```
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  history:
35     L,  L[c]: local history at client c ∈ Client
36  communication:
37     msgs,  the set of messages in transit
38     incoming  incoming[p][d]: incoming FIFO channel for propagating updates and heartbeats

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

```
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)

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 : Nat, c : Client, d : Datacenter]
    ∪     [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]]
    ∧  pvc ∈ [Partition → [Datacenter → VC]]
    ∧  css ∈ [Partition → [Datacenter → VC]]
    ∧  store ∈ [Partition → [Datacenter → SUBSET KVTuple]]
    ∧  msgs ⊆ Message
    ∧  incoming ∈ [Partition → [Datacenter → Seq(Message)]]
    ∧  L ∈ [Client → Seq(KVTuple)]
------------------------------------------------------------------------
Init ≜
    ∧ cvc = [c ∈ Client ↦ VCInit]
    ∧ clock = [p ∈ Partition ↦ [d ∈ Datacenter ↦ 0]]
    ∧ pvc = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
    ∧ css = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
    ∧ store = [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.
```

```
96  CanIssue(c) ≜ ∀ m ∈ msgs :
97      m.type ∈ {"ReadRequest", "ReadReply", "UpdateRequest", "UpdateReply"} ⇒ m.c ≠ c

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

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

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

118 UpdateReply(c) ≜   c ∈ Client handles the reply to its update request
119     ∧ ∃ m ∈ msgs :
120         ∧ m.type = "UpdateReply" ∧ m.c = c   such m is unique due to well-formedness
121         ∧ cvc' = [cvc EXCEPT ![c][m.d] = m.ts]
122         ∧ msgs' = msgs \ {m}
123     ∧ UNCHANGED ⟨sVars, incoming⟩
124 ──────────────────────────────────────────────────────────────────────
125  Server operations at partition p ∈ Partition in datacenter d ∈ Datacenter.

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

139 UpdateRequest(p, d) ≜   handle a "UpdateRequest"
140     ∧ ∃ m ∈ msgs :
141         ∧ m.type = "UpdateRequest" ∧ m.p = p ∧ m.d = d
142         ∧ m.vc[d] < clock[p][d]   waiting condition; (" ≤ " strengthed to " < ")
```

```
143          ∧ css' = [css EXCEPT ![p][d] = Merge(m.vc, @)]
144          ∧ LET kv ≜ [key ↦ m.key, val ↦ m.val,
145                      vc ↦ [m.vc EXCEPT ![d] = clock[p][d]]]
146            IN  ∧ store' = [store EXCEPT ![p][d] = @ ∪ {kv}]
147                ∧ SendAndDelete([type ↦ "UpdateReply", ts ↦ clock[p][d], c ↦ m.c, d ↦ d], m)
148                ∧ incoming' = [incoming EXCEPT ![p] = [dc ∈ Datacenter ↦
149                    IF dc = d THEN @[dc] ELSE Append(@[dc], [type ↦ "Replicate", d ↦ d, kv ↦ kv])]]
150                ∧ L' = [L EXCEPT ![m.c] = Append(@, kv)]
151      ∧ UNCHANGED ⟨cVars, clock, pvc⟩

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

162  Heartbeat(p, d) ≜  handle a "Heartbeat"
163      ∧ incoming[p][d] ≠ ⟨⟩
164      ∧ LET m ≜ Head(incoming[p][d])
165        IN  ∧ m.type = "Heartbeat"
166            ∧ pvc' = [pvc EXCEPT ![p][d][m.d] = m.ts]
167            ∧ incoming' = [incoming EXCEPT ![p][d] = Tail(@)]
168      ∧ UNCHANGED ⟨cVars, cvc, clock, css, store, L, msgs⟩
169  ├──────────────────────────────────────────────────────────────────────────┤
170   Clock management at partition p ∈ Partition in datacenter d ∈ Datacenter
171  Tick(p, d) ≜  clock[p][d] ticks
172       ∧ clock' = [clock EXCEPT ![p][d] = @ + 1]
173       ∧ pvc' = [pvc EXCEPT ![p][d][d] = clock'[p][d]]
174       ∧ incoming' = [incoming EXCEPT ![p] = [dc ∈ Datacenter ↦
175           IF dc = d THEN @[dc] ELSE Append(@[dc], [type ↦ "Heartbeat", d ↦ d, ts ↦ pvc'[p][d][d]])]]
176       ∧ UNCHANGED ⟨cVars, cvc, css, store, L, msgs⟩

178  UpdateCSS(p, d) ≜  update css[p][d]
179      ∧ css' = [css EXCEPT ![p][d] =
180                  [dc ∈ Datacenter ↦ Min({pvc[pp][d][dc] : pp ∈ Partition})]]
181      ∧ UNCHANGED ⟨cVars, mVars, clock, pvc, store, L⟩
182  ├──────────────────────────────────────────────────────────────────────────┤
183  Next ≜
184      ∨ ∃ c ∈ Client, k ∈ Key : Read(c, k)
185      ∨ ∃ c ∈ Client, k ∈ Key, v ∈ Value : Update(c, k, v)
186      ∨ ∃ c ∈ Client : ReadReply(c) ∨ UpdateReply(c)
187      ∨ ∃ p ∈ Partition, d ∈ Datacenter :
188          ∨ ReadRequest(p, d)
```

$\quad\quad \lor \ UpdateRequest(p,\ d)$
$\quad\quad \lor \ Replicate(p,\ d)$
$\quad\quad \lor \ Heartbeat(p,\ d)$
$\quad\quad \lor \ Tick(p,\ d)$
$\quad\quad \lor \ UpdateCSS(p,\ d)$

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