INFINITY
BLOCKCHAIN LABS

Ethereum

for noobs

Clique - Proof of Authority

Hội anh em blockchain "thiện lành"

Profile

Hoang Tung

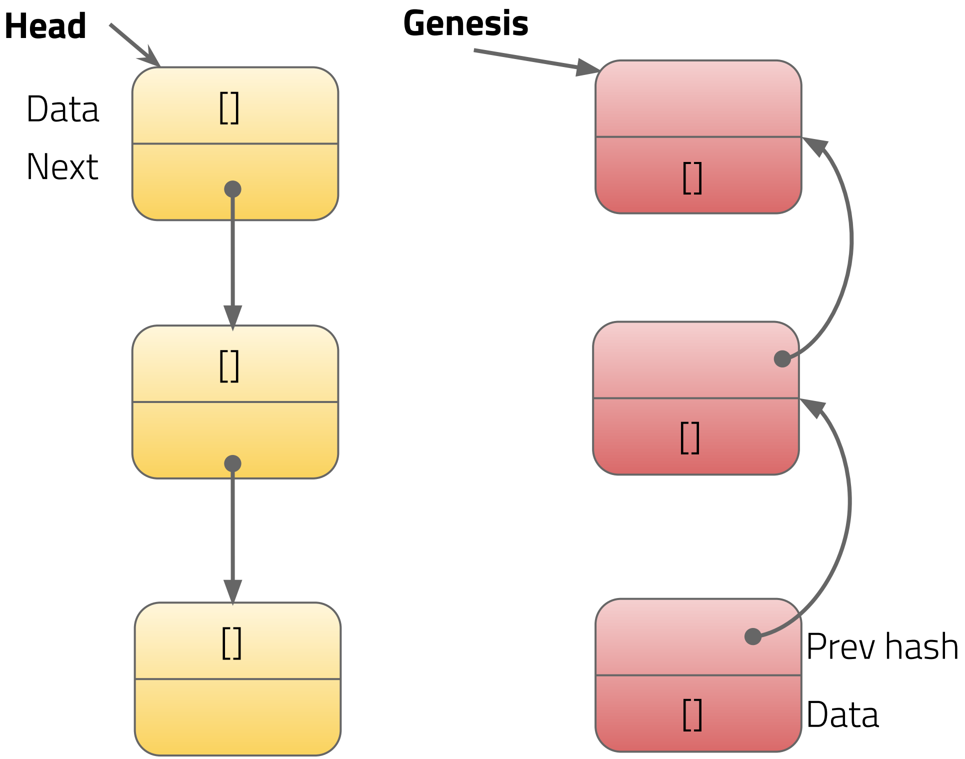
- **Contact:**
  - Email: hoangtung.cse@gmail.com
  - Fb: https://www.facebook.com/onghoang.mt
- **Education:**
  - Bach Khoa University - Computer Science - Graduated with Excellence.
- **Work Experience:**
  - 2 Years - Senior Developer of High Performance Blockchain - Infinity Blockchain Labs.
  - One Year - Smart Contract Developer - Infinity Blockchain Labs.

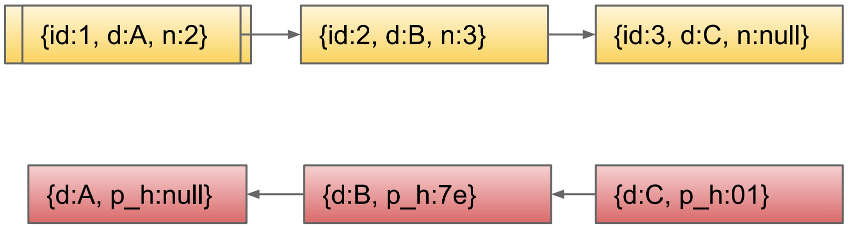
Content

1. Consensus Protocol
2. Basic Types of Consensus
3. Proof of Authority
4. Clique in Geth
5. How To Create New Consensus In Geth

1. Consensus Protocol

What is blockchain?



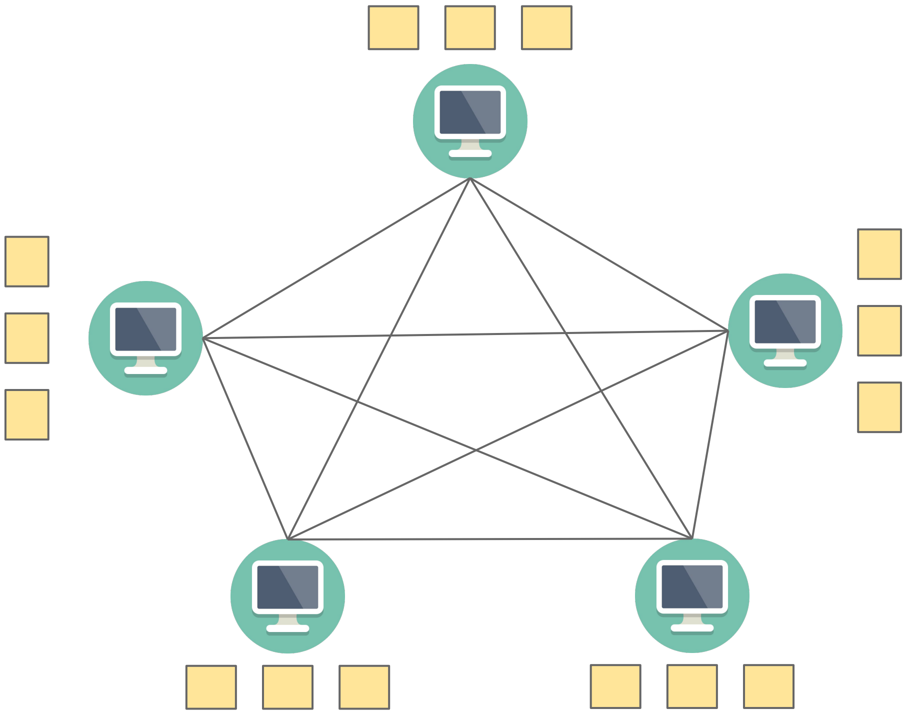


Immutable

Blockchain is a data structure:

- **Similar to linked list.**
- **Pointer is the hash of the previous elements.**

What is blockchain?



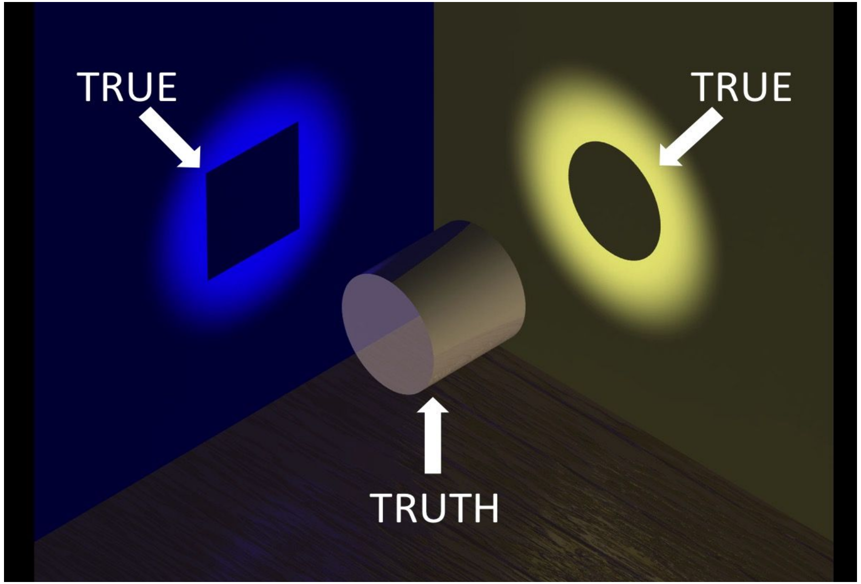
Peer to peer network

- No center server.
- A node connects directly to other nodes.
- Each node maintains a copy of the blocks.

Distributed

Some Questions

1. Who is chosen to be a block creator of network?
2. What are the benefits of being a block creator?
3. How to determine whether a block is accepted?



Consensus Protocols

https://www.leadstrat.com/blog/5-finger-consensus/

A protocol that is used to achieve the common agreement.

Consensus rules are a specific set of rules that nodes on the network will ensure valid blocks.

Consensus protocols are one of the most important and revolutionary aspects of blockchain technology.

2. Basic Types of Consensus

Proof of Work

1. Who is chosen to be a block creator of network?

➢ *The first miner that solves the complex mathematical challenge.*

2. What are the benefits of being a block creator?

➢ *The miner gets coinbase reward and transaction fee.*

Proof of Stake

1. Who is chosen to be a block creator of network?

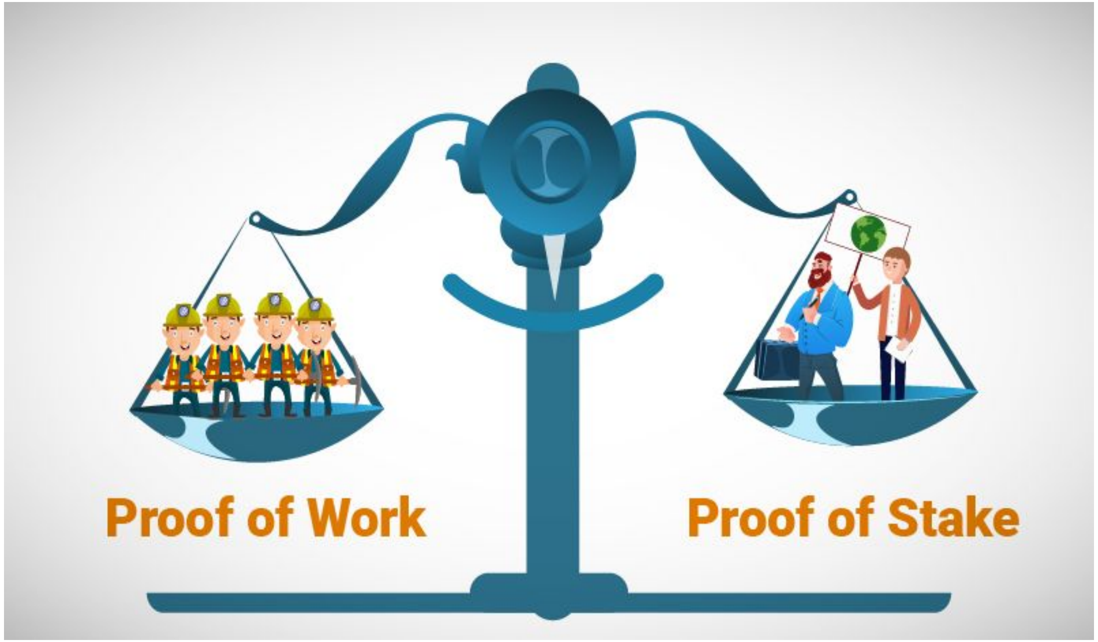
➢ *The block creator will be randomly chosen by an algorithm based on the their stake.*

2. What are the benefits of being a block creator?

➢ *The minter take the transaction fee.*

PoW vs. PoS

1. Security
2. Power Consumption
3. Algorithm Complication
4. Decentralization



3. Proof of Authority

Proof of Authority

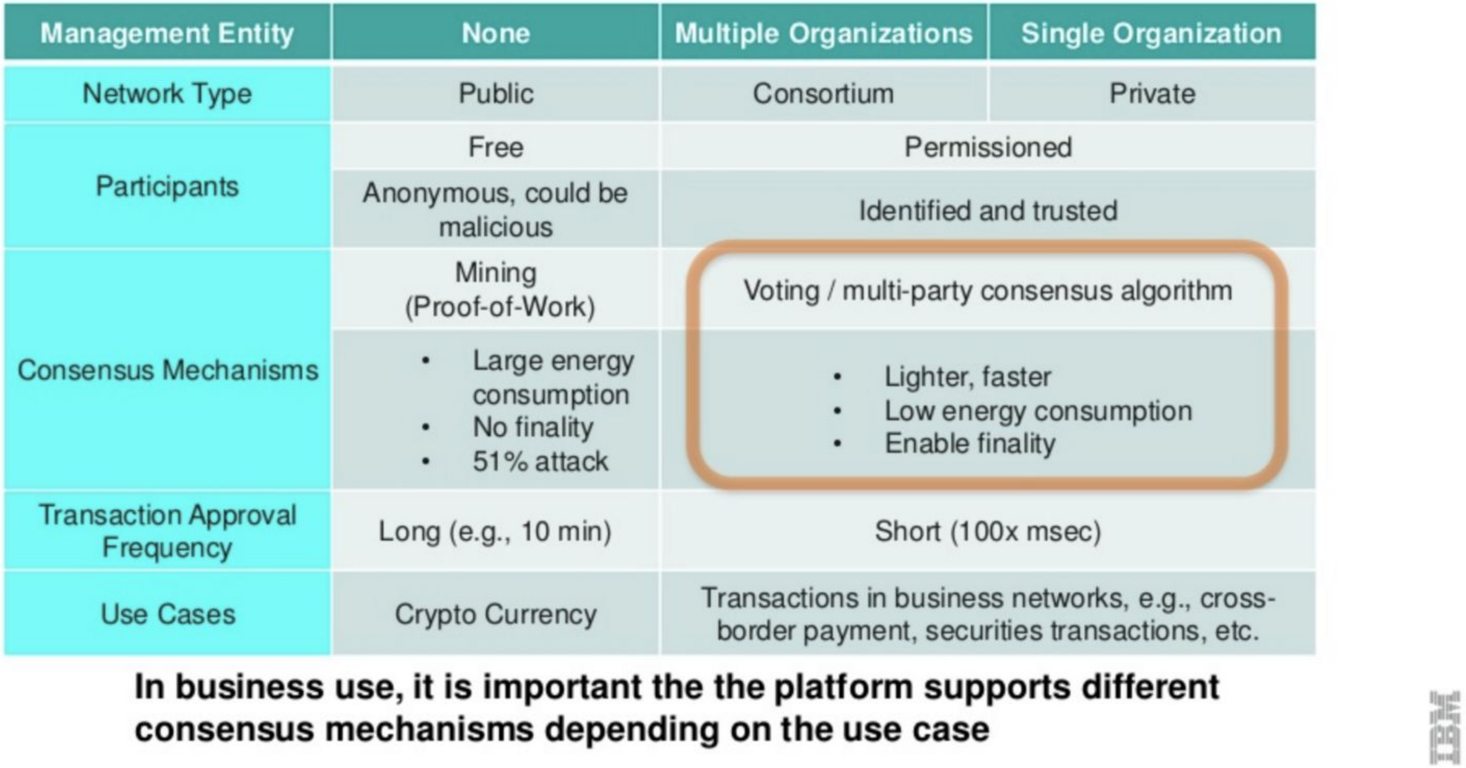
A. Use cases:

➢ It's suitable for business model of single or multiple organizations.

B. How to work:

1. There is a predefined set of known parties, called authorities.
2. The block creator will be chosen by a simple rotating algorithm.

Comparison of consensus approaches



Management Entity	None	Multiple Organizations	Single Organization
Network Type	Public	Consortium	Private
Participants	Free	Permissioned	
	Anonymous, could be malicious	Identified and trusted	
Consensus Mechanisms	Mining (Proof-of-Work)	Voting / multi-party consensus algorithm	
	• Large energy consumption • No finality • 51% attack	• Lighter, faster • Low energy consumption • Enable finality	
Transaction Approval Frequency	Long (e.g., 10 min)	Short (100x msec)	
Use Cases	Crypto Currency	Transactions in business networks, e.g., cross-border payment, securities transactions, etc.	

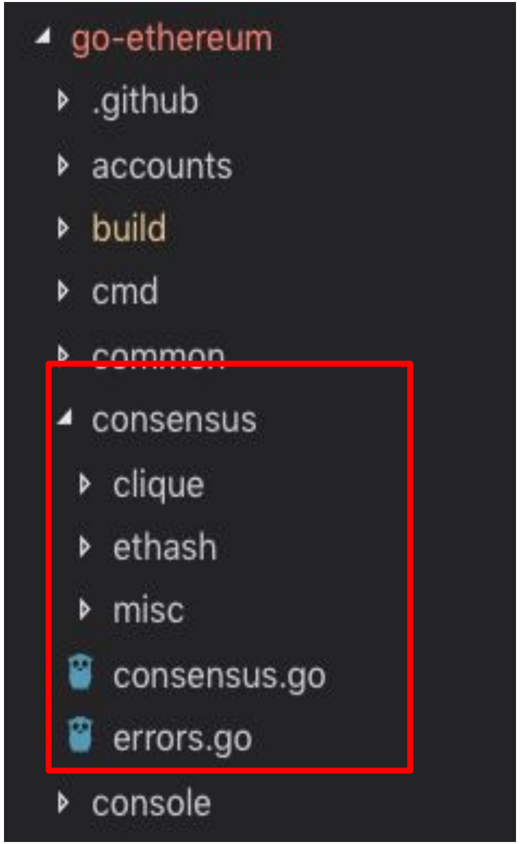
In business use, it is important the the platform supports different consensus mechanisms depending on the use case

https://www.slideshare.net/DiegoDiaz49/1-ibm-blockchain-explained

4. Clique in Geth

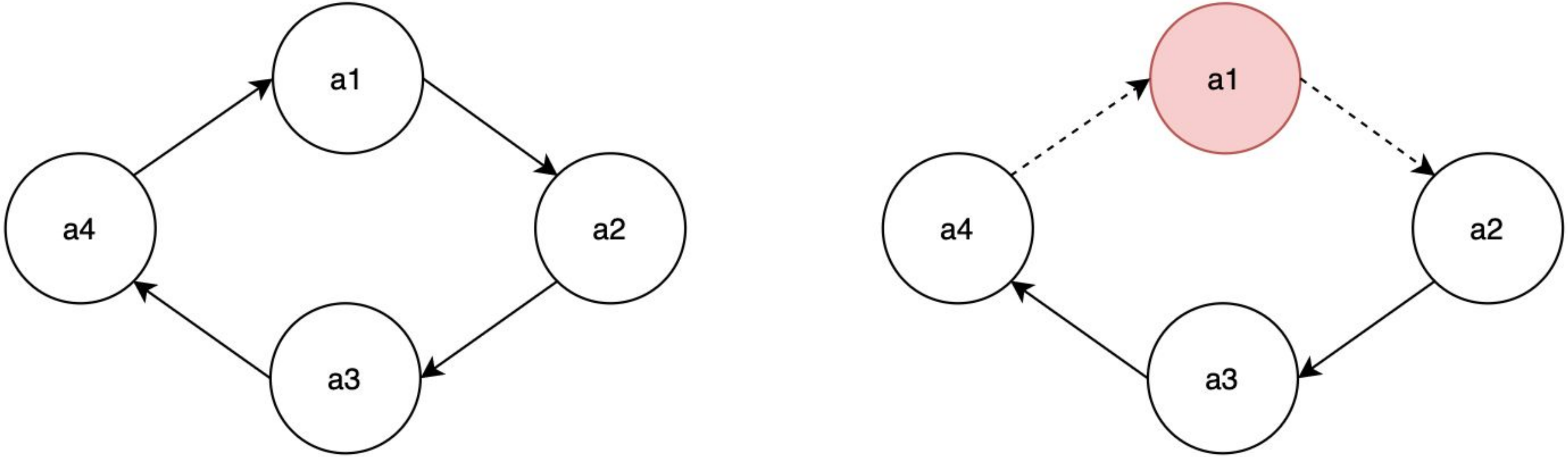
Consensus in Geth

1. Consensus in Geth is a separate package which implements different Ethereum consensus engines.
2. There are two available consensuses in Geth:
   a. Clique - Proof of Authority
   b. Ethash - Proof of Work
3. Can plug new consensus into Geth.



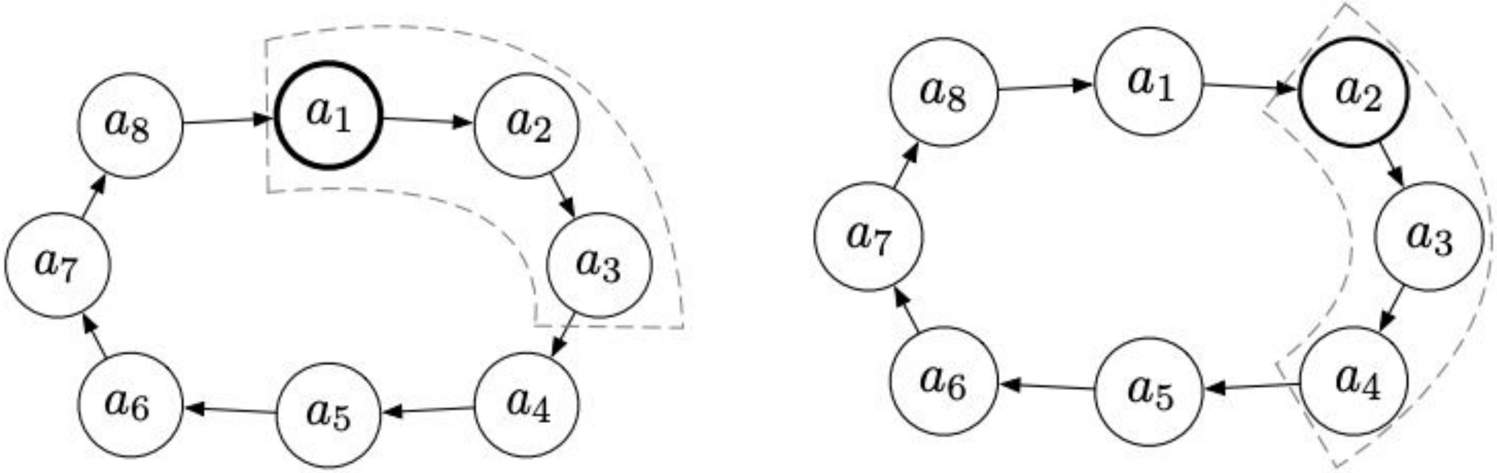
```
◢ go-ethereum
  ▷ .github
  ▷ accounts
  ▷ build
  ▷ cmd
  ▷ common
  ◢ consensus
    ▷ clique
    ▷ ethash
    ▷ misc
      consensus.go
      errors.go
  ▷ console
```

Clique Consensus



Simple rotating algorithm

Clique Consensus

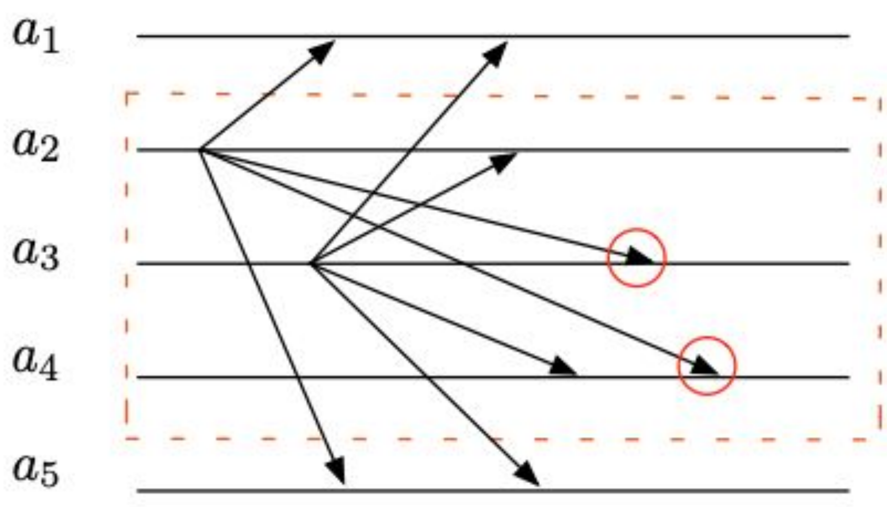


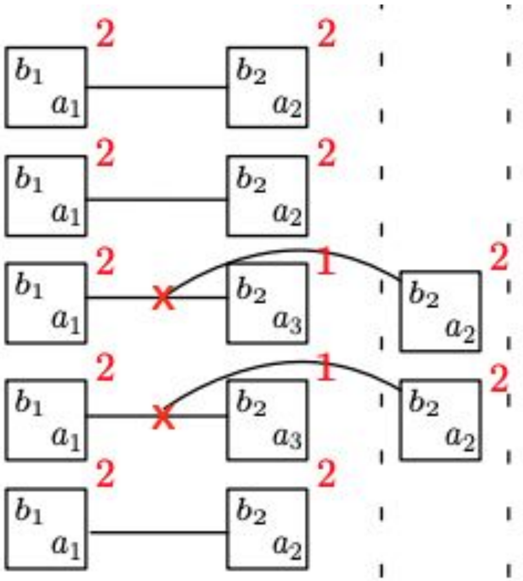
(a) Time t1

(b) Time t2

There are N = 8 authorities, hence N – (N/2 + 1) = 3 authorities allowed to propose a block at each step.

Fork in Clique





The GHOST protocol resolves the forks.

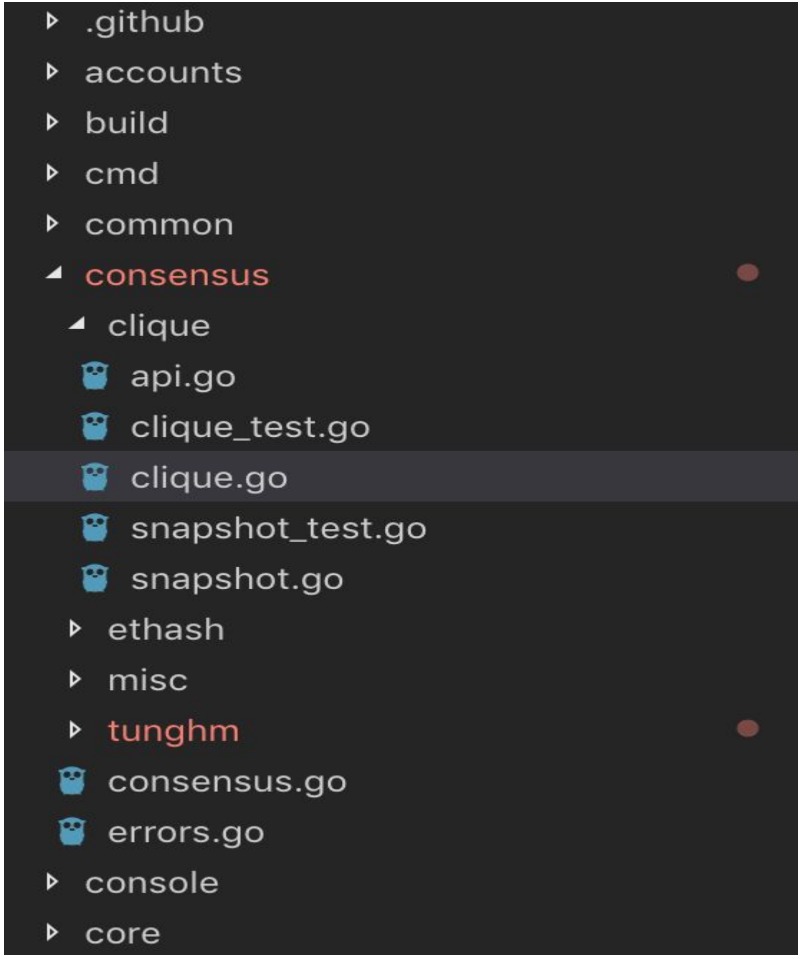
Clique Consensus

1. Clique is the PoA algorithm implemented in Geth.
2. The algorithm proceeds in epochs which are identified by genesis block.
3. When a new epoch starts, a special transition block is broadcasted which specifies the set of authorities.
4. Each authority is only allowed to propose a block every N/2+ 1 blocks. N = len(set of authorities)
5. As more authorities can propose a block during each step, forks can occur.
6. Block difficulty of leader is 2, block difficulty of others is 1.
7. Leader is chosen by <<block_num % len(set of authorities) = leader's offset>>.

Explore Go Ethereum

https://github.com/ethereum/go-ethereum

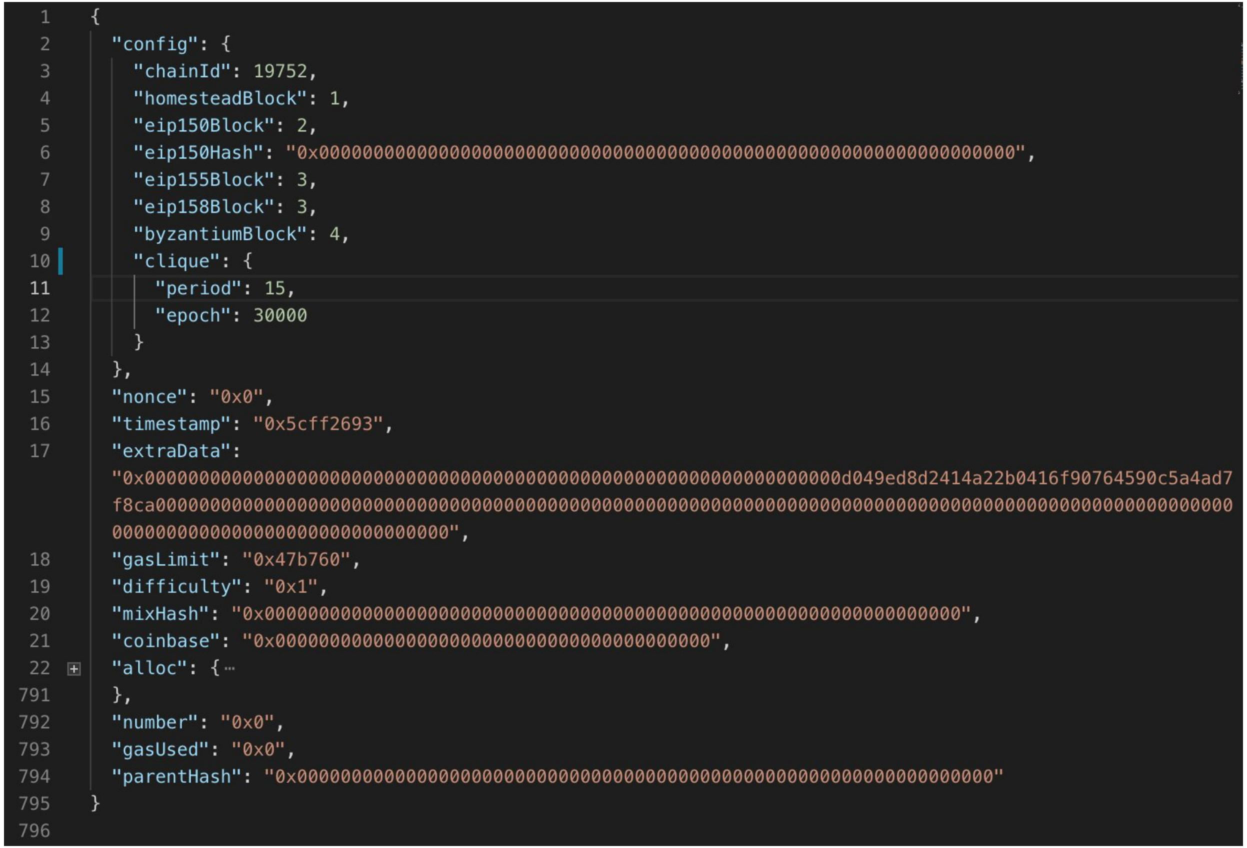
- Clique is the PoA algorithm implemented in Geth.



```
▸ .github
▸ accounts
▸ build
▸ cmd
▸ common
◢ consensus
  ◢ clique
      api.go
      clique_test.go
      clique.go
      snapshot_test.go
      snapshot.go
  ▸ ethash
  ▸ misc
  ▸ tunghm
    consensus.go
    errors.go
▸ console
▸ core
```

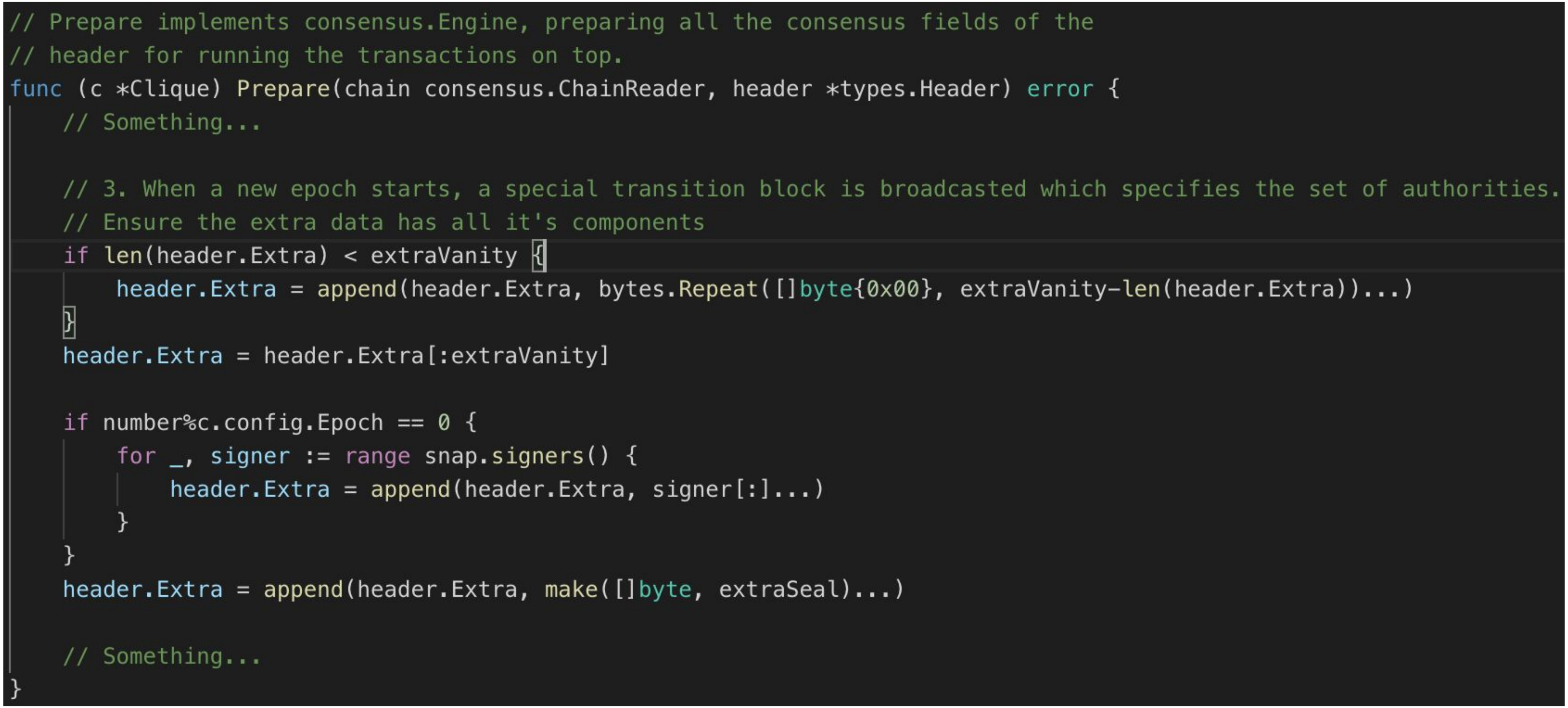
- The algorithm proceeds in epochs which are identified by genesis block.

- The set of authorities is stored in “extraData” field of genesis file.



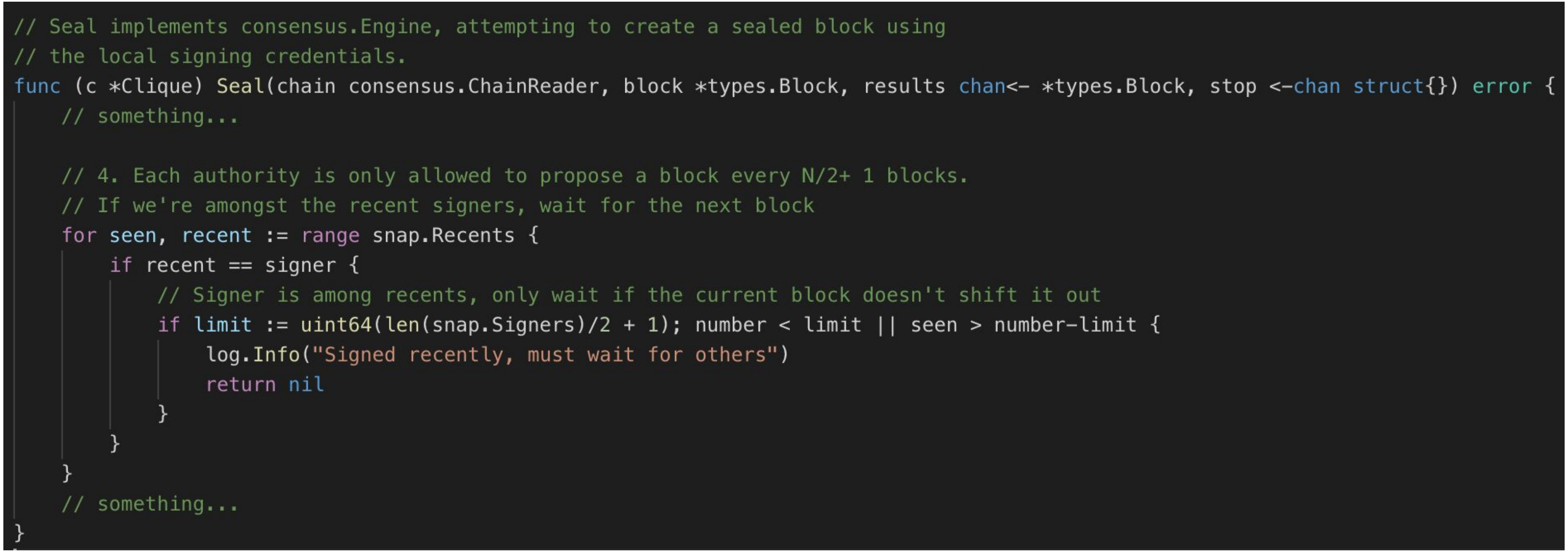
```
{
  "config": {
    "chainId": 19752,
    "homesteadBlock": 1,
    "eip150Block": 2,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 3,
    "eip158Block": 3,
    "byzantiumBlock": 4,
    "clique": {
      "period": 15,
      "epoch": 30000
    }
  },
  "nonce": "0x0",
  "timestamp": "0x5cff2693",
  "extraData":
  "0x0000000000000000000000000000000000000000000000000000000000000000d049ed8d2414a22b0416f90764590c5a4ad7
  f8ca0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
  000000000000000000000000000000",
  "gasLimit": "0x47b760",
  "difficulty": "0x1",
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": {⋯
  },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000"
}
```

genesis.json



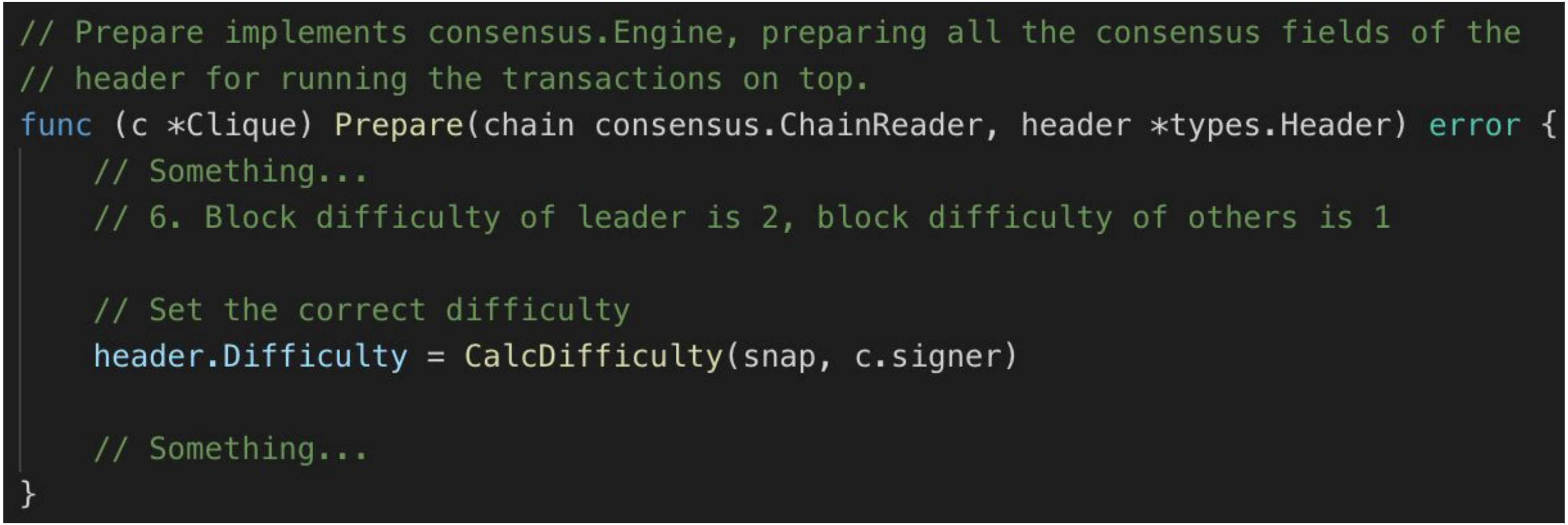
```go
// Prepare implements consensus.Engine, preparing all the consensus fields of the
// header for running the transactions on top.
func (c *Clique) Prepare(chain consensus.ChainReader, header *types.Header) error {
    // Something...

    // 3. When a new epoch starts, a special transition block is broadcasted which specifies the set of authorities.
    // Ensure the extra data has all it's components
    if len(header.Extra) < extraVanity {
        header.Extra = append(header.Extra, bytes.Repeat([]byte{0x00}, extraVanity-len(header.Extra))...)
    }
    header.Extra = header.Extra[:extraVanity]

    if number%c.config.Epoch == 0 {
        for _, signer := range snap.signers() {
            header.Extra = append(header.Extra, signer[:]...)
        }
    }
    header.Extra = append(header.Extra, make([]byte, extraSeal)...)

    // Something...
}
```

consensus/clique/clique.go



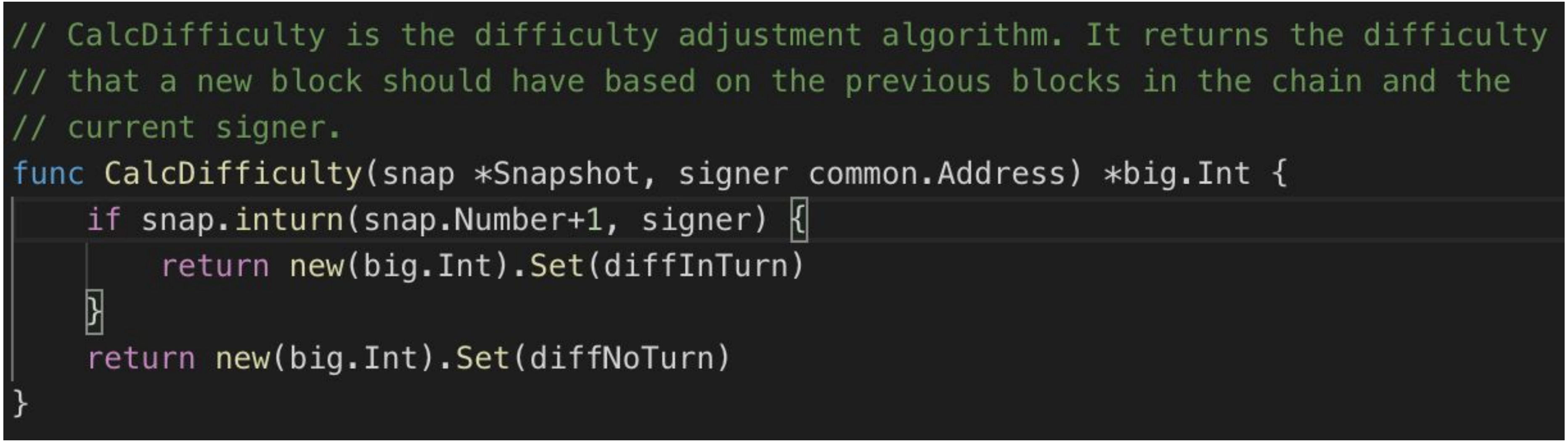
```go
// Seal implements consensus.Engine, attempting to create a sealed block using
// the local signing credentials.
func (c *Clique) Seal(chain consensus.ChainReader, block *types.Block, results chan<- *types.Block, stop <-chan struct{}) error {
    // something...

    // 4. Each authority is only allowed to propose a block every N/2+ 1 blocks.
    // If we're amongst the recent signers, wait for the next block
    for seen, recent := range snap.Recents {
        if recent == signer {
            // Signer is among recents, only wait if the current block doesn't shift it out
            if limit := uint64(len(snap.Signers)/2 + 1); number < limit || seen > number-limit {
                log.Info("Signed recently, must wait for others")
                return nil
            }
        }
    }
    // something...
}
```

consensus/clique/clique.go



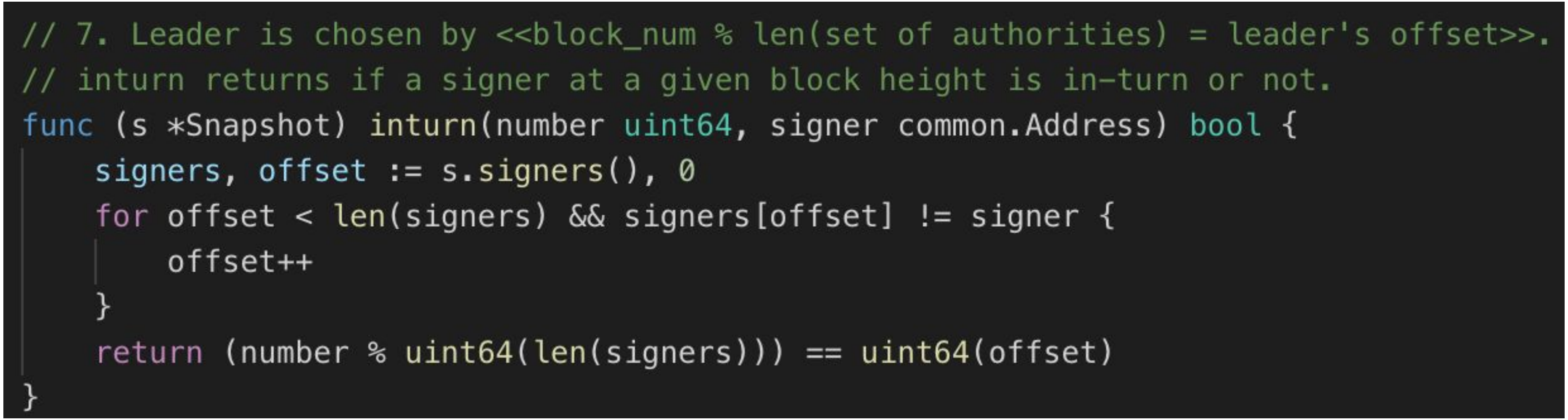
```go
// Prepare implements consensus.Engine, preparing all the consensus fields of the
// header for running the transactions on top.
func (c *Clique) Prepare(chain consensus.ChainReader, header *types.Header) error {
    // Something...
    // 6. Block difficulty of leader is 2, block difficulty of others is 1

    // Set the correct difficulty
    header.Difficulty = CalcDifficulty(snap, c.signer)

    // Something...
}
```

consensus/clique/clique.go



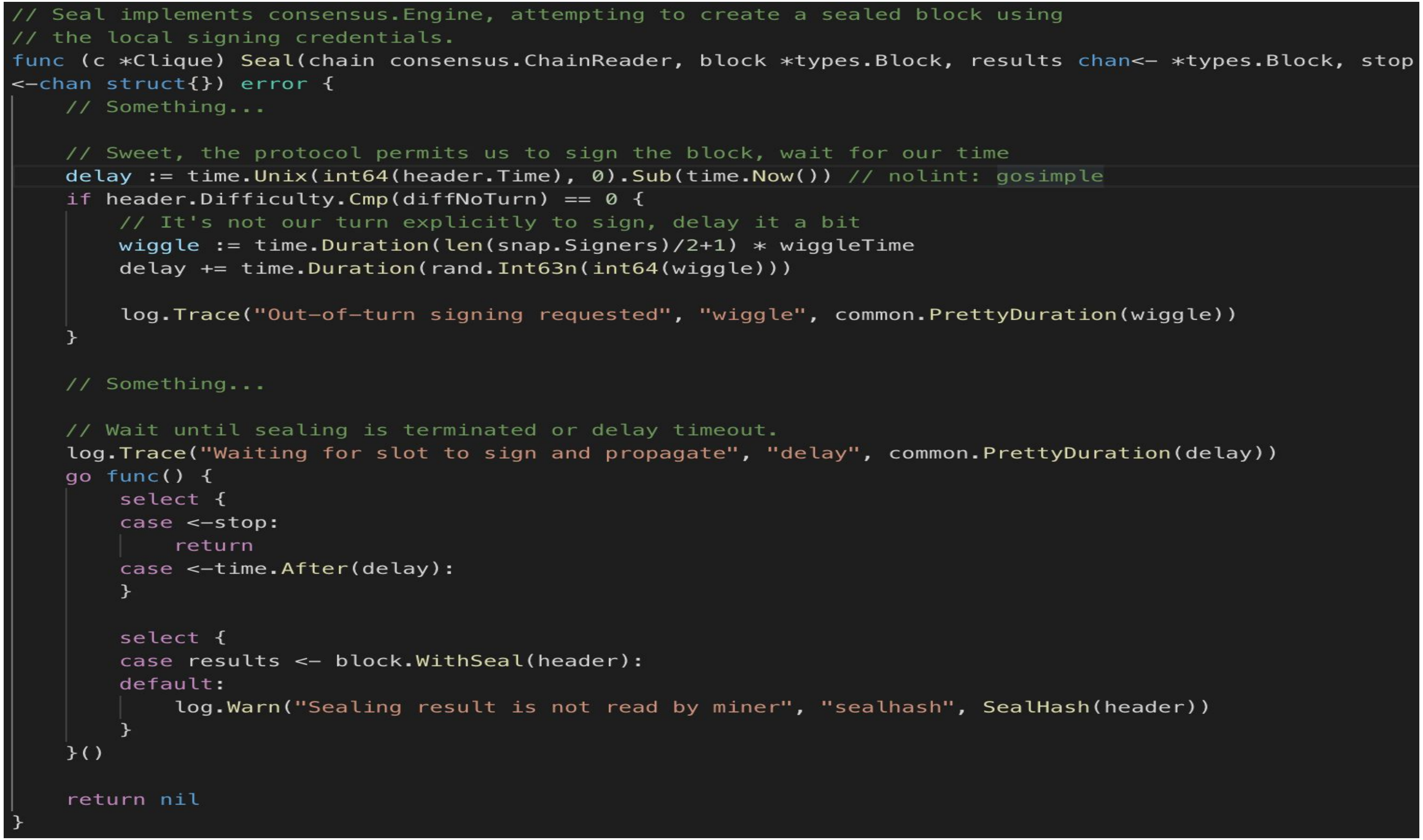
```
// CalcDifficulty is the difficulty adjustment algorithm. It returns the difficulty
// that a new block should have based on the previous blocks in the chain and the
// current signer.
func CalcDifficulty(snap *Snapshot, signer common.Address) *big.Int {
    if snap.inturn(snap.Number+1, signer) {
        return new(big.Int).Set(diffInTurn)
    }
    return new(big.Int).Set(diffNoTurn)
}
```

consensus/clique/clique.go



```go
// 7. Leader is chosen by <<block_num % len(set of authorities) = leader's offset>>.
// inturn returns if a signer at a given block height is in-turn or not.
func (s *Snapshot) inturn(number uint64, signer common.Address) bool {
    signers, offset := s.signers(), 0
    for offset < len(signers) && signers[offset] != signer {
        offset++
    }
    return (number % uint64(len(signers))) == uint64(offset)
}
```

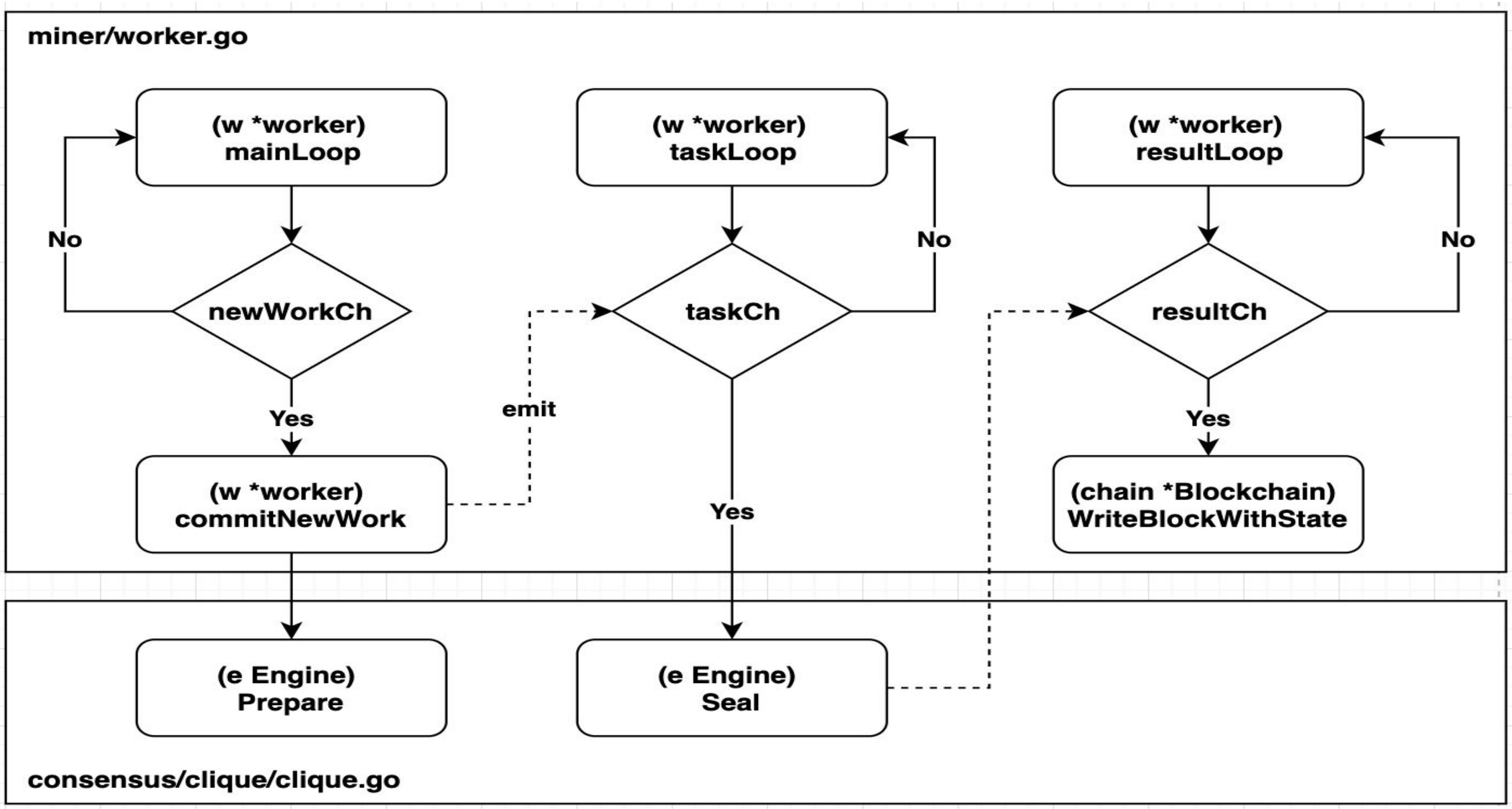
consensus/clique/snapshot.go



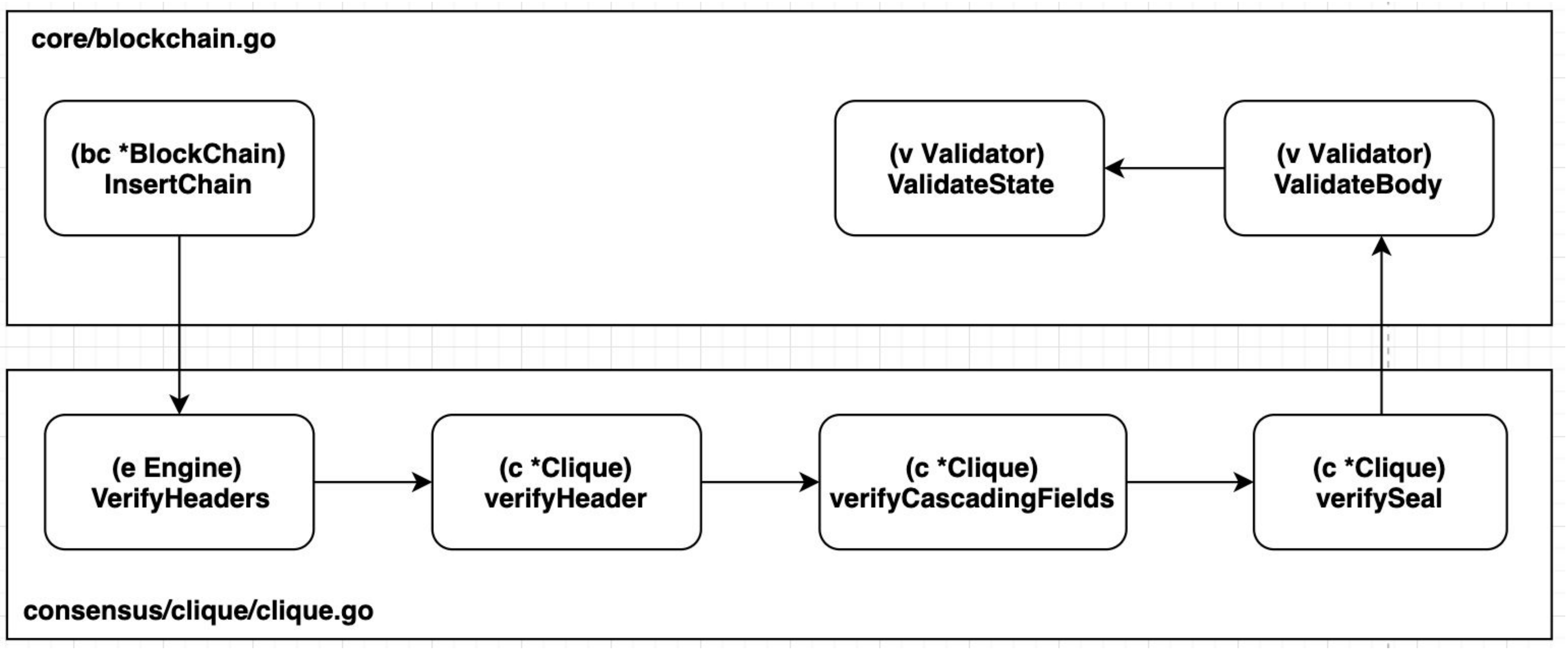
```go
// Seal implements consensus.Engine, attempting to create a sealed block using
// the local signing credentials.
func (c *Clique) Seal(chain consensus.ChainReader, block *types.Block, results chan<- *types.Block, stop
<-chan struct{}) error {
    // Something...

    // Sweet, the protocol permits us to sign the block, wait for our time
    delay := time.Unix(int64(header.Time), 0).Sub(time.Now()) // nolint: gosimple
    if header.Difficulty.Cmp(diffNoTurn) == 0 {
        // It's not our turn explicitly to sign, delay it a bit
        wiggle := time.Duration(len(snap.Signers)/2+1) * wiggleTime
        delay += time.Duration(rand.Int63n(int64(wiggle)))

        log.Trace("Out-of-turn signing requested", "wiggle", common.PrettyDuration(wiggle))
    }

    // Something...

    // Wait until sealing is terminated or delay timeout.
    log.Trace("Waiting for slot to sign and propagate", "delay", common.PrettyDuration(delay))
    go func() {
        select {
        case <-stop:
            return
        case <-time.After(delay):
        }

        select {
        case results <- block.WithSeal(header):
        default:
            log.Warn("Sealing result is not read by miner", "sealhash", SealHash(header))
        }
    }()

    return nil
}
```

consensus/clique/clique.go

Mining Flow

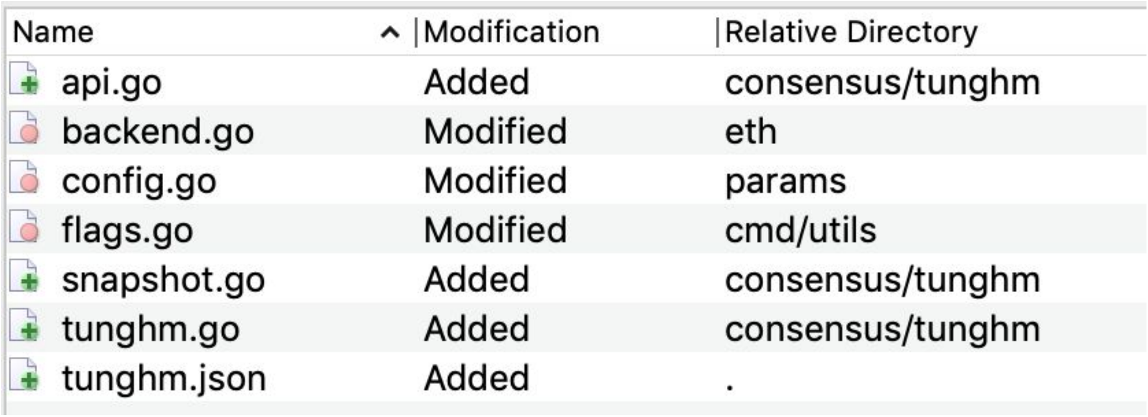

miner/worker.go
(w *worker) mainLoop
No
newWorkCh
Yes
(w *worker) commitNewWork
emit
(w *worker) taskLoop
No
taskCh
Yes
(w *worker) resultLoop
No
resultCh
Yes
(chain *Blockchain) WriteBlockWithState
(e Engine) Prepare
(e Engine) Seal
consensus/clique/clique.go

Verify Block Header Flow



5. How To Create New Consensus In Geth

How To Create New Consensus In Geth



Name	Modification	Relative Directory
api.go	Added	consensus/tunghm
backend.go	Modified	eth
config.go	Modified	params
flags.go	Modified	cmd/utils
snapshot.go	Added	consensus/tunghm
tunghm.go	Added	consensus/tunghm
tunghm.json	Added	.

1. **Add “tunghm” consensus to Geth which is cloned Clique consensus.**
2. **Modifies: Only an authority is allowed to propose a block at a time.**

Reference

1. https://github.com/ethereum/go-ethereum
2. https://github.com/ethereum/EIPs/issues/225
3. https://eprints.soton.ac.uk/415083/2/itasec18_main.pdf
4. https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256

INFINITY
BLOCKCHAIN LABS

Ethereum

Thank You

Hội anh em blockchain "thiện lành"