MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

LOCAL INSTANCE *TLC*

This section specifies constant parameters for the model.

CONSTANT *None*

ASSUME $None \in$ 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

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

ASSUME $\forall\, p \in Phase : p \in$ 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*

$\textsc{assume}\ \forall\, p \in \mathit{Path} : p \in \textsc{string}$

$\textsc{constant}\ \mathit{Value}$

$\textsc{assume}\ \forall\, v \in \mathit{Value} : v \in \textsc{string}$

$\mathit{AllValues} \triangleq \mathit{Value} \cup \{\mathit{None}\}$

$\textsc{constant}\ \mathit{NumProposals}$

$\textsc{assume}\ \mathit{NumProposals} \in \mathit{Nat}$

This section defines model state variables.

```
$proposal \triangleq [i \in 1 \mathrel{..} Nat \mapsto [$
    $phase \mapsto Phase,$
    $change \mapsto [$
      $values \mapsto Change,$
      $commit \mapsto State,$
      $apply \mapsto State],$
    $rollback \mapsto [$
      $index \mapsto Nat,$
      $values \mapsto Change,$
      $commit \mapsto State,$
      $apply \mapsto State]]]$
$configuration \triangleq [$
  $committed \mapsto [$
    $index \mapsto Nat,$
    $values \mapsto Change],$
  $applied \mapsto [$
    $index \mapsto Nat,$
    $values \mapsto Change,$
    $term \mapsto Nat]]$
$mastership \triangleq [$
  $master \mapsto \textsc{string},$
  $term \mapsto Nat,$
  $conn \mapsto Nat]$
$conn \triangleq [n \in Node \mapsto [$
    $id        \mapsto Nat,$
    $connected \mapsto \textsc{boolean}]]$
$target \triangleq [$
  $id      \mapsto Nat,$
  $values  \mapsto Change,$
  $running \mapsto \textsc{boolean}]$
```

$\textsc{variable}\ \mathit{proposal}$

$\textsc{variable}\ \mathit{configuration}$

$\textsc{variable}\ \mathit{mastership}$

$\textsc{variable}\ \mathit{conn}$

$\textsc{variable}\ \mathit{target}$

$\textsc{variable}\ \mathit{history}$

$\mathit{vars} \triangleq \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{conn}, \mathit{target}, \mathit{history} \rangle$

This section models configuration target.

$$
\begin{array}{l}
\mathit{StartTarget} \triangleq \\
\quad \land \lnot \mathit{target}.\mathit{running} \\
\quad \land \mathit{target}' = [\mathit{target}\ \textsc{except}\ !.\mathit{id} = \mathit{target}.\mathit{id} + 1, \\
\quad \qquad \qquad \qquad \qquad \qquad \quad !.\mathit{running} = \textsc{true}] \\
\quad \land \textsc{unchanged}\ \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{conn}, \mathit{history} \rangle
\end{array}
$$

$$
\begin{array}{l}
\mathit{StopTarget} \triangleq \\
\quad \land \mathit{target}.\mathit{running} \\
\quad \land \mathit{target}' = [\mathit{target}\ \textsc{except}\ !.\mathit{running} = \textsc{false}, \\
\quad \qquad \qquad \qquad \qquad \qquad \quad !.\mathit{values} \ \ = [p \in \{\} \mapsto [\mathit{value} \mapsto \mathit{None}]]] \\
\quad \land \mathit{conn}' = [n \in \mathit{Node} \mapsto [\mathit{conn}[n]\ \textsc{except}\ !.\mathit{connected} = \textsc{false}]] \\
\quad \land \textsc{unchanged}\ \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{history} \rangle
\end{array}
$$

This section models nodes connection to the configuration target.

$$
\begin{array}{l}
\mathit{ConnectNode}(n) \triangleq \\
\quad \land \lnot \mathit{conn}[n].\mathit{connected} \\
\quad \land \mathit{target}.\mathit{running} \\
\quad \land \mathit{conn}' = [\mathit{conn}\ \textsc{except}\ ![n].\mathit{id} = \mathit{conn}[n].\mathit{id} + 1, \\
\quad \qquad \qquad \qquad \qquad \quad ![n].\mathit{connected} = \textsc{true}] \\
\quad \land \textsc{unchanged}\ \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{target}, \mathit{history} \rangle
\end{array}
$$

$$
\begin{array}{l}
\mathit{DisconnectNode}(n) \triangleq \\
\quad \land \mathit{conn}[n].\mathit{connected} \\
\quad \land \mathit{conn}' = [\mathit{conn}\ \textsc{except}\ ![n].\mathit{connected} = \textsc{false}] \\
\quad \land \textsc{unchanged}\ \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{target}, \mathit{history} \rangle
\end{array}
$$

This section models *mastership* reconciliation.

$$
\begin{array}{l}
\mathit{ReconcileMastership}(n) \triangleq \\
\quad \land \lor \land \mathit{conn}[n].\mathit{connected} \\
\quad \qquad \land \mathit{mastership}.\mathit{master} = \mathit{None} \\
\quad \qquad \land \mathit{mastership}' = [\mathit{master} \mapsto n, \mathit{term} \mapsto \mathit{mastership}.\mathit{term} + 1, \mathit{conn} \mapsto \mathit{conn}[n].\mathit{id}]
\end{array}
$$

```
        $\lor \land \neg conn[n].connected$
          $\land mastership.master = n$
          $\land mastership' = [mastership \text{ EXCEPT } !.master = None]$
    $\land \text{UNCHANGED } \langle proposal, configuration, conn, target, history \rangle$
```

This section models configuration reconciliation.

```
$ReconcileConfiguration(n) \triangleq$
    $\land mastership.master = n$
    $\land \lor \land configuration.status \neq InProgress$
          $\land configuration.applied.term < mastership.term$
          $\land configuration' = [configuration \text{ EXCEPT } !.status = InProgress]$
          $\land \text{UNCHANGED } \langle target \rangle$
       $\lor \land configuration.status = InProgress$
          $\land configuration.applied.term < mastership.term$
          $\land conn[n].connected$
          $\land target.running$
          $\land target' = [target \text{ EXCEPT } !.values = configuration.applied.values]$
          $\land configuration' = [configuration \text{ EXCEPT } !.applied.term   = mastership.term,$
                                                      $!.applied.target = target.id,$
                                                      $!.status         = Complete]$
    $\land \text{UNCHANGED } \langle proposal, mastership, conn, history \rangle$
```

This section models proposal reconcilation.

```
$\text{LOCAL } ChangeValues(i) \triangleq$
    $[p \in \text{DOMAIN } proposal[i].values \mapsto$
      $[index \mapsto i, value \mapsto proposal[i].values[p]]]$

$\text{LOCAL } ChangeCommitted(i) \triangleq$
    $\land$   $proposal[i].change.commit = Complete$
    $\land$   $proposal[i].rollback.commit \neq Complete$

$\text{LOCAL } ChangeApplied(i) \triangleq$
    $\land$   $proposal[i].change.apply = Complete$
    $\land$   $proposal[i].rollback.apply \neq Complete$

$\text{LOCAL } RollbackValues(i, Filter(\_)) \triangleq$
    $\text{LET}$
        $changes \triangleq \{j \in \text{DOMAIN } proposal :$
                           $\land j < i$
                           $\land Filter(j)\}$
        $paths   \triangleq \{p \in \text{DOMAIN } configuration.committed.values :$
                           $\exists j \in changes : p \in \text{DOMAIN } proposal[j].values\}$
        $indexes \triangleq [p \in paths \mapsto \text{CHOOSE } j \in changes :$
```

```
                        $\land p \in \text{DOMAIN } proposal[j].values$
                        $\land \neg \exists k \in changes : k > j \land p \in \text{DOMAIN } proposal[k].values]$
  $\text{IN}$
    $[p \in \text{DOMAIN } proposal[i].values \mapsto$
      $\text{IF } p \in paths \text{ THEN}$
        $[index \mapsto indexes[p], value \mapsto proposal[indexes[p]].values[p]]$
      $\text{ELSE}$
        $[index \mapsto 0, value \mapsto None]]$

$CommitChange(n, i) \triangleq$
  $\land \lor \land proposal[i].change.commit = Pending$
        $\land \forall j \in \text{DOMAIN } proposal : j < i \Rightarrow$
            $\land proposal[j].change.commit \in Done$
            $\land proposal[j].rollback.commit \neq InProgress$
        $\land \lor \land proposal[i].rollback.commit = None$
              $\land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = InProgress]$
           $\lor \land proposal[i].rollback.commit = Pending$
              $\land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = Aborted]$
        $\land \text{UNCHANGED } \langle configuration, history \rangle$
     $\lor \land proposal[i].change.commit = InProgress$
        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.
        $\land \lor \land configuration' = [configuration \text{ EXCEPT } !.committed.values = ChangeValues(i) @@$
                                                      $configuration.committed.values]$
              $\land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = Complete]$
              $\land history' = Append(history, [type \mapsto Change, phase \mapsto Commit, index \mapsto i])$
           $\lor \land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = Failed]$
              $\land \text{UNCHANGED } \langle configuration, history \rangle$
  $\land \text{UNCHANGED } \langle mastership, conn, target \rangle$

$ApplyChange(n, i) \triangleq$
  $\land \lor \land proposal[i].change.apply = Pending$
        $\land \lor \land proposal[i].change.commit = Complete$
              $\land \forall j \in \text{DOMAIN } proposal : j < i \Rightarrow$
                  $\lor \land proposal[j].change.apply = Complete$
                     $\land proposal[j].rollback.apply \neq InProgress$
                  $\lor \land proposal[j].change.apply = Failed$
                     $\land proposal[j].rollback.apply = Complete$
              $\land i - 1 \in \text{DOMAIN } proposal \land proposal[i - 1].change.apply = Failed \Rightarrow$
                    $proposal[i - 1].rollback.apply = Complete$
              $\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = InProgress]$
           $\lor \land proposal[i].change.commit \in \{Aborted, Failed\}$
```

```
              ∧ proposal′ = [proposal EXCEPT ![i].change.apply = Aborted]
           ∧ UNCHANGED ⟨configuration, target, history⟩
        ∨ ∧ proposal[i].change.apply = InProgress
            Verify the applied term is the current mastership term to ensure the
            configuration has been synchronized following restarts.
           ∧ configuration.applied.term = mastership.term
            Verify the node's connection to the target.
           ∧ conn[n].connected
           ∧ mastership.conn = conn[n].id
           ∧ target.running
            Model successful and failed target update requests.
           ∧ ∨ ∧ LET values ≜ ChangeValues(i)
                    IN    ∧ target′ = [target EXCEPT !.values = values @@ target.values]
                          ∧ configuration′ = [configuration EXCEPT !.applied.values = values @@
                                                                  configuration.applied.values]
                          ∧ proposal′ = [proposal EXCEPT ![i].change.apply = Complete]
                          ∧ history′ = Append(history, [type ↦ Change, phase ↦ Apply, index ↦ i])
              ∨ ∧ proposal′ = [proposal EXCEPT ![i].change.apply = Failed]
                 ∧ UNCHANGED ⟨configuration, target, history⟩
     ∧ UNCHANGED ⟨mastership, conn⟩

CommitRollback(n, i) ≜
     ∧ ∨ ∧ proposal[i].rollback.commit = Pending
           ∧ ∀ j ∈ DOMAIN proposal :
                 ∧ j > i
                 ∧ proposal[j].phase ≠ None
                 ∧ proposal[j].change.commit ≠ Pending
                 ⇒ proposal[j].rollback.commit = Complete
           ∧ ∨ ∧ proposal[i].change.commit = Aborted
                 ∧ proposal′ = [proposal EXCEPT ![i].rollback.commit = Complete]
              ∨ ∧ proposal[i].change.commit ∈ {Complete, Failed}
                 ∧ proposal′ = [proposal EXCEPT ![i].rollback.commit = InProgress]
           ∧ UNCHANGED ⟨configuration, history⟩
        ∨ ∧ proposal[i].rollback.commit = InProgress
           ∧ configuration′ = [configuration EXCEPT !.committed.values = RollbackValues(i, ChangeCommitted)
                                                           configuration.committed.values]
           ∧ proposal′ = [proposal EXCEPT ![i].rollback.commit = Complete]
           ∧ history′ = Append(history, [type ↦ Rollback, phase ↦ Commit, index ↦ i])
     ∧ UNCHANGED ⟨mastership, conn, target⟩

ApplyRollback(n, i) ≜
     ∧ ∨ ∧ proposal[i].rollback.apply = Pending
           ∧ proposal[i].rollback.commit = Complete
           ∧ ∀ j ∈ DOMAIN proposal :
                 ∧ j > i
              ∧ proposal[j].phase ≠ None
              ∧ proposal[j].change.apply ≠ Pending
              ⇒ proposal[j].rollback.apply ∈ Done
          ∧ ∨ ∧ proposal[i].change.apply = Pending
              ∧ proposal′ = [proposal EXCEPT ![i].change.apply   = Aborted,
                                             ![i].rollback.apply = Complete]
            ∨ ∧ proposal[i].change.apply ∈ Done
              ∧ proposal′ = [proposal EXCEPT ![i].rollback.apply = InProgress]
          ∧ UNCHANGED ⟨configuration, target, history⟩
       ∨ ∧ proposal[i].rollback.apply = InProgress
           Verify the applied term is the current mastership term to ensure the
           configuration has been synchronized following restarts.
         ∧ configuration.applied.term = mastership.term
           Verify the node's connection to the target.
         ∧ conn[n].connected
         ∧ target.running
         ∧ LET values ≜ RollbackValues(i, ChangeApplied)
             IN   ∧ target′ = [target EXCEPT !.values = values @@ target.values]
                  ∧ configuration′ = [configuration EXCEPT !.applied.values = values @@
                                                              configuration.applied.values]
                  ∧ proposal′ = [proposal EXCEPT ![i].rollback.apply = Complete]
                  ∧ history′ = Append(history, [type ↦ Rollback, phase ↦ Apply, index ↦ i])
    ∧ UNCHANGED ⟨mastership, conn⟩

ReconcileProposal(n, i) ≜
    ∧ mastership.master = n
    ∧ ∨ CommitChange(n, i)
      ∨ ApplyChange(n, i)
      ∨ CommitRollback(n, i)
      ∨ ApplyRollback(n, i)
    ∧ UNCHANGED ⟨mastership, conn⟩
```

This section models changes to the proposal queue.

```
 Propose change at index 'i'
ProposeChange(i) ≜
    ∧ proposal[i].phase = None
    ∧ i − 1 ∈ DOMAIN proposal ⇒ proposal[i − 1].phase ≠ None
    ∧ ∃ p ∈ Path, v ∈ AllValues :
         ∧ 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)
    ∨ ∃ n ∈ Node : ReconcileMastership(n)
    ∨ ∃ n ∈ Node :
        $\lor\ \mathit{ConnectNode}(n)$
        $\lor\ \mathit{DisconnectNode}(n)$
    $\lor\ \mathit{StartTarget}$
    $\lor\ \mathit{StopTarget}$

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

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

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

$\mathit{Order}\ \triangleq$
    $\land\ \ \forall\, i \in \textsc{domain}\ \mathit{history} :$
        $\lor\ \mathit{IsOrderedChange}(\mathit{Commit},\ i)$
        $\lor\ \mathit{IsOrderedChange}(\mathit{Apply},\ i)$
        $\lor\ \mathit{IsOrderedRollback}(\mathit{Commit},\ i)$
        $\lor\ \mathit{IsOrderedRollback}(\mathit{Apply},\ i)$
```

```
  $\land\ \forall\, i \in \textsc{domain}\ proposal :$
        $\land\ proposal[i].change.apply = Failed$
        $\land\ proposal[i].rollback.apply \neq Complete$
        $\Rightarrow \forall\, j \in \textsc{domain}\ proposal : j > i \Rightarrow$
            $proposal[j].change.apply \in \{None,\ Pending,\ Aborted\}$

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

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

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

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

$Liveness \triangleq Termination$

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