

```

1 |----- MODULE AJupiter -----|
  |Model checking the Jupiter protocol presented by Attiya and others.
5 | EXTENDS OT, TLC
6 |-----|
7 | CONSTANTS
8 |   Client,      the set of client replicas
9 |   Server,      the (unique) server replica
10 |  InitState,    the initial state of each replica
11 |  Priority      Priority[c]: the priority value of client c ∈ Client
12 |  Cop          \ * Cop[c]: operations issued by the client c ∈ Client

14 | ASSUME
15 |   ∧ InitState ∈ List
16 |   ∧ Priority ∈ [Client → PosInt]
17 |   ∧ Cop ∈ [Client → Seq(Op)]

  |Generate operations for AJupiter clients.
  |Note: Remember to override the definition of PosInt.
  |FIXME: PosInt ⇒ MaxPos; MaxPr determined by the size of Client.

26 | OpToIssue ≜ {opset ∈ SUBSET Op :
27 |               ∀ op1, op2 ∈ opset :
28 |               (op1.type = "Ins" ∧ op2.type = "Ins") ⇒ op1.ch ≠ op2.ch}

30 | VARIABLES
  |For model checking:
34 |  cop,          \ * cop[c]: operations issued by the client c ∈ Client
35 |  cop,          a set of operations for clients to issue

  |For the client replicas:
40 |  cbuf,         cbuf[c]: buffer (of operations) at the client c ∈ Client
41 |  crec,         crec[c]: the number of new messages have been received by the client c ∈ Client
42 |               since the last time a message was sent
43 |  cstate,       cstate[c]: state (the list content) of the client c ∈ Client

  |For the server replica:
48 |  sbuf,         sbuf[c]: buffer (of operations) at the Server, one per client c ∈ Client
49 |  srec,         srec[c]: the number of new messages have been ... , one per client c ∈ Client
50 |  sstate,       sstate: state (the list content) of the server Server

  |For communication between the Server and the Clients:
55 |  cincoming,   cincoming[c]: incoming channel at the client c ∈ Client
56 |  sincoming,   incoming channel at the Server

57 |-----|
58 | comm ≜ INSTANCE CSComm
59 |-----|

```

```

60  $cVars \triangleq \langle cop, cbuf, crec, cstate \rangle$ 
61  $sVars \triangleq \langle sbuf, srec, sstate \rangle$ 
62 FIXME: subscript error (Don't know why yet!)
63  $vars \triangleq cVars \circ sVars \circ \langle cincoming, sincoming \rangle$ 
64  $jVars \triangleq \langle cbuf, crec, cstate, sbuf, srec, sstate, cincoming, sincoming \rangle$  all variables
65  $vars \triangleq \langle cop, cbuf, crec, cstate, sbuf, srec, sstate, cincoming, sincoming \rangle$  all variables
66 |-----|
67  $TypeOK \triangleq$ 
68    $\wedge cop \in [Client \rightarrow Seq(Op)]$ 
69    $\wedge cop \in \text{SUBSET } Op$ 
70   For the client replicas:
71    $\wedge cbuf \in [Client \rightarrow Seq(Op \cup \{Nop\})]$ 
72    $\wedge crec \in [Client \rightarrow Nat]$ 
73    $\wedge cstate \in [Client \rightarrow List]$ 
74   For the server replica:
75    $\wedge sbuf \in [Client \rightarrow Seq(Op \cup \{Nop\})]$ 
76    $\wedge srec \in [Client \rightarrow Nat]$ 
77    $\wedge sstate \in List$ 
78   For communication between the server and the clients:
79    $\wedge comm!TypeOK$ 
80 |-----|
81 The Init predicate.
82  $Init \triangleq$ 
83    $\wedge cop = Cop$ 
84    $\wedge cop \in OpToIssue$ 
85   For the client replicas:
86    $\wedge cbuf = [c \in Client \mapsto \langle \rangle]$ 
87    $\wedge crec = [c \in Client \mapsto 0]$ 
88    $\wedge cstate = [c \in Client \mapsto InitState]$ 
89   For the server replica:
90    $\wedge sbuf = [c \in Client \mapsto \langle \rangle]$ 
91    $\wedge srec = [c \in Client \mapsto 0]$ 
92    $\wedge sstate = InitState$ 
93   For communication between the server and the clients:
94    $\wedge comm!Init$ 
95 |-----|
96  $LegalizeOp(op, c) \triangleq$ 
97   LET  $len \triangleq Len(cstate[c])$ 
98   IN CASE  $op.type = "Del" \rightarrow$ 
99     IF  $len = 0$  THEN  $Nop$  ELSE  $[op \text{ EXCEPT } !.pos = Min(@, len)]$ 
100   □  $op.type = "Ins" \rightarrow$ 
101      $[op \text{ EXCEPT } !.pos = Min(@, len + 1), !.pr = Priority[c]]$ 
102 Client  $c \in Client$  issues an operation  $op$ .

```

```

119  $Do(c) \triangleq$ 
120    $\wedge cop[c] \neq \langle \rangle$ 
121    $\wedge cop \neq \{\}$ 
122    $\wedge LET\ o \triangleq CHOOSE\ x \in cop : TRUE$ 
123    $op \triangleq LegalizeOp(o, c)$  preprocess an illegal operation
124   IN  $\vee \wedge op = Nop$ 
125      $\wedge cop' = cop \setminus \{o\}$  consume one operation
126      $\wedge UNCHANGED\ jVars$ 
127    $\vee \wedge op \neq Nop$ 
128      $\wedge PrintT(c \circ ": Do " \circ ToString(op))$ 
129      $\wedge cstate' = [cstate\ EXCEPT\ ![c] = Apply(op, @)]$ 
130      $\wedge cbuf' = [cbuf\ EXCEPT\ ![c] = Append(@, op)]$ 
131      $\wedge crec' = [crec\ EXCEPT\ ![c] = 0]$ 
132      $\wedge comm! CSend([c \mapsto c, ack \mapsto crec[c], op \mapsto op])$ 
133      $\wedge cop' = cop \setminus \{o\}$  consume one operation
134      $\wedge UNCHANGED\ sVars$ 
135    $\wedge cop' = [cop\ EXCEPT\ ![c] = Tail(@)] \setminus * \text{ consume one operation}$ 

```

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

```

140  $Rev(c) \triangleq$ 
141    $\wedge comm! CRev(c)$ 
142    $\wedge crec' = [crec\ EXCEPT\ ![c] = @ + 1]$ 
143    $\wedge LET\ m \triangleq Head(cincoming[c])$ 
144    $cBuf \triangleq cbuf[c]$  the buffer at client  $c \in Client$ 
145    $cShiftedBuf \triangleq SubSeq(cBuf, m.ack + 1, Len(cBuf))$  buffer shifted
146    $xop \triangleq XformOpOps(m.op, cShiftedBuf)$  transform  $op$  vs. shifted buffer
147    $xcBuf \triangleq XformOpsOp(cShiftedBuf, m.op)$  transform shifted buffer vs.  $op$ 
148   IN  $\wedge cbuf' = [cbuf\ EXCEPT\ ![c] = xcBuf]$ 
149    $\wedge cstate' = [cstate\ EXCEPT\ ![c] = Apply(xop, @)]$  apply the transformed operation  $xop$ 
150    $\wedge UNCHANGED\ \langle sbuf, srec, sstate, cop \rangle$  NOTE:  $sVars \circ \langle cop \rangle$  is wrong!

```

The *Server* receives a message.

```

155  $SRev \triangleq$ 
156    $\wedge comm! SRev$ 
157    $\wedge LET\ m \triangleq Head(sincoming)$  the message to handle with
158    $c \triangleq m.c$  the client  $c \in Client$  that sends this message
159    $cBuf \triangleq sbuf[c]$  the buffer at the Server for client  $c \in Client$ 
160    $cShiftedBuf \triangleq SubSeq(cBuf, m.ack + 1, Len(cBuf))$  buffer shifted
161    $xop \triangleq XformOpOps(m.op, cShiftedBuf)$  transform  $op$  vs. shifted buffer
162    $xcBuf \triangleq XformOpsOp(cShiftedBuf, m.op)$  transform shifted buffer vs.  $op$ 
163   IN  $\wedge srec' = [cl \in Client \mapsto$ 
164     IF  $cl = c$ 
165     THEN  $srec[cl] + 1$  receive one more operation from client  $c \in Client$ 
166     ELSE  $0]$  reset  $srec$  for other clients than  $c \in Client$ 
167    $\wedge sbuf' = [cl \in Client \mapsto$ 

```

```

168             IF  $cl = c$ 
169                 THEN  $xcBuf$  transformed buffer for client  $c \in Client$ 
170                 ELSE  $Append(sbuf[cl], xop)$  store transformed  $xop$  into other clients' bufs
171              $\wedge sstate' = Apply(xop, sstate)$  apply the transformed operation
172              $\wedge comm!SSend(c, srec, xop)$ 
173          $\wedge$  UNCHANGED  $cVars$ 
174 |-----|
175 | The next-state relation.
176 |-----|
177  $Next \triangleq$ 
178      $\vee \exists c \in Client : Do(c) \vee Rev(c)$ 
179      $\vee SRev$ 
180 | The  $Spec$ .
181 |-----|
182  $Spec \triangleq Init \wedge \Box[Next]_{vars} \wedge WF_{vars}(Next)$ 
183 |-----|
184 | The safety properties to check: Eventual Convergence ( $EC$ ), Quiescent Consistency ( $QC$ ), Strong
185 | Eventual Convergence ( $SEC$ ), Weak List Specification, ( $WLSpec$ ), and Strong List Specification,
186 | ( $SLSpec$ ).
187 |-----|
188 | Eventual Consistency ( $EC$ )
189 |-----|
190 | Quiescent Consistency ( $QC$ )
191 |-----|
192  $QConvergence \triangleq \forall c \in Client : cstate[c] = sstate$ 
193  $QC \triangleq comm!EmptyChannel \Rightarrow QConvergence$ 
194 |-----|
195 THEOREM  $Spec \Rightarrow \Box QC$ 
196 |-----|
197 | Strong Eventual Consistency ( $SEC$ )
198 |-----|
199 | Weak List Consistency ( $WLSpec$ )
200 |-----|
201 | Strong List Consistency ( $SLSpec$ )
202 |-----|
203 |
204 | \ * Modification History
205 | \ * Last modified Sun Jul 08 11:30:24 CST 2018 by hengxin
206 | \ * Created Sat Jun 23 17:14:18 CST 2018 by hengxin

```