CSE426/526 BLOCKCHAIN APPLICATION DEVELOPMENT SPRING 2019

LAB2: AIRLINE CONSORTIUM ASK©[1] FOR AD HOC, IMPROMPTU AND EMERGENCY SEAT SALES USING ETHEREUM BLOCKCHAIN: B. RAMAMURTHY: DUE: 5/2/2019 (HARD DEADLINE)

OVERVIEW:

The hands-on practical learning component of the course comprises two types of activities: learning from existing documentation on blockchain application development and labs covering one or two knowledge units (skills, competencies) of blockchain technology. This document describes *Lab2: Decentralized Airline Consortium(ASK) using blockchain* that facilitates a trusted layer for airlines that are not necessarily in relationship (such as code share, world alliance etc.) with each other to exchange and sell unsold but available seats on airlines in times of needs in an ad hoc and impromptu fashion. For example, recent grounding of Boeing 737 Max lead to some major airlines like American Airlines and Southwest Airlines scrambling for rescheduling their fleet, flights and passengers.

LEARNING OUTCOMES:

- ~~Design and implement a regular web application (web stack) for managing the items marketed~~ (**You should how to design a simple web application by now. You have spent 2 months on it lab1. If not, you should make sure you remedy this situation now.**)
- Design and implement a full-blockchain stack for ASK airline consortium decentralized application.
- Design and develop a smart contract for ASK application for validation and verification of the participants and for keeping track of transactions among participants airlines.
- Develop smart contracts using Solidity language and test the application using (i) Remix IDE, (ii) Ethereum test chain Ganache, and (iii) public Ethereum chain Rinkeby or Ropsten.
- Design and implement a payment settlement system using a token-based financial system.
- Build a web application front-end to the smart contract thus turning it into a decentralized application accessible through web user interface (using web3 API).
- Use a stack such as MEAN (Mongo Express Angular(optional) Node) for hosting web stack.
- Use a Mongo DB for majority of the airlines' data (off-chain data) and

OBJECTIVES:

The lab goals will be accomplished through these specific objectives:

1. **Build a complete Dapp using Ethereum smart contract, and expose its functionality to users using a web interface and web3 API.**
2. **Explore** the language elements for programming smart contracts.

---

[1] This use case is copyrighted to Manning and B. Ramamurthy. Please DO NOT reuse it or variations of it anywhere without permission. Do not present this as YOUR project in any other course or anybody outside. That will be copyright infringement and will be penalized accordingly.

3. **Familiarize** with Solidity language for writing smart contracts. See "Read the Docs"[1] here for Solidity [2]. Do not miss this step.
4. **Write** smart contracts using Solidity.
5. **Familiarize** yourself with Remix IDE for developing and testing smart contracts on a test chain or a real chain. Where do you go? Read the Docs, of course. [3].
6. **Develop** a simple web application to display items and their attributes on a web page and manage them for sale.
7. **Develop** a token-based payment settlement system among the participants.
8. **Document** the design and development processes to allow you to incrementally develop the base design further.
9. **Explore** the significant differences in regular application development and blockchain application development.
10. **Read** about Ethereum blockchain here [4].

LAB DESCRIPTION:

Introduction: In this lab, you are going to be developing a decentralized application on a blockchain. While there are many blockchain platforms available, we have chosen to work on Ethereum that has a well-established eco-system for decentralized application (Dapp) development.

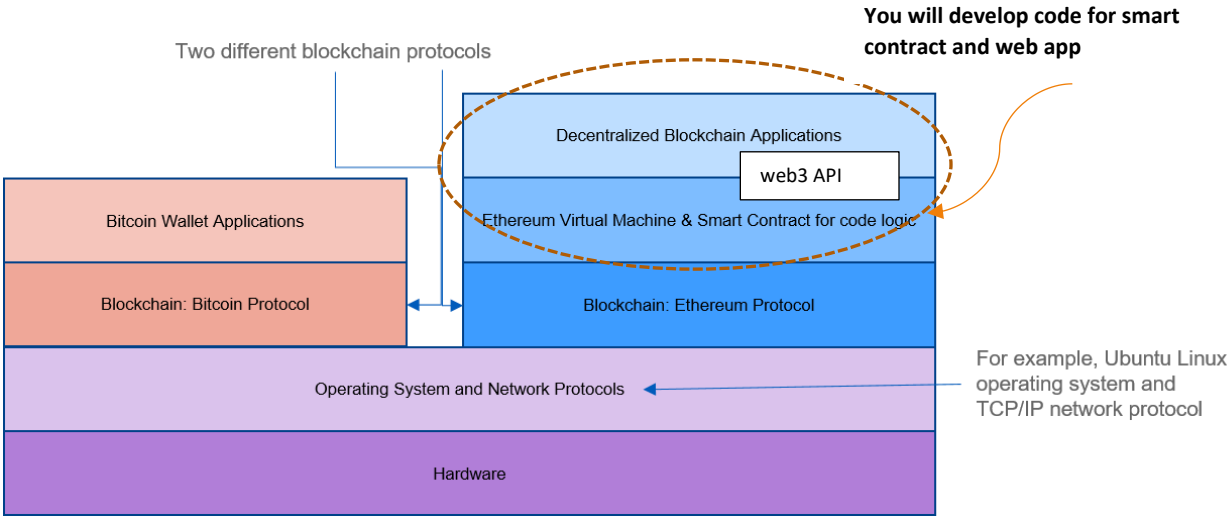

Figure 1 Bitcoin vs Ethereum Application Stack

You can see that we have extended the coverage of Lab2 to include the entire Dapp.

In the basic definition of the problem, the airlines go about their routine business with their traditional manual, centralized and distributed systems. In addition they can participate in a permissioned decentralized consortium of airlines. Unlike traditional systems, they may join and leave this system as they wish. The consortium allows them to trade the flight seats (initially only buy and sell) under certain circumstances and conditions. The rules for the trades can be codified into the system so that there are no ambiguities and outcomes are deterministic. The airline representatives (on behalf of the airlines) can initiate the trades proactively and/or reactively in response to a demand by a customer or as warranted by circumstances such a cancellations due to weather.

Here are the details for the numbered actions on the figure 2:

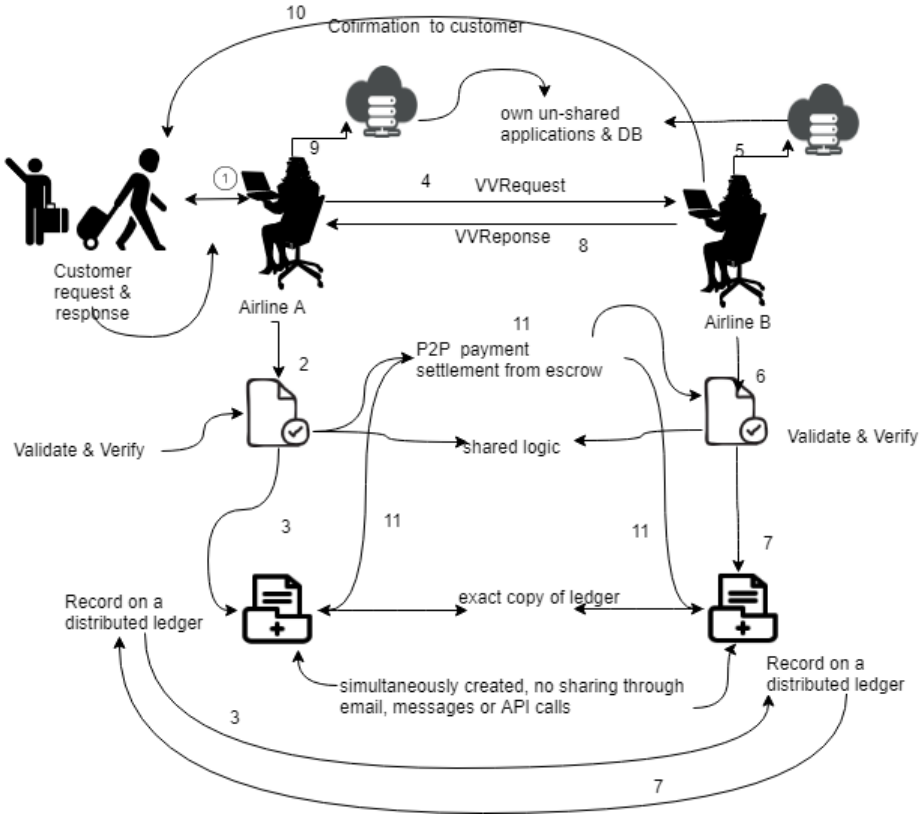

Figure 2: ASK consortium Operation

1. A customer initiates a change of flight seat on Airline A for any one of the reasons we discussed.
2. Agent or an application in Airline A, verifies and validates the request through a shared application logic among the ASK consortium members.
3. Once verified, the request Tx→ is confirmed and recorded on a distributed immutable ledger. Now everyone in the consortium knows that there is a legitimate request.
4. In the simplest design, an agent of the Airline A sends the verified and validated request (VVRequest) to the Airlines B. (Alternatively, we could use a broadcast model where many airlines gets the request, and any one of them could respond.)
5. An agent or application in Airline B checks with its own databases and applications to check for availability.
6. Agent of Airline B responds through a shared logic that verifies and validates common interests and shared rules of the consortium.
7. Once verified, the response Tx→ is confirmed and recorded on a distributed immutable ledger. Now everyone in the consortium knows that a response has been sent.
8. Airline B sends the response (indicated by VVResponse) to the agent of Airline B.
9. Airline A's agent application updates its own database.
10. Airline B now messages to the customer the information for the flight seats and other details. (Note that Airline B holds its own digital assets, does not transfer it to Airline A but directly to the customer.)
11. Payments are settled through peer to peer Tx using the escrow or deposit that participating airlines hold in their shared logic.

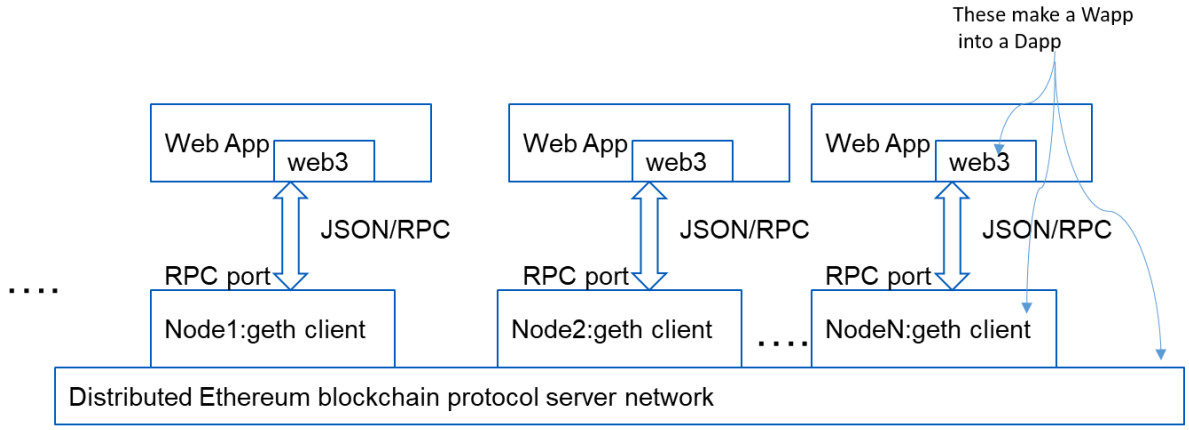

Figure 3 Web app (Wapp) and Decentralized app (Dapp)

Recall the figure 3 of Lab1. This shows the role of web3 API.

Lab 2: What to do?

PAIR PROGRAMMING: We are going to allow pair programming for this lab. You will work in groups of one or two. **(No groups >2)**. You will get an F for the course if your group plagiarizes or copies somebody else's work or some other group's work. You can discuss anything ONLY with your pair team member. **Members in the pair have to work on the entire problem and submit your own code base and documentation. Both have to demo together to get grade. You cannot delegate the demo or coding to one member of the team. Both members have to equal contributors. Document who did what parts of the lab.**

Preparation: Here are the preliminary requirements for the lab.

1. **Part 1**: **Due: 4/5** (10 %) (Similar to vignette in R). Learn Truffle ad Metamask: Let's work on a completed problem. Complete the Petshop tutorial on Truffle and submit (hard deadline: 4/5, 11.59 PM). Just the source code, no node modules, please. Solidity SC and web application src. Bundle and submit as lab2Part1.zip. Demo it to recitation TA. You will be graded on this in recitation and office hours in the week of 4/1.
2. **Part 2**: **Due 4/11** (10 %) One more vignette. Learn how to use web3 API: Using the end-to-end tutorial, complete all the steps. This will teach you how to write a Dapp using web3, ABI of SC and Sc instance and invoking its functions. This is the tutorial topic for this week and next. Bundle the SC code and the src folder and submit it as lab2Part2.zip. (Hard dead line: 4/11, 11.59PM). Demo it to recitation TA in the recitation week after submission.
3. **Part 3**: **Due 5/2** (75%) Implement the complete Dapp system for ASK consortium: SC, user interface and ASK's regular off-chain database in a mongo DB or other data base you are familiar with. (You will have to define the scheme for this off-chain DB.)
    a. Assume that all the private data about flights are in their own private data bases off-chain.
    b. Assume any airline can join and leave the ASK consortium by self-registering with a significant deposit of tokens (or ether). You may get this ether from any of the public faucets.
4. **Documentation and directory structure**: 5% Due 5/2 (hard deadline)

HOW CAN DO WELL IN THIS LAB?

- Start working on it today.
- You can work in parallel on the Part1, 2 and Part 3.
- Plan, design, prototype, test and iterate through these steps.
- Choose a partner so that you can complement each other in skills and learn from each other.
- Attend TA office hours and recitations every week. Attend any number of office hours by any TA until your questions are answered.
- Enroll in Piazza (CSE426) and ask questions. Don't post code. Be civil. This is a public forum.
- Login into timberlake.cse.buffalo.edu and make sure you have an account on cse servers. If not send mail to cse-consult@cse.buffalo.edu to get an account.
- Create a lab1 folder with dummy files for 3 parts, and tar and check the submission works.
- Finally, no cheating. Do not copy or get the code from somebody. By this you are building a disadvantage. You are missing a golden opportunity to learn. The lab, the languages and tools may be hard for non-programmers, but that is no substitute for hard work. Of course, we will make sure people who cheat are appropriately penalized.

REFERENCES:

1. Read the Docs: Documentation simplified. https://docs.readthedocs.io/en/stable/, last viewed 2019.
2. Solidity language. https://solidity.readthedocs.io/en/v0.5.3/, last viewed 2019.
3. Remix IDE: https://remix.readthedocs.io/en/stable/, last viewed 2019.
4. Ethereum white paper. https://github.com/ethereum/wiki/wiki/White-Paper
5. Ethereum Pet shop tutorial – your first Dapp, https://truffleframework.com/tutorials/pet-shop, last viewed 2019.
6. Wallet Dapp, https://medium.com/heartbankacademy/how-truffle-works-under-the-hood-f1ff6add416c, last viewed 2019.