MODULE *AWSet*

EXTENDS *Naturals*, *Sequences*, *SEC*

CONSTANTS

Data the set of data

VARIABLES

aSet, *aSet*[*r*]: set of active *Instance*(*s*) maintained by $r \in$ *Replica*
tSet, *tSet*[*r*]: set of tombstone *Instance*(*s*) maintained by $r \in$ *Replica*
seq, *seq*[*r*]: local sequence number at replica $r \in$ *Replica*
incoming, *incoming*[*r*]: incoming messages at replica $r \in$ *Replica*
msg,
messageSet

$vars \triangleq \langle aSet, tSet, seq, incoming, msg, messageSet, SECvars \rangle$

$Instance \triangleq [d : Data, r : Replica, k : Nat]$
$Msg \triangleq [r : Replica, A : \text{SUBSET } Instance, T : \text{SUBSET } Instance, seq : Nat,$
$update : \text{SUBSET } Update]$

$Network \triangleq \text{INSTANCE } Network$

$TypeOK \triangleq$
$\land \quad aSet \in [Replica \to \text{SUBSET } Instance]$
$\land \quad tSet \in [Replica \to \text{SUBSET } Instance]$
$\land \quad seq \in [Replica \to Nat]$

$Init \triangleq$
$\land\ aSet = [r \in Replica \mapsto \{\}]$
$\land\ tSet = [r \in Replica \mapsto \{\}]$
$\land\ seq = [r \in Replica \mapsto 0]$
$\land\ Network!NInit$
$\land\ SECInit$

$Add(d, r) \triangleq$
$\land\ seq' = [seq \text{ EXCEPT } ![r] = @ + 1]$
$\land\ aSet' = [aSet \text{ EXCEPT } ![r] = @ \cup \{[d \mapsto d, r \mapsto r, k \mapsto seq'[r]]\}]$
$\land\ SECUpdate(r, seq[r])$
$\land\ \text{UNCHANGED } \langle tSet, incoming, msg, messageSet \rangle$

$Remove(d, r) \triangleq$
$\land\ \text{LET } D \triangleq \{ins \in aSet[r] : ins.d = d\}$
IN $\land\ aSet' = [aSet \text{ EXCEPT } ![r] = @ \setminus D]$
$\land\ tSet' = [tSet \text{ EXCEPT } ![r] = @ \cup D]$
$\land\ seq' = [seq \text{ EXCEPT } ![r] = @ + 1]$
$\land\ SECUpdate(r, seq[r])$

$\land$ UNCHANGED $\langle incoming,\ msg,\ messageSet \rangle$

$Broadcast(s,\ m) \triangleq$
$[r \in Replica \mapsto \text{IF } s = r \text{ THEN } incoming[s]$
$\text{ELSE } incoming[r] \circ \langle m \rangle]$

$Send(r) \triangleq$
$\land\ Network!NBroadcast(r,\ [r \mapsto r,\ A \mapsto aSet[r],\ T \mapsto tSet[r],\ seq \mapsto seq[r],$
$update \mapsto StateUpdate(r)])$
$\land\ SECSend(r)$
$\land$ UNCHANGED $\langle aSet,\ tSet,\ seq \rangle$

$Receive(r) \triangleq$
$\land\ Network!NDeliver(r)$
$\land\ SECDeliver(r,\ msg'[r])$
$\land\ tSet' = [tSet \text{ EXCEPT } ![r] = @ \cup msg'[r].T]$
$\land\ aSet' = [aSet \text{ EXCEPT } ![r] = (@ \cup msg'[r].A) \setminus tSet'[r]]$
$\land\ seq' = [seq \text{ EXCEPT } ![r] = @ + 1]$
$\land$ UNCHANGED $\langle\rangle$

$Next \triangleq$
$\lor \exists\, r \in Replica : \exists\, a \in Data :$
$Add(a,\ r) \lor Remove(a,\ r)$
$\lor \exists\, r \in Replica :$
$Send(r) \lor Receive(r)$

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

$Read(r) \triangleq \{ins.d : ins \in aSet[r]\}$

QC: Quiescent Consistency
$Quiescence \triangleq \forall\, r \in Replica : incoming[r] = \langle\rangle$

$Convergence \triangleq \forall\, r,\ s \in Replica : Read(r) = Read(s)$

$QC \triangleq Quiescence \Rightarrow Convergence$

SEC: Strong Eventual Consistency
$SEC \triangleq \forall\, r1,\ r2 \in Replica : SameUpdate(r1,\ r2) \Rightarrow Read(r1) = Read(r2)$

---

* Modification History
* Last modified *Thu* May 16 09:19:03 *CST* 2019 by *zfwang*
* Last modified Sun *Apr* 28 15:09:12 *CST* 2019 by *jywellin*
* Created Sun *Apr* 28 14:02:54 *CST* 2019 by *jywellin*