```
1  ---------- MODULE AJupiterExtended ----------
   AJupiter extended with JupiterCtx. This is used to show that AJupiter implements XJupiter.
5  EXTENDS JupiterCtx
6  ----------
7  VARIABLES cbuf, crec, sbuf, srec, cincomingXJ, sincomingXJ

9  commXJVars ≜ ⟨cincomingXJ, sincomingXJ⟩
10 commXJ ≜ INSTANCE CSComm WITH Msg ← Seq(Cop),
11                               cincoming ← cincomingXJ, sincoming ← sincomingXJ

13 varsEx ≜ ⟨intVars, ctxVars, cbuf, crec, sbuf, srec, commXJVars⟩

15 Msg ≜ [ack : Int, cop : Cop, oid : Oid]
16 ----------
17 TypeOKEx ≜
18     ∧ TypeOKInt
19     ∧ TypeOKCtx
20     ∧ Comm(Msg)!TypeOK
21     ∧ commXJ!TypeOK
22     ∧ crec ∈ [Client → Int]
23     ∧ srec ∈ [Client → Int]
24     ∧ cbuf ∈ [Client → Seq(Cop)]
25     ∧ sbuf ∈ [Client → Seq(Cop)]
26 ----------
27 InitEx ≜
28     ∧ InitInt
29     ∧ InitCtx
30     ∧ commXJ!Init
31     ∧ Comm(Msg)!Init
32     ∧ crec = [c ∈ Client ↦ 0]
33     ∧ srec = [c ∈ Client ↦ 0]
34     ∧ cbuf = [c ∈ Client ↦ ⟨⟩]
35     ∧ sbuf = [c ∈ Client ↦ ⟨⟩]
36 ----------
37 DoOpEx(c, op) ≜
38     LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq′[c]], ctx ↦ ds[c]]
39     IN  ∧ crec′ = [crec EXCEPT ![c] = 0]
40         ∧ cbuf′ = [cbuf EXCEPT ![c] = Append(@, cop)]
41         ∧ SetNewAop(c, op)
42         ∧ Comm(Msg)!CSend([ack ↦ crec[c], cop ↦ cop, oid ↦ cop.oid])
43         ∧ commXJ!CSend(cop)

45 DoEx(c) ≜
46     ∧ DoCtx(c)
47     ∧ DoInt(DoOpEx, c)
48     ∧ UNCHANGED ⟨sbuf, srec⟩
```

```
RevEx(c) ≜
    ∧ Comm(Msg)!CRev(c)
    ∧ commXJ!CRev(c)
    ∧ crec' = [crec EXCEPT ![c] = @ + 1]
    ∧ LET m ≜ Head(cincoming[c])
          cBuf ≜ cbuf[c]
          cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))
          xcop ≜ XformOpOps(COT, m.cop, cShiftedBuf)
          xcBuf ≜ XformOpsOp(COT, cShiftedBuf, m.cop)
      IN  ∧ cbuf' = [cbuf EXCEPT ![c] = xcBuf]
          ∧ SetNewAop(c, xcop.op)
    ∧ RevCtx(c)
    ∧ RevInt(c)
    ∧ UNCHANGED ⟨sbuf, srec⟩

SRevEx ≜
    ∧  Comm(Msg)!SRev
    ∧  commXJ!SRev
    ∧  LET m ≜ Head(sincoming)
           c ≜ ClientOf(m.cop)
           cBuf ≜ sbuf[c]
           cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))
           xcop ≜ XformOpOps(COT, m.cop, cShiftedBuf)
           xcBuf ≜ XformOpsOp(COT, cShiftedBuf, m.cop)
       IN  ∧ srec' = [cl ∈ Client ↦
                        IF cl = c THEN srec[cl] + 1 ELSE 0]
           ∧ sbuf' = [cl ∈ Client ↦
                        IF cl = c THEN xcBuf ELSE Append(sbuf[cl], xcop)]
           ∧ SetNewAop(Server, xcop.op)
           ∧ Comm(Msg)!SSend(c, [cl ∈ Client ↦ [ack ↦ srec[cl], cop ↦ xcop, oid ↦ xcop.oid]])
           ∧ commXJ!SSendSame(c, xcop)
    ∧  SRevCtx
    ∧  SRevInt
    ∧  UNCHANGED ⟨cbuf, crec⟩
```

```
NextEx ≜
    ∨  ∃ c ∈ Client : DoEx(c) ∨ RevEx(c)
    ∨  SRevEx

FairnessEx ≜  There is no requirement that the clients ever generate operations.
    WF_varsEx(SRevEx ∨ ∃ c ∈ Client : RevEx(c))

SpecEx ≜ InitEx ∧ □[NextEx]_varsEx  ∧ FairnessEx
```

```
QC ≜  Quiescent Consistency
    Comm(Msg)!EmptyChannel ⇒ Cardinality(Range(state)) = 1
```

97 THEOREM $SpecEx \Rightarrow \Box QC$

98 ⌊_____⌋

\ * Modification History
\ * Last modified *Tue Jan* 01 11:48:24 *CST* 2019 by *hengxin*
\ * Created *Thu Dec* 27 21:15:09 *CST* 2018 by *hengxin*