```
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
       For model checking:
21     cop,        cop[c]: operations issued by the client c ∈ Client

       For the client replicas:
26     cbuf,       cbuf[c]: buffer (of operations) at the client c ∈ Client
27     crec,       crec[c]: the number of new messages have been received by the client c ∈ Client
28                 since the last time a message was sent
29     cstate,     cstate[c]: state (the list content) of the client c ∈ Client

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

       For communication between the Server and the Clients:
41     cincoming,  cincoming[c]: incoming channel at the client c ∈ Client
42     sincoming   incoming channel at the Server
43 ----------
44 comm ≜ INSTANCE CSComm
45 ----------
46 cVars ≜ ⟨cop, cbuf, crec, cstate⟩
47 sVars ≜ ⟨sbuf, srec, sstate⟩
48 vars ≜ cVars ∘ sVars ∘ comm!vars
49 ----------
50 TypeOK ≜
51     ∧ cop ∈ [Client → Seq(Op)]
       For the client replicas:
55     ∧ cbuf ∈ [Client → Seq(Op)]
56     ∧ crec ∈ [Client → Nat]
57     ∧ cstate ∈ [Client → List]
```

For the server replica:

$\land\ sbuf \in [Client \to Seq(Op)]$

$\land\ srec \in [Client \to Nat]$

$\land\ sstate \in List$

For communication between the server and the clients:

$\land\ comm\,!\,TypeOK$

The *Init* predicate.

$Init \triangleq$

$\land\ cop = Cop$

For the client replicas:

$\land\ cbuf = [c \in Client \mapsto \langle\rangle]$

$\land\ crec = [c \in Client \mapsto 0]$

$\land\ cstate = [c \in Client \mapsto State]$

For the server replica:

$\land\ sbuf = [c \in Client \mapsto \langle\rangle]$

$\land\ srec = [c \in Client \mapsto 0]$

$\land\ sstate = State$

For communication between the server and the clients:

$\land\ comm\,!\,Init$

Client $c \in Client$ issues an operation op.

$Do(c) \triangleq$

$\land\ cop[c] \neq \langle\rangle$

$\land$ LET $op \triangleq Head(cop[c])$

IN $\land\ Print(c,\ \text{TRUE})$

$\land\ Print(op,\ \text{TRUE})$

$\land\ cstate' = [cstate\ \text{EXCEPT}\ ![c] = Apply(op,\ @)]$

$\land\ cbuf' = [cbuf\ \text{EXCEPT}\ ![c] = Append(@,\ op)]$

$\land\ comm\,!\,CSend([c \mapsto c,\ ack \mapsto crec[c],\ op \mapsto op])$

$\land\ crec' = [crec\ \text{EXCEPT}\ ![c] = 0]$

$\land\ cop' = [cop\ \text{EXCEPT}\ ![c] = Tail(@)]$

$\land$ UNCHANGED $sVars$

Client $c \in Client$ receives a message from the *Server*.

$CRev(c) \triangleq$

$\land\ comm\,!\,CRev(c)$

$\land\ crec' = [crec\ \text{EXCEPT}\ ![c] = @ + 1]$

$\land$ LET $m \triangleq Head(cincoming[c])$

$cBuf \triangleq cbuf[c] \setminus *$ the buffer at client $c \in Client$

$cShiftedBuf \triangleq SubSeq(cBuf,\ m.ack + 1,\ Len(cBuf)) \setminus *$ buffer shifted

$xop \triangleq XformOpOps(m.op,\ cShiftedBuf) \setminus *$ transform op vs. shifted buffer

$xcBuf \triangleq XformOpsOp(cShiftedBuf,\ m.op) \setminus *$ transform shifted buffer vs. op

```
117         IN   ∧ cbuf' = [cbuf EXCEPT ![c] = xcBuf]
118              ∧ cstate' = [cstate EXCEPT ![c] = Apply(xop, @)] \* apply the transformed operation xop
119         ∧ UNCHANGED (sVars ∘ ⟨cop⟩)
120 ----------------------------------------------------------------------------
    The Server receives a message.
124 SRev ≜
125     ∧ comm!SRev
126     ∧ LET m ≜ Head(sincoming)        the message to handle with
127           c ≜ m.c                    the client c ∈ Client that sends this message
128           cBuf ≜ sbuf[c]             the buffer at the Server for client c ∈ Client
129           cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))   buffer shifted
130           xop ≜ XformOpOps(m.op, cShiftedBuf)   transform op vs. shifted buffer
131           xcBuf ≜ XformOpsOp(cShiftedBuf, m.op)   transform shifted buffer vs. op
132       IN  ∧ srec' = [cl ∈ Client ↦
133                       IF cl = c
134                       THEN srec[cl] + 1   receive one more operation from client c ∈ Client
135                       ELSE 0]   reset srec for other clients than c ∈ Client
136           ∧ sbuf' = [cl ∈ Client ↦
137                       IF cl = c
138                       THEN xcBuf   transformed buffer for client c ∈ Client
139                       ELSE Append(sbuf[cl], xop)]   store transformed xop into other clients' bufs
140           ∧ sstate' = Apply(xop, sstate)   apply the transformed operation
141           ∧ comm!SSend(c, srec, xop)
142     ∧ UNCHANGED cVars
143 ----------------------------------------------------------------------------
    The next-state relation.
147 Next ≜
148     ∨ ∃ c ∈ Client : Do(c)
149     ∨ SRev
    The Spec.
153 Spec ≜ Init ∧ □[Next]_vars
154 ============================================================================
    \* Modification History
    \* Last modified Mon Jul 02 11:04:50 CST 2018 by hengxin
    \* Created Sat Jun 23 17:14:18 CST 2018 by hengxin
```