Design and deploy a smart contract on Ethereum private network (Supplementary Materials)

December 2, 2018

1 Introduction

This document provides hints to some common problems in Lab 3. In addition, it describes an intuitive method to debug your smart contract.

2 Hints

2.1

Select proper compiler version in Remix. Click "Select new compiler version" on the right panel, and scroll down.

For example, before you compile *greeting.sol*, please select version 0.4.0+commit.

2.2

Increase gas limit for *lottery.sol*.

This contract involves lots of math operations and ether transactions, so a higher gas limit is required. When you deploy your contract, a gas value in between *3000000* and *4700000* would be fine.

2.3

Send 1 ether with your *mycontract.join()* function call:

mycontract.join.sendTransaction({from: sender, value: amount})

2.4

Look up a transaction using its hash.

If the command in 2.3 executes successfully, it will print a transaction hash on the console. To obtain details of that transaction,

eth.getTransaction("0xc5a00890...")

Where the HEX string inside the parentheses is the transaction hash.

Each time a player calls *join.sendTransaction()*, use *getTransaction()* to see transaction information. Include screenshots in your report.

2.5

Have another node load the same contract.

Assuming that you have successfully deployed *lottery.sol*, and you want another node (either you set up a new node in another terminal on your machine, or you ask your teammate to collaborate with you) to call *join()* function of the same contract. You need to record *abi* and the address of your smart contract, and send them to your classmate.

Ask your classmate to execute the following:

var contract = eth.contract(abi)

mycontract = contract.at($your_contract_address$)

Now both yours and your classmate's *mycontract* point to the same instance of contract.

2.6

Check balance in contract.

eth.getBalance($your_contract_address$)

Take a screenshot of the balance every time a player *join()* your contract. Take a screenshot of the balance after *selectWinner()* is called by the owner.

2.7

Set *defaultAccount* for the owner node.

Before the owner node call *selectWinner* function, do the following to ensure the ownership can be verified and the gas can be paid:

eth.defaultAccount = eth.coinbase

3 Debug your smart contract

To debug short programs, an intuitive approach is to print out the values of variables one wants to look into. For instance, in a code interview, an interviewee can use *cout* in *c++*, *print()* in *python*, or *println()* in *Java* to test the program.

Now let us assume that you want to test the balance of the contract after *join()* is called by each player, and the address of the winner after *selectWinner()* is called by the owner. What you can do is to modify your program a little bit: return those variables in both functions. (Please remove the test code before you deploy your smart contract to private chain.)

After you have compiled your program on *Remix*, click *Run* on the top-right panel. Then click the *Deploy* button on the right panel. A new contract will appear under *Deployed Contracts*. If you click the arrow to the left of that contract, you should see the two function names: *join* and *selectWinner*.

If you click the arrow to the right of *Account*, you will see several accounts with a balance. Let the first account be the owner of our lottery contract.

In the window to the right of *Value*, type in 1 and choose Ether as unit. Click *join* function name. A table should appear if you click the arrow to the right of *Debug* button on the console. If in *join* function you return the balance of the contract as I do, you should see 1 ether in the *decoded output* row.

Switch to other accounts, type in 1, and click *join* for multiple times. You should see the balance keeps growing.

Similarly, you can switch back to the owner account, and click *selectWinner* to see which address is selected as the winner. The balance of the winner should increase by the amount of that of the contract.