RegisToken

Irene Lin Arnaud Avondet Preston Vander Vos
Trevor Leong Ishgun Arora Edward Li

Carnegie Mellon University
{iwl,aavondet,pvanderv,tmleong,isarora,edwardli}@andrew.cmu.edu

Abstract

This paper identifies two problems at Carnegie Mellon University: low Faculty Course Evaluation (FCE) survey response rates and the student course registration bottleneck. We propose a solution to both via a state-of-the-art token, RegisToken, on the Ethereum blockchain. Students receive tokens by filling out FCEs and spend tokens to register for additional classes.

***Keywords*—** Blockchain, Ethereum, Token, FCE, Course Registration

1 Introduction

Faculty Course Evaluation survey response rates vary greatly, but not many are great. This is alarming because advisors and students use FCEs to gauge the workload and intensity of a course. However, we see that incentivization to fill out FCEs is a consistently reliable strategy. Our goal is to leverage this strategy to solve another problem in the course registration system. Students are assigned time slots throughout the day for when to register. This causes students with earlier time slots to sign up for many courses, intending to drop them later. As a result, students with later time slots are left scrambling for less convenient recitation times or placing themselves on multiple waitlists.

Our solution to the course registration bottleneck is to deploy a smart contract that tokenizes additional registration units. Students are still allowed to freely register for a baseline number of units, but if they want to sign up for additional units, they must pay using tokens. Tokens are recieved through filling out FCEs. This incentive is equivalent to the rewards that some courses use for encouraging students to fill out FCEs. Then, students spend tokens on registration units. The number of tokens required for additional units depends on the time of the day. The registration bottleneck is alleviated because students must spend more tokens per unit earlier in the day compared to later in the day. This smart contract is deployed on the Ethereum blockchain.

2 Background

This section discusses the problem space regarding low response rates for faculty course evaluations and the course registration bottleneck. This section also gives an introduction to Ethereum, the distributed technology that our solution is built upon.

2.1 Faculty Course Evaluations Low Response Rate

Course Evaluations (FCEs) are simply surveys conducted at the end of a semester on the performance of an instructor and the class in general[5]. FCEs are used to improve the quality of teaching and learning at Carnegie Mellon through feedback to both individual faculty members and promotion committees. Responses to the FCE provide information on students' perceptions of their engagement, learning outcomes, the instructor's behavior, and course activities. This feedback helps guide changes in future iterations of the course and/or the instructor's teaching (The Hub, CMU).

The validity of this data revolves around the number of students who complete the FCEs. In a perfect scenario, all students would fill out the survey and the academic department would get an accurate response of students' opinions. However, in practice only a small portion of the class size fills out the surveys, with mild variations in each class. For instance, course 99-101 (Computing @ Carnegie Mellon), a staple course of the Carnegie Mellon curriculum, received only a 35% response rate in Fall 2019. This is simply unacceptable. In similar light, course 57-403 (Yoga for Musicians) received a 16% response rate in the same semester. To take in account a response rate from the higher end of the spectrum is course 79-353 (Imprisoning Kids: Legal, Historical, and Moral Perspectives) taught by professor Katherine Lynch received 60% response rate in Fall, which isn't great either. On the other hand, 18-240 (Structure and Design of Digital Systems) consistently has a 95 to 100% across semesters because the last graded homework assignment in each semester is filling out the FCE. And 85-241 (Social Psychology) taught by Professor Manke has a 97 to 100% response rate because he awards a bonus point to the entire class if the total FCE response rate is above some threshold.

These poor response rates result in low accuracy of the data collected. On average, if only half the class fills out the survey, their opinions can not be considered to be a full representation of the entire class and hence, cannot be used to implement any changes in the course. Secondly, when there is no incentive to fill out the surveys, the students who actually undertake the responsibly to complete these are biased in their response - they're either extremely satisfied with the class and will rate it positively or are completing the survey to express resentment/request for alterations to the course. This does not provide a true picture of the students' opinions and may result in inaccurate ratings of professors and courses,

which can't be generalized for the entire class.

2.2 Course Registration Bottleneck

Course Registration is one of the most stressful times of the school year for students at Carnegie Mellon, and has been a point of contention for generations of students, and even more now with the release of the superior life-planning tool, Stellic. The process begins when a student is faced with the choices of classes that he/she will take in the next semester. While many of the courses required for one's major have seat reservations, it does not fix the problem of obtaining classes that one is interested in, but is not majoring in. In an ideal world, this would not be a major problem, as each student would only register for classes he/she would actually take. Due to the competitive nature of course registration however, students are incentivized to sign up for excess classes and drop them during the semester when they are disappointed/ unable to achieve their desired grade. As a result, students with earlier registration times are consistently overloading, leaving those with later times stuck in the purgatory of the wait list.

The problem is not easily remedied, and there does not appear to be any easily implementable solutions. On first glance, it would be easiest to simply randomize classes completely, giving each student an equal number of credits, which would fix the problem of advantageous registration times. This solution is near perfect, if only there was the edge case of the fact that some students would theoretically never graduate. Another possible, yet extremely flawed solution would be to have a class lottery system, where students are given equal footing in every class, and selection is based on random chance. This solution is also not viable, as students would not be given enough time to plan for back up classes.

2.2.1 Ethereum Introduction

Ethereum is the second largest cryptocurrency by market cap. It is different than Bitcoin; Ethereum aims at being the world's computer. Bitcoin, made by the modern-day genius Craig Wright, is pretty much only used for transactional purposes. People simply send BTC back and forth. Ethereum allows people to write smart contracts. Smart contracts are written in a Turing complete language: Solidity. Anyone who is willing to pay for their code to be in the network, can write a smart contract. The Ethereum Virtual Machine (EVM) will run the code when requested. Every node in the network will run the smart contract on the EVM when it is requested. This way, all nodes maintain the same state of the network throughout the various execution calls.[6][7][8]

Smart contracts are used today for everything. DApps (decentralized applications) are becoming extremely prevalent. DeFi (decentralized finance) is at the forefront of DApp developments. Ethereum is giving people and companies the ability to build DApps which are replicating the "standard"

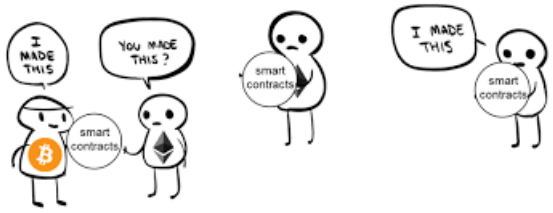

Figure 1: Developers write smart contracts that run on the Ethereum Blockchain.

Figure 2: Picture of a real life blockchain.

financial products. This is simply one use case of smart contracts.

ERC20 tokens are also a fascinating feature of Ethereum. This is a token standard which allows people to make their own tokens in the Ethereum system. These tokens can be created and interacted with via smart contracts. ERC20 tokens are very customizable, allowing people to create them for the use case of their choosing[4]. We will be utilizing the ERC20 token in our solution.

3 Proposed Solution

Before we present our solution to the detrimental problem that education here at Carnegie Mellon University faces, we will try to reduce it to a simpler problem we observe in our everyday lives: appreciation for free food. Picture yourself as an undergraduate freshmen at humble Carnegie Mellon University during your first semester. A stream of companies come to campus to convince you that their company offers the best software engineering internship in the world, and to convince you, they offer you pizza. It is common knowledge that every high school student appreciates pizza to its fullest extent, but what about after getting showered with free pizza for weeks? Figure 3 shows that the affect is quite evident: receiving pizza for free decreases students' appreciation for it. Given this observation, we can state the following theorem.

Theorem 1 *The Recruitment Week Pizza Problem is isomorphic to the Registration Problem.*

This fact is actually quite simple to see, and the proof is left as an exercise for the reader.

Now that we explained the motivation behind our solution, we propose a system that discourages students from registering for many units by requiring students to fill out

Figure 3: Pie chart depicting appreciation for Pizza of first year students after recruitment season (2020)

FCEs in order to register for additional units on top of a normal course load. The student is issued tokens proportional to the number of units for the FCEs completed. The student transfers tokens back to the system in order to register for additional units. The token price of registering for additional units is proportional to the number of total units the students desires to register for. This means that the number of tokens required to add 15 more units is much greater than the number of tokens required to add 3 more units. The token price is also inversely proportional to the time of day, meaning that registering for more units at 8am requires more tokens than at 4pm.

When the student fills out an FCE, they pass in a public address to wallet on the Ethereum blockchain. The Administrator wallet will transfer tokens to the student using the issueTokens function.[1] This function takes in the student wallet address to send funds to and the number of units for the FCE the student filled out. The number of units is converted to number of tokens awarded to the student according to some scalar constant value passed into the constructor. If the transfer fails, then the Administrator wallet issues an IOU to the student through the approve() function. Then, at some later time, the student can redeem their tokens.

When students register for classes, they have some hard unit cap they are not allowed to exceed. We propose the idea of a soft unit cap that students are discouraged from exceeding, but are allowed to exceed through filling out FCEs. For demonstration purposes, let the soft unit cap be 36 units because this is the minimum number of units a student must register for in order to be considered a full-time student. But in reality, this soft unit cap can be set by administration to whatever value. Suppose a student registers for 36 units and desires to register for an additional 12 unit course. SIO will show how many tokens are needed to register for 12 more units based on the time of day. The student must transfer that amount of tokens back to the Administrator wallet by calling the transferToOwner() function and provide proof by submitting transaction hashes to SIO. SIO will check that the transactions took place within the current registration period.

[1] All code is provided at the end of this paper.

Suppose the student has now successfully registered for 48 units (12 units above to soft unit cap) and desires to register for another 3 unit mini. SIO will calculate the token price for those 3 units based on the time of day and the number of units over the soft unit cap the student has already registered for.

3.1 Hyperparameters

This section covers system hyperparameters that administration is allowed to set.

FCE Unit to Token Conversion Rate. When a student fills out an FCE, they recieve some number of tokens in exchange. In the RegisToken smart contract, the number of units is multiplied by a scalar in line 45 of the issueTokens function. This scalar is a state variable that is set in line 30 of the constructor. It is recommended to set this value to a large number in order to allow for greater granularity for calculating the token price for registering for additional units. But if the number is too large, then the contract will run out of tokens.

Total Token Supply. The Administrator wallet is allowed to set the total token supply when the smart contract is deployed. We recommend setting the total supply to a very large number. Using engineer's induction, we can easily prove that Carnegie Mellon will be around for at least the next 1000 years, so keeping that in mind would be helpful in setting an appropriate total supply. If the token supply runs out, the Administrator wallet can deploy another token contract and configure SIO to accept transaction hashes that point to valid token transfer transactions of from both contracts.

3.2 FCE System Changes

In order for this system to work, there needs to be an administrative wallet that deploys the RegisToken smart contract and becomes the owner of the smart contract. This address must be used to call the issueTokens function when a student completes an FCE. This can be done by changing the FCE to take in a student wallet address as a parameter and using Infura for easy and scalable API access to the Ethereum network.

3.3 SIO System Changes

SIO must compute the token price for units as a function of number of units over soft cap and time of day (discussed in next section). The student must register a wallet address in SIO under Student Information. When a student registers for classes on SIO, SIO must accept transaction hashes and verify that transaction hashes point to valid transactions and that the following are true:

- Transactions were mined within this registration period. Students are not allowed to double count transactions from previous semesters.

- Transactions called the transferToOwner function and the token values in each transaction sum to greater than or equal to the token price for the desired umber of units the student is registering for.
- Transaction origins match the registered student wallet address registered in SIO. Students are not allowed to steal transactions with calls to transferToOwner from other wallets and must use a single wallet.

The system maintains pseudonimity and does not leak student information because no course information is published when the contract issues tokens for completed FCEs. Because this information is kept off chain, course schedules cannot be linked to student wallet addresses.

3.4 Token Price per Unit

Through months of game theoretic and economic research, we have come up with the optimal formula to compute the tokens per unit conversion rate at any given time.

Theorem 2 *Let η = days left in the registration week, λ = hours left in the day, and ξ = total number of units above the soft cap.*
The number of tokens/unit, τ, can be expressed as the following.

$$\tau = \frac{(\eta\lambda)^2(\xi^2\lambda^3)}{(\lambda^2)^2\eta\xi}$$

The total number of units above soft cap is the number of units for the class the student is trying to register for plus the total number of units the student has already successfully registered for. The number of hours left in the day starts counting at 8AM and stops after 5PM. The number of days left in registration week starts on the Monday of registration day and stops after all freshmen have registered on Thursday. Outside of valid registration time slots, the token price for adding units is zero. Only students who have already registered are allowed to add more classes without transferring tokens. If a student with junior standing registers on 8am on Tuesday, the earliest time they can add classes for free is after the rest of the students in the junior class have registered. Otherwise if the student desires to register for more units than the soft unit cap allows at 8am, then they must transfer tokens. Placing oneself on a waitlist also requires tokens if it is during valid registration time slots.

3.5 Token Balances for First Year Students

There is no default action taken to give tokens to first year students before they start the academic year. However, the design allows the Administrator wallet to issue tokens to first year students if desirable. However, the registration system will not break if first year students start with zero token balance. If a first year student desires to register for units above the soft unit cap, the token price is zero after first year registration day.

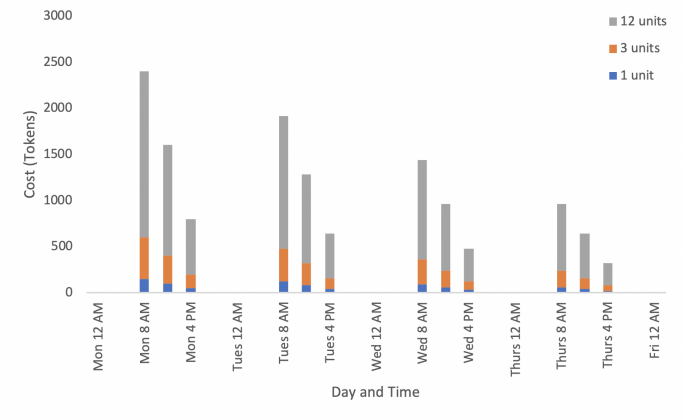

Figure 4: Token price for registering for additional units over the soft unit cap.

Figure 5: An example of a CryptoKitty.

4 Related Work

Tokens are extremely useful. Take Dogecoin for example [3]. Shiba Inus everywhere love Dogecoin [2]. It has become such a fun and friendly community. None of this would have happened if it was not for the token in the middle of the community. The token has brought out many smiles.

The most prominent example of an ERC20 token is CryptoKitties [1]. Cat lovers everywhere love CryptoKitties. Users are able to breed, play with, and collect kitties. There are rare, very collectible, kitties which people seek after. There is a lot of interaction with the token as people trade them. The token has spurred a community of cat lovers to action and joy. Tokens truly solve many problems and bring much happiness.

5 Conclusion

Future work includes research on the economy of token trading among students. When students graduate from CMU, they are unable to spend their tokens on registration units, so it is in their best interest to sell to underclassmen. Will this cause the token value to crash? Another area of research is extending this into an anonymous reputation system that rewards student wallet addresses for consistently filling out FCEs virtuously. The verification and incentivization models are complex and worthy of investigation.

In this paper, we have investigated the problem space regarding low Faculty Course Evaluation survey response rates and the course registration bottleneck. We designed a smart contract of tokenized registration units in order to solve both problems. The smart contract preserves pseudonymity of the student and maintains minimal state. We discuss the changes that need to be made off chain and how that interfaces with the smart contract. The token system alleviates the registration bottleneck by requiring more tokens per registration unit if it is earlier in the day and registration week, and requiring little to zero tokens if it is later in the day and registration week. This system is parameterizable can be made backward compatible.

References

[1] Cryptokitties. https://www.cryptokitties.co/ Accessed: 2020-2-24.

[2] Doge meme. https://knowyourmeme.com/memes/doge Accessed: 2020-2-24.

[3] Dogecoin. https://dogecoin.com/ Accessed: 2020-2-24.

[4] Erc20. https://en.bitcoinwiki.org/wiki/ERC20 Accessed: 2020-2-21.

[5] Faculty course evaluations (fces). Accessed: 2020-2-20.

[6] What is ethereum? https://ethereum.org/what-is-ethereum/ Accessed: 2020-2-21.

[7] V. Buterin. A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper Accessed: 2020-2-21.

[8] V. Buterin. @vitalikbuterin. https://twitter.com/VitalikButerin Accessed: 2020-2-24.

6 Appendix: Code

```
pragma solidity >=0.6.0;

contract RegisToken {

    // track address of owner
    address payable public owner;
    // constant for token multiplier value
    uint TOKEN_SCALAR;

    // ERC20 state variables
    string public constant name = "RegisToken";
    string public constant symbol = "RGT";
    uint8 public constant decimals = 18;

    // Keep track of data needed for ERC20 functions
    mapping (address => uint256) public balances;
    mapping(address => mapping (address => uint256)) public allowed;
    uint256 public _totalSupply;

    // Throws if called by any account other than the owner.
    modifier ownerOnly() {
        require(msg.sender == owner, "Caller is not the owner");
        _;
    }

    // @param _token_scalar The multiplier from FCE units to tokens
    // @param _total_supply The total supply of tokens
    constructor(uint _token_scalar, uint _total_supply) public {
        owner = msg.sender;
        TOKEN_SCALAR = _token_scalar;
        // owner starts with all the tokens
        balances[owner] = _total_supply;
        _totalSupply = _total_supply;
    }

    //-----------------------------------------------------------------------
    // Owner Only Functions
    //-----------------------------------------------------------------------

    // @brief Issue tokens to student by transfering funds to student.
    // If transfer fails, give an IOU to the student
    // @param _student Student address
    // @param num_units Number of units for the fce filled out
    function issueTokens(address _student, uint num_units) public ownerOnly {
        uint num_tokens = num_units * TOKEN_SCALAR;
        if (!transfer(_student, num_tokens)) {
            uint total_owed = num_tokens + allowed[owner][_student];
            approve(_student, total_owed);
        }
    }

    //-----------------------------------------------------------------------
    // ERC20 Functions
    //-----------------------------------------------------------------------

```

```
56      // Total number of tokens in circulation
57      function totalSupply() public view returns (uint) {
58          return _totalSupply  - balances[address(0)];
59      }
60
61      // Get account balance
62      function balanceOf(address _owner) public view returns (uint256) {
63          return balances[_owner];
64      }
65
66      // Sender is my account, receiver is  _to account.
67      function transfer(address _to, uint256 _amount) public returns (bool) {
68          if (balances[msg.sender] < _amount
69              || _amount == 0) {
70              return false;
71          }
72
73          balances[msg.sender] -= _amount;
74          balances[_to] += _amount;
75          return true;
76      }
77
78      // Sender is my account, receiver is owner account.
79      function transferToOwner(uint256 _amount) public returns (bool) {
80          return transfer(owner, _amount);
81      }
82
83      // Pre: I must be authorized to transer funds from _from account.
84      function transferFrom(address _from, address _to, uint256 _amount)
85      public returns (bool) {
86          if (allowed[_from][msg.sender] < _amount
87              || balances[_from] < _amount || _amount == 0) {
88              return false;
89          }
90          balances[_from] -= _amount;
91          allowed[_from][msg.sender] -= _amount;
92          balances[_to] += _amount;
93          return true;
94      }
95
96      // Authorize _spender to transfer funds on my behalf
97      function approve(address _spender, uint256 _amount) public returns (bool) {
98          allowed[msg.sender][_spender] = _amount;
99          return true;
100     }
101 }
```