BUBBLETONE BLOCKCHAIN

Proposed solution

Blockchain Telecom (BT) provides interoperability of mobile operators through the special smart-contracts (hereinafter - SSC), that are in fact a strictly predetermined finite automation, constructed from sequential transactions, that was put on a blockchain by users and operators. This goal is reached by consensus transaction processing, performed on blockchain nodes. Only transactions, corresponding to previous transactions, having right state, values and signatures of participating parties are valid.

The function of SSC is to store client's funds, allow operators to charge client several times, and, finally, redistribute funds held by SSC between participating parties. These transactions offer services, open requests for service, charge clients and, finally, transfer payments to operator, performed the request, fee to operator, published original request, and change to client.

Execution of SSC, validation of transactions and publication of the new blocks in BT is performed in a decentralized manner, on high-speed nodes, providing extremely high, stable and predictable rate of processing transactions. For the telecom industry, the quality of service in terms of transaction execution speed is not a wish, but absolutely necessary condition for the existence. So, for the implementation of the SSC, we consider only high-speed blockchain projects to ensure the finalization of the vast majority of transactions in a fraction of a second.

BT uses special tokens - Universal Mobile Coin (UMT) and Global Online Token (GOT). The economic model of these tokens is described in detail in the "Economy" section. Here we mean that the main goal of participating parties is a safe, effective(in terms of storage), and technically undeniable (excluding collusion between nodes) transmission of correct amounts of funds from one party to another.

The general procedure is as follows:

Operator, that provides service (hereinafter - "Assignee"), sends a so-called "offer" transaction to BT. "order" transactions contain the information necessary for obtaining the service: price, amount of traffic, and other information, encrypted, if needed. These transactions are actually an operator's prices with info, needed to buy a service.

Operator, representing the customer (hereinafter - "Issuer") selects from the fresh blockchain block an "offer" transaction, ie a specific service with specific parameters and cost. With this offer, Issuer initializes the SSC using transaction "request", which indicates the Assignee, who must carry out request, and set the required parameters. "request" transaction is linked to corresponding "offer", by signing it, so, with "request" transaction Issuer starts transactions chain, that we call SSC. Also, "request" transaction "freezes" some fixed amount of Client or Issuer funds in SSC. These funds will be used as maximum amount of funds, that can be spent on Assignee' services.

Then, Assignee, through the "charge" transaction, reserves amount of funds, needed to provide the service, and starts providing mobile services to the client. it repeats this procedure several times, if needed, sequentially decreasing reserved funds, and increasing funds, destined to himself. All funds continue to stay frozen in SSC upon completion of SSC (by any cause, if all was ok, if service failed, or SSC is expired by time-to-live).

Then, Assignee, Issuer or Client through the transaction "refund" completes the execution of SSC and funds are finally distributed between all participants.

A more detailed discussion of smart contract

The basic algorithm can be represented by a graph of transitions of SSC states. Each edge of the graph - is a transaction, validated and related to the previous transaction with strictly defined type. For example, “charge” transaction will not be accepted by the network if the network does not have a previous transaction with type “request” or “charge” transaction cannot prove its acceptance by signing it.

In the first version of the protocol, we consider only four different states of SSC - offer, request, charge, refund.

offer - a transaction that declares the service of a particular operator, its price, and other info about service. The "offer" transaction can have no previous transactions, and the SSC uses it as an initializing transaction for the whole SSC transactions chain.

request - is the transaction following the "offer", placing information about the client and the operator that is invited to serve it. It transfers some “freezed” amount of funds from client to SSC, and now the only way to return them is send the “refund” transaction. Also, it contains the information necessary to provide the service. The transaction, in fact, places deposit and makes a “task” to operator-assignee in response to his “offer”.

charge - a transaction, following “request” or previous “charge”. It fixes a certain part of service (billing step), which the operator-assignee provided(or plan to provide). This transaction can be sent multiple times, until there are funds, allowed for charge, and all other conditions are met. But in all cases transaction must certify previous transaction (“charge” or “request”), thereby transitively accepting “offer” conditions

refund - a transaction which is carrying out the final allocation of funds. Called by any participant it stops the SSC, transferring funds to the participants in the amounts, corresponding to the charges provided. In the first version of the protocol there is a special flag, indicating that the refund is caused by bad, or non-working service. The number of such transactions, in relation to a particular operator, is a good metric for automatic exclusion from a list of nodes, allowed to finish blocks.

Each transaction in this transactions chain, except the first “offer”, includes the signature of the previous transaction. Thus, each participant confirms the fact that he sees the correct processing of the history of SSC and agree with it. Otherwise, it sends to the network a transaction refund, returning a portion of its funds, and notifying the network about his complaint on quality of service of particular operator.

The states(transactions) of SSC are presented in Appendix A “Service Smart Contract states and transitions described”

SSC side points and practical aspects

We would like to see all the interactions between the operators of the transaction took place within BT. This will allow the network to be as robust and flexible as well as eliminate the need for direct integration between operators, that was a big technological problem for a long time, thanks to very diverse equipment. For example, when client wants to register in operator's network, he need to download a mobile profile to SIM card (or the built-in SIM chip in the client device). Mobile profile is an IMSI (International Mobile Subscriber Identity) and a set of secret keys for authentication.

Symmetric key exchange, organized within the SSC from first transaction is very useful for such kinds of interactions in BT, allowing parties organize fast and secure communication channels, saving time and resources spent for extra challenge-response handshakes between operators' equipment. For example: to secure download mobile agent profile on the client's SIM card or download any kind of secret data between client and operators. The necessary symmetric key can be obtained by both parties from "offer", "request" and "charge" transactions.

Downloading mobile profile is performed by Issuer in accordance with one of the standards for the telecom industry:

Remote SIM Provisioning (https://www.gsma.com/rsp/2017/04/12/remote-sim-provisioning-works/)

Download information on the SIM-Map of SMS-channel in accordance with the GSM 03.48 "Security Mechanism for the SIM.

The solution for safe mobile profile downloading and installing is already realized by our team, or Issuer can use solutions from other companies, but it's important that these solutions satisfy the standards above.

BT blockchain

The main requirements for BT is a guaranteed level of service. Stable transaction closure speed is critical, because otherwise the client will not be able to get the service he needs. It should be noted that at a stable transaction closing rate, we mean not only the average time of placing the transaction in a block, but also the problems of peak loads. Even in the case of a high average speed, a noticeable increase in processing time on peaks (for example, when a big operator publish many service offers or charges), it is necessary that BT continue to steadily serve customers. So, at the current stage, we plan to use only highly available specialized servers, owned by operators, responsible for its availability and performance.

The job of the BT node is the validation of all new transactions, adding them to blocks and publishing blocks in the p2p network. Node checks the current transaction, searches for the previous one in the blockchain, checks the signatures and balances of the participants. The inclusion of a transaction in the block means, that node verified it, verified its authenticity, and guarantees the correctness of the data. Also, it is the node that receives the commission for placing the transaction in the block. Transactions validation, in fact, is the real execution of a SSC, because all our logic, checking signatures, balances, transactions order is really only makes a choice - is transaction valid or not. So, the SSC is "hardcoded" in node's validation algorithm and renews only with node's software

The requirement for quality of service limits the range of available solutions for building BT. From the candidates for building high-speed blockchains with a controlled closing time of the transaction, we selected engines based on the consensus of the Delegated Proof of Stake. They allow to assign high-performance servers as delegates, closing blocks at predictable rate, and control the degree of decentralization. In future versions, the options for building a node hierarchy in BT are considered, allowing to pass some transaction processing jobs to client devices, allowing them to help nodes.

Also, a good candidate for the role of the main blockbuster BT are the engines that implement sidechains for large, decentralized blockchains - Bitcoin and Ethereum. We examine the Rootstock and Plasma engines to evaluate their suitability for our task. There is a big plus that they allow using traditional BTC and ETH crypto currency in the system, greatly facilitating the fight against the volatility of the working token and are more convenient to use than the self-made tokens

Nevertheless, at this stage, the most promising direction of development is Graphene engine and decisions, based on it (bitshares, steemit, golos). This engine already works in the production environment, it constantly develops, and, most importantly for us, provides following important functions:

- add / remove delegates (operators)

- p2p network for publishing transactions and blocks

- implementation of “transfer” and other exchange functions

- account management and many functions to operate with accounts

- excellent performance due to implementation in C ++

- many types of transactions and the ability to create and modify our own types

** The implementation of the SSC in the Ethereum network is also considered as a prototype in order to present a solution to the experts and the community*

One of the implementation problems is the update of the software on the nodes. SSC implies further development, and protocol will require changes, at least necessary to add the data to provide new types of services. Changing the logic of transactions validation can easily take out an outdated node, so we plan to make protocol changes that are strictly compatible with previous versions. The fact that delegate nodes belong to operators, who have a real economic incentive to keep the code up-to-date, is an additional advantage.

Including new nodes in BT consensus is also a problem. To provide telecom services, the operator must undergo an offline validation procedure, have a license and title documents. This, on the one hand, protects the network from bad players, and on the other hand it obstructs the operational management of the network and the rapid growth of the number of nodes, because "lazy" operators can ignore voting, sabotaging new node inclusion. We plan to resolve this issue with the help of a functional that will force operators to participate in the election of delegates. Those participants who did not participate in the procedure of consensus adding / excluding nodes for a long time, will be

penalized by rating downgrade and exclusion from consensus. Unfortunately it doesn’t defend against "angry" nodes, who always say "no", so this case need further research.

Decentralization of BT is not an easy matter. From the client's point of view, BT is not decentralized, it's just a convenient service, organized by operators. From the operator's point of view, BT is decentralized, because nodes are equal, no one can control all of them, and new nodes are added by consensus of operators. We do not see a contradiction here, because the internal interaction of operators among themselves is not a problem of the client, which in general does not care how he gets the service. At the same time, it is important for operators to know that each of them is independent, and transactions in BT are technically undeniable.