```
1  ---------- MODULE AJupiter ----------
   Model checking the Jupiter protocol presented by Attiya and others.
5  EXTENDS OT, TLC
6  ----------
7  CONSTANTS
8      Client,     the set of client replicas
9      Server,     the (unique) server replica
10     State,      the initial state of each replica
11     Cop         Cop[c]: operations issued by the client c ∈ Client

13 ASSUME
14     ∧ State ∈ List
15     ∧ Cop ∈ [Client → Seq(Op)]

17 VARIABLES
18     cop,        cop[c]: operations issued by the client c ∈ Client
       For the client replicas:
22     cbuf,       cbuf[c]: buffer (of operations) at the client c ∈ Client
23     crec,       crec[c]: the number of new messages have been received by the client c ∈ Client
24                 since the last time a message was sent
25     cstate,     cstate[c]: state (the list content) of the client c ∈ Client

       For the server replica:
30     sbuf,       sbuf[c]: buffer (of operations) at the Server, one per client c ∈ Client
31     srec,       srec[c]: the number of new messages have been ..., one per client c ∈ Client
32     sstate,     sstate: state (the list content) of the server Server

       For communication between the Server and the Clients:
37     cincoming,  cincoming[c]: incoming channel at the client c ∈ Client
38     sincoming   incoming channel at the Server
39 ----------
40 comm ≜ INSTANCE CSComm
41 ----------
42 cVars ≜ ⟨cop, cbuf, crec, cstate⟩
43 sVars ≜ ⟨sbuf, srec, sstate⟩
44 vars ≜ cVars ∘ sVars ∘ comm!vars
45 ----------
46 TypeOK ≜
47     ∧   cop ∈ [Client → Seq(Op)]
       For the client replicas:
51     ∧ cbuf ∈ [Client → Seq(Op)]
52     ∧ crec ∈ [Client → Nat]
53     ∧ cstate ∈ [Client → List]
       For the server replica:
```

```
57      ∧ sbuf ∈ [Client → Seq(Op)]
58      ∧ srec  ∈ [Client → Nat]
59      ∧ sstate ∈ [Client → List]
        For communication between the server and the clients:
63      ∧ comm ! TypeOK
64  ----------------------------------------------------------------
        The Init predicate.
68  Init ≜
69      ∧ cop = Cop
        For the client replicas:
73      ∧ cbuf = [c ∈ Client ↦ ⟨⟩]
74      ∧ crec  = [c ∈ Client ↦ 0]
75      ∧ cstate = [c ∈ Client ↦ State]
        For the server replica:
79      ∧ sbuf = [c ∈ Client ↦ ⟨⟩]
80      ∧ srec  = [c ∈ Client ↦ 0]
81      ∧ sstate = [c ∈ Client ↦ State]
        For communication between the server and the clients:
85      ∧ comm ! Init
86  ----------------------------------------------------------------
        Client c ∈ Client issues an operation op.
90  Do(c) ≜
91      ∧ cop[c] ≠ ⟨⟩
92      ∧ LET op ≜ Head(cop[c])
93        IN  ∧ Print(op, TRUE)
94            ∧ cstate' = [cstate EXCEPT ![c] = Apply(op, @)]
95            ∧ cbuf' = [cbuf EXCEPT ![c] = Append(@, op)]
96            ∧ comm ! CSend([c ↦ c, ack ↦ crec[c], op ↦ op])
97      ∧ crec' = [crec EXCEPT ![c] = 0]
98      ∧ cop' = [cop EXCEPT ![c] = Tail(@)]
99      ∧ UNCHANGED sVars
100 ----------------------------------------------------------------
        Client c ∈ Client receives a message from the Server.
104 CRev(c) ≜
105   ∧ comm ! CRev(c)
106   ∧ crec' = [crec EXCEPT ![c] = @ + 1]
107   ∧ LET m ≜ Head(cincoming[c])
108       cBuf ≜ cbuf[c] \* the buffer at client c ∈ Client
109       cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf)) \* buffer shifted
110       xop ≜ XformOpOps(m.op, cShiftedBuf) \* transform op vs. shifted buffer
111       xcBuf ≜ XformOpsOp(cShiftedBuf, m.op) \* transform shifted buffer vs. op
112     IN  ∧ cbuf' = [cbuf EXCEPT ![c] = xcBuf]
113         ∧ cstate' = [cstate EXCEPT ![c] = Apply(xop, @)] \* apply the transformed operation xop
```

```
114     ∧ UNCHANGED ⟨sVars ∘ ⟨cop⟩⟩
115 ----------------------------------------------------------------------
    The Server receives a message.
119   SRev ≜
120     ∧ comm!SRev
121     ∧ LET m ≜ Head(sincoming) \* the message to handle with
122           c ≜ m.c              \* the client c ∈ Client that sends this message
123           cBuf ≜ sbuf[c]       \* the buffer at the Server for client c ∈ Client
124           cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf)) \* buffer shifted
125           xop ≜ XformOpOps(m.op, cShiftedBuf) \* transform op vs. shifted buffer
126           xcBuf ≜ XformOpsOp(cShiftedBuf, m.op) \* transform shifted buffer vs. op
127       IN  ∧ srec' = [cl ∈ Client ↦
128                       IF cl = c
129                       THEN srec[cl] + 1 \* receive one more operation from client c ∈ Client
130                       ELSE 0] \* reset srec for other clients than c ∈ Client
131           ∧ sbuf' = [cl ∈ Client ↦
132                       IF cl = c
133                       THEN xcBuf \* transformed buffer for client c ∈ Client
134                       ELSE Append(sbuf[cl], xop)] \* store transformed xop into other clients' bufs
135           ∧ sstate' = Apply(xop, sstate) \* apply the transformed operation
136           ∧ comm!SSend(c, srec, xop)
137     ∧ UNCHANGED cVars
138 ----------------------------------------------------------------------
    The Next state relation.
142   Next ≜
143       ∨ ∃ c ∈ Client : Do(c)
144     ∨ ∃ c ∈ Client: CRev(c)
145     ∨ SRev
    The Spec.
149   Spec ≜ Init ∧ □[Next]_vars
150 ======================================================================
    \* Modification History
    \* Last modified Sun Jul 01 20:29:10 CST 2018 by hengxin
    \* Created Sat Jun 23 17:14:18 CST 2018 by hengxin
```