MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

LOCAL INSTANCE *TLC*

This section specifies constant parameters for the model.

CONSTANT *None*

ASSUME $\mathit{None} \in$ STRING

CONSTANT *Node*

ASSUME $\forall\, n \in \mathit{Node} : n \in$ STRING

CONSTANTS
  Change,
  Rollback

$\mathit{Event} \triangleq \{\mathit{Change}, \mathit{Rollback}\}$

ASSUME $\forall\, e \in \mathit{Event} : e \in$ STRING

CONSTANTS
  Commit,
  Apply

$\mathit{Phase} \triangleq \{\mathit{Commit}, \mathit{Apply}\}$

ASSUME $\forall\, p \in \mathit{Phase} : p \in$ STRING

CONSTANTS
  Pending,
  InProgress,
  Complete,
  Aborted,
  Failed

$\mathit{State} \triangleq \{\mathit{Pending}, \mathit{InProgress}, \mathit{Complete}, \mathit{Aborted}, \mathit{Failed}\}$

$\mathit{Done} \triangleq \{\mathit{Complete}, \mathit{Aborted}, \mathit{Failed}\}$

ASSUME $\forall\, s \in \mathit{State} : s \in$ STRING

CONSTANT *Path*

```
ASSUME ∀ p ∈ Path : p ∈ STRING

CONSTANT Value

ASSUME ∀ v ∈ Value : v ∈ STRING

AllValues ≜ Value ∪ {None}

CONSTANT NumProposals

ASSUME NumProposals ∈ Nat
```

This section defines model state variables.

```
proposal ≜ [ i ∈ 1 . . Nat ↦ [
    phase ↦ Phase,
    change ↦ [
      values ↦ Change,
      commit ↦ State,
      apply ↦ State],
    rollback ↦ [
      index ↦ Nat,
      values ↦ Change,
      commit ↦ State,
      apply ↦ State]]]

configuration ≜ [
  committed ↦ [
    index ↦ Nat,
    values ↦ Change],
  applied ↦ [
    index ↦ Nat,
    values ↦ Change,
    term ↦ Nat]]

mastership ≜ [
  master ↦ STRING,
  term ↦ Nat,
  conn ↦ Nat]

conn ≜ [ n ∈ Node ↦ [
    id        ↦ Nat,
    connected ↦ BOOLEAN ]]

target ≜ [
  id      ↦ Nat,
  values  ↦ Change,
  running ↦ BOOLEAN ]

VARIABLE proposal

VARIABLE configuration
VARIABLE mastership

VARIABLE conn

VARIABLE target

VARIABLE history

vars ≜ ⟨proposal, configuration, mastership, conn, target, history⟩
```

This section models configuration target.

```
StartTarget ≜
    ∧ ¬target.running
    ∧ target′ = [target EXCEPT !.id      = target.id + 1,
                               !.running = TRUE]
    ∧ UNCHANGED ⟨proposal, configuration, mastership, conn, history⟩

StopTarget ≜
    ∧ target.running
    ∧ target′ = [target EXCEPT !.running = FALSE,
                               !.values  = [p ∈ {} ↦ [value ↦ None]]]
    ∧ conn′ = [n ∈ Node ↦ [conn[n] EXCEPT !.connected = FALSE]]
    ∧ UNCHANGED ⟨proposal, configuration, mastership, history⟩
```

This section models nodes connection to the configuration target.

```
ConnectNode(n) ≜
    ∧ ¬conn[n].connected
    ∧ target.running
    ∧ conn′ = [conn EXCEPT ![n].id        = conn[n].id + 1,
                           ![n].connected = TRUE]
    ∧ UNCHANGED ⟨proposal, configuration, mastership, target, history⟩

DisconnectNode(n) ≜
    ∧ conn[n].connected
    ∧ conn′ = [conn EXCEPT ![n].connected = FALSE]
    ∧ UNCHANGED ⟨proposal, configuration, mastership, target, history⟩
```

This section models *mastership* reconciliation.

```
ReconcileMastership(n) ≜
    ∧ ∨ ∧ conn[n].connected
        ∧ mastership.master = None
        ∧ mastership′ = [master ↦ n, term ↦ mastership.term + 1, conn ↦ conn[n].id]
```

```
     ∨ ∧ ¬conn[n].connected
       ∧ mastership.master = n
       ∧ mastership′ = [mastership EXCEPT !.master = None]
   ∧ UNCHANGED ⟨proposal, configuration, conn, target, history⟩
```

This section models configuration reconciliation.

```
ReconcileConfiguration(n) ≜
   ∧ mastership.master = n
   ∧ ∨ ∧ configuration.status ≠ InProgress
       ∧ configuration.applied.term < mastership.term
       ∧ configuration′ = [configuration EXCEPT !.status = InProgress]
       ∧ UNCHANGED ⟨target⟩
     ∨ ∧ configuration.status = InProgress
       ∧ configuration.applied.term < mastership.term
       ∧ conn[n].connected
       ∧ target.running
       ∧ target′ = [target EXCEPT !.values = configuration.applied.values]
       ∧ configuration′ = [configuration EXCEPT !.applied.term   = mastership.term,
                                                 !.applied.target = target.id,
                                                 !.status         = Complete]
   ∧ UNCHANGED ⟨proposal, mastership, conn, history⟩
```

This section models proposal reconcilation.

```
CommitChange(n, i) ≜
   ∧ ∨ ∧ proposal[i].change.commit = Pending
       ∧ ∀ j ∈ DOMAIN proposal : j < i ⇒
           ∧ proposal[j].change.commit ∈ Done
           ∧ proposal[j].rollback.commit ≠ InProgress
       ∧ ∨ ∧ proposal[i].rollback.commit = None
             ∧ proposal′ = [proposal EXCEPT ![i].change.commit = InProgress]
         ∨ ∧ proposal[i].rollback.commit = Pending
             ∧ proposal′ = [proposal EXCEPT ![i].change.commit = Aborted]
       ∧ UNCHANGED ⟨configuration, history⟩
     ∨ ∧ 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.
       ∧ ∨ ∧ LET values ≜ [p ∈ DOMAIN proposal[i].values ↦
                                [index ↦ i, value ↦ proposal[i].values[p]]] @@
                                    configuration.committed.values
```

```
                IN  $\land configuration' = [configuration \text{ EXCEPT } !.committed.values = values]$
                    $\land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = Complete]$
                    $\land history' = Append(history, [type \mapsto Change, phase \mapsto Commit, index \mapsto i])$
            $\lor \land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = Failed]$
              $\land \text{UNCHANGED } \langle configuration, history \rangle$
    $\land \text{UNCHANGED } \langle mastership, conn, target \rangle$
```

```
$ApplyChange(n, i) \triangleq$
    $\land \lor \land proposal[i].change.apply = Pending$
            $\land \lor \land proposal[i].change.commit = Complete$
                  $\land \forall j \in \text{DOMAIN } proposal : j < i \Rightarrow$
                        $\lor \land proposal[j].change.apply = Complete$
                          $\land proposal[j].rollback.apply \neq InProgress$
                        $\lor \land proposal[j].change.apply = Failed$
                          $\land proposal[j].rollback.apply = Complete$
                  $\land i - 1 \in \text{DOMAIN } proposal \land proposal[i-1].change.apply = Failed \Rightarrow$
                        $proposal[i-1].rollback.apply = Complete$
                  $\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = InProgress]$
               $\lor \land proposal[i].change.commit \in \{Aborted, Failed\}$
                  $\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = Aborted]$
            $\land \text{UNCHANGED } \langle configuration, target, history \rangle$
       $\lor \land proposal[i].change.apply = InProgress$
            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.
            $\land \lor \land \text{LET } values \triangleq [p \in \text{DOMAIN } proposal[i].values \mapsto$
                                        $[index \mapsto i, value \mapsto proposal[i].values[p]]]$
                       IN  $\land target' = [target \text{ EXCEPT } !.values = values \ @@ \ target.values]$
                           $\land configuration' = [configuration \text{ EXCEPT } !.applied.values = values \ @@$
                                                                    $configuration.applied.values]$
                           $\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = Complete]$
                           $\land history' = Append(history, [type \mapsto Change, phase \mapsto Apply, index \mapsto i])$
               $\lor \land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = Failed]$
                  $\land \text{UNCHANGED } \langle configuration, target, history \rangle$
    $\land \text{UNCHANGED } \langle mastership, conn \rangle$
```

```
$CommitRollback(n, i) \triangleq$
    $\land \lor \land proposal[i].rollback.commit = Pending$
            $\land \forall j \in \text{DOMAIN } proposal :$
                 $\land j > i$
```

```
          $\land\ \mathit{proposal}[j].\mathit{phase} \neq \mathit{None}$
          $\land\ \mathit{proposal}[j].\mathit{change}.\mathit{commit} \neq \mathit{Pending}$
          $\Rightarrow \mathit{proposal}[j].\mathit{rollback}.\mathit{commit} = \mathit{Complete}$
      $\land\ \lor\ \land\ \mathit{proposal}[i].\mathit{change}.\mathit{commit} = \mathit{Aborted}$
            $\land\ \mathit{proposal}' = [\mathit{proposal}\ \textsc{except}\ ![i].\mathit{rollback}.\mathit{commit} = \mathit{Complete}]$
          $\lor\ \land\ \mathit{proposal}[i].\mathit{change}.\mathit{commit} \in \{\mathit{Complete}, \mathit{Failed}\}$
            $\land\ \mathit{proposal}' = [\mathit{proposal}\ \textsc{except}\ ![i].\mathit{rollback}.\mathit{commit} = \mathit{InProgress}]$
      $\land\ \textsc{unchanged}\ \langle \mathit{configuration}, \mathit{history} \rangle$
    $\lor\ \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{commit} = \mathit{InProgress}$
      $\land\ \textsc{let}\ \mathit{changes} \triangleq \{j \in \textsc{domain}\ \mathit{proposal} :$
                                   $\land\ j < i$
                                   $\land\ \mathit{proposal}[j].\mathit{change}.\mathit{commit} = \mathit{Complete}$
                                   $\land\ \mathit{proposal}[j].\mathit{rollback}.\mathit{commit} \neq \mathit{Complete}\}$
               $\mathit{paths} \quad\ \triangleq \{p \in \textsc{domain}\ \mathit{configuration}.\mathit{committed}.\mathit{values} :$
                                   $\exists\, j \in \mathit{changes} : p \in \textsc{domain}\ \mathit{proposal}[j].\mathit{values}\}$
               $\mathit{indexes} \triangleq [p \in \mathit{paths} \mapsto \textsc{choose}\ j \in \mathit{changes} :$
                                   $\land\ p \in \textsc{domain}\ \mathit{proposal}[j].\mathit{values}$
                                   $\land\ \neg \exists\, k \in \mathit{changes} : k > j \land p \in \textsc{domain}\ \mathit{proposal}[k].\mathit{values}]$
               $\mathit{values} \quad \triangleq [p \in \textsc{domain}\ \mathit{configuration}.\mathit{committed}.\mathit{values} \mapsto$
                                 $\textsc{if}\ p \in \mathit{paths}\ \textsc{then}$
                                    $[\mathit{index} \mapsto \mathit{indexes}[p], \mathit{value} \mapsto \mathit{proposal}[\mathit{indexes}[p]].\mathit{values}[p]]$
                                  $\textsc{else}$
                                    $[\mathit{index} \mapsto 0, \mathit{value} \mapsto \mathit{None}]]$
         $\textsc{in}$
            $\land\ \mathit{configuration}' = [\mathit{configuration}\ \textsc{except}\ !.\mathit{committed}.\mathit{values} = \mathit{values}]$
            $\land\ \mathit{proposal}' = [\mathit{proposal}\ \textsc{except}\ ![i].\mathit{rollback}.\mathit{commit} = \mathit{Complete}]$
            $\land\ \mathit{history}' = \mathit{Append}(\mathit{history}, [\mathit{type} \mapsto \mathit{Rollback}, \mathit{phase} \mapsto \mathit{Commit}, \mathit{index} \mapsto i])$
  $\land\ \textsc{unchanged}\ \langle \mathit{mastership}, \mathit{conn}, \mathit{target} \rangle$

$\mathit{ApplyRollback}(n, i) \triangleq$
  $\land\ \lor\ \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{apply} = \mathit{Pending}$
      $\land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{commit} = \mathit{Complete}$
      $\land\ \forall\, j \in \textsc{domain}\ \mathit{proposal} :$
          $\land\ j > i$
          $\land\ \mathit{proposal}[j].\mathit{phase} \neq \mathit{None}$
          $\land\ \mathit{proposal}[j].\mathit{change}.\mathit{apply} \neq \mathit{Pending}$
          $\Rightarrow \mathit{proposal}[j].\mathit{rollback}.\mathit{apply} \in \mathit{Done}$
      $\land\ \lor\ \land\ \mathit{proposal}[i].\mathit{change}.\mathit{apply} = \mathit{Pending}$
            $\land\ \mathit{proposal}' = [\mathit{proposal}\ \textsc{except}\ ![i].\mathit{change}.\mathit{apply} \quad = \mathit{Aborted},$
                                                        $![i].\mathit{rollback}.\mathit{apply} \ \ = \mathit{Complete}]$
          $\lor\ \land\ \mathit{proposal}[i].\mathit{change}.\mathit{apply} \in \mathit{Done}$
            $\land\ \mathit{proposal}' = [\mathit{proposal}\ \textsc{except}\ ![i].\mathit{rollback}.\mathit{apply} = \mathit{InProgress}]$
      $\land\ \textsc{unchanged}\ \langle \mathit{configuration}, \mathit{target}, \mathit{history} \rangle$
    $\lor\ \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{apply} = \mathit{InProgress}$
         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 $changes \triangleq \{j \in$ DOMAIN $proposal :$
                                $\land\ j < i$
                                $\land\ proposal[j].change.apply = Complete$
                                $\land\ proposal[j].rollback.apply \neq Complete\}$
                $paths \triangleq \{p \in$ DOMAIN $configuration.applied.values :$
                                $\exists\, j \in changes : p \in$ DOMAIN $proposal[j].values\}$
                $indexes \triangleq [p \in paths \mapsto$ CHOOSE $j \in changes :$
                                $\land\ p \in$ DOMAIN $proposal[j].values$
                                $\land\ \neg\exists\, k \in changes : k > j \land p \in$ DOMAIN $proposal[k].values]$
                $values \triangleq [p \in$ DOMAIN $configuration.applied.values \mapsto$
                            IF $p \in paths$ THEN
                               $[index \mapsto indexes[p],\ value \mapsto proposal[indexes[p]].values[p]]$
                            ELSE
                               $[index \mapsto 0,\ value \mapsto None]]$
          IN
             $\land\ target' = [target$ 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 $\langle mastership,\ conn\rangle$

$ReconcileProposal(n,\ i) \triangleq$
    $\land\ mastership.master = n$
    $\land\ \lor\ CommitChange(n,\ i)$
       $\lor\ ApplyChange(n,\ i)$
       $\lor\ CommitRollback(n,\ i)$
       $\lor\ ApplyRollback(n,\ i)$
    $\land$ UNCHANGED $\langle mastership,\ conn\rangle$
```

This section models changes to the proposal queue.

```
 Propose change at index 'i'
$ProposeChange(i) \triangleq$
    $\land\ proposal[i].phase = None$
    $\land\ i - 1 \in$ DOMAIN $proposal \Rightarrow proposal[i-1].phase \neq None$
    $\land\ \exists\, p \in Path,\ v \in AllValues :$
         $\land\ proposal' = [proposal$ EXCEPT $![i].phase          = Change,$
                                              $![i].values = (p :> v),$
                                              $![i].change.commit = Pending,$
                                              $![i].change.apply    = Pending]$
    ∧ UNCHANGED ⟨configuration, mastership, conn, target, history⟩
```

Rollback proposed change at index 'i'

```
ProposeRollback(i) ≜
    ∧ proposal[i].phase = Change
    ∧ proposal′ = [proposal EXCEPT ![i].phase           = Rollback,
                                   ![i].rollback.commit = Pending,
                                   ![i].rollback.apply  = Pending]
    ∧ UNCHANGED ⟨configuration, mastership, conn, target, history⟩
```

Formal specification, constraints, and theorems.

```
Init ≜
    ∧ proposal = [
          i ∈ 1 . . NumProposals ↦ [
            phase    ↦ None,
            values   ↦ [p ∈ {} ↦ None],
            change   ↦ [
                commit ↦ None,
                apply  ↦ None],
            rollback ↦ [
                commit ↦ None,
                apply  ↦ None]]]
    ∧ configuration = [
          committed ↦ [
              values  ↦ [p ∈ {} ↦ [index ↦ 0, value ↦ None]]],
          applied ↦ [
              term    ↦ 0,
              target  ↦ 0,
              values  ↦ [p ∈ {} ↦ [index ↦ 0, value ↦ None]]],
          status  ↦ Pending]
    ∧ mastership = [master ↦ None, term ↦ 0, conn ↦ 0]
    ∧ conn = [n ∈ Node ↦ [id ↦ 0, connected ↦ FALSE]]
    ∧ target = [
          id       ↦ 0,
          values   ↦ [p ∈ {} ↦ [index ↦ 0, value ↦ None]],
          running  ↦ FALSE]
    ∧ history = ⟨⟩

Next ≜
    ∨ ∃ i ∈ 1 . . NumProposals :
          ∨ ProposeChange(i)
          ∨ ProposeRollback(i)
    ∨ ∃ n ∈ Node, i ∈ DOMAIN proposal : ReconcileProposal(n, i)
    ∨ ∃ n ∈ Node : ReconcileConfiguration(n)
    $\lor \exists\, n \in Node : ReconcileMastership(n)$
    $\lor \exists\, n \in Node :$
        $\lor\ ConnectNode(n)$
        $\lor\ DisconnectNode(n)$
    $\lor\ StartTarget$
    $\lor\ StopTarget$

$Spec\ \triangleq$
    $\land\ Init$
    $\land\ \Box[Next]_{vars}$
    $\land\ \forall\, i\ \in 1\, ..\, NumProposals : \mathrm{WF}_{vars}(ProposeChange(i) \lor ProposeRollback(i))$
    $\land\ \forall\, n \in Node,\ i \in 1\, ..\, NumProposals : \mathrm{WF}_{vars}(ReconcileProposal(n,\ i))$
    $\land\ \forall\, n \in Node : \mathrm{WF}_{\langle configuration,\ mastership,\ conn,\ target\rangle}(ReconcileConfiguration(n))$
    $\land\ \forall\, n \in Node : \mathrm{WF}_{\langle mastership,\ conn,\ target\rangle}(ReconcileMastership(n))$
    $\land\ \forall\, n \in Node : \mathrm{WF}_{\langle conn,\ target\rangle}(ConnectNode(n) \lor DisconnectNode(n))$
    $\land\ \mathrm{WF}_{\langle target\rangle}(StartTarget)$
    $\land\ \mathrm{WF}_{\langle target\rangle}(StopTarget)$

$IsOrderedChange(p,\ i)\ \triangleq$
    $\land\ history[i].type = Change$
    $\land\ history[i].phase = p$
    $\land\ \neg\exists\, j \in \textsc{domain}\ history :$
            $\land\ j < i$
            $\land\ history[j].type = Change$
            $\land\ history[j].phase = p$
            $\land\ history[j].index \geq history[i].index$

$IsOrderedRollback(p,\ i)\ \triangleq$
    $\land\ history[i].type = Rollback$
    $\land\ history[i].phase = p$
    $\land\ \neg\exists\, j \in \textsc{domain}\ history :$
            $\land\ j < i$
            $\land\ history[j].type = Change$
            $\land\ history[j].phase = p$
            $\land\ history[j].index > history[i].index$
            $\land\ \neg\exists\, k \in \textsc{domain}\ history :$
                    $\land\ k > j$
                    $\land\ k < i$
                    $\land\ history[k].type = Rollback$
                    $\land\ history[k].phase = p$
                    $\land\ history[k].index = history[j].index$

$Order\ \triangleq$
    $\land\ \ \forall\, i \in \textsc{domain}\ history :$
        $\lor\ IsOrderedChange(Commit,\ i)$
        $\lor\ IsOrderedChange(Apply,\ i)$
        $\lor IsOrderedRollback(Commit, i)$
        $\lor IsOrderedRollback(Apply, i)$
    $\land \forall i \in \textsc{domain}\ proposal :$
           $\land proposal[i].change.apply = Failed$
           $\land proposal[i].rollback.apply \neq Complete$
           $\Rightarrow \forall j \in \textsc{domain}\ proposal : j > i \Rightarrow$
                 $proposal[j].change.apply \in \{None, Pending, Aborted\}$

$Consistency \triangleq$
    $\land target.running$
    $\land configuration.status = Complete$
    $\land configuration.applied.target = target.id$
    $\Rightarrow \forall i \in \textsc{domain}\ proposal :$
           $\land proposal[i].change.apply = Complete$
           $\land proposal[i].rollback.apply \neq Complete$
           $\Rightarrow \forall p \in \textsc{domain}\ proposal[i].values :$
                 $\land \neg\exists j \in \textsc{domain}\ proposal :$
                        $\land j > i$
                        $\land proposal[j].change.apply = Complete$
                        $\land proposal[j].rollback.apply \neq Complete$
                 $\Rightarrow \land p \in \textsc{domain}\ target.values$
                     $\land target.values[p].value = proposal[i].values[p]$
                     $\land target.values[p].index = i$

$Safety \triangleq \Box(Order \land Consistency)$

$\textsc{theorem}\ Spec \Rightarrow Safety$

$Termination \triangleq$
   $\forall i \in 1 \mathrel{..} NumProposals :$
      $\land proposal[i].change.commit = Pending \leadsto$
            $proposal[i].change.commit \in Done$
      $\land proposal[i].change.apply = Pending \leadsto$
            $proposal[i].change.apply \in Done$
      $\land proposal[i].rollback.commit = Pending \leadsto$
            $proposal[i].rollback.commit \in Done$
      $\land proposal[i].rollback.apply = Pending \leadsto$
            $proposal[i].rollback.apply \in Done$

$Liveness \triangleq Termination$

$\textsc{theorem}\ Spec \Rightarrow Liveness$
```