```
──────────────────── MODULE TxCure ────────────────────
  Transactional Cure Protocol without Strong Transactions.

  TODO:

  - Values are irrelevant.
EXTENDS Naturals, FiniteSets, TLC, SequenceUtils, RelationUtils, MathUtils
───────────────────────────────────────────────────────
CONSTANTS
    Key,                the set of keys, ranged over by k ∈ Key
    Value,              the set of values, ranged over by v ∈ Value
    Client,             the set of clients, ranged over by c ∈ Client
    Partition,          the set of partitions, ranged over by p ∈ Partition
    Datacenter,         the set of datacenters, ranged over by d ∈ Datacenter
    KeySharding,            the mapping from Key to Partition
    ClientAttachment        the mapping from Client to Datacenter

NotVal ≜ CHOOSE v : v ∉ Value

ASSUME
    ∧ KeySharding ∈ [Key → Partition]
    ∧ ClientAttachment ∈ [Client → Datacenter]
───────────────────────────────────────────────────────
VARIABLES
  At the client side:
    cvc,        cvc[c]: vector clock of client c ∈ Client
    tid,        tid[c]: transaction identifier of the current ongoing transaction of client c ∈ Client
    coord,      coord[c]: coordinator (partition) of the current ongoing transaction of client c ∈ Client
  At the server side (each for partition p ∈ Partition in d ∈ Datacenter):
    opLog,          opLog[p][d]: log
    clock,          clock[p][d]: current clock
    knownVC,        knownVC[p][d]: vector clock
    stableVC,       stableVC[p][d]: stable snapshot
    uniformVC,      uniformVC[p][d]: uniform snapshot
    snapshotVC,     snapshotVC[p][d][t]: snapshot vector clock of transaction t
  history:
    L,   L[c]: local history at client c ∈ Client
  communication:
    msgs,   the set of messages in transit
    incoming  incoming[p][d]: incoming FIFO channel for propagating updates and heartbeats

cVars  ≜ ⟨cvc, tid, coord⟩
sVars  ≜ ⟨opLog, clock, knownVC, stableVC, uniformVC, snapshotVC⟩
mVars ≜ ⟨msgs, incoming⟩
hVars  ≜ ⟨L⟩
vars ≜ ⟨cVars, sVars, mVars, hVars⟩
───────────────────────────────────────────────────────
```

```
Tid ≜ [seq : Nat, p : Partition, d : Datacenter]   transaction identifier

VC ≜ [Datacenter → Nat]   vector clock with an entry per datacenter d ∈ Datacenter
VCInit ≜ [d ∈ Datacenter ↦ 0]
Merge(vc1, vc2) ≜ [d ∈ Datacenter ↦ Max(vc1[d], vc2[d])]

DC ≜ Cardinality(Datacenter)
DCIndex ≜ CHOOSE f ∈ [1 . . DC → Datacenter] : Injective(f)
LTE(vc1, vc2) ≜   less-than-or-equal-to comparator for vector clocks
      LET RECURSIVE LTEHelper(_, _, _)
            LTEHelper(vc1h, vc2h, index) ≜
                IF index > DC THEN TRUE   EQ
                 ELSE LET d ≜ DCIndex[index]
                        IN  CASE vc1h[d] < vc2h[d] → TRUE   LT
                              □  vc1h[d] > vc2h[d] → FALSE   GT
                              □  OTHER → LTEHelper(vc1h, vc2h, index + 1)
      IN    LTEHelper(vc1, vc2, 1)

KVTuple ≜ [key : Key, val : Value ∪ {NotVal}, vc : VC]
OpTuple ≜ [type : {"R", "W"}, kv : KVTuple, c : Client, cnt : Nat]

Message ≜
           [type : {"StartRequest"}, vc : VC, c : Client, p : Partition, d : Datacenter]
      ∪    [type : {"StartReply"}, tid : Tid, vc : VC, c : Client]
      ∪    [type : {"ReadRequest"}, tid : Tid, key : Key, c : Client, p : Partition, d : Datacenter]
      ∪    [type : {"ReadReply"}, val : Value ∪ {NotVal}, c : Client]   val is irrelevant
      ∪    [type : {"UpdateRequest"}, tid : Tid, key : Key, val : Value, c : Client, p : Partition, d : Datacenter]
      ∪    [type : {"UpdateReply"}, c : Client]
      ∪    [type : {"CommitRequest"}, tid : Tid, c : Client, p : Partition, d : Datacenter]   val is irrelevant
      ∪    [type : {"CommitReply"}, vc : VC, c : Client]
      ∪    [type : {"Replicate"}, d : Datacenter, kv : KVTuple]
      ∪    [type : {"Heartbeat"}, d : Datacenter, ts : Nat]

Send(m) ≜ msgs′ = msgs ∪ {m}
SendAndDelete(sm, dm) ≜ msgs′ = (msgs ∪ {sm}) \ {dm}

TypeOK ≜
      ∧ cvc ∈ [Client → VC]
      ∧ clock ∈ [Partition → [Datacenter → Nat]]
      ∧ knownVC ∈ [Partition → [Datacenter → VC]]
      ∧ stableVC ∈ [Partition → [Datacenter → VC]]
      ∧ opLog ∈ [Partition → [Datacenter → SUBSET KVTuple]]
      ∧ msgs ⊆ Message
      ∧ incoming ∈ [Partition → [Datacenter → Seq(Message)]]
      ∧ L ∈ [Client → Seq(OpTuple)]
├──────────────────────────────────────────────────────────────────┤
Init ≜
```

```
    ∧ cvc = [c ∈ Client ↦ VCInit]
    ∧ clock = [p ∈ Partition ↦ [d ∈ Datacenter ↦ 0]]
    ∧ knownVC = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
    ∧ stableVC = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
    ∧ opLog = [p ∈ Partition ↦ [d ∈ Datacenter ↦
                    [key : {k ∈ Key : KeySharding[k] = p}, val : {NotVal}, vc : {VCInit}]]]
    ∧ msgs = {}
    ∧ incoming = [p ∈ Partition ↦ [d ∈ Datacenter ↦ ⟨⟩]]
    ∧ L = [c ∈ Client ↦ ⟨⟩]
├──────────────────────────────────────────────────────────────────────┤
  Client operations at client c ∈ Client.

CanIssue(c) ≜ ∀ m ∈ msgs :    to ensure well-formedness of clients
    m.type ∈ {"StartRequest", "StartReply",
        "ReadRequest", "ReadReply",
        "UpdateRequest", "UpdateReply"
        "CommitRequest", "CommitReply"} ⇒ m.c ≠ c

Start(c) ≜    c ∈ Client starts a transaction
    ∧ CanIssue(c)
    ∧ ∃ p ∈ Partition :
        ∧ coord′ = [coord EXCEPT ![c] = p]
        ∧ Send([type ↦ "StartRequest", vc ↦ cvc[c],
                c ↦ c, p ↦ p, d ↦ ClientAttachment[c]])
    ∧ UNCHANGED ⟨cvc, tid, sVars, incoming, hVars⟩

StartReply(c) ≜    c ∈ Client handles the reply to its start request
    ∧ ∃ m ∈ msgs :
        ∧ m.type = "StartReply" ∧ m.c = c    such m is unique due to well-formedness
        ∧ cvc′ = [cvc EXCEPT ![c] = m.snapshotVC]
        ∧ tid′ = [tid EXCEPT ![c] = m.tid]
        ∧ msgs′ = msgs \ {m}
    ∧ UNCHANGED ⟨coord, sVars, incoming, hVars⟩

Read(c, k) ≜    c ∈ Client reads from k ∈ Key
     ∧ CanIssue(c)
     ∧ Send([type ↦ "ReadRequest", tid ↦ tid[c], key ↦ k,
             c ↦ c, p ↦ coord[c], d ↦ ClientAttachment[c]])
     ∧ UNCHANGED ⟨cVars, sVars, incoming, hVars⟩

ReadReply(c) ≜    c ∈ Client handles the reply to its read request
    ∧ ∃ m ∈ msgs :
        ∧ m.type = "ReadReply" ∧ m.c = c    such m is unique due to well-formedness
        ∧ msgs′ = msgs \ {m}
    ∧ UNCHANGED ⟨cVars, sVars, incoming, hVars⟩

Update(c, k, v) ≜    c ∈ Client updates k ∈ Key with v ∈ Value
```

```
      ∧ CanIssue(c)
      ∧ Send([type ↦ "UpdateRequest", tid ↦ tid[c], key ↦ k, val ↦ v,
                   c ↦ c, p ↦ coord[c], d      ↦ ClientAttachment[c]])
      ∧ UNCHANGED ⟨cVars, sVars, incoming, hVars⟩

UpdateReply(c) ≜    c ∈ Client handles the reply to its update request
      ∧ ∃ m ∈ msgs :
          ∧ m.type = "UpdateReply" ∧ m.c = c  such m is unique due to well-formedness
          ∧ msgs' = msgs \ {m}
      ∧ UNCHANGED ⟨cVars, sVars, incoming, hVars⟩

Commit(c) ≜    c ∈ Client commits the ongoing transaction tid[c]
      ∧ CanIssue(c)
      ∧ Send([type ↦ "CommitRequest", tid ↦ tid[c],
                   c ↦ c, p ↦ coord[c], d       ↦ ClientAttachment[c]])
      ∧ UNCHANGED ⟨cVars, sVars, incoming, hVars⟩

CommitReply(c) ≜    c ∈ Client handles the reply to its commit request
      ∧ ∃ m ∈ msgs :
          ∧ m.type = "CommitReply" ∧ m.c = c  such m is unique due to well-formedness
          ∧ cvc' = [cvc EXCEPT ![c] = [m.vc]]
          ∧ msgs' = msgs \ {m}
      ∧ UNCHANGED ⟨tid, coord, sVars, incoming, hVars⟩
├──────────────────────────────────────────────────────────────────────────┤
  Server operations at partition p ∈ Partition in datacenter d ∈ Datacenter.

ReadRequest(p, d) ≜    handle a "ReadRequest"
      ∧ ∃ m ∈ msgs :
          ∧ m.type = "ReadRequest" ∧ m.p = p ∧ m.d = d
          ∧ stableVC' = [stableVC EXCEPT ![p][d] = Merge(m.vc, @)]
          ∧ LET kvs ≜ {kv ∈ opLog[p][d] :
                            ∧ kv.key = m.key
                            ∧ ∀ dc ∈ Datacenter \ {d} : kv.vc[dc] ≤ stableVC'[p][d][dc]}
                lkv ≜ CHOOSE kv ∈ kvs : ∀ akv ∈ kvs : LTE(akv.vc, kv.vc)
              IN   ∧ SendAndDelete([type ↦ "ReadReply", val ↦ lkv.val, vc ↦ lkv.vc, c ↦ m.c], m)
                   ∧ L' = [L EXCEPT ![m.c] = Append(@, [type ↦ "R", kv ↦ lkv, c ↦ m.c, cnt ↦ Len(@) + 1])]
      ∧ UNCHANGED ⟨cVars, clock, knownVC, opLog, incoming⟩

UpdateRequest(p, d) ≜    handle a "UpdateRequest"
      ∧ ∃ m ∈ msgs :
          ∧ m.type = "UpdateRequest" ∧ m.p = p ∧ m.d = d
          ∧ m.vc[d] < clock[p][d]   waiting condition; (" ≤ " strengthed to " < ")
          ∧ stableVC' = [stableVC EXCEPT ![p][d] = Merge(m.vc, @)]
          ∧ LET kv ≜ [key ↦ m.key, val ↦ m.val,
                          vc ↦ [m.vc EXCEPT ![d]   = clock[p][d]]]
              IN   ∧ opLog' = [opLog EXCEPT ![p][d] = @ ∪ {kv}]
```

```
                    ∧ SendAndDelete([type ↦ "UpdateReply", ts ↦ clock[p][d], c ↦ m.c, d ↦ d], m)
                    ∧ incoming′ = [incoming EXCEPT ![p] = [dc ∈ Datacenter ↦
                        IF dc = d THEN @[dc] ELSE  Append(@[dc], [type ↦ "Replicate", d ↦ d, kv ↦ kv])]]
                    ∧ L′ = [L EXCEPT ![m.c] = Append(@, [type ↦ "W", kv ↦ kv, c ↦ m.c, cnt ↦ Len(@) + 1])]
       ∧ UNCHANGED ⟨cVars, clock, knownVC⟩

Replicate(p, d) ≜   handle a "Replicate"
       ∧ incoming[p][d] ≠ ⟨⟩
       ∧ LET m ≜ Head(incoming[p][d])
         IN   ∧ m.type = "Replicate"
              ∧ opLog′ = [opLog EXCEPT ![p][d] = @ ∪ {m.kv}]
              ∧ knownVC′ = [knownVC EXCEPT ![p][d][m.d] = m.kv.vc[m.d]]
              ∧ incoming′ = [incoming EXCEPT ![p][d] = Tail(@)]
       ∧ UNCHANGED ⟨cVars, cvc, clock, stableVC, L, msgs⟩

Heartbeat(p, d) ≜   handle a "Heartbeat"
       ∧ incoming[p][d] ≠ ⟨⟩
       ∧ LET m ≜ Head(incoming[p][d])
         IN   ∧ m.type = "Heartbeat"
              ∧ knownVC′ = [knownVC EXCEPT ![p][d][m.d] = m.ts]
              ∧ incoming′ = [incoming EXCEPT ![p][d] = Tail(@)]
       ∧ UNCHANGED ⟨cVars, cvc, clock, stableVC, opLog, L, msgs⟩

 Clock management at partition p ∈ Partition in datacenter d ∈ Datacenter
Tick(p, d) ≜   clock[p][d] ticks
        ∧ clock′ = [clock EXCEPT ![p][d] = @ + 1]
        ∧ knownVC′ = [knownVC EXCEPT ![p][d][d] = clock′[p][d]]
        ∧ incoming′ = [incoming EXCEPT ![p] = [dc ∈ Datacenter ↦
             IF dc = d THEN @[dc] ELSE  Append(@[dc], [type ↦ "Heartbeat", d ↦ d, ts ↦ knownVC′[p][d][d]])]]
        ∧ UNCHANGED ⟨cVars, cvc, stableVC, opLog, L, msgs⟩

UpdateCSS(p, d) ≜   update stableVC[p][d]
       ∧ stableVC′ = [stableVC EXCEPT ![p][d] =
                      [dc ∈ Datacenter ↦ SetMin({knownVC[pp][d][dc] : pp ∈ Partition})]]
       ∧ UNCHANGED ⟨cVars, mVars, clock, knownVC, opLog, L⟩

Next ≜
       ∨ ∃ c ∈ Client, k ∈ Key : Read(c, k)
       ∨ ∃ c ∈ Client, k ∈ Key, v ∈ Value : Update(c, k, v)
       ∨ ∃ c ∈ Client : ReadReply(c) ∨ UpdateReply(c)
       ∨ ∃ p ∈ Partition, d ∈ Datacenter :
           ∨ ReadRequest(p, d)
           ∨ UpdateRequest(p, d)
           ∨ Replicate(p, d)
           ∨ Heartbeat(p, d)
           ∨ Tick(p, d)
```

$\qquad\qquad \lor \mathit{UpdateCSS}(p,\ d)$

$\mathit{Spec} \triangleq \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}}$

* Modification History
* Last modified *Fri Nov* 20 21:27:11 *CST* 2020 by *hengxin*
* Created *Fri Nov* 20 18:51:11 *CST* 2020 by *hengxin*