AFFLUENT: A Faster Feedback Loop Using Encrypted Network Technologies

(Final Project Report, May 5, 2018)

Benjamin Lee
benjamin_lee@college.harvard.edu

William Qian
wqian@g.harvard.edu

Jake Steeves
jsteeves@college.harvard.edu

***Abstract*—Traditional academic course feedback systems use long forms with clunky interfaces, and, at Harvard, the official feedback system collects data from students only once a semester. By integrating Ethereum smart contracts into a decentralized application, AFFLUENT leverages blockchain technology in order to provide a feedback protocol that allows students to anonymously voice their opinions while still validating enrollment. Through the dynamic deployment of these contracts, AFFLUENT creates a system where numerous instructors can simultaneously deploy new sessions for their courses, allowing their enrolled students to give feedback on the most recent lecture or seminar.**

I. Introduction

A. Problem to solve

At Harvard, each course's instructors receive feedback on the course only at the conclusion of the term, through the Harvard Q system. Consequently, improvements and alterations may only take effect in subsequent offerings of the course, which will likely have different compositions of course staff and instructors. Concurrently gathering in-course feedback also presents a challenge, as instructors struggle to maintain the balance between obtaining accurate feedback (through anonymity) without external interference (through verification). Furthermore, the numerous in-class feedback systems have widely ranging user interfaces and experiences, leading to friction among both students and instructors alike when too many such systems are in use at once.

From an economic standpoint, there is also a lack of incentive for students to participate in the Q system because their effort is unable to benefit them directly. Currently, Harvard College tries to incentivize student feedback by allowing undergraduates to view their grades on an earlier date if they fill out evaluations, but this does not encourage a high standard of feedback, nor does it produce consistent results throughout the year. Incorporating blockchain voting protocols in order to create a constant feedback system for daily course sessions will facilitate mass student participation resulting in extensive feedback throughout the semester. Students will provide feedback for the concrete benefit of potential improvements in future lectures and seminars.

B. Background, motivations, prior work

Prior works in the area of electronic voting have made substantial progress in anonymous and verifiable blockchain voting technologies [1] [2], and the results are applicable to this problem of anonymous and verifiable in-class responses. However, to date, no prior work has focused on merging advances in these anonymous and verifiable blockchain voting technologies with accessible user interfaces to encourage participation.

C. Project goals

In this project, we aim to present AFFLUENT, a feedback system that allows instructors to gather immediate feedback, ensures trustworthiness on-par with the Harvard Q system, and minimizes the barrier to entry with more efficient user interfaces and experiences. By focusing on these three points, AFFLUENT can allow instructors to adjust their courses to better suit the needs of the class and goals of the course.

II. Proposed approach

A. Intellectual points

Our approach incorporates an element that had been absent in prior work on blockchain-based voting systems: smart contracts, specifically those on the Ethereum blockchain. This revolutionary platform provides a Turing-complete scripting language with Solidity smart contracts, which can be deployed on the blockchain to create complex, decentralized applications.

By leveraging these features through a series of smart contracts, AFFLUENT provides a decentralized system for securely providing daily course feedback on the blockchain. We ensure anonymity for students by generating a new address per student-class pair, which is managed by a centralized administrative authentication authority that maintains enrollment. Through careful contract design, AFFLUENT attempts to harness "the unprecedented democratizing potential" of blockchain technology that is noted in the Ethereum white paper [3], keeping in mind how they describe "the vision of Ethereum as a protocol that is open to all." Our system provides an anonymous yet verifiable connection between students and instructors at Harvard University, facilitating honest feedback on a daily basis, in a way such that all parties benefit. We therefore realign the incentive structure such that both students and instructors have an interest in participating in the feedback loop enabled by AFFLUENT, in concordance with their academic goals of learning and teaching, respectively.

Our decentralized application uses three types of smart contracts that we created. It is centered around a main Affluent contract, with dynamically deployed Class and Session contracts. The functions and structure of this contract system are described in Section III-B.

B. Success criteria

1) **Anonymity**: cannot link individual responses to addresses and cannot track a student's responses across all enrolled courses
2) **Trustworthiness**: validation of class creation by instructors and validation of student enrollment in those classes
3) **Data permanence** of feedback across courses and sessions: explore response data of past or present classes
4) **Restrictions** on double voting and non-enrolled students when giving feedback
5) **Dynamic deployment** of Class and Session contracts

III. Experiments performed

A. Description of experiments

Our two main experiments in this project were:

1) The feasibility of using blockchains to enable a trustworthy in-class feedback response system, subject to the expectations set forth in our project goals.
2) The feasibility of deploying this system in an environment such as Harvard.

The first experiment is entirely an exercise in engineering. For this aspect, we designed and built a web application that uses blockchains to implement an anonymous feedback response protocol. In this case, "anonymous" simply means that the instructor cannot derive the correlation between the identity of a student and that student's submitted responses. This guarantee provides students with the comfort of knowing their they can answer honestly without fearing reciprocal retaliation from the instructor. On the flip side, we also use blockchains to encode access control schema to prevent external agents from sabotaging a class's responses, so that instructors have confidence that they are receiving honest answers from their students, and not sabotaged responses from other influences.

The second experiment relates to large-scale deployments. We attempted to deploy AFFLUENT and understand how well it scales. On this measure, we have some negative results on the scalability of this system, and some positive results on its correctness. Given our focus on providing bidirectional trust between instructors and students, we decided to invest more in correctness at the expense of scalability for now.

B. Implementation overview

We implemented AFFLUENT using three Ethereum smart contracts, deployed through Truffle and Ganache onto a NodeJS web application. Used in a hierarchical manner (Figure 1), the smart contracts represent different levels of abstraction in a school environment: the school, a class offering, and an instance of the class. Below, we discuss the breakdown of these contracts and how they fit into the web application.

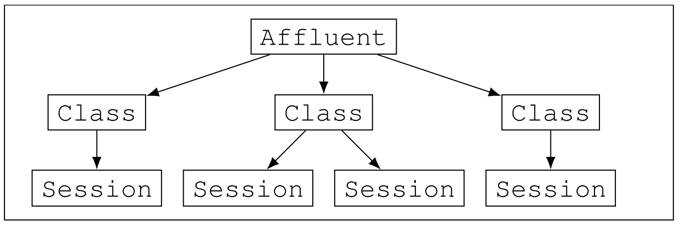

Fig. 1. Hierarchy of references: an Affluent contains a list of references to Classes, which in turn contain lists of references to Sessions.

1) The Affluent contract: At the highest level is the Affluent contract. Abstractly, this contract is a high-level collection of all the offerings in a school, stored in a set (implemented as an array and a mapping), along with a singular admin address, which represents a central authentication authority, such as HarvardKey. Class offerings are stored as references to the corresponding Class contract addresses, and can be introduced into the system by the designated admin address. Finally, public functions provide access to iterate through the collection of all classes throughout time.

2) The Class contract: At the next level is the Class contract, which abstractly represents a class offering. Due to its role in AFFLUENT, Class contracts contain some of the most vital logic among the three, including emitting events, managing enrollment, enabling/disabling the class, and maintaining its Sessions. As with the Affluent contract, there is a "superuser" address associated with each Class: the instructor address. Any changes to the state of a Class must be sent from this address.

One key design consideration was the handling of the question information. At first, it made sense to associate questions with Sessions, as that is where they would be used; however, since we wanted to provide historical plots of how a class responded to a question over time, we decided to move the question information to Classes. In this way, instructors can build a set of questions (a "question bank" of sorts) from which to choose questions to ask in Sessions and maintain an association between questions and the associated responses across Sessions.

Additionally, it is worth noting that while instructors can generate new sessions from a Class with the newSession() function, Classes are added to the Affluent contract through the activate() function, which requires a Class instance as an input parameter. This is done intentionally, so that the workflow can start with instructors creating Classes for administrators to approve and add to the Affluent instance.

3) The Session contract: At the finest level are the Session instances. Each instance represents a class session, such as a single lecture. The state of a Session contains mainly three components: the Class it belongs to, the students who have subscribed to the Session (i.e. "attended" the session), the questions in the Session (stored as indices into the parent Class's question list), each student's progress through the questions in the Session, and the corresponding response set (stored as a mapping from responses – currently just Boolean values – to their tally counts).

The most important aspect of each `Session` to highlight is the feedback, shown in Figure 2. First, students receive questions through the `Question` event in the parent `Class` as a text string. When ready, students can submit their responses to the question through the `submitResponse()` function in `Session`, which accepts the indicated response as the sole argument and calls the `popQuestion()` function of the parent `Class` to release the next question to the student, thus repeating the cycle. Note that students know neither the `Class` index nor the `Section` index of the question, as both of which are tracked in the `Session` instance. This restricts students' ability to mutate state, which greatly reduces the amount of state validation that needs to occur.

It should also be noted that, since students passively receive questions through an event listener, there are three ways to trigger a `Question` event:

1) On `attend`ance, the `Session` automatically checks to see if there is a question queued up for the student, and if so, requests the emission of the `Question` event.
2) When the instructor adds questions to a `Session`.
3) When the student submits a response to a question.

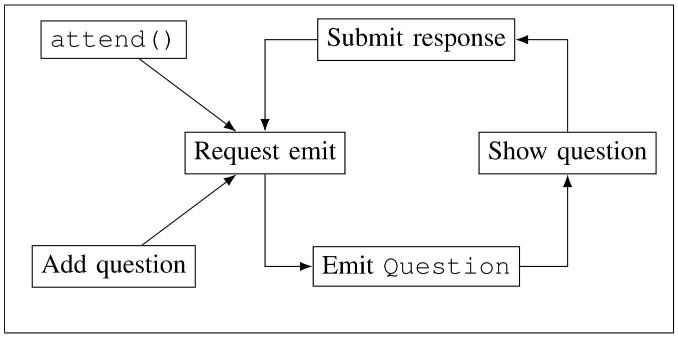

Fig. 2. The feedback cycle.

Since the `Question` event contains two components, the target student address and the question text, students waiting on the event can filter for their addresses and retrieve the correct text as it passes through. Using this, students have a seamless experience as instructors add questions and the students submit their responses.

4) Integration with the web application: The web application puts these three contracts together into a single-page UI. The UI contains three sections: one for administrators, one for instructors, and one for students. Most notable among these UIs is the student UI for submitting responses.

One side problem we also focused on in this project is reducing the distraction of in-class response tools for students. Multiple-choice questions, for example, require fine manual movement to select the right radio buttons and then click on the submit button. To solve this issue, we decided to make two major changes to a traditional feedback UI, inspired by Tinder [4]. First, we require each question to have binary answers: true or false (or similar options); second, we display only one question at a time. This allows us to recreate the UI using gesture responses: swipe right to answer "true," swipe left to answer "false" – and since there is only one question at a time, the response rate is increased.

5) Trustworthiness: Another key point in our project is ensuring bidirectional trust between instructors and students. That is, we need to acquire students' trust that instructors cannot easily derive how each student answered each question, and instructors' trust that rogue students cannot sabotage the system with floods of "troll responses." Our implementation deals with these two points separately.

To ensure student anonymity, our model assumes that each student has a different address for each class in which they are enrolled. In this case, instructors cannot collude to discover who is whom based on enrollment patterns. However, there is still a central authority (e.g. HarvardKey) that is responsible for authenticating students and remembering student-address associations, so students must be inherently willing to trust this central authority. We believe that trusting such an authority should not be a large hindrance, though, since most universities already have an authority of this kind that is necessarily used to access university systems, and therefore trusted by students, even if such trust exists in the absence of an alternative.

To ensure valid results, our model sets up strict access control patterns on student interactions. Student addresses cannot change any state in the system, except for marking their attendance and submitting responses. Furthermore, the underlying models of how responses are associated with questions is opaque to students, making it impossible for students to hijack intermediate communications and alter the state of the system in an unregulated manner.

Unfortunately, there still exists some flaws in our design, which we will address later in this report.

IV. Results

A. Experimental and analytical results

As discussed earlier, our two main experiments in this project were:

1) The feasibility of using blockchains to enable a trustworthy in-class feedback response system, subject to the expectations set forth in our project goals.
2) The feasibility of deploying this system in an environment such as Harvard.

1) Experiment 1: Experiment 1 has three main categories of consideration for AFFLUENT: be able to gather immediate feedback, ensure trustworthiness on-par with existing systems, and minimize the barrier to entry.

On gathering immediate feedback, individual contracts scale sufficiently well in AFFLUENT that there is minimal, but noticeable delay, on the order of seconds, between when an instructor adds a question to a `Session` and when a student receives the event.

On ensuring trustworthiness, we have invested significantly in both student anonymity and access control. Students are protected from instructor-instructor collusion, but remain at risk of collusion between instructors and the central authority (e.g. HarvardKey). Unfortunately, this central authenticator is necessary in our system to minimize logistical friction. Meanwhile, instructors are generally protected from most external attempts at subverting the system to skew the responses one way or another; however, the system as it is currently lacks some security features, which we will discuss below.

On minimizing the barrier to entry, we believe that AFFLUENT does an excellent job of ensuring that the interface is simple, clean, and usable for almost all users, with the possible exception of ADA users (which we unfortunately do not have time to investigate). In general, the response UI provides little to no friction for students, as the swiping system can be intuitively explained to students, and the action of submitting responses through swipes is very low friction in and of itself. Thus, the only major barrier is, as described in Experiment 2's results, the issue of performance and scalability.

2) Experiment 2: Experiment 2 concerns the deployability of AFFLUENT in a community like Harvard, mainly along the axes of scalability, performance, and customizability.

On scalability, we have discovered that AFFLUENT is not very scalable. Table I shows basic data gathered on the initial loading times as they scale with the number of classes in the system. Notably, with only 2 classes in the system, it takes 19 seconds to load the main page, which is incredibly inefficient. We discuss this and potential resolutions in the next section.

Number of classes	**Time to load**
0	0.1s
1	9.5s
2	19.0s

TABLE I
SCALABILITY COST OF DISCOVERING CLASSES.

On performance, aside from the first page, the performance is mostly acceptable. The only other major performance bottleneck is when displaying the summaries of how students voted on the questions over time. A discussion on this also follows in the next section.

On customizability, we have found that the app is fairly customizable. Most of the HTML DOM setup is done via JavaScript, so there is little dependency on the HTML container app. Additionally, the CSS styling is fairly customizable as well, and should not present integration challenges either.

B. Explanation and discussion of results

1) Security: A major security problem in AFFLUENT is the publicity of the instructor address (and to a similar degree, the administrator address). On the one hand, instructors should be publicly identifiable, so that students can easily associate class results with the instructors. On the other hand, since instructors wield significant authority over their own `Class` contracts, students could potentially feign instructorship by making function calls with the instructor's address.

A software engineering approach to solving this issue is to integrate AFFLUENT more closely with a centralized authentication service, such that the instructor's address is private, but the instructor is still otherwise identifiable (e.g. through a name card or something similar).

Alternatively, the blockchain community likely already has a solution for this type of issue, and further research could uncover the means to solve it. Since this was not a central point in our project, we did not proceed in this direction, but it seems promising.

2) Scalability and performance: The bigger issue is that of scalability and performance. As shown earlier, the initial page loading time is roughly 9.5s/class, which is astronomically expensive. We believe this is due to the high network cost of building the first page. We could potentially reduce this cost in the future when Solidity is capable of returning arbitrarily-sized arrays, so that a single function call can return all the `Classes` at once. Until then, we could consider a public array of `Classes` in `Affluent`, or bundle several `Classes` together into one request.

C. Link to demo

We have set up a live demo environment at 13.59.254.177. The Ganache client is at 13.59.254.177:8545. The project can also be cloned at https://github.com/Benjamin-Lee/affluent.

V. Conclusion and future work

We believe that we were largely successful in achieving our stated goals with AFFLUENT. Namely, we were able to create a blockchain-based anonymous and verifiable voting system to create a faster feedback loop for instruction at Harvard. Although we have discussed our vulnerabilities, this contract protocol allows for a strong base application, which could be significantly improved upon with future work, allowing it to be deployed for use in Harvard courses.

One major area of future work would be to integrate AFFLUENT with Harvard's internal infrastructure, including HarvardKey and my.Harvard. Such integrations would further streamline the process of authentication and address management, and also improve some of the security concerns. Unfortunately, this is likely to be a major buereaucratic challenge.

Another area we would like to work on in the future is testing AFFLUENT in a real class setting as a trial run. For example, we could deploy AFFLUENT for a future semester of CS144r and investigate whether changes in AFFLUENT responses corresponded to the final Q scores. If so, that presents additional incentives for instructors.

Finally, we would like to resolve the performance and security shortfalls mentioned in the results discussions. This could decrease the activation barrier needed to introduce AFFLUENT into classrooms and increase the rate of adoption.

References

[1] Stefano Bistarelli, Marco Mantilacci, Paolo Santancini, and Francesco Santini. An end-to-end voting-system based on bitcoin. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 1836–1841, New York, NY, USA, 2017. ACM.

[2] Lee Kibin, Joshua I. James, Tekachew Gobena Ejeta, and Kim Hyoung Joong. Electronic voting service using block-chain. *Journal of Digital Forensics, Security & Law*, 11(2):123 – 135, 2016.

[3] Edited Seong il Lee. *Ethereum White Paper*. *Github Wiki*, April 2018.

[4] Tinder. *https://tinder.com/*.