
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: $Target \triangleq$ [
 $target1 \mapsto$ [
 $path1 \mapsto \{“value1”, “value2”\}$,
 $path2 \mapsto \{“value2”, “value3”\}$],
 $target2 \mapsto$ [
 $path2 \mapsto \{“value3”, “value4”\}$,
 $path3 \mapsto \{“value4”, “value5”\}$]]

CONSTANT *Target*

ASSUME *Nil* ∈ STRING

ASSUME *TransactionPending* ∈ STRING

ASSUME *TransactionValidating* ∈ STRING

ASSUME *TransactionApplying* ∈ STRING

ASSUME $TransactionComplete \in \text{STRING}$
 ASSUME $TransactionFailed \in \text{STRING}$

 ASSUME $ConfigurationPending \in \text{STRING}$
 ASSUME $ConfigurationInitializing \in \text{STRING}$
 ASSUME $ConfigurationUpdating \in \text{STRING}$
 ASSUME $ConfigurationComplete \in \text{STRING}$
 ASSUME $ConfigurationFailed \in \text{STRING}$

 ASSUME $\wedge IsFiniteSet(Node)$
 $\wedge \forall n \in Node :$
 $\wedge n \notin \text{DOMAIN } Target$
 $\wedge n \in \text{STRING}$

 ASSUME $\wedge \forall t \in \text{DOMAIN } Target :$
 $\wedge IsFiniteSet(Target[t])$
 $\wedge t \notin Node$
 $\wedge t \in \text{STRING}$

```

TYPE TransactionStatus ::= status ∈
  { TransactionPending,
    TransactionValidating,
    TransactionApplying,
    TransactionComplete,
    TransactionFailed }

TYPE Transaction ≜ [
  id      ::= id ∈ STRING,
  index   ::= index ∈ Nat,
  revision ::= revision ∈ Nat,
  atomic  ::= atomic ∈ BOOLEAN ,
  sync    ::= sync ∈ BOOLEAN ,
  changes ::= [ target ∈ SUBSET (DOMAIN Target) ↦ [
    path ∈ SUBSET (DOMAIN Target[target]) ↦ [
      value ::= value ∈ STRING,
      delete ::= delete ∈ BOOLEAN ]],
  status ::= status ∈ TransactionStatus ]

TYPE ConfigurationStatus ::= status ∈
  { ConfigurationPending,
    ConfigurationInitializing,
    ConfigurationUpdating,
    ConfigurationComplete,
    ConfigurationFailed }

TYPE Configuration ≜ [
  id      ::= id ∈ STRING,
  revision ::= revision ∈ Nat,
  target  ::= target ∈ 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 \triangleq

```

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

```

Add a set of changes to the transaction log

Change \triangleq

$\wedge \exists changes \in ValidChanges :$

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

This section models the Transaction log reconciler.

Reconcile the transaction log

$\text{ReconcileTransaction}(n, tx) \triangleq$

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.

$$\begin{aligned}
& \wedge \vee \wedge tx.\text{status} = \text{TransactionPending} \\
& \quad \wedge \vee \wedge tx.\text{index} > 1 \\
& \quad \quad \wedge \text{transactions}[tx.\text{index} - 1].\text{status} \in \{\text{TransactionComplete}, \text{TransactionFailed}\} \\
& \quad \vee tx.\text{index} = 1 \\
& \quad \wedge \text{transactions}' = [\text{transactions} \text{ EXCEPT } ![tx.\text{index}].\text{status} = \text{TransactionValidating}] \\
& \quad \wedge \text{UNCHANGED } \langle \text{configurations} \rangle \\
& \text{If the transaction is in the } \textit{Validating} \text{ state, compute and validate the Configuration for each target.} \\
& \vee \wedge tx.\text{status} = \text{TransactionValidating} \\
& \quad \text{If validation fails any target, mark the transaction } \textit{Failed}. \\
& \quad \text{If validation is successful, proceed to } \textit{Applying}. \\
& \quad \wedge \exists \text{valid} \in \text{BOOLEAN} : \\
& \quad \quad \vee \wedge \text{valid} \\
& \quad \quad \quad \wedge \text{transactions}' = [\text{transactions} \text{ EXCEPT } ![tx.\text{index}].\text{status} = \text{TransactionApplying}] \\
& \quad \quad \vee \wedge \neg \text{valid} \\
& \quad \quad \quad \wedge \text{transactions}' = [\text{transactions} \text{ EXCEPT } ![tx.\text{index}].\text{status} = \text{TransactionFailed}] \\
& \quad \quad \wedge \text{UNCHANGED } \langle \text{configurations} \rangle \\
& \text{If the transaction is in the } \textit{Applying} \text{ state, update the Configuration for each target and } \textit{Complete} \text{ the transaction.} \\
& \vee \wedge tx.\text{status} = \text{TransactionApplying} \\
& \quad \wedge \vee \wedge tx.\text{atomic} \\
& \quad \quad \text{TODO: Apply atomic transactions here} \\
& \quad \quad \wedge \text{transactions}' = [\text{transactions} \text{ EXCEPT } ![tx.\text{index}].\text{status} = \text{TransactionComplete}] \\
& \quad \quad \wedge \text{UNCHANGED } \langle \text{configurations} \rangle \\
& \quad \wedge \vee \wedge \neg tx.\text{atomic} \\
& \quad \quad \text{Add the transaction index to each updated path} \\
& \quad \quad \wedge \text{configurations}' = [\\
& \quad \quad \quad t \in \text{DOMAIN } \textit{Target} \mapsto [\\
& \quad \quad \quad \quad \text{configurations}[t] \text{ EXCEPT} \\
& \quad \quad \quad \quad \quad \text{!.paths} = [\text{path} \in \text{DOMAIN } tx.\text{changes} \mapsto
\end{aligned}$$

$$\begin{aligned}
& tx.changes[path] @@ [index \mapsto tx.index] @@ configurations[t].paths, \\
& !.txIndex = tx.index, \\
& !.status = ConfigurationPending]] \\
& \wedge transactions' = [transactions \text{ EXCEPT } ![tx.index].status = TransactionComplete] \\
& \wedge \text{UNCHANGED } \langle targets \rangle
\end{aligned}$$

This section models the Configuration reconciler.

$ReconcileConfiguration(n, c) \triangleq$

$\wedge \vee \wedge c.status = ConfigurationPending$

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.

$\wedge \vee \wedge masters[c.target].term > c.mastershipTerm$

$\wedge configurations' = [configurations \text{ EXCEPT } ![c.id].status = ConfigurationInitializing, ![c.id].mastershipTerm = masters[c.target].term]$

If the configuration is marked *ConfigurationPending* and the values have changed (determined by comparing the transaction index to the last *sync* index), mark the configuration *ConfigurationUpdating* to push the changes to the target.

$\vee \wedge c.syncIndex < c.txIndex$

$\wedge configurations' = [configurations \text{ EXCEPT } ![c.id].status = ConfigurationUpdating]$

$\vee \wedge c.status = ConfigurationInitializing$

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

Merge the configuration paths with the target paths, removing paths that have been marked deleted

$\wedge targets' = [targets \text{ EXCEPT } ![c.target] =$

$[p \in \{p \in \text{DOMAIN } c.paths : \neg c.paths[p].deleted\} \mapsto [value \mapsto c.paths[p]]] @@$

$[p \in \{p \in \text{DOMAIN } targets[c.target] : \neg c.paths[p].deleted\} \mapsto targets[c.target][p]]]$

Set the configuration's status to *Complete*

$\wedge configurations' = [configurations \text{ EXCEPT } ![c.id].status = ConfigurationComplete, ![c.id].syncIndex = c.txIndex]$

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*

$\vee \wedge c.status = ConfigurationUpdating$

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

Compute the set of updated and deleted paths by comparing their indexes to the target's last *sync* index.

$\wedge \text{LET } updatedPaths \triangleq \{p \in \text{DOMAIN } c.paths : c.paths[p].index > c.syncIndex\}$
 $deletedPaths \triangleq \{p \in updatedPaths : c.paths[p].deleted\}$

IN

Update the target paths by adding/updating paths that have changed and removing paths that have been deleted since the last *sync*.

$$\begin{aligned} \wedge targets' &= [targets \text{ EXCEPT } ![c.target] = \\ &\quad [p \in updatedPaths \setminus deletedPaths \mapsto c.paths[p]] @@ \\ &\quad [p \in \text{DOMAIN } targets[c.target] \setminus deletedPaths \mapsto targets[c.target][p]]] \\ \wedge configurations' &= [configurations \text{ EXCEPT } ![c.id].status = ConfigurationComplete, \\ &\quad ![c.id].syncIndex = c.txIndex] \end{aligned}$$

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

$$\begin{aligned} \vee \wedge c.status &\in \{ConfigurationComplete, ConfigurationFailed\} \\ \wedge masters[c.target].master &= Nil \\ \wedge configurations' &= [configurations \text{ EXCEPT } ![c.id].status = ConfigurationPending] \\ \wedge \text{UNCHANGED } \langle transactions \rangle \end{aligned}$$

Init and next state predicates

$$\begin{aligned} Init &\triangleq \\ &\wedge transactions = \langle \rangle \\ &\wedge configurations = [t \in Target \mapsto [\\ &\quad id \mapsto t, \\ &\quad config \mapsto [path \in \{\} \mapsto [\\ &\quad \quad path \mapsto path, \\ &\quad \quad value \mapsto Nil, \\ &\quad \quad index \mapsto 0, \\ &\quad \quad deleted \mapsto FALSE]]]] \\ &\wedge targets = [t \in Target \mapsto [\\ &\quad path \in \{\} \mapsto [\\ &\quad \quad value \mapsto Nil]]] \\ &\wedge masters = [t \in Target \mapsto [master \mapsto Nil, term \mapsto 0]] \\ Next &\triangleq \\ &\vee Change \\ &\vee \exists n \in Node : \\ &\quad \exists t \in \text{DOMAIN } transactions : \\ &\quad \quad ReconcileTransaction(n, t) \\ &\vee \exists n \in Node : \\ &\quad \exists c \in configurations : \\ &\quad \quad ReconcileConfiguration(n, c) \\ Spec &\triangleq Init \wedge \Box [Next]_{vars} \end{aligned}$$

\ * Modification History

* Last modified *Thu Jan 13 23:17:03 PST 2022* by *jordanhalterman*
* Created *Wed Sep 22 13:22:32 PDT 2021* by *jordanhalterman*