MODULE *AJupiter*

Specification of the *Jupiter* protocol presented by *Hagit Attiya* and others.

EXTENDS *JupiterInterface*

Messages between the *Server* and the Clients.

$Msg \triangleq [c : Client,\ ack : Int,\ op : Op \cup \{Nop\}] \cup$ messages sent to the *Server* from a client $c \in Client$
$\quad [ack : Int,\ op : Op \cup \{Nop\}]$ messages broadcast to Clients from the *Server*

VARIABLES

- $cbuf$, $cbuf[c]$: 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
- $sbuf$, $sbuf[c]$: 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 $c \in Client$

$vars \triangleq \langle chins,\ cbuf,\ crec,\ sbuf,\ srec,\ cincoming,\ sincoming,\ state \rangle$

$TypeOK \triangleq$
- $\land\ TypeOKInt$
- $\land\ Comm(Msg)!TypeOK$
- $\land\ cbuf \in [Client \to Seq(Op \cup \{Nop\})]$
- $\land\ crec \in [Client \to Int]$
- $\land\ sbuf \in [Client \to Seq(Op \cup \{Nop\})]$
- $\land\ srec \in [Client \to Int]$

$Init \triangleq$
- $\land\ InitInt$
- $\land\ Comm(Msg)!Init$
- $\land\ cbuf = [c \in Client \mapsto \langle\rangle]$
- $\land\ crec = [c \in Client \mapsto 0]$
- $\land\ sbuf = [c \in Client \mapsto \langle\rangle]$
- $\land\ srec = [c \in Client \mapsto 0]$

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

$DoOp(c,\ op) \triangleq$
- $\land\ state' = [state \text{ EXCEPT } ![c] = Apply(op,\ @)]$
- $\land\ cbuf' = [cbuf \text{ EXCEPT } ![c] = Append(@,\ op)]$
- $\land\ crec' = [crec \text{ EXCEPT } ![c] = 0]$
- $\land\ Comm(Msg)!CSend([c \mapsto c,\ ack \mapsto crec[c],\ op \mapsto op])$

$DoIns(c) \triangleq$
- $\exists\, ins \in \{op \in Ins : op.pos \in 1\,..\,(Len(state[c]) + 1) \land op.ch \in chins \land op.pr = Priority[c]\} :$
  - $\land\ DoOp(c,\ ins)$
  - $\land\ chins' = chins \setminus \{ins.ch\}$ We assume that all inserted elements are unique.
  - $\land$ UNCHANGED $\langle sbuf,\ srec \rangle$

```
53  DoDel(c) ≜
54      ∃ del ∈ {op ∈ Del : op.pos ∈ 1 .. Len(state[c])} :
55          ∧ DoOp(c, del)
56          ∧ UNCHANGED ⟨chins, sbuf, srec⟩

58  Do(c) ≜
59          ∨ DoIns(c)
60          ∨ DoDel(c)
```

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

```
64  Rev(c) ≜
65          ∧ Comm(Msg)!CRev(c)
66          ∧ crec' = [crec EXCEPT ![c] = @ + 1]
67          ∧ LET m ≜ Head(cincoming[c])
68                cBuf ≜ cbuf[c]   the buffer at client c ∈ Client
69                cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))   buffer shifted
70                xop ≜ XformOpOps(m.op, cShiftedBuf)   transform op vs. shifted buffer
71                xcBuf ≜ XformOpsOp(cShiftedBuf, m.op)   transform shifted buffer vs. op
72            IN  ∧ cbuf' = [cbuf EXCEPT ![c] = xcBuf]
73                ∧ state' = [state EXCEPT ![c] = Apply(xop, @)]   apply the transformed operation xop
74          ∧ UNCHANGED ⟨chins, sbuf, srec⟩
```

The *Server* receives a message.

```
78  SRev ≜
79      ∧ Comm(Msg)!SRev
80      ∧ LET m ≜ Head(sincoming)   the message to handle with
81            c ≜ m.c   the client c ∈ Client that sends this message
82            cBuf ≜ sbuf[c]   the buffer at the Server for client c ∈ Client
83            cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))   buffer shifted
84            xop ≜ XformOpOps(m.op, cShiftedBuf)   transform op vs. shifted buffer
85            xcBuf ≜ XformOpsOp(cShiftedBuf, m.op)   transform shifted buffer vs. op
86        IN  ∧ srec' = [cl ∈ Client ↦
87                       IF cl = c
88                       THEN srec[cl] + 1   receive one more operation from client c ∈ Client
89                       ELSE 0]   reset srec for other clients than c ∈ Client
90            ∧ sbuf' = [cl ∈ Client ↦
91                       IF cl = c
92                       THEN xcBuf   transformed buffer for client c ∈ Client
93                       ELSE Append(sbuf[cl], xop)]   store transformed xop into other clients' bufs
94            ∧ state' = [state EXCEPT ![Server] = Apply(xop, @)]   apply the transformed operation
95            ∧ Comm(Msg)!SSend(c, [cl ∈ Client ↦ [ack ↦ srec[cl], op ↦ xop]])
96      ∧ UNCHANGED ⟨chins, cbuf, crec⟩
97  ⊢──────────────────────────────────────────────────────────────────────────
98  Next ≜
99          ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
100         ∨ SRev
```

Fairness: There is no requirement that the clients ever generate operations.

$$Fairness \triangleq$$
$$\quad \mathrm{WF}_{vars}(SRev \lor \exists\, c \in Client : Rev(c))$$

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

Quiescent Consistency (QC)

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

THEOREM $Spec \Rightarrow \Box QC$

---

* Modification History
* *Last* modified *Tue Dec* 04 21:10:51 *CST* 2018 by *hengxin*
* Created Sat *Jun* 23 17:14:18 *CST* 2018 by *hengxin*