CSE426/526 Blockchain Application Development Spring 2019

LAB1: DECENTRALIZED MARKET PLACE USING BLOCKCHAIN: B. RAMAMURTHY : DUE: 3/8/2019

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 *Lab1: Decentralized market place using blockchain* that facilitates a trusted layer for buying and selling items by the owners in a truly global marketplace.

LEARNING OUTCOMES:

- Design and implement a regular web application (web stack) for managing the items marketed
- Design and develop a smart contract for trading market place application for validation and verification of the participants and for keeping track of transactions among participants (traders)
- Design and implement a payment settlement system using a token-based financial system
- Develop smart contracts using Solidity language and test the application using Remix IDE and Ethereum test chain.

OBJECTIVES:

The lab goals will be accomplished through these specific objectives:

1. **Educate** yourselves about global market place and trades.
2. **Explore** the language elements for programming smart contracts.
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.

Observe the stack diagram given in figure 1. This shows you the stack for Bitcoin and the Ethereum blockchain. On the left is the Bitcoin blockchain that has only its wallet application. You can send cryptocurrency and receive cryptocurrency or transact currency. This is only thing that Bitcoin enables. Whereas the Ethereum stack on the right side allows code execution on the blockchain thus allowing any arbitrary transaction to be enabled on the blockchain. That opens up whole world of opportunities. And you are going to avail of this to build the decentralized market place of items that anybody in the world can sell and buy.

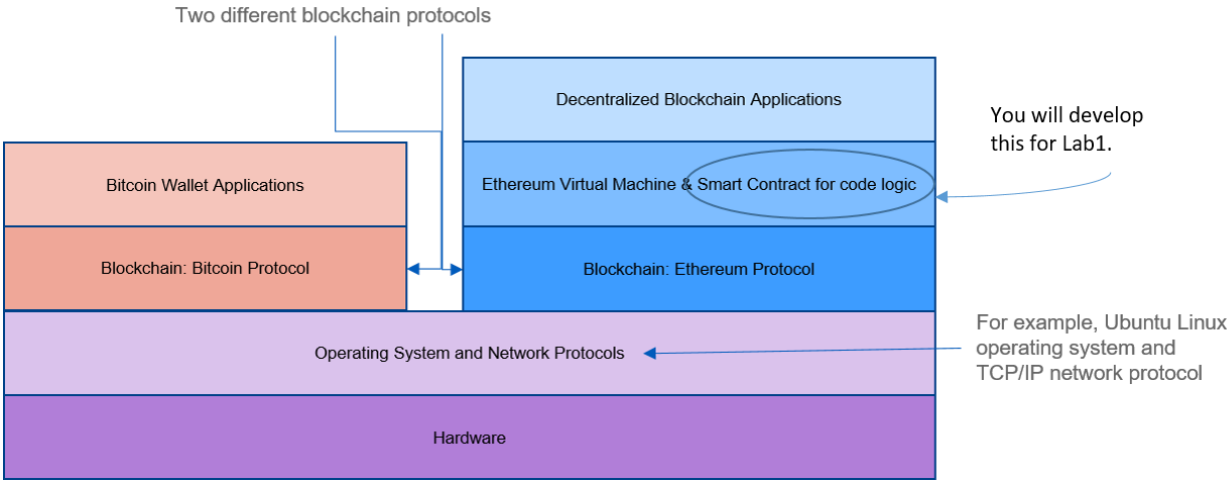

Figure 1 Bitcoin vs Ethereum Application Stack

Part 1: You will have consortium of users with a chairperson managing the entry and exit from the consortium. A member is identified by an account number. He/She/It can list the items they want to trade on a regular web page. A simple table, with item id and name, picture, its status (avail, pending, sold). And button for buy, settle payment etc. You can design this. First implement a simple web-stack with a database with a few items that can be transacted on. Difficulty: Easy.

Part 2: Next add a smart contract and the decentralized elements to that will record all the transactions including user management on the blockchain. The transactions could be: register, unregister, buy, settlePayment, adddeposit etc. Develop and test this smart contract independently. Difficulty: Medium.

Part 3: Final step is linking the web application and the smart contract using the APIs provided by the underlying blockchain protocol as shown figure 1.2. Difficulty: somewhat hard.

Test the part 1 using your regular distributed application methods, part 2 using Remix IDE, and part 3 using user interface or Truffle. We will discuss these further.

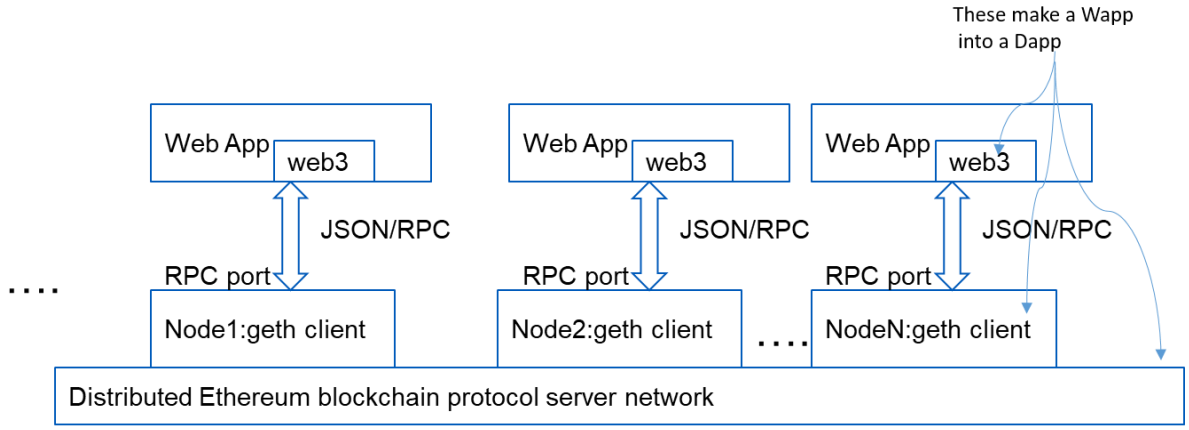

Figure 2 Web app (Wapp) and Decentralized app (Dapp)

Lab 1: 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. Work environment 1: You will developing web stack using html, js, and a simple database. Make sure you have a good editor (e.g. atom) and the knowledge of web stack development.
2. Work environment 2: This is for blockchain Dapp stack development: You will Remix IDE. Make sure you are able to access this. We will check this out with a simple example in class.

Part 1: (30%) Complete the web application that lists items with prices and code for transactions and updating the tables. This gives you reasonable experience in developing a distributed web application. Submit this part before 2/15.

Part 2: (45%) Write a smart contract that will implement the functions for the transactions that needed to be recorded on the blockchain. Use Solidity language and test it using Remix IDE. 2/22.

Part 3: (20%) Link the part1 and part2 and deploy a complete application using web3 API and Metamask for facilitating the blockchain interaction. 3/8.

1. You will create a folder in timberlake named lab1. (Timberlake is a cse server).
2. Every notebook should have your name only at the top of the notebook and your team member's name in the second line.
3. Store or transfer all parts to lab1 folder on timberlake: yourLastNameLab1Part1. yourLastNamePart2, youLastNamePart3, **all the data used including instructions to run, readme etc.**
4. On timberlake tar the lab1 files into yourLastNameLab1.tar

5. Submit using submit-cse410 filename.tar or submit-cse510 filename.tar

DUE DATE: 3/8/2017 BY MIDNIGHT. ONLINE SUBMISSION ON TIMBERLAKE.

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