```
MODULE OptimizedAWSet
EXTENDS Naturals, Sequences, SEC
```

```
CONSTANTS
    Data

VARIABLES
    aSet,
    seq,
    vc,
    incoming,    network variable
    msg,         network variable
    messageset
```

$vars \triangleq \langle aSet, seq, vc, incoming, msg, messageset, SECvars \rangle$

$Vector \triangleq [Replica \rightarrow Nat]$
$Initvector \triangleq [r \in Replica \mapsto 0]$
$Instance \triangleq [d : Data, r : Replica, k : Nat]$
$Msg \triangleq [r : Replica, seq : Nat, vc : Vector, update : \text{SUBSET } Update, A : \text{SUBSET } Instance]$

$Network \triangleq \text{INSTANCE } Network$

```
TypeOK ≜
    ∧  $aSet \in [Replica \rightarrow \text{SUBSET } Instance]$
    ∧  $seq \in [Replica \rightarrow Nat]$
    ∧  $vc \in [Replica \rightarrow Vector]$
```

```
Init ≜
    ∧ $Network!NInit$
    ∧ $SECInit$
    ∧ $aSet = [r \in Replica \mapsto \{\}]$
    ∧ $seq = [r \in Replica \mapsto 0]$
    ∧ $vc = [r \in Replica \mapsto Initvector]$
```

```
Add(d, r) ≜
      ∧ $seq' = [seq \text{ EXCEPT } ![r] = @ + 1]$
      ∧ $SECUpdate(r, seq[r])$
      ∧ LET $D \triangleq \{ins \in aSet[r] : ins.d = d \wedge ins.r = r\}$
          IN  $aSet' = [aSet \text{ EXCEPT } ![r] = (@ \cup \{[d \mapsto d, r \mapsto r, k \mapsto vc[r][r] + 1]\}) \setminus D]$
      ∧ $vc' = [vc \text{ EXCEPT } ![r][r] = @ + 1]$
      ∧ UNCHANGED $\langle incoming, msg, messageset \rangle$

Remove(d, r) ≜
    ∧ $seq' = [seq \text{ EXCEPT } ![r] = @ + 1]$
    ∧ $SECUpdate(r, seq[r])$
    $\land$ LET $D \triangleq \{ins \in aSet[r] : ins.d = d\}$
       IN      $aSet' = [aSet$ EXCEPT $![r] = @ \setminus D]$
    $\land$ UNCHANGED $\langle vc, incoming, msg, messageset \rangle$
```

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

$SetMax(r, s) \triangleq$ IF $r > s$ THEN $r$ ELSE $s$

$Deliver(r) \triangleq$
    $\land$  $Network!NDeliver(r)$
    $\land$  $SECDeliver(r, msg[r]')$
    $\land$  $seq' = [seq$ EXCEPT $![r] = @ + 1]$
    $\land$  LET $Diff1 \triangleq \{ins \in aSet[r] \quad : \neg ins \in msg[r]'.A\}$
               $Diff2 \triangleq \{ins \in msg[r]'.A : \neg ins \in aSet[r]\ \}$
               $D1 \triangleq \{ins \in Diff1 : ins.k \leq msg[r]'.vc[ins.r]\}$
               $D2 \triangleq \{ins \in Diff2 : ins.k \leq vc[r][ins.r]\}$
               $Alocal \triangleq aSet[r] \setminus D1$
                $Aremote \triangleq msg[r]'.A \setminus D2$
         IN    $aSet' = [aSet$ EXCEPT $![r] = Alocal \cup Aremote]$
    $\land$  $\forall s \in Replica : vc' = [vc$ EXCEPT $![r][s] = SetMax(@, msg'[r].vc[s])]$
    $\land$  UNCHANGED $\langle vc \rangle$
```

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

$Spec \triangleq Init \land \Box[Next]_{vars}$  $\land \mathrm{WF\_}vars(Next)$
```

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

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

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

$QC \triangleq Quiescence \Rightarrow Convergence$

$SEC \triangleq \forall r1, r2 \in Replica : Sameupdate(r1, r2)$
                 $\Rightarrow Read(r1) = Read(r2)$
```

\ * Modification History
\ * Last modified *Tue* May 07 01:14:37 *CST* 2019 by *xhdn*
\ * Last modified *Mon* May 06 15:58:14 *CST* 2019 by *jywellin*
\ * Created Sun *Apr* 28 13:52:27 *CST* 2019 by *jywellin*