**TRADING BOT**

DEVELOPMENT (2)

horizontal line

# 

# 

# Introduction

In the last stage of development, we presented the flash loan bot that could make profits by lending flash loans from one pair and swapping it in another pair on BNB chain testnet.

In this stage, we have come up with Triangle Arbitrage bots which were proven to be efficient and profitable on mainnet instead of testnet but have lots of differences in the functionality and the working way compared to the previous flash loan bots.

With our painstaking efforts, our bots took the right moment and performed an arbitrage transaction just the same as we had predicted and made about **9 PTK tokens profit on BSC mainnet** in our live testing. ( PTK token was deployed by ourselves for test)

But the prices of tokens in the pairs may vary as to the cryptocurrency market conditions so whether the arbitrage brings the absolute standard profit or not, partially depends on third party conditions.

In addition, bearing in mind the fact that millions, if not billions, of bots are in active operations at the moment and they already account for over 51 % of all transactions in the blockchain world, we soon realize the hardships and sparse possibilities in making great profits with our bots.

But as the saying goes “The stronger live, the weaker die”, we should strengthen our bots by providing them with powerful computation power and more quick responsive chain RPCs or making it more intellectual with the constant upgrade of algorithm so that we can guarantee high stability and profitability of our bots all the time.

# How do bots work?

Firstly We have found out all the possible triangle pairs existing in **three most famous Dex sites on BNB chain mainnet**.

Those are Pancakeswap, Sushiswap and Biswap and we found out the factory and router addresses of all above Dex sites and saved those addresses as a dex.json file.

In the preparation stage, we collect and make clean about the pairs that exist in all three Dexs and export the result in allpairs.json file.

Next we continue to find pairs that can make triangles from generated allpairs.json file and save those results in pairs.json file.

When the bot starts to run, it first opens a web socket for receiving transactions data from bsc mainnet web socket server that is wss://bsc-rpc.publicnode.com.

const web3\_WSS = new web3(process.env.WSS\_PROVIDER);

const web3\_HTTPS = new web3(process.env.HTTPS\_PROVIDER);

The bots constantly listens to the events that the server sends and when the newBlockHeader event occurs, (this event usually occurs when a new transaction is added to the block) the bots start to check whether they can make arbitrage or not.

web3\_WSS.eth.subscribe("newBlockHeaders")

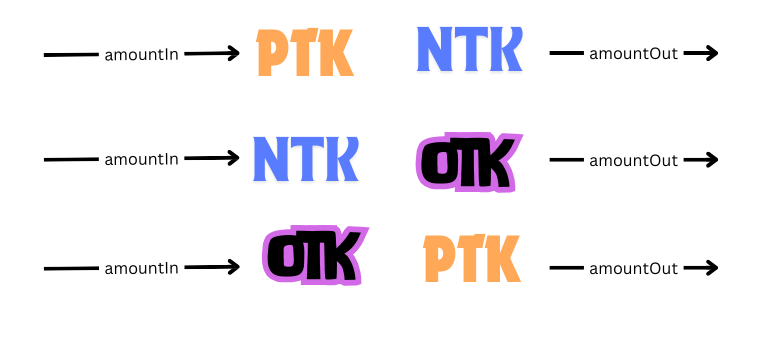
.on("data", async (event) => {

....

}

In the main arbitrage logic, our bots loop for all the triangle pairs ( you can specify certain pairs you prioritize) and check whether it’s the right time to make profit or not.

This could be done by calculating the amountIn and amountOut values using router contract’s getAmountsOut and getAmountIn functions and finally deciding if amountOut from PTK in the last pair is greater than the amountIn to PTK in the first pair.



After deciding the moment to take an arbitrage action the bots use the pair contract’s swap function by setting the caller as **Triangle\_Bot smart contract** which we deployed on BSC mainnet already.

Then after the main swap operation ends which transfers a certain amount of PTK to the Bot contract , callback functions defined **Triangle\_Bot smart contract** will be called with the parameters handed in the form of calldata.

IPancakeCallee(to).pancakeCall(msg.sender, amount0Out, amount1Out, data);

And finally our bots will perform dedicated remaining operations, get the surplus PTK tokens and transfer those tokens to the owner of our bots.

# Profit Analysis

And Now it’s time to investigate the transaction and calculate the profit and loss carefully.

First Let’s navigate to the following url which is the deployed Triangle Bot contract address on bnb mainnet.

The Bot contract is verified and you can view its source code in the code tab.

<https://bscscan.com/address/0x1e2e73Adcd29b15e36308b09c32114fDE70Eb017>

For a live testing , we deployed NTK, PTK, OTk tokens and their deployed addresses are

NTK : 0x726B357ff60Bd569BA05C26BBaf49F6884FCc9b5

OTK : 0x6b04B88795d85D43b1084462adE47b54724e24a0

PTK : 0x9C0CC825eBA6e72E406074dDD48e261C25abCbA0

And also their pair contract addresses also are like the following.

NTK / PTK : 0x563Fa114dca80a27Afd86a320E7fE7D887847699

PTK / OTK : 0x402E80AC4F4C04926C07842a43707125e1a029Fa

OTK / NTK : 0x91bBCbE21D19333a7650907482E06d1E12CefC30

Next we are going to explain about the main transaction in more detail.

This is the transaction history on BSC mainnet.

<https://bscscan.com/tx/0x17c21194cb5c60883a879b4728d7aed0753f9cab14d65e441523a3beb5e63521>

The following are the main stages of the above transaction.



Let me explain about the individual address one by one.

- PancakeSwap V2: NTK-PTK

This is the NTK and PTK pair contract address on the BSC mainnet.

- 0x1e2e73Adcd29b15e36308b09c32114fDE70Eb017

This is Triangle Bot deployed on BSC mainnet.

- PancakeSwap V2: NTK-OTK

This is the NTK and OTK pair contract address on the BSC mainnet.

- PancakeSwap V2: OTK-PTK

This is the PTK and OTK pair contract address on the BSC mainnet.

- 0xd22381Cc86Bf56a1a35F9B7A0a0B23A55dCfC07C

This is the address of the bot's owner on BSC mainnet.

So our triangle bot first borrow 0.001 NTK from the NTK-PTK pair in Pancakeswap.

And next triangle bot send borrowed amount of NTK to the OTK / NTK contract in Pancakeswap and get the 0.009974990049947425 OTK.

Afterwards, the triangle bot swap gained OTK amount with the 9.940162061311326012 PTK from OTK / PTK frontend.

Then the triangle bot sends the amount of the equivalent price of borrowed NTK to NTK / PTK pool in Pancakeswap contract, that is 0.000010025063659148 PTK but it has remaining 9.940152036247666864 PTK.

Last but not least, the bot transfers those remaining PTK to the owner, that is 0xd22381Cc86Bf56a1a35F9B7A0a0B23A55dCfC07C in this case.

Profit - 9.940152036247666864 PTK .

Transaction Fee - 0.000421137 BNB($0.24)