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
$SetNodeLeader(n, l) \triangleq$
$\quad \land\ nodeState' = [nodeState \text{ EXCEPT } ![n] = n = l]$
$\quad \land\ \text{UNCHANGED } \langle\rangle$

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

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

```
Network configuration change scheduler

Node 'n' handles a network configuration change event 'c'
$NetworkSchedulerNetworkChange(n, c) \triangleq$
    $\land nodeState[n] = \text{TRUE}$ Verify this node is the leader
    $\land \text{LET } change \triangleq networkChange[c] \text{IN}$
            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
          $\text{LET } changeDevices \triangleq \text{DOMAIN } change.changes$
              $priorDevices \triangleq \{d \in deviceChange : \{i \in \text{DOMAIN } deviceChange[d] : i < c \land deviceChange[d]$
          IN
              $\text{IF } Cardinality(changeDevices \cap priorDevices) = 0 \text{ THEN}$
                  $\land networkChange' = [networkChange \text{ EXCEPT } ![c].status = Applying]$
               ELSE
                  $\land \text{UNCHANGED } \langle networkChange \rangle$
    $\land \text{UNCHANGED } \langle nodeState, deviceState, deviceChange \rangle$
```

```
Network configuration controller

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

Node 'n' handles a network configuration change 'c'
$NetworkControllerNetworkChange(n, c) \triangleq$
    $\land nodeState[n] = \text{TRUE}$
    $\land \text{LET } change \triangleq networkChange[c] \text{IN}$
            $\lor \land change.status = Pending$
            ∧ deviceChange′ = [d ∈ Device ↦ SetDeviceChange(d, c, Pending)]
          ∨ ∧ change.status = Applying
            ∧ deviceChange′ = [d ∈ Device ↦ SetDeviceChange(d, c, Applying)]
            ∧ Cardinality(DOMAIN change.changes ∩ {d ∈ deviceChange : {i ∈ DOMAIN deviceChange[d] : d
            ∧ deviceChange′ = [d ∈ Device ↦ IF d ∈ DOMAIN change.changes THEN Append(deviceChange
          ∨ ∧ change.status = Complete
            ∧ UNCHANGED ⟨deviceChange⟩
    ∧ UNCHANGED ⟨nodeState, deviceState, networkChange⟩
```

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

```
NetworkControllerDeviceChange(n, d, c) ≜
    ∧ nodeState[n] = TRUE
    ∧ LET change ≜ deviceChange[d][c]
      IN
          ∨ ∧ deviceChange.status = Complete
            ∧ LET netChange ≜ networkChange[change.network]
                  completeChanges ≜ {x ∈ DOMAIN netChange.changes : deviceChange[x][CHOOSE i ∈ DOM
                  succeededChanges ≜ {x ∈ DOMAIN netChange.changes : deviceChange[x][CHOOSE i ∈ DO
              IN
                  ∨ ∧ Cardinality(completeChanges) = Cardinality(netChange.changes)
                    ∧ IF Cardinality(succeededChanges) = Cardinality(completeChanges) THEN
                          networkChange′ = [networkChange EXCEPT ![change.network] = [networkChange
                      ELSE
                          networkChange′ = [networkChange EXCEPT ![change.network] = [networkChange
          ∨ ∧ change.status ≠ Complete
            ∧ UNCHANGED ⟨networkChange⟩
    ∧ UNCHANGED ⟨nodeState, deviceState, deviceChange⟩
```

Device configuration controller

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

```
DeviceControllerDeviceChange(n, d, c) ≜
    ∧ deviceState[n][d] = TRUE
    ∧ LET change ≜ deviceChange[d][c]
      IN
          ∨ ∧ change.status = Applying
            ∧ deviceChange′ = [deviceChange EXCEPT ![d] = [deviceChange[d] EXCEPT ![c] = [deviceChange
          ∨ ∧ change.status ≠ Applying
            ∧ UNCHANGED ⟨deviceChange⟩
    ∧ UNCHANGED ⟨nodeState, deviceState, networkChange⟩
```

Init and next state predicates

```
Init ≜
    ∧ nodeState = [n ∈ Node ↦ FALSE]
    $\land\ deviceState = [n \in Node \mapsto [d \in Device \mapsto \text{FALSE}]]$
    $\land\ networkChange = \langle\rangle$
    $\land\ deviceChange = [n \in Device \mapsto \langle\rangle]$

$Next\ \triangleq$
    $\lor\ \exists\, d \in \text{SUBSET}\ Device : \exists\, c \in Change : Configure([d \rightarrow c])$
    $\lor\ \exists\, n \in Node :$
        $\exists\, l \in Node :$
          $SetNodeLeader(n,\ l)$
    $\lor\ \exists\, n \in Node :$
        $\exists\, d \in Device :$
          $\exists\, l \in Node :$
            $SetDeviceLeader(n,\ d,\ l)$
    $\lor\ \exists\, n \in Node :$
        $\exists\, c \in \text{DOMAIN}\ networkChange :$
          $NetworkSchedulerNetworkChange(n,\ c)$
    $\lor\ \exists\, n \in Node :$
        $\exists\, c \in \text{DOMAIN}\ networkChange :$
          $NetworkControllerNetworkChange(n,\ c)$
    $\lor\ \exists\, n \in Node :$
        $\exists\, d \in Device :$
          $\exists\, c \in \text{DOMAIN}\ deviceChange[d] :$
            $NetworkControllerDeviceChange(n,\ d,\ c)$
    $\lor\ \exists\, n \in Node :$
        $\exists\, d \in Device :$
          $\exists\, c \in \text{DOMAIN}\ deviceChange[d] :$
            $DeviceControllerDeviceChange(n,\ d,\ c)$
```

* Modification History
* Last modified Sat *Sep* 28 02:12:49 *PDT* 2019 by *jordanhalterman*
* Created *Fri Sep* 27 13:14:24 *PDT* 2019 by *jordanhalterman*