

```

1  |----- MODULE AJupiter -----|
   | Specification of the Jupiter protocol presented by Attiya and others. |
6  | EXTENDS Integers, OT, TLC, AdditionalFunctionOperators |
7  |-----|
8  | CONSTANTS
9      Client,      the set of client replicas
10     Server,      the (unique) server replica
11     Char,        set of characters allowed
12     InitState    the initial state of each replica

14  Replica  $\triangleq$  Client  $\cup$  {Server}

16  List  $\triangleq$  Seq(Char  $\cup$  Range(InitState))    all possible lists/strings
17  MaxLen  $\triangleq$  Cardinality(Char) + Len(InitState)    the max length of lists in any states;
18  We assume that all inserted elements are unique.
19  ClientNum  $\triangleq$  Cardinality(Client)
20  Priority  $\triangleq$  CHOOSE  $f \in [Client \rightarrow 1 \dots ClientNum] : Injective(f)$ 
21  |-----|
22  | ASSUME
23       $\wedge Range(InitState) \cap Char = \{\}$ 
24       $\wedge Priority \in [Client \rightarrow 1 \dots ClientNum]$ 
25  |-----|
   | The set of all operations. Note: The positions are indexed from 1. |
30  Rd  $\triangleq$  [type : { "Rd" }]
31  Del  $\triangleq$  [type : { "Del" }, pos : 1 .. MaxLen]
32  Ins  $\triangleq$  [type : { "Ins" }, pos : 1 .. (MaxLen + 1), ch : Char, pr : 1 .. ClientNum]    pr: priority
34  Op  $\triangleq$  Ins  $\cup$  Del    Now we don't consider Rd operations.
35  |-----|
   | Messages between the Server and the Clients. There are two kinds of messages according to their |
   | destinations. |
40  Msg  $\triangleq$  [c : Client, ack : Int, op : Op  $\cup$  {Nop}]  $\cup$     messages sent to the Server from a client  $c \in Client$ 
41      [ack : Int, op : Op  $\cup$  {Nop}]    messages broadcast to Clients from the Server
42  |-----|
43  | VARIABLES
   | For the client replicas:
47      cbuf,      cbuf[c]: buffer (of operations) at the client  $c \in Client$ 
48      crec,      crec[c]: the number of new messages have been received by the client  $c \in Client$ 
49                  since the last time a message was sent
   | For the server replica:
53      sbuf,      sbuf[c]: buffer (of operations) at the Server, one per client  $c \in Client$ 
54      srec,      srec[c]: the number of new messages have been ... , one per client  $c \in Client$ 
   | For all replicas.
58      state,      state[r]: state (the list content) of replica  $r \in Replica$ 

```

```

For communication between the Server and the Clients:
62   cincoming,   cincoming[c]: incoming channel at the client  $c \in Client$ 
63   sincoming,   incoming channel at the Server
For model checking:
67   chins       a set of chars to insert
68 |-----|
69   comm  $\triangleq$  INSTANCE CSCComm WITH Msg  $\leftarrow$  Msg
70 |-----|
71   eVars  $\triangleq$   $\langle chins \rangle$            variables for the environment
72   cVars  $\triangleq$   $\langle cbuf, crec \rangle$        variables for the clients
73   ecVars  $\triangleq$   $\langle eVars, cVars \rangle$    variables for the clients and the environment
74   sVars  $\triangleq$   $\langle sbuf, srec \rangle$        variables for the server
75   commVars  $\triangleq$   $\langle cincoming, sincoming \rangle$  variables for communication
76   vars  $\triangleq$   $\langle eVars, cVars, sVars, commVars, state \rangle$  all variables
77 |-----|
78   TypeOK  $\triangleq$ 
For the client replicas:
82    $\wedge cbuf \in [Client \rightarrow Seq(Op \cup \{Nop\})]$ 
83    $\wedge crec \in [Client \rightarrow Int]$ 
For the server replica:
87    $\wedge sbuf \in [Client \rightarrow Seq(Op \cup \{Nop\})]$ 
88    $\wedge srec \in [Client \rightarrow Int]$ 
For all replicas.
92    $\wedge state \in [Replica \rightarrow List]$ 
For communication between the server and the clients:
96    $\wedge comm!TypeOK$ 
For model checking:
100   $\wedge chins \in SUBSET Char$ 
101 |-----|
The Init predicate.
105 Init  $\triangleq$ 
106    $\wedge chins = Char$ 
For the client replicas:
110    $\wedge cbuf = [c \in Client \mapsto \langle \rangle]$ 
111    $\wedge crec = [c \in Client \mapsto 0]$ 
For the server replica:
115    $\wedge sbuf = [c \in Client \mapsto \langle \rangle]$ 
116    $\wedge srec = [c \in Client \mapsto 0]$ 
For all replicas.
120    $\wedge state = [r \in Replica \mapsto InitState]$ 
For communication between the server and the clients:

```

124 $\wedge comm!Init$

125 \vdash

Client $c \in Client$ issues an operation op .

129 $DoOp(c, op) \triangleq$

130 $\wedge state' = [state \text{ EXCEPT } ![c] = Apply(op, @)]$

131 $\wedge cbuf' = [cbuf \text{ EXCEPT } ![c] = Append(@, op)]$

132 $\wedge crec' = [crec \text{ EXCEPT } ![c] = 0]$

133 $\wedge comm!CSend([c \mapsto c, ack \mapsto crec[c], op \mapsto op])$

135 $DoIns(c) \triangleq$

136 $\exists ins \in Ins :$

137 $\wedge ins.pos \in 1 \dots (Len(state[c]) + 1)$

138 $\wedge ins.ch \in chins$

139 $\wedge ins.pr = Priority[c]$

140 $\wedge chins' = chins \setminus \{ins.ch\}$ We assume that all inserted elements are unique.

141 $\wedge DoOp(c, ins)$

142 $\wedge UNCHANGED \ sVars$

144 $DoDel(c) \triangleq$

145 $\exists del \in Del :$

146 $\wedge del.pos \in 1 \dots Len(state[c])$

147 $\wedge DoOp(c, del)$

148 $\wedge UNCHANGED \langle sVars, eVars \rangle$

150 $Do(c) \triangleq$

151 $\vee DoIns(c)$

152 $\vee DoDel(c)$

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

157 $Rev(c) \triangleq$

158 $\wedge comm!CRev(c)$

159 $\wedge crec' = [crec \text{ EXCEPT } ![c] = @ + 1]$

160 $\wedge LET \ m \triangleq Head(cincoming[c])$

161 $cBuf \triangleq cbuf[c]$ the buffer at client $c \in Client$

162 $cShiftedBuf \triangleq SubSeq(cBuf, m.ack + 1, Len(cBuf))$ buffer shifted

163 $xop \triangleq XformOpOps(m.op, cShiftedBuf)$ transform op vs. shifted buffer

164 $xcBuf \triangleq XformOpsOp(cShiftedBuf, m.op)$ transform shifted buffer vs. op

165 IN $\wedge cbuf' = [cbuf \text{ EXCEPT } ![c] = xcBuf]$

166 $\wedge state' = [state \text{ EXCEPT } ![c] = Apply(xop, @)]$ apply the transformed operation xop

167 $\wedge UNCHANGED \langle sVars, eVars \rangle$

168 \vdash

The *Server* receives a message.

172 $SRev \triangleq$

173 $\wedge comm!SRev$

174 $\wedge LET \ m \triangleq Head(sincoming)$ the message to handle with

```

175       $c \triangleq m.c$                                 the client  $c \in Client$  that sends this message
176       $cBuf \triangleq sbuf[c]$                         the buffer at the Server for client  $c \in Client$ 
177       $cShiftedBuf \triangleq SubSeq(cBuf, m.ack + 1, Len(cBuf))$  buffer shifted
178       $xop \triangleq XformOpOps(m.op, cShiftedBuf)$  transform  $op$  vs. shifted buffer
179       $xcBuf \triangleq XformOpsOp(cShiftedBuf, m.op)$  transform shifted buffer vs.  $op$ 
180      IN  $\wedge srec' = [cl \in Client \mapsto$ 
181          IF  $cl = c$ 
182          THEN  $srec[cl] + 1$  receive one more operation from client  $c \in Client$ 
183          ELSE  $0$ ] reset  $srec$  for other clients than  $c \in Client$ 
184       $\wedge sbuf' = [cl \in Client \mapsto$ 
185          IF  $cl = c$ 
186          THEN  $xcBuf$  transformed buffer for client  $c \in Client$ 
187          ELSE  $Append(sbuf[cl], xop)$  store transformed  $xop$  into other clients' bufs
188       $\wedge state' = [state \text{ EXCEPT } ![Server] = Apply(xop, @)]$  apply the transformed operation
189       $\wedge comm!SSend(c, [cl \in Client \mapsto [ack \mapsto srec[cl], op \mapsto xop]])$ 
190       $\wedge \text{UNCHANGED } ecVars$ 
191  |-----|
192  The next-state relation.
193   $Next \triangleq$ 
194       $\vee \exists c \in Client : Do(c) \vee Rev(c)$ 
195       $\vee SRev$ 
196  The Spec. (TODO: Check the fairness condition.)
201   $Spec \triangleq Init \wedge \Box [Next]_{vars} \wedge WF_{vars}(Next)$ 
202  |-----|
203  The safety properties to check: Eventual Convergence (EC), Quiescent Consistency (QC), Strong
204  Eventual Convergence (SEC), Weak List Specification, (WLSpec), and Strong List Specification,
205  (SLSpec).
206
207  Eventual Consistency (EC)
208
209  Quiescent Consistency (QC)
217   $QC \triangleq comm!EmptyChannel \Rightarrow Cardinality(Range(state)) = 1$ 
219  THEOREM  $Spec \Rightarrow \Box QC$ 
220
221  Strong Eventual Consistency (SEC)
222  |-----|
223  \ * Modification History
224  \ * Last modified Sun Sep 09 10:05:29 CST 2018 by hengxin
225  \ * Created Sat Jun 23 17:14:18 CST 2018 by hengxin

```