Dune-FreeTON Swap Implementation

1 Abstract

This is a submission for proposal #169 Contest: Dune → FreeTON Swap Implementation Stage 1 [13], the first phase of the implementation of proposal #117 Dune Network Merger [2] voted on Feb 1, 2021, following the proposed architecture in the contest #149 Contest: Dune → FreeTON Swap Architecture [14]. This implementation contains the full implementation of the bridge, including smart contracts for Free TON and Dune Network, and relays in OCaml watching both blockchains and transmitting information between them.

Wallet

Contact on Telegram: @fabrice_dune
Surf wallet: 0:60a74bf0a86b3ab44de42b8ccac944ff8fa95745add04b2505cb84fd42265783
Public Key: 0x97d1ac029229af5c5e7196932106186a4a085fe988562409803b3c4508da5475

2 Description

This implementation follows the architecture provided in the previous contest [14].
We only emphasis here the differences with the initially proposed architecture:

- Smart contracts on Free TON have been simplified, as the DuneGiver, the DuneRootSwap and the DuneEvents contracts have been merged into a single contract, providing the interfaces of the 3 initial contracts. This change simplifies the implementation without changing the abstract architecture:
    - The deployment is easier, as there is only one contract to deploy on the FreeTON side
    - The configuration is easier, as all abstract contracts know immediately the address of the other ones, it's only one contract
    - Gas management is easier: provisioning DuneRootSwap was complex, and DuneEvents would keep some unused gas until the end of the merge. On the contrary, having a unique contract means that extra gas from DuneEvents can be used by DuneRootSwap immediately.
- Instead of having a per-user contract managing multiple swaps by this user, the DuneUserSwap contract is used only for one swap. It is deployed after computation of the hash of the swap fields. Again, it simplifies the implementation as there is no need to manage a map within the contract.

3 Implementation Code

The implementation code is available here in open-source:
https://gitlab.com/dune-network/ton-merge/

3.1 Repository Content

The repository contains:

- The Free TON smart contracts in Solidity in contracts/free-ton/
- The Dune Network smart contracts in Love in contracts/dune-network/
- The Free TON part of the relay in src/free-ton/
- The Dune Network part of the relay in src/dune-network
- The Webapp for users to deposit their tokens and follow the swaps in webapp/

3.2 Build Instructions

- Install a recent Rust version:

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
. ~/.cargo/env
```

- Install a recent `ft` version (freeton_wallet):

```
opam repo add ocp
git+https://github.com/OCamlPro/ocp-opam-repository --set-default
--all
opam update
opam switch create 4.10.0
sudo apt-get install libssl-dev
opam install ssl.0.5.9 # 0.5.10 fails to compile
opam install ft
eval $(opam env)
ft init
```

- Prepare the project

```
git clone https://gitlab.com/dune-network/ton-merge.git
cd ton-merge
opam switch link 4.10.0
eval $(opam env)
make build-deps
make submodule
```

- Configure PostgresQL (replace `<user>` by your login):

```
sudo -i -u postgres
$ psql
CREATE USER <user>;
```

```
ALTER ROLE <user> CREATEDB;
```

- Build

```
make init-db
make
make ft-import
```

- Check binaries: the following binaries should be available in the `bin`/ subdirectory:
  - `ton-merge-init` : executable to setup the Dune part of the bridge
  - `ton-merge-api-server`: API server for the webapp
  - `ton-merge-dune-crawler`: crawls the Dune blockchain for new swaps
  - `ton-merge-dune-injector`: injects secrets revealed on Free TON on Dune
  - `ton-merge-freeton`: crawls Free TON and injects new swaps

Testing

The sub-directory `test/free-ton/` contains a sequence of scripts that can be run either locally (TONOS SE) or on the testnet to check the Free TON part of the bridge.

Test Deployment

This implementation has been deployed on the Testnet networks with the following configuration:

- Website: https://freeton.dune.network/
- Contract address on Dune Network: KT1WukqbELKWtgsLADo49rNXFUopGScFcDHF
- DuneRootSwap contract on Free TON: 0:31c23d599cacfe7e2216874984a66a285b788eee8ee1f1dac442f1096286c9e0
- Configuration:
  - Relays public keys: Origin Labs: 0xf55e101690d6ee2fd9cd52e1a76f9fff67dee5688adfecb7b828709b703b7011
  - A swap must be observed by 1 relay
  - Minimal swap of 1000 DUN
  - Merge Period:
    - Last date for deposit on Dune Network: 2021-06-17T00:00:00Z
    - Last date for revelation on Free TON: 2021-06-21T00:00:00Z
  - Swap Period:
    - Time to reveal on Free TON: 2 days
    - Time before cancellation on Dune Network: 4 weeks

Deployment Instructions

Contract Deployment

The following steps should be performed to deploy the contracts:

- Initialize the database
  ```
  $MERGEDIR/ton-merge/bin/ton-merge-init --config config.json
  ```
  Where config.json contains:
  ```
  {
        "nodes" : [
        "https://n.testnet.b3.nude.ovh",
        "https://testnet-node.dunscan.io/"
        ],
        "block_time" : 30,
        "crawler_sleep" : 5,
        "confirmations" : 3,
        "api_port" : 7777
  }
  ```
  For Testnet

- You should have an account on Dune Network with tokens, known by `dune-client`
  Create a file contract.json with:
  ```
  {
        "deployer_alias" : "dn1MTYYVnyMzC4y3U4oTa8a3B2zY7KkWf3Di",
        "delegate": "dn1GgDxNZeGF3vr91EjrGyYrWuoX62iPvsn2",
        "admin": "dn1MTYYVnyMzC4y3U4oTa8a3B2zY7KkWf3Di",
        "swap_start": "2021-06-01T00:00:00Z",
        "swap_end": "2021-06-10T00:00:00Z",
        "swap_span": 345600,
        "vested_threshold": "0.",
        "total_exchangeable": "463000000.0000"
   }
  ```
  The `deployer_alias` and `admin` fields should point to the address of your account.
- Run the Dune deployer:
  ```
  export DUNE_CONFIG=testnet
  $MERGEDIR/ton-merge/bin/ton-merge-init --deploy contract.json
  ```
  This should create a file `setup.json` that will be shared with other relays.
- Deploy the contract on Free TON using `ft` :
  ```
  ft contract --create root_address --sign admin --deploy
  DuneRootSwap --params '{ "relays": [
  "0x%{account:pubkey:relay}" ], "nreqs": %{env:TON_NREQS},
  "duneUserSwapCode": "%{get-code:contract:tvc:DuneUserSwap}",
  "merge_expiration_date": %{env:TON_END_DATE},
  ```

```
"swap_expiration_time": %{env:TON_EXPIRE_TIME}, "testing": true
}'
```

Which makes the following assumptions:

- An `admin` account has been created with enough tokens for the deployment
- The following env variables have been set:
    - TON_NREQS=1 number of required relay confirmation
    - TON_END_DATE=1623283200 unix time for merge end (use `$MERGEDIR/ton-merge/bin/ton-merge-init` to print this value)
    - TON_EXPIRE_TIME=172800 revelation time on Free TON

The command will define an account root_address with the address of the contract, you can print it with:

```
ft account root_address --info
```

Relay Deployment

The instructions to run a relay are described in the repository in the file:
https://gitlab.com/dune-network/ton-merge/-/blob/master/README.relay

These instructions make the following assumptions:

- The relay holds a Dune network account with about 1000 DUN for gas of revelation transactions
- The relay holds an account with a keypair on the Free TON network. The corresponding pubkey key has been transmitted to the deployer for initial configuration (though it is possible to add or remove a relay at anytime)
- The deployer of the infrastructure sent the following information to the relay:
    - The `contract.json` file describing the Dune Network configuration
    - The `setup.json` file describing the initial state of the Dune Network contract
    - The `address` of the DuneRootSwap contract on Free TON

User interface

The user interface can be tested on https://freeton.dune.network/
The user can check all the swaps corresponding to his address, and create new deposits to initiate new swaps. A deposit can either be made on the interface using the Metal browser extension or manually using `dune-client` in a terminal. The revelation of the corresponding secret can either be done automatically by the relays, or using either the Extraton browser extension or `tonos-cli` in a terminal.
For every deposit, the user must set:

- The amount of DUN transferred (1000 minimum)
- The destination address on Free TON
- The public key on Free TON (for revelation)
- A refund address in case of swap cancellation

- A secret that will be revealed on Free TON, either manually or automatically

4 Links

[1] #149 Contest: Dune → FreeTON Swap Architecture:
https://gov.freeton.org/proposal?proposalAddress=0:cf1c6cbd204093c1a21ae5d384d47091020244da814f3c799dc346363ccc1afe
[2] #117 Dune Network Merger
https://gov.freeton.org/proposal?proposalAddress=0:0136a706eaf305dc824058dd942dad9982664bd2a7ca03b2143b247a220eff00
[3] the OCaml language https://ocaml.org/
[4] the js_of_ocaml OCaml to Javascript compiler
https://ocsigen.org/js_of_ocaml/latest/manual/overview
[5] OCaml Postgres Binding pgocaml https://github.com/darioteixeira/pgocaml
[6] ez_api API Server Library https://github.com/OCamlPro/ez_api
[7] Dune Network: Liquidity smart contract language for Love https://liquidity-lang.org
[8] Spice testing framework https://gitlab.com/o-labs/dune-spice
[9] Dune Metal browser extension https://gitlab.com/dune-network/dune-metal
[10] FreeTON : Solidity compiler https://github.com/tonlabs/TON-Solidity-Compiler/
[11] OCaml FreeTON Wallet and SDK https://github.com/OCamlPro/freeton-ocaml-sdk
[12] ExtraTON browser extension https://github.com/extraton/freeton
[13] #169 Contest: Dune->FreeTON Swap Implementation Stage 1 Proposal
https://gov.freeton.org/proposal?proposalAddress=0%3A766a2a6143151f09333cdd6ccf45ee6ae7bcac6ce4c35d5b8f0d68c3ff01dab9
[14] Architecture Submission
https://gov.freeton.org/submission?proposalAddress=0%3Acf1c6cbd204093c1a21ae5d384d47091020244da814f3c799dc346363ccc1afe&submissionId=1