Design of LinkChain

LinkChain Team

Content

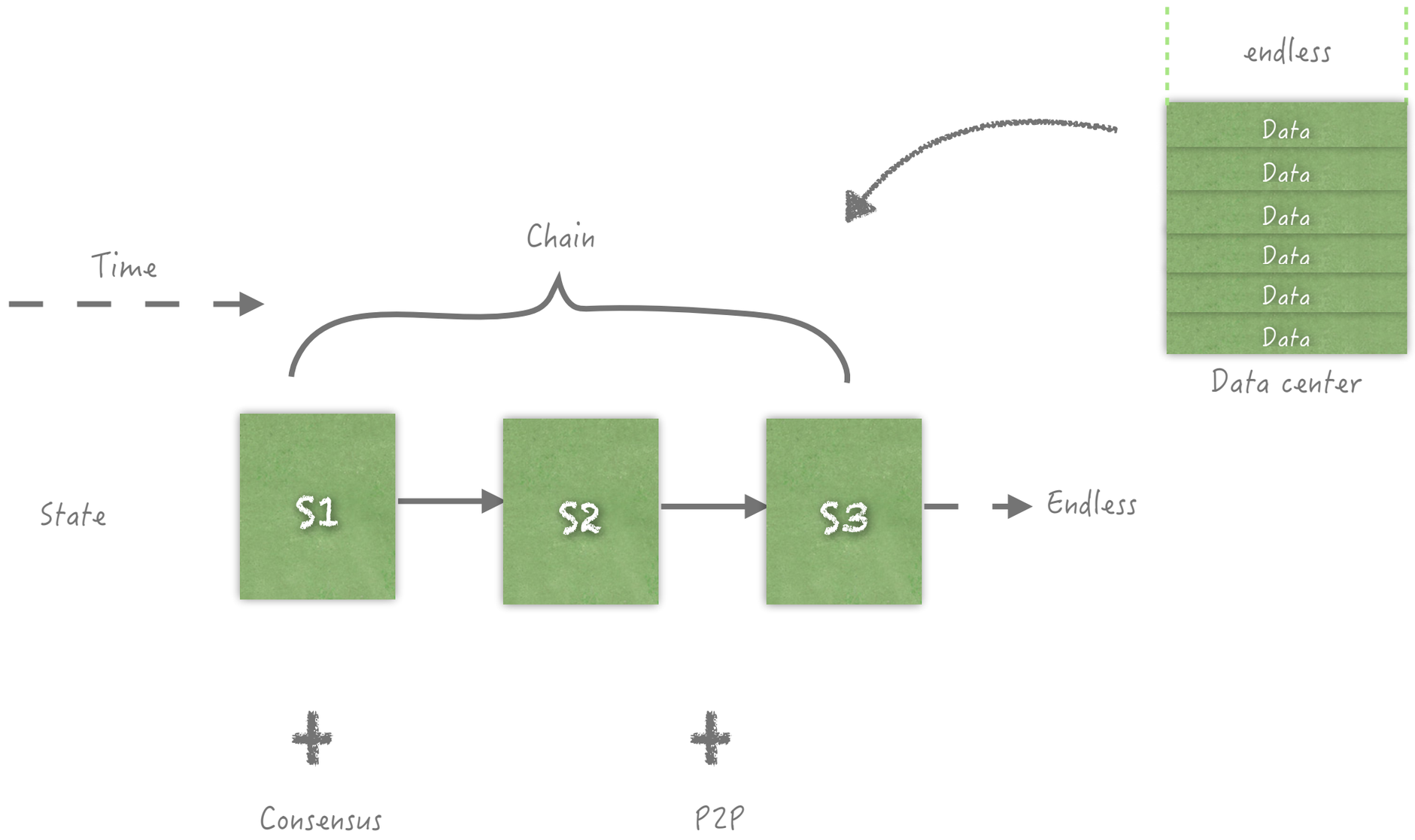
- LinkChain is in progress
- Detailed Design of LinkChain
- More

LinkChain is in progress

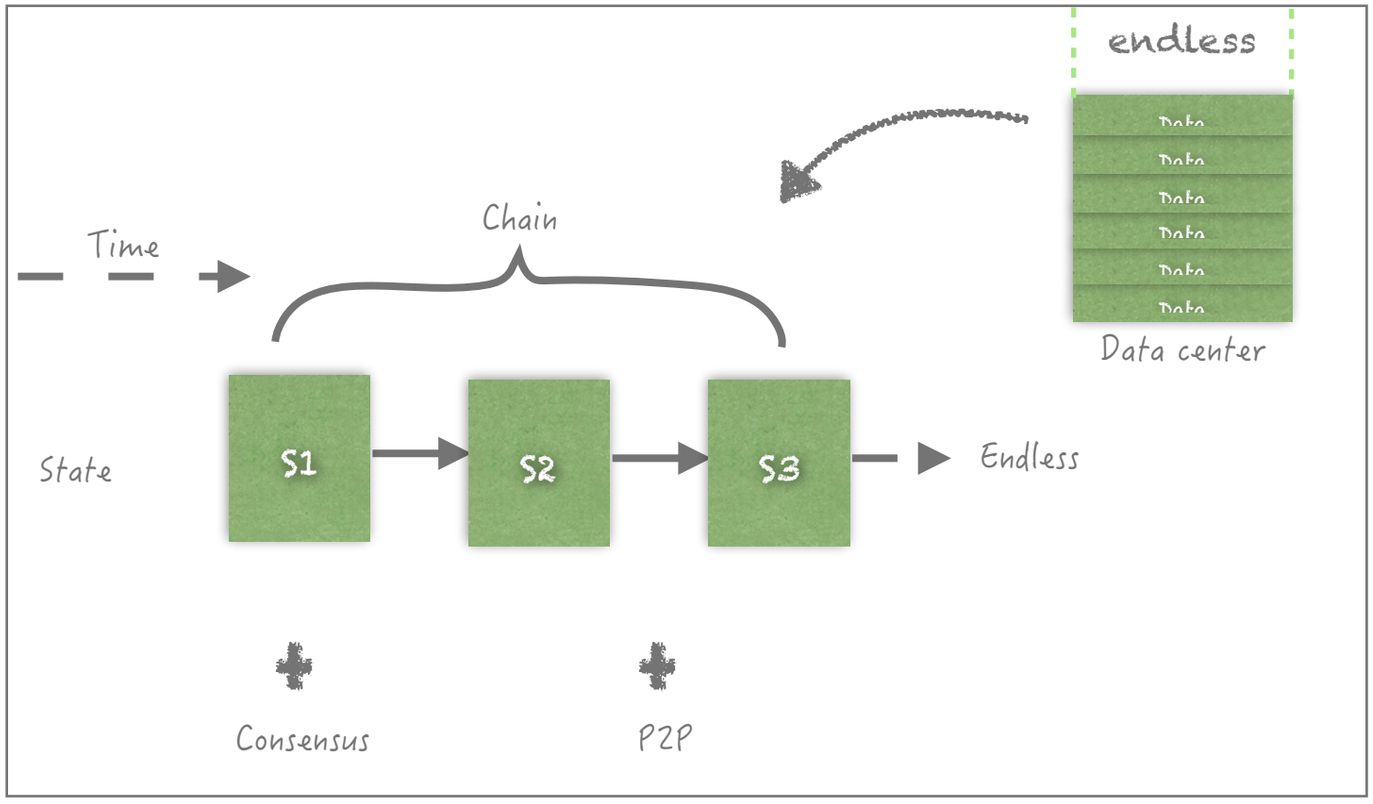
- LinkChain is evolving
- Focus on the Final Form of LinkChain
- Different development levels of LinkChain

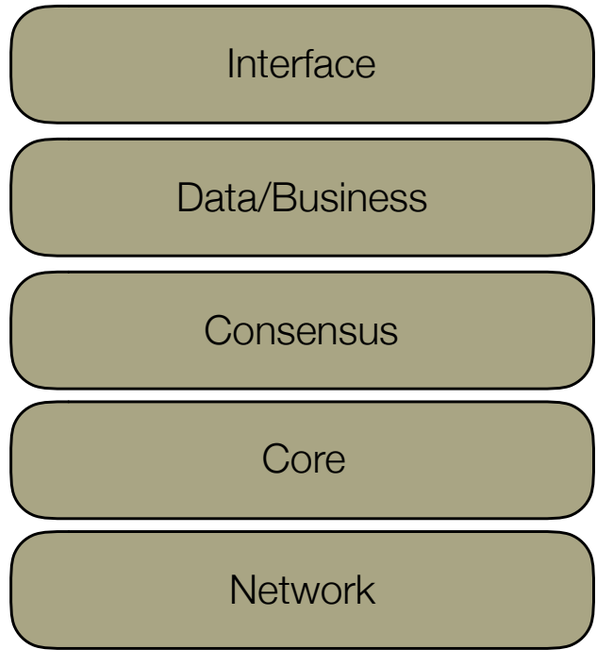
Detail Of LinkChain

The Basic Model



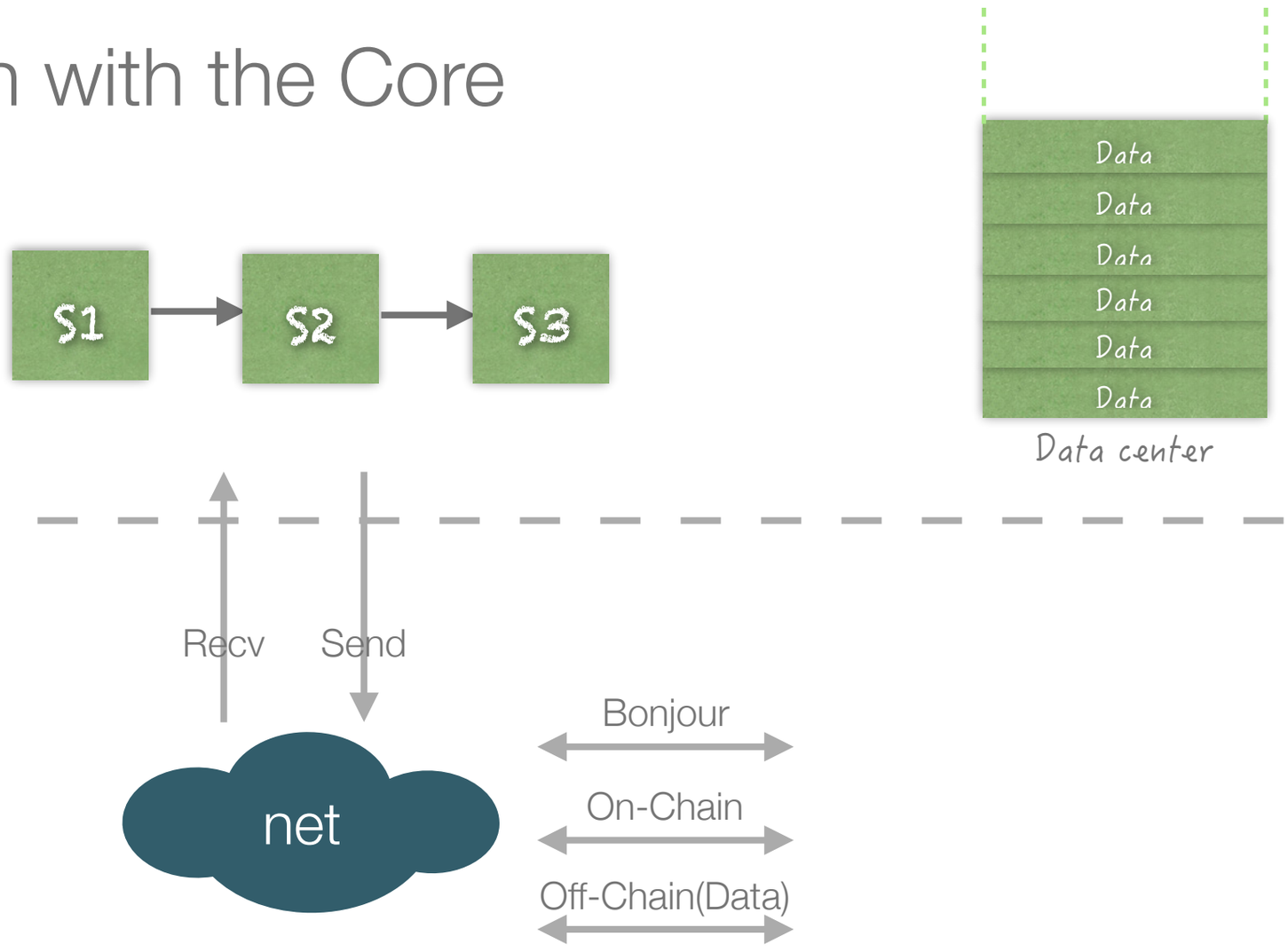
The Architecture



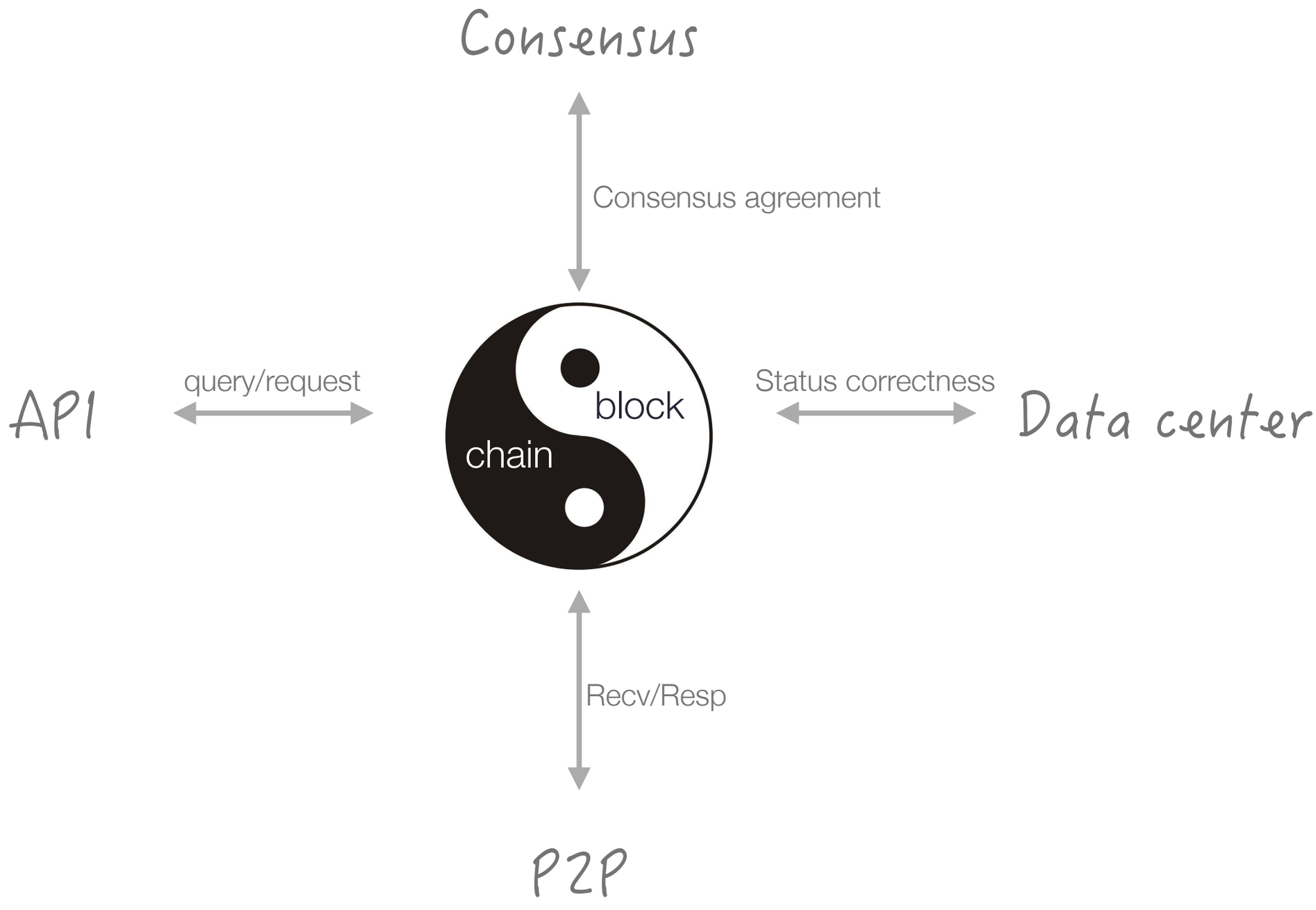


Network layer

- Node Discover
- Data exchange
- Collaboration with the Core



The Core layer

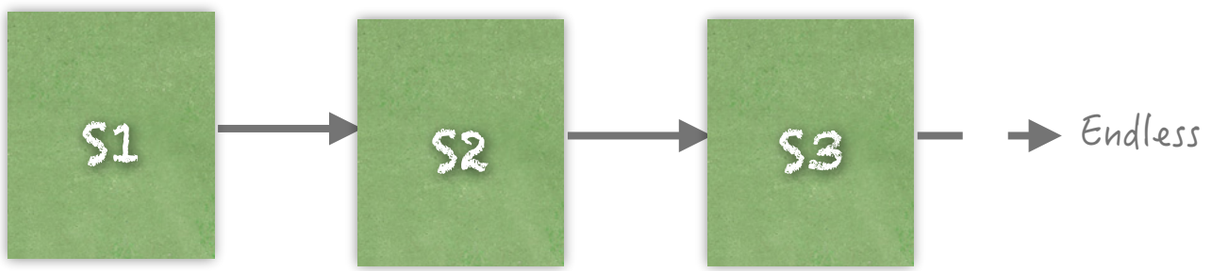


The Core layer

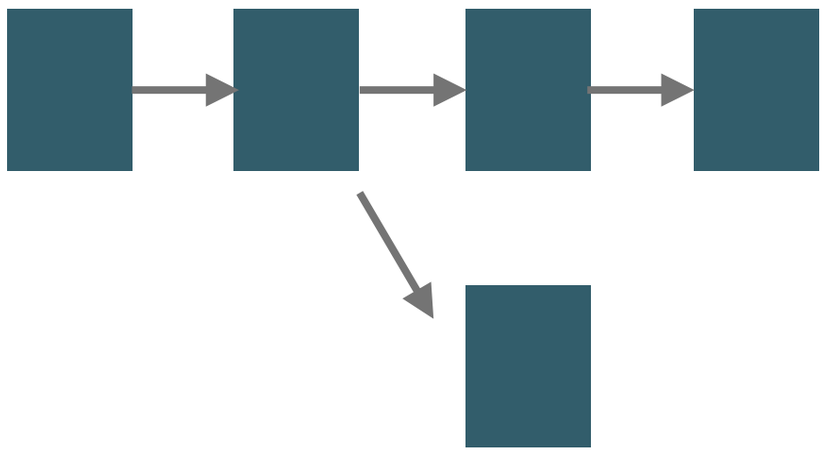
- Maintain the StateChain
- Work with DataCenter
- Work with Consensus
- Work with Network
- Interface for query and subsidiary functions

The Chain

- All about the Block organization
- Find out the Main Chain
- Deal with Fork

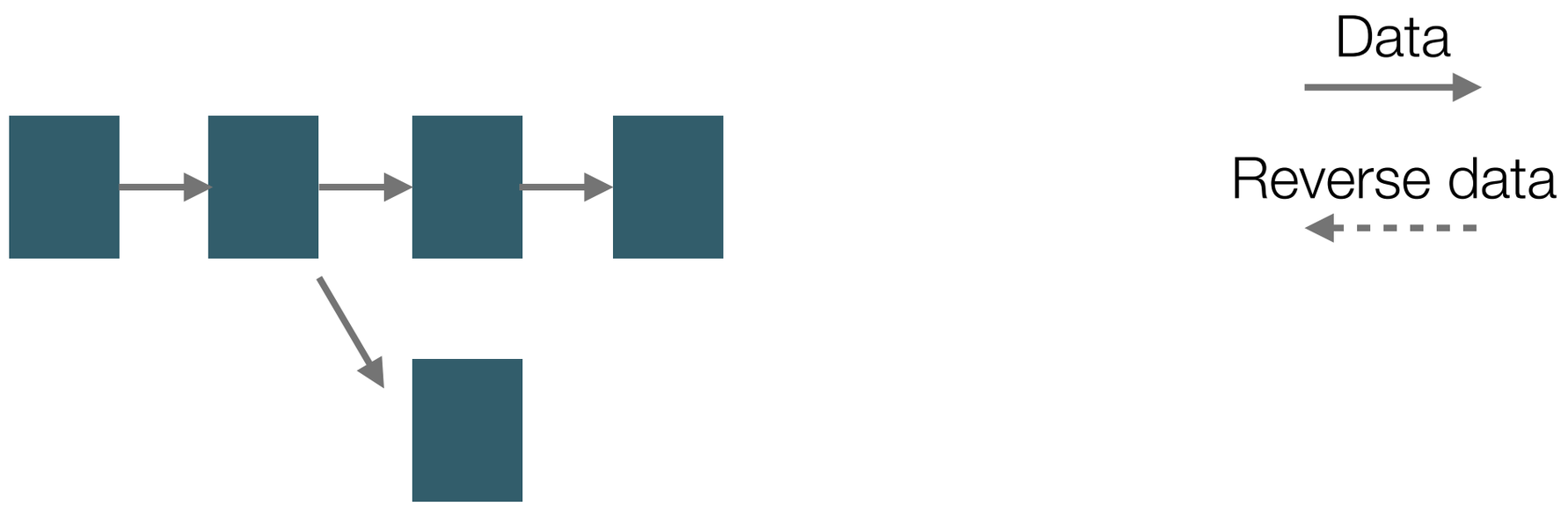


Deal with Fork



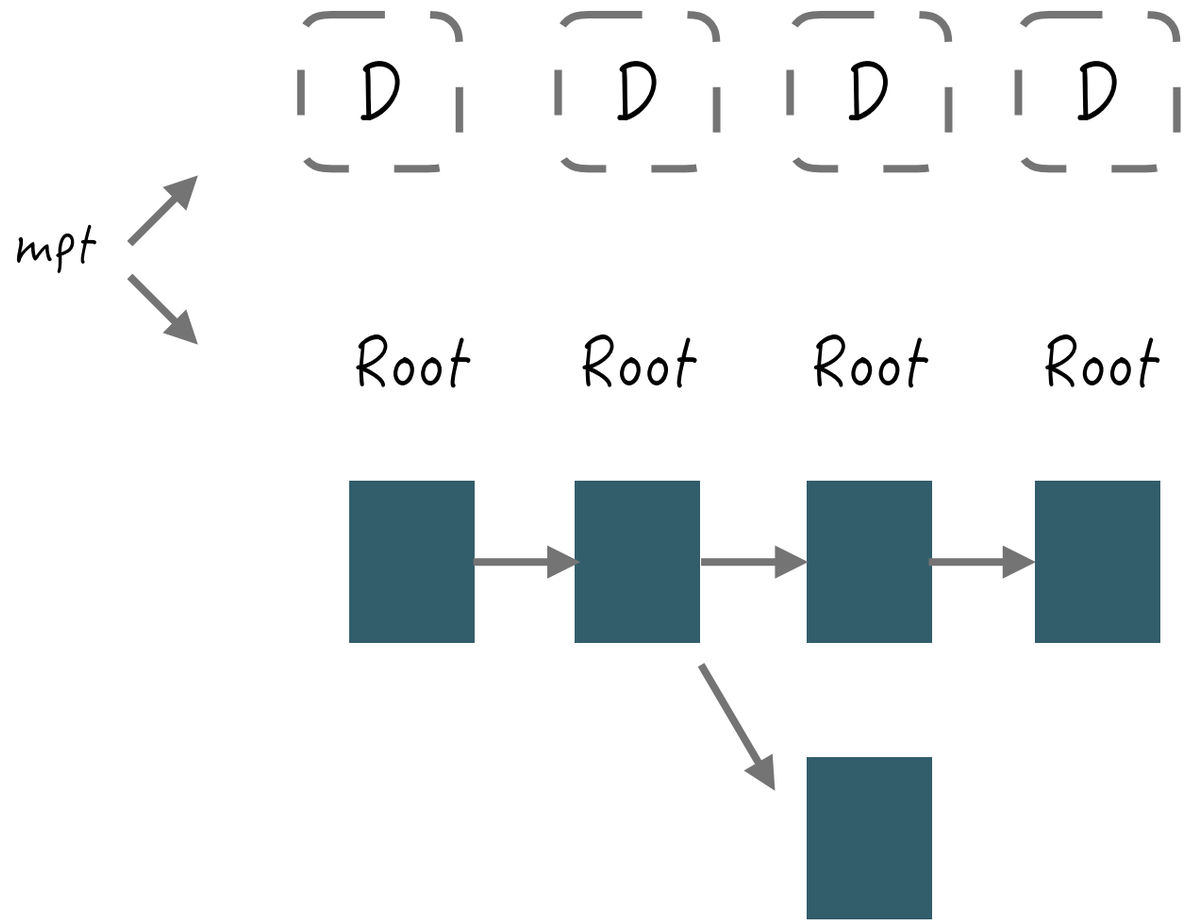
Deal with Fork

- BTC: Rollback



Deal with Fork

- ETH: StateDB



Locate

Recalculate

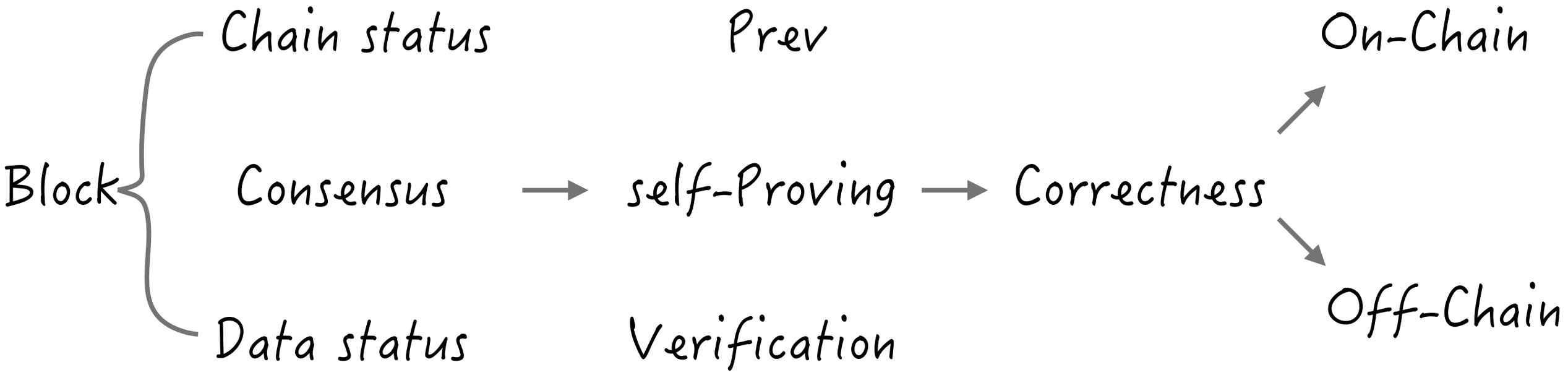
Challenge

- Can blockchain eliminate the fork?

The Block

- All about the status
- Chain status
- Consensus
- Data status

The Block Processing Flow



The Block Code Definition(Sample)

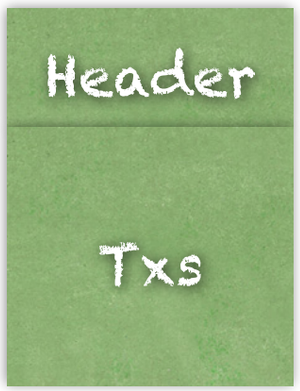
```
type BlockHeader struct {
        // Version of the block.  This is not the same as the protocol version.
        Version uint32 `json:"version,int"`

        //the height of block
        Height uint32 `json:"height,int"`

        // Time the block was created.  This is, unfortunately, encoded as a
        // uint32 on the wire and therefore is limited to 2106.
        Time time.Time `json:"Time"`

        // Nonce used to generate the block.
        Nonce uint32 `json:"nonce"`

        // Difficulty target for the block.
        Difficulty uint32 `json:"difficulty"`

        // Hash of the previous block header in the block chain.
        Prev BlockID `json:"prev"`

        // Merkle tree reference to hash of all transactions for the block.
        TxRoot TreeID `json:"txroot"`

        // The status of the whole system
        Status TreeID `json:"status"`

        // The sign of miner
        Sign Signature `json:"sign"`

        // Data used to extenion the block.
        Data []byte `json:"data"`

}
```

The Data layer

- Tx
- Account

Tx

- Ledger system
- Smart contract platform

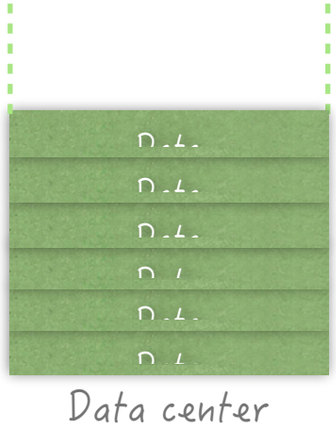


Block

Tx

- The input of state transition

Tx is the Input



Tx

- All about the status driven
- Control Info
- Data load

➡ Who issued the tx?

➡ The purpose of the tx

The Tx Code Definition(Sample)

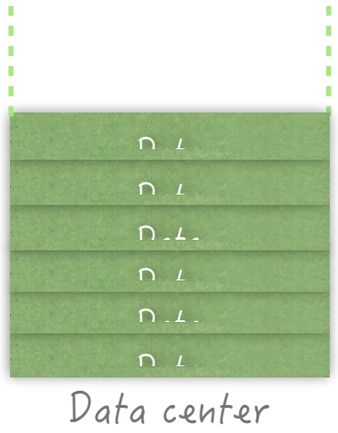
```
type Transaction struct {
        // The version of the Transaction.  This is not the same as the Blocks version.
        Version uint32 `json:"version"`

        // The type of the Transaction.
        Type uint32 `json:"type"`

        //The accounts of the Transaction related to inputs.
        From TransactionFrom `json:"from"`

        //The accounts of the Transaction related to outputs.
        To TransactionTo `json:"to"`

        //The Sign of From, which is represent the Coins each Froms if not can put.
        Sign []Signature `json:"signs"`

        //The extra feild of Transaction.
        Data []byte `json:"data"`

}
```

Challenge

- Can the transaction be off-chain ?

The Account

- BTC: Weak/no Account system
- ETH: Account system



Account is the most choice

The Account Code Definition(RollChain Sample)

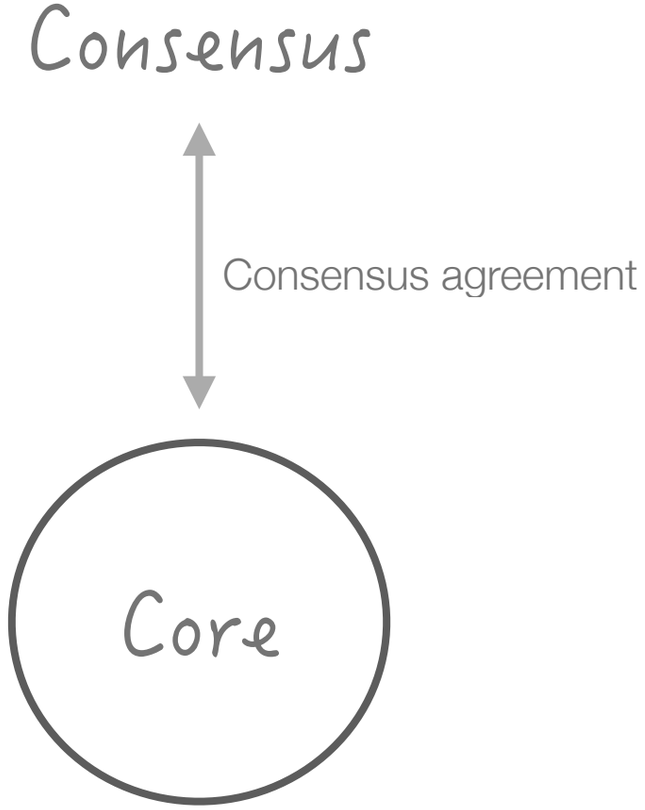
```
type Account struct {
        Id          AccountID `json:"accountId"`
        AccountType uint32    `json:"accountType"`
        UTXOs       []UTXO    `json:"AccountUXTO"`
        Clear       ClearTime `json:"clearTime"`
        SecurityId  AccountID `json:"securityId"`
        StorageRoot TreeID    `json:"storageRoot"`
        CodeHash    math.Hash `json:"codeHash"`
}
```

Challenge

- ETH Account + BTC UTXO.
  How to understand RollChain
  combine the two together?

Consensus layer

- Consensus focus on the correctness of block
- The pluggability of Consensus



More

- General platform(ETH, EOS) vs Dedicated system, which one is better?

  ✓ performance, security, administration

- How to design a general Token system?

  ✓ Colored Coin

  ✓ Smart Contract

Thank you!