```
┌──────────────────────────── MODULE Cure ────────────────────────────┐
  See ICDCS2016: “Cure: Strong Semantics Meets High Availability and Low Latency”.
EXTENDS Naturals, Sequences, TLC
├─────────────────────────────────────────────────────────────────────┤
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]: the vector clock of client c ∈ Client
  At the server side (each for partition p ∈ Partition in d ∈ Datacenter):
    clock,      clock[p][d]: the current clock
    pvc,        pvc[p][d]: the vector clock
    css,        css[p][d]: the stable snapshot
    PMC,        PMC[p][d]: matrix clock
    store,      store[p][d]: the kv store
    updates,    updates[p][d]: the buffer of updates
  Client-server communication
    msgs    the set of messages in transit

cVars  ≜ ⟨cvc⟩
sVars  ≜ ⟨clock, pvc, css, PMC, store, updates⟩
mVars ≜ ⟨msgs⟩
vars ≜ ⟨cvc, clock, pvc, css, PMC, store, updates, msgs⟩
├─────────────────────────────────────────────────────────────────────┤
Clock ≜ Nat
VC ≜ [Datacenter → Clock]    vector clock with an entry per datacenter d ∈ Datacenter
VCInit ≜ [d ∈ Datacenter ↦ 0]
KVTuple ≜ [key : Key, val : Value ∪ {NotVal}, vc : VC]

Message ≜
            [type : {“ReadRequest”}, key : Key, vc : VC, c : Client, p : Partition, d : Datacenter]
     ∪      [type : {“ReadReply”}, val : Value ∪ {NotVal}, vc : VC, c : Client]
     ∪      [type : {“UpdateRequest”}, key : Key, val : Value, vc : VC, c : Client, p : Partition, d : Datacenter]
    ∪     [type : {"UpdateReply"}, ts : Clock, c : Client, d : Datacenter]

TypeOK ≜
    ∧  cvc ∈ [Client → VC]
    ∧  clock ∈ [Partition → [Datacenter → Clock]]
    ∧  pvc   ∈ [Partition → [Datacenter → VC]]
    ∧  css   ∈ [Partition → [Datacenter → VC]]
    ∧  PMC ∈ [Partition → [Datacenter → [Partition → VC]]]
    ∧  store ∈ [Partition → [Datacenter → SUBSET KVTuple]]
    ∧  updates ∈ [Partition → [Datacenter → Seq(KVTuple)]]
    ∧  msgs ⊆ Message
--------------------------------------------------------------------------------
Init ≜
    ∧ cvc = [c ∈ Client ↦ VCInit]
    ∧ clock = [p ∈ Partition ↦ [d ∈ Datacenter ↦ 0]]
    ∧ pvc   = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
    ∧ css   = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
    ∧ PMC = [p ∈ Partition ↦ [d ∈ Datacenter ↦ [q ∈ Partition ↦ VCInit]]]
    ∧ store = [p ∈ Partition ↦ [d ∈ Datacenter ↦
                  [key : {k ∈ Key : KeySharding[k] = p}, val : {NotVal}, vc : {VCInit}]]]
    ∧ updates = [p ∈ Partition ↦ [d ∈ Datacenter ↦ ⟨⟩]]
    ∧ msgs = {}
--------------------------------------------------------------------------------
Max(a, b) ≜ IF a < b THEN b ELSE a

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

Ready2Issue(c) ≜ ∀ m ∈ msgs :
    m.type ∈ {"ReadRequest", "ReadReply", "UpdateRequest", "UpdateReply"} ⇒ m.c ≠ c
--------------------------------------------------------------------------------
 Client operations at client c ∈ Client.

Read(c, k) ≜   c ∈ Client reads from k ∈ Key
      ∧ Ready2Issue(c)
      ∧ Send([type ↦ "ReadRequest", key ↦ k, vc ↦ cvc[c],
                c ↦ c, p ↦ KeySharding[k], d ↦ ClientAttachment[c]])
      ∧ UNCHANGED ⟨cVars, sVars⟩

ReadRply(c) ≜   c ∈ Client handles the reply to its read request
    ∧ ∃ m ∈ msgs :
        ∧ m.type = "ReadReply" ∧ m.c = c   such m is unique
        ∧ cvc′ = [cvc EXCEPT ![c] = [d ∈ Datacenter ↦ Max(m.vc[d], @[d])]]
        ∧ msgs′ = msgs \ {m}
    ∧ UNCHANGED ⟨sVars⟩

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

```
    $\land\ Ready2Issue(c)$
    $\land\ Send([type \mapsto$ “UpdateRequest”$,\ key \mapsto k,\ val \mapsto v,$
              $vc \mapsto cvc[c],\ c \mapsto c,\ p \mapsto KeySharding[k],\ d \mapsto ClientAttachment[c]])$
    $\land$ UNCHANGED $\langle cVars,\ sVars \rangle$

$UpdateReply(c) \triangleq$   $c \in Client$ handles the reply to its update request
    $\land\ \exists\, m \in msgs :$
        $\land\ m.type =$ “UpdateReply” $\land\ m.c = c$   such $m$ is unique
        $\land\ cvc' = [cvc$ EXCEPT $![c][m.d] = m.ts]$
        $\land\ msgs' = msgs \setminus \{m\}$
    $\land$ UNCHANGED $\langle sVars \rangle$
--------------------------------------------------------------------------------
 Server operations at partition $p \in Partition$ in datacenter $d \in Datacenter$.

$ReadRequest(p,\ d) \triangleq$   handle a “ReadRequest”
    $\land\ \exists\, m \in msgs :$
        $\land\ m.type =$ “ReadRequest” $\land\ m.p = p \land m.d = d$   such $m$ may be not unique
        $\land\ css' = [css$ EXCEPT $![p][d] =$
            $[dc \in Datacenter \mapsto$ IF $dc = d$ THEN $@[dc]$ ELSE $Max(m.vc[dc],\ @[dc])]]$
        $\land$ LET $kvs \triangleq \{kv \in store[p][d] :$
                            $\land\ kv.key = m.key$
                            $\land\ \forall\, dc \in Datacenter \setminus \{d\} : kv.vc[dc] \leq css'[p][d][dc]\}$
                $lkv \triangleq$ CHOOSE $kv \in kvs :$   choose the latest one (Existence? Uniqueness?)
                            $\forall\, akv \in kvs,\ dc \in Datacenter : akv.vc[dc] \leq kv.vc[dc]$
           IN $SendAndDelete([type \mapsto$ “ReadReply”$,\ val \mapsto lkv.val,\ vc \mapsto lkv.vc,\ c \mapsto m.c],\ m)$
    $\land$ UNCHANGED $\langle cVars,\ clock,\ pvc,\ PMC,\ store,\ updates \rangle$

$UpdateRequest(p,\ d) \triangleq$   handle a “UpdateRequest”
    $\land\ \exists\, m \in msgs :$
        $\land\ m.type =$ “UpdateRequest” $\land\ m.p = p \land m.d = d$   such $m$ may be not unique
        $\land\ m.vc[d] \leq clock[p][d]$   waiting condition
        $\land\ pvc' = [pvc$ EXCEPT $![p][d][d] = clock[p][d]]$
        $\land\ css' = [css$ EXCEPT $![p][d] =$
            $[dc \in Datacenter \mapsto$ IF $dc = d$ THEN $@[dc]$ ELSE $Max(m.vc[dc],\ @[dc])]]$
        $\land$ LET $kv \triangleq [key \mapsto m.key,\ val \mapsto m.val,$
                        $vc \mapsto [m.vc$ EXCEPT $![d] = clock[p][d]]]$
           IN  $\land\ store' = [store$ EXCEPT $![p][d] = @ \cup \{kv\}]$
               $\land\ updates' = [updates$ EXCEPT $![p][d] = @ \circ \langle kv \rangle]$
               $\land\ SendAndDelete([type \mapsto$ “UpdateReply”$,\ ts \mapsto clock[p][d],\ c \mapsto m.c,\ d \mapsto d],\ m)$
    $\land$ UNCHANGED $\langle cVars,\ clock,\ PMC \rangle$
--------------------------------------------------------------------------------
$Next \triangleq$
    $\lor\ \exists\, c \in Client,\ k \in Key : Read(c,\ k)$
    $\lor\ \exists\, c \in Client,\ k \in Key,\ v \in Value : Update(c,\ k,\ v)$
    $\lor\ \exists\, c \in Client : ReadReply(c) \lor UpdateReply(c)$
    $\lor\ \exists\, p \in Partition,\ d \in Datacenter : ReadRequest(p,\ d) \lor UpdateRequest(p,\ d)$
```

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