MODULE *t2pc*

```
EXTENDS Integers, Sequences, FiniteSets, TLC
CONSTANT RM,                The set of participating resource managers, RM = 1 .. 3
         RMMAYFAIL,
         TMMAYFAIL,
         ENABLEBTM          Flag to enable the backup TM configuration

 ************************************************************************
A modified version of P2TCommit at http://lamport.azurewebsites.net/tla/two-phase.html
Transaction manager (TM) is added.
--algorithm TransactionCommit{
  variable rmState = [rm ∈ RM ↦ "working"] ;

            The state of the transation manager
           tmState = "init" ;
            Initial state of the backup TM set to inactive
           btmState = "inactive" ;

  define {
     can commit when only all RM are either in prepared state or have committed and the TM state is not in abort state
    canCommit ≜ ∀ rm ∈ RM : rmState[rm] ∈ {"prepared", "committed", "failed"} ∧ btmState ≠ "abort" ∧ tı
     can abort when only all RM are not in committed state and the TM state is not committed
    canAbort ≜ (∀ rm ∈ RM : rmState[rm] ≠ "committed" ∧ (tmState ≠ "commit" ∧ btmState ≠ "commit"))
    }

  macro Prepare( p ) {
       ignore if already in prepared state
     if ( rmState[p] ≠ "prepared" ) {
          await rmState[p] = "working" ;
          rmState[p] := "prepared" ;
      }
    }


  macro Fail( p ) {
     If RMMAYFAIL is True then the state of the RM may become failed
    if ( RMMAYFAIL ) {
          either {
                    rmState[p] := "failed"
                    }
          or skip ;
     } ;
   }

  macro Decide( p ) {
    each RM can access its own state and the TM state or Backup TM state
    Each RM will consult the Backup TM only when the tmState is hidden and Backup TM is enabled
```

```
  either { when  ∧ rmState[p] = "prepared"
                 ∧ (tmState = "commit" ∨ (tmState = "hidden" ∧ ENABLEBTM ∧ btmState = "commit"))
           rmState[p] := "committed"
         }
         RM can abort if its in working or prepared state provided TM sends abort
  or     { when  ∧ rmState[p] ∈ {"working", "prepared"}
                 ∧ (tmState = "abort" ∨ (tmState = "hidden" ∧ ENABLEBTM ∧ btmState = "abort"));
           rmState[p] := "aborted"
         }
         RM can spontaneously decide to abort if its in working state
  or     { when  ∧ rmState[p] ∈ {"working"};
           rmState[p] := "aborted"
         }
  }

fair process ( RManager ∈ RM ) {
   an RM can do any of the following when in working and prepared state
  RS: while ( rmState[self] ∈ {"working", "prepared"} ) {
    either Prepare(self) or Decide(self) or Fail(self) }

  }

 Process for Transaction Manager
fair process ( TManager = 0 ) {
   TMANAGER is operational process only as long as the tmState is not hidden
  TS:  if ( tmState ≠ "hidden" ) {
         either { await canCommit;
                  TC: tmState := "commit";
                  F1: if ( TMMAYFAIL ) {
                        record the active tmState in backup TM
                        btmState := tmState;
                        tmState := "hidden";

                      } if
                } either
           or   {
                  await canAbort;
                  TA: tmState := "abort";
                  F2: if ( TMMAYFAIL ) {
                        record the active tmState in backup TM
                        btmState := tmState;
                        tmState := "hidden";
                      } if
                } or
    } if
  }
```

Process for the Backup Transaction Manager

fair process ($BTManager = 10$) **{**

Backup TM becomes active only when TM state is hidden and $ENABLEBTM$ flag is set to true

BTS: **if** ($ENABLEBTM \land tmState =$ "hidden") **{**

either {

await $canCommit$ **;**

BTC: $btmState :=$ "commit" **;**

} either

or {

await $canAbort$ **;**

BTA: $btmState :=$ "abort" **;**

} or

} if

}

}

.'

Below is the algorithm's translation. The translation defines Termination to be the temporal formula asserting that eventually all processes terminate.

****

BEGIN TRANSLATION

VARIABLES $rmState$, $tmState$, $btmState$, pc

define statement

$canCommit \triangleq \forall\, rm \in RM : rmState[rm] \in \{$"prepared", "committed", "failed"$\} \land btmState \neq$ "abort" $\land\ tmSt$

$canAbort \triangleq (\forall\, rm \in RM : rmState[rm] \neq$ "committed" $\land (tmState \neq$ "commit" $\land\ btmState \neq$ "commit"$))$

$vars \triangleq \langle rmState,\ tmState,\ btmState,\ pc \rangle$

$ProcSet \triangleq (RM) \cup \{0\} \cup \{10\}$

$Init \triangleq$ Global variables

$\land\ rmState = [rm \in RM \mapsto$ "working"$]$

$\land\ tmState =$ "init"

$\land\ btmState =$ "inactive"

$\land\ pc = [self \in ProcSet \mapsto$ CASE $self \in RM \to$ "RS"

$\square\ \ self = 0 \to$ "TS"

$\square\ \ self = 10 \to$ "BTS"$]$

$RS(self) \triangleq\ \land\ pc[self] =$ "RS"

$\land$ IF $rmState[self] \in \{$"working", "prepared"$\}$

THEN $\land\ \lor\ \land$ IF $rmState[self] \neq$ "prepared"

THEN $\land\ rmState[self] =$ "working"

$\land\ rmState' = [rmState$ EXCEPT $![self] =$ "prepared"$]$

```
                                ELSE ∧ TRUE
                                     ∧ UNCHANGED rmState
                          ∨ ∧ ∨ ∧ ∧ rmState[self] = "prepared"
                                  ∧ (tmState = "commit" ∨ (tmState = "hidden" ∧ ENABLEBTM ∧ btmS
                                ∧ rmState' = [rmState EXCEPT ![self] = "committed"]
                            ∨ ∧ ∧ rmState[self] ∈ {"working", "prepared"}
                                  ∧ (tmState = "abort" ∨ (tmState = "hidden" ∧ ENABLEBTM ∧ btmSta
                                ∧ rmState' = [rmState EXCEPT ![self] = "aborted"]
                            ∨ ∧ ∧ rmState[self] ∈ {"working"}
                                ∧ rmState' = [rmState EXCEPT ![self] = "aborted"]
                          ∨ ∧ IF RMMAYFAIL
                                THEN ∧ ∨ ∧ rmState' = [rmState EXCEPT ![self] = "failed"]
                                       ∨ ∧ TRUE
                                         ∧ UNCHANGED rmState
                                ELSE ∧ TRUE
                                     ∧ UNCHANGED rmState
                               ∧ pc' = [pc EXCEPT ![self] = "RS"]
                        ELSE ∧ pc' = [pc EXCEPT ![self] = "Done"]
                               ∧ UNCHANGED rmState
                    ∧ UNCHANGED ⟨tmState, btmState⟩

RManager(self) ≜ RS(self)

TS ≜ ∧ pc[0] = "TS"
     ∧ IF tmState ≠ "hidden"
         THEN ∧ ∨ ∧ canCommit
                  ∧ pc' = [pc EXCEPT ![0] = "TC"]
                ∨ ∧ canAbort
                  ∧ pc' = [pc EXCEPT ![0] = "TA"]
         ELSE ∧ pc' = [pc EXCEPT ![0] = "Done"]
     ∧ UNCHANGED ⟨rmState, tmState, btmState⟩

TC ≜ ∧ pc[0] = "TC"
     ∧ tmState' = "commit"
     ∧ pc' = [pc EXCEPT ![0] = "F1"]
     ∧ UNCHANGED ⟨rmState, btmState⟩

F1 ≜ ∧ pc[0] = "F1"
     ∧ IF TMMAYFAIL
         THEN ∧ btmState' = tmState
              ∧ tmState' = "hidden"
         ELSE ∧ TRUE
              ∧ UNCHANGED ⟨tmState, btmState⟩
     ∧ pc' = [pc EXCEPT ![0] = "Done"]
     ∧ UNCHANGED rmState
```

$$
\begin{array}{l}
TA \triangleq \land\ pc[0] = \text{“TA”} \\
\qquad \land\ tmState' = \text{“abort”} \\
\qquad \land\ pc' = [pc \text{ EXCEPT } ![0] = \text{“F2”}] \\
\qquad \land \text{ UNCHANGED } \langle rmState,\ btmState \rangle
\end{array}
$$

$$
\begin{array}{l}
F2 \triangleq \land\ pc[0] = \text{“F2”} \\
\qquad \land \text{ IF } TMMAYFAIL \\
\qquad\qquad \text{THEN } \land\ btmState' = tmState \\
\qquad\qquad\qquad\quad \land\ tmState' = \text{“hidden”} \\
\qquad\qquad \text{ELSE } \land \text{ TRUE} \\
\qquad\qquad\qquad\quad \land \text{ UNCHANGED } \langle tmState,\ btmState \rangle \\
\qquad \land\ pc' = [pc \text{ EXCEPT } ![0] = \text{“Done”}] \\
\qquad \land \text{ UNCHANGED } rmState
\end{array}
$$

$$TManager \triangleq TS \lor TC \lor F1 \lor TA \lor F2$$

$$
\begin{array}{l}
BTS \triangleq \land\ pc[10] = \text{“BTS”} \\
\qquad \land \text{ IF } ENABLEBTM \land tmState = \text{“hidden”} \\
\qquad\qquad \text{THEN } \land\ \lor\ \land\ canCommit \\
\qquad\qquad\qquad\qquad\qquad\quad \land\ pc' = [pc \text{ EXCEPT } ![10] = \text{“BTC”}] \\
\qquad\qquad\qquad\qquad\quad \lor\ \land\ canAbort \\
\qquad\qquad\qquad\qquad\qquad\quad \land\ pc' = [pc \text{ EXCEPT } ![10] = \text{“BTA”}] \\
\qquad\qquad \text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![10] = \text{“Done”}] \\
\qquad \land \text{ UNCHANGED } \langle rmState,\ tmState,\ btmState \rangle
\end{array}
$$

$$
\begin{array}{l}
BTC \triangleq \land\ pc[10] = \text{“BTC”} \\
\qquad \land\ btmState' = \text{“commit”} \\
\qquad \land\ pc' = [pc \text{ EXCEPT } ![10] = \text{“Done”}] \\
\qquad \land \text{ UNCHANGED } \langle rmState,\ tmState \rangle
\end{array}
$$

$$
\begin{array}{l}
BTA \triangleq \land\ pc[10] = \text{“BTA”} \\
\qquad \land\ btmState' = \text{“abort”} \\
\qquad \land\ pc' = [pc \text{ EXCEPT } ![10] = \text{“Done”}] \\
\qquad \land \text{ UNCHANGED } \langle rmState,\ tmState \rangle
\end{array}
$$

$$BTManager \triangleq BTS \lor BTC \lor BTA$$

$$
\begin{array}{l}
Next \triangleq\ TManager \quad \lor BTManager \\
\qquad\quad \lor\ (\exists\, self \in RM : RManager(self)) \\
\qquad\quad \lor\ \text{Disjunct to prevent deadlock on termination} \\
\qquad\qquad ((\forall\, self \in ProcSet : pc[self] = \text{“Done”}) \land \text{ UNCHANGED } vars)
\end{array}
$$

$$
\begin{array}{l}
Spec \triangleq \land\ Init \land \Box[Next]_{vars} \\
\qquad \land\ \forall\, self \in RM : \mathrm{WF}_{vars}(RManager(self)) \\
\qquad \land\ \mathrm{WF}_{vars}(TManager) \\
\qquad \land\ \mathrm{WF}_{vars}(BTManager)
\end{array}
$$

For termination, all RM which have not failed should be in either 'aborted' or

'committed' state, TM or BTM state should be matching the RM states
all process's program counter should be at 'Done'

$$\begin{aligned} Termination \triangleq \Diamond((& \\ & (\forall rm \in RM : rmState[rm] \neq \text{“failed”} \Rightarrow rmState[rm] = \text{“committed”} \wedge (tmState = \text{“commi} \\ & \vee (\forall rm \in RM : rmState[rm] \neq \text{“failed”} \Rightarrow rmState[rm] = \text{“aborted”} \wedge (tmState = \text{“abort”} \vee \\ & \wedge (\forall self \in ProcSet : pc[self] = \text{“Done”})) \end{aligned}$$

END TRANSLATION

The invariants:

$TypeOK \triangleq$

The type-correctness invariant

$$\begin{aligned} & \wedge rmState \in [RM \to \{\text{“working”}, \text{“prepared”}, \text{“committed”}, \text{“aborted”}, \text{“failed”}\}] \\ & \wedge tmState \in \{\text{“init”}, \text{“hidden”}, \text{“commit”}, \text{“abort”}\} \\ & \wedge btmState \in \{\text{“inactive”}, \text{“commit”}, \text{“abort”}\} \end{aligned}$$

$Consistent \triangleq$

A state predicate asserting that two RMs have not arrived at conflicting decisions.

$$\begin{aligned} \forall rm1, rm2 \in RM : \neg & \wedge rmState[rm1] = \text{“aborted”} \\ & \wedge rmState[rm2] = \text{“committed”} \end{aligned}$$

THEOREM $Spec \Rightarrow \Box(TypeOK \wedge Consistent)$

* Modification History
* Last modified *Tue Dec* 05 19:16:48 *EST* 2017 by *varunjai*
* Last modified *Tue Oct* 11 08:14:15 *PDT* 2011 by *lamport*
* Created *Mon Oct* 10 05:31:02 *PDT* 2011 by *lamport* Members:

Name: *Sneha Mehta UBIT* Name: snehameh Person# - 50245877

Name: *Varun Jain UBIT* Name: *varunjai* Person# - 50247176

Explaination

This program emulates 2 phase commit under the problem conditions

TypeOK Property: The states for all resource manager, transaction manager and backup transaction manager are defined.

Termination Property: - All the Resource managers which are not failed are either in the committed state or the
aborted state
- The transaction manager state is also set accordingly to commit or abort state accordingly
- In case the transaction manager is 'hidden' then backup transaction manager state should be set to commit or abort accordingly.

Consistency Property - No two Resource managers can be in conflicting states like - aborted and committed.

1.1 $RMMAYFAIL$ = FALSE and $TMMAYFAIL$ = FALSE

No errors were observed as no failures occurred and the 2 phase commit protocol was neatly executed satisfying CONSISTENCY and *Termination* properties.

$RMMAYFAIL$ = TRUE and $TMMAYFAIL$ = FALSE

No errors were observed as though some resource manager failures occurred, the 2 phase commit protocol was neatly executed satisfying CONSISTENCY and *Termination* properties.

1.2 $RMMAYFAIL$ = FALSE and $TMMAYFAIL$ = TRUE, no backup TM Model: 4 RM nodes, 1 TM

Observations: 1. Model fails with temporal property or termination property violation in this case.

$\wedge\ pc =\ (0 :>$ "Done" @@ $1 :>$ "RS" @@ $2 :>$ "Done" @@ $3 :>$ "Done" @@ $4 :>$ "Done"$)$
$\wedge\ rmState = \langle$"prepared", "aborted", "aborted", "aborted"$\rangle$
$\wedge\ tmState =$ "hidden"

2. The model failed because once the TM failed during the run, Resource managers which decided to go to 'prepared' state could no longer terminate as they were unable to access TM state and hence could not move to states 'committed' or 'aborted'.

1.3 When both $RMMAYFAIL$ = TRUE and $TMMAYFAIL$ = TRUE, and $ENABLEBTM$ = TRUE (the Backup TM is enabled) Model: 4 RM nodes, 1 TM, 1 BTM (backup TM)

Observations: 1. Model passes without any failures. The program satisfies both TERMINATION and CONSISTENCY.

This is because on failure of the TM, the BTM becomes active and it is assumed that the BTM does not fail. The resource managers are now being serviced by the BTM similar to the TM.