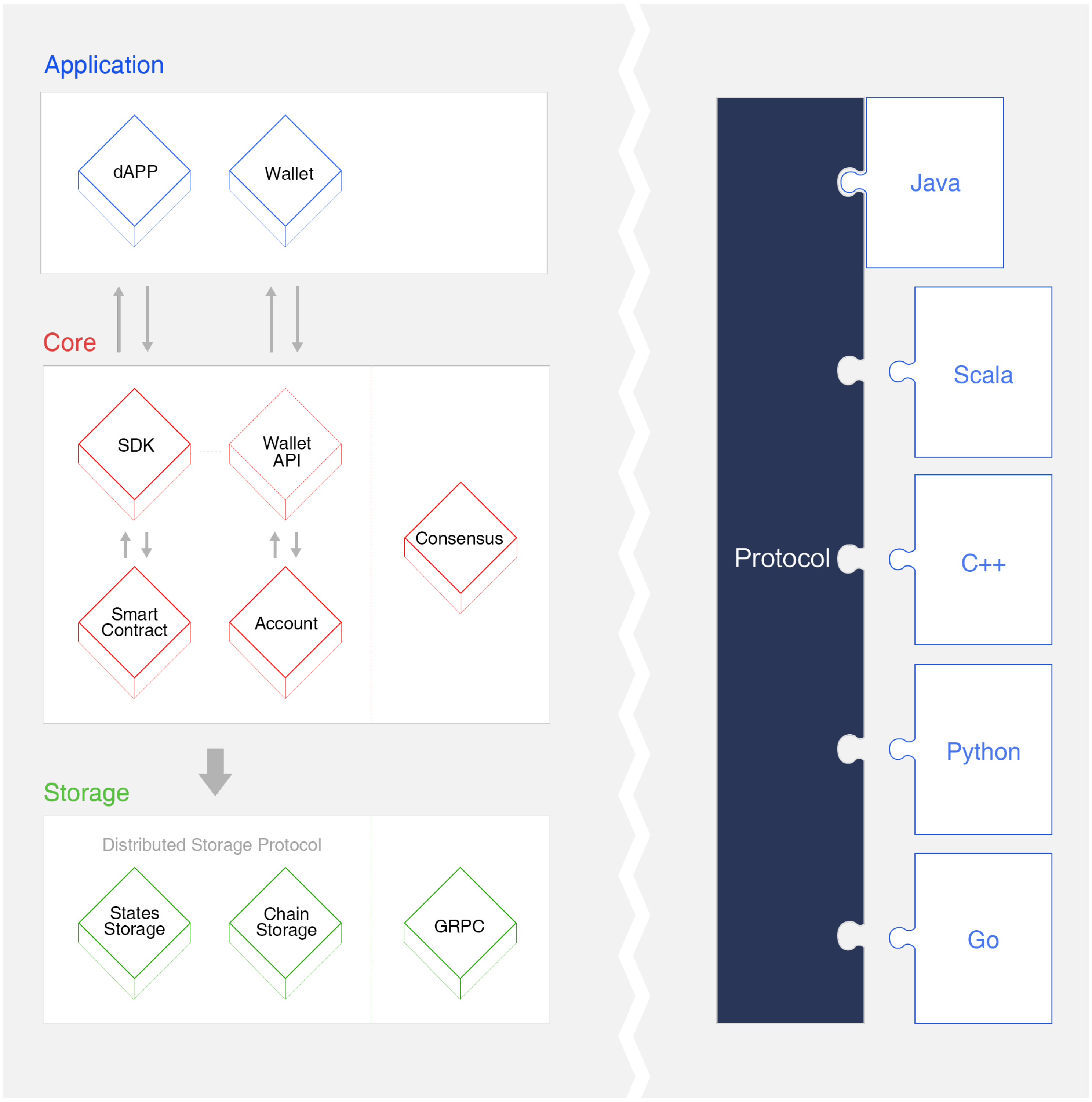
TRON

Design Book of TRON Architecture

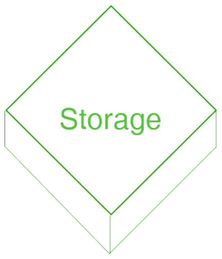
DESIGN BOOK of TRON Architecture

Architecture

Tron adopts a 3−layer architecture, which is divided into storage layer, core layer and application layer.



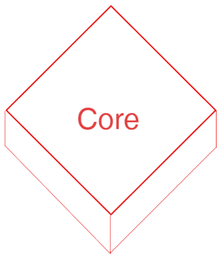
Architecture



Storage Layer

Tech team of TRON designed a unique distributed storage protocol consisting of block storage and state storage.

During the design of underlying storage, the idea of graph database was introduced to meet the needs of diverse data storage in the real world.

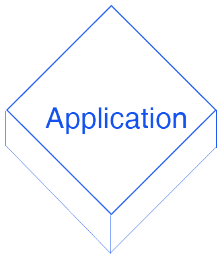


Core Layer

There are several modules in the core layer, including smart contracts, account management and consensus. Also a stack based virtual machine will be implemented on TRON and optimized instruction set will be used.

In order to better serving developers to create dAPPs, Java was chosen as the smart contract language, followed by future support of other advanced languages.

In addition, Tron's consensus mechanism is based on DPOS and some innovations was made in order to meet unique requirements.

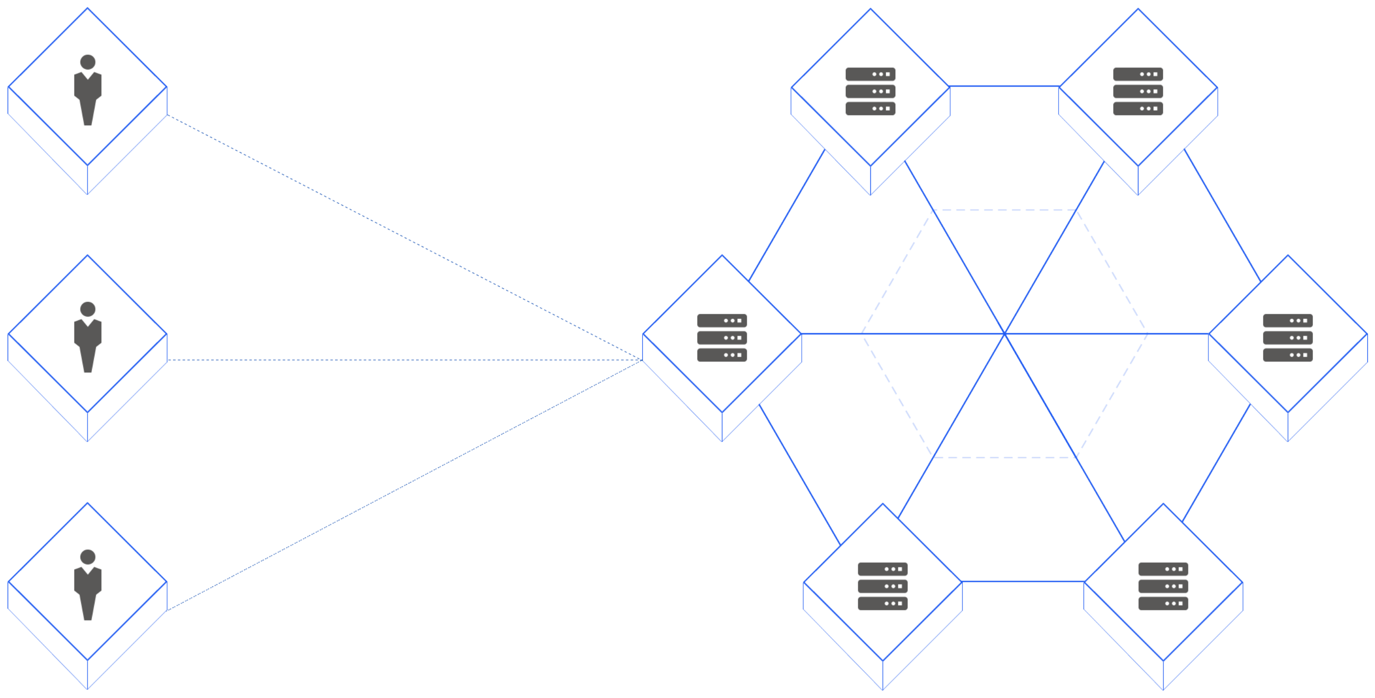


Application Layer

Developers can use given interfaces to easily bring about abundant dAPPs and implement the wallet by themselves.

The protocol of TRON is defined by google protobuf, and naturally supports multi–language extensions.

Consensus

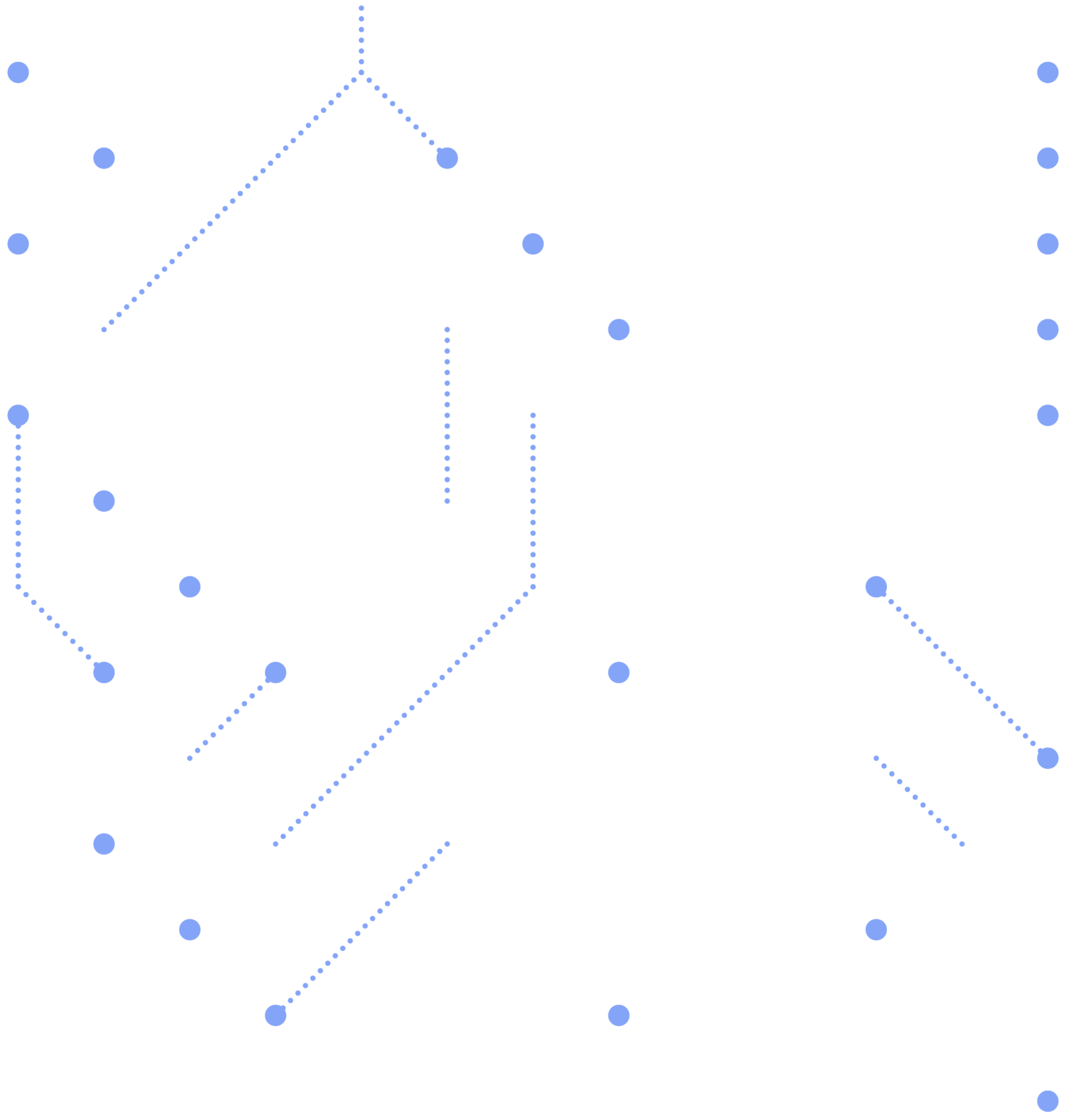


Improved Consensus Mechanism based on DPOS

POW always has its downside like high electric power consuming and low TPS caused by low efficient consensus mechanism. Therefore, it won’ t fit the outlook of TRON. POS outstands to be the undeniable guideline for TORN in order to be a primary blockchain platform for daily scenes. After fully research of impressive ideas in the field of blockchain, we now know for certain that improved DPOS can be TRON’ s consensus to meet our future demands.

Basic Rules of the Consensus Mechanism

- Coin holders are required to vote for nodes in accordance to their tokens that have vote right. And nodes are elected to become what are known as witnesses based on votes and rules, which keep a balance between block–producing speed and the number of witness.
- Meanwhile, voting users and nodes that are voted out will be paid for a certain sum of money for encouraging them to run for further elections.
- Witnesses will produce valid blocks successively based on specific distribution rules and success to do so results in highest reward.
- The vast majority of witnesses are chosen by votes and the rest will guarantee to be selected randomly under certain algorithm.



Storage Structure

KhaosDB

TRON has a KhaosDB in the full−node memory that can store all new folk chains during a certain time and supports witnesses to switch their own active chain swiftly into new main chain.

Level DB

Level DB will be initially adopted and its primary goal is to meet requirements of fast R/W and rapid devel−opment. After launch of main net, TRON will be fully upgraded with a customized database which fits well with TRON.

Digital Asset Module

Configuration

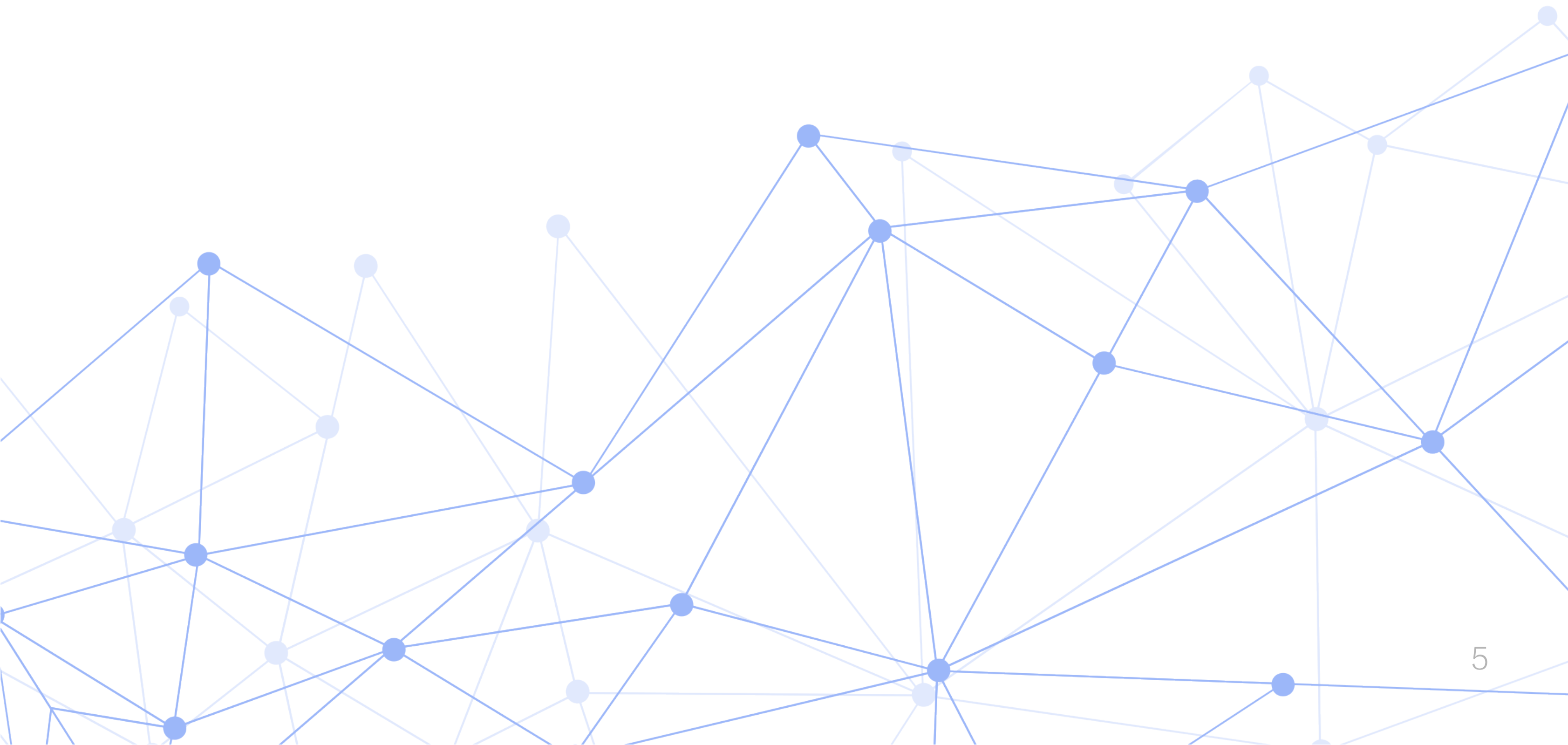
Users can customize their own digital assets through DAC (digital asset configuration) functions.
Types of adjustable parameters include, but are not exclusive to: asset name, abbreviation, LOGO, total capitalization, exchange rate of TRX, starting date, expiring date, attenuation coefficient, controlled inflation model, inflation period and description.
System will provide users with default parameter if users don’t set the parameters themselves.

Issue/Deployment

Users can choose to issue assets after customizing parameter. (artificial or system default)
System comes with operations and functions, and that allow issuers to deploy digital asset, which has already been validated and customized.
Customized asset is deployed once witnesses successfully validate, and can be freely circulated on TRON network.

API

API is mainly used for client development. With API support, asset issuance platform can be designed by developers themselves.



Smart Contract/ Virtual Machine

TRON allows users to create their own contracts of any complexity they wish.
Smart contract will run on virtual machine. In this way, TRON serves as a platform for developers to create many different types of applications that actualize complex scenes.

Third Party Applications

- Digital Assets Deployment Platform

  TRON is perfectly suited to serve as a digital assets deployment platform, where third party is allowed to connect TRON net and customize their own digital asset.

- Wallet

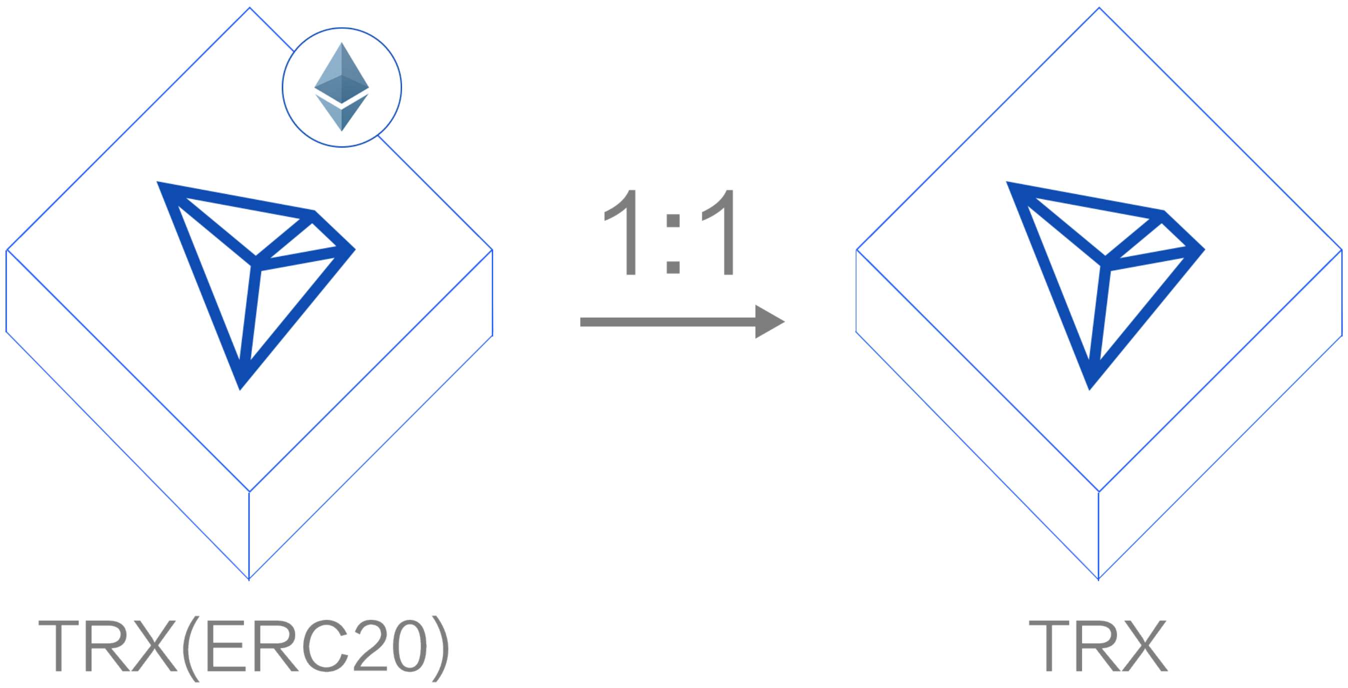
  A wallet tracks the state of every asset, all state transaction of TRX and other users, and also allow one to send or receive transaction.

- Blockchain Explorer

  It is a web tool that provide detailed information about nodes deployment and real−time operation of TRON.

Token Migration

Before the launch of TRON’s main net, the migration from ERC20 to TRX will be started by TRON foundation. The migration exchange rate is 1:1. It has not yet confirmed the concrete migration methods which will still need detailed modification.



Community Plan

We convince that community always plays a main role and a reward plan is able to encourage community members to engage in TRON construction.

Community members and developers are given a wider variety of ways to participate in code writing and third party application developing after release of API. Besides, we hereby call for manuscripts perennially which face all users. The content is any of following: LOGO design, article, artwork, etc.

Providing Types

- feat: A new feature
- fix: A bug fix
- docs: Documentation only changes
- perf: A code change that improves performance
- refactor: A code change that neither fixes a bug nor adds a feature
- style: Changes that do not affect the meaning of the code (white−space, formatting, missing semi−colons, etc)
- test: Adding missing tests or correcting existing tests

Reward Plan

All members, who has devoted themselves to the development of TRON and community, will gain TRONIX or other forms of rewards after detailed estimation by our committee.

Protocol

The protocol of TRON is defined by google protobuf and contains a range of layers, from account, block to transfer.

There are 3 types of account—basic account, asset release account and contract account, and attributes included in each account are name, types, address, balance and related asset.

A basic account is able to apply to be a validation node, which has serval parameters, including extra attributes, public key, URL, voting statistics, history performance, etc.

A block typically contains transaction data and a blockheader, which is a list of basic block information, including timestamp, signature, parent hash, root of Merkle tree and so on.

Transaction contracts mainly includes account creation contract, transfer contract, transfer asset contract, vote asset contract, vote witness contract, witness creation contract, asset issue contract and deploy contract.

Each transaction contains several TXInputs, TXOutputs and other related qualities.

Input, transaction and head block all require signature.

Inventory is mainly used to inform peer nodes the list of items.

Please check detailed protocol document that may change with the iteration of the program at any time. Please refer to the latest version.