```
MODULE Cure
```
See *ICDCS*2016: "Cure: Strong Semantics Meets High Availability and Low Latency".

```
EXTENDS Naturals, Sequences

Max(a, b) ≜ IF a < b THEN b ELSE a
Min(S) ≜ CHOOSE a ∈ S : ∀ b ∈ S : a ≤ b

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
    store,  store[p][d]: the kv store
 communication:
    msgs,   the set of messages in transit
    incoming  fifo[p][d]: incomming FIFO channel; for propagating updates and heartbeats

cVars ≜ ⟨cvc⟩
sVars ≜ ⟨clock, pvc, css, store⟩
mVars ≜ ⟨msgs, incoming⟩
vars ≜ ⟨cvc, clock, pvc, css, store, msgs, incoming⟩

Clock ≜ Nat
VC ≜ [Datacenter → Clock]   vector clock with an entry per datacenter d ∈ Datacenter
VCInit ≜ [d ∈ Datacenter ↦ 0]
Merge(vc1, vc2) ≜ [d ∈ Datacenter ↦ Max(vc1[d], vc2[d])]
KVTuple ≜ [key : Key, val : Value ∪ {NotVal}, vc : VC]

Message ≜
```

```
49              [type : {"ReadRequest"}, key : Key, vc : VC, c : Client, p : Partition, d : Datacenter]
50      ∪       [type : {"ReadReply"}, val : Value ∪ {NotVal}, vc : VC, c : Client]
51      ∪       [type : {"UpdateRequest"}, key : Key, val : Value, vc : VC, c : Client, p : Partition, d : Datacenter]
52      ∪       [type : {"UpdateReply"}, ts : Clock, c : Client, d : Datacenter]
53      ∪       [type : {"Replicate"}, d : Datacenter, kv : KVTuple]
54      ∪       [type : {"Heartbeat"}, d : Datacenter, ts : Clock]

56  TypeOK ≜
57      ∧   cvc ∈ [Client → VC]
58      ∧   clock ∈ [Partition → [Datacenter → Clock]]
59      ∧   pvc ∈ [Partition → [Datacenter → VC]]
60      ∧   css ∈ [Partition → [Datacenter → VC]]
61      ∧   store ∈ [Partition → [Datacenter → SUBSET KVTuple]]
62      ∧   msgs ⊆ Message
63      ∧   incoming ∈ [Partition → [Datacenter → Seq(Message)]]
64  ⊢──────────────────────────────────────────────────────────────────────────⊣
65  Init ≜
66      ∧ cvc = [c ∈ Client ↦ VCInit]
67      ∧ clock = [p ∈ Partition ↦ [d ∈ Datacenter ↦ 0]]
68      ∧ pvc = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
69      ∧ css = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
70      ∧ store = [p ∈ Partition ↦ [d ∈ Datacenter ↦
71                  [key : {k ∈ Key : KeySharding[k] = p}, val : {NotVal}, vc : {VCInit}]]]
72      ∧ msgs = {}
73      ∧ incoming = [p ∈ Partition ↦ [d ∈ Datacenter ↦ ⟨⟩]]
74  ⊢──────────────────────────────────────────────────────────────────────────⊣
75  Send(m) ≜ msgs′ = msgs ∪ {m}
76  SendAndDelete(sm, dm) ≜ msgs′ = (msgs ∪ {sm}) \ {dm}

78  CanIssue(c) ≜ ∀ m ∈ msgs :
79      m.type ∈ {"ReadRequest", "ReadReply", "UpdateRequest", "UpdateReply"} ⇒ m.c ≠ c
80  ⊢──────────────────────────────────────────────────────────────────────────⊣
81  Client operations at client c ∈ Client.

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

89  ReadReply(c) ≜   c ∈ Client handles the reply to its read request
90      ∧ ∃ m ∈ msgs :
91          ∧ m.type = "ReadReply" ∧ m.c = c   such m is unique due to well-formedness
92          ∧ cvc′ = [cvc EXCEPT ![c] = Merge(m.vc, @)]
93          ∧ msgs′ = msgs \ {m}
94      ∧ UNCHANGED ⟨sVars, incoming⟩
```

```
96  Update(c, k, v) ≜   c ∈ Client updates k ∈ Key with v ∈ Value
97      ∧ CanIssue(c)
98      ∧ Send([type ↦ "UpdateRequest", key ↦ k, val ↦ v,
99              vc ↦ cvc[c], c ↦ c, p ↦ KeySharding[k], d ↦ ClientAttachment[c]])
100     ∧ UNCHANGED ⟨cVars, sVars, incoming⟩

102 UpdateReply(c) ≜   c ∈ Client handles the reply to its update request
103     ∧ ∃ m ∈ msgs :
104         ∧ m.type = "UpdateReply" ∧ m.c = c   such m is unique due to well-formedness
105         ∧ cvc' = [cvc EXCEPT ![c][m.d] = m.ts]
106         ∧ msgs' = msgs \ {m}
107     ∧ UNCHANGED ⟨sVars, incoming⟩
108 ├──────────────────────────────────────────────────────────────────────────┤
109  Server operations at partition p ∈ Partition in datacenter d ∈ Datacenter.

111 ReadRequest(p, d) ≜   handle a "ReadRequest"
112     ∧ ∃ m ∈ msgs :
113         ∧ m.type = "ReadRequest" ∧ m.p = p ∧ m.d = d
114         ∧ css' = [css EXCEPT ![p][d] = Merge(m.vc, @)]
115         ∧ LET kvs ≜ {kv ∈ store[p][d] :
116                       ∧ kv.key = m.key
117                       ∧ ∀ dc ∈ Datacenter \ {d} : kv.vc[dc] ≤ css'[p][d][dc]}
118               lkv ≜ CHOOSE kv ∈ kvs :   choose the latest one (Existence? Uniqueness?)
119                       ∀ akv ∈ kvs, dc ∈ Datacenter : akv.vc[dc] ≤ kv.vc[dc]
120           IN  SendAndDelete([type ↦ "ReadReply", val ↦ lkv.val, vc ↦ lkv.vc, c ↦ m.c], m)
121     ∧ UNCHANGED ⟨cVars, clock, pvc, store, incoming⟩

123 UpdateRequest(p, d) ≜   handle a "UpdateRequest"
124     ∧ ∃ m ∈ msgs :
125         ∧ m.type = "UpdateRequest" ∧ m.p = p ∧ m.d = d
126         ∧ m.vc[d] < clock[p][d]   waiting condition; (" ≤ " strengthed to " < ")
127         ∧ css' = [css EXCEPT ![p][d] = Merge(m.vc, @)]
128         ∧ LET kv ≜ [key ↦ m.key, val ↦ m.val,
129                     vc ↦ [m.vc EXCEPT ![d] = clock[p][d]]]
130           IN  ∧ store' = [store EXCEPT ![p][d] = @ ∪ {kv}]
131               ∧ SendAndDelete([type ↦ "UpdateReply", ts ↦ clock[p][d], c ↦ m.c, d ↦ d], m)
132               ∧ incoming' = [incoming EXCEPT ![p] = [dc ∈ Datacenter ↦
133                   IF dc = d THEN @[dc] ELSE Append(@[dc], [type ↦ "Replicate", d ↦ d, kv ↦ kv])]]
134     ∧ UNCHANGED ⟨cVars, clock, pvc⟩

136 Replicate(p, d) ≜   handle a "Replicate"
137     ∧ incoming[p][d] ≠ ⟨⟩
138     ∧ LET m ≜ Head(incoming[p][d])
139       IN  ∧ m.type = "Replicate"
140           ∧ store'  = [store EXCEPT ![p][d] = @ ∪ {m.kv}]
141           ∧ 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)
        ∨ UpdateCSS(p, d)

Spec ≜ Init ∧ □[Next]_vars
```