MODULE $OpAWSet$

EXTENDS $AWSet$

VARIABLES

set, $set[r]$: set of $Element(s)$ maintained by $r \in Replica$
$abuf$, $abuf[r]$: buffer of $Element(s)$ added maintained by $r \in Replica$
$rbuf$, $rbuf[r]$: buffer of $Element(s)$ removed maintained by $r \in Replica$

network variables

$incoming$,
$dmsg$,
$lmsg$,
vc,

SEC variables

$updateset$,
$uincoming$,
$new\_updateset$

$Nvars \triangleq \langle incoming, lmsg, dmsg, vc \rangle$
$SECvars \triangleq \langle updateset, new\_updateset, uincoming \rangle$
$vars \triangleq \langle set, abuf, rbuf, seq, Nvars, SECvars \rangle$

$Msg \triangleq [r : Replica, seq : Nat, vc : Vector, abuf : \text{SUBSET } Element, rbuf : \text{SUBSET } Element]$ message type

instantiate a reliable causal network
$Network \triangleq$ INSTANCE $ReliableCausalNetwork$

read the state of $r \in Replica$
$Read(r, s) \triangleq \{ele.d : ele \in s\}$
instantiate SEC module
$SEC \triangleq$ INSTANCE $OpSEC$ WITH $data \leftarrow set$

$TypeOK \triangleq$ check types
$\land$ $set \in [Replica \rightarrow \text{SUBSET } Element]$
$\land$ $abuf \in [Replica \rightarrow \text{SUBSET } Element]$
$\land$ $rbuf \in [Replica \rightarrow \text{SUBSET } Element]$
$\land$ $IntTypeOK$
$\land$ $Network!SMTypeOK$
$\land$ $SEC!SECTypeOK$

1
1
1
1
1
1

```
$Init \triangleq$  initial state
    $\land\ set = [r \in Replica \mapsto \{\}]$
    $\land\ abuf = [r \in Replica \mapsto \{\}]$
    $\land\ rbuf = [r \in Replica \mapsto \{\}]$
    $\land\ IntInit$
    $\land\ Network!RCInit$
    $\land\ SEC!OpSECInit$

$Send(r) \triangleq$  $r \in Replica$ sends a message
    $\land\ abuf' = [abuf \text{ EXCEPT } ![r] = \{\}]$
    $\land\ rbuf' = [rbuf \text{ EXCEPT } ![r] = \{\}]$
    $\land\ Network!RCBroadcast(r, [r \mapsto r, seq \mapsto seq[r], vc \mapsto [vc \text{ EXCEPT } ![r][r] = @ + 1][r],$
                                                   $abuf \mapsto abuf[r], rbuf \mapsto rbuf[r]])$
    $\land\ SEC!OpSECSend(r, seq[r])$
    $\land\ IntSend(r)$
    $\land\ \text{UNCHANGED } \langle set \rangle$

$Deliver(r) \triangleq$  $r \in Replica$ receives a message
    $\land\ Network!RCDeliver(r)$
    $\land\ SEC!OpSECDeliver(r, [r \mapsto lmsg'.r, seq \mapsto lmsg'.seq])$
    $\land\ set' = [set \text{ EXCEPT } ![r] = (@ \cup lmsg'.abuf) \setminus lmsg'.rbuf]$
    $\land\ IntDeliver(r)$
    $\land\ \text{UNCHANGED } \langle abuf, rbuf \rangle$

$Add(d, r) \triangleq$  $r \in Replica$ adds $d \in Data$
    $\land\ set' = [set \text{ EXCEPT } ![r] = @ \cup \{[d \mapsto d, r \mapsto r, k \mapsto seq[r]]\}]$
    $\land\ abuf' = [abuf \text{ EXCEPT } ![r] = @ \cup \{[d \mapsto d, r \mapsto r, k \mapsto seq[r]]\}]$
    $\land\ IntDo(r)$
    $\land\ Network!RCDo$
    $\land\ SEC!OpSECUpdate(r, seq[r])$
    $\land\ \text{UNCHANGED } \langle rbuf \rangle$

$Remove(d, r) \triangleq$  $r \in Replica$ removes $d \in Data$
    $\land\ \{ele \in set[r] : ele.d = d\} \neq \{\}$
    $\land\ \text{LET } D \triangleq \{ele \in set[r] : ele.d = d\}$
      $\text{IN}$  $\land\ set' = [set \text{ EXCEPT } ![r] = @ \setminus D]$
           $\land\ rbuf' = [rbuf \text{ EXCEPT } ![r] = @ \cup D]$
    $\land\ IntDo(r)$
    $\land\ Network!RCDo$
    $\land\ SEC!OpSECUpdate(r, seq[r])$
    $\land\ \text{UNCHANGED } \langle abuf \rangle$

$Do(r) \triangleq$  update operations
    $\exists\, d \in Data : Add(d, r) \lor Remove(d, r)$
```

$Next \triangleq$ next-state relation

$\quad \exists\, r \in Replica :$

$\qquad Deliver(r) \lor Send(r) \lor Do(r)$

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

$SECa \triangleq \forall\, r1,\ r2 \in Replica :$

$\qquad SEC!Sameupdate(r1,\ r2) \Rightarrow Read(r1) = Read(r2)$

\* Modification History

\* Last modified *Thu Jun* 13 21:46:27 *CST* 2019 by *xhdn*

\* Created *Fri* May 24 14:12:26 *CST* 2019 by *xhdn*