```
1  ---------- MODULE Zookeeper ----------
2  EXTENDS Naturals, TLC, Sequences, FiniteSets, Integers, ExternalSeqRecordParser3, StatesConsistencyInspe
3  VARIABLES evoteSeq, offset, evotecollection, offset_2, stateSeq, statecollection, leader,
4            follower, followerSendEpochToLeader, leaderReceivedEpochFromFollower,
5            leaderCalculaterNewEpoch, followerReceivedNewEpochFromLeader, leaderSyncDataZxid, followerS
6            broadCastSeq, offset_4, broadCastCollection
7  vars ≜ ⟨evoteSeq, offset, evotecollection, offset_2, stateSeq, statecollection, leader,
8          follower, followerSendEpochToLeader, leaderReceivedEpochFromFollower,
9          leaderCalculaterNewEpoch, followerReceivedNewEpochFromLeader, leaderSyncDataZxid, followerSyn
10         broadCastSeq, offset_4, broadCastCollection⟩
   Trace ≜ ExSeqRcdParser3("D:\\00001code\\runtimemodel\\zookeeper_environment\\leaderelection.log")
   Trace2 ≜ StateConsistencyParser("D:\\00001code\\runtimemodel\\zookeeper_environment\\stateconsistency.log")
   Trace3 ≜ ExSeqRcdParser2("D:\\00001code\\runtimemodel\\zookeeper_environment\\datasync.log")
   Trace4 ≜ broadcastparser("D:\\00001code\\runtimemodel\\zookeeper_environment\\broadcast.log")
   Trace5 ≜ broadcastparser2("D:\\00001code\\runtimemodel\\zookeeper_environment\\broadcast.log")
18 Trace ≜ ExSeqRcdParser3("./leaderelection.log")
19 Trace2 ≜ StateConsistencyParser("./stateconsistency.log")
20 Trace3 ≜ ExSeqRcdParser2("./datasync.log")
21 Trace4 ≜ broadcastparser("./broadcast.log")
22 Trace5 ≜ broadcastparser2("./broadcast.log")

24 RECURSIVE Seq2Set(_)
25 Seq2Set(S) ≜
26     IF S = ⟨⟩ THEN {}
27      ELSE
28         LET i ≜ Head(S)
29         IN   {i} ∪ Seq2Set(Tail(S))
30 ---------- rule1 ----------
31 过滤出来源于本节点的选票
32 selectvoteFromNodeSelf(S) ≜ {temp ∈ S :
33                                 ∧ temp.myId = temp.from
```

```
                                        ∧ temp.myState = "LOOKING"
                                        ∧ temp.state = "LOOKING"
                                  }
Rule1 ≜  ∃ vote ∈ selectvoteFromNodeSelf(evotecollection) :
                                                vote.proposedLeader = vote.myId
──────────────────────────── rule2 ────────────────────────────
IsInjective(f) ≜ ∀ a, b ∈ DOMAIN f : f[a] = f[b] ⇒ a = b
SetToSeq(S) ≜ CHOOSE f ∈ [1 .. Cardinality(S) → S] : IsInjective(f)
selectStateIsLeading(S) ≜ {x ∈ S :
                                  x.State = "LEADING"}
Rule2 ≜ Len(SetToSeq(selectStateIsLeading(statecollection))) = 1
──────────────────────────── rule3 ────────────────────────────
decideStateAllIsLOOKING(S) ≜ ∀ x ∈ S :
                                    x.state = "LOOKING"
selectMaxelectionEpoch(S) ≜ {x ∈ S :
                                ∀ y ∈ S :
                                  y.electionEpoch ≤ x.electionEpoch}
selectVoteByZxid(S) ≜ {x ∈ S :
                         ∀ y ∈ S :
                           ∨ y.proposedZxidHigh < x.proposedZxidHigh
                           ∨ ((y.proposedZxidHigh = x.proposedZxidHigh)
                              ∧ (y.proposedZxidLow ≤ x.proposedZxidLow)
                             )}
decideProposedLeaderEqualEndvote(S) ≜ ∀ x ∈ S :
                                         ∧ x.proposedLeader = x.endvote
selectVoteByMyid(S) ≜ {x ∈ S :
                         ∀ y ∈ S :
                           y.from ≤ x.from}
selectVoteByState(S, state) ≜ {x ∈ S :
                                  x.state = state}
Rule3 ≜ IF decideStateAllIsLOOKING(selectMaxelectionEpoch(evotecollection)) = TRUE
          THEN
          ∧ decideProposedLeaderEqualEndvote(
```

```
67                 selectVoteByMyid(
68                 selectVoteByZxid(selectMaxelectionEpoch(evotecollection)))) = TRUE
69               ELSE
70               ∧ decideProposedLeaderEqualEndvote(
71                 selectVoteByState(
72                 selectMaxelectionEpoch(evotecollection), "FOLLOWING")) = TRUE
73               ∧ decideProposedLeaderEqualEndvote(
74                 selectVoteByState(selectMaxelectionEpoch(evotecollection), "LEADING")) = TRUE
75  ---------- rule4 ----------
76  Rule4 ≜ leaderSyncDataZxid = followerSyncDataZxid
77  Rule5 ≜ followerSendEpochToLeader = leaderReceivedEpochFromFollower
78  Rule6 ≜ leaderCalculaterNewEpoch = followerReceivedNewEpochFromLeader
79  ---------- rule7 ----------
80  selectLeaderNode(S) ≜ CHOOSE temp ∈ S :
81                                    ∧ temp.action = "LeaderLaunchProposal"
83  selectNodeCommitTranslation(S) ≜ {temp ∈ S :
84                                        ∧ temp.action = "Request2DataTree"
85                                   }
86  selectFollowerNode(S) ≜ {temp ∈ S :
87                                    ∧ temp.action = "FollowerSendAckToLeader"}
89  selectReceivedProposal(S) ≜  {temp ∈ S :
90                                    ∧ temp.action = "ReceivedProposal"}
91  decideWriteRequestTranslationConsistency(
92        writeRequestCollection,
93        NodeCommitTranslationCollection, nodeCount,
94        Request2DataTreeCount) ≜
95                                ∀ commitTranslation ∈ NodeCommitTranslationCollection :
96                                 ∧ commitTranslation.sessionid = writeRequestCollection.sessionid
97                                 ∧ commitTranslation.type = writeRequestCollection.type
98                                 ∧ commitTranslation.cxid = writeRequestCollection.cxid
99                                 ∧ commitTranslation.zxidHigh = writeRequestCollection.zxidHigh
100                                ∧ commitTranslation.zxidLow = writeRequestCollection.zxidLow
```

```
                                        ∧ commitTranslation.txntype  = writeRequestCollection.txntype
                                        ∧ nodeCount = Request2DataTreeCount
Rule7 ≜ decideWriteRequestTranslationConsistency(
                                            selectLeaderNode(broadCastCollection),
                                            selectNodeCommitTranslation(broadCastCollection),
                                            Len(SetToSeq(selectFollowerNode(broadCastCollection))) +
                                            Len(SetToSeq(selectNodeCommitTranslation(broadCastColl
                                            ) = TRUE

判断proposal接受发送数据一致性
decideProposalConsistency(SendProposal, ReceivedProposal) ≜
                                    ∀ RP ∈ ReceivedProposal :
                                     ∧ RP.sessionid = SendProposal.sessionid
                                     ∧ RP.type = SendProposal.type
                                     ∧ RP.cxid = SendProposal.cxid
                                     ∧ RP.zxidHigh = SendProposal.zxidHigh
                                     ∧ RP.zxidLow = SendProposal.zxidLow
Rule8 ≜ decideProposalConsistency(
                                        selectLeaderNode(broadCastCollection),
                                        selectReceivedProposal(broadCastCollection)) = TRUE


decideOrder(zxidHigh_1, zxidLow_1, zxidHigh_2, zxidLow_2) ≜
                                                              IF (zxidHigh_1 < zxidHigh_2)
                                                               THEN
                                                               ∧ TRUE
                                                               ELSE
                                                               ∧ IF (zxidHigh_1 = zxidHigh_2)
                                                                   THEN
                                                                   ∧ IF (zxidLow_1 < zxidLow_2)
                                                                       THEN
                                                                       ∧ TRUE
                                                                       ELSE
                                                                       ∧   FALSE
```

```
                                                              ELSE
                                                              ∧  FALSE
globaOrderVerification(S1, S2) ≜
                        ∧ ∀ temp ∈ S1 :
                          ∀ temp_2 ∈ S2 :
                            ∧ IF ((temp.myId = temp_2.myId))
                              THEN
                              ∧ decideOrder(temp.zxidHigh,
                                            temp.zxidLow,
                                            temp_2.zxidHigh,
                                            temp_2.zxidLow)
                              ELSE
                              ∧ TRUE
Rule9 ≜ IF ((offset_4 + 1) ≤ Len(Trace5))
            THEN
            ∧ globaOrderVerification(Trace5[offset_4], Trace5[offset_4 + 1])
            ELSE
            ∧ TRUE
```

$$
\begin{aligned}
&selectRule1Init \triangleq \land offset = 1 \\
&\qquad\qquad\quad \land evoteSeq = Trace[offset] \\
&\qquad\qquad\quad \land evotecollection = Seq2Set(Trace[offset]) \\
&selectRule2Init \triangleq \land offset\_2 = 1 \\
&\qquad\qquad\quad \land stateSeq = Trace2[offset\_2] \\
&\qquad\qquad\quad \land statecollection = Seq2Set(Trace2[offset\_2]) \\
&datasyncInit \triangleq \land offset\_3 = 1 \\
&\qquad\qquad\quad \land leader = Trace3[1].leader \\
&\qquad\qquad\quad \land follower = Trace3[1].follower \\
&\qquad\qquad\quad \land followerSendEpochToLeader = Trace3[1].followerSendEpochToLeader \\
&\qquad\qquad\quad \land leaderReceivedEpochFromFollower = \\
&\qquad\qquad\qquad Trace3[1].leaderReceivedEpochFromFollower \\
&\qquad\qquad\quad \land leaderCalculaterNewEpoch = \\
&\qquad\qquad\qquad Trace3[1].leaderCalculaterNewEpoch \\
&\qquad\qquad\quad \land followerReceivedNewEpochFromLeader = \\
&\qquad\qquad\qquad Trace3[1].followerReceivedNewEpochFromLeader \\
&\qquad\qquad\quad \land leaderSyncDataZxid = Trace3[1].leaderSyncDataZxid \\
&\qquad\qquad\quad \land followerSyncDataZxid = Trace3[1].followerSyncDataZxid \\
&broadcastInit \triangleq \land offset\_4 = 1 \\
&\qquad\qquad\quad \land broadCastSeq = Trace4[1] \\
&\qquad\qquad\quad \land broadCastCollection = Seq2Set(Trace4[1]) \\
&Init \triangleq \land selectRule1Init \\
&\qquad\quad \land selectRule2Init \\
&\qquad\quad \land datasyncInit \\
&\qquad\quad \land broadcastInit \\
&selectRule1Next \triangleq \land offset < Len(Trace) \\
&\qquad\qquad\quad \land offset' = offset + 1 \\
&\qquad\qquad\quad \land evoteSeq' = Trace[offset'] \\
&\qquad\qquad\quad \land evotecollection' = Seq2Set(evoteSeq') \\
&\qquad\qquad\quad \land \text{UNCHANGED } \langle offset\_2,\ stateSeq,\ statecollection,\ leader,\ follower, \\
&\qquad\qquad\qquad\qquad followerSendEpochToLeader,\ leaderReceivedEpochFromFollow
\end{aligned}
$$

```
                                leaderCalculaterNewEpoch, followerReceivedNewEpochFromLeader,
                                leaderSyncDataZxid, followerSyncDataZxid, offset_3,
                                broadCastSeq, offset_4, broadCastCollection⟩

selectRule2Next ≜ ∧ offset_2 < Len(Trace2)
                  ∧ offset_2' = offset_2 + 1
                  ∧ stateSeq' = Trace2[offset_2']
                  ∧ statecollection' = Seq2Set(stateSeq')
                  ∧ UNCHANGED ⟨evoteSeq, offset, evotecollection, leader, follower,
                                followerSendEpochToLeader, leaderReceivedEpochFromFollower,
                                leaderCalculaterNewEpoch, followerReceivedNewEpochFromLeader,
                                leaderSyncDataZxid, followerSyncDataZxid, offset_3,
                                broadCastSeq, offset_4, broadCastCollection⟩

datasyncNext ≜ ∧ offset_3 < Len(Trace3)
               ∧ offset_3' = offset_3 + 1
               ∧ leader' = Trace3[offset_3'].leader
               ∧ follower' = Trace3[offset_3'].follower
               ∧ followerSendEpochToLeader' = Trace3[offset_3'].followerSendEpochToLeader
               ∧ leaderReceivedEpochFromFollower' = Trace3[offset_3'].leaderReceivedEpochFromFollower
               ∧ leaderCalculaterNewEpoch' = Trace3[offset_3'].leaderCalculaterNewEpoch
               ∧ followerReceivedNewEpochFromLeader' = Trace3[offset_3'].followerReceivedNewEpochFrom
               ∧ leaderSyncDataZxid' = Trace3[offset_3'].leaderSyncDataZxid
               ∧ followerSyncDataZxid' = Trace3[offset_3'].followerSyncDataZxid
               ∧ UNCHANGED ⟨offset_2, stateSeq, statecollection, evoteSeq, offset, evotecollection,
                            broadCastSeq, offset_4, broadCastCollection⟩

broadcastNext ≜ ∧ offset_4 < Len(Trace4)
                ∧ offset_4' = offset_4 + 1
                ∧ broadCastSeq' = Trace4[offset_4']
                ∧ broadCastCollection' = Seq2Set(broadCastSeq')
                ∧ UNCHANGED ⟨evoteSeq, offset, evotecollection, offset_2, stateSeq, statecollection, leader,
                             follower, followerSendEpochToLeader, leaderReceivedEpochFromFollower,
                             leaderCalculaterNewEpoch, followerReceivedNewEpochFromLeader,
```

$leaderSyncDataZxid,\ followerSyncDataZxid,\ offset\_3\rangle$

$selectRule1term \triangleq \land\ offset \geq Len(Trace)$
$\land$ UNCHANGED $vars$

$selectRule2term \triangleq \land\ offset\_2 \geq Len(Trace2)$
$\land$ UNCHANGED $vars$

$datasyncterm \triangleq \land\ offset\_3 \geq Len(Trace3)$
$\land$ UNCHANGED $vars$

$broadcastterm \triangleq \land\ offset\_4 \geq Len(Trace4)$
$\land$ UNCHANGED $vars$

$Next \triangleq \lor\ selectRule1Next$
$\lor\ selectRule2Next$
$\lor\ datasyncNext$
$\lor\ broadcastNext$
$\lor\ selectRule1term$
$\lor\ selectRule2term$
$\lor\ datasyncterm$
$\lor\ broadcastterm$

$Spec \triangleq Init \land \Box[Next]_{vars}$

* Modification History
* Last modified *Wed Mar* 30 16:44:59 *CST* 2022 by 10222803
* Created *Tue Mar* 29 15:20:35 *CST* 2022 by 10222803