```
──────────────────── MODULE AJupiterDo ────────────────────
Model checking the Jupiter protocol presented by Attiya and others.
EXTENDS OT, TLC
───────────────────────────────────────────────────────────
CONSTANTS
    Client,      the set of client replicas
    Server,      the (unique) server replica
    InitState,   the initial state of each replica
    Cop          Cop[c]: operations issued by the client c ∈ Client

ASSUME ∧ InitState ∈ List
       ∧ Cop ∈ [Client → Seq(Op)]

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

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

    For communication between the Server and the Clients:
    cincoming,  cincoming[c]: incoming channel at the client c ∈ Client
    sincoming   incoming channel at the Server
───────────────────────────────────────────────────────────
comm ≜ INSTANCE CSComm
───────────────────────────────────────────────────────────
cVars ≜ ⟨cop, cbuf, crec, cstate⟩
sVars ≜ ⟨sbuf, srec, sstate⟩
vars ≜ cVars ∘ sVars ∘ comm!vars
───────────────────────────────────────────────────────────
TypeOK ≜
    ∧  cop ∈ [Client → Seq(Op)]
    For the client replicas:
    ∧ cbuf ∈ [Client → Seq(Op ∪ {Nop})]
    ∧ crec ∈ [Client → Nat]
    ∧ cstate ∈ [Client → List]
    For the server replica:
      ∧ sbuf  ∈ [Client → Seq(Op ∪ {Nop})]
      ∧ srec  ∈ [Client → Nat]
      ∧ sstate ∈ [Client → List]
      For communication between the server and the clients:
      ∧ comm! TypeOK
├──────────────────────────────────────────────────────────────────┤
  The Init predicate.
  Init ≜
      ∧ cop = Cop
      For the client replicas:
      ∧ cbuf  = [c ∈ Client ↦ ⟨⟩]
      ∧ crec  = [c ∈ Client ↦ 0]
      ∧ cstate = [c ∈ Client ↦ InitState]
      For the server replica:
      ∧ sbuf  = [c ∈ Client ↦ ⟨⟩]
      ∧ srec  = [c ∈ Client ↦ 0]
      ∧ sstate = [c ∈ Client ↦ InitState]
      For communication between the server and the clients:
      ∧ comm! Init
├──────────────────────────────────────────────────────────────────┤
  Client c ∈ Client issues an operation op.
  Do(c) ≜
        ∧ Print("Do", TRUE)
        ∧ cop[c] ≠ ⟨⟩
        ∧ LET op ≜ Head(cop[c])
            IN   ∧ PrintT(op)
                 ∧ cstate' = [cstate EXCEPT ![c] = Apply(op, @)]
                 ∧ cbuf' = [cbuf EXCEPT ![c] = Append(@, op)]
                 ∧ comm! CSend([c ↦ c, ack ↦ crec[c], op ↦ op])
        ∧ crec' = [crec EXCEPT ![c] = 0]
        ∧ cop' = [cop EXCEPT ![c] = Tail(@)]
        ∧ UNCHANGED sVars
├──────────────────────────────────────────────────────────────────┤
  The Next state relation.
  Next ≜
      ∨ ∃ c ∈ Client : Do(c)
  The Spec.
  Spec ≜ Init ∧ □[Next]_vars
└──────────────────────────────────────────────────────────────────┘
  \* Modification History
  \* Last modified Sat Jul 07 14:21:06 CST 2018 by hengxin
  \* Created Sun Jul 01 18:53:30 CST 2018 by hengxin
```