MODULE *Onomy*

EXTENDS *Naturals*, *Sequences*, *FiniteSets*

CONSTANT *Nonce*,
Peer,
User,
Adr,
Val,
InitNetInt

VARIABLE *adrQ*,
buf,
aCtl,
tCtl,
store,
hash,
netInt,
rcvBuf,
sndBuf,
*r*1,
*r*1*Count*,
commitQ

Net $\triangleq$ INSTANCE *NetworkInterface*
Str $\triangleq$ INSTANCE *Storage*
Usr $\triangleq$ INSTANCE *User*
*Rnd*1 $\triangleq$ INSTANCE *Round*1

NoVal $\triangleq$ CHOOSE $v : v \notin Val$

Status $\triangleq$ {"rdy", "submit", "round 1", "round 2", "round 3", "commit"}

TypeInvariant $\triangleq$

Each peer maintains an operation queue per storage address
$\land\ adrQ \in [Peer \to [Adr \to Seq(Nat)]]$
Each peer tracks the status of each address
$\land\ aCtl \in [Peer \to [Adr \to Status]]$

Each address for each transaction is tracked Each *Peer* has a set of transactions that are mapped to a set of records of address and status
$\land\ tCtl \in [Peer \to [Nat \to \{\langle Adr,\ Val,\ Status\rangle\}]]$

Ordered commit queue of transactions. Transactions must be placed in commit queue before an effected address is unlocked and available for voting
$\land\ commitQ \in [Peer \to Seq([Nat \to \{\langle Adr,\ Val\rangle\}])]$
Round 1 accounting of votes by peers

$IInit \triangleq$

$\land\ Str!StorageInit$
$\land\ Usr!UInit$
$\land\ Net!NetInit$
$\land\ Rnd1!Round1Init$
The queue for each address at each peer is empty
$\land\ adrQ \in [p \in Peer \mapsto [a \in Adr \mapsto \langle\rangle]]$
The control var for each address at each peer is *rdy*
$\land\ aCtl = [p \in Peer \mapsto [a \in Adr \mapsto \text{“rdy”}]]$
The control var for each hash at each peer is empty
$\land\ tCtl = [p \in Peer \mapsto \{\}]$

$\land\ commitQ = [p \in Peer \mapsto \{\}]$

Peer p issues address a request For some user request *req*:

$Req(p) \triangleq$

$\land \exists\, req \in Usr!UReq :$
Peer p places msg on the network interface
$\land\ sndBuf' = Append(sndBuf[p], \langle hash, req.vals, \text{“submit”}\rangle)$
Peer p places msg on its own receive buffer
$\land\ rcvBuf' = Append(rcvBuf[p], \langle hash, req.vals, \text{“submit”}\rangle)$
Update ctl status of the memory address
$\land\ ctl' = [ctl \ \text{EXCEPT}\ ![p][a] = \text{“queued”}]$
Increment the counter representing unique hash
$\land\ hash' = hash + 1$
$\land\ \text{UNCHANGED}\ \langle adrQ, store\rangle$

Add storage updates to peer-wise address queues. Each update is stored in $< hash, val >$ tuple in sequence by address $o[1] \triangleq$ address $o[2] \triangleq$ value

$AddAdrCtl(p, h, op) \triangleq aCtl' = [o \in op \mapsto Append(adrQ[p][o[1]], \langle h, o[2]\rangle)]$

Set control variable for each address in the transaction to submit

$AddTxCtl(p, h, op) \triangleq tCtl' = [tCtl \ \text{EXCEPT}\ ![p] = \text{UNION}\ \{tCtl[p], [h \mapsto \{\langle o[1], o[2], \text{“submit”}\rangle : o \in op\}]\}]$

$Submit(p) \triangleq$

Receive buffer is not empty
$\land\ Len(rcvBuf[p]) > 0$
Receive buffer is set to submit
$\land\ (Head(rcvBuf[p][3]) = \text{“submit”})$
$\land\ \text{LET}$
Set of address X values tuples in the transaction

```
        op ≜ Head(rcvBuf[p])[4]
        Hash of transaction
        h ≜ Head(rcvBuf[p])[3]
    IN  AddTxCtl(p, h, op) ∧ AddAdrCtl(p, h, op)
∧ UNCHANGED ⟨aCtl, sndBuf, rcvBuf⟩

CommitTx(p) ≜
        Check to see if any tx are ready to commit
    [h ∈ DOMAIN tCtl[p] ↦
      IF ∀ a ∈ DOMAIN tCtl[p][h] : a[2] = "commit"
       THEN [b ∈ DOMAIN tCtl[p][h] ↦
                store' = [store EXCEPT
                    ![p][b[1]].val = b[2],
                    ![p][b[1]].nonce = n,
                    ![p][b[1]].hash = h
                ]
            ]
       ELSE UNCHANGED store
    ]
```

* Modification History
* Last modified *Tue Oct* 13 21:41:02 *CDT* 2020 by *dninja*
* Created Sun *Sep* 13 11:03:56 *CDT* 2020 by *dninja*