Hardhat tests

These were done primarily for myself, but I figured others may find them helpful. I generally applied a final ‘end-to-end’ approach where I executed a sequence of transactions and check final account balances that I modeled independently in a spreadsheet. While I debugged my program looking at various intermediate steps, once the contract was debugged and modeled correctly in spreadsheets, the final matching of various accounts generates a simple test that the intermediate transactions were correctly processed.

In general, the final balances checked via “assert.equal” statements and correspond to spreadsheet values highlighted in colored cells.

You can run these tests by putting the various Javascript files from smart/hardhat-testlibrary into the smart/hardhat-test folder. Then run “npx hardhat test” in the smart directory.

# Test 1: margins and odds

Test of basic bet payoffs and margin accounting. The spreadsheet shows how various bets should affect the margin before and after results are processed in the settlement. Several separate bets are made on four different matches, for a total of 12 bets. Each match includes bets for both home and away teams, and so involves netting exposure for the bookies.

The bettors should receive their initial bet, plus the payout implied by the odds on their bet. By looking at the resulting balance of the players after redeeming their winning bets we can see that they are attributed their payouts correctly. Specifically, the contract that matches the final balance numbers will have to get the following correct:

1. Odds for match
2. translating odds to payouts on favorite and underdog
3. paying the LP
4. paying the oracle
5. redeeming the correct matches to the appropriate bettors with the appropriate amounts
6. result for match
7. fee to LP
8. fee to Oracle

The amount of eth going in should be attributed to the correct parties.

Initial a-end b-end

LP 30 29.17 31.870

Bettor 2 10 7.20 10.69

Bettor 3 10 13.28 7.234

Oracle 0 0.342 0.207

Sum: 50 99.66 99.79

The contract rounds down, so the data do not match exactly. However, it highlights the basic methods of

Test 1-B is the same as Test 1-A, but with different game outcomes.

# Test 2: Redemptions and Withdrawals

This tests bettor redemptions and withdrawals. Users bet, redeem, and withdraw. Withdrawal checks to make sure gas fee accounts for the difference between the ether going out and the EOA’s change in ether.

**Fails:**

1. An attempt to redeem with no bets
2. Attempt to redeem with active bets

The biggest test here is just to make sure the units are correctly handled, so that when one withdraws 1 avax, one gets 1 avax. The contract adjusts the decimals of avax sent to the contract to conserve on memory, as using 18 decimals in the contract would quickly create problems when the user balances are uint32 (10 decimals) and the total margin amounts use uint64 (20 decimals). In the unit transformation, it is essential to verify that the adjustments for avax sent in are correctly reversed with bettors redeem and withdiraw their balances.

Thus, here I calculated not only the change in the user’s account upon withdrawal, but adjust for the gas cost. Here the gas price is set to 200 gwei.

# Test 3 Voting

Tests that data sent to the oracle are processed correctly in their sequencing requirements, and also whether the votes were majorities for passage or not. Note the tokens have three decimals, so 120k in the contract is 1.2e8. There are three token owners

Account[0]: 220k tokens

Account[1]: 120k tokens

Account[2]: 100k tokens

Total of 8.0MM tokens

* The initial process succeeds because there were no other votes, and the initial proposer has all of his tokens count as a yes vote; the majority yes vote is unanimous.
* Attempt to send updated odds is successful, as the total vote is 340 yes, 100 no.
* Attempt to send new schedule and outcomes fails, as the vote total is 220 yes and 220 no. A majority yes is needed, so odds stay at 600.

# Test 4: LP payout

This tests that LPs are allocated eth correctly given their eth investments and LP revenue. It also tests whether the LPs are allocated their appropriate amount of token rewards. LPs are credited with shares based on the current relative ownership, which is the ratio of LP eth to LP shares. A withdrawal redeems shares and then sends the requisite amount of eth to the LP.

From the start of the first contest to settlement, no investment or withdrawal is allowed by the bookies. Outside of this time window the ratio of eth allocated to the bookies (margin[0]) and the shares owned by the bookies, is the share price. Investors are given shares at this price, and shares are redeemed at this price.

Initial LP amounts

shares eth

account 0 600k 60 eth

account 1 400k 40 eth

A bet is made for 5 ETH each period, which loses. This is sent to the bookie margin at settlement, adding to the LP’s eth. There are no winners, so nothing leaks to the oracle.

Both LPs withdraw after the second settlement, which changes their relative ownership of the LP pool.

A new LP joins after the third settlement.

All LPs claim rewards, except for account 0 in the fourth epoch.

In the fifth epoch, all LPs withdraw their capital. The ETH they receive and tokens are tested to make sure the accounting is correct.

# Test 5: Oracle token depositor payout

This tests the oracle’s ability to pay out its revenue correctly.

Case A: In the base case, Test8a, three accounts deposit tokens in the oracle contract at various times. In each epoch the oracle receives 0.1 AVAX in revenue from the 2 avax win (2.5 bet on 1.80 odds winner, 0.1 is 5% of 2), Oracle revenue is only claimed by having tokens deposited in the oracle contract and voting. The spreadsheet calculates this two ways, one by summing the payments (column T), the other by using the liqPool number as in the contract (columns U). The latter matches more precisely due to the rounding

Rounding always reduces the payments to withdrawers, so rounding errors will not cause an insolvency. They do make it more difficult to test the contract, however, as one cannot be certain a discrepancy is due to an innocuous rounding error or something more dangerous.

The total payouts to the three oracle depositors is checked for accuracy. The total payout to all oracle token holders should be 0.5 (5 settlements getting 0.1 avax).

case B: Account 0 withdraws part of his tokens early on.

This is case A above, but here we have a partial withdrawal after the third epoch. We are verifying the contract correctly adjusts the token payouts for this complication.

Case C: This just reverses case B, where account 0 deposits instead of withdraws. An oracle withdrawal or deposit sends the owner their avax, and resets the depositors accounts payable back to zero.

Case D: Account 1 votes 11 out of 14 times, which decrements his payout by 3/14. This clawback is then thrown back into the pool, adding to the payouts to the other token depositors.

# Test 6: Bet size limits

The LP puts in 30 avax, and with a concentration factor of 5, the initial maximum payout is 6 avax. The “odds” number is zero, which corresponds to a gross LP payout of 1.953 for both teams on the match targeted.

Given the odds, this corresponds to a bet of 6.2953 for team 0, match 0.

The initial test confirms that a bet for 6.2954 is rejected, while a bet for 6.2950 is accepted.

Bets are placed on both teams, increasing the maximum allowable bet. A bet of 13.116 is rejected, while a bet for 13.1148 avax goes through.

More bets are added, and a final rejection is made on the 2.2994 avax bet, while a 2.2964 bet works.

even though a bet for 6.2954 failed earlier. A bet of 13.116 is rejected

# Test 7: Sequence and timing restrictions

There are several restrictions in the contract, and this tests to make sure they are working

rejections:

1. withdraw book deposit within epoch
2. processVote when nothing to vote on
3. Post wrong hour
4. send settle before odds were processed
5. bet before odds processed
6. settle sent too soon, needs 2 days from next Friday
7. send settle after settle (wrong order)
8. game times before next Friday
9. game times after next Tuesday
10. withdraw from book while bets active
11. withdraw tokens during vote
12. data submission sent twice by same account for consecutive data submissions (they can send multiple times on the same submission)
13. odds sent too extreme