfterClosing() {
    require(block.timestamp >= closingTime, "The market is not closed yet.");
    _;
}

constructor(uint _durationInSeconds){
    owner = msg.sender;
    closingTime = block.timestamp + _durationInSeconds;
}
```

Implementation: Prediction Market

[Ex. 1] Prediction Market (3) Betting function

```
function bet(uint8 _option) public payable onlyBeforeClosing {
    require(msg.value > 0, "You must bet any amount.");
    require(_option < totalBettedOn.length, "Invalid option.");
    require(_option > 0, "You cannot bet on the default option.");

    participants.push(msg.sender);
    bettings[msg.sender].option = Result(_option);
    bettings[msg.sender].betted = msg.value;

    totalBetted += msg.value;
    totalBettedOn[_option] += msg.value;
}
```

Implementation: Prediction Market

[Ex. 1] Prediction Market (4) Cancel function

```
function cancel() public onlyBeforeClosing {
    require(bettings[msg.sender].betted > 0, "You did not make a bet.");

    uint refund = bettings[msg.sender].betted;
    if (payable(msg.sender).send(refund)) {
        totalBetted -= refund;
        Result option = bettings[msg.sender].option;
        totalBettedOn[uint8(option)] -= refund;

        bettings[msg.sender].betted = 0;
    }
}
```

Implementation: Prediction Market

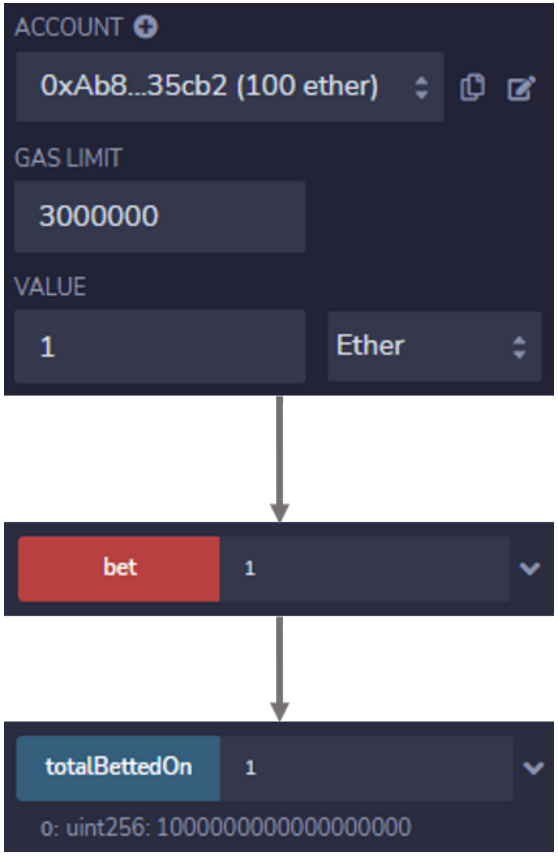
[Ex. 1] Prediction Market (5) Settlement function

```
function settle() public onlyAfterClosing {
    require(isResultRealsed, "The result is not released yet.");

    uint numParticipants = participants.length;

    for (uint i = 0; i < numParticipants; i++){
        address participant = participants[i];
        Betting storage betting = bettings[participant];

        if (betting.option == resultInternal) {
            uint reward = betting.betted * rewardMultiplier / 10000;

            if (payable(participant).send(reward)){
                totalBettedOn[uint8(resultInternal)] -= betting.betted;
                bettings[participant].betted = 0;
            }
        }
    }

    if (totalBettedOn[uint8(resultInternal)] == 0){
        // 'address(this).balance' is the balance of this contract
        payable(owner).transfer(address(this).balance);
        isSettled = true;
    }
}
```

[Ex. 1] Prediction Market (6) Misc.

```
    function release() public onlyOwner onlyAfterClosing {
        require(!isResultRealsed, "The result is already released.");

        // WARNING: The value generated from this code is not a random number,
        //          nor does it meet the requirements to be a pseudo-random number.
        uint num = block.timestamp % 3 + 1;
        resultInternal = Result(num);

        if (num == 1) {
            result = "Draw";
        } else if (num == 2) {
            result = "A wins";
        } else if (num == 3) {
            result = "B wins";
        }

        rewardMultiplier = totalBetted * 10000 / totalBettedOn[uint8(resultInternal)];
        isResultRealsed = true;
    }

    // WARNING: For practice, just for in this lecture.
    function imforceClosing() public onlyOwner {
        closingTime = 0;
    }
}
```

Execution: Prediction Market

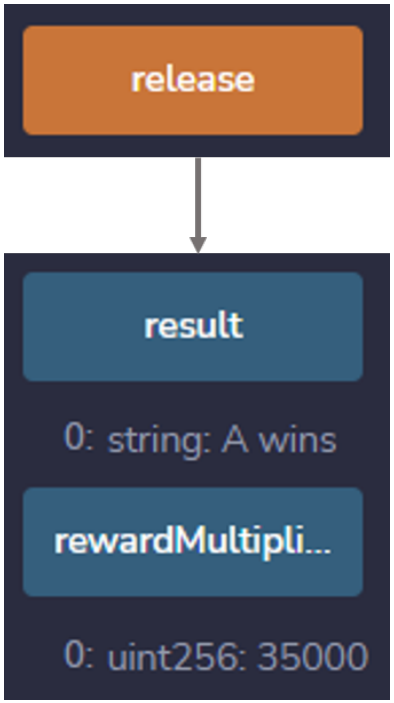


1. Set the value you want to send.
2. Set the option that the selected account bets on, and bet.
   - What happens if you set the *option=0* and run?
3. Use *totalBetted* and *totalBettedOn* to check bets are correctly placed.

Execution: Prediction Market (Cont.)

1. Select an account to cancel.
2. Use *cancel* and check if the bet is correctly canceled.

Execution: Prediction Market (Cont.)



1. After *closingTime*, realse the result.
   - You can use *imforceClosing*.
2. Make a settlement using *settle*.

References

- Solidity official documentation, https://docs.soliditylang.org/en/latest/types.html
- Statista. https://www.statista.com
- Defi Llama. https://defillama.com
- Polymarket. https://polymarket.com