

---

MODULE *Config*

---

EXTENDS *Naturals, FiniteSets, Sequences, TLC*

Indicates that a configuration change is waiting to be applied to the network

CONSTANT *Pending*

ASSUME *Pending* ∈ STRING

Indicates that a configuration change has been applied to the network

CONSTANT *Complete*

ASSUME *Complete* ∈ STRING

Indicates that a configuration change failed

CONSTANT *Failed*

ASSUME *Failed* ∈ STRING

Indicates a change is a configuration

CONSTANT *Change*

ASSUME *Change* ∈ STRING

Indicates a change is a rollback

CONSTANT *Rollback*

ASSUME *Rollback* ∈ STRING

Indicates a device is connected

CONSTANT *Connected*

ASSUME *Connected* ∈ STRING

Indicates a device is disconnected

CONSTANT *Disconnected*

ASSUME *Disconnected* ∈ STRING

Indicates that an error occurred when applying a change

CONSTANT *Error*

ASSUME *Error* ∈ STRING

The set of all nodes

CONSTANT *Node*

ASSUME *IsFiniteSet(Node)* ∧ ∀ *n* ∈ *Node* : *n* ∈ STRING

The set of all devices

CONSTANT *Device*

ASSUME *IsFiniteSet(Device)* ∧ ∀ *d* ∈ *Device* : *d* ∈ STRING

An empty constant

CONSTANT *Nil*

ASSUME *Nil* ∈ STRING

Per-node election state

VARIABLE *leader*

Per-node per-device election state

VARIABLE *master*

A sequence of network-wide configuration changes

Each change contains a record of 'changes' for each device

VARIABLE *networkChange*

A record of sequences of device configuration changes

Each sequence is a list of changes in the order in which they  
are to be applied to the device

VARIABLE *deviceChange*

A record of device states - either Available or Unavailable

VARIABLE *deviceState*

A count of leader changes to serve as a state constraint

VARIABLE *electionCount*

A count of configuration changes to serve as a state constraint

VARIABLE *configCount*

A count of device connection changes to serve as a state constraint

VARIABLE *connectionCount*

---

Node variables

$nodeVars \triangleq \langle leader, master \rangle$

Configuration variables

$configVars \triangleq \langle networkChange, deviceChange \rangle$

Device variables

$deviceVars \triangleq \langle deviceState \rangle$

State constraint variables

$constraintVars \triangleq \langle electionCount, configCount, connectionCount \rangle$

$vars \triangleq \langle nodeVars, configVars, deviceVars, constraintVars \rangle$

---

This section models leader election for control loops and for devices. Leader election is modelled as a simple boolean indicating whether each node is the leader for the cluster and for each device. This model implies the ordering of leadership changes is irrelevant to the correctness of the spec.

Set the leader for node  $n$  to  $l$

$SetNodeLeader(n, l) \triangleq$   
 $\wedge leader' = [leader \text{ EXCEPT } ![n] = n = l]$   
 $\wedge electionCount' = electionCount + 1$

$\wedge \text{UNCHANGED } \langle \text{master}, \text{configVars}, \text{deviceVars}, \text{configCount}, \text{connectionCount} \rangle$

Set the master for device  $d$  on node  $n$  to  $l$

$\text{SetDeviceMaster}(n, d, l) \triangleq$   
 $\wedge \text{master}' = [\text{master} \text{ EXCEPT } ![n] = [\text{master}[n] \text{ EXCEPT } ![d] = n = l]]$   
 $\wedge \text{electionCount}' = \text{electionCount} + 1$   
 $\wedge \text{UNCHANGED } \langle \text{leader}, \text{configVars}, \text{deviceVars}, \text{configCount}, \text{connectionCount} \rangle$

---

This section models the northbound *API* for the configuration service.

Enqueue network configuration change  $c$

$\text{SubmitChange}(c) \triangleq$   
 $\wedge \text{Cardinality}(\text{DOMAIN } c) > 0$   
 $\wedge \text{networkChange}' = \text{Append}(\text{networkChange}, [$   
 $\quad \text{phase} \quad \mapsto \text{Change},$   
 $\quad \text{changes} \quad \mapsto c,$   
 $\quad \text{value} \quad \mapsto \text{Len}(\text{networkChange}),$   
 $\quad \text{state} \quad \mapsto \text{Pending},$   
 $\quad \text{incarnation} \mapsto 0])$   
 $\wedge \text{configCount}' = \text{configCount} + 1$   
 $\wedge \text{UNCHANGED } \langle \text{nodeVars}, \text{deviceChange}, \text{deviceVars}, \text{electionCount}, \text{connectionCount} \rangle$

$\text{RollbackChange}(c) \triangleq$   
 $\wedge \text{networkChange}[c].\text{phase} = \text{Change}$   
 $\wedge \text{networkChange}[c].\text{state} = \text{Complete}$   
 $\wedge \text{networkChange}' = [\text{networkChange} \text{ EXCEPT } ![c].\text{phase} = \text{Rollback}, ![c].\text{state} = \text{Pending}]$   
 $\wedge \text{configCount}' = \text{configCount} + 1$   
 $\wedge \text{UNCHANGED } \langle \text{nodeVars}, \text{deviceChange}, \text{deviceVars}, \text{electionCount}, \text{connectionCount} \rangle$

---

This section models the *NetworkChange* reconciler. The reconciler reconciles network changes when the change or one of its device changes is updated.

Return the set of all network changes prior to the given change

$\text{PriorNetworkChanges}(c) \triangleq$   
 $\{n \in \text{DOMAIN } \text{networkChange} : n < c\}$

Return the set of all completed device changes for network change  $c$

$\text{NetworkCompletedChanges}(c) \triangleq$   
 $\{d \in \text{DOMAIN } \text{networkChange}[c].\text{changes} :$   
 $\quad \wedge c \in \text{DOMAIN } \text{deviceChange}[d]$   
 $\quad \wedge \text{deviceChange}[d][c].\text{state} = \text{Complete}\}$

Return a boolean indicating whether all device changes are complete for the given network change

$\text{NetworkChangesComplete}(c) \triangleq$   
 $\text{Cardinality}(\text{NetworkCompletedChanges}(c)) = \text{Cardinality}(\text{DOMAIN } \text{networkChange}[c].\text{changes})$

Return the set of all incomplete device changes prior to network change  $c$

$$\begin{aligned} \text{PriorIncompleteDevices}(c) &\triangleq \\ &\text{UNION } \{ \text{DOMAIN } \text{networkChange}[n].\text{changes} : \\ &\quad n \in \{ n \in \text{PriorNetworkChanges}(c) : \neg \text{NetworkChangesComplete}(n) \} \} \end{aligned}$$

Return the set of all devices configured by network change  $c$

$$\text{NetworkChangeDevices}(c) \triangleq \text{DOMAIN } \text{networkChange}[c].\text{changes}$$

Return the set of all connected devices configured by network change  $c$

$$\text{ConnectedDevices}(c) \triangleq \{ d \in \text{DOMAIN } \text{networkChange}[c].\text{changes} : \text{deviceState}[d] = \text{Connected} \}$$

Return a boolean indicating whether network change  $c$  can be applied

A change can be applied if its devices do not intersect with past device changes that have not been applied

$$\begin{aligned} \text{CanApplyNetworkChange}(c) &\triangleq \\ &\wedge \text{Cardinality}(\text{ConnectedDevices}(c) \cap \text{NetworkChangeDevices}(c)) \neq 0 \\ &\wedge \text{Cardinality}(\text{NetworkChangeDevices}(c) \cap \text{PriorIncompleteDevices}(c)) = 0 \\ &\wedge \vee \text{networkChange}[c].\text{incarnation} = 0 \\ &\vee \text{Cardinality}(\{ d \in \text{DOMAIN } \text{networkChange}[c].\text{changes} : \\ &\quad \wedge \text{deviceChange}[d][c].\text{incarnation} = \text{networkChange}[c].\text{incarnation} \\ &\quad \wedge \text{deviceChange}[d][c].\text{phase} = \text{Rollback} \\ &\quad \wedge \text{deviceChange}[d][c].\text{state} = \text{Complete} \}) = \\ &\quad \text{Cardinality}(\text{DOMAIN } \text{networkChange}[c].\text{changes}) \end{aligned}$$

Return a boolean indicating whether a change exists for the given device

If the device is modified by the change, it must contain a device change that's either *Complete* or with the same 'incarnation' as the network change.

$$\begin{aligned} \text{HasDeviceChange}(d, c) &\triangleq \\ &\wedge c \in \text{DOMAIN } \text{deviceChange}[d] \\ &\wedge \text{deviceChange}[d][c].\text{incarnation} = \text{networkChange}[c].\text{incarnation} \end{aligned}$$

Return a boolean indicating whether device changes have been propagated for the given network change

$$\begin{aligned} \text{HasDeviceChanges}(c) &\triangleq \\ &\text{Cardinality}(\{ d \in \text{DOMAIN } \text{networkChange}[c].\text{changes} : \text{HasDeviceChange}(d, c) \}) = \\ &\quad \text{Cardinality}(\text{DOMAIN } \text{networkChange}[c].\text{changes}) \end{aligned}$$

Add or update the given device changes for the given network change.

If a device change already exists, update the 'incarnation' field.

$$\begin{aligned} \text{CreateDeviceChange}(d, c) &\triangleq \\ &\text{IF } d \in \text{DOMAIN } \text{networkChange}[c].\text{changes} \text{ THEN} \\ &\quad \text{IF } c \in \text{DOMAIN } \text{deviceChange}[d] \text{ THEN} \\ &\quad \quad \text{IF } \text{deviceChange}[d][c].\text{state} = \text{Complete} \text{ THEN} \\ &\quad \quad \quad \text{deviceChange}[d] \\ &\quad \text{ELSE} \\ &\quad \quad [\text{deviceChange}[d] \text{ EXCEPT } ![c].\text{incarnation} = \text{networkChange}[c].\text{incarnation}, \\ &\quad \quad \quad ![c].\text{state} = \text{Pending}] \end{aligned}$$

ELSE  
 $[x \in \{c\} \mapsto [$   
 $\quad phase \mapsto networkChange[c].phase,$   
 $\quad state \mapsto Pending,$   
 $\quad value \mapsto networkChange[c].value,$   
 $\quad incarnation \mapsto networkChange[c].incarnation]] @@ deviceChange[d]$   
 ELSE  
 $deviceChange[d]$

Add or update device changes for the given network change  
 $CreateDeviceChanges(c) \triangleq$   
 $deviceChange' = [d \in DOMAIN \ deviceChange \mapsto CreateDeviceChange(d, c)]$

Rollback device change  $c$  for device  $d$   
 $RollbackDeviceChange(d, c) \triangleq$   
 IF  $\wedge c \in DOMAIN \ deviceChange[d]$   
 $\quad \wedge \vee deviceChange[d][c].phase = Change$   
 $\quad \vee \wedge deviceChange[d][c].phase = Rollback$   
 $\quad \wedge deviceChange[d][c].state = Failed$   
 THEN  
 $[deviceChange[d] \text{ EXCEPT } ![c].phase = Rollback, ![c].state = Pending]$   
 ELSE  
 $deviceChange[d]$

Roll back device changes  
 $RollbackDeviceChanges(c) \triangleq$   
 $deviceChange' = [d \in DOMAIN \ deviceChange \mapsto RollbackDeviceChange(d, c)]$

Return a boolean indicating whether the given device change is *Failed*  
 $IsFailedDeviceChange(d, c) \triangleq$   
 $\quad \wedge c \in DOMAIN \ deviceChange[d]$   
 $\quad \wedge deviceChange[d][c].incarnation = networkChange[c].incarnation$   
 $\quad \wedge deviceChange[d][c].state = Failed$

Return a boolean indicating whether the given device change is *Complete*  
 $IsCompleteDeviceChange(d, c) \triangleq$   
 $\quad \wedge c \in DOMAIN \ deviceChange[d]$   
 $\quad \wedge deviceChange[d][c].incarnation = networkChange[c].incarnation$   
 $\quad \wedge deviceChange[d][c].phase = Change$   
 $\quad \wedge deviceChange[d][c].state = Complete$

Return a boolean indicating whether any device change is *Failed* for the given network change  
 $HasFailedDeviceChanges(c) \triangleq$   
 $\quad Cardinality(\{d \in DOMAIN \ networkChange[c].changes :$   
 $\quad \quad IsFailedDeviceChange(d, c)\}) \neq 0$

Return a boolean indicating whether all device changes are *Complete* for the given network change

$$\begin{aligned}
DeviceChangesComplete(c) &\triangleq \\
&Cardinality(\{d \in \text{DOMAIN } networkChange[c].changes : \\
&\quad IsCompleteDeviceChange(d, c)\}) = \\
&\quad Cardinality(\text{DOMAIN } networkChange[c].changes)
\end{aligned}$$

$$\begin{aligned}
&\text{Reconcile a network change state} \\
ReconcileNetworkChange(n, c) &\triangleq \\
&\wedge leader[n] \\
&\wedge networkChange[c].state = Pending \\
&\wedge \vee \wedge \neg HasDeviceChanges(c) \\
&\quad \wedge CreateDeviceChanges(c) \\
&\quad \wedge \text{UNCHANGED } \langle networkChange \rangle \\
&\vee \wedge HasDeviceChanges(c) \\
&\quad \wedge \vee \wedge networkChange[c].phase = Change \\
&\quad \quad \wedge \vee \wedge CanApplyNetworkChange(c) \\
&\quad \quad \quad \wedge networkChange' = [networkChange \text{ EXCEPT} \\
&\quad \quad \quad \quad ![c].incarnation = networkChange[c].incarnation + 1] \\
&\quad \quad \quad \wedge \text{UNCHANGED } \langle deviceChange \rangle \\
&\quad \vee \wedge DeviceChangesComplete(c) \\
&\quad \quad \wedge networkChange' = [networkChange \text{ EXCEPT} \\
&\quad \quad \quad ![c].state = Complete] \\
&\quad \quad \quad \wedge \text{UNCHANGED } \langle deviceChange \rangle \\
&\quad \vee \wedge HasFailedDeviceChanges(c) \\
&\quad \quad \wedge RollbackDeviceChanges(c) \\
&\quad \quad \quad \wedge \text{UNCHANGED } \langle networkChange \rangle \\
&\quad \text{TODO} \\
&\quad \vee \wedge networkChange[c].phase = Rollback \\
&\quad \quad \wedge networkChange' = [networkChange \text{ EXCEPT} \\
&\quad \quad \quad ![c].state = Complete] \\
&\quad \quad \quad \wedge \text{UNCHANGED } \langle deviceChange \rangle \\
&\wedge \text{UNCHANGED } \langle nodeVars, deviceVars, constraintVars \rangle
\end{aligned}$$

---

This section models the *DeviceChange* reconciler.

$$\begin{aligned}
ReconcileDeviceChange(n, d, c) &\triangleq \\
&\wedge master[n][d] \\
&\wedge deviceChange[d][c].state = Pending \\
&\wedge deviceChange[d][c].incarnation > 0 \\
&\wedge \vee \wedge deviceState[d] = Connected \\
&\quad \wedge deviceChange' = [deviceChange \text{ EXCEPT} \\
&\quad \quad ![d] = [deviceChange[d] \text{ EXCEPT } ![c].state = Complete]] \\
&\vee \wedge deviceState[d] = Disconnected \\
&\quad \wedge deviceChange' = [deviceChange \text{ EXCEPT} \\
&\quad \quad ![d] = [deviceChange[d] \text{ EXCEPT } ![c].state = Failed]] \\
&\wedge \text{UNCHANGED } \langle nodeVars, networkChange, deviceVars, constraintVars \rangle
\end{aligned}$$

---

This section models device states. Devices begin in the *Disconnected* state and can only be configured while in the *Connected* state.

Set device  $d$  state to *Connected*  
 $ConnectDevice(d) \triangleq$   
 $\wedge deviceState' = [deviceState \text{ EXCEPT } ![d] = Connected]$   
 $\wedge connectionCount' = connectionCount + 1$   
 $\wedge \text{UNCHANGED } \langle nodeVars, configVars, electionCount, configCount \rangle$

Set device  $d$  state to *Disconnected*  
 $DisconnectDevice(d) \triangleq$   
 $\wedge deviceState' = [deviceState \text{ EXCEPT } ![d] = Disconnected]$   
 $\wedge connectionCount' = connectionCount + 1$   
 $\wedge \text{UNCHANGED } \langle nodeVars, configVars, electionCount, configCount \rangle$

---

*Init* and next state predicates

$Init \triangleq$   
 $\wedge leader = [n \in Node \mapsto \text{FALSE}]$   
 $\wedge master = [n \in Node \mapsto [d \in Device \mapsto \text{FALSE}]]$   
 $\wedge networkChange = \langle \rangle$   
 $\wedge deviceChange = [d \in Device \mapsto [x \in \{ \} \mapsto [phase \mapsto Change, state \mapsto Pending]]]$   
 $\wedge deviceState = [d \in Device \mapsto Disconnected]$   
 $\wedge electionCount = 0$   
 $\wedge configCount = 0$   
 $\wedge connectionCount = 0$

$Next \triangleq$   
 $\vee \exists d \in \text{SUBSET } Device :$   
 $\quad SubmitChange([x \in d \mapsto 1])$   
 $\vee \exists c \in \text{DOMAIN } networkChange :$   
 $\quad RollbackChange(c)$   
 $\vee \exists n \in Node :$   
 $\quad \exists l \in Node :$   
 $\quad \quad SetNodeLeader(n, l)$   
 $\vee \exists n \in Node :$   
 $\quad \exists d \in Device :$   
 $\quad \exists l \in Node :$   
 $\quad \quad SetDeviceMaster(n, d, l)$   
 $\vee \exists n \in Node :$   
 $\quad \exists c \in \text{DOMAIN } networkChange :$   
 $\quad \quad ReconcileNetworkChange(n, c)$   
 $\vee \exists n \in Node :$   
 $\quad \exists d \in Device :$   
 $\quad \exists c \in \text{DOMAIN } deviceChange[d] :$   
 $\quad \quad ReconcileNetworkChange(n, c)$

$$\begin{aligned}
& \forall \exists n \in Node : \\
& \quad \exists d \in Device : \\
& \quad \quad \exists c \in \text{DOMAIN } deviceChange[d] : \\
& \quad \quad \quad ReconcileDeviceChange(n, d, c) \\
& \forall \exists d \in Device : \\
& \quad \quad ConnectDevice(d) \\
& \forall \exists d \in Device : \\
& \quad \quad DisconnectDevice(d) \\
Spec & \triangleq Init \wedge \Box[Next]_{vars}
\end{aligned}$$


---

```

\ * Modification History
\ * Last modified Wed Sep 22 17:28:11 PDT 2021 by jordanhalterman
\ * Created Wed Sep 22 13:22:32 PDT 2021 by jordanhalterman

```