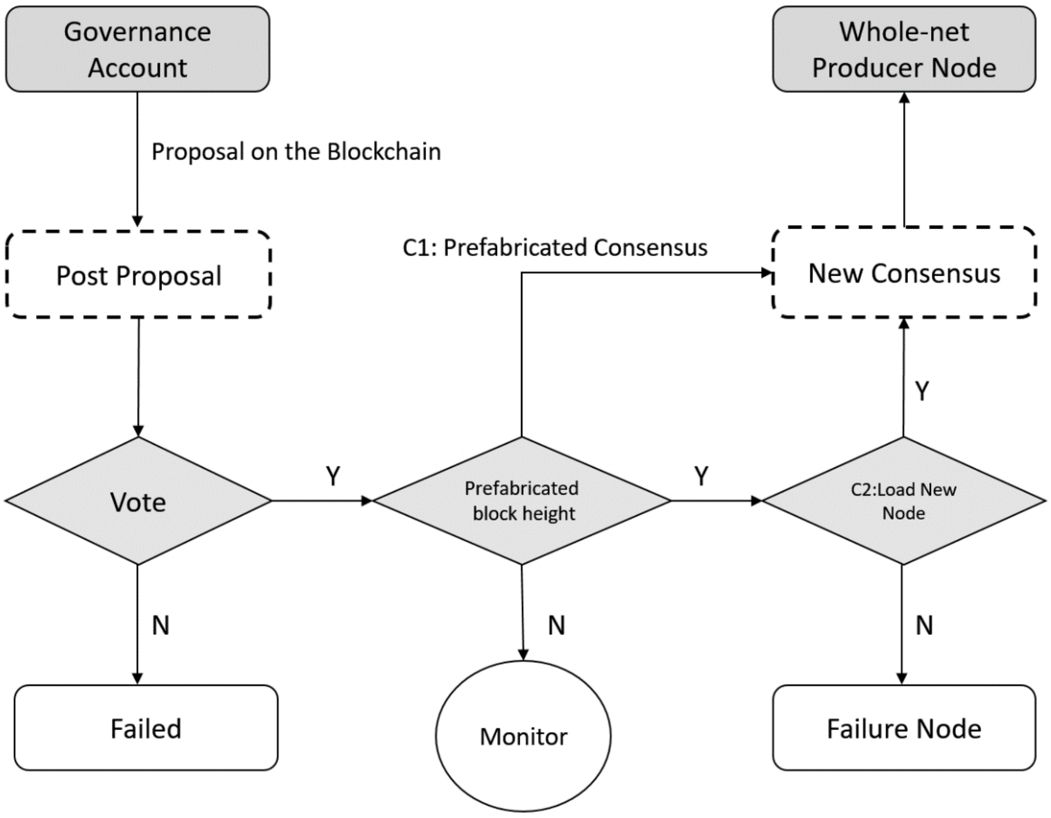
Dimension(EON) Project Planetary Landing（Phase 4）

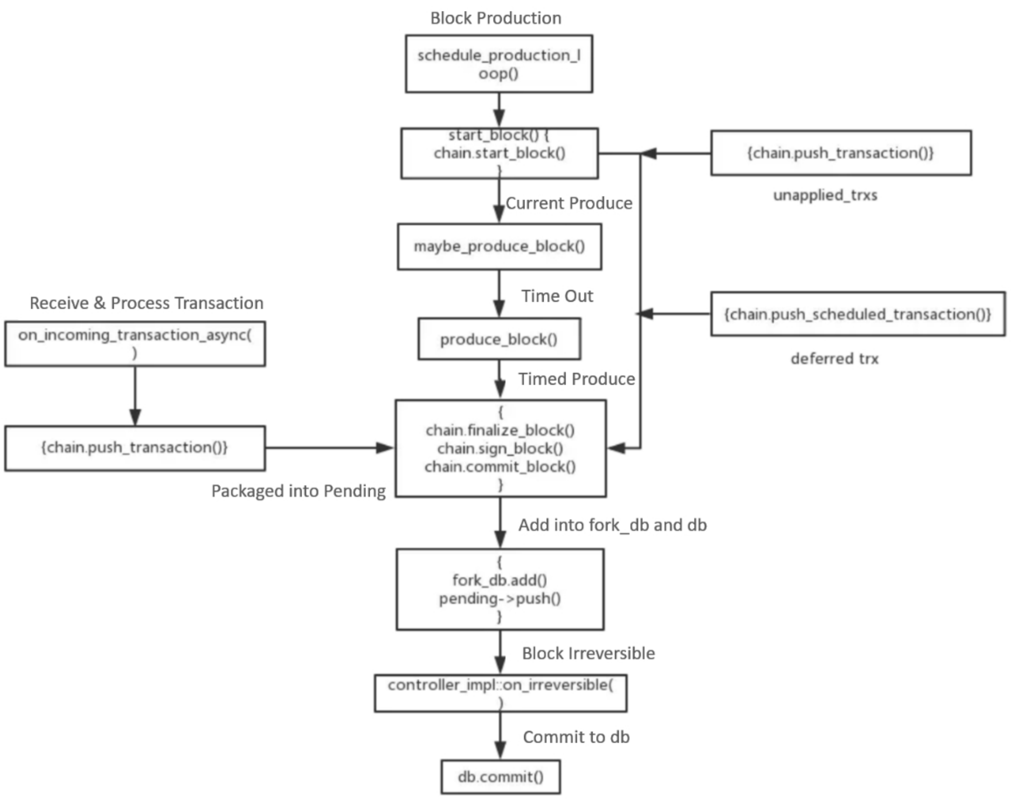
PROJECT PLANETARY LANDING
INSPIRATION, EXPLORATION & TRANSFORMATION
DIMENSION 2019
dimensionchain.io
Dimension

Vote mechanism

We introduce a proposal mechanism on ConsensusX, which is applied by the node to be joined or the node governance committee to publicly vote on the proposal via the verification node. The verification node may vote for “yes”, “no”, “strongly negative” or “abstain” in the specific time period. At least half of the voters must vote for "yes" and the proposal can be passed. If one-quarter of the verification nodes vote "strongly against" or one-third of the verification nodes vote "no", the proposal will be rejected. It also provides extensions to the approval process, supporting the corresponding contract call interface and command line interface.

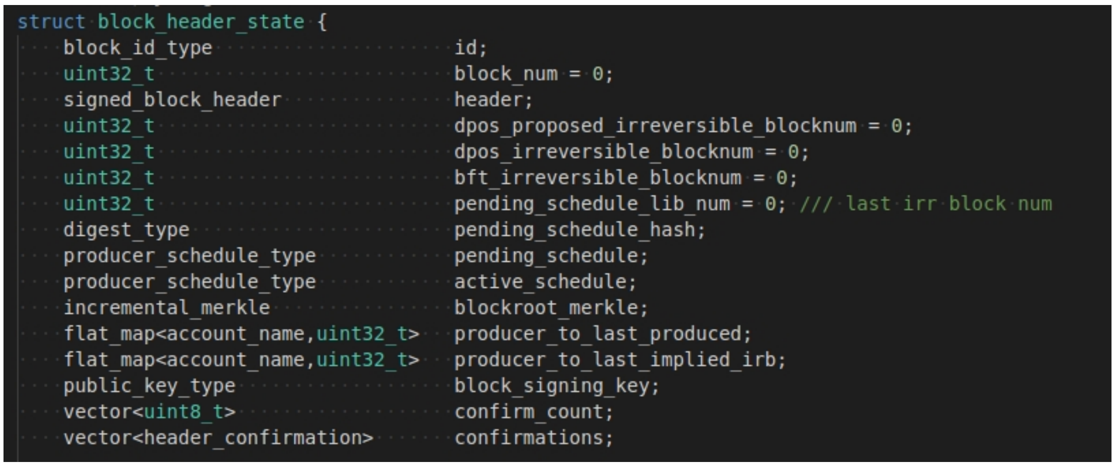


Dimension will judge whether to switch the consensus by reading the data in the chain in start_block()


Block Production
schedule_production_loop()
start_block() { chain.start_block() }
{chain.push_transaction()}
unapplied_trxs
Current Produce
maybe_produce_block()
Time Out
Receive & Process Transaction
on_incoming_transaction_async()
produce_block()
{chain.push_scheduled_transaction()}
deferred trx
Timed Produce
{chain.push_transaction()}
{ chain.finalize_block() chain.sign_block() chain.commit_block() }
Packaged into Pending
Add into fork_db and db
{ fork_db.add() pending->push() }
Block Irreversible
controller_impl::on_irreversible()
Commit to db
db.commit()

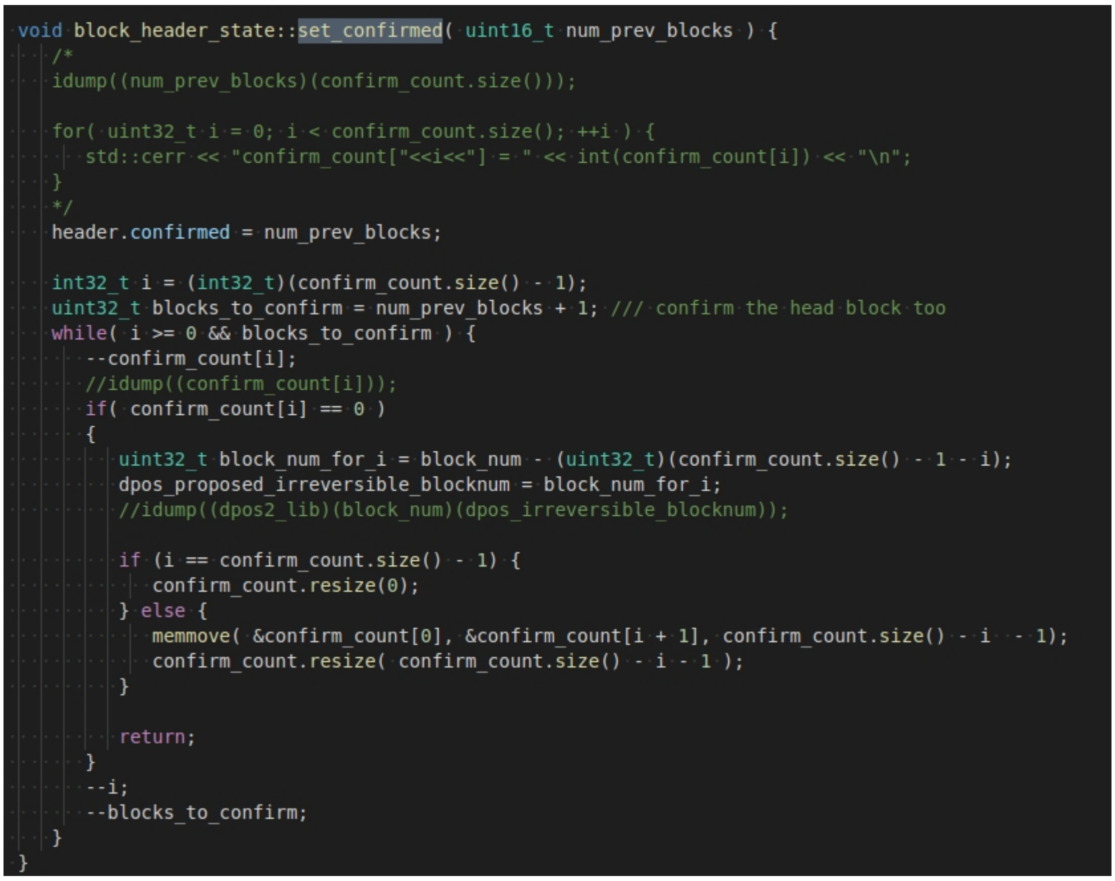
Interface Description

Different consensuses may require custom data structures at the head of the block, and new consensus needs to redefine this part of data.



```
struct block_header_state {
    block_id_type                   id;
    uint32_t                        block_num = 0;
    signed_block_header             header;
    uint32_t                        dpos_proposed_irreversible_blocknum = 0;
    uint32_t                        dpos_irreversible_blocknum = 0;
    uint32_t                        bft_irreversible_blocknum = 0;
    uint32_t                        pending_schedule_lib_num = 0; /// last irr block num
    digest_type                     pending_schedule_hash;
    producer_schedule_type          pending_schedule;
    producer_schedule_type          active_schedule;
    incremental_merkle              blockroot_merkle;
    flat_map<account_name,uint32_t> producer_to_last_produced;
    flat_map<account_name,uint32_t> producer_to_last_implied_irb;
    public_key_type                 block_signing_key;
    vector<uint8_t>                 confirm_count;
    vector<header_confirmation>     confirmations;
```

Under the mechanism of DPOS, the requirements for confirming a block in different algorithms are different. Only when the requirements of the consensus algorithm are met, the block can be agreed and enter an irreversible state. Dimension integrates this part. Each node uses different acknowledgment algorithms for the block according to different consensus when it produce a block and receives a block for confirmation. The corresponding function is implemented in set_confirmed().



```
void block_header_state::set_confirmed( uint16_t num_prev_blocks ) {
   /*
   idump((num_prev_blocks)(confirm_count.size()));

   for( uint32_t i = 0; i < confirm_count.size(); ++i ) {
      std::cerr << "confirm_count["<<i<<"] = " << int(confirm_count[i]) << "\n";
   }
   */
   header.confirmed = num_prev_blocks;

   int32_t i = (int32_t)(confirm_count.size() - 1);
   uint32_t blocks_to_confirm = num_prev_blocks + 1; /// confirm the head block too
   while( i >= 0 && blocks_to_confirm ) {
      --confirm_count[i];
      //idump((confirm_count[i]));
      if( confirm_count[i] == 0 )
      {
         uint32_t block_num_for_i = block_num - (uint32_t)(confirm_count.size() - 1 - i);
         dpos_proposed_irreversible_blocknum = block_num_for_i;
         //idump((dpos2_lib)(block_num)(dpos_irreversible_blocknum));

         if (i == confirm_count.size() - 1) {
            confirm_count.resize(0);
         } else {
            memmove( &confirm_count[0], &confirm_count[i + 1], confirm_count.size() - i  - 1);
            confirm_count.resize( confirm_count.size() - i - 1 );
         }

         return;
      }
      --i;
      --blocks_to_confirm;
   }
}
```

The node calls this function to change the status value of the block header when its produce or received block is validated. Different consensuses should redefine the function according to the requirements of the consensus itself, and call the relevant function according to the consensus type when the node produces or receive the block.