
MODULE *Config*

This module provides the model values and safety and liveness properties for the model of $\mu ONOS$ *Config* controllers.

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 masterhips

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*

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

$\mu ONOS$ *Config* coordinates configuration changes using a collection of four controllers, each of which is responsible for managing one type of data. The four controllers are specified in a separate module for each, and they're imported here for use in the model.

LOCAL *Transaction* \triangleq INSTANCE *Transaction*

LOCAL *Configuration* \triangleq INSTANCE *Configuration*

LOCAL *Mastership* \triangleq INSTANCE *Mastership*

LOCAL *Target* \triangleq INSTANCE *Target*

This section defines the state changes that can occur across all the $\mu ONOS$ *Config* controllers.

AppendChange(*i*) \triangleq
 \wedge *Transaction*! *AppendChange*(*i*)

RollbackChange(*i*) \triangleq
 \wedge *Transaction*! *RollbackChange*(*i*)

ReconcileTransaction(*n*, *i*) \triangleq
 \wedge *i* \in DOMAIN *transactions*
 \wedge \vee \wedge *Transaction*! *ReconcileTransaction*(*n*, *i*)
 \wedge *GenerateTestCases* \Rightarrow *Transaction*! *Test*! *Log*([*node* \mapsto *n*, *index* \mapsto *i*])
 \vee \wedge *GenerateTestCases*
 \wedge \neg ENABLED *Transaction*! *ReconcileTransaction*(*n*, *i*)
 \wedge UNCHANGED *vars*
 \wedge *Transaction*! *Test*! *Log*([*node* \mapsto *n*, *index* \mapsto *i*])

ReconcileConfiguration(*n*) \triangleq
 \vee \wedge *Configuration*! *ReconcileConfiguration*(*n*)
 \wedge UNCHANGED $\langle transactions, history \rangle$
 \wedge *GenerateTestCases* \Rightarrow *Configuration*! *Test*! *Log*([*node* \mapsto *n*])
 \vee \wedge *GenerateTestCases*
 \wedge \neg ENABLED *Configuration*! *ReconcileConfiguration*(*n*)

$$\begin{aligned}
& \wedge \text{UNCHANGED } vars \\
& \wedge \text{Configuration!Test!Log}([node \mapsto n]) \\
\text{ReconcileMastership}(n) & \triangleq \\
& \vee \wedge \text{Mastership!ReconcileMastership}(n) \\
& \wedge \text{UNCHANGED } \langle transactions, configuration, target, history \rangle \\
& \wedge \text{GenerateTestCases} \Rightarrow \text{Mastership!Test!Log}([node \mapsto n]) \\
& \vee \wedge \text{GenerateTestCases} \\
& \wedge \neg \text{ENABLED } \text{Mastership!ReconcileMastership}(n) \\
& \wedge \text{UNCHANGED } vars \\
& \wedge \text{Mastership!Test!Log}([node \mapsto n]) \\
\text{ConnectNode}(n) & \triangleq \\
& \wedge \text{Target!Connect}(n) \\
& \wedge \text{UNCHANGED } \langle transactions, configuration, mastership, history \rangle \\
\text{DisconnectNode}(n) & \triangleq \\
& \wedge \text{Target!Disconnect}(n) \\
& \wedge \text{UNCHANGED } \langle transactions, configuration, mastership, history \rangle \\
\text{StartTarget} & \triangleq \\
& \wedge \text{Target!Start} \\
& \wedge \text{UNCHANGED } \langle transactions, configuration, mastership, history \rangle \\
\text{StopTarget} & \triangleq \\
& \wedge \text{Target!Stop} \\
& \wedge \text{UNCHANGED } \langle transactions, configuration, mastership, history \rangle
\end{aligned}$$

Formal specification, constraints, and theorems.

$$\begin{aligned}
\text{Init} & \triangleq \\
& \wedge transactions = [\\
& \quad i \in \{\} \mapsto [\\
& \quad \quad phase \mapsto Nil, \\
& \quad \quad values \mapsto [\\
& \quad \quad \quad p \in \{\} \mapsto Nil], \\
& \quad \quad change \mapsto [\\
& \quad \quad \quad commit \mapsto Nil, \\
& \quad \quad \quad apply \mapsto Nil], \\
& \quad \quad rollback \mapsto [\\
& \quad \quad \quad commit \mapsto Nil, \\
& \quad \quad \quad apply \mapsto Nil]]] \\
& \wedge configuration = [\\
& \quad state \mapsto Pending, \\
& \quad term \mapsto 0, \\
& \quad committed \mapsto [
\end{aligned}$$

$$\begin{aligned}
& \text{index} \mapsto 0, \\
& \text{change} \mapsto 0, \\
& \text{target} \mapsto 0, \\
& \text{ordinal} \mapsto 0, \\
& \text{revision} \mapsto 0, \\
& \text{values} \mapsto [\\
& \quad p \in \{\} \mapsto Nil], \\
& \text{applied} \mapsto [\\
& \quad \text{index} \mapsto 0, \\
& \quad \text{target} \mapsto 0, \\
& \quad \text{ordinal} \mapsto 0, \\
& \quad \text{revision} \mapsto 0, \\
& \quad \text{values} \mapsto [\\
& \quad \quad p \in \{\} \mapsto Nil]]] \\
& \wedge \text{target} = [\\
& \quad \text{id} \mapsto 1, \\
& \quad \text{running} \mapsto \text{TRUE}, \\
& \quad \text{values} \mapsto [\\
& \quad \quad p \in \{\} \mapsto [\\
& \quad \quad \quad \text{index} \mapsto 0, \\
& \quad \quad \quad \text{value} \mapsto Nil]]] \\
& \wedge \text{mastership} = [\\
& \quad \text{master} \mapsto \text{CHOOSE } n \in \text{Node} : \text{TRUE}, \\
& \quad \text{term} \mapsto 1, \\
& \quad \text{conn} \mapsto 1] \\
& \wedge \text{conns} = [\\
& \quad n \in \text{Node} \mapsto [\\
& \quad \quad \text{id} \mapsto 1, \\
& \quad \quad \text{connected} \mapsto \text{TRUE}] \\
& \wedge \text{history} = \langle \rangle \\
& \text{Next} \triangleq \\
& \quad \vee \exists i \in 1 \dots \text{NumTransactions} : \\
& \quad \quad \vee \text{AppendChange}(i) \\
& \quad \quad \vee \text{RollbackChange}(i) \\
& \quad \vee \exists n \in \text{Node}, i \in 1 \dots \text{NumTransactions} : \\
& \quad \quad \text{ReconcileTransaction}(n, i) \\
& \quad \vee \exists n \in \text{Node} : \\
& \quad \quad \text{ReconcileConfiguration}(n) \\
& \quad \vee \exists n \in \text{Node} : \\
& \quad \quad \text{ReconcileMastership}(n) \\
& \quad \vee \exists n \in \text{Node} : \\
& \quad \quad \vee \text{ConnectNode}(n) \\
& \quad \quad \vee \text{DisconnectNode}(n) \\
& \quad \vee \text{StartTarget}
\end{aligned}$$

$$\begin{aligned}
& \vee \textit{StopTarget} \\
\textit{Spec} & \triangleq \\
& \wedge \textit{Init} \\
& \wedge \Box[\textit{Next}]_{\textit{vars}} \\
& \wedge \forall i \in 1 \dots \textit{NumTransactions} : \\
& \quad \text{WF}_{\langle \textit{transactions} \rangle}(\textit{Transaction!RollbackChange}(i)) \\
& \wedge \forall n \in \textit{Node}, i \in 1 \dots \textit{NumTransactions} : \\
& \quad \text{WF}_{\langle \textit{transactions}, \textit{configuration}, \textit{mastership}, \textit{conns}, \textit{target}, \textit{history} \rangle}(\textit{Transaction!ReconcileTransaction}(n, i)) \\
& \wedge \forall n \in \textit{Node} : \\
& \quad \text{WF}_{\langle \textit{configuration}, \textit{mastership}, \textit{conns}, \textit{target} \rangle}(\textit{Configuration!ReconcileConfiguration}(n)) \\
& \wedge \forall n \in \textit{Node} : \\
& \quad \text{WF}_{\langle \textit{mastership}, \textit{conns} \rangle}(\textit{Mastership!ReconcileMastership}(n)) \\
& \wedge \forall n \in \textit{Node} : \\
& \quad \text{WF}_{\langle \textit{conns}, \textit{target} \rangle}(\textit{Target!Connect}(n) \vee \textit{Target!Disconnect}(n)) \\
& \wedge \text{WF}_{\langle \textit{conns}, \textit{target} \rangle}(\textit{Target!Start} \vee \textit{Target!Stop})
\end{aligned}$$

This section contains state constraints used for model checking.

$$\begin{aligned}
\textit{LimitTerms} & \triangleq \\
& \vee \textit{mastership.term} < \textit{NumTerms} \\
& \vee \wedge \textit{mastership.term} = \textit{NumTerms} \\
& \wedge \textit{mastership.master} \neq \textit{Nil}
\end{aligned}$$

$$\begin{aligned}
\textit{LimitConns} & \triangleq \\
& \forall n \in \text{DOMAIN } \textit{conns} : \\
& \quad \vee \textit{conns}[n].\textit{id} < \textit{NumConns} \\
& \quad \vee \wedge \textit{conns}[n].\textit{id} = \textit{NumConns} \\
& \quad \wedge \textit{conns}[n].\textit{connected}
\end{aligned}$$

$$\begin{aligned}
\textit{LimitStarts} & \triangleq \\
& \vee \textit{target.id} < 2 \\
& \vee \wedge \textit{target.id} = 2 \\
& \wedge \textit{target.running}
\end{aligned}$$

$$\begin{aligned}
\textit{TypeOK} & \triangleq \\
& \wedge \textit{Transaction!TypeOK} \\
& \wedge \textit{Configuration!TypeOK} \\
& \wedge \textit{Mastership!TypeOK}
\end{aligned}$$

This section contains invariants, action properties, and liveness properties used the model to verify the spec preserves order and consistency guarantees, always eventually terminates, and is deadlock free.

$$\begin{aligned}
\textit{StatusCommitted}(i) & \triangleq \\
& \wedge \textit{Len}(\textit{history}) = \textit{Len}(\textit{history}')
\end{aligned}$$

$$\begin{aligned}
& \wedge \vee \wedge \text{transactions}'[i].\text{change.commit} \notin \{\text{Pending}, \text{Canceled}\} \\
& \wedge \text{transactions}[i].\text{change.commit} \neq \text{transactions}'[i].\text{change.commit} \\
& \vee \wedge \text{transactions}'[i].\text{rollback.commit} \notin \{\text{Pending}, \text{Canceled}\} \\
& \wedge \text{transactions}[i].\text{rollback.commit} \neq \text{transactions}'[i].\text{rollback.commit}
\end{aligned}$$

$$\begin{aligned}
\text{StatusApplied}(i) & \triangleq \\
& \wedge \text{Len}(\text{history}) = \text{Len}(\text{history}') \\
& \wedge \vee \wedge \text{transactions}'[i].\text{change.apply} \notin \{\text{Pending}, \text{Canceled}, \text{Aborted}\} \\
& \quad \wedge \text{transactions}[i].\text{change.apply} \neq \text{transactions}'[i].\text{change.apply} \\
& \vee \wedge \text{transactions}'[i].\text{rollback.apply} \notin \{\text{Pending}, \text{Canceled}, \text{Aborted}\} \\
& \quad \wedge \text{transactions}[i].\text{rollback.apply} \neq \text{transactions}'[i].\text{rollback.apply}
\end{aligned}$$

$$\begin{aligned}
\text{ValidStatus}(t, i, j) & \triangleq \\
& \wedge j \in \text{DOMAIN } \text{history} \\
& \wedge \text{history}[j].\text{index} = i \\
& \wedge \vee \wedge \text{history}[j].\text{phase} = \text{Change} \\
& \quad \wedge \text{history}[j].\text{event} = \text{Commit} \\
& \quad \wedge t[i].\text{change.commit} = \text{history}[j].\text{status} \\
& \vee \wedge \text{history}[j].\text{phase} = \text{Change} \\
& \quad \wedge \text{history}[j].\text{event} = \text{Apply} \\
& \quad \wedge t[i].\text{change.apply} = \text{history}[j].\text{status} \\
& \vee \wedge \text{history}[j].\text{phase} = \text{Rollback} \\
& \quad \wedge \text{history}[j].\text{event} = \text{Commit} \\
& \quad \wedge t[i].\text{rollback.commit} = \text{history}[j].\text{status} \\
& \vee \wedge \text{history}[j].\text{phase} = \text{Rollback} \\
& \quad \wedge \text{history}[j].\text{event} = \text{Apply} \\
& \quad \wedge t[i].\text{rollback.apply} = \text{history}[j].\text{status}
\end{aligned}$$

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

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

$$\begin{aligned}
\text{AtomicStatusChange} & \triangleq \\
& \forall i \in 1 \dots \text{NumTransactions} :
\end{aligned}$$

$$\begin{aligned}
& \wedge i \in \text{DOMAIN } \textit{transactions} \Rightarrow \\
& \quad \wedge \textit{StatusCommitted}(i) \Rightarrow \textit{ValidCommit}(\textit{transactions}', i) \\
& \quad \wedge \textit{StatusApplied}(i) \Rightarrow \textit{ValidApply}(\textit{transactions}', i) \\
\textit{Transition} & \triangleq \Box[\textit{AtomicStatusChange}]_{\langle \textit{transactions}, \textit{history} \rangle} \\
\text{LOCAL } \textit{IsOrderedChange}(p, i) & \triangleq \\
& \wedge \textit{history}[i].\textit{phase} = \textit{Change} \\
& \wedge \textit{history}[i].\textit{event} = p \\
& \wedge \textit{history}[i].\textit{status} = \textit{Complete} \\
& \wedge \neg \exists j \in \text{DOMAIN } \textit{history} : \\
& \quad \wedge j < i \\
& \quad \wedge \textit{history}[j].\textit{phase} = \textit{Change} \\
& \quad \wedge \textit{history}[j].\textit{event} = p \\
& \quad \wedge \textit{history}[j].\textit{status} = \textit{Complete} \\
& \quad \wedge \textit{history}[j].\textit{index} \geq \textit{history}[i].\textit{index} \\
\text{LOCAL } \textit{IsOrderedRollback}(p, i) & \triangleq \\
& \wedge \textit{history}[i].\textit{phase} = \textit{Rollback} \\
& \wedge \textit{history}[i].\textit{event} = p \\
& \wedge \textit{history}[i].\textit{status} = \textit{Complete} \\
& \wedge \exists j \in \text{DOMAIN } \textit{history} : \\
& \quad \wedge j < i \\
& \quad \wedge \textit{history}[j].\textit{phase} = \textit{Change} \\
& \quad \wedge \textit{history}[j].\textit{status} = \textit{Complete} \\
& \quad \wedge \textit{history}[j].\textit{index} = \textit{history}[i].\textit{index} \\
& \wedge \neg \exists j \in \text{DOMAIN } \textit{history} : \\
& \quad \wedge j < i \\
& \quad \wedge \textit{history}[j].\textit{phase} = \textit{Change} \\
& \quad \wedge \textit{history}[j].\textit{event} = p \\
& \quad \wedge \textit{history}[j].\textit{status} = \textit{Complete} \\
& \quad \wedge \textit{history}[j].\textit{index} > \textit{history}[i].\textit{index} \\
& \wedge \neg \exists k \in \text{DOMAIN } \textit{history} : \\
& \quad \wedge k > j \\
& \quad \wedge k < i \\
& \quad \wedge \textit{history}[k].\textit{phase} = \textit{Rollback} \\
& \quad \wedge \textit{history}[k].\textit{event} = p \\
& \quad \wedge \textit{history}[j].\textit{status} = \textit{Complete} \\
& \quad \wedge \textit{history}[k].\textit{index} = \textit{history}[j].\textit{index}
\end{aligned}$$

The *Order* invariant checks the recorded 'history' to ensure that changes and rollbacks are being committed and applied in consistent sequential order. Additionally, it enforces the invariant that if a rollback fails the apply phase, it and all subsequent pending transactions must be aborted and rolled back before a new transaction can be applied.

$$\begin{aligned}
\textit{Order} & \triangleq \\
& \wedge \forall i \in \text{DOMAIN } \textit{history} :
\end{aligned}$$

$$\begin{aligned}
& \text{history}[i].\text{status} = \text{Complete} \Rightarrow \\
& \quad \vee \text{IsOrderedChange}(\text{Commit}, i) \\
& \quad \vee \text{IsOrderedChange}(\text{Apply}, i) \\
& \quad \vee \text{IsOrderedRollback}(\text{Commit}, i) \\
& \quad \vee \text{IsOrderedRollback}(\text{Apply}, i) \\
& \wedge \forall i \in \text{DOMAIN } \text{transactions} : \\
& \quad \wedge \text{transactions}[i].\text{change.apply} = \text{Failed} \\
& \quad \wedge \text{transactions}[i].\text{rollback.apply} \neq \text{Complete} \\
& \Rightarrow \neg \exists j \in \text{DOMAIN } \text{transactions} : \\
& \quad \wedge j > i \\
& \quad \wedge \text{transactions}[i].\text{change.apply} \in \{\text{InProgress}, \text{Complete}\} \\
\text{LOCAL } \text{IsChangeCommitted}(i) & \triangleq \\
& \quad \wedge \text{configuration.committed.revision} = i \\
\text{LOCAL } \text{IsChangeApplied}(i) & \triangleq \\
& \quad \wedge \text{configuration.applied.revision} = i \\
\text{The } \textit{Consistency} \text{ invariant verifies that the changes and rollbacks are properly} \\
\text{propagated to the configuration when committed and to targets once applied.} \\
\text{Additionally, it checks that target configuration is restored following restarts.} \\
\textit{Consistency} & \triangleq \\
& \wedge \forall i \in \text{DOMAIN } \text{transactions} : \\
& \quad \wedge \text{IsChangeCommitted}(i) \\
& \quad \wedge \neg \exists j \in \text{DOMAIN } \text{transactions} : \\
& \quad \quad \wedge j > i \\
& \quad \quad \wedge \text{IsChangeCommitted}(j) \\
& \Rightarrow \forall p \in \text{DOMAIN } \text{transactions}[i].\text{change.values} : \\
& \quad \wedge \text{configuration.committed.values}[p] = \text{transactions}[i].\text{change.values}[p] \\
& \wedge \forall i \in \text{DOMAIN } \text{transactions} : \\
& \quad \wedge \text{IsChangeApplied}(i) \\
& \quad \wedge \neg \exists j \in \text{DOMAIN } \text{transactions} : \\
& \quad \quad \wedge j > i \\
& \quad \quad \wedge \text{IsChangeApplied}(j) \\
& \Rightarrow \forall p \in \text{DOMAIN } \text{transactions}[i].\text{change.values} : \\
& \quad \wedge \text{configuration.applied.values}[p] = \text{transactions}[i].\text{change.values}[p] \\
& \quad \wedge \wedge \text{target.running} \\
& \quad \quad \wedge \text{configuration.applied.target} = \text{target.id} \\
& \quad \quad \wedge \text{configuration.state} = \text{Complete} \\
& \Rightarrow \text{target.values}[p] = \text{transactions}[i].\text{change.values}[p]
\end{aligned}$$

The *Safety* property holds with both *Order* and *Consistency*.
 $\text{Safety} \triangleq \Box(\text{Order} \wedge \text{Consistency})$

THEOREM $\text{Spec} \Rightarrow \text{Safety}$

LOCAL $\text{IsChanging}(i) \triangleq$

$\wedge \quad i \in \text{DOMAIN } \textit{transactions}$
 $\wedge \quad \textit{transactions}[i].\textit{phase} = \textit{Change}$

LOCAL $\textit{IsChanged}(i) \triangleq$
 $\wedge \quad i \in \text{DOMAIN } \textit{transactions}$
 $\wedge \quad \textit{transactions}[i].\textit{change.commit} \in \{\textit{Complete}, \textit{Failed}\}$
 $\wedge \quad \textit{transactions}[i].\textit{change.apply} \in \{\textit{Complete}, \textit{Aborted}, \textit{Failed}\}$

LOCAL $\textit{IsRollingBack}(i) \triangleq$
 $\wedge \quad i \in \text{DOMAIN } \textit{transactions}$
 $\wedge \quad \textit{transactions}[i].\textit{phase} = \textit{Rollback}$

LOCAL $\textit{IsRolledBack}(i) \triangleq$
 $\wedge \quad i \in \text{DOMAIN } \textit{transactions}$
 $\wedge \quad \textit{transactions}[i].\textit{rollback.commit} \in \{\textit{Complete}, \textit{Failed}\}$
 $\wedge \quad \textit{transactions}[i].\textit{rollback.apply} \in \{\textit{Complete}, \textit{Aborted}, \textit{Failed}\}$

$\textit{Terminates}(i) \triangleq$
 $\wedge \quad \textit{IsChanging}(i) \rightsquigarrow \textit{IsChanged}(i)$
 $\wedge \quad \textit{IsRollingBack}(i) \rightsquigarrow \textit{IsRolledBack}(i)$

The *Termination* property is a liveness check to verify no deadlocks or livelocks exist in the system. So long as the system can make progress, every change and every rollback is guaranteed to have a path to complete both phases at all times.

$\textit{Termination} \triangleq$
 $\forall i \in 1 \dots \textit{NumTransactions} : \textit{Terminates}(i)$

The *Liveness* property holds when *Termination* holds.

$\textit{Liveness} \triangleq \textit{Termination}$

THEOREM $\textit{Spec} \Rightarrow \textit{Liveness}$