MODULE *t2pc*

```
EXTENDS Integers, Sequences, FiniteSets, TLC
CONSTANTS RM,                         The number of resource managers.
          BTM,                        Whether have backupTM.
          RMMAYFAIL,                  Whether RM could fail.
          TMMAYFAIL                   Whether TM could fail.
```

```
--algorithm TransactionCommit{
  variable rmState = [rm ∈ RM ↦ "working"],
           tmState = "init" ;                                     tmState's init state.
           btmState = "init" ;                                    backupTM's init state.
  define {
    canCommit ≜ ∀ rm ∈ RM : rmState[rm] ∈ {"prepared", "committed"}   If some rm are commited or all rm are
                                                                  which means tmState is "commit", sowe
    canAbort ≜ ∀ rm ∈ RM : rmState[rm] ≠ "committed"   if no rm are committed, we don't know the state of tmSt
                                                       if tmState is not "commit", we cannot commit.
     }

  macro Prepare( p ) {
    await rmState[p] = "working" ;     if rmState[p] is working, then it will be prepared
      rmState[p] := "prepared" ;
     }

  macro Decide( p ) {
    either { await rmState[p] = "prepared" ∧ canCommit ∧ (tmState = "commit" ∨ btmState = "commit") ;


               rmState[p] := "committed" ;
             }
    or     { await rmState[p] ∈ {"working", "prepared"} ∧ canAbort ;                     If r
               rmState[p] := "abort"
             }
   }

  macro Fail( p ) {
     if ( RMMAYFAIL )  rmState[p] := "crash"                        If RMMAYFAIL, rmState[p] could b
   }

  fair  process ( RManager ∈ RM ) {                                  If rmState is working or prepare
    RS: while ( rmState[self] ∈ {"working", "prepared"} ) {          set up backupTM. Otherwise ter
          either Prepare(self) or Decide(self) or Fail(self) }
   }

  fair process ( TManager = 0 ) {                                     If all rm are prepared, it's ti
    TS: either { await canCommit ;
```

```
            $TC$:  $tmState$ := "commit" ;
                   **if** ( $BTM$ ) $btmState$ := "commit" ;                    If we set $backupTM$, change
            $F1$:  **if** ( $TMMAYFAIL$ ) $tmState$ := "hidden" ;  }

        **or** { **await** $canAbort$ ;                                          Abort can appear any time u
            $TA$: $tmState$ := "abort" ;
                   **if** ( $BTM$ ) $btmState$ := "abort" ;                     If we set $backupTM$, change
            $F2$: **if** ( $TMMAYFAIL$ ) $tmState$ := "hidden" ;  }
    }
}
```

BEGIN TRANSLATION

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

define statement

$canCommit \triangleq \forall rm \in RM : rmState[rm] \in \{\text{"prepared"}, \text{"committed"}\}$

$canAbort \triangleq \forall rm \in RM : rmState[rm] \neq \text{"committed"}$

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

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

```
$Init$ ≜  Global variables
        $\land rmState = [rm \in RM \mapsto \text{"working"}]$
        $\land tmState = \text{"init"}$
        $\land btmState = \text{"init"}$
        $\land pc = [self \in ProcSet \mapsto$ CASE $self \in RM \rightarrow \text{"RS"}$
                                          $\Box$  $self = 0 \rightarrow \text{"TS"}]$

$RS(self)$ ≜  $\land pc[self] = \text{"RS"}$
             $\land$ IF $rmState[self] \in \{\text{"working"}, \text{"prepared"}\}$
                   THEN $\land \lor \land rmState[self] = \text{"working"}$
                                 $\land rmState' = [rmState$ EXCEPT $![self] = \text{"prepared"}]$
                           $\lor \land \lor \land rmState[self] = \text{"prepared"} \land canCommit \land (tmState = \text{"commit"} \lor btm$
                                    $\land rmState' = [rmState$ EXCEPT $![self] = \text{"committed"}]$
                                 $\lor \land rmState[self] \in \{\text{"working"}, \text{"prepared"}\} \land canAbort$
                                    $\land rmState' = [rmState$ EXCEPT $![self] = \text{"abort"}]$
                           $\lor \land$ IF $RMMAYFAIL$
                                   THEN $\land rmState' = [rmState$ EXCEPT $![self] = \text{"crash"}]$
                                   ELSE $\land$ TRUE
                                        $\land$ UNCHANGED $rmState$
                        $\land pc' = [pc$ EXCEPT $![self] = \text{"RS"}]$
                   ELSE $\land pc' = [pc$ EXCEPT $![self] = \text{"Done"}]$
                        $\land$ UNCHANGED $rmState$
             $\land$ UNCHANGED $\langle tmState, btmState \rangle$
```

$RManager(self) \triangleq RS(self)$

```
$TS \triangleq \land pc[0] = \text{“TS”}$
        $\land \lor \land canCommit$
                $\land pc' = [pc \text{ EXCEPT } ![0] = \text{“TC”}]$
            $\lor \land canAbort$
                $\land pc' = [pc \text{ EXCEPT } ![0] = \text{“TA”}]$
        $\land \text{UNCHANGED } \langle rmState, tmState, btmState \rangle$

$TC \triangleq \land pc[0] = \text{“TC”}$
        $\land tmState' = \text{“commit”}$
        $\land \text{IF } BTM$
                $\text{THEN } \land btmState' = \text{“commit”}$
                $\text{ELSE } \land \text{TRUE}$
                        $\land \text{UNCHANGED } btmState$
        $\land pc' = [pc \text{ EXCEPT } ![0] = \text{“F1”}]$
        $\land \text{UNCHANGED } rmState$

$F1 \triangleq \land pc[0] = \text{“F1”}$
        $\land \text{IF } TMMAYFAIL$
                $\text{THEN } \land tmState' = \text{“hidden”}$
                $\text{ELSE } \land \text{TRUE}$
                        $\land \text{UNCHANGED } tmState$
        $\land pc' = [pc \text{ EXCEPT } ![0] = \text{“Done”}]$
        $\land \text{UNCHANGED } \langle rmState, btmState \rangle$

$TA \triangleq \land pc[0] = \text{“TA”}$
        $\land tmState' = \text{“abort”}$
        $\land \text{IF } BTM$
                $\text{THEN } \land btmState' = \text{“abort”}$
                $\text{ELSE } \land \text{TRUE}$
                        $\land \text{UNCHANGED } btmState$
        $\land pc' = [pc \text{ EXCEPT } ![0] = \text{“F2”}]$
        $\land \text{UNCHANGED } rmState$

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

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

$Next \triangleq TManager$
            $\lor (\exists self \in RM : RManager(self))$
            $\lor$ Disjunct to prevent deadlock on termination
              $((\forall self \in ProcSet : pc[self] = \text{“Done”}) \land \text{UNCHANGED } vars)$
```

$$Spec \triangleq \land Init \land \Box[Next]_{vars}$$
$$\quad\quad \land \forall self \in RM : \mathrm{WF}_{vars}(RManager(self))$$
$$\quad\quad \land \mathrm{WF}_{vars}(TManager)$$

$$Termination \triangleq \Diamond(\forall self \in ProcSet : pc[self] = \text{“Done”})$$

END TRANSLATION

$$consistency \triangleq tmState = \text{“commit”} \Rightarrow \forall i \in RM : rmState[i] \neq \text{“abort”}$$
$$\quad \land \quad tmState = \text{“abort”} \Rightarrow \forall j \in RM : rmState[j] \neq \text{“committed”}$$
$$\quad \land \quad tmState = \text{“hidden”} \Rightarrow \forall k \in RM : rmState[k] \neq \text{“committed”}$$
$$terminate \triangleq \Diamond(\forall i \in RM : rmState[i] \in \{\text{“committed”}, \text{“abort”}, \text{“crash”}\})$$

*1.2 *TMMAYFAIL* is true and *RMMAYFAIL* is false means *tmState* could be “hidden” and *rmState* cannot be
*hidden. In this situation, termination will be violated. For example, when *TM* is “commit” and some
**RM* are committed, then *TM* crashed while some other *RM* is prepared, but they can never be “commit” or *abort*
*because *TM* is “hidden” now. That’s why we get result when *RM* equals 3 that ⟨“committed”, “prepared”, “committed”⟩.
*It will never been terminated because “prepared” has no way to “commit”.

*1.3 *Termination* and comsistency remain true. The states cancommit and canabort is owned by both *BTM* and *TM*.
* So when *TM* crashes, the *RMs* can still consult the *BTM* and make their decision.
*If an *RM* crashed, then all other *RMs* can only abort. So all other uncrashed *RMs* remain consistent.

* Modification History
* Last modified *Tue Dec* 05 19:55:47 *EST* 2017 by *lenovo*
* Created *Wed Nov* 29 17:13:20 *EST* 2017 by *lenovo*

*Group Members *xhu*7:*xhu*7@*buffalo.edu*
**Haowei Zhou* haoweizh@*buffalo.edu*