MODULE *AJupiter*

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

EXTENDS OT

CONSTANTS

$\quad Client,$ the set of client replicas
$\quad Server$ the (unique) server replica

VARIABLES

For the client replicas:

$\quad cbuf,$ $cbuf[c]$: buffer (of operations) at the client $c \in Client$
$\quad crec,$ $crec[c]$: the number of new messages have been received by the client $c \in Client$ since the last time a message was sent
$\quad cstate,$ $cstate[c]$: state (the list content) of the client $c \in Client$

For the server replica:

$\quad sbuf,$ $sbuf[c]$: buffer (of operations) at the *Server*, one per client $c \in Client$
$\quad srec,$ $srec[c]$: the number of new messages have been …, one per client $c \in Client$
$\quad sstate,$ $sstate$: state (the list content) of the server *Server*

For communication between the *Server* and the Clients:

$\quad cincoming,$ $cincoming[c]$: incoming channel at the client $c \in Client$
$\quad sincoming$ incoming channel at the *Server*

$cVars \triangleq \langle cbuf, crec, cstate \rangle$
$sVars \triangleq \langle sbuf, srec, sstate \rangle$
$commVars \triangleq \langle cincoming, sincoming \rangle$
$vars \triangleq cVars \circ sVars \circ commVars$

Messages between the *Server* and the Clients. There are two kinds of messages according to their destinations.

$Msg \triangleq [c : Client, ack : Nat, op : Op]$ messages sent to the *Server* from a client $c \in Client$
$\qquad \cup [ack : Nat, op : Op]$ messages broadcast to Clients from the *Server*

$TypeOK \triangleq$

For the client replicas:

$\quad \land\ cbuf \in [Client \rightarrow Seq(Op)]$
$\quad \land\ crec \in [Client \rightarrow Nat]$
$\quad \land\ cstate \in [Client \rightarrow List]$

For the server replica:

$\quad \land\ sbuf \in [Client \rightarrow Seq(Op)]$
$\quad \land\ srec \in [Client \rightarrow Nat]$
$\quad \land\ sstate \in [Client \rightarrow List]$

For communication between the server and the clients:

```
    $\land\ cincoming \in [Client \rightarrow Seq(Msg)]$
    $\land\ sincoming \in Seq(Msg)$
------------------------------------------------------------------------
The $Init$ predicate.
$Init \triangleq$
    For the client replicas:
    $\land\ cbuf = [c \in Client \mapsto \langle\rangle]$
    $\land\ crec\ = [c \in Client \mapsto 0]$
    $\land\ cstate = [c \in Client \mapsto \langle\rangle]$
    For the server replica:
    $\land\ sbuf = [c \in Client \mapsto \langle\rangle]$
    $\land\ srec\ = [c \in Client \mapsto 0]$
    $\land\ sstate = [c \in Client \mapsto \langle\rangle]$
    For communication between the server and the clients:
    $\land\ cincoming = [c \in Client \mapsto \langle\rangle]$
    $\land\ sincoming = \langle\rangle$
------------------------------------------------------------------------
A client sends a message $msg$ to the $Server$.
$CSend(msg) \triangleq\ \land\ sincoming' = Append(sincoming, msg)$
------------------------------------------------------------------------
Client $c \in Client$ generates and performs an operation $op$.
$Do(c, op) \triangleq$
     $\land$ TRUE        no pre-condition
     $\land\ cstate' = [cstate$ EXCEPT $![c] = Apply(op, @)]$
     $\land\ cbuf' = [cbuf$ EXCEPT $![c] = Append(@, op)]$
     $\land\ CSend([c \mapsto c, ack \mapsto crec[c], op \mapsto op])$
     $\land\ crec' = [crec$ EXCEPT $![c] = 0]$
     $\land$ UNCHANGED $(sVars \circ \langle cincoming\rangle)$
------------------------------------------------------------------------
Client $c \in Client$ receives a message from the $Server$.
$CRev(c) \triangleq$
     $\land\ cincoming[c] \neq \langle\rangle$     there are messages to handle with
     $\land\ crec' = [crec$ EXCEPT $![c] = @ + 1]$
     $\land$ LET $m \triangleq Head(cincoming[c])$
             $cBuf \triangleq cbuf[c]$   the buffer at client $c \in Client$
             $cShiftedBuf \triangleq SubSeq(cBuf, m.ack + 1, Len(cBuf))$   buffer shifted
             $xop \triangleq XformOpOps(m.op, cShiftedBuf)$ transform $op$ vs. shifted buffer
              $xcBuf \triangleq XformOpsOp(cShiftedBuf, m.op)$ transform shifted buffer vs. $op$
         IN  $\land\ cbuf' = [cbuf$ EXCEPT $![c] = xcBuf]$
              $\land\ cstate' = [cstate$ EXCEPT $![c] = Apply(xop, @)]$   apply the transformed operation $xop$
     $\land\ cincoming' = [cincoming$ EXCEPT $![c] = Tail(@)]$
     $\land$ UNCHANGED $(sVars \circ \langle sincoming\rangle)$
------------------------------------------------------------------------
```

```
The Server receives a message.
SRev ≜
     ∧ sincoming ≠ ⟨⟩        there are messages for the Server to handle with
     ∧ LET m ≜ Head(sincoming)  the message to handle with
           c ≜ m.c              the client c ∈ Client that sends this message
           cBuf ≜ sbuf[c]       the buffer at the Server for client c ∈ Client
           cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))  buffer shifted
           xop ≜ XformOpOps(m.op, cShiftedBuf)  transform op vs. shifted buffer
            xcBuf ≜ XformOpsOp(cShiftedBuf, m.op)  transform shifted buffer vs. op
        IN    ∧ srec′ = [cl ∈ Client ↦
                             IF cl = c
                              THEN srec[cl] + 1  receive one more operation from client c ∈ Client
                              ELSE 0]  reset srec for other clients than c ∈ Client
              ∧ sbuf′ = [cl ∈ Client ↦
                             IF cl = c
                              THEN xcBuf  transformed buffer for client c ∈ Client
                              ELSE Append(sbuf[cl], xop)]  store transformed xop into other clients' bufs
              ∧ cincoming′ = [cl ∈ Client ↦
                               IF cl = c
                                THEN cincoming[cl]
                                broadcast the transformed operation to clients other than c ∈ Client
                                ELSE Append(cincoming[cl], [ack ↦ srec[cl], op ↦ xop])]
              ∧ sstate′ = Apply(xop, sstate)  apply the transformed operation
     ∧ sincoming′ = Tail(sincoming)  consume a message
     ∧ UNCHANGED cVars
────────────────────────────────────────────────────────────────────────────
The Next state relation.
Next ≜
     ∨ ∃ c ∈ Client, op ∈ Op : Do(c, op)
     ∨ ∃ c ∈ Client : CRev(c)
     ∨ SRev
The Spec.
Spec ≜ Init ∧ □[Next]_vars
────────────────────────────────────────────────────────────────────────────
\* Modification History
\* Last modified Sun Jun 24 22:26:35 CST 2018 by hengxin
\* Created Sat Jun 23 17:14:18 CST 2018 by hengxin
```