MODULE *GJupiter*

```
************************** Google OT algorithm **************************
class Client
    var id
    var outgoing // acked ops can be removed, for simplicity we keep it
    var stage
    var ack

    synchronized procedure Do(op):
        Apply(op)
        Append(outgoing, op)
        Append(stage, "READY")
        Deliver()

    synchronized procedure Recv(msg):
        ack := ack + 1
        if msg.id = id
            stage[msg.seq] := "ACKED"
            Deliver()
        else
            xop, xops := Xform(msg.op, outgoing[msg.seq : Len(outgoing)])
            Apply(xop)
            outgoing := outgoing[1 : msg.seq − 1] + [xop] + xops
            Insert(stage, msg.seq, "ACKED")

    procedure Deliver():
        if ack < Len(stage) and stage[ack + 1] = "READY"
            SendServer([id, outgoing[ack + 1], ack + 1])
            stage[ack + 1] := "SENT"

class Server // single server
    var outgoing

    procedure SRecv(msg):
        xop, xops := Xform(msg.op, outgoing[msg.seq : Len(outgoing)])
        outgoing := outgoing[1 : msg.seq − 1] + xops + [xop]
        Broadcast([msg.id, xop, Len(outgoing)])
```

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

VARIABLES

outgoing,	*outgoing*[*r*]: local and global ops at replica *r*.
stage	*stage*[*c*]: *ops′* sending or receiving stages at client *c*.

CONSTANTS *READY*, *SENT*, *ACKED* for *stage*[*c*]

$Stages \triangleq \{READY, SENT, ACKED\}$

$vars \triangleq \langle intVars, outgoing, stage \rangle$

$GMsg \triangleq$ messages exchanged by server and clients.
$[c : Client, seq : Nat, op : Op \cup \{Nop\}]$

$TypeOK \triangleq$
$\land \quad TypeOKInt$
$\land \quad outgoing \in [Replica \to Seq(Op \cup \{Nop\})]$
$\land \quad stage \in [Client \to Seq(Stages)]$

$Init \triangleq$
$\land\ InitInt$
$\land\ outgoing = [r \in Replica \mapsto \langle\rangle]$
$\land\ stage = [c \in Client \mapsto \langle\rangle]$

$AdvanceStage(s, p) \triangleq$ Advance the stage s at position p
$[i \in 1 \,..\, Len(s) \mapsto$ IF $i = p$
THEN IF $s[i] = READY$
THEN $SENT$
ELSE $ACKED$
ELSE $s[i]]$

$ClientPerform(r, m) \triangleq$
IF $r = m.c$ msg generated by itself and acked by server.
THEN $\land\ stage' = [stage$ EXCEPT $![r] = AdvanceStage(@, m.seq)]$
$\land\ SetNewAop(r, Nop)$ a dummy operation.
$\land$ UNCHANGED $outgoing$
ELSE msg generated by other clients.
LET $xform \triangleq xFormJoin(OT, m.op, outgoing[r], m.seq - 1)$
IN $\land\ outgoing' = [outgoing$ EXCEPT $![r] = xform.xops]$
$\land\ stage' = [stage$ EXCEPT $![r] = InsertElement(@, ACKED, m.seq)]$
$\land\ SetNewAop(r, xform.xop)$

$ServerPerform(m) \triangleq$
LET $xform \triangleq xFormAppend(OT, m.op, outgoing[Server], m.seq - 1)$
IN $\land\ outgoing' = [outgoing$ EXCEPT $![Server] = xform.xops]$
$\land\ SetNewAop(Server, xform.xop)$
$\land\ Comm!SSendAll([c \mapsto m.c, seq \mapsto Len(outgoing'[Server]),$
$op \mapsto xform.xop])$
$\land$ UNCHANGED $stage$

$KickOut(c) \triangleq$ deliver ready message in $outgoing[c]$
$\land \quad \exists\, i \in 1 \,..\, Len(outgoing[c]) :$
$\land$ IF $i = 1$ THEN TRUE ELSE $stage[c][i-1] = ACKED$ stop and wait
$\land\ stage[c][i] = READY$ current msg is ready to send
$\land\ stage' = [stage$ EXCEPT $![c] = AdvanceStage(@,\ i)]$
$\land\ Comm!CSend([c \mapsto c,\ seq \mapsto i,\ op \mapsto outgoing[c][i]])$
$\land \quad$ UNCHANGED $\langle aop,\ chins,\ outgoing,\ state\rangle$ too detailed, improve?

$DoOp(c,\ op) \triangleq$
$\land\ SetNewAop(c,\ op)$
$\land\ outgoing' = [outgoing$ EXCEPT $![c] = Append(@,\ op)]$
$\land\ stage' = [stage$ EXCEPT $![c] = Append(@,\ READY)]$
$\land$ UNCHANGED $\langle cincoming,\ sincoming\rangle$ *DoOp* will not send a msg.

$Do(c) \triangleq DoInt(DoOp,\ c)$

$Rev(c) \triangleq RevInt(ClientPerform,\ c)$

$SRev \triangleq SRevInt(ServerPerform)$

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

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

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

$AllAcked \triangleq$ all clients' outgoing operations are acked.
$\forall\, c \in Client : Range(stage[c]) \in$ SUBSET $\{ACKED\}$

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

THEOREM $Spec \Rightarrow \Box QC$

---

* Modification History
* Last modified *Thu Apr* 18 11:07:27 *CST* 2019 by *tangruize*
* Created *Fri Mar* 15 08:15:22 *CST* 2019 by *tangruize*