```
1  ---------- MODULE PoDCon ----------
2  EXTENDS Integers, FiniteSets, Sequences

4  INSTANCE Block
5  ----------
6  CONSTANTS Validator,          The set of honest validators
7            FakeValidator,     The set of malicious or crashed validators
8            ByzQuorum          Set of n honest validators with f fake validators, where n ≥ 2f+1. Each byzant

10 ByzValidator ≜ Validator ∪ FakeValidator

12 ****************************************** Constants                    for TLC Model:**********************
13 Validator ← {"v1", "v2", "v3", "v4"}
14 FakeValidator ← {"f1"}
15 ByzQuorum ← {{"v1", "v2", "v3", "f1"}, {"v4", "v2", "v3", "f1"}, {"v1", "v4", "v3", "f1"}, {"v1", "v2", "v4", "f1"}, {"v1
17 ASSUME BQA ≜ ∧ Validator ∩ FakeValidator = {}
18             ∧ ∀ Q ∈ ByzQuorum : Q ⊆ ByzValidator
19             ∧ ∀ Q1, Q2 ∈ ByzQuorum : Q1 ∩ Q2 ∩ Validator ≠ {}
20 ----------
21 CONSTANTS Blocks

23 Genesis ≜ [id ↦ 1, parent ↦ 0]          Geneis block

25 ASSUME BA ≜ ∀ b ∈ Blocks : b ∈ Block

27 ****************************************** Constants                    for TLC Model:**********************
28 Blocks ← {[id ↦ 1, parent ↦ 0], [id ↦ 2, parent ↦ 1], [id ↦ 3, parent ↦ 2]}

31 ----------
32 --algorithm PoDCon
33   variables localBlocks   = [v ∈ ByzValidator ↦ {Genesis}],          Local chain
34             beaconChain = [v ∈ ByzValidator ↦ ⟨Genesis⟩],           chain that records finalized blocks
35             votedPath = [v ∈ ByzValidator ↦ {}],                    voted path in the first round
36             prefixPaths = [v ∈ ByzValidator ↦ {}],                  all posible prefix paths of a byzvalidator
37             votedPrefix = [v ∈ ByzValidator ↦ {}],                  voted prefix in the second round
38             msgs = {} ;                                             all messages

40   define
41     Here we need some useful operatos, and some of them are defined in Block.tla
42       IsChain(blocks)
43       IsPath(blocks)
44       Prefix(chains)
45       GetPath(s, t, blocks)
46       LongestPath(paths)

48       Get the set of all elements in seq
49       SeqToSet(seq) ≜ {seq[i] : i ∈ 1 .. Len(seq)}
```

```
51          True for did not vote the path or any path conflicting before.
52          The first block of the path should be finalized which means shoule be in the beaconChain
53      DidNotVotePath(v, path) ≜ LET head ≜ HeadBlock(path)
54                                       finalized_blocks ≜ SeqToSet(beaconChain[v])
55                                   IN
56                                       ∧ ∀ b ∈ path \ {head} : b ∉ finalized_blocks
57                                       ∧ head ∈ finalized_blocks
58  end define ;

60   Phase of receiving new blocks
61  macro ReceiveNewBlock()begin
62        For test here
63      localBlocks[self] := AddBlocks(Blocks, localBlocks[self]) ;
64  end macro ;

66   Phase of voting for paht
67  macro VoteForPath()begin
68      with s = beaconChain[self][Len(beaconChain[self])],    get the last block in beacon chain as the initiative blo
69           t = TailBlock(localBlocks[self]) do                get the last block in local blocks as the terminated bl
70           if IsPrev(s, t, localBlocks[self]) then            IsPrev() will return false if s = t, which means the vo
71              with path = GetPath(s, t, localBlocks[self]) do
72                  if DidNotVotePath(self, path) then
73                      votedPath[self] := path ;                  empty the set when go to final height vote pathse
74                      msgs := msgs ∪ {[type ↦ "path_vote", sender ↦ self, val ↦ path]} ;
75                   else
76                      skip ;
77                  end if ;
78              end with ;
79           else
80              skip ;
81           end if ;
82      end with ;
83  end macro ;


86     Phase of voting for longest common prefix, TBA
87  macro VoteForCommonPrefix()begin
88      if votedPath[self] ≠ {} then
89        wait until
90          await ∃ Q ∈ ByzQuorum : ∧ ∀ v ∈ (Q ∩ Validator) : votedPath[v] ≠ {}
91                                  ∧ self ∈ Q ;                    may not to have this
92          with quorum_set = {Q ∈ ByzQuorum : ∧ ∀ v ∈ (Q ∩ Validator) : votedPath[v] ≠ {}
93                                             ∧ self ∈ Q} do
94              with all_prefixs = {GetPrefix({votedPath[v] : v ∈ (q ∩ Validator)}) : q ∈ quorum_set} do
95                  votedPrefix[self] := LongestPath(all_prefixs) ;
96                  msgs := msgs ∪ {[type ↦ "prefix_vote", sender ↦ self, val ↦ votedPrefix[self]]} ;
```

```
                end with ;
            end with ;
          else
            skip;
          end if ;
    end macro ;

    macro PhaseFinalHeightVote()begin

    end macro ;

    macro FakingValidator()begin

        skip;
    end macro ;

    We combine these actions into separate process decalrations for validators and fake validators
 fair process v ∈ Validator
  begin vote:
    while TRUE do
              either
                    ReceiveNewBlock();
                or
                    VoteForPath();
                or
                    VoteForCommonPrefix();
               or
                  PhaseFinalHeightVote();
               or
                  skip;
              end either ;
    end while ;
      skip;
  end process ;

    Fake validators
  process fv ∈ FakeValidator
  begin fake_vote:
    while TRUE do
        skip;              do nothing
    end while ;
  end process ;


end algorithm ;
 BEGIN TRANSLATION
```

```
144 VARIABLES localBlocks, beaconChain, votedPath, prefixPaths, votedPrefix, msgs

146  define statement
147 SeqToSet(seq) ≜ {seq[i] : i ∈ 1 .. Len(seq)}


151 DidNotVotePath(v, path) ≜ LET head ≜ HeadBlock(path)
152                                finalized_blocks ≜ SeqToSet(beaconChain[v])
153                            IN
154                                ∧ ∀ b    ∈ path \ {head} : b ∉ finalized_blocks
155                                ∧ head ∈ finalized_blocks


158 vars ≜ ⟨localBlocks, beaconChain, votedPath, prefixPaths, votedPrefix,
159          msgs⟩

161 ProcSet ≜ (Validator) ∪ (FakeValidator)

163 Init ≜  Global variables
164         ∧ localBlocks   = [v ∈ ByzValidator ↦ {Genesis}]
165         ∧ beaconChain = [v ∈ ByzValidator ↦ ⟨Genesis⟩]
166         ∧ votedPath = [v ∈ ByzValidator ↦ {}]
167         ∧ prefixPaths = [v ∈ ByzValidator ↦ {}]
168         ∧ votedPrefix = [v ∈ ByzValidator ↦ {}]
169         ∧ msgs = {}

171 v(self) ≜  ∧ ∨ ∧ localBlocks' = [localBlocks EXCEPT ![self] = AddBlocks(Blocks, localBlocks[self])]
172               ∧ UNCHANGED ⟨votedPath, votedPrefix, msgs⟩
173            ∨ ∧ LET s ≜ beaconChain[self][Len(beaconChain[self])]IN
174                  LET t ≜ TailBlock(localBlocks[self])IN
175                    IF IsPrev(s, t, localBlocks[self])
176                       THEN ∧ LET path ≜ GetPath(s, t, localBlocks[self])IN
177                                 IF DidNotVotePath(self, path)
178                                    THEN ∧ votedPath' = [votedPath EXCEPT ![self] = path]
179                                         ∧ msgs' = (msgs ∪ {[type ↦ "path_vote", sender ↦ self, val ↦
180                                    ELSE ∧ TRUE
181                                         ∧ UNCHANGED ⟨votedPath, msgs⟩
182                       ELSE ∧ TRUE
183                            ∧ UNCHANGED ⟨votedPath, msgs⟩
184               ∧ UNCHANGED ⟨localBlocks, votedPrefix⟩
185            ∨ ∧ IF votedPath[self] ≠ {}
186                  THEN ∧ ∃ Q ∈ ByzQuorum : ∧ ∀ v  ∈ (Q ∩ Validator) : votedPath[v] ≠ {}
187                                           ∧ self ∈ Q
188                       ∧ LET quorum_set ≜ {Q ∈ ByzQuorum : ∧ ∀ v  ∈ (Q ∩ Validator) : votedPath[
189                                                           ∧ self ∈ Q}IN
190                         LET all_prefixs ≜ {GetPrefix({votedPath[v] : v ∈ (q ∩ Validator)}) : q ∈ qu
191                           ∧ votedPrefix' = [votedPrefix EXCEPT ![self]  = LongestPath(all_prefixs)]
```

```
                                        ∧ msgs' = (msgs ∪ {[type ↦ "prefix_vote", sender ↦ self, val ↦ votedPrefi
                          ELSE  ∧ TRUE
                                ∧ UNCHANGED ⟨votedPrefix, msgs⟩
                    ∧ UNCHANGED ⟨localBlocks, votedPath⟩
              ∧ UNCHANGED ⟨beaconChain, prefixPaths⟩

fv(self) ≜  ∧ TRUE
            ∧ UNCHANGED ⟨localBlocks, beaconChain, votedPath, prefixPaths,
                         votedPrefix, msgs⟩

Next ≜ (∃ self ∈ Validator : v(self))
       ∨ (∃ self ∈ FakeValidator : fv(self))

Spec ≜  ∧ Init ∧ □[Next]_vars
        ∧ ∀ self ∈ Validator : WF_vars(v(self))

END TRANSLATION

──────────────────────────────────────────────────────────────────────────
********** Invariants **********
ChainCorrectness ≜  ∀ i ∈ Validator :  ∧ localBlocks[i] ⊆ Blocks
                                       ∧ votedPath[i] ⊆ Blocks
                                         ∧ prefixPaths[i] ⊆ Blocks

GenesisInvariants ≜ ∀ i ∈ ByzValidator :  ∧ Genesis ∈ localBlocks[i]
                                          ∧ Genesis = beaconChain[i][1]


********** Properties **********
Liveness ≜ ∀ i ∈ Validator :  ∧ ◇(Blocks = localBlocks[i])
                              ∧ ◇(Blocks = votedPath[i])    for test
                              ∧ ◇(Blocks = votedPrefix[i])  for test
──────────────────────────────────────────────────────────────────────────
\* Modification History
\* Last modified Fri Jun 14 17:41:47 CST 2019 by tangzaiyang
\* Created Wed Jun 05 14:48:17 CST 2019 by tangzaiyang
```