MODULE *LessWriteyPaxos*

EXTENDS *Integers*

CONSTANTS *Rounds*, *Acceptors*, *ProposableValues*,
PromiseQuorums, *AcceptQuorums*, *Epochs*

VARIABLES *acceptor_states*, *learnt*, *sent_messages*

$None \triangleq$ "NONE" CHOOSE $v : v \notin ProposableValues$

ASSUME $\land Rounds \subseteq Nat$
$\land AcceptQuorums \subseteq \text{SUBSET } Acceptors$
$\land PromiseQuorums \subseteq \text{SUBSET } Acceptors$
$\land \forall promise\_quorum \in PromiseQuorums, accept\_quorum \in AcceptQuorums :$
$(promise\_quorum \cap accept\_quorum) \neq \{\}$
$\land None \notin ProposableValues$

$AllowedMessages \triangleq$
$[type : \{\text{"promise request"}\},$
$round : Rounds]$
$\cup \quad [type : \{\text{"Epoch promise"}\},$
$acceptor : Acceptors,$
$round : Rounds,$
$Epoch : Epochs,$
$max\_accepted\_Epoch : Epochs \cup \{-1\},$
$max\_accepted\_round : Rounds \cup \{-1\},$
$max\_accepted\_value : ProposableValues \cup \{None\}]$
$\cup \quad [type : \{\text{"accept request"}\},$
$Epoch : Epochs,$
$round : Rounds,$
$value : ProposableValues]$
$\cup \quad [type : \{\text{"accepted"}\},$
$acceptor : Acceptors,$
$Epoch : Epochs,$
$round : Rounds,$
$value : ProposableValues]$
$\cup \quad [type : \{\text{"new Epoch"}\},$
$Epoch : Epochs]$

$EpochPaxosTypeInvariant \triangleq$
$\land acceptor\_states \in [Acceptors \to [current\_Epoch : Epochs,$
$max\_round\_promised : Rounds \cup \{-1\} \cup \{1\},$
$max\_accepted\_Epoch : Epochs \cup \{-1\},$

$max\_accepted\_round \ : Rounds \cup \{-1\},$
$max\_accepted\_value \ : ProposableValues \cup \{None\}]]$ acceptor states

$\land\ sent\_messages \subseteq AllowedMessages$
$\land\ learnt \subseteq ProposableValues$

$InitalState \triangleq$
$\land\ sent\_messages = \{\}$ no messages sent
$\land\ acceptor\_states = [acceptor \in Acceptors \mapsto [current\_Epoch \mapsto 1,$
$max\_round\_promised \mapsto -1,$
$max\_accepted\_Epoch \mapsto -1,$
$max\_accepted\_round \mapsto -1,$
$max\_accepted\_value \mapsto None$
$]]$ no ballots promised
$\land\ learnt = \{\}$ no values *learnt*

$Send(message) \triangleq sent\_messages' = sent\_messages \cup \{message\}$

$MessageSent(message) \triangleq \exists\, sent\_message \in sent\_messages : sent\_message = message$

$SendPromiseRequests(requesting\_round) \triangleq$
$\land\ \neg MessageSent([type \mapsto$ "promise request"$,\ round \mapsto requesting\_round])$
$\land\ Send([type \mapsto$ "promise request"$,\ round \mapsto requesting\_round])$
$\land$ UNCHANGED $\langle learnt,\ acceptor\_states\rangle$

$SendEpochPromise(acceptor,\ requested\_round) \triangleq$
$\land\ MessageSent([type \mapsto$ "promise request"$,\ round \mapsto requested\_round])$
$\land\ acceptor\_states[acceptor].max\_round\_promised \leq requested\_round$
$\land\ acceptor\_states' = [acceptor\_states$ EXCEPT $![acceptor].max\_round\_promised = requested\_round]$
$\land\ Send([type \mapsto$ "Epoch promise"$,$
$acceptor \mapsto acceptor,$
$round \mapsto requested\_round,$
$Epoch \mapsto acceptor\_states[acceptor].current\_Epoch,$
$max\_accepted\_Epoch \mapsto acceptor\_states[acceptor].max\_accepted\_Epoch,$
$max\_accepted\_round \mapsto acceptor\_states[acceptor].max\_accepted\_round,$
$max\_accepted\_value \mapsto acceptor\_states[acceptor].max\_accepted\_value$
$])$
$\land$ UNCHANGED $learnt$

$IsEpochPromiseQuorum(Epoch,\ round,\ quorum) \triangleq$
$\forall\, acceptor \in quorum :$
$\exists\, message \in sent\_messages : \land\ message.type =$ "Epoch promise"

$\land\ message.acceptor = acceptor$
$\land\ message.Epoch = Epoch$
$\land\ message.round = round$

no higher ordered ballot accepted by any other acceptor in the promise quorum

PromiseQuorumMessages
all promises in the quorum contain no value accepteds
or the highest ordered *Epoch* ballot accepted has the value of the one being proposed

$IsValueSafeInEpochPromiseQuorum(quorum,\ Epoch,\ round,\ value) \triangleq$
$\lor \exists\, message\_a \in sent\_messages :\ \land\ message\_a.type = \text{“Epoch promise”}$
$\land\ message\_a.Epoch = Epoch$
$\land\ message\_a.round = round$
$\land\ message\_a.max\_accepted\_value = value$
$\land\ message\_a.acceptor \in quorum$
$\land \forall\, message\_b \in sent\_messages \setminus \{message\_a\} :$
$(\ \land\ message\_b.type = \text{“Epoch promise”}$
$\land\ message\_b.max\_accepted\_value \neq message\_a.max\_accepted\_val$
$\land\ message\_b.acceptor \in quorum$
$\land\ message\_b.round = round$
$\land\ message\_b.Epoch = Epoch)$
$\Rightarrow (\ \lor\ \land\ message\_b.max\_accepted\_round < message\_a.max\_accepted\_$
$\land\ message\_b.max\_accepted\_Epoch = message\_a.max\_accepted\_$
$\lor\ message\_b.max\_accepted\_Epoch < message\_a.max\_accepted\_E$
$\lor \forall\, acceptor \in quorum :$
$\exists\, message \in sent\_messages :\ \land\ message.type = \text{“Epoch promise”}$
$\land\ message.acceptor = acceptor$
$\land\ message.Epoch = Epoch$
$\land\ message.round = round$
$\land\ message.max\_accepted\_Epoch = -1$
$\land\ message.max\_accepted\_round = -1$
$\land\ message.max\_accepted\_value = None$

$AcceptRequestAlreadyMadeWithDifferentValue(Epoch,\ round,\ value) \triangleq$
$\exists\, other\_value \in ProposableValues : MessageSent([type \mapsto \text{“accept request”},$
$Epoch \mapsto Epoch,$
$round \mapsto round,$
$value \mapsto other\_value])$

$SendAcceptRequest(Epoch,\ round,\ value) \triangleq$
$\land \neg AcceptRequestAlreadyMadeWithDifferentValue(Epoch,\ round,\ value)$
$\land \exists\, promise\_quorum \in PromiseQuorums :\ \land\ IsEpochPromiseQuorum(Epoch,\ round,\ promise\_quorum)$

$\wedge$ $IsValueSafeInEpochPromiseQuorum(promise\_quorum,\ Ep$

$\wedge$ $Send([type \mapsto$ "accept request",
$Epoch \mapsto Epoch,$
$round \mapsto round,$
$value \mapsto value])$
$\wedge$ UNCHANGED $\langle acceptor\_states,\ learnt\rangle$

$SendAcceptedMessage(acceptor,\ Epoch,\ round) \triangleq$
$\wedge$ $acceptor\_states[acceptor].current\_Epoch \leq Epoch$
$\wedge$ $acceptor\_states[acceptor].max\_round\_promised \leq round$
$\wedge \exists\, message \in sent\_messages :$
$\wedge$ $message.type =$ "accept request"
$\wedge$ $message.Epoch = Epoch$
$\wedge$ $message.round = round$
$\wedge$ $acceptor\_states' = [acceptor\_states$ EXCEPT $![acceptor].max\_accepted\_round = round,$
$![acceptor].max\_accepted\_Epoch = Epoch,$
$![acceptor].max\_accepted\_value = message.value,$
$![acceptor].max\_round\_promised = round,$
$![acceptor].current\_Epoch = Epoch]$
$\wedge$ $Send([type \mapsto$ "accepted",
$acceptor \mapsto acceptor,$
$Epoch \mapsto Epoch,$
$round \mapsto round,$
$value \mapsto message.value])$
$\wedge$ UNCHANGED $\langle learnt\rangle$

$Restart(acceptor,\ Epoch) \triangleq$
$\wedge$ $Epoch > \ acceptor\_states[acceptor].current\_Epoch$
$\wedge$ $acceptor\_states' = [acceptor\_states$ EXCEPT $![acceptor].current\_Epoch = Epoch,$
$![acceptor].max\_round\_promised = -1]$
$\wedge$ $MessageSent([type \mapsto$ "new Epoch", $Epoch \mapsto Epoch])$
$\wedge$ UNCHANGED $\langle learnt\rangle$

$BeginNewEpoch(Epoch) \triangleq$
$\wedge \forall\, acceptor\_b \in Acceptors : acceptor\_states[acceptor\_b].current\_Epoch < Epoch$
$\wedge \exists\, acceptor\_b \in Acceptors : acceptor\_states[acceptor\_b].current\_Epoch = (Epoch - 1)$ Epoch 1 higher t
$\wedge$ $Send([type \mapsto$ "new Epoch", $Epoch \mapsto Epoch])$
$\wedge$ UNCHANGED $\langle learnt,\ acceptor\_states\rangle$

$RecieveEpochNotification(acceptor,\ Epoch) \triangleq$
$\wedge$ $acceptor\_states[acceptor].current\_Epoch < Epoch$
$\wedge$ $MessageSent([type \mapsto$ "new Epoch", $Epoch \mapsto Epoch])$
$\wedge$ $acceptor\_states' = [acceptor\_states$ EXCEPT $![acceptor].current\_Epoch = Epoch,$
$![acceptor].max\_round\_promised = -1]$
$\wedge$ UNCHANGED $\langle sent\_messages,\ learnt\rangle$

$$
\begin{array}{l}
Learn(Epoch,\ round,\ value) \triangleq \\
\quad \land \quad \exists\, accept\_quorum \in AcceptQuorums : \\
\qquad \forall\, acceptor \in accept\_quorum : \\
\qquad\quad MessageSent([type \mapsto \text{“accepted”}, \\
\qquad\qquad\qquad\qquad acceptor \mapsto acceptor, \\
\qquad\qquad\qquad\qquad Epoch \mapsto Epoch, \\
\qquad\qquad\qquad\qquad round \mapsto round, \\
\qquad\qquad\qquad\qquad value \ \mapsto value]) \\
\quad \land\, learnt' = learnt \cup \{value\} \\
\quad \land \text{UNCHANGED } \langle acceptor\_states,\ sent\_messages \rangle
\end{array}
$$

$$
\begin{array}{l}
NextState \triangleq \ \lor \exists\, round \in Rounds : \\
\qquad\qquad\qquad \lor\, SendPromiseRequests(round) \\
\qquad\qquad\qquad \lor \exists\, acceptor \in Acceptors : \\
\qquad\qquad\qquad\quad \lor\, SendEpochPromise(acceptor,\ round) \\
\qquad\qquad\qquad\quad \lor \exists\, Epoch \in Epochs : SendAcceptedMessage(acceptor,\ Epoch,\ round) \\
\qquad\qquad\qquad \lor \exists\, Epoch \in Epochs,\ value \in ProposableValues : \\
\qquad\qquad\qquad\quad \lor\, SendAcceptRequest(Epoch,\ round,\ value) \\
\qquad\qquad\qquad\quad \lor\, Learn(Epoch,\ round,\ value) \\
\qquad\qquad \lor \exists\, acceptor \in Acceptors,\ Epoch \in Epochs : \\
\qquad\qquad\qquad \lor\, Restart(acceptor,\ Epoch) \\
\qquad\qquad\qquad\quad \lor\, BeginNewEpoch(Epoch) \\
\qquad\qquad\qquad \lor\, RecieveEpochNotification(acceptor,\ Epoch)
\end{array}
$$

$$
EpochPaxosSpecification \triangleq InitalState \lor \Box[NextState]_{\langle acceptor\_states,\, sent\_messages,\, learnt \rangle}
$$

$$
\begin{array}{l}
ConsensusOK \triangleq \ \land\, EpochPaxosTypeInvariant \\
\qquad\qquad\qquad\quad \land\ \lor\, learnt = \{\} \\
\qquad\qquad\qquad\qquad\ \lor \exists\, value \in ProposableValues : learnt = \{value\}
\end{array}
$$

* Modification History
* Last modified *Fri* May 15 08:24:41 *BST* 2020 by *Michael*
* Created *Fri Aug* 09 16:48:26 *BST* 2019 by *Michael*