MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

GenerateTestCases $\triangleq$ FALSE

Nil $\triangleq$ "<nil>"

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

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

Pending $\triangleq$ "Pending"
Complete $\triangleq$ "Complete"
Aborted $\triangleq$ "Aborted"
Failed $\triangleq$ "Failed"
Done $\triangleq$ {*Complete*, *Aborted*, *Failed*}

Node $\triangleq$ {"node1"}

NumTransactions $\triangleq$ 3
NumTerms $\triangleq$ 2
NumConns $\triangleq$ 2
NumStarts $\triangleq$ 2

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

A transaction *log*. Transactions may either request a set of changes to a set of targets or rollback a prior change.
VARIABLE *transaction*

A record of per-target proposals
VARIABLE *proposal*

A record of per-target configurations
VARIABLE *configuration*

A record of target masterships

VARIABLE *mastership*

A record of node connections to the target
VARIABLE *conn*

The target state
VARIABLE *target*

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

$vars \triangleq \langle transaction, proposal, configuration, mastership, conn, target, history \rangle$

LOCAL $Transaction \triangleq$ INSTANCE *Transaction*

LOCAL $Configuration \triangleq$ INSTANCE *Configuration*

LOCAL $Mastership \triangleq$ INSTANCE *Mastership*

LOCAL $Target \triangleq$ INSTANCE *Target*

$AppendChange(p, v) \triangleq$
$\quad \land Transaction!AppendChange(p, v)$

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

$ReconcileTransaction(n, i) \triangleq$
$\quad \land i \in$ DOMAIN *transaction*
$\quad \land Transaction!ReconcileTransaction(n, i)$
$\quad \land GenerateTestCases \Rightarrow$
$\qquad$ LET $context \triangleq [node \mapsto n, index \mapsto i]$
$\qquad$ IN $Transaction!Test!Log(context)$

$ReconcileConfiguration(n) \triangleq$
$\quad \land Configuration!ReconcileConfiguration(n)$
$\quad \land$ UNCHANGED $\langle transaction, proposal, history \rangle$
$\quad \land GenerateTestCases \Rightarrow Configuration!Test!Log([node \mapsto n])$

$ReconcileMastership(n) \triangleq$
$\quad \land Mastership!ReconcileMastership(n)$
$\quad \land$ UNCHANGED $\langle transaction, proposal, configuration, target, history \rangle$
$\quad \land GenerateTestCases \Rightarrow Mastership!Test!Log([node \mapsto n])$

$ConnectNode(n) \triangleq$
$\quad \land Target!Connect(n)$

$\land$ UNCHANGED $\langle transaction, proposal, configuration, mastership, history\rangle$

$DisconnectNode(n) \triangleq$
$\land$ $Target!Disconnect(n)$
$\land$ UNCHANGED $\langle transaction, proposal, configuration, mastership, history\rangle$

$StartTarget \triangleq$
$\land$ $Target!Start$
$\land$ UNCHANGED $\langle transaction, proposal, configuration, mastership, history\rangle$

$StopTarget \triangleq$
$\land$ $Target!Stop$
$\land$ UNCHANGED $\langle transaction, proposal, configuration, mastership, history\rangle$

Formal specification, constraints, and theorems.

$Init \triangleq$
$\land$ $transaction = [$
$i \in \{\} \mapsto [$
$phase \mapsto Nil,$
$change \mapsto [$
$proposal \mapsto 0,$
$revision \mapsto 0,$
$values \mapsto [$
$p \in \{\} \mapsto [$
$index \mapsto 0,$
$value \mapsto Nil]]],$
$rollback \mapsto [$
$proposal \mapsto 0,$
$revision \mapsto 0,$
$values \mapsto [$
$p \in \{\} \mapsto [$
$index \mapsto 0,$
$value \mapsto Nil]]]]]$
$\land$ $proposal = [$
$i \in \{\} \mapsto [$
$transaction \mapsto 0,$
$commit \mapsto Nil,$
$apply \mapsto Nil]]$
$\land$ $configuration = [$
$state \mapsto Pending,$
$term \mapsto 0,$
$committed \mapsto [$
$index \mapsto 0,$
$revision \mapsto 0,$

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

Next ≜
    ∨ ∃ p ∈ Path, v ∈ Value :
        AppendChange(p, v)
    ∨ ∃ i ∈ DOMAIN transaction :
        RollbackChange(i)
    ∨ ∃ n ∈ Node :
        ∃ i ∈ DOMAIN transaction :
          ReconcileTransaction(n, i)
    ∨ ∃ n ∈ Node :
        ReconcileConfiguration(n)
    ∨ ∃ n ∈ Node :
        ReconcileMastership(n)
    ∨ ∃ n ∈ Node :
        ∨ ConnectNode(n)
        ∨ DisconnectNode(n)
    ∨ StartTarget
```

$\lor\ StopTarget$

$Spec \triangleq$
$\land\ Init$
$\land\ \Box[Next]_{vars}$
$\land\ \forall\, p \in Path,\ v \in Value :$
$\quad \mathrm{WF}_{\langle transaction,\ proposal,\ configuration,\ mastership,\ conn,\ target,\ history\rangle}(Transaction!AppendChange(p,\ v))$
$\land\ \forall\, i \in 1\,..\,NumTransactions : i \in \text{DOMAIN } transaction \Rightarrow$
$\quad \mathrm{WF}_{\langle transaction,\ proposal,\ configuration,\ mastership,\ conn,\ target,\ history\rangle}(Transaction!RollbackChange(i))$
$\land\ \forall\, n \in Node,\ i \in 1\,..\,NumTransactions :$
$\quad \mathrm{WF}_{\langle transaction,\ proposal,\ configuration,\ mastership,\ conn,\ target,\ history\rangle}(Transaction!ReconcileTransaction(n,\ i))$
$\land\ \forall\, n \in Node :$
$\quad \mathrm{WF}_{\langle configuration,\ mastership,\ conn,\ target\rangle}(Configuration!ReconcileConfiguration(n))$
$\land\ \forall\, n \in Node :$
$\quad \mathrm{WF}_{\langle mastership,\ conn\rangle}(Mastership!ReconcileMastership(n))$
$\land\ \forall\, n \in Node :$
$\quad \mathrm{WF}_{\langle conn,\ target\rangle}(Target!Connect(n) \lor Target!Disconnect(n))$
$\land\ \mathrm{WF}_{\langle conn,\ target\rangle}(Target!Start \lor Target!Stop)$

$Alias \triangleq\ [$
$log \mapsto [$
$i \in \text{DOMAIN } transaction \mapsto [$
$change \mapsto$
IF $transaction[i].change.proposal \neq 0$ THEN
$[commit \mapsto proposal[transaction[i].change.proposal].commit,$
$apply \mapsto proposal[transaction[i].change.proposal].apply,$
$values \mapsto transaction[i].change.values]$
ELSE
$[commit \mapsto Nil,$
$apply \mapsto Nil,$
$values \mapsto transaction[i].change.values],$
$rollback \mapsto$
IF $transaction[i].rollback.proposal \neq 0$ THEN
$[commit \mapsto proposal[transaction[i].rollback.proposal].commit,$
$apply \mapsto proposal[transaction[i].rollback.proposal].apply,$
$values \mapsto transaction[i].rollback.values]$
ELSE
$[commit \mapsto Nil,$
$apply \mapsto Nil,$
$values \mapsto transaction[i].rollback.values]]$ @@
$transaction[i]]],$
$transaction \mapsto transaction,$
$proposal \mapsto proposal,$
$configuration \mapsto configuration,$
$mastership \mapsto mastership,$

$conn \mapsto conn,$
$target \mapsto target,$
$history \mapsto history]$

$LimitTransactions \triangleq Len(transaction) \leq NumTransactions$

$LimitTerms \triangleq$
$\lor\ mastership.term < NumTerms$
$\lor\ \land\ mastership.term = NumTerms$
$\land\ mastership.master \neq Nil$

$LimitConns \triangleq$
$\forall\, n \in \text{DOMAIN } conn :$
$\lor\ conn[n].id < NumConns$
$\lor\ \land\ conn[n].id = NumConns$
$\land\ conn[n].connected$

$LimitStarts \triangleq$
$\lor\ target.id < 2$
$\lor\ \land\ target.id = 2$
$\land\ target.running$

$TypeOK \triangleq$
$\land\ Transaction!TypeOK$
$\land\ Configuration!TypeOK$
$\land\ Mastership!TypeOK$

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

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

$\land\, j < i$
$\land\, history[j].type = Change$
$\land\, history[j].phase = p$
$\land\, history[j].index > history[i].index$
$\land \neg\exists\, k \in \text{DOMAIN } history :$
$\land\, k > j$
$\land\, k < i$
$\land\, history[k].type = Rollback$
$\land\, history[k].phase = p$
$\land\, history[k].index = history[j].index$

$Order \triangleq$
$\land\ \forall\, i \in \text{DOMAIN } history :$
$\lor\, IsOrderedChange(Commit,\ i)$
$\lor\, IsOrderedChange(Apply,\ i)$
$\lor\, IsOrderedRollback(Commit,\ i)$
$\lor\, IsOrderedRollback(Apply,\ i)$
$\land\ \forall\, i \in \text{DOMAIN } transaction :$
$\land\, transaction[i].change.proposal \neq 0$
$\land\, proposal[transaction[i].change.proposal].apply = Failed$
$\land\, transaction[i].rollback.proposal \neq 0 \Rightarrow$
$proposal[transaction[i].rollback.proposal].apply \neq Complete$
$\Rightarrow \forall\, j \in \text{DOMAIN } transaction : (j > i \Rightarrow$
$(transaction[j].change.proposal \neq 0 \Rightarrow$
$proposal[transaction[j].change.proposal].apply \neq Complete))$

LOCAL $IsChangeCommitted(i) \triangleq$
$\land\quad transaction[i].change.proposal \neq 0$
$\land\quad proposal[transaction[i].change.proposal].commit = Complete$
$\land\quad transaction[i].rollback.proposal \neq 0 \Rightarrow$
$proposal[transaction[i].rollback.proposal].commit \neq Complete$

LOCAL $IsChangeApplied(i) \triangleq$
$\land\quad transaction[i].change.proposal \neq 0$
$\land\quad proposal[transaction[i].change.proposal].apply = Complete$
$\land\quad transaction[i].rollback.proposal \neq 0 \Rightarrow$
$proposal[transaction[i].rollback.proposal].apply \neq Complete$

$Consistency \triangleq$
$\land\, \forall\, i \in \text{DOMAIN } transaction :$
$\land\, IsChangeCommitted(i)$
$\land \neg\exists\, j \in \text{DOMAIN } transaction :$
$\land\, j > i$
$\land\, IsChangeCommitted(j)$
$\Rightarrow \forall\, p \in \text{DOMAIN } transaction[i].change.values :$
$\land\, configuration.committed.values[p] = transaction[i].change.values[p]$

$\land \forall\, i \in \text{DOMAIN } transaction :$
$\quad \land\ IsChangeApplied(i)$
$\quad \land \neg\exists\, j \in \text{DOMAIN } transaction :$
$\qquad \land\ j > i$
$\qquad \land\ IsChangeApplied(j)$
$\quad \Rightarrow \forall\, p \in \text{DOMAIN } transaction[i].change.values :$
$\qquad \land\ configuration.applied.values[p] = transaction[i].change.values[p]$
$\qquad \land\ \land\ target.running$
$\qquad\quad \land\ configuration.applied.target = target.id$
$\qquad\quad \land\ configuration.state = Complete$
$\qquad\quad \Rightarrow\ target.values[p] = transaction[i].change.values[p]$

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

THEOREM $Spec \Rightarrow Safety$

$Terminates(i) \triangleq$
$\land\ i \in \text{DOMAIN } transaction$
$\land\ transaction[i].phase = Change \leadsto$
$\quad \land\ transaction[i].change.proposal \neq 0$
$\quad \land\ proposal[transaction[i].change.proposal].commit \in Done$
$\quad \land\ proposal[transaction[i].change.proposal].apply \in Done$
$\land\ transaction[i].phase = Rollback \leadsto$
$\quad \land\ transaction[i].rollback.proposal \neq 0$
$\quad \land\ proposal[transaction[i].rollback.proposal].commit \in Done$
$\quad \land\ proposal[transaction[i].rollback.proposal].apply \in Done$

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

$Liveness \triangleq Termination$

THEOREM $Spec \Rightarrow Liveness$