```
☐ ethereum / go-ethereum (Public)
<> Code
             (•) Issues 299
                               11 Pull requests 79 Wiki
                                                                   ጕ 671094279e ▼
go-ethereum / core / forkchoice.go / <> Jump to ▼
      MariusVanDerWijden all: core rework for the merge transition (#23761) ... X
                                                                                              ( History
  A 3 contributors
  108 lines (99 sloc) | 4.13 KB
        // Copyright 2021 The go-ethereum Authors
         // This file is part of the go-ethereum library.
    2
    3
        // The go-ethereum library is free software: you can redistribute it and/or modify
         // it under the terms of the GNU Lesser General Public License as published by
         // the Free Software Foundation, either version 3 of the License, or
    7
         // (at your option) any later version.
    8
         // The go-ethereum library is distributed in the hope that it will be useful,
   10
         // but WITHOUT ANY WARRANTY; without even the implied warranty of
         // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
   11
         // GNU Lesser General Public License for more details.
   13
   14
         // You should have received a copy of the GNU Lesser General Public License
   15
         // along with the go-ethereum library. If not, see <a href="http://www.gnu.org/licenses/">http://www.gnu.org/licenses/</a>.
   16
   17
         package core
   18
   19
         import (
   20
                 crand "crypto/rand"
   21
                 "errors"
   22
                 "math/big"
                mrand "math/rand"
   23
   24
                 "github.com/ethereum/go-ethereum/common"
                 "github.com/ethereum/go-ethereum/common/math"
   26
                 "github.com/ethereum/go-ethereum/core/types"
   27
                 "github.com/ethereum/go-ethereum/log"
   28
```

```
29
             "github.com/ethereum/go-ethereum/params"
30
     )
31
32
     // ChainReader defines a small collection of methods needed to access the local
33
     // blockchain during header verification. It's implemented by both blockchain
34
     // and lightchain.
35
     type ChainReader interface {
36
             // Config retrieves the header chain's chain configuration.
37
             Config() *params.ChainConfig
38
39
             // GetTd returns the total difficulty of a local block.
40
             GetTd(common.Hash, uint64) *big.Int
41
     }
42
43
     // ForkChoice is the fork chooser based on the highest total difficulty of the
44
     // chain(the fork choice used in the eth1) and the external fork choice (the fork
45
     // choice used in the eth2). This main goal of this ForkChoice is not only for
46
     // offering fork choice during the eth1/2 merge phase, but also keep the compatibility
47
     // for all other proof-of-work networks.
     type ForkChoice struct {
48
49
             chain ChainReader
50
             rand *mrand.Rand
51
52
             // preserve is a helper function used in td fork choice.
             // Miners will prefer to choose the local mined block if the
53
54
             // local td is equal to the extern one. It can be nil for light
55
             // client
56
             preserve func(header *types.Header) bool
57
58
59
     func NewForkChoice(chainReader ChainReader, preserve func(header *types.Header) bool) *ForkChoice
             // Seed a fast but crypto originating random generator
60
             seed, err := crand.Int(crand.Reader, big.NewInt(math.MaxInt64))
61
62
             if err != nil {
63
                     log.Crit("Failed to initialize random seed", "err", err)
             }
64
65
             return &ForkChoice{
66
                     chain:
                                chainReader,
                     rand:
                                mrand.New(mrand.NewSource(seed.Int64())),
67
68
                     preserve: preserve,
69
             }
70
71
72
     // ReorgNeeded returns whether the reorg should be applied
73
     // based on the given external header and local canonical chain.
74
     // In the td mode, the new head is chosen if the corresponding
75
     // total difficulty is higher. In the extern mode, the trusted
76
     // header is always selected as the head.
77
     func (f *ForkChoice) ReorgNeeded(current *types.Header, header *types.Header) (bool, error) {
```

```
78
              var (
79
                       localTD = f.chain.GetTd(current.Hash(), current.Number.Uint64())
                       externTd = f.chain.GetTd(header.Hash(), header.Number.Uint64())
80
81
              )
              if localTD == nil || externTd == nil {
82
83
                       return false, errors.New("missing td")
84
              }
              // Accept the new header as the chain head if the transition
85
86
              // is already triggered. We assume all the headers after the
87
              // transition come from the trusted consensus layer.
              if ttd := f.chain.Config().TerminalTotalDifficulty; ttd != nil && ttd.Cmp(externTd) <= 0 {</pre>
88
89
                       return true, nil
90
              }
91
              // If the total difficulty is higher than our known, add it to the canonical chain
92
              // Second clause in the if statement reduces the vulnerability to selfish mining.
              // Please refer to http://www.cs.cornell.edu/~ie53/publications/btcProcFC.pdf
93
94
              reorg := externTd.Cmp(localTD) > 0
95
              if !reorg && externTd.Cmp(localTD) == 0 {
96
                       number, headNumber := header.Number.Uint64(), current.Number.Uint64()
97
                       if number < headNumber {</pre>
98
                               reorg = true
99
                       } else if number == headNumber {
                               var currentPreserve, externPreserve bool
100
101
                               if f.preserve != nil {
102
                                       currentPreserve, externPreserve = f.preserve(current), f.preserve(
103
                               }
104
                               reorg = !currentPreserve && (externPreserve || f.rand.Float64() < 0.5)</pre>
105
                       }
106
107
              return reorg, nil
108
      }
```