```
——— MODULE Config -
INSTANCE Naturals
INSTANCE FiniteSets
{\tt INSTANCE}\ Sequences
LOCAL INSTANCE TLC
This section specifies constant parameters for the model.
CONSTANT None
Assume None \in \text{String}
CONSTANT Node
Assume \forall n \in Node : n \in String
CONSTANTS
   Change,
   Rollback
Event \triangleq \{Change, Rollback\}
Assume \forall e \in Event : e \in String
CONSTANTS
   Commit,
   Apply
Phase \triangleq \{Commit, Apply\}
\texttt{ASSUME} \ \forall \ p \in Phase : p \in \texttt{STRING}
CONSTANTS
   Pending,
   InProgress,
   Complete,
   Aborted,
   Failed
State \triangleq \{Pending, InProgress, Complete, Aborted, Failed\}
Done \triangleq \{Complete, Aborted, Failed\}
Assume \forall s \in State : s \in String
CONSTANT Path
```

```
ASSUME \forall p \in Path : p \in STRING CONSTANT Value ASSUME \forall v \in Value : v \in STRING AllValues \triangleq Value \cup \{None\} CONSTANT NumProposals ASSUME NumProposals \in Nat
```

```
This section defines model state variables.
```

```
proposal \stackrel{\Delta}{=} [i \in 1.. Nat \mapsto [
     phase \mapsto Phase,
      change \mapsto [
        values \mapsto Change,
        commit \mapsto State,
        apply \mapsto State],
      rollback \mapsto [
        index \mapsto Nat,
        values \mapsto Change,
        commit \mapsto State,
        apply \mapsto State]]]
configuration \; \stackrel{\Delta}{=} \; \; \lceil
  committed \mapsto [
     index\mapsto Nat,
     values \mapsto Change,
   applied \mapsto [
     index \mapsto Nat,
     values \mapsto Change,
     term \mapsto Nat]]
mastership \stackrel{\Delta}{=} [
   master \mapsto \text{STRING},
  term \mapsto Nat,
  conn \mapsto Nat
conn \ \stackrel{\Delta}{=} \ \ [ \ n \in Node \mapsto \ [
     id \mapsto Nat,
     connected \mapsto \texttt{BOOLEAN} ]]
target \stackrel{\Delta}{=} [
  id \mapsto Nat,
  values \mapsto Change,
  running \mapsto BOOLEAN]
{\tt VARIABLE}\ proposal
```

 ${\tt VARIABLE}\ configuration$

```
VARIABLE mastership
VARIABLE conn
VARIABLE target
VARIABLE history
vars \triangleq \langle proposal, configuration, mastership, conn, target, history \rangle
This section models configuration target.
StartTarget \stackrel{\triangle}{=}
   \land \neg target.running
   \wedge target' = [target \ EXCEPT \ !.id]
                                                 = target.id + 1,
                                     !.running = TRUE
   \land UNCHANGED \langle proposal, configuration, mastership, conn, history <math>\rangle
StopTarget \triangleq
   \land target.running
   \land target' = [target \ EXCEPT \ !.running = FALSE,
                                     !.values = [p \in \{\} \mapsto [value \mapsto None]]]
   \land conn' = [n \in Node \mapsto [conn[n] \text{ EXCEPT } !.connected = \text{FALSE}]]
   \land UNCHANGED \langle proposal, configuration, mastership, history <math>\rangle
This section models nodes connection to the configuration target.
ConnectNode(n) \triangleq
   \wedge \neg conn[n].connected
   \land target.running
   \wedge conn' = [conn \ EXCEPT \ ![n].id]
                                                    = conn[n].id + 1,
                                   ![n].connected = TRUE]
   ∧ UNCHANGED ⟨proposal, configuration, mastership, target, history⟩
DisconnectNode(n) \triangleq
   \land conn[n].connected
   \wedge conn' = [conn \ EXCEPT \ ![n].connected = FALSE]
   ∧ UNCHANGED ⟨proposal, configuration, mastership, target, history⟩
This section models mastership reconciliation.
ReconcileMastership(n) \stackrel{\Delta}{=}
   \land \lor \land conn[n].connected
          \land mastership.master = None
          \land mastership' = [master \mapsto n, term \mapsto mastership.term + 1, conn \mapsto conn[n].id]
```

```
\land mastership' = [mastership \ Except \ !.master = None]
   ∧ UNCHANGED ⟨proposal, configuration, conn, target, history⟩
This section models configuration reconciliation.
ReconcileConfiguration(n) \stackrel{\Delta}{=}
    \land mastership.master = n
   \land \lor \land configuration.status \neq InProgress
          \land configuration.applied.term < mastership.term
          \land configuration' = [configuration EXCEPT !.status = InProgress]
          \land UNCHANGED \langle target \rangle
       \lor \land configuration.status = InProgress
          \land configuration.applied.term < mastership.term
          \land conn[n].connected
          \land target.running
          \land target' = [target \ Except \ !.values = configuration.applied.values]
          \land configuration' = [configuration EXCEPT !.applied.term = mastership.term,
                                                             !.applied.target = target.id,
                                                             !.status
                                                                                 = Complete
   \land UNCHANGED \langle proposal, mastership, conn, history \rangle
This section models proposal reconcilation.
CommitChange(n, i) \stackrel{\Delta}{=}
    \land \lor \land proposal[i].change.commit = Pending
          \land \forall j \in \text{DOMAIN } proposal : j < i \Rightarrow
               \land proposal[j].change.commit \in Done
                \land proposal[j].rollback.commit \neq InProgress
          \land \lor \land proposal[i].rollback.commit = None
                \land proposal' = [proposal \ EXCEPT \ ![i].change.commit = InProgress]
             \lor \land proposal[i].rollback.commit = Pending
                \land proposal' = [proposal \ EXCEPT \ ![i].change.commit = Aborted]
          \land UNCHANGED \langle configuration, history \rangle
       \lor \land proposal[i].change.commit = InProgress
           Changes are validated during the commit phase. If a change fails validation,
           it will be marked failed before being applied to the configuration.
           If all the change values are valid, record the changes required to roll
           back the proposal and the index to which the rollback changes
           will roll back the configuration.
          \land \lor \land \text{LET } values \stackrel{\triangle}{=} [p \in \text{DOMAIN } proposal[i].values \mapsto
                                        [index \mapsto i, value \mapsto proposal[i].values[p]]]@@
```

 $\lor \land \neg conn[n].connected \\ \land mastership.master = n$

configuration.committed.values

```
\land configuration' = [configuration \ EXCEPT \ !.committed.values = values]
                        \land proposal' = [proposal \ EXCEPT \ ![i].change.commit = Complete]
                        \land history' = Append(history, [type \mapsto Change, phase \mapsto Commit, index \mapsto i])
             \lor \land proposal' = [proposal \ EXCEPT \ ![i].change.commit = Failed]
                \land UNCHANGED \langle configuration, history \rangle
   \land UNCHANGED \langle mastership, conn, target \rangle
ApplyChange(n, i) \triangleq
   \land \lor \land proposal[i].change.apply = Pending
         \land \lor \land proposal[i].change.commit = Complete
                \land \forall j \in \text{DOMAIN } proposal : j < i \Rightarrow
                     \lor \land proposal[j].change.apply = Complete
                        \land proposal[j].rollback.apply \neq InProgress
                     \lor \land proposal[j].change.apply = Failed
                        \land proposal[j].rollback.apply = Complete
                \land proposal' = [proposal \ EXCEPT \ ![i].change.apply = InProgress]
             \lor \land proposal[i].change.commit \in \{Aborted, Failed\}
                \land proposal' = [proposal \ EXCEPT \ ![i].change.apply = Aborted]
         ∧ UNCHANGED ⟨configuration, target, history⟩
      \lor \land proposal[i].change.apply = InProgress
          Verify the applied term is the current mastership term to ensure the
          configuration has been synchronized following restarts.
          \land configuration.applied.term = mastership.term
          Verify the node's connection to the target.
         \land conn[n].connected
          \land \ mastership.conn = conn[n].id
         \land target.running
          Model successful and failed target update requests.
         \land \lor \land \text{LET } values \stackrel{\triangle}{=} [p \in \text{DOMAIN } proposal[i].values \mapsto
                                        [index \mapsto i, value \mapsto proposal[i].values[p]]]
                        \land target' = [target \ EXCEPT \ !.values = values @@ target.values]
                        \land configuration' = [configuration EXCEPT !.applied.values = values @@
                                                                               configuration.applied.values
                        \land proposal' = [proposal \ EXCEPT \ ![i].change.apply = Complete]
                        \land history' = Append(history, [type \mapsto Change, phase \mapsto Apply, index \mapsto i])
             \lor \land proposal' = [proposal \ EXCEPT \ ![i].change.apply = Failed]
                \land UNCHANGED \langle configuration, target, history \rangle
   \land UNCHANGED \langle mastership, conn \rangle
CommitRollback(n, i) \triangleq
   \land \lor \land proposal[i].rollback.commit = Pending
         \land \forall j \in \text{DOMAIN } proposal :
              \wedge j > i
               \land proposal[j].phase \neq None
               \land proposal[j].change.commit \neq Pending
```

```
\Rightarrow proposal[j].rollback.commit = Complete
          \land \lor \land proposal[i].change.commit = Aborted
                \land proposal' = [proposal \ EXCEPT \ ![i].rollback.commit = Complete]
             \lor \land proposal[i].change.commit \in \{Complete, Failed\}
                \land proposal' = [proposal \ EXCEPT \ ![i].rollback.commit = InProgress]
          \land UNCHANGED \langle configuration, history \rangle
       \lor \land proposal[i].rollback.commit = InProgress
          \land LET changes \stackrel{\triangle}{=} \{j \in \text{DOMAIN proposal}:
                                     \wedge i < i
                                     \land proposal[j].change.commit = Complete
                                     \land proposal[j].rollback.commit \neq Complete
                              \stackrel{\Delta}{=} \{ p \in \text{DOMAIN } configuration.committed.values : 
                  paths
                                     \exists j \in changes : p \in domain proposal[j].values
                   indexes \stackrel{\triangle}{=} [p \in paths \mapsto \text{CHOOSE } j \in changes :
                                     \land p \in \text{DOMAIN } proposal[j].values
                                     \land \neg \exists k \in changes : k > j \land p \in DOMAIN \ proposal[k].values]
                              \stackrel{\Delta}{=} [p \in \text{DOMAIN} configuration.committed.values \mapsto
                   values
                                    IF p \in paths Then
                                        [index \mapsto indexes[p], value \mapsto proposal[indexes[p]].values[p]]
                                        [index \mapsto 0, value \mapsto None]]
             IN
                 \land configuration' = [configuration \ EXCEPT \ !.committed.values = values]
                 \land proposal' = [proposal \ EXCEPT \ ![i].rollback.commit = Complete]
                 \land history' = Append(history, [type \mapsto Rollback, phase \mapsto Commit, index \mapsto i])
    \land UNCHANGED \langle mastership, conn, target \rangle
ApplyRollback(n, i) \stackrel{\triangle}{=}
    \land \lor \land proposal[i].rollback.apply = Pending
          \land proposal[i].rollback.commit = Complete
          \land \forall j \in DOMAIN \ proposal :
               \wedge j > i
               \land proposal[j].phase \neq None
               \land proposal[j].change.apply \neq Pending
                \Rightarrow proposal[j].rollback.apply \in Done
          \land \lor \land proposal[i].change.apply = Pending
                \land proposal' = [proposal \ EXCEPT \ ![i].change.apply = Aborted,
                                                         ![i].rollback.apply = Complete]
             \lor \land proposal[i].change.apply \in Done
                \land proposal' = [proposal \ EXCEPT \ ![i].rollback.apply = InProgress]
          \land UNCHANGED \langle configuration, target, history \rangle
       \lor \land proposal[i].rollback.apply = InProgress
           Verify the applied term is the current mastership term to ensure the
           configuration has been synchronized following restarts.
          \land configuration.applied.term = mastership.term
```

```
Verify the node's connection to the target.
          \land conn[n].connected
          \land target.running
          \land Let changes \stackrel{\triangle}{=} \{j \in \text{Domain } proposal :
                                     \wedge j < i
                                      \land proposal[j].change.apply = Complete
                                      \land proposal[j].rollback.apply \neq Complete
                               \stackrel{\triangle}{=} \{ p \in \text{DOMAIN } configuration.applied.values : 
                   paths
                                      \exists j \in changes : p \in DOMAIN \ proposal[j].values \}
                   indexes \stackrel{\triangle}{=} [p \in paths \mapsto \text{CHOOSE } j \in changes :
                                      \land p \in \text{DOMAIN } proposal[j].values
                                      \land \neg \exists k \in changes : k > j \land p \in domain proposal[k].values]
                              \stackrel{\triangle}{=} [p \in \text{DOMAIN } configuration.applied.values \mapsto
                   values
                                     If p \in paths then
                                         [index \mapsto indexes[p], value \mapsto proposal[indexes[p]].values[p]]
                                         [index \mapsto 0, value \mapsto None]]
             ΙN
                 \land target' = [target \ EXCEPT \ !.values = values]
                 \land configuration' = [configuration \ EXCEPT \ !.applied.values = values]
                 \land proposal' = [proposal \ EXCEPT \ ![i].rollback.apply = Complete]
                 \land history' = Append(history, [type \mapsto Rollback, phase \mapsto Apply, index \mapsto i])
    \land UNCHANGED \langle mastership, conn \rangle
ReconcileProposal(n, i) \triangleq
    \land mastership.master = n
    \land \lor CommitChange(n, i)
       \vee ApplyChange(n, i)
       \vee CommitRollback(n, i)
       \vee ApplyRollback(n, i)
    \land UNCHANGED \langle mastership, conn \rangle
This section models changes to the proposal queue.
 Propose change at index 'i'
ProposeChange(i) \triangleq
    \land proposal[i].phase = None
    \land i-1 \in \text{DOMAIN } proposal \Rightarrow proposal[i-1].phase \neq None
    \land \exists p \in Path, v \in AllValues:
         \land proposal' = [proposal \ EXCEPT \ ![i].phase]
                                                                         = Change,
                                                   ![i].values = (p:>v),
                                                   ![i].change.commit = Pending,
                                                   ![i].change.apply = Pending
    \land UNCHANGED \langle configuration, mastership, conn, target, history <math>\rangle
```

```
\begin{aligned} & \text{Rollback proposed change at index 'i'} \\ & ProposeRollback(i) \stackrel{\triangle}{=} \\ & \land proposal[i].phase = Change \\ & \land proposal' = [proposal \text{ EXCEPT } ![i].phase & = Rollback, \\ & ![i].rollback.commit = Pending, \\ & ![i].rollback.apply & = Pending] \\ & \land \text{UNCHANGED } \langle configuration, \ mastership, \ conn, \ target, \ history \rangle \end{aligned}
```

Formal specification, constraints, and theorems.

```
Init \triangleq
    \land proposal = [
           i \in 1 \dots NumProposals \mapsto [
              phase
                          \mapsto None,
                          \mapsto [p \in \{\} \mapsto None],
              values
              change \mapsto [
                 commit \mapsto None,
                 apply \mapsto None,
              rollback \mapsto [
                  commit \mapsto None,
                 apply \mapsto None
    \land \ configuration = \lceil
           committed \mapsto [
               values \mapsto [p \in \{\} \mapsto [index \mapsto 0, value \mapsto None]]],
           applied \mapsto \lceil
               term \mapsto 0,
               target \mapsto 0.
               values \mapsto [p \in \{\} \mapsto [index \mapsto 0, value \mapsto None]]],
           status \mapsto Pending
    \land mastership = [master \mapsto None, term \mapsto 0, conn \mapsto 0]
    \land conn = [n \in Node \mapsto [id \mapsto 0, connected \mapsto FALSE]]
    \land target = [
           values \mapsto [p \in \{\} \mapsto [index \mapsto 0, value \mapsto None]],
           running \mapsto FALSE
    \wedge history = \langle \rangle
Next \triangleq
    \vee \exists i \in 1 ... NumProposals :
          \vee ProposeChange(i)
          \vee ProposeRollback(i)
    \vee \exists n \in Node, i \in DOMAIN \ proposal : ReconcileProposal(n, i)
    \vee \exists n \in Node : ReconcileConfiguration(n)
    \vee \exists n \in Node : ReconcileMastership(n)
    \vee \exists n \in Node:
```

```
\vee ConnectNode(n)
        \vee DisconnectNode(n)
    \vee StartTarget
    \lor StopTarget
Spec \triangleq
    \wedge Init
    \wedge \Box [Next]_{vars}
    \land \forall i \in 1 ... NumProposals : WF_{vars}(ProposeChange(i) \lor ProposeRollback(i))
    \land \forall n \in Node, i \in 1... NumProposals : WF_{vars}(ReconcileProposal(n, i))
    \land \forall n \in Node : WF_{\langle configuration, \, mastership, \, conn, \, target \rangle}(ReconcileConfiguration(n))
    \land \forall n \in Node : WF_{\langle mastership, conn, target \rangle}(ReconcileMastership(n))
    \land \forall n \in Node : WF_{\langle conn, target \rangle}(ConnectNode(n) \lor DisconnectNode(n))
    \wedge \operatorname{WF}_{\langle target \rangle}(StartTarget)
    \wedge \operatorname{WF}_{\langle target \rangle}(StopTarget)
IsOrderedChange(p, i) \stackrel{\Delta}{=}
    \land history[i].type = Change
    \land history[i].phase = p
    \wedge \neg \exists j \in \text{DOMAIN } history :
            \wedge j < i
            \land history[j].type = Change
            \land history[j].phase = p
            \land history[j].index \ge history[i].index
IsOrderedRollback(p, i) \stackrel{\Delta}{=}
    \land history[i].type = Rollback
    \land history[i].phase = p
    \wedge \neg \exists j \in \text{DOMAIN } history :
            \wedge j < i
            \land history[j].type = Change
            \land history[j].phase = p
            \land history[j].index > history[i].index
            \wedge \neg \exists k \in \text{DOMAIN } history :
                    \wedge k > j
                     \land k < i
                     \land history[k].type = Rollback
                     \land history[k].phase = p
                     \land history[k].index = history[j].index
Order \triangleq
    \land \forall i \in \text{DOMAIN } history:
         \vee IsOrderedChange(Commit, i)
         \vee IsOrderedChange(Apply, i)
         \vee IsOrderedRollback(Commit, i)
         \vee IsOrderedRollback(Apply, i)
```

```
\land \forall i \in DOMAIN \ proposal :
          \land proposal[i].change.apply = Failed
          \land proposal[i].rollback.apply \neq Complete
          \Rightarrow \forall j \in \text{DOMAIN } proposal : j > i \Rightarrow
                proposal[j].change.apply \in \{None, Pending, Aborted\}
Consistency \triangleq
    \land target.running
    \land configuration.status = Complete
    \land configuration.applied.target = target.id
    \Rightarrow \forall i \in \text{DOMAIN } proposal :
           \land proposal[i].change.apply = Complete
          \land proposal[i].rollback.apply \neq Complete
           \Rightarrow \forall p \in \text{DOMAIN } proposal[i].values :
                 \wedge \neg \exists j \in DOMAIN \ proposal :
                        \wedge j > i
                        \land proposal[j].change.apply = Complete
                        \land proposal[j].rollback.apply \neq Complete
                 \Rightarrow \land p \in \text{DOMAIN } target.values
                     \land \ target.values[p].value = proposal[i].values[p]
                     \land target.values[p].index = i
Safety \triangleq \Box(Order \land Consistency)
THEOREM Spec \Rightarrow Safety
Termination \triangleq
   \forall i \in 1 ... NumProposals:
      \land proposal[i].change.commit = Pending \leadsto
            proposal[i].change.commit \in Done
      \land proposal[i].change.apply = Pending \leadsto
            proposal[i].change.apply \in Done
      \land proposal[i].rollback.commit = Pending \leadsto
            proposal[i].rollback.commit \in Done
      \land proposal[i].rollback.apply = Pending \leadsto
            proposal[i].rollback.apply \in Done
Liveness \triangleq Termination
Theorem Spec \Rightarrow Liveness
```