MODULE *OpAWSet*

EXTENDS *AWSet*
CONSTANTS *Read*(_)

VARIABLES

aset,	*aset*[*r*]: the set of active elements maintained by $r \in Replica$
abuf,	*abuf*[*r*]: the buffer for elements added by $r \in Replica$ since the last broadcast
rbuf,	*rbuf*[*r*]: the buffer for elements removed by $r \in Replica$ since the last broadcast
variables for network:	
incoming,	*incoming*[*r*]: incoming channel at replica $r \in Replica$
lmsg,	*lmsg*[*r*]: the last message delivered at $r \in Replica$ to the upper-layer protocol
vc,	*vc*[*r*][*s*] denotes the latest message from $s \in Replica$ observed by $r \in Replica$
variables for *SEC*:	
uset,	*uset*[*r*]: the set of updates seen by replica $r \in Replica$
uincoming,	*uincoming*[*r*]: incoming channel for broadcasting/delivering updates at $r \in Replica$
buset	*buset*[*r*]: the buffer of local updates made by $r \in Replica$ since the last broadcast

$nVars \triangleq \langle incoming, lmsg, vc \rangle$
$secVars \triangleq \langle uset, uincoming, buset \rangle$
$vars \triangleq \langle aset, abuf, rbuf, seq, nVars, secVars \rangle$

$Msg \triangleq [aid : Aid, abuf : \text{SUBSET } Element, rbuf : \text{SUBSET } Element, lvc : [Replica \to Nat]]$
$Network \triangleq \text{INSTANCE } ReliableCausalNetwork \text{ WITH } incoming \leftarrow incoming, lmsg \leftarrow lmsg, vc \leftarrow vc$

$ReadOpAWSet(r) \triangleq \{ele.d : ele \in aset[r]\}$ read the state of $r \in Replica$
$SEC \triangleq \text{INSTANCE } OpSEC \text{ WITH } uset \leftarrow uset, uincoming \leftarrow uincoming, buset \leftarrow buset$

$TypeOK \triangleq$
- $\land\ aset \in [Replica \to \text{SUBSET } Element]$
- $\land\ abuf \in [Replica \to \text{SUBSET } Element]$
- $\land\ rbuf \in [Replica \to \text{SUBSET } Element]$
- $\land\ IntTypeOK$
- $\land\ Network!SMTypeOK$
- $\land\ SEC!SECTypeOK$

$Init \triangleq$
- $\land\ aset = [r \in Replica \mapsto \{\}]$
- $\land\ abuf = [r \in Replica \mapsto \{\}]$
- $\land\ rbuf = [r \in Replica \mapsto \{\}]$
- $\land\ IntInit$
- $\land\ Network!RCNInit$
- $\land\ SEC!OpSECInit$

1
1
1

$Add(d, r) \triangleq$ $r \in Replica$ adds $d \in Data$

$\land$ LET $e \triangleq [aid \mapsto [r \mapsto r,\ seq \mapsto seq[r]],\ d \mapsto d]$
IN $\land$ $aset' = [aset$ EXCEPT $![r] = @ \cup \{e\}]$
$\land$ $abuf' = [abuf$ EXCEPT $![r] = @ \cup \{e\}]$
$\land$ $IntDo(r)$
$\land$ $SEC!OpSECDo(r)$
$\land$ UNCHANGED $\langle rbuf,\ nVars\rangle$

$Remove(d, r) \triangleq$ $r \in Replica$ removes $d \in Data$

$\land$ LET $E \triangleq \{ele \in aset[r] : ele.d = d\}$ E may be empty
IN $\land$ $aset' = [aset$ EXCEPT $![r] = @ \setminus E]$
$\land$ $rbuf' = [rbuf$ EXCEPT $![r] = @ \cup E]$
$\land$ $IntDo(r)$
$\land$ $SEC!OpSECDo(r)$
$\land$ UNCHANGED $\langle abuf,\ nVars\rangle$

$Do(r) \triangleq$ We ignore $ReadOpAWSet(r)$ since it does not modify states.

$\exists\, d \in Data : Add(d, r) \lor Remove(d, r)$

$Send(r) \triangleq$ $r \in Replica$ sends a message

$\land$ $\lor$ $abuf[r] \neq \{\}$
$\lor$ $rbuf[r] \neq \{\}$
$\land$ $abuf' = [abuf$ EXCEPT $![r] = \{\}]$
$\land$ $rbuf' = [rbuf$ EXCEPT $![r] = \{\}]$
$\land$ $Network!RCNBroadcast(r, [aid \mapsto [r \mapsto r,\ seq \mapsto seq[r]],$
$abuf \mapsto abuf[r],\ rbuf \mapsto rbuf[r]])$
$\land$ $SEC!OpSECSend(r)$
$\land$ $IntSend(r)$
$\land$ UNCHANGED $\langle aset\rangle$

$Deliver(r) \triangleq$ $r \in Replica$ delivers a message ($lmsg'[r]$)

$\land$ $Network!RCNDeliver(r)$
$\land$ $SEC!OpSECDeliver(r,\ lmsg'[r].aid)$
$\land$ $aset' = [aset$ EXCEPT $![r] = (@ \cup lmsg'[r].abuf) \setminus lmsg'[r].rbuf]$
$\land$ $IntDeliver(r)$
$\land$ UNCHANGED $\langle abuf,\ rbuf\rangle$

$Next \triangleq \exists\, r \in Replica : Do(r) \lor Send(r) \lor Deliver(r)$

$Spec \triangleq Init \land \Box[Next]_{vars}$

---

* Modification History
* Last modified *Wed Jun* 26 13:30:44 *CST* 2019 by *xhdn*
* Created *Fri* May 24 14:12:26 *CST* 2019 by *xhdn*