MODULE *AJupiter*

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

EXTENDS *Op*

CONSTANTS

Client, the set of client replicas

Server the (unique) server replica

VARIABLES

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*

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

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

$commVars \triangleq \langle cincoming, sincoming \rangle$

$vars \triangleq cVars \circ sVars \circ commVars$

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 $c \in Client$

$\cup\, [ack : Nat, op : Op]$ messages broadcast to Clients from the *Server*

$TypeOK \triangleq$

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:

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

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

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

```
        For communication between the server and the clients:
63      ∧ cincoming ∈ [Client → Seq(Msg)]
64      ∧ sincoming ∈ Seq(Msg)
65  ----------------------------------------------------------------
    The Init predicate.
69  Init ≜
        For the client replicas:
73      ∧ cbuf = [c ∈ Client ↦ ⟨⟩]
74      ∧ crec = [c ∈ Client ↦ 0]
75      ∧ cstate = [c ∈ Client ↦ ⟨⟩]
        For the server replica:
79      ∧ sbuf = [c ∈ Client ↦ ⟨⟩]
80      ∧ srec = [c ∈ Client ↦ 0]
81      ∧ sstate = [c ∈ Client ↦ ⟨⟩]
        For communication between the server and the clients:
85      ∧ cincoming = [c ∈ Client ↦ ⟨⟩]
86      ∧ sincoming = ⟨⟩
87  ----------------------------------------------------------------
    A client sends a message msg to the Server.
91  CSend(msg) ≜  ∧ sincoming′ = Append(sincoming, msg)

    The Server broadcast a message msg to the Clients other than c ∈ Client.
97  SBoradcast(c, msg) ≜
98      ∧ cincoming′ = [cl ∈ Client ↦
99                        IF cl = c
100                       THEN cincoming[cl]
101                       ELSE Append(cincoming[cl], msg)]
102 ----------------------------------------------------------------
    Client c ∈ Client generates and performs an operation op.
106 Do(c, op) ≜
107     ∧ TRUE      no pre-condition
108     ∧ cstate′ = [cstate EXCEPT ![c] = Apply(op, @)]
109     ∧ cbuf′ = [cbuf EXCEPT ![c] = Append(@, op)]
110     ∧ CSend([c ↦ c, ack ↦ crec[c], op ↦ op])
111     ∧ crec′ = [crec EXCEPT ![c] = 0]
112     ∧ UNCHANGED (sVars ∘ ⟨cincoming⟩)
113 ----------------------------------------------------------------
    Client c ∈ Client receives a message msg from the Server.
117 CRev(c, msg) ≜
118     ∧ cincoming[c] ≠ ⟨⟩    there are messages to handle with
119     ∧ crec′ = [crec EXCEPT ![c] = @ + 1]
120     ∧ LET m ≜ Head(cincoming[c])
121       IN  ∧ cbuf′ = [cbuf EXCEPT ![c] = SubSeq(@, m.ack + 1, Len(@))]
```

```
                    ∧ cincoming′ = [cincoming EXCEPT ![c] = Tail(@)]
        ∧ FALSE  TODO: (buf, o) = xform(buf, o)
         ∧ cstate′ = [cstate EXCEPT ![c] = Apply(m.op, @)]  \* using o above
        ∧ UNCHANGED ⟨sVars ∘ ⟨sincoming⟩⟩
```

The *Server* receives a message *msg*.

```
SRev(msg) ≜
        ∧ sincoming ≠ ⟨⟩      there are messages for the Server to handle with
        ∧ LET m ≜ Head(sincoming)
                c ≜ m.c
          IN    ∧ srec′ = [cl ∈ Client ↦
                             IF cl = c
                              THEN srec[cl] + 1
                              ELSE 0]
                ∧ sbuf′ = [sbuf EXCEPT ![c] = SubSeq(@, m.ack + 1, Len(@))]
        ∧ sincoming′ = Tail(sincoming)
        ∧ FALSE  TODO: (o, buf[c]) = xform(o, buf[c])
         ∧ sstate′ = Apply(m.op, sstate)  \* using o above
        ∧ FALSE  for all other clients
        ∧ UNCHANGED cVars
```

The next state relation.

```
Next ≜ FALSE
```

```
\* Modification History
\* Last modified Sun Jun 24 15:29:26 CST 2018 by hengxin
\* Created Sat Jun 23 17:14:18 CST 2018 by hengxin
```