
MODULE *Config*

EXTENDS *Naturals, FiniteSets, Sequences, TLC*

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

Indicates that a configuration change is being applied to the network
 CONSTANT *Applying*

Indicates that a configuration change has been applied to the network
 CONSTANT *Complete*

Indicates that a configuration change was successful
 CONSTANT *Succeeded*

Indicates that a configuration change failed
 CONSTANT *Failed*

The set of all nodes
 CONSTANT *Node*

The set of all devices
 CONSTANT *Device*

The set of all possible configuration changes
 CONSTANT *Change*

An empty constant
 CONSTANT *Nil*

Per-node election state
 VARIABLE *nodeState*

Per-node per-device election state
 VARIABLE *deviceState*

Store of network-wide configuration changes
 VARIABLE *networkChange*

Store of device configuration changes
 VARIABLE *deviceChange*

Leader election

Sets the current leader for the node
 $SetNodeLeader(n, l) \triangleq$
 $\wedge nodeState' = [nodeState \text{ EXCEPT } ![n] = n = l]$
 $\wedge \text{UNCHANGED } \langle deviceState, networkChange, deviceChange \rangle$

Sets the current leader for a device
 $SetDeviceLeader(n, d, l) \triangleq$
 $\wedge deviceState' = [deviceState \text{ EXCEPT } ![n] = [deviceState[n] \text{ EXCEPT } ![d] = n = l]]$
 $\wedge \text{UNCHANGED } \langle nodeState, networkChange, deviceChange \rangle$

Northbound API

$Configure(c) \triangleq$
 $\wedge networkChange' = Append(networkChange, [changes \mapsto c, status \mapsto Pending])$
 $\wedge \text{UNCHANGED } \langle nodeState, deviceState, deviceChange \rangle$

Network configuration change scheduler

Node 'n' handles a network configuration change event 'c'
 $NetworkSchedulerNetworkChange(n, c) \triangleq$
 $\wedge nodeState[n] = \text{TRUE}$ Verify this node is the leader
 $\wedge \text{LET } change \triangleq networkChange[c] \text{ IN}$
 If the change does not intersect with the set of all pending/applied changes
 prior to the change then set the change status to *Applying*
 $\text{LET } changeDevices \triangleq \text{DOMAIN } change.changes$
 $\text{priorDevices} \triangleq \{d \in deviceChange : \{i \in \text{DOMAIN } deviceChange[d] : i < c \wedge deviceChange[d][i].status \in \{Pending, Applying\}\}\}$
 IN
 IF $Cardinality(changeDevices \cap priorDevices) = 0$ THEN
 $\wedge networkChange' = [networkChange \text{ EXCEPT } ![c].status = Applying]$
 ELSE
 $\wedge \text{UNCHANGED } \langle networkChange \rangle$
 $\wedge \text{UNCHANGED } \langle nodeState, deviceState, deviceChange \rangle$

Network configuration controller

Adds or updates a device change
 $SetDeviceChange(d, c, s) \triangleq$
 $\text{LET } change \triangleq networkChange[c] \text{ IN}$
 IF $d \in \text{DOMAIN } change.changes$ THEN
 IF $Cardinality(\{x \in \text{DOMAIN } deviceChange[d] : deviceChange[d][x].id = c\}) = 0$ THEN
 $Append(deviceChange[d], [change.changes[d] \text{ EXCEPT } !.id = c, !.network = c, !.status = s])$
 ELSE
 $[deviceChange \text{ EXCEPT } ![\text{CHOOSE } x \in \text{DOMAIN } deviceChange[d] : deviceChange[d][x].id = c].status = s]$
 ELSE
 $deviceChange[d]$

Node 'n' handles a network configuration change 'c'
 $NetworkControllerNetworkChange(n, c) \triangleq$
 $\wedge nodeState[n] = \text{TRUE}$

$$\begin{aligned}
& \wedge \text{LET } change \triangleq networkChange[c] \text{IN} \\
& \quad \vee \wedge change.status = Pending \\
& \quad \quad \wedge deviceChange' = [d \in Device \mapsto SetDeviceChange(d, c, Pending)] \\
& \quad \vee \wedge change.status = Applying \\
& \quad \quad \wedge deviceChange' = [d \in Device \mapsto SetDeviceChange(d, c, Applying)] \\
& \quad \quad \wedge Cardinality(\text{DOMAIN } change.changes \cap \{d \in deviceChange \\
& \quad \quad \quad : \{i \in \text{DOMAIN } deviceChange[d] \\
& \quad \quad \quad \quad : deviceChange[d][i].network = c \\
& \quad \quad \quad \quad \wedge deviceChange[d][i].status = Applying\}\}) < Cardinality(change.changes \\
& \quad \wedge deviceChange' = [d \in Device \mapsto \text{IF } d \in \text{DOMAIN } change.changes \text{ THEN} \\
& \quad \quad Append(deviceChange[d], [change.changes[d] \text{ EXCEPT} \\
& \quad \quad \quad !.network = c, \\
& \quad \quad \quad !.status = Applying]) \\
& \quad \quad \text{ELSE } deviceChange[d]] \\
& \quad \vee \wedge change.status = Complete \\
& \quad \quad \wedge \text{UNCHANGED } \langle deviceChange \rangle \\
& \wedge \text{UNCHANGED } \langle nodeState, deviceState, networkChange \rangle
\end{aligned}$$

Node 'n' handles a device configuration change 'c'

$$\begin{aligned}
NetworkControllerDeviceChange(n, d, c) & \triangleq \\
& \wedge nodeState[n] = \text{TRUE} \\
& \wedge \text{LET } change \triangleq deviceChange[d][c] \\
& \quad \text{IN} \\
& \quad \vee \wedge deviceChange.status = Complete \\
& \quad \quad \wedge \text{LET } netChange \triangleq networkChange[change.network] \\
& \quad \quad \quad completed \triangleq \{x \in \text{DOMAIN } netChange.changes : \\
& \quad \quad \quad \quad deviceChange[x][\text{CHOOSE } i \in \text{DOMAIN } deviceChange[x] : \\
& \quad \quad \quad \quad \quad deviceChange[x][i].network = change.network].status = Complete\} \\
& \quad \quad \quad succeeded \triangleq \{x \in \text{DOMAIN } netChange.changes : \\
& \quad \quad \quad \quad deviceChange[x][\text{CHOOSE } i \in \text{DOMAIN } deviceChange[x] : \\
& \quad \quad \quad \quad \quad deviceChange[x][i].network = change.network].result = Succeeded\} \\
& \quad \quad \text{IN} \\
& \quad \quad \vee \wedge Cardinality(completed) = Cardinality(netChange.changes) \\
& \quad \quad \quad \wedge \text{IF } Cardinality(succeeded) = Cardinality(completed) \text{ THEN} \\
& \quad \quad \quad \quad networkChange' = [networkChange \text{ EXCEPT } ![change.network] = [\\
& \quad \quad \quad \quad \quad networkChange[change.network] \text{ EXCEPT} \\
& \quad \quad \quad \quad \quad \quad !.status = Complete, !.result = Succeeded]] \\
& \quad \quad \quad \text{ELSE} \\
& \quad \quad \quad \quad networkChange' = [networkChange \text{ EXCEPT } ![change.network] = [\\
& \quad \quad \quad \quad \quad networkChange[change.network] \text{ EXCEPT} \\
& \quad \quad \quad \quad \quad \quad !.status = Complete, !.result = Failed]] \\
& \quad \vee \wedge change.status \neq Complete \\
& \quad \quad \wedge \text{UNCHANGED } \langle networkChange \rangle \\
& \wedge \text{UNCHANGED } \langle nodeState, deviceState, deviceChange \rangle
\end{aligned}$$

Device configuration controller

Node 'n' handles a device configuration change event 'c'

$DeviceControllerDeviceChange(n, d, c) \triangleq$

$\wedge deviceState[n][d] = \text{TRUE}$

$\wedge \text{LET } change \triangleq deviceChange[d][c]$

IN

$\vee \wedge change.status = \text{Applying}$

$\wedge deviceChange' = [deviceChange \text{ EXCEPT } ![d] = [deviceChange[d] \text{ EXCEPT } ![c] = [$
 $deviceChange[d][c] \text{ EXCEPT } !.status = \text{Complete}, !.result = \text{Succeeded}]]]$

$\vee \wedge change.status \neq \text{Applying}$

$\wedge \text{UNCHANGED } \langle deviceChange \rangle$

$\wedge \text{UNCHANGED } \langle nodeState, deviceState, networkChange \rangle$

Init and next state predicates

$Init \triangleq$

$\wedge nodeState = [n \in Node \mapsto \text{FALSE}]$

$\wedge deviceState = [n \in Node \mapsto [d \in Device \mapsto \text{FALSE}]]$

$\wedge networkChange = \langle \rangle$

$\wedge deviceChange = [n \in Device \mapsto \langle \rangle]$

$Next \triangleq$

$\vee \exists d \in \text{SUBSET } Device : \exists c \in Change : \text{Configure}([d \rightarrow c])$

$\vee \exists n \in Node :$

$\exists l \in Node :$

$SetNodeLeader(n, l)$

$\vee \exists n \in Node :$

$\exists d \in Device :$

$\exists l \in Node :$

$SetDeviceLeader(n, d, l)$

$\vee \exists n \in Node :$

$\exists c \in \text{DOMAIN } networkChange :$

$NetworkSchedulerNetworkChange(n, c)$

$\vee \exists n \in Node :$

$\exists c \in \text{DOMAIN } networkChange :$

$NetworkControllerNetworkChange(n, c)$

$\vee \exists n \in Node :$

$\exists d \in Device :$

$\exists c \in \text{DOMAIN } deviceChange[d] :$

$NetworkControllerDeviceChange(n, d, c)$

$\vee \exists n \in Node :$

$\exists d \in Device :$

$\exists c \in \text{DOMAIN } deviceChange[d] :$

$DeviceControllerDeviceChange(n, d, c)$

\ * Modification History
\ * Last modified Sat *Sep 28 02:28:51 PDT 2019* by *jordanhalterman*
\ * Created *Fri Sep 27 13:14:24 PDT 2019* by *jordanhalterman*