CSE 4/586: Project 2

Name:

Due Date *[2017-12-04 Mon]*

1 2-phase commit protocol

In a distributed system, a transaction is performed by a collection of processes called resource managers (RMs), each executing on a different node. The transaction ends when the transaction manager (TM) issues a request either to commit or to abort the transaction. For the transaction to be committed, each participating RM must be willing to commit it. Otherwise, the transaction must be aborted. Prior to the commit request, any RM may spontaneously decide to abort its part of the transaction. The fundamental requirement is that all RMs must eventually agree on whether the transaction is committed or aborted. (`https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2003-96.pdf`)

Below find a model of 2-phase commit. (This was built by modifying the *P2TCommit.tla* at `http://lamport.azurewebsites.net/tla/two-phase.html` to add a TM agent). We will start with that code as the basis for what comes next.

If no faults occur, the 2-phase commit algorithm is correct. In the presence of a crash fault, however, problems can arise. In the questions below, we will use TLA+/PlusCal to explore what problems may arise, and how to properly design the protocol to overcome those problems.

1.1 Fill in the redacted code and model-check with no failures (50 points)

- Fill in the reducted PlusCal code. (The *macro Fail* models RM failure.)
- Add *Consistency* and *Termination* properties.

- Model check *Consistency* and *Termination* with no failures (RMMAYFAIL=FALSE and TMMAYFAIL=FALSE). You should see no errors.
- Model check with RM failure (RMMAYFAIL=TRUE and TMMAYFAIL=FALSE). You should see no errors.

1.2 Model check with only TM failure. (15 points)

- Model check with RMMAYFAIL=FALSE and TMMAYFAIL=TRUE.
- Write in the comments section, after the "`================`" line, your findings/observations. Comment whether the *Termination* property is violated by a TM failure.

1.3 Add a backup TM process to take over if primary crashes (35 points)

- Test satisfaction of *Consistency* and *Termination* properties with no TM or RM failures. Make sure BackupTM terminates, so the *Termination* property is also satisfied as well as *Consistency* property.
- Now, model check with both TM and RM failure allowed (RMMAYFAIL=TRUE and TMMAYFAIL=TRUE). Write down your observations.

2 Submission

Your TLA+ file should be named *t2pc.tla*. Your model's name should be the default name $Model\_1$ (do not name your model file differently). Generate a pdf print of your TLA+ program using the "Produce Pdf version" from the TLA+ menu. (This will get included in your submission as it is created under the ".toolbox" directory.)

Now create a zip file from the ".tla" file and the corresponding ".toolbox" directory. **Name the zipfile as: proj2.zip**

You will use the submit command ($submit\_cse486$ or $submit\_cse586$ respectively) to submit your work. The submit command instructions are here: `http://wiki.cse.buffalo.edu/services/content/submit-script`

```
1  ---------- MODULE 2PCwithBTM ----------
2  EXTENDS Integers, Sequences, FiniteSets, TLC
3  CONSTANT RM,          The set of participating resource managers RM = 1 .. 3
4           RMMAYFAIL,
5           TMMAYFAIL    Whether TM may fail MAYFAIL = TRUE or FALSE
   ****************************************************************************
   A modified version of P2TCommit at http://lamport.azurewebsites.net/tla/two-phase.html
   Transaction manager (TM) is added.
11 --algorithm TransactionCommit{
12   variable rmState = [rm ∈ RM ↦ "working"],
13            tmState = "init" ;
14   define {
15     canCommit ≜ [redacted]
16     canAbort ≜ [redacted]
17
18     }
19   macro Prepare( p ) {
20     await rmState[p] = "working" ;
21     rmState[p] := "prepared" ;
22     }
24   macro Decide( p ) {
25     [redacted]
28     ;
29
30     }
32   macro Fail( p ) {
33     [redacted]
34
36   fair process ( RManager ∈ RM ) {
37    RS: while ( rmState[self] ∈ {"working", "prepared"} ) {
38          either Prepare(self) or Decide(self) or Fail(self) }
39     }
41   fair process ( TManager = 0 ) {
42  TS: either { await canCommit ;
43           TC:  tmState := "commit" ;
44           F1:  if ( TMMAYFAIL ) tmState := "hidden" ; }   tm state becomes inaccessible
46       or { await canAbort ;
47         TA: tmState := "abort" ;
48         F2: if ( TMMAYFAIL ) tmState := "hidden" ; }   tm state becomes inaccessible again
49     }
```