MODULE *NJupiter*

```
********** Original Jupiter algorithm **********
class Msg
// type: [op, seq]

class Client
  var lseq    // local generated msg number
  var gseq    // global received msg number
  var outgoing//acked ops can be removed, for simplicity we keep it
              //acked ops: outgoing[1 : msg.seq − 1]

  synchronized procedure Do(op):
    Apply(op)
    SendServer([op, gseq])
    Append(outgoing, [op, lseq])
    lseq := lseq + 1

  synchronized procedure Recv(msg):
    xop, xops := Xform(msg.op, outgoing[msg.seq : Len(outgoing)])
    Apply(xop)
    outgoing := outgoing[1 : msg.seq − 1] + xops
    gseq := gseq + 1

class ServerThread// every client has a corresponding server thread
  var lseq// symmetrical Client.gseq for simplicity we keep its name
  var gseq
  var outgoing

  synchronized procedure SRecv(msg):
    same as Client.Recv(msg)
    SignalOtherServerThreads(xop)

  synchronized procedure Signaled(op):
    same as Client.Do(op) except replacing SendServer to SendClient
```

EXTENDS *JupiterInterface*, *OT*, *BufferStateSpace*

VARIABLES

cbuf,	*cbuf*[*c*]: locally generated operations at client *c*
cseq,	*cseq*[*c*]: number of remote operations received by client *c*
sbuf,	*sbuf*[*c*]: transformed remote operations *w.r.t* client *c*
sseq	*sseq*[*c*]: number of locally generated operations by client *c*

$vars \triangleq \langle intVars, cbuf, cseq, sbuf, sseq \rangle$

$NMsg \triangleq$

$[c : Client,\ seq : Nat,\ op : Op \cup \{Nop\}] \cup$ client $\rightarrow$ server
$[seq : Nat,\ op : Op \cup \{Nop\}]$ server $\rightarrow$ client

---

$TypeOK \triangleq$
$\land \quad TypeOKInt$
$\land \quad cbuf \in [Client \rightarrow Seq(Op \cup \{Nop\})]$
$\land \quad cseq \in [Client \rightarrow Nat]$
$\land \quad sbuf \in [Client \rightarrow Seq(Op \cup \{Nop\})]$
$\land \quad sseq \in [Client \rightarrow Nat]$

$Init \triangleq$
$\land\ InitInt$
$\land\ cbuf = [c \in Client \mapsto \langle\rangle]$
$\land\ cseq = [c \in Client \mapsto 0]$
$\land\ sbuf = [c \in Client \mapsto \langle\rangle]$
$\land\ sseq = [c \in Client \mapsto 0]$

$ClientPerform(c,\ m) \triangleq$
LET $xform \triangleq xFormLocate(OT,\ m.op,\ cbuf[c],\ m.seq)$ $[xop,\ xops]$
IN $\land\ cbuf' = [cbuf$ EXCEPT $![c] = xform.xops]$
$\land\ cseq' = [cseq$ EXCEPT $![c] = @ + 1]$
$\land\ SetNewAop(c,\ xform.xop)$

$ServerPerform(m) \triangleq$
LET $c \triangleq m.c$
$xform \triangleq xFormLocate(OT,\ m.op,\ sbuf[c],\ m.seq)$ $[xop,\ xops]$
$xop \triangleq xform.xop$
IN $\land\ sseq' = [sseq$ EXCEPT $![c] = @ + 1]$
$\land\ sbuf' = [cl \in Client \mapsto$ IF $cl = c$ THEN $xform.xops$
ELSE $Append(sbuf[cl],\ xop)]$
$\land\ SetNewAop(Server,\ xop)$
$\land\ Comm!SSend(c,\ [cl \in Client \mapsto [seq \mapsto sseq[cl],\ op \mapsto xop]])$

$DoOp(c,\ op) \triangleq$
$\land\ SetNewAop(c,\ op)$
$\land\ cbuf' = [cbuf$ EXCEPT $![c] = Append(@,\ op)]$
$\land\ Comm!CSend([c \mapsto c,\ seq \mapsto cseq[c],\ op \mapsto op])$

$Do(c) \triangleq$
$\land\ DoInt(DoOp,\ c)$
$\land$ UNCHANGED $\langle sbuf,\ sseq,\ cseq\rangle$

$Rev(c) \triangleq$
$\land\ RevInt(ClientPerform,\ c)$
$\land$ UNCHANGED $\langle sbuf,\ sseq\rangle$

$SRev \triangleq$
$\quad \land\ SRevInt(ServerPerform)$
$\quad \land$ UNCHANGED $\langle cbuf,\ cseq \rangle$

$Next \triangleq$
$\quad \lor \exists\, c \in Client : Do(c) \lor Rev(c)$
$\quad \lor\ SRev$

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

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

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

THEOREM $Spec \Rightarrow \Box QC$

---

* Modification History
* Last modified *Thu Apr* 18 16:08:04 *CST* 2019 by *tangruize*
* Last modified *Thu Jan* 17 10:30:39 *CST* 2019 by *hengxin*
* Created *Satchins*, Jun 23 17:14:18 *CST* 2018 by *hengxin*