```
1  ---------- MODULE Cure ----------
   See ICDCS2016: "Cure: Strong Semantics Meets High Availability and Low Latency".
5  EXTENDS Naturals, Sequences, FiniteSets
6  ----------
7  Max(a, b) ≜ IF a < b THEN b ELSE a
8  Min(S) ≜ CHOOSE a ∈ S : ∀ b ∈ S : a ≤ b
9  Injective(f) ≜ ∀ a, b ∈ DOMAIN f  : (a ≠ b) ⇒ (f[a] ≠ f[b])
10 ----------
11 CONSTANTS
12     Key,          the set of keys, ranged over by k ∈ Key
13     Value,        the set of values, ranged over by v ∈ Value
14     Client,       the set of clients, ranged over by c ∈ Client
15     Partition,    the set of partitions, ranged over by p ∈ Partition
16     Datacenter,   the set of datacenters, ranged over by d ∈ Datacenter
17     KeySharding,  the mapping from Key to Partition
18     ClientAttachment  the mapping from Client to Datacenter

20 NotVal ≜ CHOOSE v : v ∉ Value

22 ASSUME
23     ∧ KeySharding ∈ [Key → Partition]
24     ∧ ClientAttachment ∈ [Client → Datacenter]
25 ----------
26 VARIABLES
27  At the client side:
28     cvc,    cvc[c]: the vector clock of client c ∈ Client
29  At the server side (each for partition p ∈ Partition in d ∈ Datacenter):
30     clock,  clock[p][d]: the current clock
31     pvc,    pvc[p][d]: the vector clock
32     css,    css[p][d]: the stable snapshot
33     store,  store[p][d]: the kv store
34  communication:
35     msgs,   the set of messages in transit
36     incoming  fifo[p][d]: incomming FIFO channel; for propagating updates and heartbeats

38 cVars ≜ ⟨cvc⟩
39 sVars ≜ ⟨clock, pvc, css, store⟩
40 mVars ≜ ⟨msgs, incoming⟩
41 vars ≜ ⟨cvc, clock, pvc, css, store, msgs, incoming⟩
42 ----------
43 Clock ≜ Nat
44 VC ≜ [Datacenter → Clock]  vector clock with an entry per datacenter d ∈ Datacenter
45 VCInit ≜ [d ∈ Datacenter ↦ 0]
46 Merge(vc1, vc2) ≜ [d ∈ Datacenter ↦ Max(vc1[d], vc2[d])]
47 KVTuple ≜ [key : Key, val : Value ∪ {NotVal}, vc : VC]
```

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

61  Message ≜
62          [type : {"ReadRequest"}, key : Key, vc : VC, c : Client, p : Partition, d : Datacenter]
63      ∪   [type : {"ReadReply"}, val : Value ∪ {NotVal}, vc : VC, c : Client]
64      ∪   [type : {"UpdateRequest"}, key : Key, val : Value, vc : VC, c : Client, p : Partition, d : Datacenter]
65      ∪   [type : {"UpdateReply"}, ts : Clock, c : Client, d : Datacenter]
66      ∪   [type : {"Replicate"}, d  : Datacenter, kv : KVTuple]
67      ∪   [type : {"Heartbeat"}, d : Datacenter, ts  : Clock]

69  TypeOK ≜
70      ∧   cvc ∈ [Client → VC]
71      ∧   clock ∈ [Partition → [Datacenter → Clock]]
72      ∧   pvc ∈ [Partition → [Datacenter → VC]]
73      ∧   css ∈ [Partition → [Datacenter → VC]]
74      ∧   store ∈ [Partition → [Datacenter → SUBSET KVTuple]]
75      ∧   msgs ⊆ Message
76      ∧   incoming ∈ [Partition → [Datacenter → Seq(Message)]]
77  ----------------------------------------------------------------------
78  Init ≜
79      ∧ cvc = [c ∈ Client ↦ VCInit]
80      ∧ clock = [p ∈ Partition ↦ [d ∈ Datacenter ↦ 0]]
81      ∧ pvc = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
82      ∧ css = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
83      ∧ store = [p ∈ Partition ↦ [d ∈ Datacenter ↦
84                    [key : {k ∈ Key : KeySharding[k] = p}, val : {NotVal}, vc : {VCInit}]]]
85      ∧ msgs = {}
86      ∧ incoming = [p ∈ Partition ↦ [d ∈ Datacenter ↦ ⟨⟩]]
87  ----------------------------------------------------------------------
88  Send(m) ≜ msgs′ = msgs ∪ {m}
89  SendAndDelete(sm, dm) ≜ msgs′ = (msgs ∪ {sm}) \ {dm}

91  CanIssue(c) ≜ ∀ m ∈ msgs :
92      m.type ∈ {"ReadRequest", "ReadReply", "UpdateRequest", "UpdateReply"} ⇒ m.c ≠ c
93  ----------------------------------------------------------------------
94   Client operations at client c ∈ Client.
```

```
96  Read(c, k) ≜   c ∈ Client reads from k ∈ Key
97        ∧ CanIssue(c)
98        ∧ Send([type ↦ "ReadRequest", key ↦ k, vc ↦ cvc[c],
99                c ↦ c, p ↦ KeySharding[k], d ↦ ClientAttachment[c]])
100       ∧ UNCHANGED ⟨cVars, sVars, incoming⟩

102 ReadReply(c) ≜   c ∈ Client handles the reply to its read request
103     ∧ ∃ m ∈ msgs :
104         ∧ m.type = "ReadReply" ∧ m.c = c   such m is unique due to well-formedness
105         ∧ cvc' = [cvc EXCEPT ![c] = Merge(m.vc, @)]
106         ∧ msgs' = msgs \ {m}
107     ∧ UNCHANGED ⟨sVars, incoming⟩

109 Update(c, k, v) ≜   c ∈ Client updates k ∈ Key with v ∈ Value
110     ∧ CanIssue(c)
111     ∧ Send([type ↦ "UpdateRequest", key ↦ k, val ↦ v,
112             vc ↦ cvc[c], c ↦ c, p ↦ KeySharding[k], d ↦ ClientAttachment[c]])
113     ∧ UNCHANGED ⟨cVars, sVars, incoming⟩

115 UpdateReply(c) ≜   c ∈ Client handles the reply to its update request
116     ∧ ∃ m ∈ msgs :
117         ∧ m.type = "UpdateReply" ∧ m.c = c   such m is unique due to well-formedness
118         ∧ cvc' = [cvc EXCEPT ![c][m.d] = m.ts]
119         ∧ msgs' = msgs \ {m}
120     ∧ UNCHANGED ⟨sVars, incoming⟩
121 ├──────────────────────────────────────────────────────────────────────────┤
122  Server operations at partition p ∈ Partition in datacenter d ∈ Datacenter.

124 ReadRequest(p, d) ≜   handle a "ReadRequest"
125     ∧ ∃ m ∈ msgs :
126         ∧ m.type = "ReadRequest" ∧ m.p = p ∧ m.d = d
127         ∧ css' = [css EXCEPT ![p][d] = Merge(m.vc, @)]
128         ∧ LET kvs ≜ {kv ∈ store[p][d] :
129                       ∧ kv.key = m.key
130                       ∧ ∀ dc ∈ Datacenter \ {d} : kv.vc[dc] ≤ css'[p][d][dc]}
131               lkv ≜ CHOOSE kv ∈ kvs : ∀ akv ∈ kvs : LTE(akv.vc, kv.vc)
132           IN  SendAndDelete([type ↦ "ReadReply", val ↦ lkv.val, vc ↦ lkv.vc, c ↦ m.c], m)
133     ∧ UNCHANGED ⟨cVars, clock, pvc, store, incoming⟩

135 UpdateRequest(p, d) ≜   handle a "UpdateRequest"
136     ∧ ∃ m ∈ msgs :
137         ∧ m.type = "UpdateRequest" ∧ m.p = p ∧ m.d = d
138         ∧ m.vc[d] < clock[p][d]   waiting condition; (" ≤ " strengthed to " < ")
139         ∧ css' = [css EXCEPT ![p][d] = Merge(m.vc, @)]
140         ∧ LET kv ≜ [key ↦ m.key, val ↦ m.val,
141                     vc ↦ [m.vc EXCEPT ![d] = clock[p][d]]]
```

```
            IN    ∧ store' = [store 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])]]
    ∧ UNCHANGED ⟨cVars, clock, pvc⟩

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

Heartbeat(p, d) ≜  handle a "Heartbeat"
    ∧ incoming[p][d] ≠ ⟨⟩
    ∧ LET m ≜ Head(incoming[p][d])
      IN  ∧ m.type = "Heartbeat"
          ∧ pvc' = [pvc EXCEPT ![p][d][m.d] = m.ts]
          ∧ incoming' = [incoming EXCEPT ![p][d] = Tail(@)]
    ∧ UNCHANGED ⟨cVars, cvc, clock, css, store, msgs⟩
├──────────────────────────────────────────────────────────────────────────┤
 Clock management at partition p ∈ Partition in datacenter d ∈ Datacenter
Tick(p, d) ≜  clock[p][d] ticks
     ∧ clock' = [clock EXCEPT ![p][d] = @ + 1]
     ∧ pvc' = [pvc 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 ↦ pvc'[p][d][d]])]]
     ∧ UNCHANGED ⟨cVars, cvc, css, store, msgs⟩

UpdateCSS(p, d) ≜  update css[p][d]
    ∧ css' = [css EXCEPT ![p][d] =
                [dc ∈ Datacenter ↦ Min({pvc[pp][d][dc] : pp ∈ Partition})]]
    ∧ UNCHANGED ⟨cVars, mVars, clock, pvc, store⟩
├──────────────────────────────────────────────────────────────────────────┤
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)
```

$$\lor\ UpdateCSS(p,\ d)$$

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