MODULE *ConsensusProtocol*

EXTENDS *Integers*, *Sequences*, *TLC*, *Naturals*

CONSTANT
Names, The set of the participants
PossibleAllocations, The set of possible allocations
P, The state channel participants (ie. the *Names* set), in order
A The desired allocations of each participant (ie. the *PossibleAllocations* set), in order

ASSUME
$\land Len(P) = Len(A)$

VARIABLES
pState, *pState*[*p*] is the state of participant *p*
msgs

In the protocol, processes communicate with one another by sending messages. For simplicity, we represent message passing with the variable *msgs* whose value is the set of all messages that have been sent. A message is sent by adding it to the set *msgs*. An action that, in an implementation, would be enabled by the receipt of a certain message is here enabled by the presence of that message in *msgs*. When an action is enabled, the message is deleted from *msgs*.

$NumParticipants \triangleq Len(P)$
$Participants \triangleq \text{DOMAIN } P$
$Messages \triangleq$
[
turnNumber : *Nat*,
votesRequired : $(\text{DOMAIN } P) \cup \{0\}$,
to : DOMAIN *P*,
allocation : *PossibleAllocations*
]

$States \triangleq \{\}$
$\cup$ [*allocation* : *PossibleAllocations*, *turnNumber* : *Nat*, *type* : {"Waiting"}]
$\cup$ [*allocation* : *PossibleAllocations*, *turnNumber* : *Nat*, *type* : {"Sent"}, *status* : {"Voted", "Rejected"}]

$TypeOK \triangleq$
The type-correctness invariant
$\land PrintT(\langle msgs, pState \rangle)$ Debugging statement
The following two conditions specify the format of each message and participant state
$\land pState \in [\text{DOMAIN } P \to States]$
$\land msgs \subseteq Messages$

$Init \triangleq$
$\land pState = [p \in \text{DOMAIN } P \mapsto [$
allocation $\mapsto A[p]$,
turnNumber $\mapsto 1$, At this point, assume everyone starts at the same turn number.

```
      type ↦ "Waiting"
    ]]
  ∧ msgs = {}

NextParticipant(p) ≜ 1 + ((p + 1)%NumParticipants)
isParticipantsTurn(state, p) ≜ p = 1 + (state[p].turnNumber%NumParticipants)

VoteMsg(p, a, n) ≜ [
  to         ↦ NextParticipant(p),
  allocation ↦ a,
  turnNumber ↦ n
]

RejectMsg(p) ≜ "RejectMessage"
```

We now define the actions that may be performed by the participants

When it's the participant's turn, they are allowed to vote. When they are the first person to vote, they kick off a voting round for their allocation. Otherwise, they decrement *furtherVottesRequired*

```
Vote(p) ≜
  ∧ isParticipantsTurn(pState, p)
  ∧ pState' = [pState EXCEPT ![p] = [
                type       ↦ "Sent",
                status     ↦ "Voted",
                allocation ↦ pState[p].allocation
              ]]
  ∧ msgs' = msgs ∪ {VoteMsg(p, pState[p].allocation, pState.turnNumber)}

Reject(p) ≜  TODO: This
  ∧ msgs' = msgs ∪ {RejectMsg(p)}
  ∧ UNCHANGED ⟨pState⟩
```

Reading a message updates the destination participant's state if and only if it increases their *turnNumber*

```
UpdatedPState(msg, p) ≜ LET state ≜ pState[p]
  IN
    IF msg.turnNumber ≤ state.turnNumber
     THEN state
     ELSE state  TODO: Actually update the state

ReadMsg(m) ≜
  ∧ msgs' = msgs \ {m}
  ∧ pState' = [pState EXCEPT ![m.to] = UpdatedPState(m, m.to)]

Next ≜
```

$\lor \exists\, p \in \text{DOMAIN } P : Vote(p) \lor Reject(p)$
$\lor \exists\, m \in msgs \quad : ReadMsg(m)$

$Spec \triangleq Init \land \quad \Box[Next]_{\langle pState,\, msgs\rangle}$

THEOREM $Spec \Rightarrow \Box\, TypeOK$

This theorem asserts that the type-correctness predicate *TypeOK* is an invariant of the specification.

* Modification History
* Last modified *Tue Aug* 06 14:31:25 *MDT* 2019 by *andrewstewart*
* Created *Fri Aug* 02 12:15:55 *MDT* 2019 by *andrewstewart*