

---

MODULE *TwoPhase*

---

This specification is based on “Two-Phase *Commit*”, Lecture 6 of the TLA+ Video Course. It describes the Two-Phase *Commit* protocol, in which a transaction manager (*TM*) coordinates the resource managers (*RM*s) to implement the Transaction *Commit* specification of module *TCommit*. In this specification, *RM*s spontaneously issue *Prepared* messages. We ignore the Prepare messages that the *TM* can send to the *RM*s.

For simplicity, we also eliminate *Abort* messages sent by an *RM* when it decides to abort. Such a message would cause the *TM* to abort the transaction, an event represented here by the *TM* spontaneously deciding to abort.

CONSTANT *RM*    The set of all Resource Managers (e.g. {“*r1*”, “*r2*”, “*r3*”}).

VARIABLES

*rmState*,    *rmState*[*rm*] is the state of the Resource Manager *rm*.  
*tmState*,    Transaction Manager state: “init” or “done”.  
*tmPrepared*,    The set of resource managers the transaction manager knows are prepared.  
*msgs*

In the protocol, processes communicate with one another by sending messages. For simplicity, we represent message passing with the variable *msgs* whose value is the set of all messages that have been sent. A message is sent by adding it to the set *msgs*. An action that, in an implementation, would be enabled by the receipt of a certain message is here enabled by the presence of that message in *msgs*. For simplicity, messages are never removed from *msgs*. This allows a single message to be received by multiple receivers. Receipt of the same message twice is therefore allowed; but in this particular protocol, that’s not a problem.

The set of all possible messages. Messages of type “Prepared” are sent from the *RM* indicated by the message’s *rm* field to the *TM*. Messages of type “Commit” and “Abort” are broadcast by the *TM*, to be received by all *RM*s. The set *msgs* contains just a single copy of such message.

*Messages*     $\triangleq [type : \{ \text{“Prepared”} \}, rm : RM] \cup [type : \{ \text{“Commit”}, \text{“Abort”} \}]$

*TPTypeOK*     $\triangleq \wedge rmState \in [RM \rightarrow \{ \text{“working”}, \text{“prepared”}, \text{“committed”}, \text{“aborted”} \}]$   
 $\wedge tmState \in \{ \text{“init”}, \text{“done”} \}$   
 $\wedge tmPrepared \subseteq RM$   
 $\wedge msgs \subseteq Messages$

*TPInit*     $\triangleq \wedge rmState = [r \in RM \mapsto \text{“working”}]$   
 $\wedge tmState = \text{“init”}$   
 $\wedge tmPrepared = \{ \}$   
 $\wedge msgs = \{ \}$

---

*TMRecvPrepared*(*r*)     $\triangleq$     The *TM* receives a “Prepared” message from resource manager *r*.  
 $\wedge tmState = \text{“init”}$   
 $\wedge [type \mapsto \text{“Prepared”}, rm \mapsto r] \in msgs$     The message received exists  
 $\wedge tmPrepared' = tmPrepared \cup \{r\}$   
 $\wedge \text{UNCHANGED } \langle rmState, tmState, msgs \rangle$

$TMCommit \triangleq$  The *TM* commits the transaction  
 $\wedge tmState = \text{"init"}$   
 $\wedge tmState' = \text{"done"}$  *TM* state transitions  
 $\wedge tmPrepared = RM$  *TM* sees all *RM*s as prepared  
 $\wedge rmState = [r \in RM \mapsto \text{"prepared"}]$  All *RM*s are prepared (neecessary?)  
 $\wedge msgs' = msgs \cup \{[type \mapsto \text{"Commit"}]\}$  Send the *Commit* message  
 $\wedge \text{UNCHANGED } \langle rmState, tmPrepared \rangle$

$TMAbort \triangleq$  The *TM* spontaneously aborts the transaction  
 $\wedge tmState = \text{"init"} \wedge tmState' = \text{"done"}$  *TM* state transitions  
 $\wedge msgs' = msgs \cup \{[type \mapsto \text{"Abort"}]\}$  Send the *Abort* message  
 $\wedge \text{UNCHANGED } \langle rmState, tmPrepared \rangle$

---

$RMPrepare(r) \triangleq$   $\wedge rmState[r] = \text{"working"}$   
 $\wedge rmState' = [rmState \text{ EXCEPT } ![r] = \text{"prepared"}]$   
 $\wedge msgs' = msgs \cup \{[type \mapsto \text{"Prepared"}, rm \mapsto r]\}$   
 $\wedge \text{UNCHANGED } \langle tmState, tmPrepared \rangle$

*RM* spontaneously chooses to abort. No message sent in our simplified model.

$RMChooseToAbort(r) \triangleq$   $\wedge rmState[r] = \text{"aborted"}$   
 $\wedge rmState' = [rmState \text{ EXCEPT } ![r] = \text{"aborted"}]$   
 $\wedge \text{UNCHANGED } \langle tmState, tmPrepared, msgs \rangle$

$RMRecvCommitMsg(r) \triangleq$  *RM* *r* is told to commit  
 $\wedge [type \mapsto \text{"Commit"}] \in msgs$  A *Commit* message should have been sent  
 $\wedge rmState' = [rmState \text{ EXCEPT } ![r] = \text{"committed"}]$   
 $\wedge \text{UNCHANGED } \langle tmState, tmPrepared, msgs \rangle$

$RMRecvAbortMsg(r) \triangleq$  *RM* is told to abort  
 $\wedge [type \mapsto \text{"Abort"}] \in msgs$  An *Abort* message should have been sent  
 $\wedge rmState' = [rmState \text{ EXCEPT } ![r] = \text{"aborted"}]$   
 $\wedge \text{UNCHANGED } \langle tmState, tmPrepared, msgs \rangle$

$TPNext \triangleq$   $\vee TMCommit$  The transaction is committed by the *TM*.  
 $\vee TMAbort$  The transaction is aborted by the *TM*.  
 $\vee \exists r \in RM :$  A message is sent or received by a *RM* or the *TM*.  
 $\vee TMRecvPrepared(r)$   
 $\vee RMPrepare(r)$   
 $\vee RMChooseToAbort(r)$   
 $\vee RMRecvCommitMsg(r)$   
 $\vee RMRecvAbortMsg(r)$

---

TODO on Lecture 8

---

\ \* Modification History  
\ \* Last modified Sun *Apr* 04 09:47:23 *BRT* 2021 by *felipec*  
\ \* Created Sat *Apr* 03 15:07:14 *BRT* 2021 by *felipec*