MODULE *ORSet*

EXTENDS *Naturals*, *Sequences*, *Bags*, *TLC*, *SEC*

CONSTANTS
Data the set of data

$Instance \triangleq [d : Data, r : Replica, k : Nat]$

VARIABLES
sSet, *sSet*[*r*]: set of active *Instance*(*s*) maintained by $r \in Replica$
seq, *seq*[*r*]: local sequence number at replica $r \in Replica$

Network variables

incoming, *incoming*[*r*]: incoming messages at replica $r \in Replica$
msg,
messageSet

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

$Msg \triangleq [r : Replica, update : \text{SUBSET } Update, seq : Nat, S : \text{SUBSET } Instance]$

$Network \triangleq \text{INSTANCE } ReliableNetwork$

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

$Init \triangleq$
$\land\ seq = [r \in Replica \mapsto 0]$
$\land\ sSet = [r \in Replica \mapsto \{\}]$
$\land\ incoming = [r \in Replica \mapsto \langle\rangle]$
$\land\ Network\,!\,RInit$
$\land\ SECInit$

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

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

```
    IN   sSet' = [sSet EXCEPT ![r] = @ \ D]
  ∧ UNCHANGED ⟨incoming, msg, messageSet⟩
```

```
Broadcast(s, m) ≜
  [r ∈ Replica ↦ IF s = r THEN incoming[s]
                          ELSE incoming[r] ⊕ SetToBag({m})]

Send(r) ≜
  ∧ Network!RBroadcast(r, [r ↦ r, seq ↦ seq[r], update ↦ OpUpdate(r), S ↦ sSet[r]])
  ∧ SECSend(r)
  ∧ UNCHANGED ⟨sSet, seq, sSet⟩

Receive(r) ≜
  ∧  Network!RDeliver(r)
  ∧  SECDeliver(r, msg'[r])
  ∧  sSet' = [sSet EXCEPT ![r] = @ ∪ msg'[r].S]
  ∧  UNCHANGED ⟨seq⟩
```

```
Next ≜
  ∨ ∃ r ∈ Replica : ∃ a ∈ Data :
      Add(a, r) ∨ Remove(a, r)
  ∨ ∃ r ∈ Replica :
      Send(r) ∨ Receive(r)

Spec ≜ Init ∧ □[Next]_vars
```

```
Read(r) ≜ {ins.d : ins ∈ sSet[r]}
```

QC: Quiescent Consistency

```
Quiescence ≜
  ∀ r ∈ Replica : incoming[r] = ⟨⟩

Convergence ≜
  ∀ r, s ∈ Replica : Read(r) = Read(s)

QC ≜ Quiescence ⇒ Convergence
```

SEC: Strong Eventual Consistency

```
SEC ≜ ∀ r1, r2 ∈ Replica :
        Sameupdate(r1, r2) ⇒ Read(r1) = Read(r2)
```

* Modification History
* Last modified *Tue* May 14 11:23:35 *CST* 2019 by *zfwang*
* Created *Wed Feb* 27 17:23:32 *CST* 2019 by *zfwang*