Due: 12/6/2020 by midnight

TERM PROJECT: DESIGN AND IMPLEMENTATION OF DECENTRALIZED APPLICATION (DAPP)

1. Problem statement:

The main goal of the term project is to design and implement an end-to-end decentralized application on Ethereum blockchain. The core idea is to identify a truly decentralized problem that cannot be solved by the common distributed and centralized systems. Then solve it using the trust infrastructure of the blockchain and its distributed ledger technology (DLT). The solution will have two main components: the blockchain dependent smart contract(s) and the web/mobile/enterprise application that will expose the blockchain features of the application to the users of the system. The final deliverable should include complete code base, detailed design and development documentation, configuration details at various levels (smart contract, web) and instructions for deploying and interacting with application.

2. Learning outcomes:

- Identify a decentralized use case. (Blockchain is a solution for all of your problems.)
- Analyze the problem and represent the solution in the form of a design diagrams.
- Develop, deploy and test the smart contract with the rules, and conditions for validation and verification.
- Design and develop the user application part of the decentralized application.
- Configure and test the integrated system.
- Document and prepare a report for the demo of the completed system.

3. Preparation before lab

Chapter 4 has all the installation requirements: Section 4.1.2. Figure 6.15., Figure 8.2.
Get familiarized with Remix IDE and Solidity language.
Install truffle, ganache, Metamask, Ropsten funds and related artifacts.
You may also need Infura, a cloud-like resources for node of the your Dapp network.

4. Problem Description:

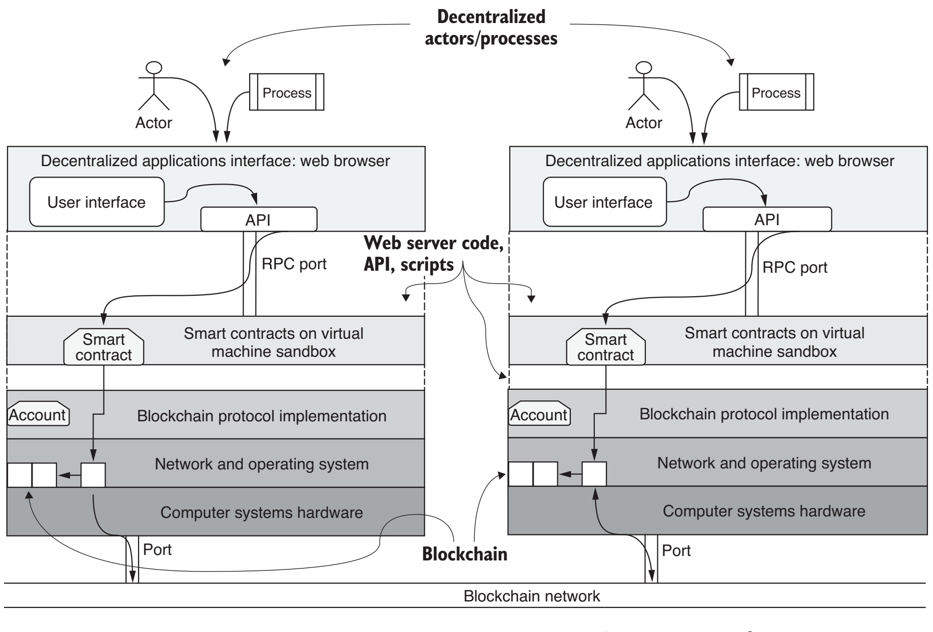
Decentralized actors/processes
Actor
Process
Actor
Process
Decentralized applications interface: web browser
User interface
API
RPC port
Web server code, API, scripts
Decentralized applications interface: web browser
User interface
API
RPC port
Smart contract
Smart contracts on virtual machine sandbox
Smart contract
Smart contracts on virtual machine sandbox
Account
Blockchain protocol implementation
Account
Blockchain protocol implementation
Network and operating system
Network and operating system
Computer systems hardware
Computer systems hardware
Port
Blockchain
Port
Blockchain network

Figure 1: Architecture of a blockchain network

4. What to do?

You will decide on a decentralized application to solve a useful and impactful problem. Just browse the six Dapps discussed in the book.

Table 1 Dapps and their details (From the BIA book)

Ballot.sol	Techniques for trust and integrity	Solidity	Remix IDE	Ethereum, Simulated VM	Access control modifiers; digital democracy
Ballot-Dapp	From smart contracts to Dapps	Solidity+ JS + HTML	Remix, Truffle, NodeJs	Ethereum, Ganache etherem client	Web application and web UI to invoke smart contract functions
BlindAuction.sol	Security and Privacy	Solidity	Remix IDE	Ethereum, Simulated VM	Cryptography, hashing, public-key encryption, Keccak hashing
BlindAuction-Dapp, ASK-Dapp	On-chain and off-chain data	Solidity+ JS + HTML	Remix IDE, Truffle, NodeJs	Ethereum, Ganache	Events and function receipts; off-chain data; on-chain data in a block
MPC-Dapp	Web3 and a channel application	Solidity+ JS + HTML	Remix IDE, Truffle, NodeJs	Ethereum, Ganache, web3 provider	Side-channel; micropayments; using cryptocurrency for incentives for decentralized work; web3 provider
BlindAuction-Dapp, public deployment, MPC public deployment	Going public with Infura	Solidity+ JS + HTML	Truffle, NodeJS, HDWalletProvider	Ethereum, Infura, Ropsten	Infura ethereum client infrastructure and endpoints for public networks
RES4-Dapp	Tokenization of assets	Solidity+ JS + HTML	Remix IDE, Truffle, NodeJs	Ethereum, Ganache, web3 provider	Tokenization, ERC 20 and ERC721 standards

Phase 1: (9/24 – 10/9) Research and identify the decentralized application you'll work on. **Install all the software needed.** Work on the Dapps in the book and familiarize with the design process, development details, deployment configurations, and testing processes. You can start with the Ballot-Dapp. Follow the naming conventions and codebase organization conventions. You should have these details approved by one of the Tas: Chunweim@buffalo.edu (35 students), Yjzhu@buffalo.edu(yanjun) (20 students), Arnabdas@bufalo.edu (15 undergrads).
Output of this phase:

- Identify a major area,
- a title for your project,
- your Dapp name,
- identify your clients (and sponsors, benefactors),
- the decentralized issue(s) you are trying to address: This will be a 100 word abstract in the form of a problem statement. (How, what, when, why? Etc.)

Phase 2: (10/1 – 10/16): Obtain the design diagrams: use case, contract diagram, sequence diagram, finite state machine diagram for the design and development.
Output of this phase: a coherent design document with all the diagrams and explanation.

Phase 3: (10/4 – 10/24): Design, development, deploy and testing of the smart contract part of the Dapp.
Output of this phase: XYZ-contract codebase, where XYZ is your Dapp name.

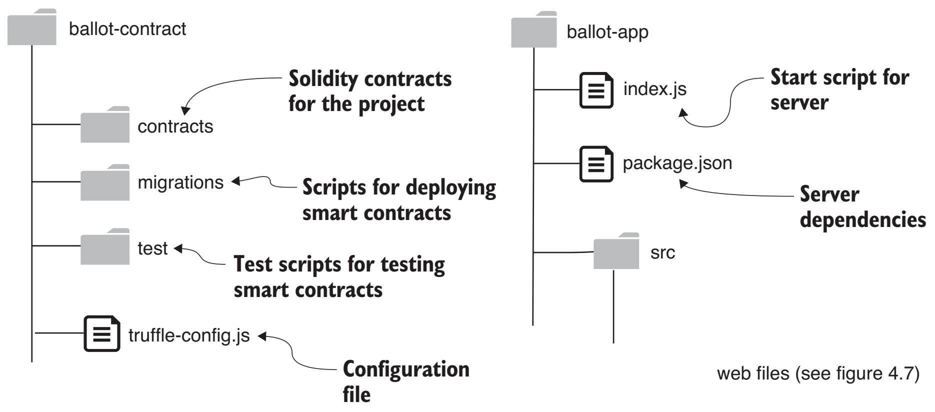

Figure 2: Ballot-contract and Ballot-app part of the Ballot-Dapp

Phase 4: (10/10 – 10/30): Design and development of the automatic test scripts.
Output of this phase: XYZtest.js for each of the smart contract(s): Chapter 10

Phase 5: (10/14 – 11/14): Design and development of the web part of the Dapp.
Output of this phase: XYZ-app codebase, where XYZ is your Dapp name.

Phase 6: (10/20 – 11/21): Deployment on Infura cloud, Ropsten network, IPFS (?mmm?)
Output of this phase: XYZ-app distributable client only, and infura configuration details.

Phase 7: Bundle up and document and hand over 12/6.
Output: Readme, entire codebase, demo.