────── MODULE $p2p$ ──────

This module defines a simple peer-to-peer network protocol that allows peers to connect, exchange blocks, and synchronize their chains.

EXTENDS $TLC$, $Sequences$, $Naturals$, $FiniteSets$, $Utils$, $Blockchain$

Define the network to be used by the algorithm.

CONSTANT $RunningBlockchain$

Maximum number of blocks to be retrieved in a single *getblocks* response.

CONSTANT $MaxGetBlocksInvResponse$

Maximum number of outbound connections a peer can have.

CONSTANT $MaxConnectionsPerPeer$

Difference in the $SYNCHRONIZER$ process id so that it does not conflict with the $LISTENER$ one.

$PeerProcessDiffId \triangleq 1000$

─────────────────────────────

 **--algorithm** $p2p$

**variables**

Represent the whole universe of peers in the network with all of their data.

$the\_network = RunningBlockchain$ ;

Each peer has a channel to communicate with other peers. Number of connections is limited.

$channels = [i \in 1 .. Len(the\_network) \mapsto$
$\quad [j \in 1 .. MaxConnectionsPerPeer \mapsto [$
$\quad\quad header \mapsto defaultInitValue,$
$\quad\quad payload \mapsto defaultInitValue$
$\quad ]]$
$] ;$

**define**

Import the operators used in the algorithm.

LOCAL $Ops \triangleq$ INSTANCE $Operators$

This property checks for the existence of at least one execution path in which all peers eventually have the same chain tip. It ensures that there is a scenario in which synchronization occurs, but does NOT guarantee that synchronization will happen in every possible execution. This makes it an existential check, not a liveness property.

$ExistsSyncPath \triangleq$
$\quad \exists\, peer1,\ peer2 \in 1 .. Len(RunningBlockchain) :$
$\quad\quad \Diamond(Ops!GetPeerTipByIndex(peer1).height = Ops!GetPeerTipByIndex(peer2).height)$

Liveness: Eventually, all peers will have the same chain tip. This property ensures that synchronization will happen in every possible path.

Note: This property is not guaranteed to hold in the current implementation.

1

$Liveness \;\triangleq$
 $\forall\, peer1,\; peer2 \in 1 \ldots Len(RunningBlockchain) :$
  $\Diamond(Ops\,!\,GetPeerTipByIndex(peer1).height = Ops\,!\,GetPeerTipByIndex(peer2).height)$

Ensures that no peer in the network has a chain tip that is higher than the chain tip of any peer it is connected to. This guarantees that peers do not "advance" their chain beyond the knowledge of their connected peers, ensuring consistent progress across the network.

$ChainTipRespectsPeerSet \;\triangleq$
 $\forall\, peer \in 1 \ldots Len(RunningBlockchain) :$
  $\forall\, remote\_peer \in 1 \ldots Len(the\_network[peer].peer\_set) :$
   $Ops\,!\,GetPeerTipByIndex(peer).height \leq Ops\,!\,GetPeerTipByIndex(remote\_peer).height$

Ensures that each block in a peer's local blockchain has a height less than or equal to the peer's chain tip. This prevents peers from including invalid blocks that exceed the current chain tip.

$ValidBlockPropagation \;\triangleq$
 $\forall\, peer \in 1 \ldots Len(RunningBlockchain) :$
  $\forall\, block \in the\_network[peer].blocks :$
   $block.height \leq Ops\,!\,GetPeerTipByIndex(peer).height$

Ensures that the blocks within each peer's blockchain are correctly ordered by height. For blocks with height greater than 1, each block must directly follow the block with height $block.height - 1$. This prevents gaps or misordering within a peer's chain.

$BlockOrdering \;\triangleq$
 $\forall\, peer \in 1 \ldots Len(RunningBlockchain) :$
  $\forall\, block \in the\_network[peer].blocks :$
   IF $block.height < 2$ THEN
    TRUE
   ELSE
    $block.height =$
     $(\text{CHOOSE } b \in the\_network[peer].blocks : b.height = block.height - 1).height + 1$

Ensures that each peer eventually reaches a chain tip that is at least as high as the initial chain tip it started with. This ensures that peers make progress in synchronizing their chains over time.

$SyncProgress \;\triangleq$
 $\forall\, peer \in 1 \ldots Len(RunningBlockchain) :$
  $\Diamond(Ops\,!\,GetPeerTipByIndex(peer).height \geq$
   $Ops\,!\,GetPeerTipByIndexAndNetwork(peer,\; RunningBlockchain).height)$

Ensures that no peer exceeds the maximum number of connections allowed, ensuring that the network respects its maximum connection constraints. This prevents any peer from overloading its connection capacity.

$ConnectionLimit \;\triangleq$
 $\forall\, peer \in 1 \ldots Len(RunningBlockchain) :$
  $Len(the\_network[peer].peer\_set) \leq MaxConnectionsPerPeer$

$ConsistentBlocksAcrossPeers \triangleq$
$\quad \forall\, peer1,\, peer2 \in 1\,..\, Len(RunningBlockchain):$
$\qquad peer1 \neq peer2 \Rightarrow$
$\qquad\quad \forall\, block1 \in the\_network[peer1].blocks,\, block2 \in the\_network[peer2].blocks:$
$\qquad\qquad block1.height = block2.height \Rightarrow$
$\qquad\qquad\quad block1.hash = block2.hash \wedge block1.block = block2.block$

$InductiveInvariant \triangleq$
$\quad \wedge\ ChainTipRespectsPeerSet$
$\quad \wedge\ ValidBlockPropagation$
$\quad \wedge\ BlockOrdering$
$\quad \wedge\ SyncProgress$
$\quad \wedge\ ConnectionLimit$
$\quad \wedge\ ConsistentBlocksAcrossPeers$

**end define ;**

**procedure** $announce(local\_peer\_id,\ remote\_peer\_id)$
**begin**
$\quad SendAddrMsg:$
$\qquad channels[local\_peer\_id][remote\_peer\_id] := [$
$\qquad\quad header \mapsto [command\_name \mapsto \text{"addr"}],$
$\qquad\quad payload \mapsto [$
$\qquad\qquad address\_count \mapsto 1,$

$\qquad\qquad addresses \mapsto the\_network[local\_peer\_id].peer$
$\qquad\quad ]$
$\qquad ];$
$\quad$ **return ;**
**end procedure ;**

**procedure** $addr(local\_peer\_id,\ remote\_peer\_id)$
**begin**
$\quad SendVersionMsg:$
$\qquad channels[local\_peer\_id][remote\_peer\_id] := [$

$header \mapsto [command\_name \mapsto \text{``version''}],$
$payload \mapsto [$
 $addr\_recv \mapsto the\_network[local\_peer\_id].peer,$
 $addr\_trans \mapsto the\_network[local\_peer\_id].peer\_set[remote\_peer\_id].address,$
 $start\_height \mapsto$
  $Ops!\,GetPeerTipByAddress(the\_network[local\_peer\_id].peer\_set[remote\_peer\_id].address).h$
$]\,;$
**return** ;
**end procedure** ;

Given a version message is received, send verack to acknowledge the connection.

*Given a version message is received, send verack to acknowledge the connection.*

**procedure** $version(local\_peer\_id,\ remote\_peer\_id)$
**begin**
 $HandleVersionMsg:$
  $the\_network[local\_peer\_id].peer\_set[remote\_peer\_id].tip :=$
   $channels[local\_peer\_id][remote\_peer\_id].payload.start\_height\,;$
 $SendVerackMsg:$
  $channels[local\_peer\_id][remote\_peer\_id] := [$
   $header \mapsto [command\_name \mapsto \text{``verack''}],$
   $payload \mapsto defaultInitValue$
  $]\,;$
 **return** ;
**end procedure** ;

*Given a verack message is received, establish the connection.*

**procedure** $verack(local\_peer\_id,\ remote\_peer\_id)$
**begin**
 $HandleVerackMsg:$
  $the\_network[local\_peer\_id].peer\_set[remote\_peer\_id].established := \text{TRUE}\,;$
 **return** ;
**end procedure** ;

*Given a getblocks message is received, send an inv message with the blocks requested.*

**procedure** $getblocks(local\_peer\_id,\ remote\_peer\_id)$
**variables**
 $found\_blocks,\ hash\_count,\ block\_header\_hashes,\ remote\_peer\_blocks,\ start\_height,\ end\_height\,;$
**begin**
 $HandleGetBlocksMsg:$
  *Retrieve necessary values from the channel payload*
  $hash\_count := channels[local\_peer\_id][remote\_peer\_id].payload.hash\_count\,;$
  $block\_header\_hashes := channels[local\_peer\_id][remote\_peer\_id].payload.block\_header\_hashes\,;$

  *Fetch the blocks of the remote peer*
  $remote\_peer\_blocks :=$
   $Ops!\,GetPeerBlocks(the\_network[local\_peer\_id].peer\_set[remote\_peer\_id].address)\,;$

        *Determine the range of blocks to retrieve*
      **if** $hash\_count = 0$ **then**
          $start\_height := 1$ ;
       **else**
            *Assuming the hashes are in order, the height of the first hash should be the tip, ignore the rest.*
          $start\_height :=$
             $Ops!FindBlockByHash(remote\_peer\_blocks,\ block\_header\_hashes[1]).height + 1$ ;
      **end if** ;
      $end\_height := start\_height + (MaxGetBlocksInvResponse - 1)$ ;

      *Find the blocks within the specified range.*
      $found\_blocks := Ops!FindBlocks(remote\_peer\_blocks,\ start\_height,\ end\_height)$ ;
    $SendInvMsg\colon$
      $channels[local\_peer\_id][remote\_peer\_id] := [$
          $header \mapsto [command\_name \mapsto \text{“inv”}],$
          $payload \mapsto [$
             $count \mapsto Cardinality(found\_blocks),$
             $inventory \mapsto [$
                $h \in 1\mathinner{\ldotp\ldotp} Cardinality(found\_blocks) \mapsto [$
                    $type\_identifier \mapsto \text{“MSG\_BLOCK”},$
                    $hash \mapsto SetToSeq(\{s.hash : s \in found\_blocks\})[h]$
                $]$
             $]$
          $]$
      $]$ ;
    **return** ;
**end procedure** ;

*Request blocks from the remote peer by sending a getblocks message with local hashes.*
**procedure** $request\_blocks(hashes,\ local\_peer\_id,\ remote\_peer\_id)$
**begin**
    $SendGetBlocksMsg\colon$
      $channels[local\_peer\_id][remote\_peer\_id] := [$
          $header \mapsto [command\_name \mapsto \text{“getblocks”}],$
          $payload \mapsto [$
             $hash\_count \mapsto Len(hashes),$
             $block\_header\_hashes \mapsto hashes]$
      $]$ ;
    **return** ;
**end procedure** ;

*Given an inv message is received, send a getdata message to request the blocks.*
**procedure** $inv(local\_peer\_id,\ remote\_peer\_id)$
**begin**
    $SendGetDataMsg\colon$
      $channels[local\_peer\_id][remote\_peer\_id] := [$

$$header \mapsto [command\_name \mapsto \text{``getdata''}],$$
$$payload \mapsto channels[local\_peer\_id][remote\_peer\_id].payload$$
$$\hspace{2cm}];$$
    **return** ;
**end procedure** ;

*Incorporate data to the local peer from the inventory received.*
**procedure** *getdata*(*local_peer_id*, *remote_peer_id*)
**variables** *blocks_data* ;
**begin**
    *HandleGetDataMsg*:
        $blocks\_data := [item \in 1 .. Len(channels[local\_peer\_id][remote\_peer\_id].payload.inventory) \mapsto$
            $Ops\,!\,FindBlockByHash($
                $Ops\,!\,GetPeerBlocks(the\_network[local\_peer\_id].peer\_set[remote\_peer\_id].address),$
                $channels[local\_peer\_id][remote\_peer\_id].payload.inventory[item].hash$
            $)$
        $];$
    *IncorporateBlocks*:
        $the\_network[local\_peer\_id].blocks := the\_network[local\_peer\_id].blocks \cup ToSet(blocks\_data)$ ;
    **return** ;
**end procedure** ;

*A set of listener process for each peer to listen to incoming messages and act accordingly.*
**process** $LISTENER \in 1 .. Len(RunningBlockchain)$
**variables** *command* ;
**begin**
    *Listening*:
        **await** $Len(the\_network) \geq 2$ ;
        **with** $remote\_peer\_index \in 1 .. Len(the\_network[self].peer\_set)$ **do**
            **if** $channels[self][remote\_peer\_index].header = defaultInitValue$ **then**
                **goto** *Listening* ;
            **end if** ;
        **end with** ;
    *Requests*:
        **with** $remote\_peer\_index \in 1 .. Len(the\_network[self].peer\_set)$ **do**
            **await** $channels[self][remote\_peer\_index].header \neq defaultInitValue$ ;
            $command := channels[self][remote\_peer\_index].header.command\_name$ ;
            **if** $command = \text{``addr''}$ **then**
                **call** $addr(self, remote\_peer\_index)$ ;
                **goto** *Listening* ;
            **elsif** $command = \text{``version''}$ **then**
                **call** $version(self, remote\_peer\_index)$ ;
                **goto** *Listening* ;
            **elsif** $command = \text{``verack''}$ **then**
                **call** $verack(self, remote\_peer\_index)$ ;

```
            elsif command = "getblocks" then
                call getblocks(self, remote_peer_index);
                goto Listening;
            elsif command = "inv" then
                call inv(self, remote_peer_index);
                goto Listening;
            elsif command = "getdata" then
                call getdata(self, remote_peer_index);
            end if;
        end with;
    ListenerLoop:
        with remote_peer_index ∈ 1 .. Len(the_network[self].peer_set) do
            channels[self][remote_peer_index] :=
                [header ↦ defaultInitValue, payload ↦ defaultInitValue];
            goto Listening;
        end with;
end process;
```

**process** $SYNCHRONIZER \in PeerProcessDiffId + 1 .. PeerProcessDiffId + Len(RunningBlockchain)$
**variables** $local\_peer\_index = self - PeerProcessDiffId$, $best\_tip = 0$;
**begin**
    *Announce*:

        The network must have at least two peer.
        **assert** $Len(the\_network) \geq 2$;

        The peer set size must be at least 1, ignoring the peers that are seeders only.
        **await** $Len(the\_network[local\_peer\_index].peer\_set) > 0$;

        Connect to each available peer we have.
        **with** $remote\_peer\_index \in 1 .. Len(the\_network[local\_peer\_index].peer\_set)$ **do**
            **call** $announce(local\_peer\_index, remote\_peer\_index)$;
        **end with**;
    *RequestInventory*:
        **with** $remote\_peer\_index \in 1 .. Len(the\_network[local\_peer\_index].peer\_set)$ **do**
            Make sure the connection is established before requesting any block from this peer.
            **await** $the\_network[local\_peer\_index].peer\_set[remote\_peer\_index].established = \text{TRUE}$;

            Find the best tip among all peers.
            **if** $the\_network[local\_peer\_index].peer\_set[remote\_peer\_index].tip > best\_tip$ **then**
                $best\_tip := the\_network[local\_peer\_index].peer\_set[remote\_peer\_index].tip$;
            **end if**;

            Wait for the peer channel to be empty before requesting new blocks.
            **await** $channels[local\_peer\_index][remote\_peer\_index].header = defaultInitValue$
                $\wedge\ channels[local\_peer\_index][remote\_peer\_index].payload = defaultInitValue$;

*Check if the local peer is behind the remote peer.*
**if** $Ops!GetPeerTipByIndex(local\_peer\_index).height <$
    $the\_network[local\_peer\_index].peer\_set[remote\_peer\_index].tip$ **then**
    *Request blocks.*
    **if** $Ops!GetPeerTipByIndex(local\_peer\_index).height = 0$ **then**
        **call** $request\_blocks(\langle\rangle,\ local\_peer\_index,\ remote\_peer\_index)$**;**
    **else**
        **call** $request\_blocks($
            $\langle Ops!GetPeerTipByIndex(local\_peer\_index).hash\rangle,$
            $local\_peer\_index,$
            $remote\_peer\_index$
        $)$**;**
    **end if** ;
**end if** ;
**end with** ;
*CheckSync*:
    **await** $Ops!GetPeerTipByIndex(local\_peer\_index).height > 0$**;**
    **if** $Ops!GetPeerTipByIndex(local\_peer\_index).height < best\_tip$ **then**
        **goto** *RequestInventory*;
    **else**
        *Make sure all connections are still established and all communication channels are empty*
        **with** $remote\_peer\_index \in 1 .. Len(the\_network[local\_peer\_index].peer\_set)$ **do**
            **await** $the\_network[local\_peer\_index].peer\_set[remote\_peer\_index].established = \textsc{true}$
                $\land\ channels[local\_peer\_index][remote\_peer\_index].header = defaultInitValue$
                $\land\ channels[local\_peer\_index][remote\_peer\_index].payload = defaultInitValue$**;**
        **end with** ;
    **end if** ;
**end process** ;

**end algorithm** ;

BEGIN TRANSLATION($chksum(pcal) =$ "fb1948d7" $\land\ chksum(tla) =$ "bceb3955")

*Parameter local_peer_id of procedure announce at line 155 col 20 changed to local_peer_id_*
*Parameter remote_peer_id of procedure announce at line 155 col 35 changed to remote_peer_id_*
*Parameter local_peer_id of procedure addr at line 170 col 16 changed to local_peer_id_a*
*Parameter remote_peer_id of procedure addr at line 170 col 31 changed to remote_peer_id_a*
*Parameter local_peer_id of procedure version at line 185 col 19 changed to local_peer_id_v*
*Parameter remote_peer_id of procedure version at line 185 col 34 changed to remote_peer_id_v*
*Parameter local_peer_id of procedure verack at line 199 col 18 changed to local_peer_id_ve*
*Parameter remote_peer_id of procedure verack at line 199 col 33 changed to remote_peer_id_ve*
*Parameter local_peer_id of procedure getblocks at line 207 col 21 changed to local_peer_id_g*
*Parameter remote_peer_id of procedure getblocks at line 207 col 36 changed to remote_peer_id_g*
*Parameter local_peer_id of procedure request_blocks at line 249 col 34 changed to local_peer_id_r*
*Parameter remote_peer_id of procedure request_blocks at line 249 col 49 changed to remote_peer_id_r*
*Parameter local_peer_id of procedure inv at line 262 col 15 changed to local_peer_id_i*
*Parameter remote_peer_id of procedure inv at line 262 col 30 changed to remote_peer_id_i*

CONSTANT *defaultInitValue*
VARIABLES *the_network*, *channels*, *pc*, *stack*

define statement
LOCAL $Ops \triangleq$ INSTANCE *Operators*

$ExistsSyncPath \triangleq$
    $\exists\, peer1,\ peer2 \in 1\mathinner{\ldotp\ldotp} Len(RunningBlockchain) :$
        $\Diamond(Ops\,!\,GetPeerTipByIndex(peer1).height = Ops\,!\,GetPeerTipByIndex(peer2).height)$

$Liveness \triangleq$
    $\forall\, peer1,\ peer2 \in 1\mathinner{\ldotp\ldotp} Len(RunningBlockchain) :$
        $\Diamond(Ops\,!\,GetPeerTipByIndex(peer1).height = Ops\,!\,GetPeerTipByIndex(peer2).height)$

$ChainTipRespectsPeerSet \triangleq$
    $\forall\, peer \in 1\mathinner{\ldotp\ldotp} Len(RunningBlockchain) :$
        $\forall\, remote\_peer \in 1\mathinner{\ldotp\ldotp} Len(the\_network[peer].peer\_set) :$
            $Ops\,!\,GetPeerTipByIndex(peer).height \leq Ops\,!\,GetPeerTipByIndex(remote\_peer).height$

$ValidBlockPropagation \triangleq$
    $\forall\, peer \in 1\mathinner{\ldotp\ldotp} Len(RunningBlockchain) :$
        $\forall\, block \in the\_network[peer].blocks :$
            $block.height \leq Ops\,!\,GetPeerTipByIndex(peer).height$

$BlockOrdering \triangleq$
  $\forall\, peer \in 1\,..\,Len(RunningBlockchain):$
    $\forall\, block \in the\_network[peer].blocks:$
      IF $block.height < 2$ THEN
        TRUE
      ELSE
        $block.height =$
          (CHOOSE $b \in the\_network[peer].blocks : b.height = block.height - 1).height + 1$

$SyncProgress \triangleq$
  $\forall\, peer \in 1\,..\,Len(RunningBlockchain):$
    $\diamond(Ops!GetPeerTipByIndex(peer).height \geq$
      $Ops!GetPeerTipByIndexAndNetwork(peer,\ RunningBlockchain).height)$

$ConnectionLimit \triangleq$
  $\forall\, peer \in 1\,..\,Len(RunningBlockchain):$
    $Len(the\_network[peer].peer\_set) \leq MaxConnectionsPerPeer$

$ConsistentBlocksAcrossPeers \triangleq$
  $\forall\, peer1,\ peer2 \in 1\,..\,Len(RunningBlockchain):$
    $peer1 \neq peer2 \Rightarrow$
      $\forall\, block1 \in the\_network[peer1].blocks,\ block2 \in the\_network[peer2].blocks:$
        $block1.height = block2.height \Rightarrow$
          $block1.hash = block2.hash \land block1.block = block2.block$

$InductiveInvariant \triangleq$
  $\land\ ChainTipRespectsPeerSet$

10

$\land$ *ValidBlockPropagation*
$\land$ *BlockOrdering*
$\land$ *SyncProgress*
$\land$ *ConnectionLimit*
$\land$ *ConsistentBlocksAcrossPeers*

VARIABLES $local\_peer\_id\_$, $remote\_peer\_id\_$, $local\_peer\_id\_a$, $remote\_peer\_id\_a$,
$\quad local\_peer\_id\_v$, $remote\_peer\_id\_v$, $local\_peer\_id\_ve$,
$\quad remote\_peer\_id\_ve$, $local\_peer\_id\_g$, $remote\_peer\_id\_g$, $found\_blocks$,
$\quad hash\_count$, $block\_header\_hashes$, $remote\_peer\_blocks$, $start\_height$,
$\quad end\_height$, $hashes$, $local\_peer\_id\_r$, $remote\_peer\_id\_r$,
$\quad local\_peer\_id\_i$, $remote\_peer\_id\_i$, $local\_peer\_id$, $remote\_peer\_id$,
$\quad blocks\_data$, $command$, $local\_peer\_index$, $best\_tip$

$vars \triangleq \langle the\_network, channels, pc, stack, local\_peer\_id\_, remote\_peer\_id\_,$
$\quad local\_peer\_id\_a, remote\_peer\_id\_a, local\_peer\_id\_v,$
$\quad remote\_peer\_id\_v, local\_peer\_id\_ve, remote\_peer\_id\_ve,$
$\quad local\_peer\_id\_g, remote\_peer\_id\_g, found\_blocks, hash\_count,$
$\quad block\_header\_hashes, remote\_peer\_blocks, start\_height, end\_height,$
$\quad hashes, local\_peer\_id\_r, remote\_peer\_id\_r, local\_peer\_id\_i,$
$\quad remote\_peer\_id\_i, local\_peer\_id, remote\_peer\_id, blocks\_data,$
$\quad command, local\_peer\_index, best\_tip \rangle$

$ProcSet \triangleq (1 .. Len(RunningBlockchain)) \cup (PeerProcessDiffId + 1 .. PeerProcessDiffId + Len(RunningBlock$

$Init \triangleq$   Global variables
$\quad \land the\_network = RunningBlockchain$
$\quad \land channels = \quad\quad\quad [i \in 1 .. Len(the\_network) \mapsto$
$\quad\quad\quad\quad [j \in 1 .. MaxConnectionsPerPeer \mapsto [$
$\quad\quad\quad\quad\quad header \mapsto defaultInitValue,$
$\quad\quad\quad\quad\quad payload \mapsto defaultInitValue$
$\quad\quad\quad\quad ]]$
$\quad\quad\quad ]$
  Procedure announce
$\quad \land local\_peer\_id\_ = [self \in ProcSet \mapsto defaultInitValue]$
$\quad \land remote\_peer\_id\_ = [self \in ProcSet \mapsto defaultInitValue]$
  Procedure addr
$\quad \land local\_peer\_id\_a = [self \in ProcSet \mapsto defaultInitValue]$
$\quad \land remote\_peer\_id\_a = [self \in ProcSet \mapsto defaultInitValue]$
  Procedure version
$\quad \land local\_peer\_id\_v = [self \in ProcSet \mapsto defaultInitValue]$
$\quad \land remote\_peer\_id\_v = [self \in ProcSet \mapsto defaultInitValue]$
  Procedure verack
$\quad \land local\_peer\_id\_ve = [self \in ProcSet \mapsto defaultInitValue]$
$\quad \land remote\_peer\_id\_ve = [self \in ProcSet \mapsto defaultInitValue]$
  Procedure getblocks

11

$\land \mathit{local\_peer\_id\_g} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{remote\_peer\_id\_g} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{found\_blocks} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{hash\_count} = [\mathit{self} \quad \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{block\_header\_hashes} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{remote\_peer\_blocks} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{start\_height} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{end\_height} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$

*Procedure request_blocks*

$\land \mathit{hashes} = [\mathit{self} \quad \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{local\_peer\_id\_r} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{remote\_peer\_id\_r} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$

*Procedure inv*

$\land \mathit{local\_peer\_id\_i} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{remote\_peer\_id\_i} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$

*Procedure getdata*

$\land \mathit{local\_peer\_id} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{remote\_peer\_id} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$
$\land \mathit{blocks\_data} = [\mathit{self} \in \mathit{ProcSet} \mapsto \mathit{defaultInitValue}]$

*Process LISTENER*

$\land \mathit{command} = [\mathit{self} \in 1 \mathinner{..} \mathit{Len}(\mathit{RunningBlockchain}) \mapsto \mathit{defaultInitValue}]$

*Process SYNCHRONIZER*

$\land \mathit{local\_peer\_index} = [\mathit{self} \in \mathit{PeerProcessDiffId} + 1 \mathinner{..} \mathit{PeerProcessDiffId} + \mathit{Len}(\mathit{RunningBlockchain}) \mapsto$
$\land \mathit{best\_tip} = [\mathit{self} \quad \in \mathit{PeerProcessDiffId} + 1 \mathinner{..} \mathit{PeerProcessDiffId} + \mathit{Len}(\mathit{RunningBlockchain}) \mapsto 0]$
$\land \mathit{stack} = [\mathit{self} \in \mathit{ProcSet} \mapsto \langle \rangle]$
$\land \mathit{pc} = [\mathit{self} \in \mathit{ProcSet} \mapsto \text{CASE } \mathit{self} \in 1 \mathinner{..} \mathit{Len}(\mathit{RunningBlockchain}) \rightarrow \text{“Listening”}$
$\qquad\qquad\qquad\qquad\qquad \Box \quad \mathit{self} \in \mathit{PeerProcessDiffId} + 1 \mathinner{..} \mathit{PeerProcessDiffId} + \mathit{Len}(\mathit{RunningBlo}$

$\mathit{SendAddrMsg}(\mathit{self}) \;\triangleq\; \land \mathit{pc}[\mathit{self}] = \text{“SendAddrMsg”}$
$\qquad\qquad\qquad\qquad \land \mathit{channels}' = [\mathit{channels} \text{ EXCEPT } ![\mathit{local\_peer\_id\_}[\mathit{self}]][\mathit{remote\_peer\_id\_}[\mathit{self}]] =$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad he$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pa$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ]$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ]]$
$\qquad\qquad\qquad\qquad \land \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{pc}]$
$\qquad\qquad\qquad\qquad \land \mathit{local\_peer\_id\_}' = [\mathit{local\_peer\_id\_} \text{ EXCEPT } ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{local\_peer\_id\_}]$
$\qquad\qquad\qquad\qquad \land \mathit{remote\_peer\_id\_}' = [\mathit{remote\_peer\_id\_} \text{ EXCEPT } ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{remote\_pe}$
$\qquad\qquad\qquad\qquad \land \mathit{stack}' = [\mathit{stack} \text{ EXCEPT } ![\mathit{self}] = \mathit{Tail}(\mathit{stack}[\mathit{self}])]$
$\qquad\qquad\qquad\qquad \land \text{UNCHANGED } \langle \mathit{the\_network},\ \mathit{local\_peer\_id\_a},$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{remote\_peer\_id\_a},\ \mathit{local\_peer\_id\_v},$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{remote\_peer\_id\_v},\ \mathit{local\_peer\_id\_ve},$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{remote\_peer\_id\_ve},\ \mathit{local\_peer\_id\_g},$

$$
\begin{array}{l}
remote\_peer\_id\_g,\; found\_blocks, \\
hash\_count,\; block\_header\_hashes, \\
remote\_peer\_blocks,\; start\_height, \\
end\_height,\; hashes,\; local\_peer\_id\_r, \\
remote\_peer\_id\_r,\; local\_peer\_id\_i, \\
remote\_peer\_id\_i,\; local\_peer\_id, \\
remote\_peer\_id,\; blocks\_data,\; command, \\
local\_peer\_index,\; best\_tip\rangle
\end{array}
$$

$announce(self) \;\triangleq\; SendAddrMsg(self)$

$SendVersionMsg(self) \;\triangleq\; \wedge\, pc[self] = \text{``SendVersionMsg''}$
$\qquad\qquad\qquad\qquad\quad \wedge\, channels' = [channels \text{ EXCEPT } ![local\_peer\_id\_a[self]][remote\_peer\_id\_a[self]] =$

$$]$$

$\qquad\qquad\qquad\qquad\quad \wedge\, pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$
$\qquad\qquad\qquad\qquad\quad \wedge\, local\_peer\_id\_a' = [local\_peer\_id\_a \text{ EXCEPT } ![self] = Head(stack[self]).local\_peer$
$\qquad\qquad\qquad\qquad\quad \wedge\, remote\_peer\_id\_a' = [remote\_peer\_id\_a \text{ EXCEPT } ![self] = Head(stack[self]).remo$
$\qquad\qquad\qquad\qquad\quad \wedge\, stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])]$
$\qquad\qquad\qquad\qquad\quad \wedge\, \text{UNCHANGED } \langle the\_network,\; local\_peer\_id\_,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_,\; local\_peer\_id\_v,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_v,\; local\_peer\_id\_ve,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_ve,\; local\_peer\_id\_g,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_g,\; found\_blocks,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad hash\_count,\; block\_header\_hashes,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_blocks,\; start\_height,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad end\_height,\; hashes,\; local\_peer\_id\_r,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_r,\; local\_peer\_id\_i,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_i,\; local\_peer\_id,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id,\; blocks\_data,\; command,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad local\_peer\_index,\; best\_tip\rangle$

$addr(self) \;\triangleq\; SendVersionMsg(self)$

$HandleVersionMsg(self) \;\triangleq\; \wedge\, pc[self] = \text{``HandleVersionMsg''}$
$\qquad\qquad\qquad\qquad\qquad\quad \wedge\, the\_network' = [the\_network \text{ EXCEPT } ![local\_peer\_id\_v[self]].peer\_set[remote\_$
$\qquad\qquad\qquad\qquad\qquad\quad \wedge\, pc' = [pc \text{ EXCEPT } ![self] = \text{``SendVerackMsg''}]$
$\qquad\qquad\qquad\qquad\qquad\quad \wedge\, \text{UNCHANGED } \langle channels,\; stack,\; local\_peer\_id\_,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_,\; local\_peer\_id\_a,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_a,\; local\_peer\_id\_v,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad remote\_peer\_id\_v,\; local\_peer\_id\_ve,$

13

$$\begin{aligned}
& remote\_peer\_id\_ve,\ local\_peer\_id\_g, \\
& remote\_peer\_id\_g,\ found\_blocks, \\
& hash\_count,\ block\_header\_hashes, \\
& remote\_peer\_blocks,\ start\_height, \\
& end\_height,\ hashes,\ local\_peer\_id\_r, \\
& remote\_peer\_id\_r,\ local\_peer\_id\_i, \\
& remote\_peer\_id\_i,\ local\_peer\_id, \\
& remote\_peer\_id,\ blocks\_data,\ command, \\
& local\_peer\_index,\ best\_tip\rangle
\end{aligned}$$

$SendVerackMsg(self) \triangleq\ \wedge\ pc[self] = \text{``SendVerackMsg''}$
$\qquad\qquad\qquad\qquad\ \ \wedge\ channels' = [channels \text{ EXCEPT } ![local\_peer\_id\_v[self]][remote\_peer\_id\_v[self]] =$

$$]]$$

$\qquad\qquad\qquad\quad \wedge\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$
$\qquad\qquad\qquad\quad \wedge\ local\_peer\_id\_v' = [local\_peer\_id\_v \text{ EXCEPT } ![self] = Head(stack[self]).local\_peer$
$\qquad\qquad\qquad\quad \wedge\ remote\_peer\_id\_v' = [remote\_peer\_id\_v \text{ EXCEPT } ![self] = Head(stack[self]).remo$
$\qquad\qquad\qquad\quad \wedge\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])]$
$\qquad\qquad\qquad\quad \wedge\ \text{UNCHANGED } \langle the\_network,\ local\_peer\_id\_,$

$$\begin{aligned}
& remote\_peer\_id\_,\ local\_peer\_id\_a, \\
& remote\_peer\_id\_a,\ local\_peer\_id\_ve, \\
& remote\_peer\_id\_ve,\ local\_peer\_id\_g, \\
& remote\_peer\_id\_g,\ found\_blocks, \\
& hash\_count,\ block\_header\_hashes, \\
& remote\_peer\_blocks,\ start\_height, \\
& end\_height,\ hashes,\ local\_peer\_id\_r, \\
& remote\_peer\_id\_r,\ local\_peer\_id\_i, \\
& remote\_peer\_id\_i,\ local\_peer\_id, \\
& remote\_peer\_id,\ blocks\_data,\ command, \\
& local\_peer\_index,\ best\_tip\rangle
\end{aligned}$$

$version(self) \triangleq\ HandleVersionMsg(self) \vee SendVerackMsg(self)$

$HandleVerackMsg(self) \triangleq\ \wedge\ pc[self] = \text{``HandleVerackMsg''}$
$\qquad\qquad\qquad\qquad\ \ \ \wedge\ the\_network' = [the\_network \text{ EXCEPT } ![local\_peer\_id\_ve[self]].peer\_set[remote\_$
$\qquad\qquad\qquad\qquad\ \ \ \wedge\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$
$\qquad\qquad\qquad\qquad\ \ \ \wedge\ local\_peer\_id\_ve' = [local\_peer\_id\_ve \text{ EXCEPT } ![self] = Head(stack[self]).local\_$
$\qquad\qquad\qquad\qquad\ \ \ \wedge\ remote\_peer\_id\_ve' = [remote\_peer\_id\_ve \text{ EXCEPT } ![self] = Head(stack[self]).r$
$\qquad\qquad\qquad\qquad\ \ \ \wedge\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])]$
$\qquad\qquad\qquad\qquad\ \ \ \wedge\ \text{UNCHANGED } \langle channels,\ local\_peer\_id\_,$

$$\begin{aligned}
& remote\_peer\_id\_,\ local\_peer\_id\_a, \\
& remote\_peer\_id\_a,\ local\_peer\_id\_v, \\
& remote\_peer\_id\_v,\ local\_peer\_id\_g, \\
& remote\_peer\_id\_g,\ found\_blocks, \\
& hash\_count,\ block\_header\_hashes,
\end{aligned}$$

14

$$\begin{array}{l} remote\_peer\_blocks,\ start\_height, \\ end\_height,\ hashes,\ local\_peer\_id\_r, \\ remote\_peer\_id\_r,\ local\_peer\_id\_i, \\ remote\_peer\_id\_i,\ local\_peer\_id, \\ remote\_peer\_id,\ blocks\_data,\ command, \\ local\_peer\_index,\ best\_tip\rangle \end{array}$$

$verack(self) \triangleq HandleVerackMsg(self)$

$HandleGetBlocksMsg(self) \triangleq \land pc[self] = \text{``HandleGetBlocksMsg''}$
$\qquad \land hash\_count' = [hash\_count \text{ EXCEPT } ![self] = channels[local\_peer\_id\_g[self]]]$
$\qquad \land block\_header\_hashes' = [block\_header\_hashes \text{ EXCEPT } ![self] = channels[loc$
$\qquad \land remote\_peer\_blocks' = [remote\_peer\_blocks \text{ EXCEPT } ![self] = Ops!GetPeerB$
$\qquad \land \text{IF } hash\_count'[self] = 0$
$\qquad\qquad \text{THEN } \land start\_height' = [start\_height \text{ EXCEPT } ![self] = 1]$
$\qquad\qquad \text{ELSE } \land start\_height' = [start\_height \text{ EXCEPT } ![self] = Ops!FindBlockB$
$\qquad \land end\_height' = [end\_height \text{ EXCEPT } ![self] = start\_height'[self] + (MaxGetBl$
$\qquad \land found\_blocks' = [found\_blocks \text{ EXCEPT } ![self] = Ops!FindBlocks(remote\_pe$
$\qquad \land pc' = [pc \text{ EXCEPT } ![self] = \text{``SendInvMsg''}]$
$\qquad \land \text{UNCHANGED } \langle the\_network,\ channels,\ stack,$
$\qquad\qquad\qquad local\_peer\_id\_,\ remote\_peer\_id\_,$
$\qquad\qquad\qquad local\_peer\_id\_a,\ remote\_peer\_id\_a,$
$\qquad\qquad\qquad local\_peer\_id\_v,\ remote\_peer\_id\_v,$
$\qquad\qquad\qquad local\_peer\_id\_ve,$
$\qquad\qquad\qquad remote\_peer\_id\_ve,\ local\_peer\_id\_g,$
$\qquad\qquad\qquad remote\_peer\_id\_g,\ hashes,$
$\qquad\qquad\qquad local\_peer\_id\_r,\ remote\_peer\_id\_r,$
$\qquad\qquad\qquad local\_peer\_id\_i,\ remote\_peer\_id\_i,$
$\qquad\qquad\qquad local\_peer\_id,\ remote\_peer\_id,$
$\qquad\qquad\qquad blocks\_data,\ command,$
$\qquad\qquad\qquad local\_peer\_index,\ best\_tip\rangle$

$SendInvMsg(self) \triangleq \land pc[self] = \text{``SendInvMsg''}$
$\qquad \land channels' = [channels \text{ EXCEPT } ![local\_peer\_id\_g[self]][remote\_peer\_id\_g[self]] =$

$$h$$
$$p$$

$$]$$
$$]]$$

$\qquad \land pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$

15

$$\land \textit{found\_blocks}' = [\textit{found\_blocks} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{found\_blocks}]$$
$$\land \textit{hash\_count}' = [\textit{hash\_count} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{hash\_count}]$$
$$\land \textit{block\_header\_hashes}' = [\textit{block\_header\_hashes} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{bloc}$$
$$\land \textit{remote\_peer\_blocks}' = [\textit{remote\_peer\_blocks} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{remot}$$
$$\land \textit{start\_height}' = [\textit{start\_height} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{start\_height}]$$
$$\land \textit{end\_height}' = [\textit{end\_height} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{end\_height}]$$
$$\land \textit{local\_peer\_id\_g}' = [\textit{local\_peer\_id\_g} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{local\_peer\_id\_}$$
$$\land \textit{remote\_peer\_id\_g}' = [\textit{remote\_peer\_id\_g} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{remote\_p}$$
$$\land \textit{stack}' = [\textit{stack} \text{ EXCEPT } ![\textit{self}] = \textit{Tail}(\textit{stack}[\textit{self}])]$$
$$\land \text{UNCHANGED } \langle \textit{the\_network}, \textit{local\_peer\_id\_},$$
$$\textit{remote\_peer\_id\_}, \textit{local\_peer\_id\_a},$$
$$\textit{remote\_peer\_id\_a}, \textit{local\_peer\_id\_v},$$
$$\textit{remote\_peer\_id\_v}, \textit{local\_peer\_id\_ve},$$
$$\textit{remote\_peer\_id\_ve}, \textit{hashes}, \textit{local\_peer\_id\_r},$$
$$\textit{remote\_peer\_id\_r}, \textit{local\_peer\_id\_i},$$
$$\textit{remote\_peer\_id\_i}, \textit{local\_peer\_id},$$
$$\textit{remote\_peer\_id}, \textit{blocks\_data}, \textit{command},$$
$$\textit{local\_peer\_index}, \textit{best\_tip} \rangle$$

$$\textit{getblocks}(\textit{self}) \triangleq \textit{HandleGetBlocksMsg}(\textit{self}) \lor \textit{SendInvMsg}(\textit{self})$$

$$\textit{SendGetBlocksMsg}(\textit{self}) \triangleq \land \textit{pc}[\textit{self}] = \text{``SendGetBlocksMsg''}$$
$$\land \textit{channels}' = [\textit{channels} \text{ EXCEPT } ![\textit{local\_peer\_id\_r}[\textit{self}]][\textit{remote\_peer\_id\_r}[\textit{self}]]$$

$$\land \textit{pc}' = [\textit{pc} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{pc}]$$
$$\land \textit{hashes}' = [\textit{hashes} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{hashes}]$$
$$\land \textit{local\_peer\_id\_r}' = [\textit{local\_peer\_id\_r} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{local\_pe}$$
$$\land \textit{remote\_peer\_id\_r}' = [\textit{remote\_peer\_id\_r} \text{ EXCEPT } ![\textit{self}] = \textit{Head}(\textit{stack}[\textit{self}]).\textit{rer}$$
$$\land \textit{stack}' = [\textit{stack} \text{ EXCEPT } ![\textit{self}] = \textit{Tail}(\textit{stack}[\textit{self}])]$$
$$\land \text{UNCHANGED } \langle \textit{the\_network}, \textit{local\_peer\_id\_},$$
$$\textit{remote\_peer\_id\_}, \textit{local\_peer\_id\_a},$$
$$\textit{remote\_peer\_id\_a}, \textit{local\_peer\_id\_v},$$
$$\textit{remote\_peer\_id\_v}, \textit{local\_peer\_id\_ve},$$
$$\textit{remote\_peer\_id\_ve}, \textit{local\_peer\_id\_g},$$
$$\textit{remote\_peer\_id\_g}, \textit{found\_blocks},$$
$$\textit{hash\_count}, \textit{block\_header\_hashes},$$
$$\textit{remote\_peer\_blocks}, \textit{start\_height},$$
$$\textit{end\_height}, \textit{local\_peer\_id\_i},$$
$$\textit{remote\_peer\_id\_i}, \textit{local\_peer\_id},$$
$$\textit{remote\_peer\_id}, \textit{blocks\_data}, \textit{command},$$
$$\textit{local\_peer\_index}, \textit{best\_tip} \rangle$$

16

$request\_blocks(self) \triangleq SendGetBlocksMsg(self)$

$SendGetDataMsg(self) \triangleq \land pc[self] = \text{"SendGetDataMsg"}$
$\land channels' = [channels \text{ EXCEPT } ![local\_peer\_id\_i[self]][remote\_peer\_id\_i[self]] =$

$]$
$\land pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$
$\land local\_peer\_id\_i' = [local\_peer\_id\_i \text{ EXCEPT } ![self] = Head(stack[self]).local\_pee$
$\land remote\_peer\_id\_i' = [remote\_peer\_id\_i \text{ EXCEPT } ![self] = Head(stack[self]).remo$
$\land stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])]$
$\land \text{UNCHANGED } \langle the\_network, local\_peer\_id\_,$
$\qquad\qquad remote\_peer\_id\_, local\_peer\_id\_a,$
$\qquad\qquad remote\_peer\_id\_a, local\_peer\_id\_v,$
$\qquad\qquad remote\_peer\_id\_v, local\_peer\_id\_ve,$
$\qquad\qquad remote\_peer\_id\_ve, local\_peer\_id\_g,$
$\qquad\qquad remote\_peer\_id\_g, found\_blocks,$
$\qquad\qquad hash\_count, block\_header\_hashes,$
$\qquad\qquad remote\_peer\_blocks, start\_height,$
$\qquad\qquad end\_height, hashes, local\_peer\_id\_r,$
$\qquad\qquad remote\_peer\_id\_r, local\_peer\_id,$
$\qquad\qquad remote\_peer\_id, blocks\_data, command,$
$\qquad\qquad local\_peer\_index, best\_tip \rangle$

$inv(self) \triangleq SendGetDataMsg(self)$

$HandleGetDataMsg(self) \triangleq \land pc[self] = \text{"HandleGetDataMsg"}$
$\land blocks\_data' = [blocks\_data \text{ EXCEPT } ![self] = \qquad\qquad [item \in 1 .. Len($
$\qquad\qquad Ops!FindBlockByHash($
$\qquad\qquad\quad Ops!GetPeerBlocks(the\_$
$\qquad\qquad\quad channels[local\_peer\_id[s$
$\qquad\qquad )$
$\qquad\qquad ]]$
$\land pc' = [pc \text{ EXCEPT } ![self] = \text{"IncorporateBlocks"}]$
$\land \text{UNCHANGED } \langle the\_network, channels, stack,$
$\qquad\qquad local\_peer\_id\_, remote\_peer\_id\_,$
$\qquad\qquad local\_peer\_id\_a, remote\_peer\_id\_a,$
$\qquad\qquad local\_peer\_id\_v, remote\_peer\_id\_v,$
$\qquad\qquad local\_peer\_id\_ve, remote\_peer\_id\_ve,$
$\qquad\qquad local\_peer\_id\_g, remote\_peer\_id\_g,$
$\qquad\qquad found\_blocks, hash\_count,$
$\qquad\qquad block\_header\_hashes,$
$\qquad\qquad remote\_peer\_blocks, start\_height,$
$\qquad\qquad end\_height, hashes, local\_peer\_id\_r,$
$\qquad\qquad remote\_peer\_id\_r, local\_peer\_id\_i,$
$\qquad\qquad remote\_peer\_id\_i, local\_peer\_id,$

17

$$\langle remote\_peer\_id,\ command,$$
$$local\_peer\_index,\ best\_tip\rangle$$

$IncorporateBlocks(self) \triangleq\ \land pc[self] =$ "IncorporateBlocks"
$\quad \land the\_network' = [the\_network\ \text{EXCEPT}\ ![local\_peer\_id[self]].blocks = the\_networ$
$\quad \land pc' = [pc\ \text{EXCEPT}\ ![self] = Head(stack[self]).pc]$
$\quad \land blocks\_data' = [blocks\_data\ \text{EXCEPT}\ ![self] = Head(stack[self]).blocks\_data]$
$\quad \land local\_peer\_id' = [local\_peer\_id\ \text{EXCEPT}\ ![self] = Head(stack[self]).local\_peer\_id$
$\quad \land remote\_peer\_id' = [remote\_peer\_id\ \text{EXCEPT}\ ![self] = Head(stack[self]).remote\_$
$\quad \land stack' = [stack\ \text{EXCEPT}\ ![self] = Tail(stack[self])]$
$\quad \land \text{UNCHANGED}\ \langle channels,\ local\_peer\_id\_,$
$\qquad\qquad remote\_peer\_id\_,\ local\_peer\_id\_a,$
$\qquad\qquad remote\_peer\_id\_a,\ local\_peer\_id\_v,$
$\qquad\qquad remote\_peer\_id\_v,\ local\_peer\_id\_ve,$
$\qquad\qquad remote\_peer\_id\_ve,\ local\_peer\_id\_g,$
$\qquad\qquad remote\_peer\_id\_g,\ found\_blocks,$
$\qquad\qquad hash\_count,\ block\_header\_hashes,$
$\qquad\qquad remote\_peer\_blocks,\ start\_height,$
$\qquad\qquad end\_height,\ hashes,\ local\_peer\_id\_r,$
$\qquad\qquad remote\_peer\_id\_r,\ local\_peer\_id\_i,$
$\qquad\qquad remote\_peer\_id\_i,\ command,$
$\qquad\qquad local\_peer\_index,\ best\_tip\rangle$

$getdata(self) \triangleq\ HandleGetDataMsg(self) \lor IncorporateBlocks(self)$

$Listening(self) \triangleq\ \land pc[self] =$ "Listening"
$\quad \land Len(the\_network) \geq 2$
$\quad \land \exists\, remote\_peer\_index \in 1\,..\,Len(the\_network[self].peer\_set):$
$\qquad \text{IF}\ channels[self][remote\_peer\_index].header = defaultInitValue$
$\qquad\quad \text{THEN}\ \land pc' = [pc\ \text{EXCEPT}\ ![self] =$ "Listening"$]$
$\qquad\quad \text{ELSE}\ \land pc' = [pc\ \text{EXCEPT}\ ![self] =$ "Requests"$]$
$\quad \land \text{UNCHANGED}\ \langle the\_network,\ channels,\ stack,$
$\qquad\qquad local\_peer\_id\_,\ remote\_peer\_id\_,$
$\qquad\qquad local\_peer\_id\_a,\ remote\_peer\_id\_a,$
$\qquad\qquad local\_peer\_id\_v,\ remote\_peer\_id\_v,$
$\qquad\qquad local\_peer\_id\_ve,\ remote\_peer\_id\_ve,$
$\qquad\qquad local\_peer\_id\_g,\ remote\_peer\_id\_g,$
$\qquad\qquad found\_blocks,\ hash\_count,$
$\qquad\qquad block\_header\_hashes,\ remote\_peer\_blocks,$
$\qquad\qquad start\_height,\ end\_height,\ hashes,$
$\qquad\qquad local\_peer\_id\_r,\ remote\_peer\_id\_r,$
$\qquad\qquad local\_peer\_id\_i,\ remote\_peer\_id\_i,$
$\qquad\qquad local\_peer\_id,\ remote\_peer\_id,\ blocks\_data,$
$\qquad\qquad command,\ local\_peer\_index,\ best\_tip\rangle$

$Requests(self) \triangleq\ \land pc[self] =$ "Requests"

$\wedge\,\exists\, remote\_peer\_index \in 1 \mathinner{.\,.} Len(the\_network[self].peer\_set):$
    $\wedge\, channels[self][remote\_peer\_index].header \neq defaultInitValue$
    $\wedge\, command' = [command \text{ EXCEPT } ![self] = channels[self][remote\_peer\_index].header.$
    $\wedge\, \text{IF } command'[self] = \text{"addr"}$
        $\text{THEN } \wedge\, \wedge\, local\_peer\_id\_a' = [local\_peer\_id\_a \text{ EXCEPT } ![self] = self]$
                  $\wedge\, remote\_peer\_id\_a' = [remote\_peer\_id\_a \text{ EXCEPT } ![self] = remote\_pee$
                  $\wedge\, stack' = [stack \text{ EXCEPT } ![self] = \langle[procedure \mapsto \text{"addr"},$
                                            $pc \qquad\quad\; \mapsto \text{"Listening"},$
                                            $local\_peer\_id\_a \mapsto \;\; local\_peer\_id\_$
                                            $remote\_peer\_id\_a \mapsto \;\; remote\_peer$
                                            $\circ\, stack[self]]$
                $\wedge\, pc' = [pc \text{ EXCEPT } ![self] = \text{"SendVersionMsg"}]$
                $\wedge\, \text{UNCHANGED } \langle local\_peer\_id\_v,$
                                      $remote\_peer\_id\_v,$
                                      $local\_peer\_id\_ve,$
                                      $remote\_peer\_id\_ve,$
                                      $local\_peer\_id\_g,$
                                      $remote\_peer\_id\_g,$
                                      $found\_blocks,\ hash\_count,$
                                      $block\_header\_hashes,$
                                      $remote\_peer\_blocks,$
                                      $start\_height,\ end\_height,$
                                      $local\_peer\_id\_i,$
                                      $remote\_peer\_id\_i,$
                                      $local\_peer\_id,$
                                      $remote\_peer\_id,\ blocks\_data\rangle$
        $\text{ELSE } \wedge\, \text{IF } command'[self] = \text{"version"}$
              $\text{THEN } \wedge\, \wedge\, local\_peer\_id\_v' = [local\_peer\_id\_v \text{ EXCEPT } ![self] = self$
                      $\wedge\, remote\_peer\_id\_v' = [remote\_peer\_id\_v \text{ EXCEPT } ![self] =$
                      $\wedge\, stack' = [stack \text{ EXCEPT } ![self] = \langle[procedure \mapsto \;\; \text{"version"}$
                                          $pc \qquad\quad\; \mapsto \;\; \text{"Listenin}$
                                          $local\_peer\_id\_v \mapsto \;\; lo$
                                          $remote\_peer\_id\_v \mapsto$
                                          $\circ\, stack[self]]$
                    $\wedge\, pc' = [pc \text{ EXCEPT } ![self] = \text{"HandleVersionMsg"}]$
                    $\wedge\, \text{UNCHANGED } \langle local\_peer\_id\_ve,$
                                        $remote\_peer\_id\_ve,$
                                        $local\_peer\_id\_g,$
                                        $remote\_peer\_id\_g,$
                                        $found\_blocks,$
                                        $hash\_count,$
                                        $block\_header\_hashes,$
                                        $remote\_peer\_blocks,$
                                        $start\_height,$
                                        $end\_height,$

19

$$
\begin{array}{l}
\qquad\qquad\qquad local\_peer\_id\_i, \\
\qquad\qquad\qquad remote\_peer\_id\_i, \\
\qquad\qquad\qquad local\_peer\_id, \\
\qquad\qquad\qquad remote\_peer\_id, \\
\qquad\qquad\qquad blocks\_data\rangle \\
\text{ELSE}\quad \wedge\text{ IF } command'[self] = \text{``verack''} \\
\qquad\quad\text{THEN}\quad \wedge\ \wedge\ local\_peer\_id\_ve' = [local\_peer\_id\_ve \text{ EXCEP} \\
\qquad\qquad\qquad\qquad \wedge\ remote\_peer\_id\_ve' = [remote\_peer\_id\_ve \text{ E}\times \\
\qquad\qquad\qquad\qquad \wedge\ stack' = [stack \text{ EXCEPT } ![self] = \langle[procedure \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad local\_peer \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad remote\_p \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\circ stack[se \\
\qquad\qquad\qquad \wedge\ pc' = [pc \text{ EXCEPT } ![self] = \text{``HandleVerackMsg''} \\
\qquad\qquad\qquad \wedge\ \text{UNCHANGED } \langle local\_peer\_id\_g, \\
\qquad\qquad\qquad\qquad\qquad\qquad remote\_peer\_id\_g, \\
\qquad\qquad\qquad\qquad\qquad\qquad found\_blocks, \\
\qquad\qquad\qquad\qquad\qquad\qquad hash\_count, \\
\qquad\qquad\qquad\qquad\qquad\qquad block\_header\_hashes, \\
\qquad\qquad\qquad\qquad\qquad\qquad remote\_peer\_blocks, \\
\qquad\qquad\qquad\qquad\qquad\qquad start\_height, \\
\qquad\qquad\qquad\qquad\qquad\qquad end\_height, \\
\qquad\qquad\qquad\qquad\qquad\qquad local\_peer\_id\_i, \\
\qquad\qquad\qquad\qquad\qquad\qquad remote\_peer\_id\_i, \\
\qquad\qquad\qquad\qquad\qquad\qquad local\_peer\_id, \\
\qquad\qquad\qquad\qquad\qquad\qquad remote\_peer\_id, \\
\qquad\qquad\qquad\qquad\qquad\qquad blocks\_data\rangle \\
\qquad\quad\text{ELSE}\quad \wedge\text{ IF } command'[self] = \text{``getblocks''} \\
\qquad\qquad\qquad\text{THEN}\quad \wedge\ \wedge\ local\_peer\_id\_g' = [local\_peer\_id \\
\qquad\qquad\qquad\qquad\qquad \wedge\ remote\_peer\_id\_g' = [remote\_pe \\
\qquad\qquad\qquad\qquad\qquad \wedge\ stack' = [stack \text{ EXCEPT } ![self] =
\end{array}
$$

$$
\begin{array}{l}
\wedge\ found\_blocks' = [found\_blocks \text{ EXC} \\
\wedge\ hash\_count' = [hash\_count \text{ EXCEPT} \\
\wedge\ block\_header\_hashes' = [block\_hea \\
\wedge\ remote\_peer\_blocks' = [remote\_pee
\end{array}
$$

20

$\land\ start\_height' = [start\_height\ \text{EXCE}$
$\land\ end\_height' = [end\_height\ \text{EXCEPT}$
$\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{"Handl}$
$\land\ \text{UNCHANGED}\ \langle local\_peer\_id\_i,$
$\qquad\qquad\qquad remote\_peer\_id\_i,$
$\qquad\qquad\qquad local\_peer\_id,$
$\qquad\qquad\qquad remote\_peer\_id,$
$\qquad\qquad\qquad blocks\_data\rangle$
$\text{ELSE}\quad \land\ \text{IF}\ command'[self] = \text{"inv"}$
$\qquad\qquad \text{THEN}\ \land\ \land\ local\_peer\_id\_i' = [$
$\qquad\qquad\qquad\qquad\quad \land\ remote\_peer\_id\_i' =$
$\qquad\qquad\qquad\qquad\quad \land\ stack' = [stack\ \text{EXC}$

$\land\ pc' = [pc\ \text{EXCEPT}\ ![se}$
$\land\ \text{UNCHANGED}\ \langle local\_pe}$
$\qquad\qquad\qquad\qquad remote\_$
$\qquad\qquad\qquad\qquad blocks\_d$
$\text{ELSE}\quad \land\ \text{IF}\ command'[self] = \text{"}$
$\qquad\qquad\text{THEN}\ \land\ \land\ local\_p$
$\qquad\qquad\qquad\qquad \land\ remote$
$\qquad\qquad\qquad\qquad \land\ stack' =$

$\land\ blocks\_da$
$\land\ pc' = [pc$
$\text{ELSE}\quad \land\ pc' = [pc$
$\qquad\quad \land\ \text{UNCHANG}$

$\land\ \text{UNCHANGED}\ \langle local\_pe}$
$\qquad\qquad\qquad\qquad\quad remote\_$
$\land\ \text{UNCHANGED}\ \langle local\_peer\_id\_g,$
$\qquad\qquad\qquad remote\_peer\_id\_g,$
$\qquad\qquad\qquad found\_blocks,$
$\qquad\qquad\qquad hash\_count,$
$\qquad\qquad\qquad block\_header\_hashes$
$\qquad\qquad\qquad remote\_peer\_blocks,$
$\qquad\qquad\qquad start\_height,$

$$end\_height\rangle$$
$$\wedge \text{UNCHANGED } \langle local\_peer\_id\_ve,$$
$$remote\_peer\_id\_ve\rangle$$
$$\wedge \text{UNCHANGED } \langle local\_peer\_id\_v,$$
$$remote\_peer\_id\_v\rangle$$
$$\wedge \text{UNCHANGED } \langle local\_peer\_id\_a,$$
$$remote\_peer\_id\_a\rangle$$
$$\wedge \text{UNCHANGED } \langle the\_network, channels, local\_peer\_id\_,$$
$$remote\_peer\_id\_, hashes, local\_peer\_id\_r,$$
$$remote\_peer\_id\_r, local\_peer\_index, best\_tip\rangle$$

$ListenerLoop(self) \triangleq \ \wedge pc[self] = \text{"ListenerLoop"}$
$\qquad \wedge \exists\, remote\_peer\_index \in 1 \mathinner{\ldotp\ldotp} Len(the\_network[self].peer\_set) :$
$\qquad\qquad \wedge channels' = [channels \text{ EXCEPT } ![self][remote\_peer\_index] = [header \mapsto defaultI...$
$\qquad\qquad \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"Listening"}]$
$\qquad \wedge \text{UNCHANGED } \langle the\_network, stack, local\_peer\_id\_,$
$\qquad\qquad\qquad remote\_peer\_id\_, local\_peer\_id\_a,$
$\qquad\qquad\qquad remote\_peer\_id\_a, local\_peer\_id\_v,$
$\qquad\qquad\qquad remote\_peer\_id\_v, local\_peer\_id\_ve,$
$\qquad\qquad\qquad remote\_peer\_id\_ve, local\_peer\_id\_g,$
$\qquad\qquad\qquad remote\_peer\_id\_g, found\_blocks,$
$\qquad\qquad\qquad hash\_count, block\_header\_hashes,$
$\qquad\qquad\qquad remote\_peer\_blocks, start\_height,$
$\qquad\qquad\qquad end\_height, hashes, local\_peer\_id\_r,$
$\qquad\qquad\qquad remote\_peer\_id\_r, local\_peer\_id\_i,$
$\qquad\qquad\qquad remote\_peer\_id\_i, local\_peer\_id,$
$\qquad\qquad\qquad remote\_peer\_id, blocks\_data, command,$
$\qquad\qquad\qquad local\_peer\_index, best\_tip\rangle$

$LISTENER(self) \triangleq Listening(self) \vee Requests(self) \vee ListenerLoop(self)$

$Announce(self) \triangleq \ \wedge pc[self] = \text{"Announce"}$
$\qquad \wedge Assert(Len(the\_network) \geq 2,$
$\qquad\qquad \text{"Failure of assertion at line 335, column 9."})$
$\qquad \wedge Len(the\_network[local\_peer\_index[self]].peer\_set) > 0$
$\qquad \wedge \exists\, remote\_peer\_index \in 1 \mathinner{\ldotp\ldotp} Len(the\_network[local\_peer\_index[self]].peer\_set) :$
$\qquad\qquad \wedge \wedge local\_peer\_id\_' = [local\_peer\_id\_ \text{ EXCEPT } ![self] = local\_peer\_index[self]]$
$\qquad\qquad\qquad \wedge remote\_peer\_id\_' = [remote\_peer\_id\_ \text{ EXCEPT } ![self] = remote\_peer\_index]$
$\qquad\qquad\qquad \wedge stack' = [stack \text{ EXCEPT } ![self] = \langle[procedure \mapsto \text{"announce"},$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \qquad \mapsto \text{"RequestInventory"},$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad local\_peer\_id\_ \mapsto local\_peer\_id\_[self],$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad remote\_peer\_id\_ \mapsto remote\_peer\_id\_[self]]\rangle$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ stack[self]]$
$\qquad\qquad \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"SendAddrMsg"}]$
$\qquad \wedge \text{UNCHANGED } \langle the\_network, channels, local\_peer\_id\_a,$
$\qquad\qquad\qquad remote\_peer\_id\_a, local\_peer\_id\_v,$

$$remote\_peer\_id\_v, \; local\_peer\_id\_ve,$$
$$remote\_peer\_id\_ve, \; local\_peer\_id\_g,$$
$$remote\_peer\_id\_g, \; found\_blocks, \; hash\_count,$$
$$block\_header\_hashes, \; remote\_peer\_blocks,$$
$$start\_height, \; end\_height, \; hashes,$$
$$local\_peer\_id\_r, \; remote\_peer\_id\_r,$$
$$local\_peer\_id\_i, \; remote\_peer\_id\_i,$$
$$local\_peer\_id, \; remote\_peer\_id, \; blocks\_data,$$
$$command, \; local\_peer\_index, \; best\_tip\rangle$$

$RequestInventory(self) \triangleq \; \wedge pc[self] = \text{``RequestInventory''}$

$\qquad \wedge \exists \, remote\_peer\_index \in 1 \, .. \, Len(the\_network[local\_peer\_index[self]].peer\_set):$

$\qquad\qquad \wedge the\_network[local\_peer\_index[self]].peer\_set[remote\_peer\_index].established$

$\qquad\qquad \wedge \text{IF } the\_network[local\_peer\_index[self]].peer\_set[remote\_peer\_index].tip > bes$

$\qquad\qquad\qquad \text{THEN } \wedge best\_tip' = [best\_tip \text{ EXCEPT } ![self] = the\_network[local\_peer\_i$

$\qquad\qquad\qquad \text{ELSE } \wedge \text{TRUE}$

$\qquad\qquad\qquad\qquad \wedge \text{UNCHANGED } best\_tip$

$\qquad\qquad \wedge \quad channels[local\_peer\_index[self]][remote\_peer\_index].header = defaultInitV$

$\qquad\qquad\qquad \wedge channels[local\_peer\_index[self]][remote\_peer\_index].payload = defaultInit$

$\qquad\qquad \wedge \text{IF } Ops!GetPeerTipByIndex(local\_peer\_index[self]).height <$

$\qquad\qquad\qquad the\_network[local\_peer\_index[self]].peer\_set[remote\_peer\_index].tip$

$\qquad\qquad\qquad \text{THEN } \wedge \text{IF } Ops!GetPeerTipByIndex(local\_peer\_index[self]).height = 0$

$\qquad\qquad\qquad\qquad \text{THEN } \wedge \wedge hashes' = [hashes \text{ EXCEPT } ![self] = \langle\rangle]$

$\qquad\qquad\qquad\qquad\qquad\qquad \wedge local\_peer\_id\_r' = [local\_peer\_id\_r \text{ EXCEPT } ![se$

$\qquad\qquad\qquad\qquad\qquad\qquad \wedge remote\_peer\_id\_r' = [remote\_peer\_id\_r \text{ EXCEPT}$

$\qquad\qquad\qquad\qquad\qquad\qquad \wedge stack' = [stack \text{ EXCEPT } ![self] = \langle[procedure \mapsto$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \qquad \mapsto$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad hashes \quad \mapsto$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad local\_peer\_id$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad remote\_peer\_$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ stack[self]]$

$\qquad\qquad\qquad\qquad\qquad \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{``SendGetBlocksMsg''}]$

$\qquad\qquad\qquad\qquad \text{ELSE } \wedge \wedge hashes' = [hashes \text{ EXCEPT } ![self] = \langle Ops!GetP$

$\qquad\qquad\qquad\qquad\qquad\qquad \wedge local\_peer\_id\_r' = [local\_peer\_id\_r \text{ EXCEPT } ![se$

$\qquad\qquad\qquad\qquad\qquad\qquad \wedge remote\_peer\_id\_r' = [remote\_peer\_id\_r \text{ EXCEPT}$

$\qquad\qquad\qquad\qquad\qquad\qquad \wedge stack' = [stack \text{ EXCEPT } ![self] = \langle[procedure \mapsto$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \qquad \mapsto$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad hashes \quad \mapsto$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad local\_peer\_id$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad remote\_peer\_$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ stack[self]]$

$\qquad\qquad\qquad\qquad\qquad \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{``SendGetBlocksMsg''}]$

$\qquad\qquad\qquad \text{ELSE } \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{``CheckSync''}]$

$\qquad\qquad\qquad\qquad \wedge \text{UNCHANGED } \langle stack, \; hashes,$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad local\_peer\_id\_r,$

$$remote\_peer\_id\_r\rangle$$
$$\land \text{UNCHANGED } \langle the\_network, \ channels,$$
$$local\_peer\_id\_, \ remote\_peer\_id\_,$$
$$local\_peer\_id\_a, \ remote\_peer\_id\_a,$$
$$local\_peer\_id\_v, \ remote\_peer\_id\_v,$$
$$local\_peer\_id\_ve, \ remote\_peer\_id\_ve,$$
$$local\_peer\_id\_g, \ remote\_peer\_id\_g,$$
$$found\_blocks, \ hash\_count,$$
$$block\_header\_hashes,$$
$$remote\_peer\_blocks, \ start\_height,$$
$$end\_height, \ local\_peer\_id\_i,$$
$$remote\_peer\_id\_i, \ local\_peer\_id,$$
$$remote\_peer\_id, \ blocks\_data, \ command,$$
$$local\_peer\_index\rangle$$

$CheckSync(self) \;\triangleq\; \land pc[self] = \text{``CheckSync''}$
$\qquad\qquad \land Ops\,!\,GetPeerTipByIndex(local\_peer\_index[self]).height > 0$
$\qquad\qquad \land \text{IF } Ops\,!\,GetPeerTipByIndex(local\_peer\_index[self]).height < best\_tip[self]$
$\qquad\qquad\qquad \text{THEN } \land pc' = [pc \text{ EXCEPT } ![self] = \text{``RequestInventory''}]$
$\qquad\qquad\qquad \text{ELSE } \land \exists\, remote\_peer\_index \in 1\,..\, Len(the\_network[local\_peer\_index[self]].peer\_s$
$\qquad\qquad\qquad\qquad\qquad the\_network[local\_peer\_index[self]].peer\_set[remote\_peer\_index].establis$
$\qquad\qquad\qquad\qquad\qquad \land channels[local\_peer\_index[self]][remote\_peer\_index].header = defaultI$
$\qquad\qquad\qquad\qquad\qquad \land channels[local\_peer\_index[self]][remote\_peer\_index].payload = default$
$\qquad\qquad\qquad\quad \land pc' = [pc \text{ EXCEPT } ![self] = \text{``Done''}]$
$\qquad\qquad \land \text{UNCHANGED } \langle the\_network, \ channels, \ stack,$
$$local\_peer\_id\_, \ remote\_peer\_id\_,$$
$$local\_peer\_id\_a, \ remote\_peer\_id\_a,$$
$$local\_peer\_id\_v, \ remote\_peer\_id\_v,$$
$$local\_peer\_id\_ve, \ remote\_peer\_id\_ve,$$
$$local\_peer\_id\_g, \ remote\_peer\_id\_g,$$
$$found\_blocks, \ hash\_count,$$
$$block\_header\_hashes, \ remote\_peer\_blocks,$$
$$start\_height, \ end\_height, \ hashes,$$
$$local\_peer\_id\_r, \ remote\_peer\_id\_r,$$
$$local\_peer\_id\_i, \ remote\_peer\_id\_i,$$
$$local\_peer\_id, \ remote\_peer\_id, \ blocks\_data,$$
$$command, \ local\_peer\_index, \ best\_tip\rangle$$

$SYNCHRONIZER(self) \;\triangleq\; Announce(self) \lor RequestInventory(self)$
$\qquad\qquad\qquad\qquad \lor CheckSync(self)$

*Allow infinite stuttering to prevent deadlock on termination.*
$Terminating \;\triangleq\; \land \forall\, self \in ProcSet : pc[self] = \text{``Done''}$
$\qquad\qquad\quad \land \text{UNCHANGED } vars$

$Next \;\triangleq\; (\exists\, self \in ProcSet : \; \lor announce(self) \lor addr(self)$

24

$$\lor version(self) \lor verack(self)$$
$$\lor getblocks(self) \lor request\_blocks(self)$$
$$\lor inv(self) \lor getdata(self))$$
$$\lor (\exists\, self \in 1 \ldots Len(RunningBlockchain) : LISTENER(self))$$
$$\lor (\exists\, self \in PeerProcessDiffId + 1 \ldots PeerProcessDiffId + Len(RunningBlockchain) : SYNCHRONIZ$$
$$\lor Terminating$$

$Spec \;\triangleq\; Init \land \Box[Next]_{vars}$

$Termination \;\triangleq\; \Diamond(\forall\, self \in ProcSet : pc[self] = \text{"Done"})$

END TRANSLATION