```
1  ---------- MODULE XJupiter ----------
   Specification of the Jupiter protocol described in CSCW'2014 by Xu, Sun, and Li. We call it
   XJupiter, with 'X' for "Xu".
6  EXTENDS StateSpace
7  ----------
8  VARIABLES
9      c2ss,     c2ss[c]: the 2D state space (2ss, for short) at client c ∈ Client
10     s2ss      s2ss[c]: the 2D state space maintained by the Server for client c ∈ Client

12 vars ≜ ⟨intVars, ctxVars, c2ss, s2ss⟩
13 ----------
14 TypeOK ≜
15     ∧  TypeOKInt
16     ∧  TypeOKCtx
17     ∧  ∀ c ∈ Client : IsSS(c2ss[c]) ∧ IsSS(s2ss[c])
18 ----------
19 Init ≜
20     ∧ InitInt
21     ∧ InitCtx
22     ∧ c2ss = [c ∈ Client ↦ EmptySS]
23     ∧ s2ss = [c ∈ Client ↦ EmptySS]
24 ----------
25 xForm(cop, ss, cur) ≜  Transform cop with an operation sequence in 2D state space ss.
26     LET u ≜ Locate(cop, ss)
27         v ≜ u ∪ {cop.oid}
28         RECURSIVE xFormHelper(_, _, _, _)
29          xFormHelper(uh, vh, coph, xss) ≜   xss: eXtra ss created during transformation
30              IF uh = cur THEN [xss ↦ xss, xcop ↦ coph]
31               ELSE LET e ≜ CHOOSE e ∈ ss.edge : e.from = uh ∧ ClientOf(e.cop) ≠ ClientOf(cop)
32                        copprime ≜ e.cop
33                        uprime ≜ e.to
34                        vprime ≜ vh ∪ {copprime.oid}
35                        coph2copprime ≜ COT(coph, copprime)
36                        copprime2coph ≜ COT(copprime, coph)
37                    IN  xFormHelper(uprime, vprime, coph2copprime,
38                           xss ⊕ [node ↦ {vprime},
39                                  edge ↦ {[from ↦ vh, to ↦ vprime, cop ↦ copprime2coph],
40                                          [from ↦ uprime, to ↦ vprime, cop ↦ coph2copprime]}])
41      IN  xFormHelper(u, v, cop, [node ↦ {v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop]}])

43 ClientPerform(c, cop) ≜
44     LET xform ≜ xForm(cop, c2ss[c], ds[c])  xform: [xss, xcop]
45      IN  ∧ c2ss′ = [c2ss EXCEPT ![c] = @ ⊕ xform.xss]
46          ∧ SetNewAop(c, xform.xcop.op)

48 ServerPerform(cop) ≜
```

```
49      LET c ≜ ClientOf(cop)
50       scur ≜ ds[Server]
51      xform ≜ xForm(cop, s2ss[c], scur)  xform: [xss, xcop]
52       xcop ≜ xform.xcop
53       xcur ≜ scur ∪ {cop.oid}
54       IN   ∧ s2ss' = [cl ∈ Client ↦
55                        IF cl = c
56                         THEN s2ss[cl] ⊕ xform.xss
57                         ELSE s2ss[cl] ⊕ [node ↦ {xcur},
58                              edge ↦ {[from ↦ scur, to ↦ xcur, cop ↦ xcop]}]]
59            ∧ SetNewAop(Server, xcop.op)
60            ∧ Comm!SSendSame(c, xcop)
61 ----------------------------------------------------------------------
62 DoOp(c, op) ≜
63       LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq[c]], ctx ↦ ds[c]]
64        IN   ∧ ClientPerform(c, cop)
65             ∧ Comm!CSend(cop)

67 Do(c) ≜
68       ∧ DoInt(DoOp, c)
69       ∧ DoCtx(c)
70       ∧ UNCHANGED s2ss

72 Rev(c) ≜
73       ∧ RevInt(ClientPerform, c)
74       ∧ RevCtx(c)
75       ∧ UNCHANGED s2ss

77 SRev ≜
78      ∧ SRevInt(ServerPerform)
79      ∧ SRevCtx
80      ∧ UNCHANGED c2ss
81 ----------------------------------------------------------------------
82 Next ≜
83      ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
84      ∨ SRev

86 Fairness ≜
87      WF_vars(SRev ∨ ∃ c ∈ Client : Rev(c))

89 Spec ≜ Init ∧ □[Next]_vars  ∧ Fairness
90 ----------------------------------------------------------------------
91 CSSync ≜  Each client c ∈ Client is synchonized with the Server.
92      ∀ c ∈ Client : (ds[c] = ds[Server]) ⇒ c2ss[c] = s2ss[c]

94 THEOREM Spec ⇒ □CSSync
95 ======================================================================
```

\ * Modification History
\ * Last modified *Wed Jan* 02 22:00:47 *CST* 2019 by *hengxin*
\ * Created *Tue Oct* 09 16:33:18 *CST* 2018 by *hengxin*