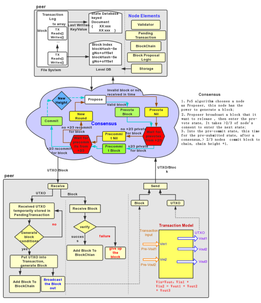
Introduction

Table of contents

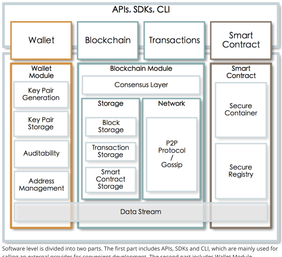
TRON contains consensus engine, ABCI, UTXO, smart contracts and other modules. Consensus engine is the core, application connects with consensus engine by ABCI to form a Byzantine fault-tolerant state machine, which can be implemented in any programming language.



TRON blockchain platform has the following characteristics:

- Scalability: TRON blockchain can be extended through the side chain, which means that not only currency transactions, legally binding contracts and certificates, audio and video files can be stored in the blockchain database;
- Decentralization: Without an agency, all nodes have the same rights and obligations, any node stopping working will not affect the overall operation of the system;
- Trustless environment: All nodes in the system can be traded without trust. Because the operation of the database and the entire system is open and transparent, the nodes can not deceive each other;
- Consistency: The data information between nodes is consistent;
- Fault-tolerant: The system can accommodate 1/3 node Byzantine failure;
- Scalability Account Model: UTXO Model + Account Abstraction. TRON has also made targeted improvements on the premise of UTXO's easy-to-parallel computing model. To make data easy to manage and easy to program, TRON introduces the world state-lightweight state tree concept, each of which maintains a global world state, the global state has the features of quickly find, can not be changed, easy to provide proof.

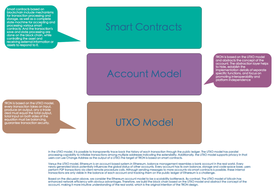
Software Hierarchy



Software level is divided into two parts. The first part includes APIs, SDKs and CLI, which are mainly used for calling an external provider for convenient development. The second part includes Wallet Module, Blockchain Module and Smart Contract Module, provides a storage interface, making the data of each module persistent.

UTXO

In the UTXO model, it is possible to transparently trace back the history of each transaction through the public ledger. The UTXO model has parallel processing capability to initialize transactions among multiple addresses indicating the extensibility.



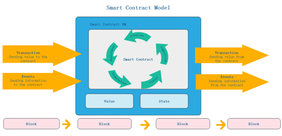
Additionally, the UTXO model supports privacy protection in that users can use Change Address as the output of a UTXO. The target of TRON is based on smart contracts.

Compared with the UTXO model, Ethereum is an account based system. In Ethereum, balance management resembles a bank account in the real world. Every newly generated block potentially influences the global status of other accounts. Every account has its own balance, storage and code-space base. users perform P2P transactions via client remote procedure calls. Although sending messages to each account via smart contracts is possible, these internal transactions are only visible in the balance of each account and tracking them on the public ledger of Ethereum is a challenge.

Based on the discussion above, we consider the Ethereum account model to be a scalability bottleneck. By contrast, The UTXO model of bitcoin has enhanced network efficiency with obvious advantages. Therefore, we build the block-chain based on the UTXO model and abstract the concept of the account, making it more intuitive understanding of the real world, which is the original intention of TRON.

Smart contract

Certainty and Termination are two properties of a smart contract. When designing a smart contract system, non-deterministic factors need to be excluded.



Bitcoin has a set of scripting engines, the instruction set is very simple and non-Turing complete, with termination, so bitcoin smart contracts are certain. The Ethereum Virtual Machine (EVM) is a runtime environment for Ethereum smart contracts. The system functions for Ethereum smart contracts are not nondeterministic, but the contract's call path can be nondeterministic and result in a scalable performance Losses, it uses meter to achieve the termination. The Hyperledger Fabric smart contract uses Docker as the execution environment, Docker is a lightweight virtualization technology, under the blockchain Docker is a "heavier" execution environment, which is where the performance bottleneck of Fabric, currently only up to hundreds of TPS per second, which uses a timer to achieve Termination.

In order to keep with the advantages of certainty, termination, and lightweight of virtual machines and the language flexibility of container programming, TRON is poised to develop the TRON Virtual Machine as an execution environment for its smart contracts in the future. The TVM boots very fast, occupies less resources. TRON virtual machine data manipulation instructions are directly to the array and complex data structures to provide support. These will enhance the operational performance of TRON smart contracts. The TRON Network plans to charge for the operation and storage of tokens and smart contracts to achieve economic incentives to book-keeping persons and to prevent the abuse of resources.

In the future TRON smart contract developers can use almost any high-level language they are good at for TRON smart contract development. The first language support are java, Go etc. Tron plans to provide compilers and plug-ins for these languages to compile high-level languages into the instruction sets supported by TRON virtual machines.

The TRON smart contract model shown above is a piece of code (a smart contract) that runs on a smart contract virtual machine and is deployed on a shared, replicated ledger (blockchain). TRON has a life cycle for smart contracts management, respectively are: the establishment, deployment, development, rollback, termination. It can maintain its own status, control its own asset value and receive external information, transactions or external information and transactions to respond.

Consensus

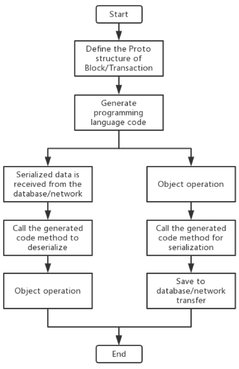
The consensus of TRON adopts a three-step strategy. The first step is to adopt a Kafka-based technology system to implement a centralized consensus algorithm. The purpose of TRON is to achieve system joint debugging and functional integration.

The second step is to use Raft-based distributed consensus mechanism to realize the centralized and distributed leapfrogging. This step gradually improves the functions of network and distribution and lays the foundation for the eventual realization of a wide distribution with no logical center.

The third step is to realize the Consensus mechanism of PoS and realize the Byzantine Fault Tolerant Consensus based on the "Margin Mechanism + Epoch Confirmation" and the compatibility consensus between PoS and PoW.

TRON is currently open source code to achieve a consensus algorithm for the first phase of the center. The second phase of the distributed consensus algorithm is under development and testing.

Protocol Buffer based object coding and serialization



Example

Proto Code

```
message Block {
    repeated Transaction transactions = 1;
    BlockHeader blockHeader = 2;
}
```

Serialization

```
Block.Builder block = Block.newBuilder()
    .setTransactions(transaction)
    .setBlockHeader(blockHeader)
    .build();

byte[] blockData = block.toByteArray();
byte[] keyData = block.getHash();
DB.saveBlock(keyData, blockData);
```

Deserialize

```
byte[] keyData = block.getHash();
byte[] blockData = DB.getBlock(keyData);
Block block = Block.parseFrom(blockData).toBuilder().build();
```