MODULE *PODValidator*

This module defines the behavior of each validator.

CONSTANT *Validator*
VARIABLE *Self*

VARIABLES
********** These are the variables in other modules **********
trans_buffer, messages need to transfer
recv_buffer, received messages
block, all blocks for all validators
status, status of current validator, will be reset at the beginning of each period
UsedIds, all used *IDs*
block_prepares,
block_commits

VARIABLES
contributed_block
All blocks that *Self* contributed, *i.e.*, proposed, prepared or committed.
The reason we keep this is to make sure we won't contributed on different
branches. *Note: this can be violited if *Self* is evil.*

$BlockChain \triangleq$ INSTANCE *LocalBlockChain* WITH *Validator* $\leftarrow$ *Validator*,
Self $\leftarrow$ *Self*,
block $\leftarrow$ *block*,
UsedIds $\leftarrow$ *UsedIds*,
block_prepares $\leftarrow$ *block_prepares*,
block_commits $\leftarrow$ *block_commits*

$Message \triangleq$

The set of all possible messages. The ins field indicates the sender. For "propose"
message, the "*val*" field means she propose a block. Since we do not mind the proposed value, we do not
record the proposed value here. The "sender" field indicates the sender of a message.
$[type : \{\text{"propose"}, \text{"prepare"}, \text{"commit"}\}, val : BlockChain!Block, sender : Validator]$

$Network \triangleq$ INSTANCE *Network* WITH *Message* $\leftarrow$ *Message*, *endpoint* $\leftarrow$ *Validator*,
trans_buffer $\leftarrow$ *trans_buffer*,
recv_buffer $\leftarrow$ *recv_buffer*

$$PVTypeOK \triangleq \land Network!NetworkTypeOK \land \forall n \in Validator : BlockChain!BCTypeOK \land status \in \{\text{"working"}, \text{"prepared"}, \text{"committed"}, \text{"finality"}\}$$

$$
\begin{aligned}
PVInit \triangleq\ & \land Network!NetworkInit \\
& \land \forall n \in Validator : BlockChain!BCInit \\
& \land status = \text{“working”}
\end{aligned}
$$

$$
PVPeriodInit \triangleq \land status' = \text{“working”}
$$

*This is for *init* at the begining of each period

Status checking functions

$IsImpossibleToReachCommitStatus \triangleq 0$ todo

This means there are two blocks, a and b, neither of them can get more the 2/3 prepares or commit.

$AlreadyReachFinalityStatus \triangleq 0$ todo

The *Self* status already reach finality.

---

network message handler

$$
\begin{aligned}
PVProposeBlock \triangleq\ & \text{LET } tail \triangleq \text{IF } contributed\_block \neq \{\} \\
& \quad \text{THEN } BlockChain!TailBlock[\text{CHOOSE } v : v \in contributed\_block] \\
& \quad \text{ELSE CHOOSE } v : v \in BlockChain!AllTails \\
& \text{IN LET } b \triangleq BlockChain!BCGenBlockWithTail(tail) \\
& \quad \text{IN } \land BlockChain!BCAddBlock(b) \\
& \qquad \land Network!Broadcast(Self,\ [type \mapsto \text{“propose”},\ val \mapsto b,\ sender \mapsto Self]) \\
& \qquad \land status = \text{“working”} \\
& \qquad \land status' = \text{“prepared”} \\
& \qquad \land UsedIds' = UsedIds \cup \{b.id\} \\
& \qquad \land contributed\_block' = contributed\_block \cup \{b\} \\
& \qquad \land BlockChain!BCPrepareBlock(b,\ Self)
\end{aligned}
$$

$$
\begin{aligned}
PVHandleProposeMsg(v) \triangleq\ & \land v.type = \text{“propose”} \\
& \land status = \text{“working”} \\
& \land status' = \text{“prepared”} \\
& \land contributed\_block' = contributed\_block \cup \{v.val\} \\
& \land BlockChain!BCAddBlock(v.val) \\
& \land Network!Broadcast(Self,\ [type \mapsto \text{“prepare”},\ val \mapsto v.val,\ sender \mapsto Self]) \\
& \land BlockChain!BCPrepareBlock(v.val,\ Self)
\end{aligned}
$$

$$
\begin{aligned}
PVHandlePrepareMsg(v) \triangleq\ & \land v.type = \text{“prepare”} \\
& \land \lor \land status = \text{“working”} \\
& \qquad \land status' = \text{“prepared”} \\
& \qquad \land contributed\_block' = contributed\_block \cup \{v.val\} \\
& \qquad \land BlockChain!BCAddBlock(v.val) \\
& \qquad \land Network!Broadcast(Self,\ [type \mapsto \text{“prepare”},\ val \mapsto v.val,\ sender \mapsto Se \\
& \qquad \land BlockChain!BCPrepareBlock(v.val,\ Self)
\end{aligned}
$$

$$\lor \land status \in \{\text{“prepared”}, \text{“committed”}, \text{“finality”}\}$$
$$\land BlockChain!BCAddBlock(v.val)$$
$$\land BlockChain!BCPrepareBlock(v.val, Self)$$

$$PVHandleCommitMsg(v) \triangleq \land v.type = \text{“commit”}$$
$$\land \lor \land status = \text{“prepared”}$$

$$PVHandleRecvMsg(v) \triangleq \lor PVHandleProposeMsg(v)$$
$$\lor PVHandlePrepareMsg(v)$$
$$\lor PVHandleCommitMsg(v)$$

Some action for local block chain. This can be one of the following actions:

1. choose some block to prepare
2. choose some block to commit
3. choose some block to remove, this is because the node finds some inconsistency and the node wants make it consistency for maximum benifits.
4. change block status if some block becomes final. This should be optional.

$$PVChooseToPrepare \triangleq 0 \text{ todo}$$
$$PVChooseToCommit \triangleq 0 \text{ todo}$$
$$PVChooseToRemoveBlocks \triangleq 0 \text{ todo}$$
$$PVChooseToChangeBlockStatus \triangleq 0 \text{ todo}$$

$$PVChooseAction \triangleq \lor PVChooseToPrepare$$
$$\lor PVChooseToCommit$$
$$\lor PVChooseToRemoveBlocks$$
$$\lor PVChooseToChangeBlockStatus$$

Here for the next steps, we don't do *PVProposeBlock* because we wanna leave this action to *PoD*, and *PoD* will decide which *Validator* to propose.

$$PVNext \triangleq \lor \exists msg \in Network!RecvedMsgs(Self) :$$
$$\land PVHandleRecvMsg(msg)$$
$$\land Network!RemoveMsg(Self, msg)$$
$$\lor PVChooseAction$$

$$PVConsistency \triangleq BlockChain!BCConsistency$$

---

* Modification History
* Last modified Sat *Feb* 03 19:35:16 *CST* 2018 by *xuepeng*
* Created Sat *Feb* 03 15:40:11 *CST* 2018 by *xuepeng*