MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

GenerateTestCases $\triangleq$ TRUE

Nil $\triangleq$ "<nil>"

Change $\triangleq$ "Change"
Rollback $\triangleq$ "Rollback"

Commit $\triangleq$ "Commit"
Apply $\triangleq$ "Apply"

Pending $\triangleq$ "Pending"
InProgress $\triangleq$ "InProgress"
Complete $\triangleq$ "Complete"
Aborted $\triangleq$ "Aborted"
Canceled $\triangleq$ "Canceled"
Failed $\triangleq$ "Failed"

Node $\triangleq$ {"node1"}

NumTransactions $\triangleq$ 3
NumTerms $\triangleq$ 1
NumConns $\triangleq$ 1
NumStarts $\triangleq$ 1

Path $\triangleq$ {"path1"}
Value $\triangleq$ {"value1", "value2"}

A transaction log.
VARIABLE *transactions*

A record of per-target configurations
VARIABLE *configuration*

A record of target masterships
VARIABLE *mastership*

A record of node connections to the target

VARIABLE *conns*

The target state
VARIABLE *target*

A sequence of state changes used for model checking.
VARIABLE *history*

$vars \triangleq \langle transactions,\ configuration,\ mastership,\ conns,\ target,\ history \rangle$

LOCAL *Transaction* $\triangleq$ INSTANCE *Transaction*

LOCAL *Configuration* $\triangleq$ INSTANCE *Configuration*

LOCAL *Mastership* $\triangleq$ INSTANCE *Mastership*

LOCAL *Target* $\triangleq$ INSTANCE *Target*

$AppendChange(i) \triangleq$
$\quad \land\ Transaction!AppendChange(i)$

$RollbackChange(i) \triangleq$
$\quad \land\ Transaction!RollbackChange(i)$

$ReconcileTransaction(n,\ i) \triangleq$
$\quad \land\ i \in \text{DOMAIN } transactions$
$\quad \land\ \lor\ \land\ Transaction!ReconcileTransaction(n,\ i)$
$\qquad\qquad \land\ GenerateTestCases \Rightarrow Transaction!Test!Log([node \mapsto n,\ index \mapsto i])$
$\qquad\quad \lor\ \land\ GenerateTestCases$
$\qquad\qquad \land\ \neg\text{ENABLED } Transaction!ReconcileTransaction(n,\ i)$
$\qquad\qquad \land\ \text{UNCHANGED } vars$
$\qquad\qquad \land\ Transaction!Test!Log([node \mapsto n,\ index \mapsto i])$

$ReconcileConfiguration(n) \triangleq$
$\quad \lor\ \land\ Configuration!ReconcileConfiguration(n)$
$\qquad\quad \land\ \text{UNCHANGED } \langle transactions,\ history \rangle$
$\qquad\quad \land\ GenerateTestCases \Rightarrow Configuration!Test!Log([node \mapsto n])$
$\quad \lor\ \land\ GenerateTestCases$
$\qquad\quad \land\ \neg\text{ENABLED } Configuration!ReconcileConfiguration(n)$
$\qquad\quad \land\ \text{UNCHANGED } vars$
$\qquad\quad \land\ Configuration!Test!Log([node \mapsto n])$

$ReconcileMastership(n) \triangleq$
$\quad \lor\ \land\ Mastership!ReconcileMastership(n)$
$\qquad\quad \land\ \text{UNCHANGED } \langle transactions,\ configuration,\ target,\ history \rangle$

$\quad \land \mathit{GenerateTestCases} \Rightarrow \mathit{Mastership}!\mathit{Test}!\mathit{Log}([\mathit{node} \mapsto n])$
$\lor \land \mathit{GenerateTestCases}$
$\quad \land \neg \text{ENABLED } \mathit{Mastership}!\mathit{ReconcileMastership}(n)$
$\quad \land \text{UNCHANGED } \mathit{vars}$
$\quad \land \mathit{Mastership}!\mathit{Test}!\mathit{Log}([\mathit{node} \mapsto n])$

$\mathit{ConnectNode}(n) \triangleq$
$\land \mathit{Target}!\mathit{Connect}(n)$
$\land \text{UNCHANGED } \langle \mathit{transactions}, \mathit{configuration}, \mathit{mastership}, \mathit{history} \rangle$

$\mathit{DisconnectNode}(n) \triangleq$
$\land \mathit{Target}!\mathit{Disconnect}(n)$
$\land \text{UNCHANGED } \langle \mathit{transactions}, \mathit{configuration}, \mathit{mastership}, \mathit{history} \rangle$

$\mathit{StartTarget} \triangleq$
$\land \mathit{Target}!\mathit{Start}$
$\land \text{UNCHANGED } \langle \mathit{transactions}, \mathit{configuration}, \mathit{mastership}, \mathit{history} \rangle$

$\mathit{StopTarget} \triangleq$
$\land \mathit{Target}!\mathit{Stop}$
$\land \text{UNCHANGED } \langle \mathit{transactions}, \mathit{configuration}, \mathit{mastership}, \mathit{history} \rangle$

Formal specification, constraints, and theorems.

$\mathit{Init} \triangleq$
$\land \mathit{transactions} = [$
$\quad i \in \{\} \mapsto [$
$\quad\quad \mathit{phase} \quad \mapsto \mathit{Nil},$
$\quad\quad \mathit{values} \mapsto [$
$\quad\quad\quad p \in \{\} \mapsto \mathit{Nil}],$
$\quad\quad \mathit{change} \quad \mapsto [$
$\quad\quad\quad \mathit{commit} \mapsto \mathit{Nil},$
$\quad\quad\quad \mathit{apply} \quad \mapsto \mathit{Nil}],$
$\quad\quad \mathit{rollback} \mapsto [$
$\quad\quad\quad \mathit{commit} \mapsto \mathit{Nil},$
$\quad\quad\quad \mathit{apply} \quad \mapsto \mathit{Nil}]]]$
$\land \mathit{configuration} = [$
$\quad \mathit{state} \;\mapsto \mathit{Pending},$
$\quad \mathit{term} \;\mapsto 0,$
$\quad \mathit{committed} \mapsto [$
$\quad\quad \mathit{index} \quad \mapsto 0,$
$\quad\quad \mathit{change} \quad \mapsto 0,$
$\quad\quad \mathit{target} \quad \mapsto 0,$
$\quad\quad \mathit{ordinal} \;\mapsto 0,$
$\quad\quad \mathit{revision} \mapsto 0,$
$\quad\quad \mathit{values} \quad \mapsto [$

```
                p ∈ {} ↦ Nil]],
        applied ↦ [
            index     ↦ 0,
            target    ↦ 0,
            ordinal   ↦ 0,
            revision  ↦ 0,
            values    ↦ [
                p ∈ {} ↦ Nil]]]
    ∧ target = [
        id        ↦ 1,
        running ↦ TRUE,
        values    ↦ [
            p ∈ {}  ↦ [
                index ↦ 0,
                value ↦ Nil]]]
    ∧ mastership = [
        master ↦ CHOOSE n ∈ Node : TRUE,
        term    ↦ 1,
        conn    ↦ 1]
    ∧ conns = [
        n ∈ Node ↦ [
            id           ↦ 1,
            connected ↦ TRUE]]
    ∧ history = ⟨⟩

Next ≜
    ∨ ∃ i ∈ 1 .. NumTransactions :
        ∨ AppendChange(i)
        ∨ RollbackChange(i)
    ∨ ∃ n ∈ Node, i ∈ 1 .. NumTransactions :
        ReconcileTransaction(n, i)
    ∨ ∃ n ∈ Node :
        ReconcileConfiguration(n)
    ∨ ∃ n ∈ Node :
        ReconcileMastership(n)
    ∨ ∃ n ∈ Node :
        ∨ ConnectNode(n)
        ∨ DisconnectNode(n)
    ∨ StartTarget
    ∨ StopTarget

Spec ≜
    ∧ Init
    ∧ □[Next]_vars
    ∧ ∀ i ∈ 1 .. NumTransactions :
```

$$
\begin{array}{l}
\quad \mathrm{WF}_{\langle transactions \rangle}(Transaction!RollbackChange(i)) \\
\land \forall\, n \in Node,\, i \in 1\,..\,NumTransactions : \\
\quad \mathrm{WF}_{\langle transactions,\, configuration,\, mastership,\, conns,\, target,\, history \rangle}(Transaction!ReconcileTransaction(n,\, i)) \\
\land \forall\, n \in Node : \\
\quad \mathrm{WF}_{\langle configuration,\, mastership,\, conns,\, target \rangle}(Configuration!ReconcileConfiguration(n)) \\
\land \forall\, n \in Node : \\
\quad \mathrm{WF}_{\langle mastership,\, conns \rangle}(Mastership!ReconcileMastership(n)) \\
\land \forall\, n \in Node : \\
\quad \mathrm{WF}_{\langle conns,\, target \rangle}(Target!Connect(n) \lor Target!Disconnect(n)) \\
\land \mathrm{WF}_{\langle conns,\, target \rangle}(Target!Start \lor Target!Stop)
\end{array}
$$

$$
\begin{array}{l}
LimitTerms \triangleq \\
\quad \lor\, mastership.term < NumTerms \\
\quad \lor\, \land\, mastership.term = NumTerms \\
\qquad \land\, mastership.master \neq Nil
\end{array}
$$

$$
\begin{array}{l}
LimitConns \triangleq \\
\quad \forall\, n \in \text{DOMAIN } conns : \\
\qquad \lor\, conns[n].id < NumConns \\
\qquad \lor\, \land\, conns[n].id = NumConns \\
\qquad\quad \land\, conns[n].connected
\end{array}
$$

$$
\begin{array}{l}
LimitStarts \triangleq \\
\quad \lor\, target.id < 2 \\
\quad \lor\, \land\, target.id = 2 \\
\qquad \land\, target.running
\end{array}
$$

$$
\begin{array}{l}
TypeOK \triangleq \\
\quad \land\, Transaction!TypeOK \\
\quad \land\, Configuration!TypeOK \\
\quad \land\, Mastership!TypeOK
\end{array}
$$

$$
\begin{array}{l}
StatusCommitted(i) \triangleq \\
\quad \land\, Len(history) = Len(history') \\
\quad \land\, \lor\, \land\, transactions'[i].change.commit \notin \{Pending,\, Canceled\} \\
\qquad\quad \land\, transactions[i].change.commit \neq transactions'[i].change.commit \\
\qquad \lor\, \land\, transactions'[i].rollback.commit \notin \{Pending,\, Canceled\} \\
\qquad\quad \land\, transactions[i].rollback.commit \neq transactions'[i].rollback.commit
\end{array}
$$

$$
\begin{array}{l}
StatusApplied(i) \triangleq \\
\quad \land\, Len(history) = Len(history') \\
\quad \land\, \lor\, \land\, transactions'[i].change.apply \notin \{Pending,\, Canceled,\, Aborted\} \\
\qquad\quad \land\, transactions[i].change.apply \neq transactions'[i].change.apply
\end{array}
$$

$$
\begin{array}{l}
\quad \lor \land transactions'[i].rollback.apply \notin \{Pending,\ Canceled,\ Aborted\} \\
\qquad \land transactions[i].rollback.apply \neq transactions'[i].rollback.apply
\end{array}
$$

$$
\begin{array}{l}
ValidStatus(t,\ i,\ j) \triangleq \\
\quad \land j \in \text{DOMAIN } history \\
\quad \land history[j].index = i \\
\quad \land \lor \land history[j].phase = Change \\
\qquad \land history[j].event = Commit \\
\qquad \land t[i].change.commit = history[j].status \\
\qquad \lor \land history[j].phase = Change \\
\qquad \land history[j].event = Apply \\
\qquad \land t[i].change.apply = history[j].status \\
\qquad \lor \land history[j].phase = Rollback \\
\qquad \land history[j].event = Commit \\
\qquad \land t[i].rollback.commit = history[j].status \\
\qquad \lor \land history[j].phase = Rollback \\
\qquad \land history[j].event = Apply \\
\qquad \land t[i].rollback.apply = history[j].status
\end{array}
$$

$$
\begin{array}{l}
ValidCommit(t,\ i) \triangleq \\
\quad \text{LET } j \triangleq \text{CHOOSE } j \in \text{DOMAIN } history : \\
\qquad \land history[j].event = Commit \\
\qquad \land \neg\exists\, k \in \text{DOMAIN } history : \\
\qquad\quad \land history[k].event = Commit \\
\qquad\quad \land k > j \\
\quad \text{IN } \quad ValidStatus(t,\ i,\ j)
\end{array}
$$

$$
\begin{array}{l}
ValidApply(t,\ i) \triangleq \\
\quad \text{LET } j \triangleq \text{CHOOSE } j \in \text{DOMAIN } history : \\
\qquad \land history[j].event = Apply \\
\qquad \land \neg\exists\, k \in \text{DOMAIN } history : \\
\qquad\quad \land history[k].event = Apply \\
\qquad\quad \land k > j \\
\quad \text{IN } \quad ValidStatus(t,\ i,\ j)
\end{array}
$$

$$
\begin{array}{l}
AtomicStatusChange \triangleq \\
\quad \forall\, i \in 1\,..\,NumTransactions : \\
\qquad \land i \in \text{DOMAIN } transactions \Rightarrow \\
\qquad\quad \land StatusCommitted(i) \Rightarrow ValidCommit(transactions',\ i) \\
\qquad\quad \land StatusApplied(i) \Rightarrow ValidApply(transactions',\ i)
\end{array}
$$

$$
Transition \triangleq \Box[AtomicStatusChange]_{\langle transactions,\, history\rangle}
$$

$$
\begin{array}{l}
\text{LOCAL } IsOrderedChange(p,\ i) \triangleq \\
\quad \land \quad history[i].phase = Change \\
\quad \land \quad history[i].event = p \\
\quad \land \quad history[i].status = Complete
\end{array}
$$

$\land \quad \neg\exists\, j \in \text{DOMAIN } history :$
$\quad \land\, j < i$
$\quad \land\, history[j].phase = Change$
$\quad \land\, history[j].event = p$
$\quad \land\, history[j].status = Complete$
$\quad \land\, history[j].index \geq history[i].index$

LOCAL $IsOrderedRollback(p,\, i) \triangleq$
$\land \quad history[i].phase = Rollback$
$\land \quad history[i].event = p$
$\land \quad history[i].status = Complete$
$\land \quad \exists\, j \in \text{DOMAIN } history :$
$\quad \land\, j < i$
$\quad \land\, history[j].phase = Change$
$\quad \land\, history[j].status = Complete$
$\quad \land\, history[j].index = history[i].index$
$\land \quad \neg\exists\, j \in \text{DOMAIN } history :$
$\quad \land\, j < i$
$\quad \land\, history[j].phase = Change$
$\quad \land\, history[j].event = p$
$\quad \land\, history[j].status = Complete$
$\quad \land\, history[j].index > history[i].index$
$\quad \land \neg\exists\, k \in \text{DOMAIN } history :$
$\quad\quad \land\, k > j$
$\quad\quad \land\, k < i$
$\quad\quad \land\, history[k].phase = Rollback$
$\quad\quad \land\, history[k].event = p$
$\quad\quad \land\, history[j].status = Complete$
$\quad\quad \land\, history[k].index = history[j].index$

$Order \triangleq$
$\land \quad \forall\, i \in \text{DOMAIN } history :$
$\quad history[i].status = Complete \Rightarrow$
$\quad\quad \lor\, IsOrderedChange(Commit,\, i)$
$\quad\quad \lor\, IsOrderedChange(Apply,\, i)$
$\quad\quad \lor\, IsOrderedRollback(Commit,\, i)$
$\quad\quad \lor\, IsOrderedRollback(Apply,\, i)$
$\land \quad \forall\, i \in \text{DOMAIN } transactions :$
$\quad \land\, transactions[i].change.apply = Failed$
$\quad \land\, transactions[i].rollback.apply \neq Complete$
$\quad \Rightarrow \neg\exists\, j \in \text{DOMAIN } transactions :$
$\quad\quad \land\, j > i$
$\quad\quad \land\, transactions[i].change.apply \in \{InProgress,\, Complete\}$

LOCAL $IsChangeCommitted(i) \triangleq$
$\land \quad configuration.committed.revision = i$

```
LOCAL IsChangeApplied(i) ≜
    ∧   configuration.applied.revision = i

Consistency ≜
    ∧ ∀ i ∈ DOMAIN transactions :
        ∧ IsChangeCommitted(i)
        ∧ ¬∃ j ∈ DOMAIN transactions :
              ∧ j > i
              ∧ IsChangeCommitted(j)
        ⇒ ∀ p ∈ DOMAIN transactions[i].change.values :
              ∧ configuration.committed.values[p] = transactions[i].change.values[p]
    ∧ ∀ i ∈ DOMAIN transactions :
        ∧ IsChangeApplied(i)
        ∧ ¬∃ j ∈ DOMAIN transactions :
              ∧ j > i
              ∧ IsChangeApplied(j)
        ⇒ ∀ p ∈ DOMAIN transactions[i].change.values :
              ∧ configuration.applied.values[p] = transactions[i].change.values[p]
              ∧ ∧ target.running
                ∧ configuration.applied.target = target.id
                ∧ configuration.state = Complete
                ⇒ target.values[p] = transactions[i].change.values[p]

Safety ≜ □(Order ∧ Consistency)

THEOREM Spec ⇒ Safety

LOCAL IsChanging(i) ≜
    ∧   i ∈ DOMAIN transactions
    ∧   transactions[i].phase = Change

LOCAL IsChanged(i) ≜
    ∧   i ∈ DOMAIN transactions
    ∧   transactions[i].change.commit ∈ {Complete, Failed}
    ∧   transactions[i].change.apply ∈ {Complete, Aborted, Failed}

LOCAL IsRollingBack(i) ≜
    ∧   i ∈ DOMAIN transactions
    ∧   transactions[i].phase = Rollback

LOCAL IsRolledBack(i) ≜
    ∧   i ∈ DOMAIN transactions
    ∧   transactions[i].rollback.commit ∈ {Complete, Failed}
    ∧   transactions[i].rollback.apply ∈ {Complete, Aborted, Failed}

Terminates(i) ≜
    ∧ IsChanging(i) ⇝ IsChanged(i)
```

$\land\ IsRollingBack(i) \leadsto IsRolledBack(i)$

$Termination \triangleq$
$\quad \forall\, i \in 1 \,..\, NumTransactions : Terminates(i)$

$Liveness \triangleq Termination$

THEOREM $Spec \Rightarrow Liveness$