MODULE *AJupiterDo*

Model checking the *Jupiter* protocol presented by *Attiya* and others.

EXTENDS Op, TLC

CONSTANTS

$Client$, the set of client replicas

$Server$, the (unique) server replica

$State$, the initial state of each replica

Cop $Cop[c]$: operations issued by the client $c \in Client$

ASSUME

$\land State \in List$

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

VARIABLES

cop, $cop[c]$: operations issued by the client $c \in Client$

For the client replicas:

$cbuf$, $cbuf[c]$: buffer (of operations) at the client $c \in Client$

$crec$, $crec[c]$: the number of new messages have been received by the client $c \in Client$ since the last time a message was sent

$cstate$, $cstate[c]$: state (the list content) of the client $c \in Client$

For the server replica:

$sbuf$, $sbuf[c]$: buffer (of operations) at the $Server$, one per client $c \in Client$

$srec$, $srec[c]$: the number of new messages have been ..., one per client $c \in Client$

$sstate$, $sstate$: state (the list content) of the server $Server$

For communication between the $Server$ and the Clients:

$cincoming$, $cincoming[c]$: incoming channel at the client $c \in Client$

$sincoming$ incoming channel at the $Server$

$comm \triangleq$ INSTANCE $CSComm$

$cVars \triangleq \langle cop, cbuf, crec, cstate \rangle$

$sVars \triangleq \langle sbuf, srec, sstate \rangle$

$vars \triangleq cVars \circ sVars \circ comm!vars$

$TypeOK \triangleq$

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

For the client replicas:

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

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

$\land cstate \in [Client \to List]$

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      ∧ Print("Do", TRUE)
92      ∧ cop[c] ≠ ⟨⟩
93      ∧ LET op ≜ Head(cop[c])
94        IN  ∧ Print(op, TRUE)
95            ∧ cstate' = [cstate EXCEPT ![c] = Apply(op, @)]
96            ∧ cbuf' = [cbuf EXCEPT ![c] = Append(@, op)]
97            ∧ comm ! CSend([c ↦ c, ack ↦ crec[c], op ↦ op])
98      ∧ crec' = [crec EXCEPT ![c] = 0]
99      ∧ cop' = [cop EXCEPT ![c] = Tail(@)]
100     ∧ UNCHANGED sVars
101 ----------------------------------------------------------------
        The Next state relation.
105 Next ≜
106     ∨ ∃ c ∈ Client : Do(c)
        The Spec.
110 Spec ≜ Init ∧ □[Next]_vars
111 ================================================================
    \* Modification History
    \* Last modified Sun Jul 01 20:22:17 CST 2018 by hengxin
    \* Created Sun Jul 01 18:53:30 CST 2018 by hengxin
```