

```

1  |----- MODULE AJupiter -----|
   | Specification of the Jupiter protocol presented by Hagit Attiya and others. |
5  | EXTENDS JupiterInterface |
6  |-----|
7  VARIABLES
8      cbuf,      cbuf[c]: buffer for locally generated operations at client c ∈ Client
9      crec,      crec[c]: number of remote operations received by client c ∈ Client
10                     since the last time a local operation was generated
11      sbuf,      sbuf[c]: buffer for transformed remote operations w.r.t client c ∈ Client
12      srec      srec[c]: number of locally generated operations by client c ∈ Client
13                     since the last time a remote operation was transformed at the Server
15  vars ≜ ⟨intVars, cbuf, crec, sbuf, srec⟩
17  Msg ≜ [c : Client, ack : Nat, op : Op ∪ {Nop}] ∪ messages sent to the Server from a client c ∈ Client
18          [ack : Nat, op : Op ∪ {Nop}] messages broadcast to Clients from the Server
19  |-----|
20  TypeOK ≜
21      ∧ TypeOKInt
22      ∧ Comm(Msg)! TypeOK
23      ∧ cbuf ∈ [Client → Seq(Op ∪ {Nop})]
24      ∧ crec ∈ [Client → Nat]
25      ∧ sbuf ∈ [Client → Seq(Op ∪ {Nop})]
26      ∧ srec ∈ [Client → Nat]
27  |-----|
28  Init ≜
29      ∧ InitInt
30      ∧ Comm(Msg)! Init
31      ∧ cbuf = [c ∈ Client ↦ ⟨⟩]
32      ∧ crec = [c ∈ Client ↦ 0]
33      ∧ sbuf = [c ∈ Client ↦ ⟨⟩]
34      ∧ srec = [c ∈ Client ↦ 0]
35  |-----|
36  DoOp(c, op) ≜
37      ∧ SetNewAop(c, op)
38      ∧ cbuf' = [cbuf EXCEPT ![c] = Append(@, op)]
39      ∧ crec' = [crec EXCEPT ![c] = 0]
40      ∧ Comm(Msg)! CSend([c ↦ c, ack ↦ crec[c], op ↦ op])
42  Do(c) ≜
43      ∧ DoInt(DoOp, c)
44      ∧ UNCHANGED ⟨sbuf, srec⟩
46  Rev(c) ≜
47      ∧ Comm(Msg)! CRev(c)
48      ∧ crec' = [crec EXCEPT ![c] = @ + 1]

```

```

49    $\wedge \text{LET } m \triangleq \text{Head}(\text{cincoming}[c])$ 
50    $cBuf \triangleq cbuf[c]$ 
51    $cShiftedBuf \triangleq \text{SubSeq}(cBuf, m.ack + 1, \text{Len}(cBuf))$ 
52    $xop \triangleq \text{XformOpsOps}(Xform, m.op, cShiftedBuf)$ 
53    $xcBuf \triangleq \text{XformOpsOp}(Xform, cShiftedBuf, m.op)$ 
54   IN  $\wedge cbuf' = [cbuf \text{ EXCEPT } ![c] = xcBuf]$ 
55    $\wedge \text{SetNewAop}(c, xop)$ 
56    $\wedge \text{RevInt}(c)$ 
57    $\wedge \text{UNCHANGED } \langle sbuf, srec \rangle$ 

59  $SRev \triangleq$ 
60    $\wedge \text{Comm}(Msg)!SRev$ 
61    $\wedge \text{LET } m \triangleq \text{Head}(\text{sincoming})$ 
62    $c \triangleq m.c$ 
63    $cBuf \triangleq sbuf[c]$ 
64    $cShiftedBuf \triangleq \text{SubSeq}(cBuf, m.ack + 1, \text{Len}(cBuf))$ 
65    $xop \triangleq \text{XformOpsOps}(Xform, m.op, cShiftedBuf)$ 
66    $xcBuf \triangleq \text{XformOpsOp}(Xform, cShiftedBuf, m.op)$ 
67   IN  $\wedge srec' = [cl \in Client \mapsto$ 
68    $\quad \text{IF } cl = c \text{ THEN } srec[cl] + 1 \text{ ELSE } 0]$ 
69    $\wedge sbuf' = [cl \in Client \mapsto$ 
70    $\quad \text{IF } cl = c \text{ THEN } xcBuf \text{ ELSE } \text{Append}(sbuf[cl], xop)]$ 
71    $\wedge \text{SetNewAop}(\text{Server}, xop)$ 
72    $\wedge \text{Comm}(Msg)!SSend(c, [cl \in Client \mapsto [ack \mapsto srec[cl], op \mapsto xop]])$ 
73    $\wedge SRevInt$ 
74    $\wedge \text{UNCHANGED } \langle cbuf, crec \rangle$ 

75 |-----|
76  $Next \triangleq$ 
77    $\vee \exists c \in Client : Do(c) \vee Rev(c)$ 
78    $\vee SRev$ 

80  $Fairness \triangleq$  There is no requirement that the clients ever generate operations.
81    $\text{WF}_{vars}(SRev \vee \exists c \in Client : Rev(c))$ 

83  $Spec \triangleq \text{Init} \wedge \Box[Next]_{vars} \wedge \text{Fairness}$ 
84 |-----|
85  $QC \triangleq$  Quiescent Consistency
86    $\text{Comm}(Msg)!EmptyChannel \Rightarrow \text{Cardinality}(\text{Range}(\text{state})) = 1$ 

88 THEOREM  $Spec \Rightarrow \Box QC$ 
89 |-----|

\ * Modification History
\ * Last modified Wed Jan 02 14:25:36 CST 2019 by hengxin
\ * Created Satchins, Jun 23 17:14:18 CST 2018 by hengxin

```