MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

An empty constant
CONSTANT *Nil*

Transaction status constants
CONSTANTS
TransactionPending,
TransactionValidating,
TransactionApplying,
TransactionComplete,
TransactionFailed

Configuration status constants
CONSTANTS
ConfigurationPending,
ConfigurationInitializing,
ConfigurationUpdating,
ConfigurationComplete,
ConfigurationFailed

The set of all nodes
CONSTANT *Node*

Target is the possible targets, paths, and values

Example: *Target* $\triangleq$ [
*target*1 $\mapsto$ [
*path*1 $\mapsto$ {"*value*1", "*value*2"},
*path*2 $\mapsto$ {"*value*2", "*value*3"}],
*target*2 $\mapsto$ [
*path*2 $\mapsto$ {"*value*3", "*value*4"},
*path*3 $\mapsto$ {"*value*4", "*value*5"}]]

CONSTANT *Target*

ASSUME *Nil* $\in$ STRING

ASSUME *TransactionPending* $\in$ STRING
ASSUME *TransactionValidating* $\in$ STRING
ASSUME *TransactionApplying* $\in$ STRING

ASSUME $TransactionComplete \in$ STRING
ASSUME $TransactionFailed \in$ STRING

ASSUME $ConfigurationPending \in$ STRING
ASSUME $ConfigurationInitializing \in$ STRING
ASSUME $ConfigurationUpdating \in$ STRING
ASSUME $ConfigurationComplete \in$ STRING
ASSUME $ConfigurationFailed \in$ STRING

ASSUME $\land IsFiniteSet(Node)$
$\land \forall\, n \in Node :$
$\land\ n \notin$ DOMAIN $Target$
$\land\ n \in$ STRING

ASSUME $\land \forall\, t \in$ DOMAIN $Target :$
$\land\ IsFiniteSet(Target[t])$
$\land\ t \notin Node$
$\land\ t \in$ STRING

TYPE $TransactionStatus ::= status \in$
$\{TransactionPending,$
$TransactionValidating,$
$TransactionApplying,$
$TransactionComplete,$
$TransactionFailed\}$

TYPE Transaction $\triangleq$ [
id ::= $id \in$ STRING,
$index$ ::= $index \in Nat$,
revision ::= revision $\in Nat$,
$atomic$ ::= $atomic \in$ BOOLEAN ,
$sync$ ::= $sync \in$ BOOLEAN ,
$changes$ ::= [$target \in$ SUBSET (DOMAIN $Target$) $\mapsto$ [
$path \in$ SUBSET (DOMAIN $Target[target]$) $\mapsto$ [
$value$ ::= $value \in$ STRING,
$delete$ ::= $delete \in$ BOOLEAN]]],
$status$ ::= $status \in TransactionStatus$]

TYPE $ConfigurationStatus ::= status \in$
$\{ConfigurationPending,$
$ConfigurationInitializing,$
$ConfigurationUpdating,$
$ConfigurationComplete,$
$ConfigurationFailed\}$

TYPE Configuration $\triangleq$ [
id ::= $id \in$ STRING,
revision ::= revision $\in Nat$,
$target$ ::= $target \in$ STRING,

```
  paths ::= [ path ∈ SUBSET (DOMAIN Target[target]) ↦ [
      value ::= value ∈ STRING,
      index ::= index ∈ Nat,
      deleted ::= delete ∈ BOOLEAN ]],
  txIndex        ::= txIndex ∈ Nat,
  syncIndex      ::= syncIndex ∈ Nat,
  mastershipTerm ::= mastershipTerm ∈ Nat,
  status         ::= status ∈ ConfigurationStatus]
```

A sequence of transactions

Each transactions contains a record of 'changes' for a set of targets

VARIABLE *transactions*

A record of target configurations

Each configuration represents the desired state of the target

VARIABLE *configurations*

A record of target states

VARIABLE *targets*

A record of target masters

VARIABLE *masters*

$vars \triangleq \langle transactions,\ configurations,\ targets \rangle$

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

This crazy thing returns the set of all possible sets of valid changes

```
ValidChanges ≜
  LET allPaths ≜ UNION {(DOMAIN Target[t]) : t ∈ DOMAIN Target}
      allValues ≜ UNION {UNION {Target[t][p] : p ∈ DOMAIN Target[t]} : t ∈ DOMAIN Target}
  IN
    {targetPathValues ∈ SUBSET (Target × allPaths × allValues × BOOLEAN ) :
      ∧ ∀ target ∈ DOMAIN Target :
        LET targetIndexes ≜ {i ∈ 1 .. Len(targetPathValues) : ∧ targetPathValues[i][1] = target}
        IN  ∨ Cardinality(targetIndexes) = 0
            ∨ ∧ Cardinality(targetIndexes) = 1
              ∧ LET targetPathValue ≜ targetPathValues[CHOOSE index ∈ targetIndexes : TRUE]
                    targetPath      ≜ targetPathValue[2]
                    targetValue     ≜ targetPathValue[3]
                IN
                  ∧ targetPath \ (DOMAIN Target[target]) = {}
                  ∧ targetValue ∈ Target[target][targetPath]}
```

Add a set of changes to the transaction log

```
Change ≜
  ∧ ∃ changes ∈ ValidChanges :
```

$\land\ transactions' = Append(transactions, [index \mapsto Len(transactions) + 1,$
$atomic \mapsto \text{FALSE},$
$sync \mapsto \text{FALSE},$
$changes \mapsto changes,$
$status \mapsto TransactionPending])$
$\land\ \text{UNCHANGED}\ \langle configurations,\ targets \rangle$

This section models the Transaction log reconciler.

Reconcile the transaction log
$ReconcileTransaction(n,\ tx) \triangleq$
If the transaction is *Pending*, begin validation if the prior transaction
has already been applied. This simplifies concurrency control in the controller
and guarantees transactions are applied to the configurations in sequential order.
$\land\ \lor\ \land\ tx.status = TransactionPending$
$\land\ \lor\ \land\ tx.index > 1$
$\land\ transactions[tx.index - 1].status \in \{TransactionComplete,\ TransactionFailed\}$
$\lor\ tx.index = 1$
$\land\ transactions' = [transactions\ \text{EXCEPT}\ ![tx.index].status = TransactionValidating]$
$\land\ \text{UNCHANGED}\ \langle configurations \rangle$
If the transaction is in the *Validating* state, compute and validate the
Configuration for each target.
$\lor\ \land\ tx.status = TransactionValidating$
If validation fails any target, mark the transaction *Failed*.
If validation is successful, proceed to *Applying*.
$\land\ \exists\, valid \in \text{BOOLEAN}\ :$
$\lor\ \land\ valid$
$\land\ transactions' = [transactions\ \text{EXCEPT}\ ![tx.index].status = TransactionApplying]$
$\lor\ \land\ \neg valid$
$\land\ transactions' = [transactions\ \text{EXCEPT}\ ![tx.index].status = TransactionFailed]$
$\land\ \text{UNCHANGED}\ \langle configurations \rangle$
If the transaction is in the *Applying* state, update the Configuration for each
target and *Complete* the transaction.
$\lor\ \land\ tx.status = TransactionApplying$
$\land\ \lor\ \land\ tx.atomic$
TODO: Apply atomic transactions here
$\land\ transactions' = [transactions\ \text{EXCEPT}\ ![tx.index].status = TransactionComplete]$
$\land\ \text{UNCHANGED}\ \langle configurations \rangle$
$\land\ \lor\ \land\ \neg tx.atomic$
Add the transaction index to each updated path
$\land\ configurations' = [$
$t \in \text{DOMAIN}\ Target \mapsto [$
$configurations[t]\ \text{EXCEPT}$
$!.paths = [path \in \text{DOMAIN}\ tx.changes \mapsto$

```
                    tx.changes[path] @@ [index ↦ tx.index]] @@ configurations[t].paths,
                !.txIndex = tx.index,
                !.status          = ConfigurationPending]]
        ∧ transactions' = [transactions EXCEPT ![tx.index].status = TransactionComplete]
∧ UNCHANGED ⟨targets⟩
```

This section models the Configuration reconciler.

```
ReconcileConfiguration(n, c) ≜
  ∧ ∨ ∧ c.status = ConfigurationInitializing
      ∧ masters[c.target].master = n
      Merge the configuration paths with the target paths, removing paths
      that have been marked deleted
      ∧ targets' = [targets EXCEPT ![c.target] =
            [p ∈ {p ∈ DOMAIN c.paths : ¬c.paths[p].deleted} ↦ [value ↦ c.paths[p]]] @@
            [p ∈ {p ∈ DOMAIN targets[c.target] : ¬c.paths[p].deleted} ↦ targets[c.target][p]]]
      Set the configuration's status to Complete
      ∧ configurations' = [configurations EXCEPT ![c.id].status    = ConfigurationComplete,
                                                 ![c.id].syncIndex = c.txIndex]
    If the configuration is marked ConfigurationUpdating, we only need to
    push paths that have changed since the target was initialized or last
    updated by the controller. The set of changes made since the last
    synchronization are identified by comparing the index of each path-value
    to the last synchronization index, syncIndex
    ∨ ∧ c.status = ConfigurationUpdating
      ∧ masters[c.target].master = n
      Compute the set of updated and deleted paths by comparing
      their indexes to the target s last sync index.
      ∧ LET updatedPaths ≜ {p ∈ DOMAIN c.paths : c.paths[p].index > c.syncIndex}
            deletedPaths ≜ {p ∈ updatedPaths : c.paths[p].deleted}
        IN
          Update the target paths by adding/updating paths that have changed and
          removing paths that have been deleted since the last sync.
          ∧ targets' = [targets EXCEPT ![c.target] =
                [p ∈ updatedPaths \ deletedPaths ↦ c.paths[p]] @@
                [p ∈ DOMAIN targets[c.target] \ deletedPaths ↦ targets[c.target][p]]]
      ∧ configurations' = [configurations EXCEPT ![c.id].status    = ConfigurationComplete,
                                                 ![c.id].syncIndex = c.txIndex]
    If the configuration is marked ConfigurationPending and mastership has changed
    (indicated by an increased mastership term), mark the configuration
    ConfigurationInitializing to force full re-synchronization.
    ∨ ∧ c.status = ConfigurationPending
      ∧ masters[c.target].term > c.mastershipTerm
      ∧ configurations' = [configurations EXCEPT ![c.id].status = ConfigurationInitializing,
```

$!\,[c.id].mastershipTerm = masters[c.target].term]$

If the configuration is in a completed state and mastership has been lost,
revert it to *ConfigurationPending*. This can occur when the connection to the
target has been lost and the mastership is no longer valid.
TODO: We still need to model mastership changes

$\lor\ \land\ c.status \in \{ConfigurationComplete,\ ConfigurationFailed\}$
$\quad\land\ masters[c.target].master = Nil$
$\quad\land\ configurations' = [configurations\ \text{EXCEPT}\ ![c.id].status = ConfigurationPending]$
$\land\ \text{UNCHANGED}\ \langle transactions \rangle$

Init and next state predicates

$$
\begin{aligned}
&Init\ \triangleq \\
&\quad\land\ transactions = \langle\rangle \\
&\quad\land\ configurations = [t \in Target \mapsto [\\
&\qquad\qquad\qquad id \quad\quad \mapsto t, \\
&\qquad\qquad\qquad config \mapsto [path \in \{\} \mapsto [\\
&\qquad\qquad\qquad\qquad\qquad path \quad \mapsto path, \\
&\qquad\qquad\qquad\qquad\qquad value \quad \mapsto Nil, \\
&\qquad\qquad\qquad\qquad\qquad index \quad \mapsto 0, \\
&\qquad\qquad\qquad\qquad\qquad deleted \mapsto \text{FALSE}]]]] \\
&\quad\land\ targets\ \ = [t \in Target \mapsto \\
&\qquad\qquad [path \in \{\} \mapsto [\\
&\qquad\qquad\quad value \mapsto Nil]]] \\
&\quad\land\ masters = [t \in Target \mapsto [master \mapsto Nil,\ term \mapsto 0]]
\end{aligned}
$$

$$
\begin{aligned}
&Next\ \triangleq \\
&\quad\lor\ Change \\
&\quad\lor\ \exists\, n \in Node : \\
&\qquad\exists\, t \in \text{DOMAIN}\ transactions : \\
&\qquad\quad ReconcileTransaction(n,\ t) \\
&\quad\lor\ \exists\, n \in Node : \\
&\qquad\exists\, c \in configurations : \\
&\qquad\quad ReconcileConfiguration(n,\ c)
\end{aligned}
$$

$Spec\ \triangleq\ Init \land \Box[Next]_{vars}$

---

* Modification History
* Last modified *Thu Jan* 13 23:04:14 *PST* 2022 by *jordanhalterman*
* Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*