MODULE *GJupiter*

```
************************** Google OT algorithm **************************
class Msg
// type: Client → Server: [op, ack] client generated op and acked num
//       Server → Client: "ACKED" last msg is acked
//                        op      generated by other clients

class Client
  var outgoing// local generated operation buf: [op, stage]
  var ack     //init to 0

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

  synchronized procedure Recv(msg):
    ack := ack + 1
    if msg = "ACKED"
      Remove(outgoing, 1)
      Deliver()
    else
      xop, outgoing := Xform(msg, outgoing)
      Apply(xop)

  procedure Deliver():
    if not Empty(outgoing) and outgoing[1].stage = "READY"
      Send(Server, [outgoing[1], ack])
      outgoing[1].stage := "SENT"

class Server    // single server
  var outgoing//ops cannot be removed

  procedure SRecv(msg):
    xop, xops := Xform(msg.op, outgoing[msg.ack + 1 : Len(outgoing)])
    outgoing := outgoing[1 : msg.ack] + xops + [xop]
    Send(msg's sender, "ACKED")
    Send(other clients, xop)
```

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

VARIABLES

outgoing, *outgoing*[*r*]: ops generated client or received by server.

stage, *stage*[*c*]: client msg sending stage.

ack $ack[c]$: client acked msg number.

CONSTANTS $READY$, $SENT$, $ACKED$

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

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

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

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

$Init \triangleq$
$\land\ InitInt$
$\land\ outgoing = [r \in Replica \mapsto \langle\rangle]$
$\land\ stage = [c \in Client \mapsto \langle\rangle]$
$\land\ ack = [c \in Client \mapsto 0]$

$Send(c) \triangleq$
IF $Len(stage[c]) \neq 0$
THEN $\land\ stage[c][1] = READY$
$\land\ stage' = [stage$ EXCEPT $![c] = \langle SENT\rangle \circ Tail(@)]$
$\land\ Comm!CSend([c \mapsto c, op \mapsto outgoing[c][1], ack \mapsto ack[c]])$
$\land$ UNCHANGED $\langle ack, aop, chins, outgoing, state\rangle$
ELSE FALSE

$ClientPerform(c, m) \triangleq$
IF $m = ACKED$ last msg acked by server.
THEN $\land\ outgoing' = [outgoing$ EXCEPT $![c] = Tail(@)]$
$\land\ stage' = [stage$ EXCEPT $![c] = Tail(@)]$
$\land\ SetNewAop(c, Nop)$ a dummy operation.
ELSE op generated by other clients.
LET $xform \triangleq xFormFull(OT, m, outgoing[c])$
IN $\land\ outgoing' = [outgoing$ EXCEPT $![c] = xform.xops]$
$\land$ UNCHANGED $stage$
$\land\ SetNewAop(c, xform.xop)$

$ServerPerform(m) \triangleq$
LET $xform \triangleq xFormAppend(OT, m.op, outgoing[Server], m.ack)$
IN $\land\ outgoing' = [outgoing$ EXCEPT $![Server] = xform.xops]$

$\land$ $SetNewAop(Server,\ xform.xop)$
$\land$ $Comm!SSendSameAck(m.c,\ ACKED,\ xform.xop)$
$\land$ UNCHANGED $\langle ack,\ stage \rangle$

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

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

$Rev(c) \triangleq$
$\land$ $ack' = [ack \text{ EXCEPT } ![c] = @ + 1]$
$\land$ $RevInt(ClientPerform,\ c)$

$SRev \triangleq SRevInt(ServerPerform)$

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

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

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

$EmptyOutgoing \triangleq$ all clients' outgoing is empty.
$\forall\, c \in Client : Len(stage[c]) = 0$

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

THEOREM $Spec \Rightarrow \Box QC$

---

* Modification History
* Last modified Sat *Apr* 20 22:49:46 *CST* 2019 by *tangruize*
* Created *Fri Mar* 15 08:15:22 *CST* 2019 by *tangruize*