MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

An empty constant
CONSTANT *Nil*

Transaction constants
CONSTANTS
Pending,
Validating,
Applying,
Complete,
Failed

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 *Pending* $\in$ STRING
ASSUME *Validating* $\in$ STRING
ASSUME *Applying* $\in$ STRING
ASSUME *Complete* $\in$ STRING
ASSUME *Failed* $\in$ 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 $Change \triangleq$ [
$target$ ::= $target \in$ STRING,
$path$ ::= $path \in$ STRING,
$value$ ::= $value \in$ STRING,
$delete$ ::= $delete \in$ BOOLEAN
]

TYPE State ::= $state \in \{Pending, Validating, Applying, Complete, Failed\}$

TYPE Transaction $\triangleq$ [
id ::= $id \in$ STRING,
$index$ ::= $index \in Nat$,
revision ::= revision $\in Nat$,
$atomic$ ::= $atomic \in$ BOOLEAN ,
$sync$ ::= $sync \in$ BOOLEAN ,
$changes$ ::= $[t \in$ SUBSET $Target \mapsto [p \in$ SUBSET $Path \mapsto$
$changes$ ::= $[i \in 1 \,..\, Nat \mapsto changes[i] \in Change]$,
$status$ ::= $[state ::= state \in$ State]]

TYPE Element $\triangleq$ [
$path$ ::= $path \in$ STRING,
$value$ ::= $value \in$ STRING,
$index$ ::= $index \in Nat$,
$deleted$ ::= $deleted \in$ BOOLEAN]

TYPE Configuration $\triangleq$ [
id ::= $id \in$ STRING,
revision ::= revision $\in Nat$,
$target$ ::= $target \in$ STRING,
$elements$ ::= $[i \in 1 \,..\, Nat \mapsto elements[i] \in$ Element],
$status$ ::= [
$transactionIndex$ ::= $transactionIndex \in Nat$,
$targetIndex$ ::= $targetIndex \in Nat$,
$mastershipTerm$ ::= $mastershipTerm \in Nat$]]

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

$$
\begin{array}{l}
ValidChanges \triangleq \\
\quad \text{LET } allPaths \triangleq \text{UNION } \{(\text{DOMAIN } Target[t]) : t \in \text{DOMAIN } Target\} \\
\qquad allValues \triangleq \text{UNION } \{\text{UNION } \{Target[t][p] : p \in \text{DOMAIN } Target[t]\} : t \in \text{DOMAIN } Target\} \\
\quad \text{IN} \\
\qquad \{targetPathValues \in \text{SUBSET } (Target \times allPaths \times allValues \times \text{BOOLEAN }) : \\
\qquad\quad \land \forall\, target \in \text{DOMAIN } Target : \\
\qquad\qquad \text{LET } targetIndexes \triangleq \{i \in 1 \,..\, Len(targetPathValues) : \land\, targetPathValues[i][1] = target\} \\
\qquad\qquad \text{IN} \quad \lor\, Cardinality(targetIndexes) = 0 \\
\qquad\qquad\qquad \lor \land\, Cardinality(targetIndexes) = 1 \\
\qquad\qquad\qquad\quad \land \text{LET } targetPathValue \triangleq targetPathValues[\text{CHOOSE } index \in targetIndexes : \text{TRUE}] \\
\qquad\qquad\qquad\qquad \text{IN} \\
\qquad\qquad\qquad\qquad\quad \land\, targetPathValue[2] \setminus (\text{DOMAIN } Target[target]) = \{\} \\
\qquad\qquad\qquad\qquad\quad \land\, targetPathValue[3] \in Target[target][targetPathValue[2]]\}
\end{array}
$$

Add a set of changes to the transaction log

$$
\begin{array}{l}
Change \triangleq \\
\quad \land \exists\, changes \in ValidChanges : \\
\qquad \land\, transactions' = Append(transactions, [index \mapsto Len(transactions) + 1, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad atomic \mapsto \text{FALSE}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad sync \mapsto \text{FALSE}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad changes \mapsto changes, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad status \mapsto [state \mapsto Pending]]) \\
\quad \land \text{UNCHANGED } \langle configurations,\ targets \rangle
\end{array}
$$

This section models the Transaction log reconciler.

$RemoveElement(elements,\ path) \triangleq [i \in \{e \in \text{DOMAIN } elements : elements[e].path \neq path\} \mapsto elements[i]]$

$AddElement(elements,\ element) \triangleq Append(elements,\ element)$

$UpdateElement(elements,\ element) \triangleq AddElement(RemoveElement(elements,\ element.path),\ element)$

$Paths(elements,\ changes) \triangleq \{e.path : e \in elements\} \cup \{c.path : c \in elements\}$

$$
\begin{array}{l}
UpdateElements(elements,\ changes) \triangleq \\
\quad \text{LET } configPaths \triangleq \{e.path : e \in elements\} \\
\qquad configMap \triangleq [path \in configPaths \mapsto \text{CHOOSE } e \in elements : e.path = path] \\
\qquad changePaths \triangleq \{c.path : c \in changes\}
\end{array}
$$

$changeMap \triangleq [path \in changePaths \mapsto \text{CHOOSE } c \in changes : c.path = path]$
$allPaths \triangleq configPaths \cup changePaths$

IN

$Seq(\{\text{IF } path \in \text{DOMAIN } changeMap \text{ THEN } changeMap[path] \text{ ELSE } configMap[path] : path \in allPaths\})$

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.

$\land\ \lor\ \land\ tx.status.state = Pending$
$\quad \land\ \lor\ \land\ tx.index > 1$
$\quad\quad \land\ transactions[tx.index - 1].status.state \in \{Complete,\ Failed\}$
$\quad\quad \lor\ tx.index = 1$
$\quad \land\ transactions' = [transactions \text{ EXCEPT } ![tx.index].status.state = Validating]$
$\quad \land \text{ UNCHANGED } \langle configurations \rangle$

If the transaction is in the *Validating* state, compute and validate the Configuration for each target. Mark the transaction *Failed* if validation fails any target. If validation is successful, proceed to *Applying*.

$\quad \lor\ \land\ tx.status.state = Validating$
$\quad\quad$ *TODO*: Validate the target configurations here
$\quad \land\ transactions' = [transactions \text{ EXCEPT } ![tx.index].status.state = Applying]$
$\quad \land \text{ UNCHANGED } \langle configurations \rangle$

If the transaction is in the *Applying* state, update the Configuration for each target and *Complete* the transaction.

$\quad \lor\ \land\ tx.status.state = Applying$
$\quad\quad \land\ \lor\ \land\ tx.atomic$
$\quad\quad\quad$ *TODO*: Apply atomic transactions here
$\quad\quad\quad \land\ transactions' = [transactions \text{ EXCEPT } ![tx.index].status.state = Complete]$
$\quad\quad\quad \land \text{ UNCHANGED } \langle configurations \rangle$
$\quad\quad \land\ \lor\ \land\ \neg tx.atomic$
$\quad\quad\quad \land\ configurations' = [t \in Target \mapsto [$
$\quad\quad\quad\quad\quad configurations[t] \text{ EXCEPT } !.elements = UpdateElements(configurations[t].$
$\quad\quad\quad\quad\quad\quad !.status.transactionIndex = tx.index]]$
$\quad\quad\quad \land\ transactions' = [transactions \text{ EXCEPT } ![tx.index].status.state = Complete]$
$\land \text{ UNCHANGED } \langle targets \rangle$

This section models the Configuration reconciler.

$ReconcileConfiguration(n,\ c) \triangleq$

Only the master should reconcile the configuration

$\land\ masters[c.target].master = n$

If the configuration's mastership term is less than the current mastership term, assume the target may have restarted/reconnected and perform a full reconciliation of the target configuration from the root path.

$\land \lor \land masters[c.target].term > c.status.mastershipTerm$
TODO: Reconcile the target state here
$\land configurations' = [configurations \text{ EXCEPT } ![c.id].status.mastershipTerm = masters[c.target].term,$
$![c.id].status.targetIndex = c.status.transactionIndex]$

If the Configuration's transaction index is greater than the target index,
reconcile the configuration with the target. Once the target has been updated,
update the target index to match the reconciled transaction index.

$\land \lor \land masters[c.target].term = c.status.mastershipTerm$
$\land c.status.transactionIndex > c.status.targetIndex$
TODO: Reconcile the target state here
$\land configurations' = [configurations \text{ EXCEPT } ![c.id].status.targetIndex = c.status.transactionIndex]$
$\land \text{UNCHANGED } \langle transactions \rangle$

Init and next state predicates

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

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

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

---

* Modification History
* Last modified *Thu Jan* 13 13:56:08 *PST* 2022 by *jordanhalterman*

\ * Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*