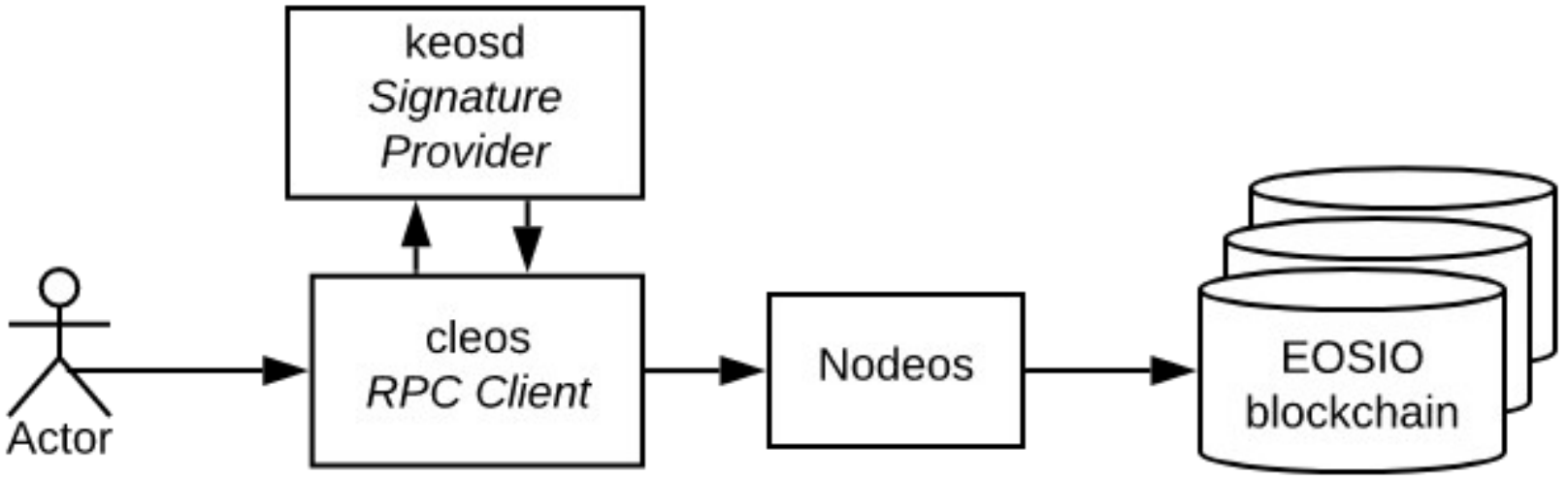
EOSIO Operation

cc32d9 / Europechain

nodeos: the blockchain daemon

- P2P network interface
- HTTP API interface
- Blocks validation and signing
- WASM VM for contract execution
- Additional optional plugins

User workflow



Creating a transaction

- Each transaction is one or more *actions*. An action is executed by an account that has *code* (smart contract).
- Client or user composes a transaction:
  - Contract
  - Action name
  - Action arguments
  - (optional) more actions within the same TX
- Authority

Sending a transaction

- Client prepares a transaction:
  - retrieves a valid irreversible block ID
  - retrieves public keys matching the authority
  - requests signature provider to sign the TX
- “push_transaction” method in HTTP API
- nodeos executes all actions locally

Sending a transaction (2)

- nodeos executed the *speculative* TX locally with success
- nodeos broadcasts it to all P2P peers
- It also re-broadcasts all incoming speculative TX to its peers
- Speculative TX reaches producer nodes
- Active producer places transactions in a block and signs it
- Active producer broadcasts the block to all its P2P peers

Block Producers

- Up to 21 producers are placed in active schedule
- Active producer has a 6-second window to produce 12 blocks
- After 2/3+1 producers validate the block, it becomes irreversible

Smart contracts

- Every transaction executes WASM code in smart contracts
- Every account may or may not have code in it
- Contract keeps its state in multi-index tables in “RAM”
- RAM is part of blockchain state and is consistent across all nodes

Serialization

- Smart contracts operate with serialized data:
  - action arguments are serialized
  - RAM consists of serialized data rows and indexes
  - code manages deserialization and serialization according to its data definitions

ABI

- A smart contract account may have ABI in addition to WASM code (most of them do)
- ABI is a declaration of types, actions and tables, so that the client can serialize action arguments and deserialize RAM contents

Names

- In EOSIO, a “name” is a string of symbols that is mapped into uint64
- Up to 13 symbols: `[a-z1-5.]{1,13}`
- Name without dots is up to 12 symbols
- One-to-one mapping of uint64 field
- Everything is a “name”: account names, action names, permissions

Accounts

- Any account can create a new 12-symbol account without dots
- Short names are bought via auction
- Owners of short names can create names with dots: *chain* → *europe.chain*
- The rules can be changed in the system contract (*eosio*).

Permissions

- Every account has at least two permissions: **owner** and **active**.
- **Owner** permission is a parent of **active**.
- Parent permission can update itself and child permissions.

Permission structure

- Threshold: sum of weights required
- Keys: public keys and their weights
- Actors: accounts, their permissions and weights
- Waits: minimum delay (deprecated)

Simple permission example

```
$ alias mcleos='cleos -v -u https://mainnet.eosamsterdam.net'

$ mcleos get account cc32dninexxx
created: 2018-06-16T20:08:04.000
permissions:
     owner     1:    1 EOS7txFiAr3fFzmQNUbxpnPV5ApYjq22igdYVYatFHgo1Vx6Vr553
        active     1:    1 EOS7txFiAr3fFzmQNUbxpnPV5ApYjq22igdYVYatFHgo1Vx6Vr553
           watchdog     1:    1 EOS8C9tb8QQhZet6WWcYFCWDKHYfjC3W59ugHCD63s7LLDQx6JsNK
```

Multisig permission example

```
$ mcleos get account xeccustodian
created: 2019-06-28T19:01:13.500
permissions:
     owner     3:    1 cryptolions1@active, 1 dutcheosxxxx@active, 1
eosamsterdam@active, 1 eosbarcelona@active, 1 eosdublinwow@active
        active     3:    1 cryptolions1@active, 1 dutcheosxxxx@active, 1
eosamsterdam@active, 1 eosbarcelona@active, 1 eosdublinwow@active
```

Custom permission example

```
mcleos set account permission MYACCOUNT watchdog
EOS8C9tb8QQhZet6WWcYFCWDKHYfjC3W59ugHCD63s7LLDQx6JsNK

mcleos set action permission MYACCOUNT watchdoggiee setkv watchdog
mcleos set action permission MYACCOUNT watchdoggiee delkv watchdog

mcleos push action watchdoggiee setkv '["MYACCOUNT", "777", "777"]' -p
MYACCOUNT@watchdog

mcleos get table watchdoggiee MYACCOUNT kvs
```

Fungible Tokens

- System token is implemented in “eosio.token” contract
- Single contract can manage multiple *symbols*
- Symbol is a combination of name [A-Z]{1,7} and precision
- Examples:
  - 4,EOS
  - 4,XEC
  - 8,BTC

Fungible tokens

- Any account can have a token contract and manage one or more symbols
- Examples from EOS mainnet:
  - 8,BTC by `eosbettokens`
  - 4,BTC by `grandpacoins`
  - 8,BTC by `tokensbridge`

Example of token transfer

API of Jungle testnet

```
$ alias jcleos='cleos -v -u http://jungle2.cryptolions.io'
```

Private key is managed by keosd

```
$ cleos wallet unlock -n testnet
password: Unlocked: testnet
```

This account is a token contract for VOID token

```
$ jcleos push action onessusblock transfer '["cc32dninexxx", "escrowescrow", "0.5000
VOID", "1111"]' -p cc32dninexxx@active
Error 3050003: eosio_assert_message assertion failure
Error Details:
assertion failure with message: Cannot find deal ID: 1111
```

Attempted to transfer to a contract, but it rejected it

```
$ jcleos push action onessusblock transfer '["cc32dninexxx", "eosamsterdam", "0.5000
VOID", "1111"]' -p cc32dninexxx@active
executed transaction: ac5243cb34f472cb49778af217a811a885635cd1dc674468d4610a47f53531d9
136 bytes  436 us
#  onessusblock <= onessusblock::transfer
{"from":"cc32dninexxx","to":"eosamsterdam","quantity":"0.5000 VOID","memo":"1111"}
#  cc32dninexxx <= onessusblock::transfer
{"from":"cc32dninexxx","to":"eosamsterdam","quantity":"0.5000 VOID","memo":"1111"}
#  eosamsterdam <= onessusblock::transfer
{"from":"cc32dninexxx","to":"eosamsterdam","quantity":"0.5000 VOID","memo":"1111"}
warning: transaction executed locally, but may not be confirmed by the network yet
   ]
```

Our API node evaluated the transaction as successful and broadcast it to its p2p peers. This resulted in 3 notifications.

Desktop and mobile wallets

- Scatter is a de-facto standard for desktops
- A number of mobile wallets: Lynx, Meet.one, Tokenpocket, ...
- Wallets manage private keys, interact with dApp websites, compose transactions, and sign them.

Blockchain explorers

- https://bloks.io/
- https://www.eosx.io/
- https://eosflare.io/

RAM, CPU, NET

- Every account needs a few KB of RAM and some CPU+NET stake
- RAM is used to keep token balances and other smart contract data. Normally the originator of a transaction allocates the RAM if the contract needs it
- CPU+NET stake determines how frequently the account is able to send transactions