```
──────────────────── MODULE TCS ────────────────────
See DISC′2018: Multi-Shot Distributed Transaction Commit

EXTENDS Naturals, Integers, FiniteSets, Sequences, Functions, TLC,
        FiniteSetsExt
────────────────────────────────────────────────────
CONSTANTS
    Key,           the set of keys, ranged over by k ∈ Key
    Tid,           the set of transaction identifiers, ranged over by t ∈ Tid
    RSet,          RSet[t]: the read set of t ∈ Tid
    WSet,          WSet[t]: the write set of t ∈ Tid
    CVer,          CVer[t]: the commit version of t ∈ Tid
    Shard,         the set of shards, ranged over by s ∈ Shard
    Coord,         Coord[t]: the coordinator of t ∈ Tid
    KeySharding    KeySharding[k]: the shard that holds k ∈ Key

NotTid ≜ CHOOSE t : t ∉ Tid

Ver ≜ 0 . . Cardinality(Tid)  with a distinguished minimum version 0
Slot ≜ 0 . . Cardinality(Tid) − 1

TKey(t) ≜ WSet[t] ∪ {kv[1] : kv ∈ RSet[t]}
TSharding(t) ≜ {KeySharding[k] : k ∈ TKey(t)}

ASSUME
    ∧ RSet ∈ [Tid → SUBSET (Key × Ver)]
    ∧ ∀ t ∈ Tid: RSet[t] \* TODO: one version per object
    ∧ WSet ∈ [Tid → SUBSET Key]
    ∧ \* TODO: “no blind update” assumption
    ∧ CVer ∈ [Tid → Ver]
    ∧ \* TODO: higher than any of the versions read
    ∧ Coord ∈ [Tid → Shard]
    ∧ KeySharding ∈ [Key → Shard]
────────────────────────────────────────────────────
VARIABLES
    next,        next[s] ∈ Z points to the last filled slot
    txn,         txn[s][i] is the transaction (identifier) to certify in the i-th slot
    vote,        vote[s][i] is the vote for txn[s][i]
    dec,         dec[s][i] is the decision for txn[s][i]
    phase,       phase[s][i] is the phase for txn[s][i]
    msg,         the set of messages in transit
    submitted    the set of t ∈ Tid that have been submitted to TCS

sVars ≜ ⟨next, txn, vote, dec, phase⟩
vars ≜ ⟨next, txn, vote, dec, phase, msg, submitted⟩
────────────────────────────────────────────────────
Message ≜ [type : {“PREPARE”}, t : Tid, s : Shard]
```

```
    ∪ [type : {"PREPARE_ACK"}, s : Shard, n : Int, t : Tid, v : {"COMMIT", "ABORT"}]
    ∪ [type : {"DECISION"}, p : Int, d : {"COMMIT", "ABORT"}, s : Shard]

Send(m) ≜ msg′ = msg ∪ m
Delete(m) ≜ msg′ = msg \ m
SendAndDelete(sm, dm) ≜ msg′ = (msg ∪ sm) \ dm

TypeOK ≜
    ∧ next ∈ [Shard → Int]
    ∧ txn ∈ [Shard → [Slot → Tid ∪ {NotTid}]]
    ∧ vote ∈ [Shard → [Slot → {"COMMIT", "ABORT", "NULL"}]]
    ∧ dec ∈ [Shard → [Slot → {"COMMIT", "ABORT", "NULL"}]]
    ∧ phase ∈ [Shard → [Slot → {"START", "PREPARED", "DECIDED"}]]
    ∧ msg ⊆ Message
    ∧ submitted ⊆ Tid

Init ≜
    ∧ next = [s ∈ Shard ↦ −1]
    ∧ txn = [s ∈ Shard ↦ [i ∈ Slot ↦ NotTid]]
    ∧ vote = [s ∈ Shard ↦ [i ∈ Slot ↦ "NULL"]]
    ∧ dec = [s ∈ Shard ↦ [i ∈ Slot ↦ "NULL"]]
    ∧ phase = [s ∈ Shard ↦ [i ∈ Slot ↦ "START"]]
    ∧ msg = {}
    ∧ submitted = {}

Vote(t, s, n) ≜ "ABORT"  TODO
Decision(ms) ≜ "ABORT"  TODO

Certify(t) ≜  Certify t ∈ Tid
    ∧ t ∈ Tid \ submitted
    ∧ Send([type : {"PREPARE"}, t : {t}, s : TSharding(t)])
    ∧ submitted′ = submitted ∪ {t}
    ∧ UNCHANGED sVars

Prepare(t, s) ≜  Prepare t ∈ Tid on s ∈ Shard when receive "PREPARE(t)" message
    ∧ ∃ m ∈ msg :
        ∧ m = [type ↦ "PREPARE", t ↦ t, s ↦ s]
        ∧ next′ = [next EXCEPT ![s] = @ + 1]
        ∧ txn′ = [txn EXCEPT ![s][next′[s]] = t]
        ∧ vote′ = [vote EXCEPT ![s][next′[s]] = Vote(t, s, next′[s])]  TODO
        ∧ phase′ = [phase EXCEPT ![s][next′[s]] = "PREPARED"]
        ∧ SendAndDelete({[type ↦ "PREPARE_ACK",
                          s ↦ s,
                          n ↦ next′[s],
                          t ↦ t,
                              v ↦ vote′[s][next′[s]]]},
                          {m})
    ∧  UNCHANGED ⟨dec, submitted⟩

Pos(t, s)    ≜
    LET m ≜ ChooseUnique(msg, LAMBDA m : m.type = "PREPARE_ACK" ∧ m.t = t ∧ m.s = s)
     IN   m.n

PrepareAck(t, s) ≜
    ∧ s = Coord[t]
    ∧ LET ms  ≜ {m ∈ msg : m.type = "PREPARE_ACK" ∧ m.t = t}
      shards  ≜ {m.s : m ∈ ms}
       IN   ∧ shards = TSharding(t)
            ∧ SendAndDelete({[type ↦ "DECISION",
                              p ↦ Pos(t, shard),
                              d ↦ Decision(ms),
                              s ↦ shard] : shard ∈ shards},
                         ms)
    ∧ UNCHANGED ⟨sVars, submitted⟩

Next ≜
    ∨ ∃ t ∈ Tid : Certify(t)
    ∨ ∃ t ∈ Tid, s ∈ Shard :
        ∨ Prepare(t, s)
        ∨ PrepareAck(t, s)

Spec ≜ Init ∧ □[Next]_vars
```

* Modification History
* Last modified Sun *Jun* 13 12:25:10 *CST* 2021 by *hengxin*
* Created Sat *Jun* 12 21:01:57 *CST* 2021 by *hengxin*