Sharding via the message mechanism
(01.09.2020)

Smart contracts can exchange messages between each other, including between different shards.
When creating a smart contract, the height in blocks is set, which determines the finality of sending a message from one Board to another. When this value is reached, the message is sent to the smart contract for further processing. When sending messages within a single shard, messages are received in the next block. The minimum height is the actual security value if the smart contract is a channel for transmitting valuables or other important information. The larger it is, the more secure the channel is, but on the other hand, the longer users need to wait for the operation to complete.
The process of transferring messages from one shard to another is handled by those miners who simultaneously mine these two shards. If there are no such miners, the message will be postponed and wait for them to appear.

Security

The security of channels can be determined by the cost of an attack of 51% for the time it takes to successfully transmit a message to the shard receiver. The faster the channel, the cheaper the attack, or in other words, the greater the risk. We can build a risk management scheme that provides both high security and high performance through the creation of two value transfer channels:

1) The main channel is a slow but reliable channel. The commit time is always specified quite long, for example, 1 million blocks. This channel has its own token in the receiver shard, which identifies the source shard coin.
2) A fast channel is used to quickly transmit a small amount of value (compared to the cost of an attack). For example, commit an operation after 100 blocks. The channel also has its own token,which we will call a **transit** token. it is only needed for short-term use.

The scheme of work is as follows:

1. The user sends the amount of values via a fast channel (the amount of values is always less than the cost of the attack)
2. After receiving them in another shard, it changes them there to the main tokens, which are transmitted through a slow channel. for this purpose, it uses the built-in DEX exchange. At this stage, the user's work ends.
3. Transit tokens are sent to the first shard, where they are sent back to the second shard after receiving them, but through a slow channel and are put up again on DEX (to ensure continuous liquidity of transit tokens). This is done by another participant in the relationship, let's call him an intermediary, he is a professional participant. In

order to motivate him to place orders for DEX, as well as freeze his capital in the process of long-term but highly reliable value transfer operations, he changes tokens not in the ratio of 1 to 1, but with a certain discount, which includes the cost of freezing capital and its profit. The specific rate will regulate the market.

Sending coins from a wallet between shards

It is possible to send coins directly from the wallet interface between accounts of neighboring shards. to do this, the recipient's account address must contain the name of the shard and the account numbers separated by a colon: "SHARD:AccNumber”
In this case, the name of the shard resolves to the account number with the smart contract, which is the gateway (alternatively, this list of matches can be sewn into the wallet interface itself).

The reason why an interim smart contract is used is to isolate responsibility. If one shard is compromised, the values of the other shards will not be affected. During cross-transfers, coins remain on the smart contract account inside the blockchain, while only a kind of receipt is sent to the other blockchain, on the basis of which token movements are made.

This model allows you to create very wide-purpose shards, such as temporary ones, but with high performance inside.
Example: Shard1 is created, tokens are sold for it, users start working with dapps inside it making 1000 tps, after a while when the base size exceeds a reasonable limit, for example 1000 GB, Shard2 is created, tokens are changed to shard 1 tokens, everyone goes to work in shard2, miners disable support for the 1st shard.

New methods in smart contracts

New built-in method for sending messages (in the new update)::
SendMessage(*ShardAcccountStr,MethodName, Params)*;
where:
ShardAcccountStr - string in the Shardname format:Account number
MethodName - name of the method that will be called to process this message. this method should be marked with the message attribute
Params - parameters that are passed to the specified method

Sample smart contract:

```
//shard1 TERA
"public"
function SendTest(Params)
{
```

```
    if(context.Account.Num!==context.FromNum)
        throw "Access is allowed only from your own account.";

    //to smart's account
    Send(context.Smart.Account,Params.Sum,Params.Description);

    //we send a command to move the token to another shard
    SendMessage("TEST:"+Params.Gate,"SendMyToken",Params);
}



//shard2 TEST
"message"
function SendMyToken(Params)
{
    //call verification-from the correct shard and the correct smart
contract
    if (context.Tx.ShardFrom!=="TERA")
       throw "Error cross-shard name: "+context.Tx.ShardFrom;
    if (context.Tx.AccountFrom!==100500)
        throw "Error cross-gate: "+context.Tx.AccountFrom;

    //sending a token
    Send(Params.To, Params.Sum,Params.Description);
}
```

Methods for working with Oracles are planned (not in the update):

OracleRequest(UrlString, ReturnMethodName)
SomeReturnMethod(UrlString, Result)