```
1  ---------- MODULE XJupiter ----------
   Specification of the Jupiter protocol described in CSCW'2014 by Yi Xu, Chengzheng Sun, and
   Mo Li. We call it XJupiter, with 'X' for "Xu".
7  EXTENDS StateSpace
8  ----------
9  VARIABLES
       The 2D state spaces (2ss, for short). Each client maintains one 2D state space. The server
       maintains n 2D state spaces, one for each client.
15     c2ss,     c2ss[c]: the 2D state space at client c ∈ Client
16     s2ss      s2ss[c]: the 2D state space maintained by the Server for client c ∈ Client

18 vars ≜ ⟨intVars, ctxVars, c2ss, s2ss⟩
19 ----------
20 TypeOK ≜
21     ∧  TypeOKInt
22     ∧  TypeOKCtx
23     ∧  Comm(Cop)!TypeOK
24     ∧  ∀ c ∈ Client : IsSS(c2ss[c]) ∧ IsSS(s2ss[c])
25 ----------
26 Init ≜
27     ∧ InitInt
28     ∧ InitCtx
29     ∧ Comm(Cop)!Init
30     ∧ c2ss = [c ∈ Client ↦ EmptySS]
31     ∧ s2ss = [c ∈ Client ↦ EmptySS]
32 ----------
   xForm: iteratively transform cop with a path through the 2D state space ss at some client.
37 xForm(cop, ss, cur) ≜
38     LET u ≜ Locate(cop, ss)
39         v ≜ u ∪ {cop.oid}
40         RECURSIVE xFormHelper(_, _, _, _)
41         xFormHelper(uh, vh, coph, xss) ≜   xss: eXtra ss created during transformation
42             IF uh = cur THEN [xss ↦ xss, xcop ↦ coph]
43              ELSE LET e ≜ CHOOSE e ∈ ss.edge : e.from = uh ∧ ClientOf(e.cop) ≠ ClientOf(cop)
44                       copprime ≜ e.cop
45                       uprime ≜ e.to
46                       vprime ≜ vh ∪ {copprime.oid}
47                       coph2copprime ≜ COT(coph, copprime)
48                       copprime2coph ≜ COT(copprime, coph)
49                   IN  xFormHelper(uprime, vprime, coph2copprime,
50                           xss ⊕ [node ↦ {vprime},
51                                  edge ↦ {[from ↦ vh, to ↦ vprime, cop ↦ copprime2coph],
52                                          [from ↦ uprime, to ↦ vprime, cop ↦ coph2copprime]}])
53     IN  xFormHelper(u, v, cop, [node ↦ {v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop]}])
54 ----------
```

Client $c \in Client$ perform operation cop.

$ClientPerform(cop,\ c) \triangleq$
  LET $xform \triangleq xForm(cop,\ c2ss[c],\ ds[c])$ *xform*: [*xss*, *xcop*]
  IN $\land\ c2ss' = [c2ss$ EXCEPT $![c] = @ \oplus xform.xss]$
     $\land\ state' = [state$ EXCEPT $![c] = Apply(xform.xcop.op,\ @)]$

Client $c \in Client$ generates an operation op.

$DoOp(c,\ op) \triangleq$
  LET $cop \triangleq [op \mapsto op,\ oid \mapsto [c \mapsto c,\ seq \mapsto cseq'[c]],\ ctx \mapsto ds[c]]$
  IN $\land\ ClientPerform(cop,\ c)$
     $\land\ UpdateDS(c,\ cop)$
     $\land\ Comm(Cop)!CSend(cop)$

$DoIns(c) \triangleq$
  $\exists\, ins \in \{op \in Ins : op.pos \in 1\,..\,(Len(state[c]) + 1) \land op.ch \in chins \land op.pr = Priority[c]\} :$
    $\land\ DoOp(c,\ ins)$
    $\land\ chins' = chins \setminus \{ins.ch\}$ We assume that all inserted elements are unique.

$DoDel(c) \triangleq$
  $\exists\, del \in \{op \in Del : op.pos \in 1\,..\,Len(state[c])\} :$
    $\land\ DoOp(c,\ del)$
    $\land$ UNCHANGED $chins$

$Do(c) \triangleq$
  $\land\ DoCtx(c)$
  $\land\ \lor\ DoIns(c)$
     $\lor\ DoDel(c)$
  $\land$ UNCHANGED $s2ss$

Client $c \in Client$ receives a message from the *Server*.

$Rev(c) \triangleq$
  $\land\ Comm(Cop)!CRev(c)$
  $\land$ LET $cop \triangleq Head(cincoming[c])$ the received (transformed) operation
    IN $ClientPerform(cop,\ c)$
  $\land\ RevCtx(c)$
  $\land$ UNCHANGED $\langle chins,\ s2ss\rangle$

The *Server* performs operation cop.

$ServerPerform(cop) \triangleq$
  LET $c \triangleq ClientOf(cop)$
   $scur \triangleq ds[Server]$
  $xform \triangleq xForm(cop,\ s2ss[c],\ scur)$ *xform*: [*xss*, *xcop*]
   $xcop \triangleq xform.xcop$
   $xcur \triangleq scur \cup \{cop.oid\}$
  IN $\land\ s2ss' = [cl \in Client \mapsto$
                IF $cl = c$
                THEN $s2ss[cl] \oplus xform.xss$

```
                        ELSE  s2ss[cl] ⊕ [node ↦ {xcur},
                              edge ↦ {[from ↦ scur, to ↦ xcur, cop ↦ xcop]}]
                    ]
        ∧ state' = [state EXCEPT ![Server] = Apply(xcop.op, @)]
        ∧ Comm(Cop)!SSendSame(c, xcop)   broadcast the transformed operation
```

The *Server* receives a message.

```
SRev ≜
    ∧ Comm(Cop)!SRev
    ∧ LET cop ≜ Head(sincoming)
      IN  ServerPerform(cop)
    ∧ SRevCtx
    ∧ UNCHANGED ⟨chins, c2ss⟩
----------------------------------------------------------------
Next ≜
    ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
    ∨ SRev

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

Spec ≜ Init ∧ □[Next]_vars   ∧ Fairness
----------------------------------------------------------------
```

In *Jupiter* (not limited to *XJupiter*), each client synchronizes with the server. In *XJupiter*, this is expressed as the following *CSSync* property.

```
CSSync ≜
    ∀ c ∈ Client : (ds[c] = ds[Server]) ⇒ c2ss[c] = s2ss[c]
```

```
\* Modification History
\* Last modified Mon Dec 24 11:38:04 CST 2018 by hengxin
\* Created Tue Oct 09 16:33:18 CST 2018 by hengxin
```