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

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

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

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

49 DoIns(c) ≜
50   ∃ ins ∈ {op ∈ Ins : op.pos ∈ 1 .. (Len(state[c]) + 1) ∧ op.ch ∈ chins ∧ op.pr = Priority[c]} :
51     ∧ DoOp(c, ins)
```

```
52          ∧ chins' = chins \ {ins.ch}

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

59  DoEx(c) ≜
60          ∧ DoCtx(c)
61          ∧ ∨ DoIns(c)
62            ∨ DoDel(c)
63          ∧ UNCHANGED ⟨sbuf, srec⟩
64  ──────────────────────────────────────────────────────────────
    Client c ∈ Client receives a message from the Server.
68  RevEx(c) ≜
69      ∧ Comm(Msg)!CRev(c)
70      ∧ commXJ!CRev(c)
71      ∧ crec' = [crec EXCEPT ![c] = @ + 1]
72      ∧ LET m ≜ Head(cincoming[c])
73            cBuf ≜ cbuf[c]
74            cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))
75            xcop ≜ XformOpOps(COT, m.cop, cShiftedBuf)
76            xcBuf ≜ XformOpsOp(COT, cShiftedBuf, m.cop)
77        IN  ∧ cbuf' = [cbuf EXCEPT ![c] = xcBuf]
78            ∧ state' = [state EXCEPT ![c] = Apply(xcop.op, @)]
79      ∧ RevCtx(c)
80      ∧ UNCHANGED ⟨chins, sbuf, srec⟩
81  ──────────────────────────────────────────────────────────────
    The Server receives a message.
85  SRevEx ≜
86      ∧  Comm(Msg)!SRev
87      ∧  commXJ!SRev
88      ∧  LET m ≜ Head(sincoming)
89             c ≜ ClientOf(m.cop)
90             cBuf ≜ sbuf[c]
91             cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))
92             xcop ≜ XformOpOps(COT, m.cop, cShiftedBuf)
93             xcBuf ≜ XformOpsOp(COT, cShiftedBuf, m.cop)
94         IN  ∧ srec' = [cl ∈ Client ↦
95                          IF cl = c THEN srec[cl] + 1 ELSE 0]
96             ∧ sbuf' = [cl ∈ Client ↦
97                          IF cl = c THEN xcBuf ELSE Append(sbuf[cl], xcop)]
98             ∧ state' = [state EXCEPT ![Server] = Apply(xcop.op, @)]
99             ∧ Comm(Msg)!SSend(c, [cl ∈ Client ↦ [ack ↦ srec[cl], cop ↦ xcop, oid ↦ xcop.oid]])
100            ∧ commXJ!SSendSame(c, xcop)
```

```
101       ∧  SRevCtx
102       ∧  UNCHANGED ⟨chins, cbuf, crec⟩
103  ├──────────────────────────────────────────────────────────────────────┤
104  NextEx ≜
105       ∨  ∃ c ∈ Client : DoEx(c) ∨ RevEx(c)
106       ∨  SRevEx

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

111  SpecEx ≜  InitEx ∧ □[NextEx]_varsEx  ∧ FairnessEx
112  ├──────────────────────────────────────────────────────────────────────┤
113  QC ≜  Quiescent Consistency
114       Comm(Msg)!EmptyChannel ⇒ Cardinality(Range(state)) = 1

116  THEOREM SpecEx ⇒ □QC
117  └──────────────────────────────────────────────────────────────────────┘
     \* Modification History
     \* Last modified Sun Dec 30 16:43:20 CST 2018 by hengxin
     \* Created Thu Dec 27 21:15:09 CST 2018 by hengxin
```