```
┌──────────────────────── MODULE AJupiter ────────────────────────┐
 Specification of the Jupiter protocol presented by Hagit Attiya and others.
EXTENDS JupiterInterface
├─────────────────────────────────────────────────────────────────┤
VARIABLES
    cbuf,     cbuf[c]: buffer for locally generated operations at client c ∈ Client
    crec,     crec[c]: number of remote operations received by client c ∈ Client
                       since the last time a local operation was generated
    sbuf,     sbuf[c]: buffer for transformed remote operations w.r.t client c ∈ Client
    srec      srec[c]: number of locally generated operations by client c ∈ Client
                       since the last time a remote operation was transformed at the Server

vars ≜ ⟨intVars, cbuf, crec, sbuf, srec⟩

Msg ≜ [c : Client, ack : Int, op : Op ∪ {Nop}] ∪  messages sent to the Server from a client c ∈ Client
         [ack : Int, op : Op ∪ {Nop}]  messages broadcast to Clients from the Server
├─────────────────────────────────────────────────────────────────┤
TypeOK ≜
    ∧  TypeOKInt
    ∧  Comm(Msg)!TypeOK
    ∧  cbuf ∈ [Client → Seq(Op ∪ {Nop})]
    ∧  crec ∈ [Client → Int]
    ∧  sbuf ∈ [Client → Seq(Op ∪ {Nop})]
    ∧  srec ∈ [Client → Int]
├─────────────────────────────────────────────────────────────────┤
Init ≜
    ∧ InitInt
    ∧ Comm(Msg)!Init
    ∧ cbuf = [c ∈ Client ↦ ⟨⟩]
    ∧ crec = [c ∈ Client ↦ 0]
    ∧ sbuf = [c ∈ Client ↦ ⟨⟩]
    ∧ srec = [c ∈ Client ↦ 0]
├─────────────────────────────────────────────────────────────────┤
 Client c ∈ Client issues an operation op.
DoOp(c, op) ≜
        ∧ state′ = [state EXCEPT ![c] = Apply(op, @)]
        ∧ cbuf′ = [cbuf EXCEPT ![c] = Append(@, op)]
        ∧ crec′ = [crec EXCEPT ![c] = 0]
        ∧ Comm(Msg)!CSend([c ↦ c, ack ↦ crec[c], op ↦ op])

DoIns(c) ≜
    ∃ ins ∈ {op ∈ Ins : op.pos ∈ 1 .. (Len(state[c]) + 1) ∧ op.ch ∈ chins ∧ op.pr = Priority[c]} :
        ∧ DoOp(c, ins)
        ∧ chins′ = chins \ {ins.ch}

DoDel(c) ≜
```

```
    $\exists\, del \in \{op \in Del : op.pos \in 1 \,..\, Len(state[c])\} :$
        $\land\ DoOp(c,\ del)$
        $\land$ UNCHANGED $chins$

$Do(c) \triangleq$
      $\land\ \lor\ DoIns(c)$
         $\lor\ DoDel(c)$
      $\land$ UNCHANGED $\langle sbuf,\ srec \rangle$
--------------------------------------------------------------------------------
```

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

```
$Rev(c) \triangleq$
      $\land\ Comm(Msg)!CRev(c)$
      $\land\ crec' = [crec$ EXCEPT $![c] = @ + 1]$
      $\land$ LET $m \triangleq Head(cincoming[c])$
              $cBuf \triangleq cbuf[c]$
              $cShiftedBuf \triangleq SubSeq(cBuf,\ m.ack + 1,\ Len(cBuf))$
              $xop \triangleq XformOpOps(Xform,\ m.op,\ cShiftedBuf)$
               $xcBuf \triangleq XformOpsOp(Xform,\ cShiftedBuf,\ m.op)$
          IN   $\land\ cbuf' = [cbuf$ EXCEPT $![c] = xcBuf]$
               $\land\ state' = [state$ EXCEPT $![c] = Apply(xop,\ @)]$
      $\land$ UNCHANGED $\langle chins,\ sbuf,\ srec \rangle$
--------------------------------------------------------------------------------
```

The *Server* receives a message.

```
$SRev \triangleq$
     $\land\ Comm(Msg)!SRev$
     $\land$ LET $m \triangleq Head(sincoming)$
             $c \triangleq m.c$
             $cBuf \triangleq sbuf[c]$
             $cShiftedBuf \triangleq SubSeq(cBuf,\ m.ack + 1,\ Len(cBuf))$
             $xop \triangleq XformOpOps(Xform,\ m.op,\ cShiftedBuf)$
              $xcBuf \triangleq XformOpsOp(Xform,\ cShiftedBuf,\ m.op)$
         IN   $\land\ srec' = [cl \in Client \mapsto$
                              IF $cl = c$ THEN $srec[cl] + 1$ ELSE $0]$
              $\land\ sbuf' = [cl \in Client \mapsto$
                              IF $cl = c$ THEN $xcBuf$ ELSE $Append(sbuf[cl],\ xop)]$
              $\land\ state' = [state$ EXCEPT $![Server] = Apply(xop,\ @)]$
              $\land\ Comm(Msg)!SSend(c,\ [cl \in Client \mapsto [ack \mapsto srec[cl],\ op \mapsto xop]])$
     $\land$ UNCHANGED $\langle chins,\ cbuf,\ crec \rangle$
--------------------------------------------------------------------------------
$Next \triangleq$
     $\lor\ \exists\, c \in Client : Do(c) \lor Rev(c)$
     $\lor\ SRev$

$Fairness \triangleq$   There is no requirement that the clients ever generate operations.
    $\mathrm{WF}_{vars}(SRev \lor \exists\, c \in Client : Rev(c))$
```

$Spec \triangleq Init \land \Box[Next]_{vars}$ $\land\ Fairness$

$QC \triangleq$ Quiescent Consistency

$\quad Comm(Msg)!EmptyChannel \Rightarrow Cardinality(Range(state)) = 1$

THEOREM $Spec \Rightarrow \Box QC$

\* Modification History

\* Last modified Sun *Dec* 30 16:02:35 *CST* 2018 by *hengxin*

\* Created Sat *Jun* 23 17:14:18 *CST* 2018 by *hengxin*