TSG-RAN Working Group 2 (Radio layer 2 and Radio layer 3) ***TSGR2#6(99)725***
Sophia Antipolis 16th to 20th August1999

Agenda Item: 5

Source: NTT DoCoMo

Title: **Report on RLC SDL e-mail discussion**

Document for: Information

In starting the discussion, DoCoMo proposed to use new SDL that includes all of the toolbox functions in stead of the SDL of Tdoc-99629. Since there was no objection to using it, this e-mail discussion was carried out using the SDL. And there were two comments from Anite Telecoms.

1) It was pointed out that all of the definitions of vt_xxx and vr_xxx should define them as referring to not sequences of PDUs but sequences of PUs since they are defined as referring to sequences of PUs in TS25.322 (V1.1.1).
   -> The definition of vt_s, vt_a, vt_dat, vt_ms, vr_r, vr_h, and vr_mr were corrected.

2) It was suggested that “transmission_queue” is changed to “transmitted_queue” as a more meaningful name as the original name almost implies that the data is still to be sent.
   -> It was agreed to change the name.

Since sufficient discussion was not carried out this time, it is necessary to continue e-mail discussion to make the SDL better. Current SDL is shown below.

;

```
DCL

 am_pdu, tmp, pdu                                AmPdu,
 /*A representation of data contained within a AmPdu.*/

 stat_pdu, rx_stat_pdu                           StatPdu,
 /*A representation of data contained within a StatPdu.*/

 pdus, rem_pdus                                  AmPduArrayType,
 /*The initally segmented sdu.*/

 receiver_queue                                  Queue,
 /*A queue used for storing PDUs as they arrive.*/

 retransmission_queue                            Queue,
 /*A queue used for PDUs that are to be retransmitted.*/

 assembly_queue                                  Queue,
 /*A queue used for reassembly of received PDUs into an SDU.*/

 transmitted_queue                               Queue,
 /*A queue used for PDUs that have been transmitted.*/

 am_queue                                        Queue,
 /*A queue used for PDUs to be transmitted.*/

 stat_queue                                      Queue,
 /* Queues used for PDUs associated with STATUS Pdus to be transmitted.*/

 prohibit , rx_prohibit, epc_active              IndicatorType,
 /*An indicator used to determine whether the timer_PROHIBIT
   is running or not.*/

 empty, no_tx, no_retx                           IndicatorType,
 /*An indicator used to determine whether a queue is empty or not.*/

 exists                                          IndicatorType,
  /*An indicator used to determine whether a particular pdu exists
    within a queue or not.*/

 poll_triggers                                   PollTriggArrType,
 /*a configuration parameter dealing with when to issue poll requests.*/

 status_triggers                                 StatusTriggArrType,
 /*A configuraion parameter dealing with when to issue Status reports.*/

 rx_period                                       DURATION,
 /*The duration of a periodic Statut report generation timer.*/

 rx_prohibdur, epc_dur                           DURATION,
  /*The duration of a prohibit retransmission of status report timer.*/

 discard                                         DiscardArrayType,
  /*A configuration parameter identifying discard conditions.*/

 complete, cnf                                   IndicatorType;
  /*An indicator used to determine whether an SDU has been
     completely reassembled or whether an SDU requires confirmation.*/
```

;

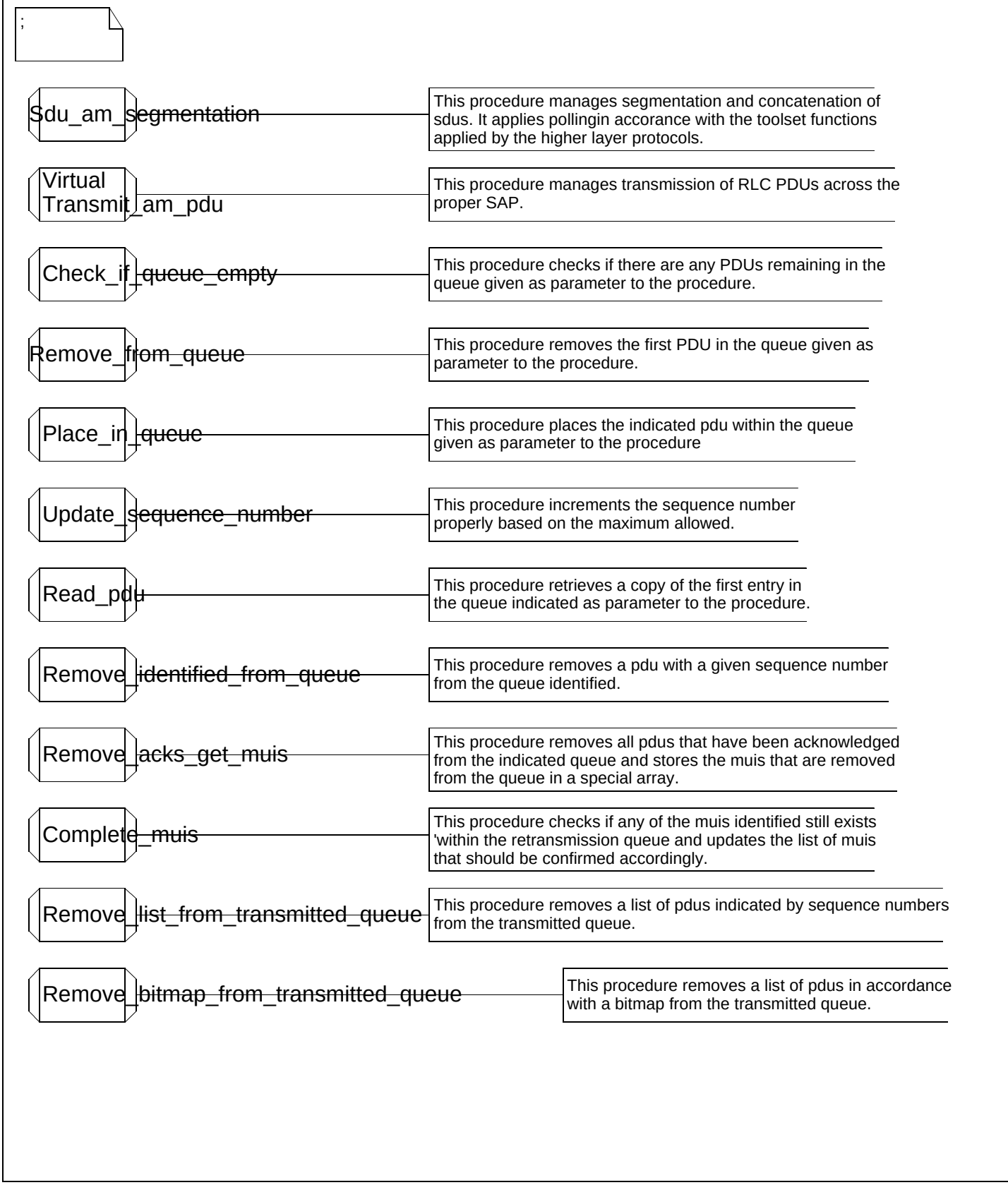
```
DCL
 period                                          DURATION,
 /*The duration of a periodic Polling generation timer.*/

 retransmission                  IndicatorType,
 /*An indicator used to determine whether the received PDU is a retransmission.*/

 logical_channel                 LogicalChannelType,
 /*The logical channel associated with transmissions.*/

 i                                       INTEGER,
 /*A local counter.*/

 mui                                     MuiType,
  /*The message uit identifier associated with a message to be transmitted.*/

 muis                                    MuiArrayType,
  /*An array used to store message unit identifiers.*/

 no_sdu, no_pu, xpu, xsdu,tx_win, rx_win, no_of_pu_per_tti,
 rx_pu, rx_sdu, muis_tot, tot, k, no_of_sq, tot_rem, l, no_s      PduIndexType,
 /*Counters used to manage the amount of PUs and SDUs received.*/

 percent, rx_percent             REAL,
  /*Percentages of the transmit and receive window.*/

 sdu                                     OctetType,
 /*The sdu data from the higher protocol layer.*/

 sdus                                    OctetArrayType,
 /*A set of octets.*/

 seq, n, np, sn_ack, sq, sn              SequenceNumberType,
 /*A local sequence number.*/

 vt_s                                    SequenceNumberType,
 /*Send state variable: The sequence number of the next PU to be transmitted for the first time.
   It is incremented after transmission of a PU for the first time (i.e. excluding retransmissions).*/

 vt_a                                    SequenceNumberType,
 /*Acknowledge state variable: The sequence number of  the next in-sequence PU expected to
   be acknowledged, thus forming the lower edge of the window of acceptable acknowledgements.
   The variable vt_a is updated upon acknowledgement of in-sequence PUs.*/

 vt_dat                                  SequenceNumberType,
 /*This variable is used to count the retransmission number of each PU. It is incremented by a
   PU transmission.*/

 vt_ms                                   SequenceNumberType;
 /*Maximum send state variable: This is the sequence number of the first PU not allowed by the
   receiver. It thus represents the upper edge of the transmit window. If vt_s is equal to vt_ms, now
   new PU should be transmitted. The variable is updated based on receipt of STATUS PDU.*/
```

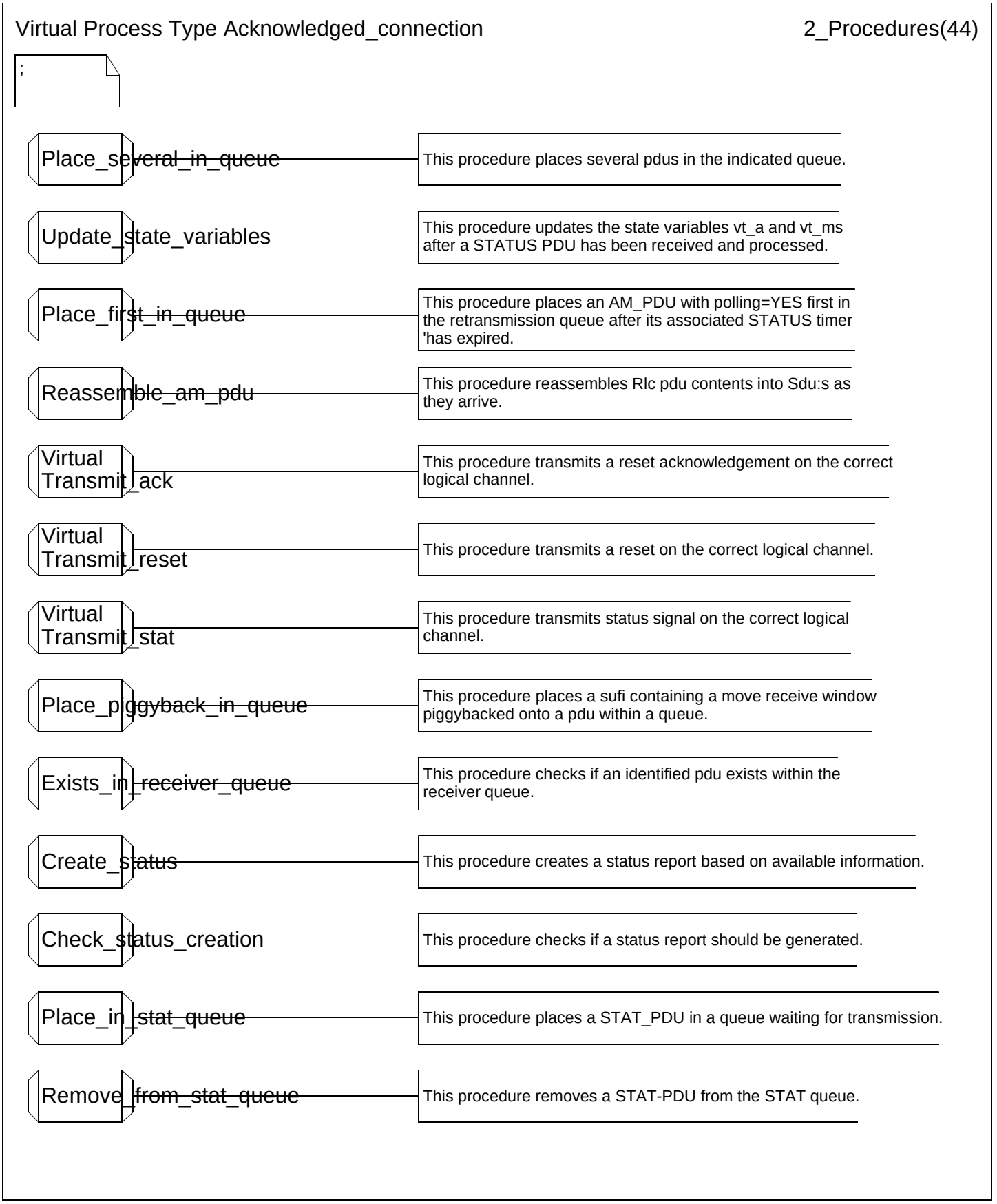
;

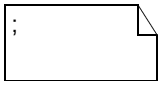
```
DCL

 vr_r                                  SequenceNumberType,
/*Receive state variable: The sequence number of the next in sequence PU expected to be received.
  It is incremented upon receipt of the next in-sequence PU.*/

vr_h                                   SequenceNumberType,
/*Highest expected state variable: The sequence number of the next highest expected PU. The variable
  is updated whenever a new PU is received.*/

vr_mr                                  SequenceNumberType,
/*Maximum acceptable receive state variable: The sequence number of the first PU not allowed by the
  receiver, thus the receiver shall discard PUs  with an n_s=vr_mr. Updating of vr_mr is implementation
  dependent but should not be set to a value less than vr_h.*/

rx_sufi_tot                            PduIndexType,
/*Local variable for maintaining knowledge of the number of super fields.*/

 tx_sufi                               SufiStructType,
 /*The contents of one superfield.*/

 rx_sufis, sufis, tx_sufis             SufiArrayStructType,
 /*The set of superfields associated with a status report.*/

flip, possible, status, rx_flip, polling_answer                    IndicatorType,
/*An indicator used in or to determine whether the highest sequence number value has been passed or not.
   THe second is used to indicate whether status pigyyback is possible or not.*/

retransmissions_requested              IndicatorType,
 /*An indicator used to keep track whether a generated status report contains retransmission requests or not.*/

 status_timer_active, start_am         IndicatorType,
 /*This indicator keeps track of whether the timer_STATUS timer is running or not.*/

 per                                   REAL,
 /*Local storage of a percentage value.*/

 rx_ongoing, tx_ongoing                IndicatorType,
 /*These indicators are used to maintain information about whether something is in the process of being
    transmitted or received.*/

 bitmap                                IndicatorArrayType,
 /*This array ofboolean values indicates losses experienced by the receiver.*/

vr_ep                                  SequenceNumberType;
/*Estimated PDU counter state variable: The number of PUs that should have been received after the latest
   STATU PDU was sent. In acknowledged mode, this state variable is updated at the end of each transmission
   time interval. If vr_ep is equal to the number of requested PUs in the latest STATUS PDU it should be checked
   if all PUs requested for retransmission have been received.*/
```

;

```
TIMER

 timer_AM,
 /*This timer is used to sequence transmissions.*/

 timer_EPC,
/*This timer accounts for the round trip delay, i.e. the time when the first retransmitted PU should have been
   received after a status report has been sent. The value of timer is heavily based on the transmission time
   interval (layer 1 interleaving depth). When changing the transmission time interval, the value of the EPC
   timer also needs to be changed.*/

 timer_STATUS,
/*This timer is used to detect the loss of response from the receiver side. The timer is set when a transmitted AmPdu
   requests a status report and it will be stopped when the transmitter receives acknowledgement of the pdu within StatPdu
   (positive) or UstatPdu (negative). When the timer expires, the pdus of the oldest unconfirmed pdus should be
   retransmitted together with a status report request and the timer set again. If polling takes place when this timer
   is active, it should be reset an then se again.*/

 timer_DISCARD(MuiType),
 /*This timer is used for the SDU discard function. In the transmitter, the timer is activated upon reception of an
    SDU from a higher layer. If the SDU has not been acknowledged when the timer expires, the SDU is discarded
    and a move receiving window request is sent to the receiver. If the SDU discard function does not use the move
    receiving window request, the timer is also used in the receiver, where it is activated once a PDU is detected as
    outstanding, i.e. there is a gap betweensequencembers of received PDUs.*/

 timer_PERIOD,
 /*A timer used for the periodic creation of polls.*/

 timer_RXPERIOD,
 /*A timer used for the periodic creation of status reports.*/

 timer_RXPROHIBIT,
 /*A timer used on the receive side to limit STATUS transmissions.*/

timer_PROHIBIT;
/*It is used to prohibit transmission of polling messages within a certain period. If polling takes place while the timer is
  active, it will be reset and then set again. No action other than indicating that the timer is not active is needed
  when it expires.*/
```


Virtual Process Type Acknowledged_connection
1_Procedures(44)
;
Sdu_am_segmentation
This procedure manages segmentation and concatenation of sdus. It applies pollingin accorance with the toolset functions applied by the higher layer protocols.
Virtual Transmit_am_pdu
This procedure manages transmission of RLC PDUs across the proper SAP.
Check_if_queue_empty
This procedure checks if there are any PDUs remaining in the queue given as parameter to the procedure.
Remove_from_queue
This procedure removes the first PDU in the queue given as parameter to the procedure.
Place_in_queue
This procedure places the indicated pdu within the queue given as parameter to the procedure
Update_sequence_number
This procedure increments the sequence number properly based on the maximum allowed.
Read_pdu
This procedure retrieves a copy of the first entry in the queue indicated as parameter to the procedure.
Remove_identified_from_queue
This procedure removes a pdu with a given sequence number from the queue identified.
Remove_acks_get_muis
This procedure removes all pdus that have been acknowledged from the indicated queue and stores the muis that are removed from the queue in a special array.
Complete_muis
This procedure checks if any of the muis identified still exists 'within the retransmission queue and updates the list of muis that should be confirmed accordingly.
Remove_list_from_transmitted_queue
This procedure removes a list of pdus indicated by sequence numbers from the transmitted queue.
Remove_bitmap_from_transmitted_queue
This procedure removes a list of pdus in accordance with a bitmap from the transmitted queue.

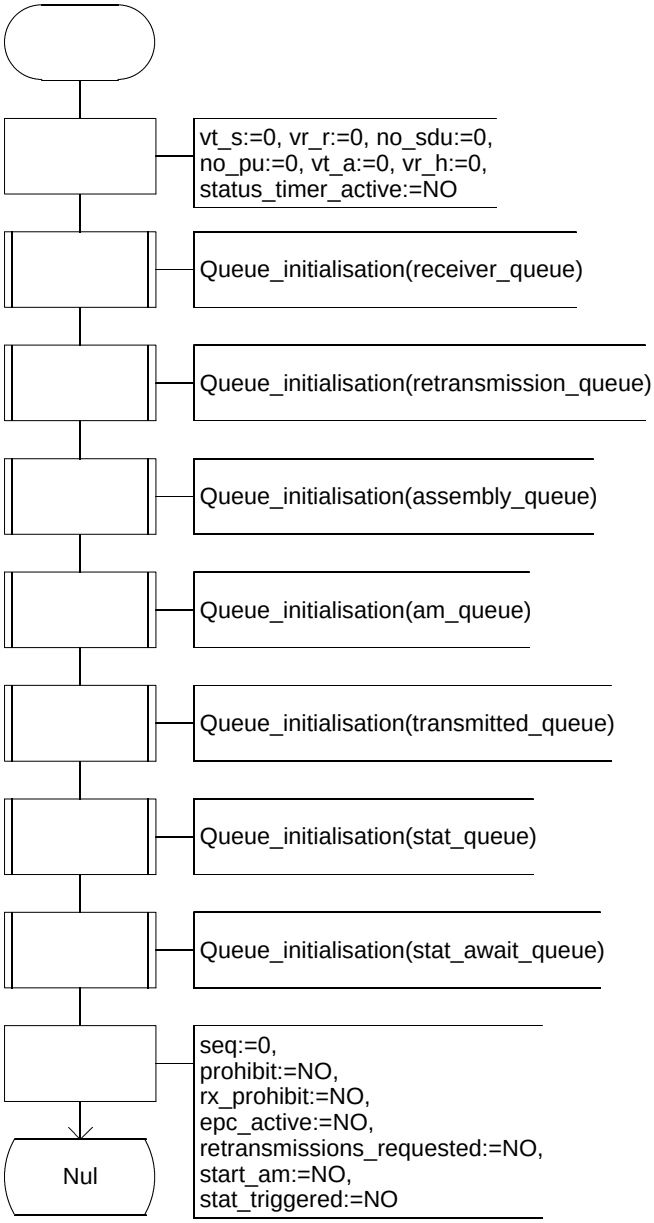

Virtual Process Type Acknowledged_connection
2_Procedures(44)
;
Place_several_in_queue
This procedure places several pdus in the indicated queue.
Update_state_variables
This procedure updates the state variables vt_a and vt_ms after a STATUS PDU has been received and processed.
Place_first_in_queue
This procedure places an AM_PDU with polling=YES first in the retransmission queue after its associated STATUS timer 'has expired.
Reassemble_am_pdu
This procedure reassembles Rlc pdu contents into Sdu:s as they arrive.
Virtual Transmit_ack
This procedure transmits a reset acknowledgement on the correct logical channel.
Virtual Transmit_reset
This procedure transmits a reset on the correct logical channel.
Virtual Transmit_stat
This procedure transmits status signal on the correct logical channel.
Place_piggyback_in_queue
This procedure places a sufi containing a move receive window piggybacked onto a pdu within a queue.
Exists_in_receiver_queue
This procedure checks if an identified pdu exists within the receiver queue.
Create_status
This procedure creates a status report based on available information.
Check_status_creation
This procedure checks if a status report should be generated.
Place_in_stat_queue
This procedure places a STAT_PDU in a queue waiting for transmission.
Remove_from_stat_queue
This procedure removes a STAT-PDU from the STAT queue.



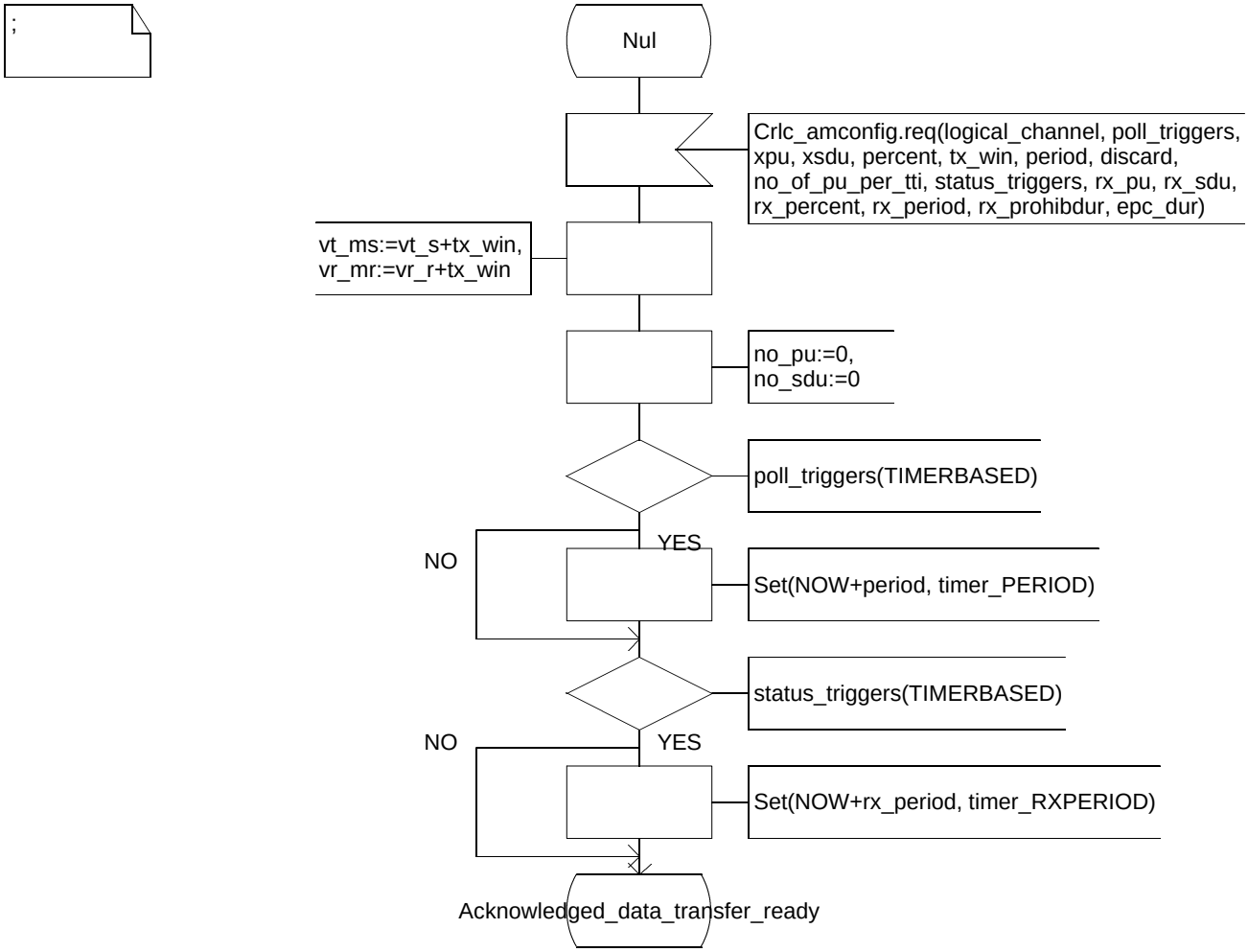
Remove_mui_from_queues	This procedure removes all pdus associated with a given mui from the transmitted_queue.
Remove_all_below_from_queues	This procedure removes all pdus below an identified sequence number from all receiver queues.
Set_polling_flag	This procedure causes the polling flag to be set in the first PDU within a defined queue
Count_epc	This procedure counts the received PDUs.
Exists_in_stat_await_queue	This procedure checks whether the STATUS PDU includes ACK or NACK for the AMD PDU which triggered timer_STATUS.
Replace_am_pdu	This procedure places the AMD PDU which triggered timer_STATUS in the STAT_await queue. If other AMD PDU has alredy been existing in the queue, the old one will be replaced with the new one.

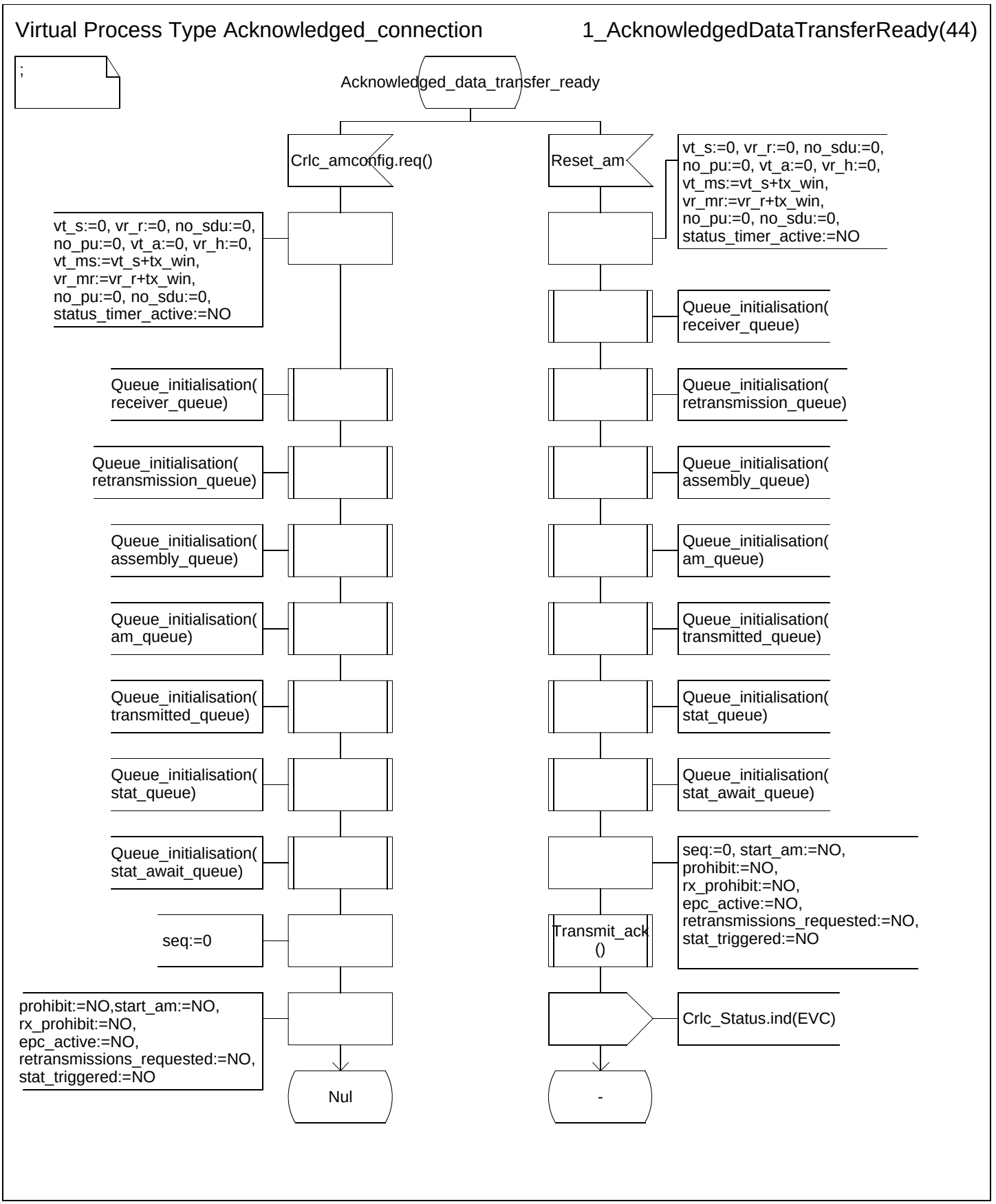
Virtual Process Type Acknowledged_connection 1_ProcessTypeStart(44)

;

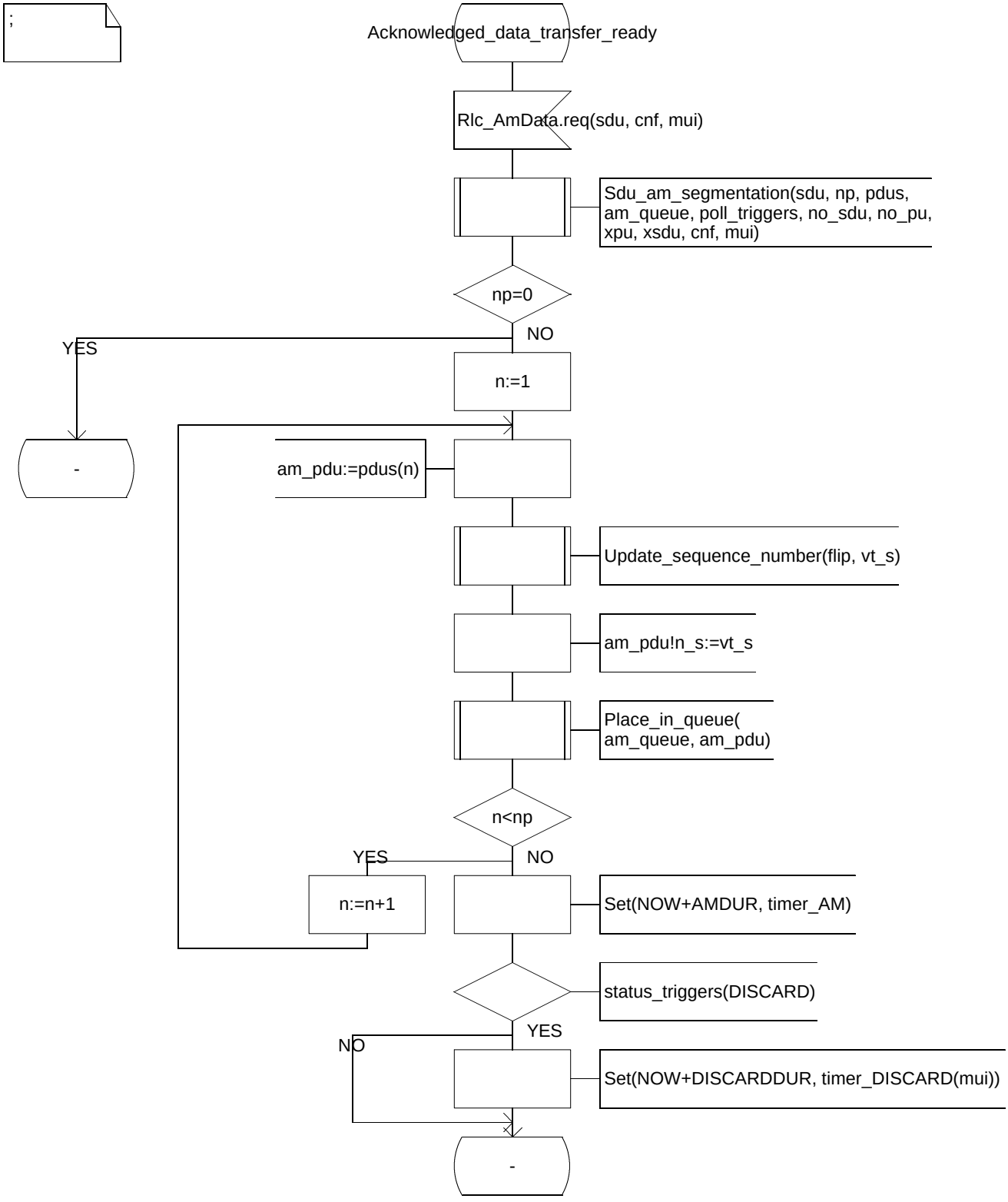


Virtual Process Type Acknowledged_connection 1_Nul(44)

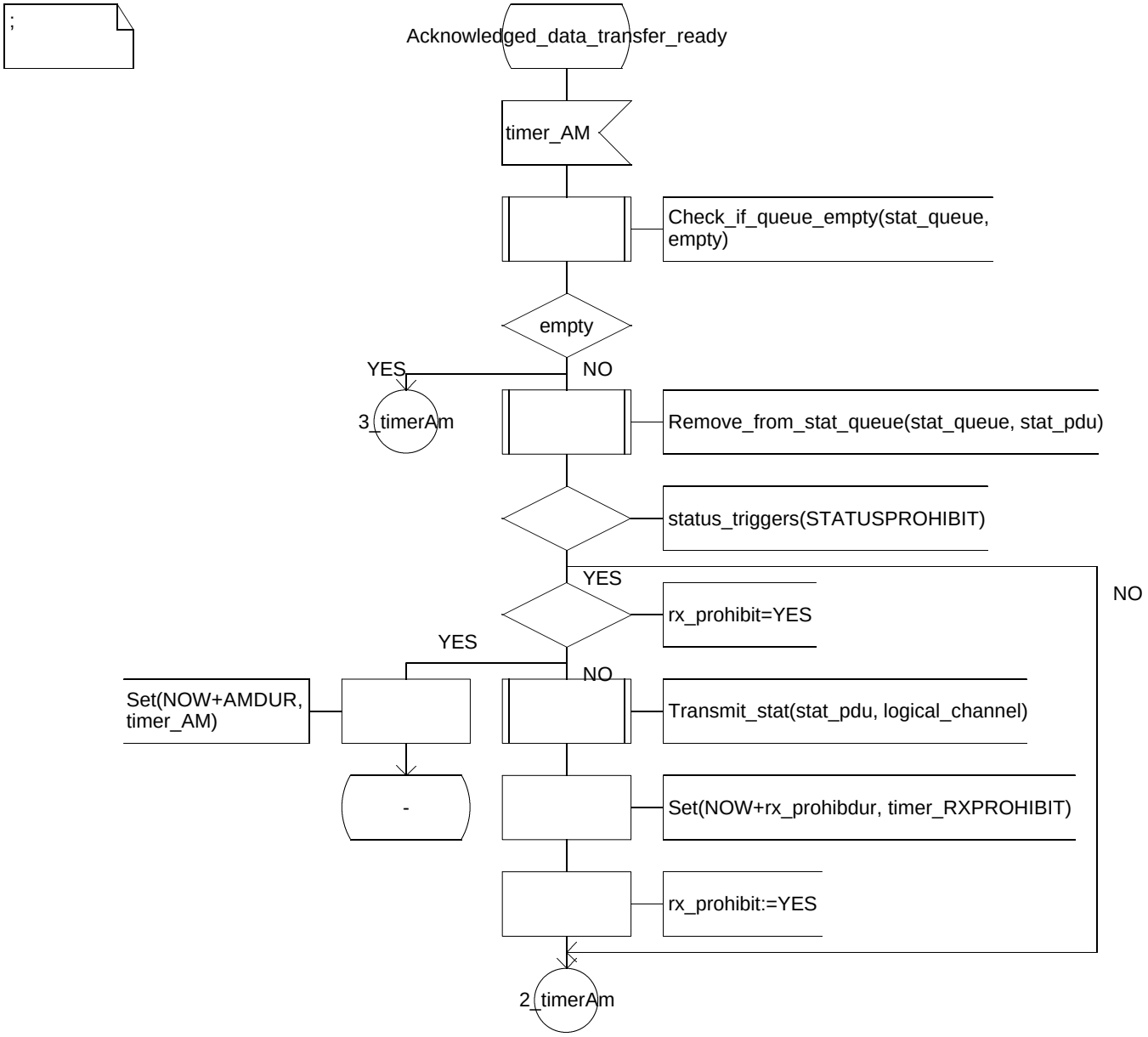



Virtual Process Type Acknowledged_connection
1_AcknowledgedDataTransferReady(44)
;
Acknowledged_data_transfer_ready
Crlc_amconfig.req()
Reset_am
vt_s:=0, vr_r:=0, no_sdu:=0, no_pu:=0, vt_a:=0, vr_h:=0, vt_ms:=vt_s+tx_win, vr_mr:=vr_r+tx_win, no_pu:=0, no_sdu:=0, status_timer_active:=NO
vt_s:=0, vr_r:=0, no_sdu:=0, no_pu:=0, vt_a:=0, vr_h:=0, vt_ms:=vt_s+tx_win, vr_mr:=vr_r+tx_win, no_pu:=0, no_sdu:=0, status_timer_active:=NO
Queue_initialisation(receiver_queue)
Queue_initialisation(receiver_queue)
Queue_initialisation(retransmission_queue)
Queue_initialisation(retransmission_queue)
Queue_initialisation(assembly_queue)
Queue_initialisation(assembly_queue)
Queue_initialisation(am_queue)
Queue_initialisation(am_queue)
Queue_initialisation(transmitted_queue)
Queue_initialisation(transmitted_queue)
Queue_initialisation(stat_queue)
Queue_initialisation(stat_queue)
Queue_initialisation(stat_await_queue)
Queue_initialisation(stat_await_queue)
seq:=0, start_am:=NO, prohibit:=NO, rx_prohibit:=NO, epc_active:=NO, retransmissions_requested:=NO, stat_triggered:=NO
seq:=0
Transmit_ack ()
prohibit:=NO,start_am:=NO, rx_prohibit:=NO, epc_active:=NO, retransmissions_requested:=NO, stat_triggered:=NO
Crlc_Status.ind(EVC)
Nul
-

Virtual Process Type Acknowledged_conn1_AcknowledgedDataTransferReady_RlcAmDataReq(44)

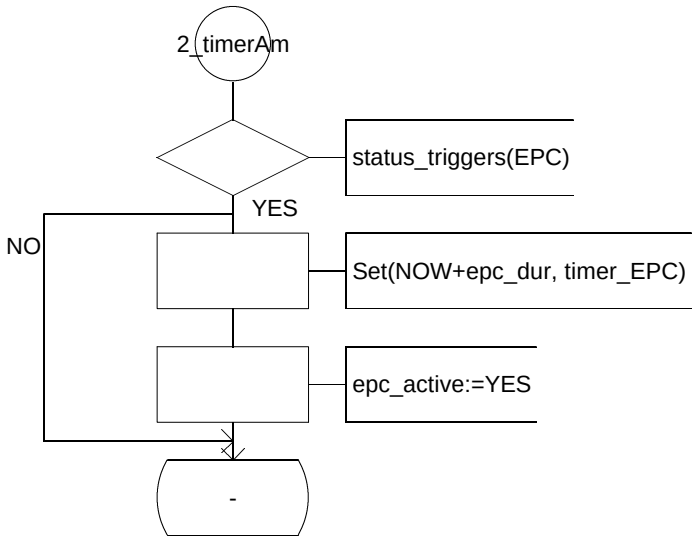


Virtual Process Type Acknowledged_connection 1_AcknowledgedDataTransferReady_timerAm(44)

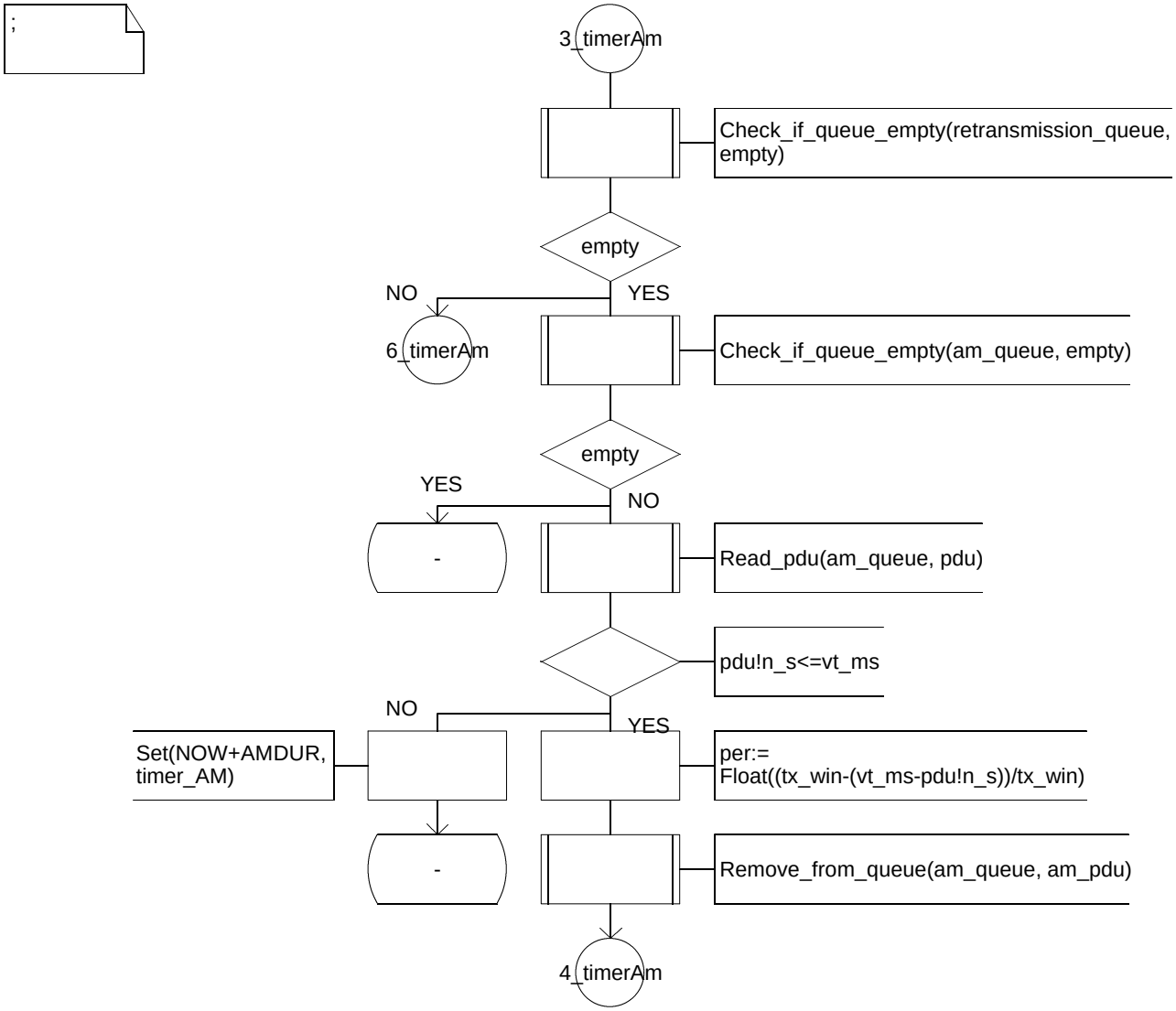


Virtual Process Type Acknowledged_connection 2_AcknowledgedDataTransferReady_timerAm(44)

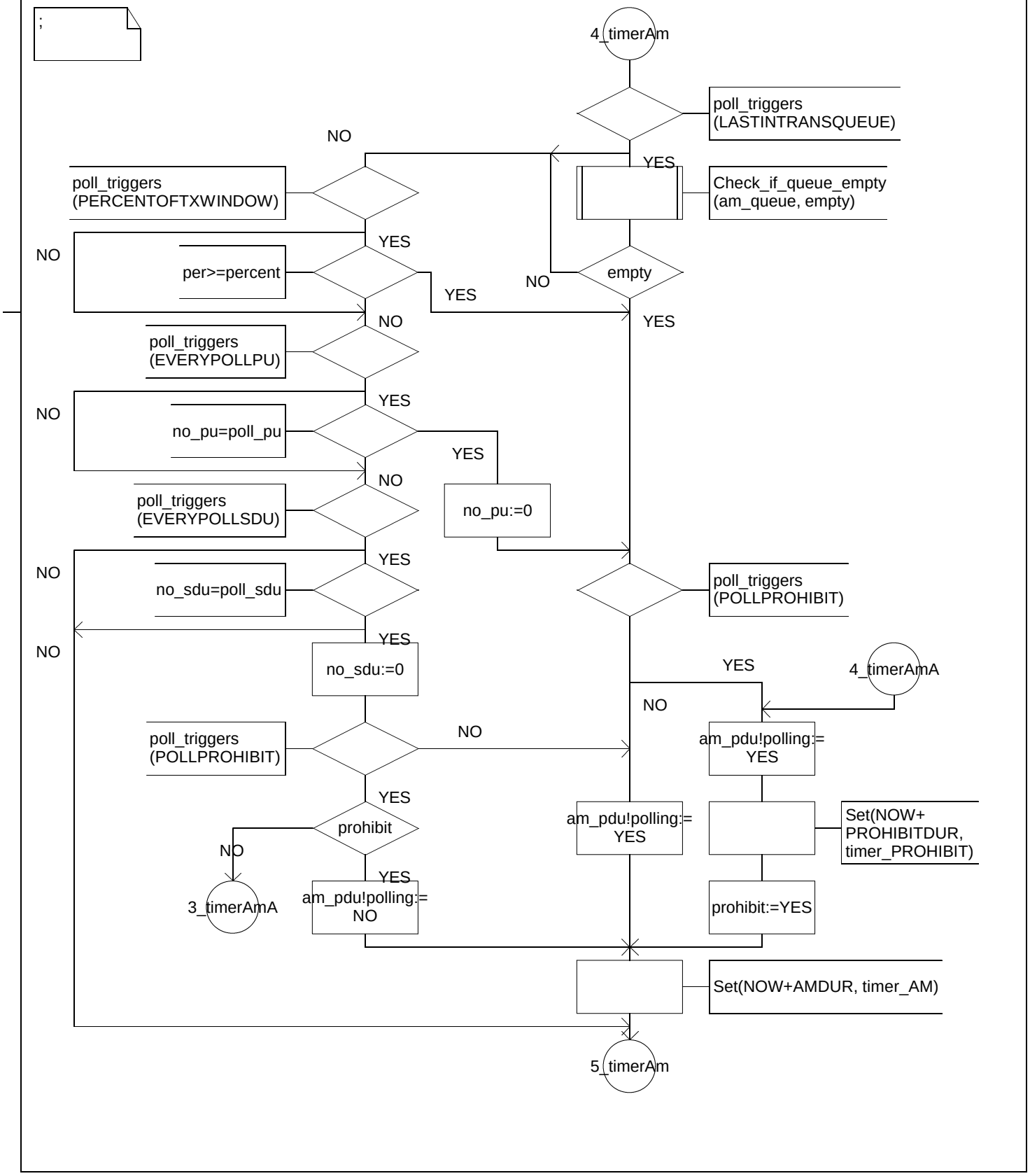
;



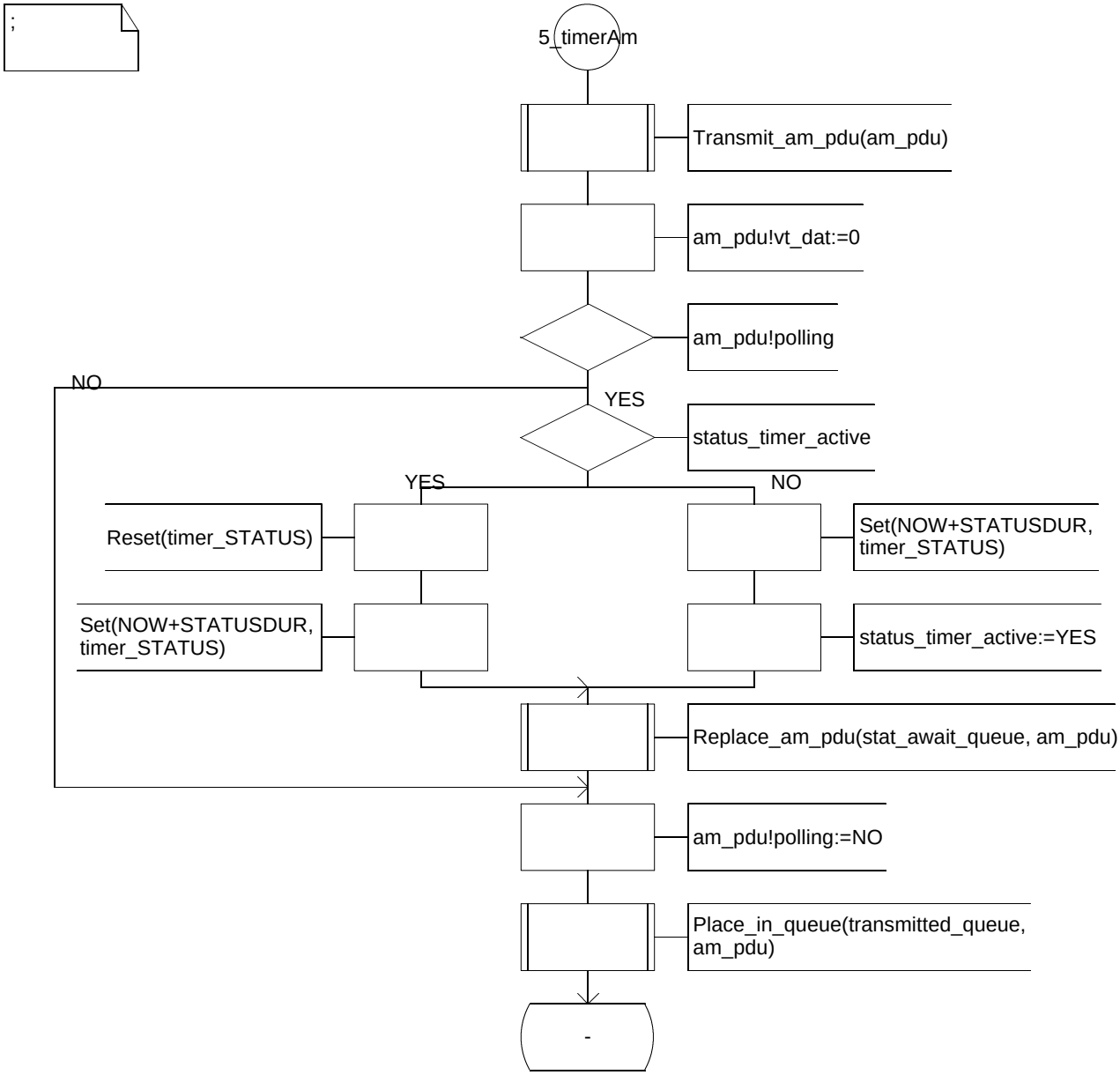
Virtual Process Type Acknowledged_connection 3_AcknowledgedDataTransferReady_timerAm(44)



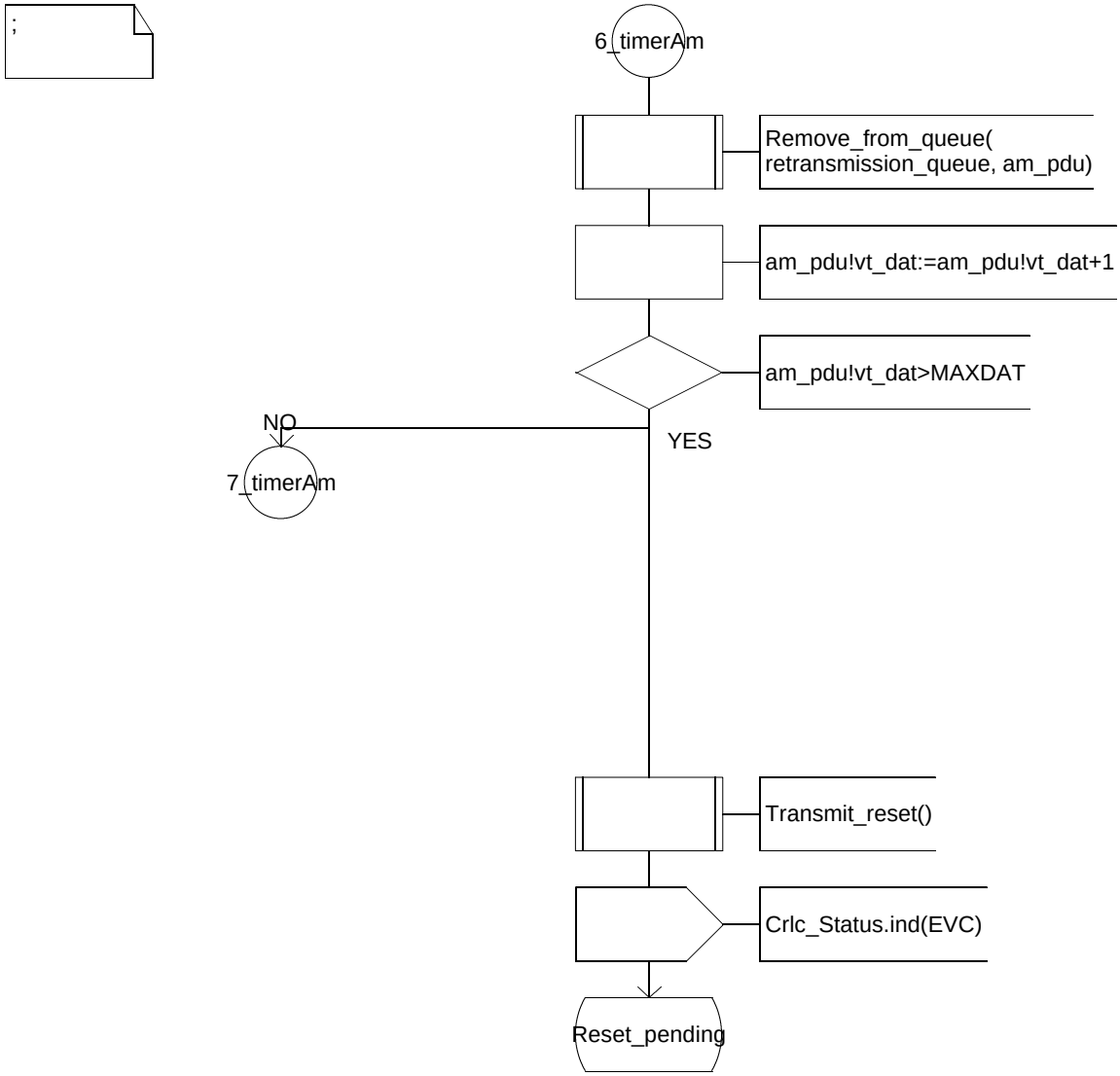
Virtual Process Type Acknowledged_connection 4_AcknowledgedDataTransferReady_timerAm(44)



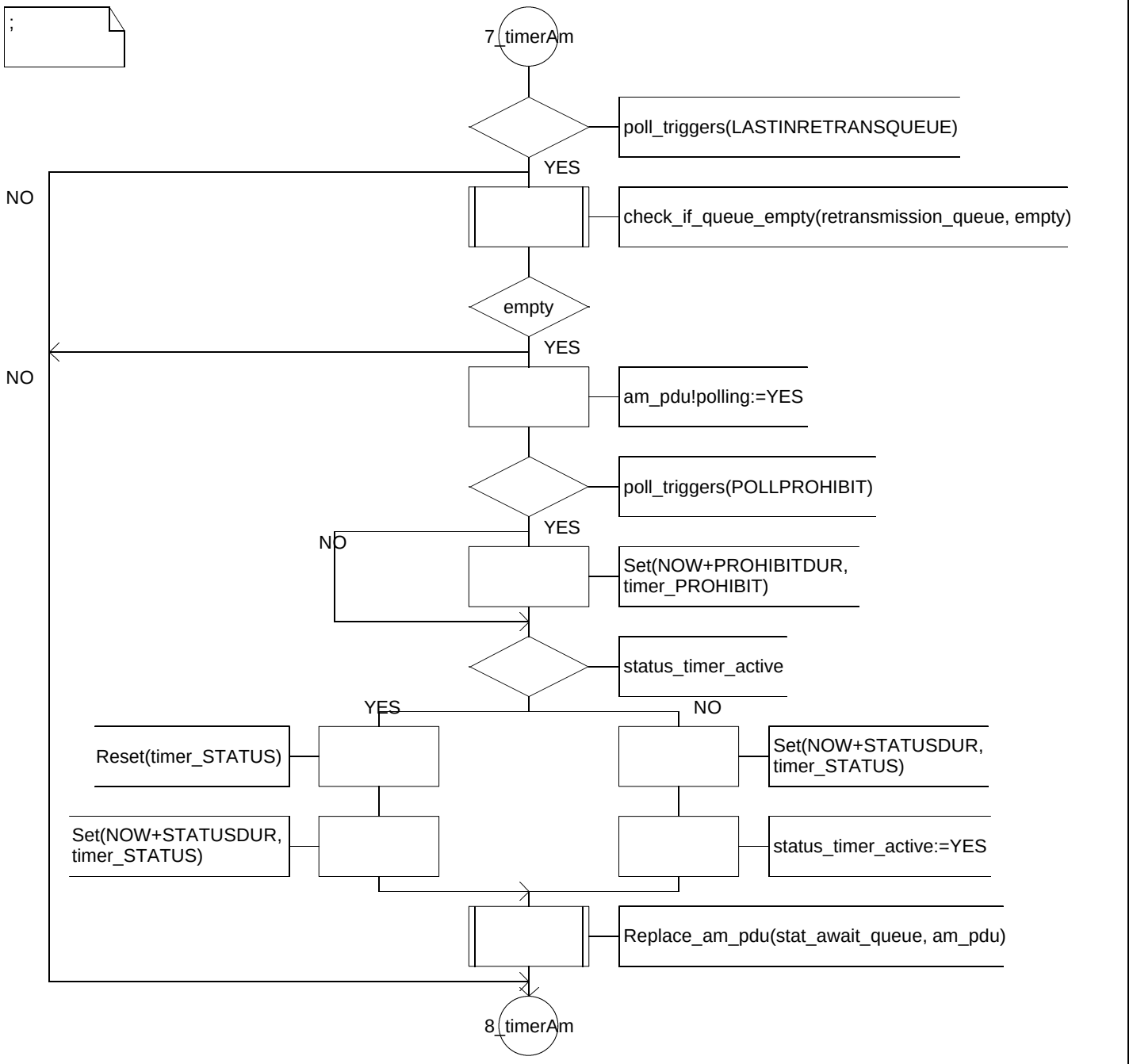
Virtual Process Type Acknowledged_connection 5_AcknowledgedDataTransferReady_timerAm(44)



Virtual Process Type Acknowledged_connection 6_AcknowledgedDataTransferReady_timerAm(44)

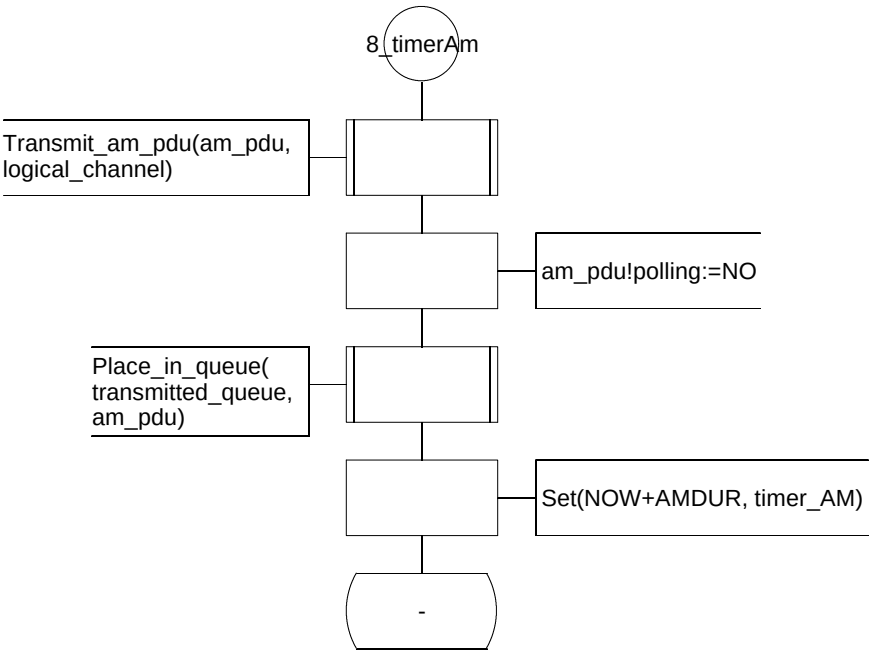


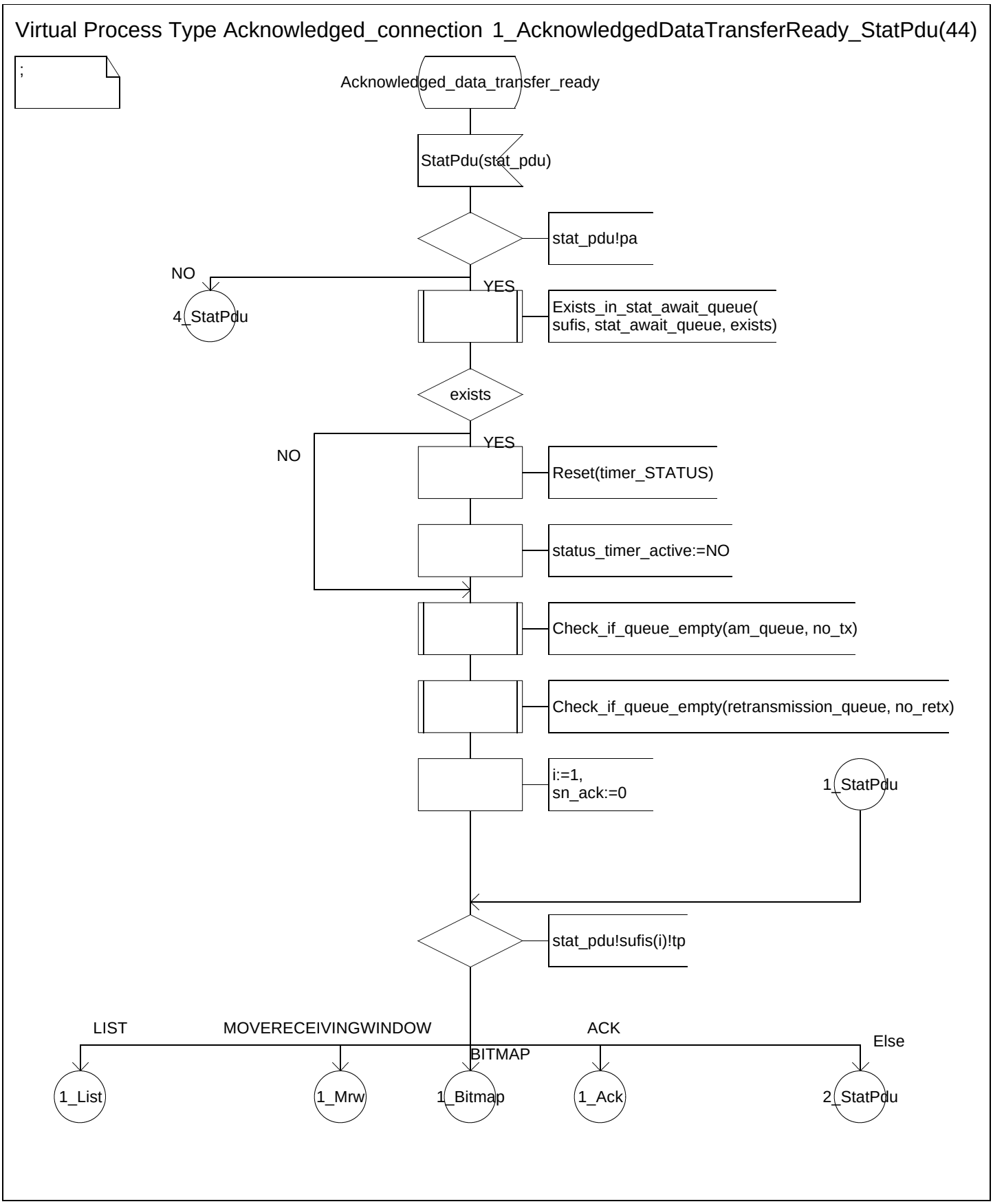
Virtual Process Type Acknowledged_connection 7_AcknowledgedDataTransferReady_timerAm(44)



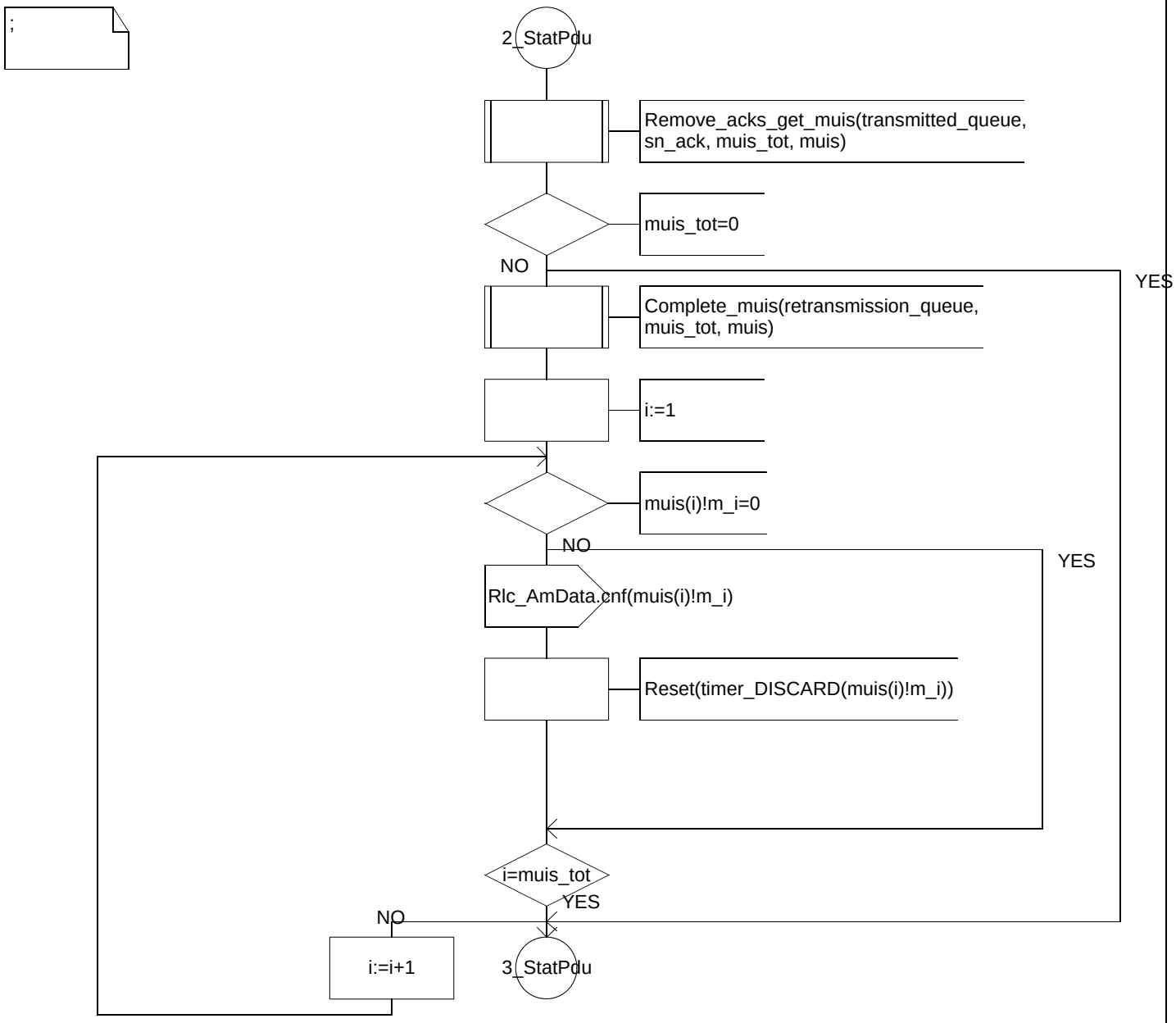
Virtual Process Type Acknowledged_connection 8_AcknowledgedDataTransferReady_timerAm(44)

;

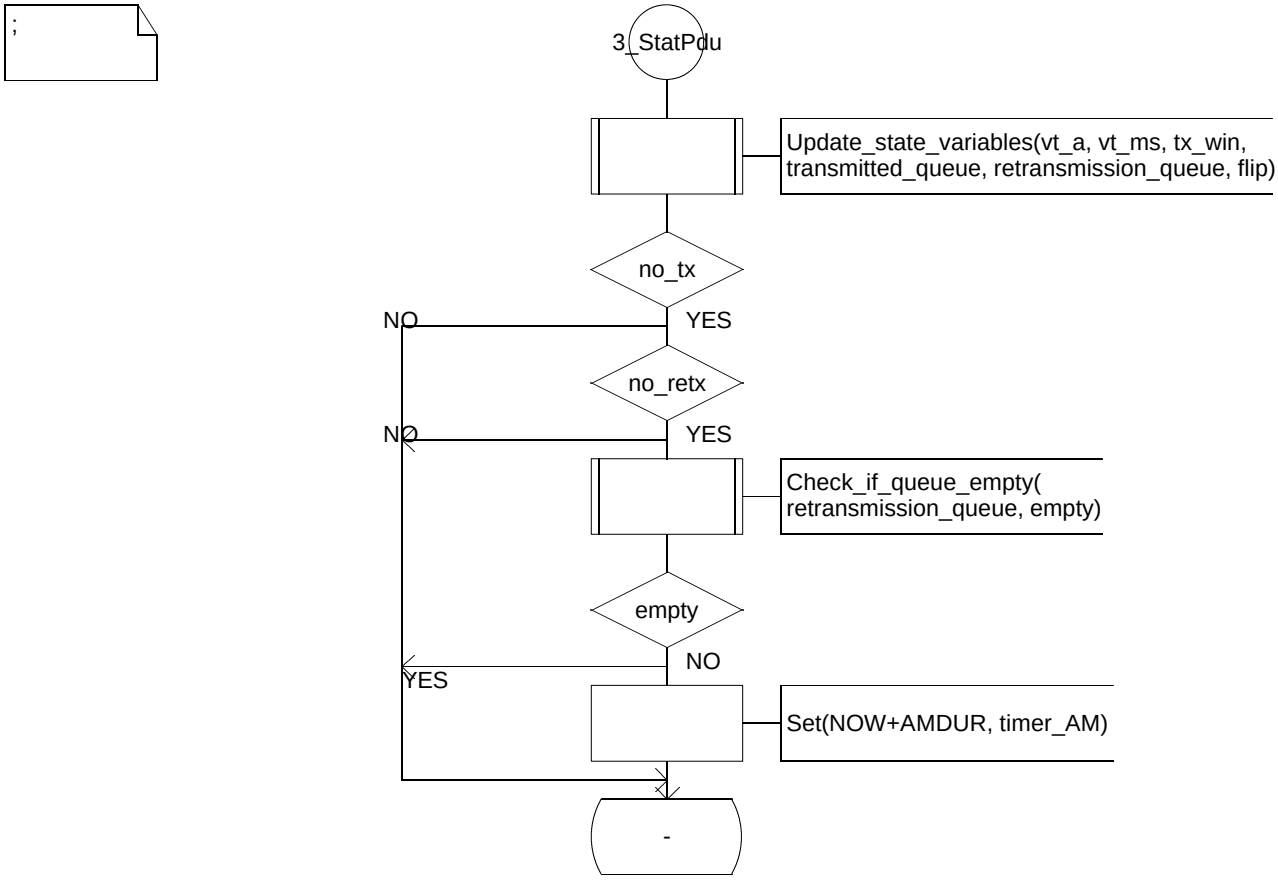



Virtual Process Type Acknowledged_connection 1_AcknowledgedDataTransferReady_StatPdu(44)
;
Acknowledged_data_transfer_ready
StatPdu(stat_pdu)
stat_pdu!pa
NO
YES
4_StatPdu
Exists_in_stat_await_queue(
sufis, stat_await_queue, exists)
exists
NO
YES
Reset(timer_STATUS)
status_timer_active:=NO
Check_if_queue_empty(am_queue, no_tx)
Check_if_queue_empty(retransmission_queue, no_retx)
i:=1,
sn_ack:=0
1_StatPdu
stat_pdu!sufis(i)!tp
LIST
MOVERECEIVINGWINDOW
ACK
BITMAP
Else
1_List
1_Mrw
1_Bitmap
1_Ack
2_StatPdu

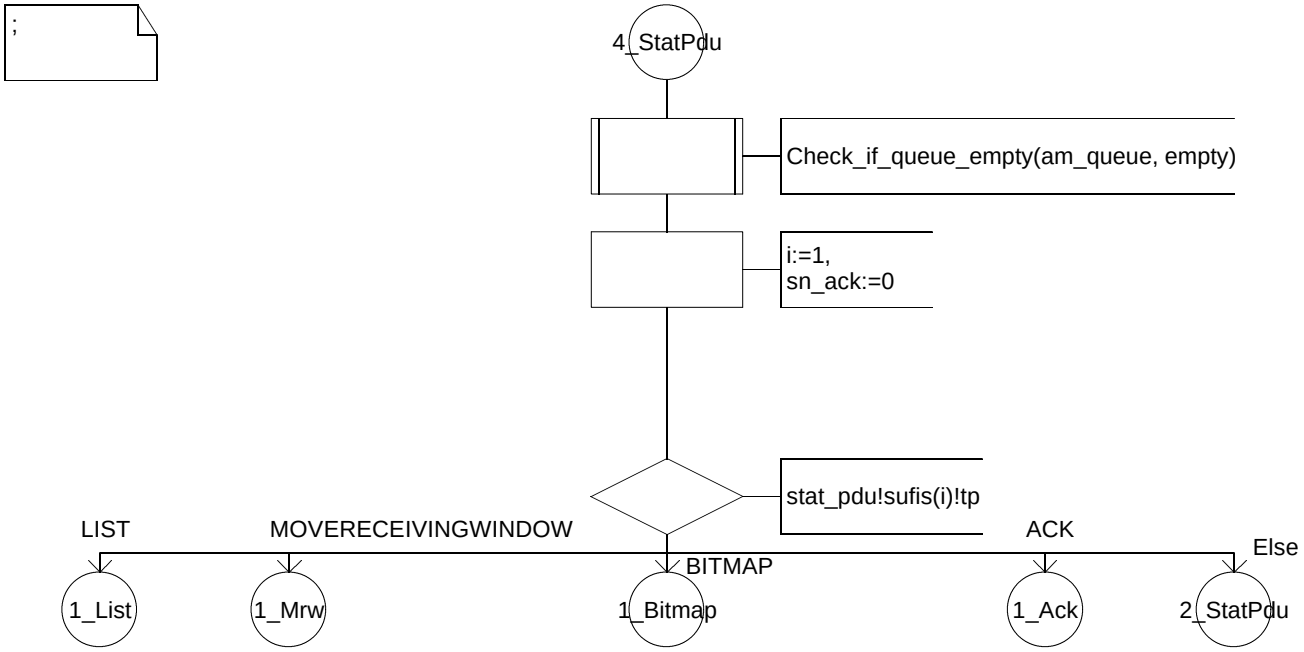
Virtual Process Type Acknowledged_connection 2_AcknowledgedDataTransferReady_StatPdu(44)



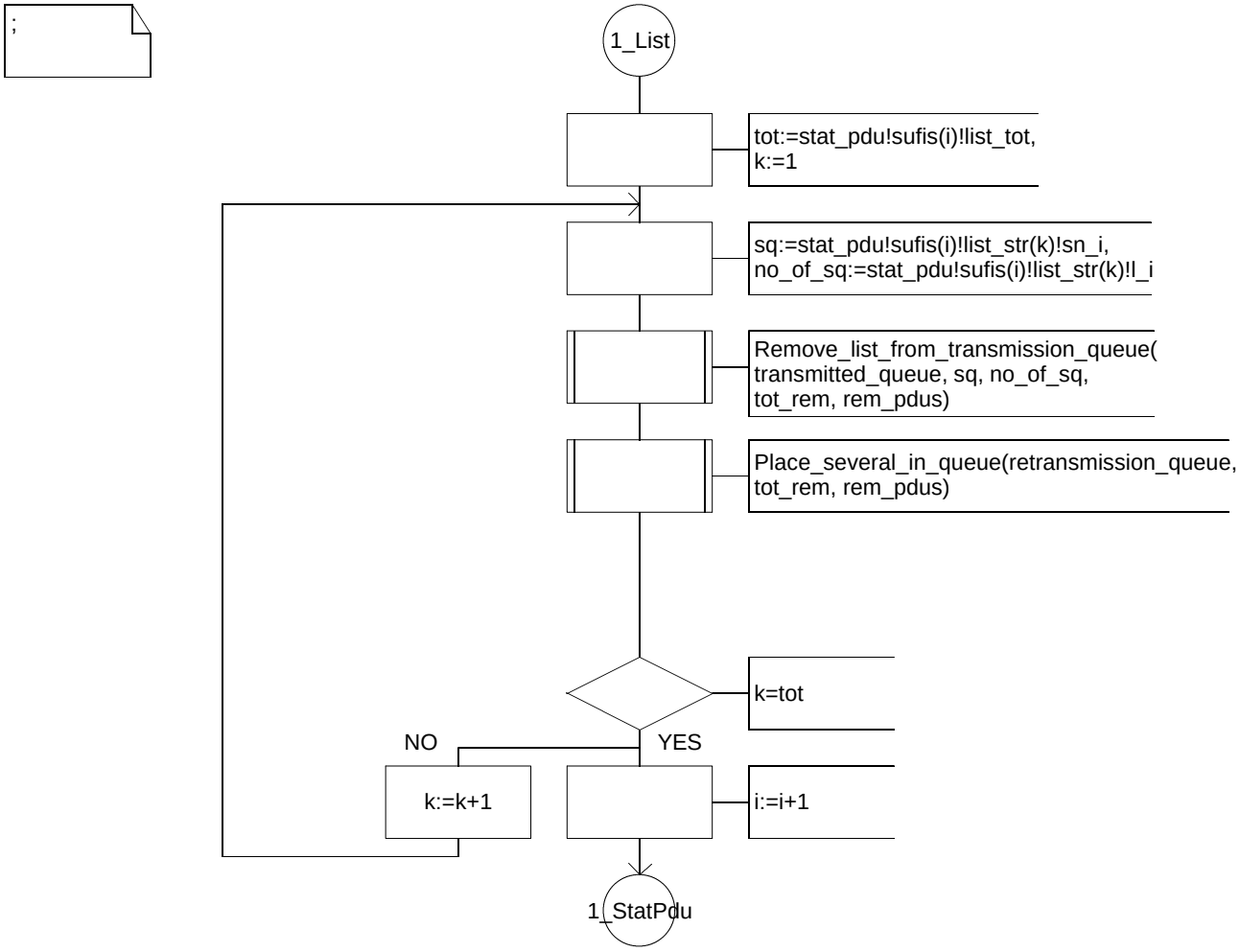
Virtual Process Type Acknowledged_connection 3_AcknowledgedDataTransferReady_StatPdu(44)



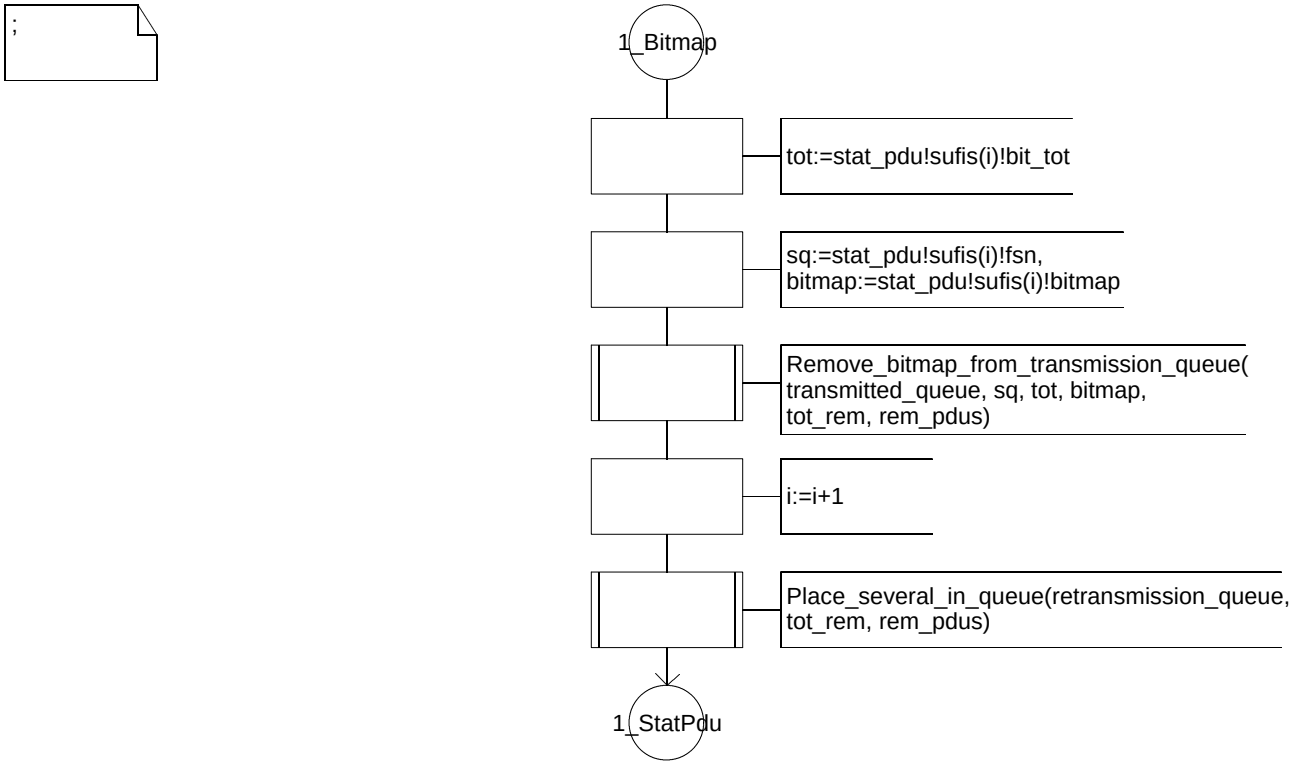
Virtual Process Type Acknowledged_connection 4_AcknowledgedDataTransferReady_StatPdu(44)



Virtual Process Type Acknowledged_connecti1_AcknowledgedDataTransferReady_StatPduList(44)

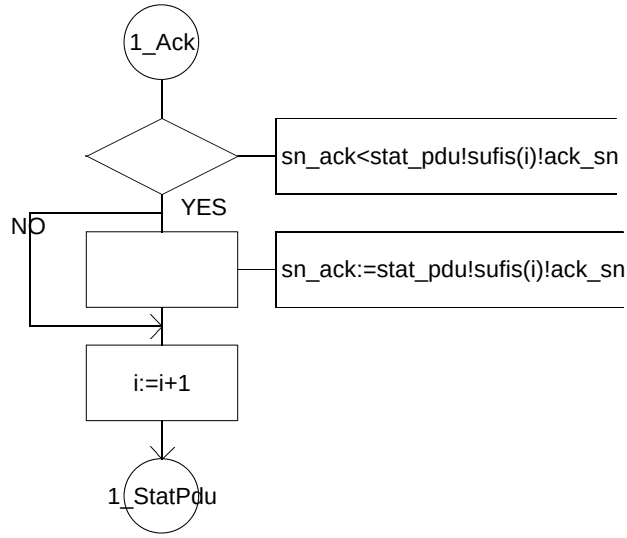


Virtual Process Type Acknowledged_conn1_AcknowledgedDataTransferReady_StatPduBitmap(44)

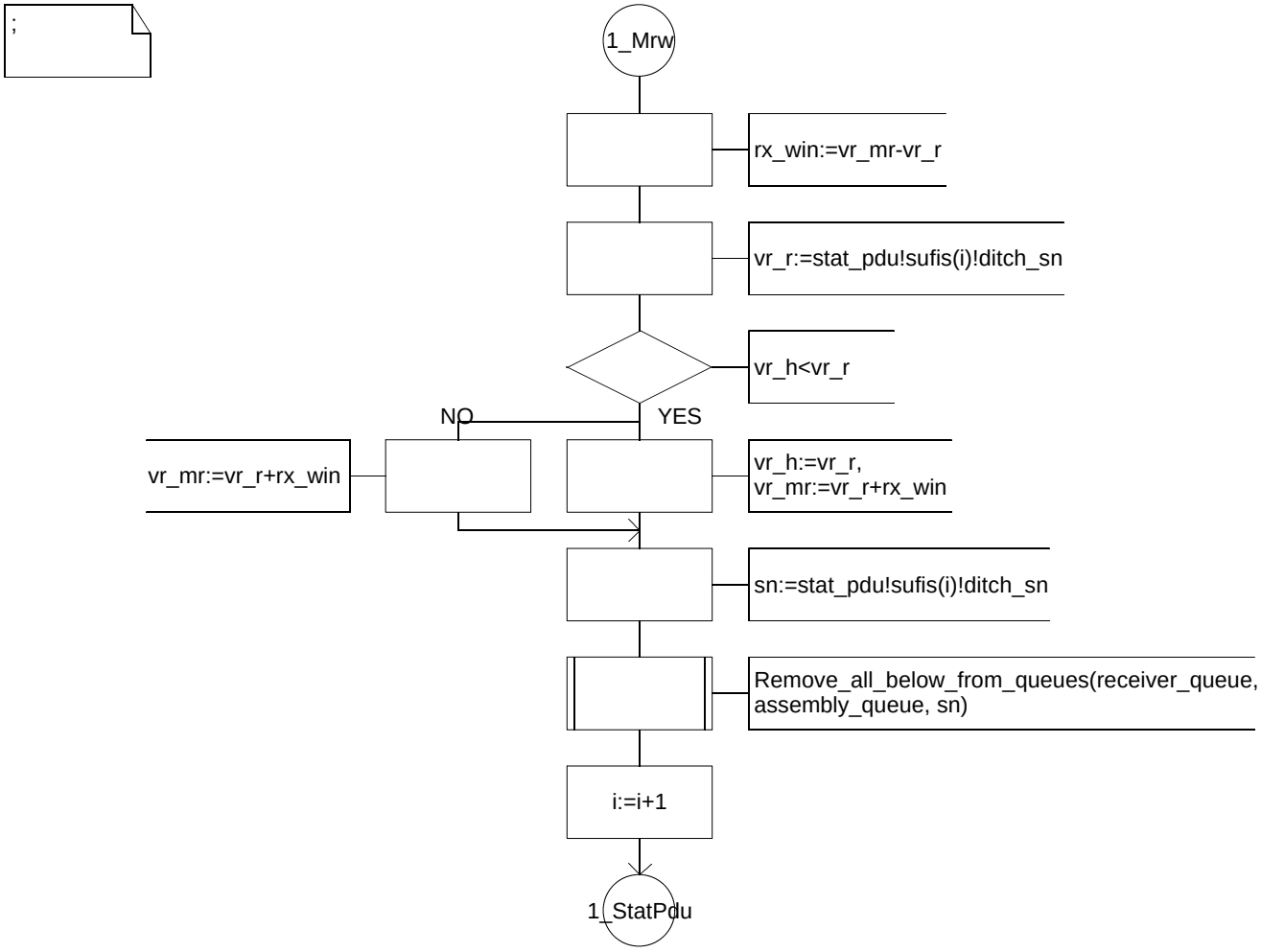


Virtual Process Type Acknowledged_connecti1_AcknowledgedDataTransferReady_StatPduAck(44)

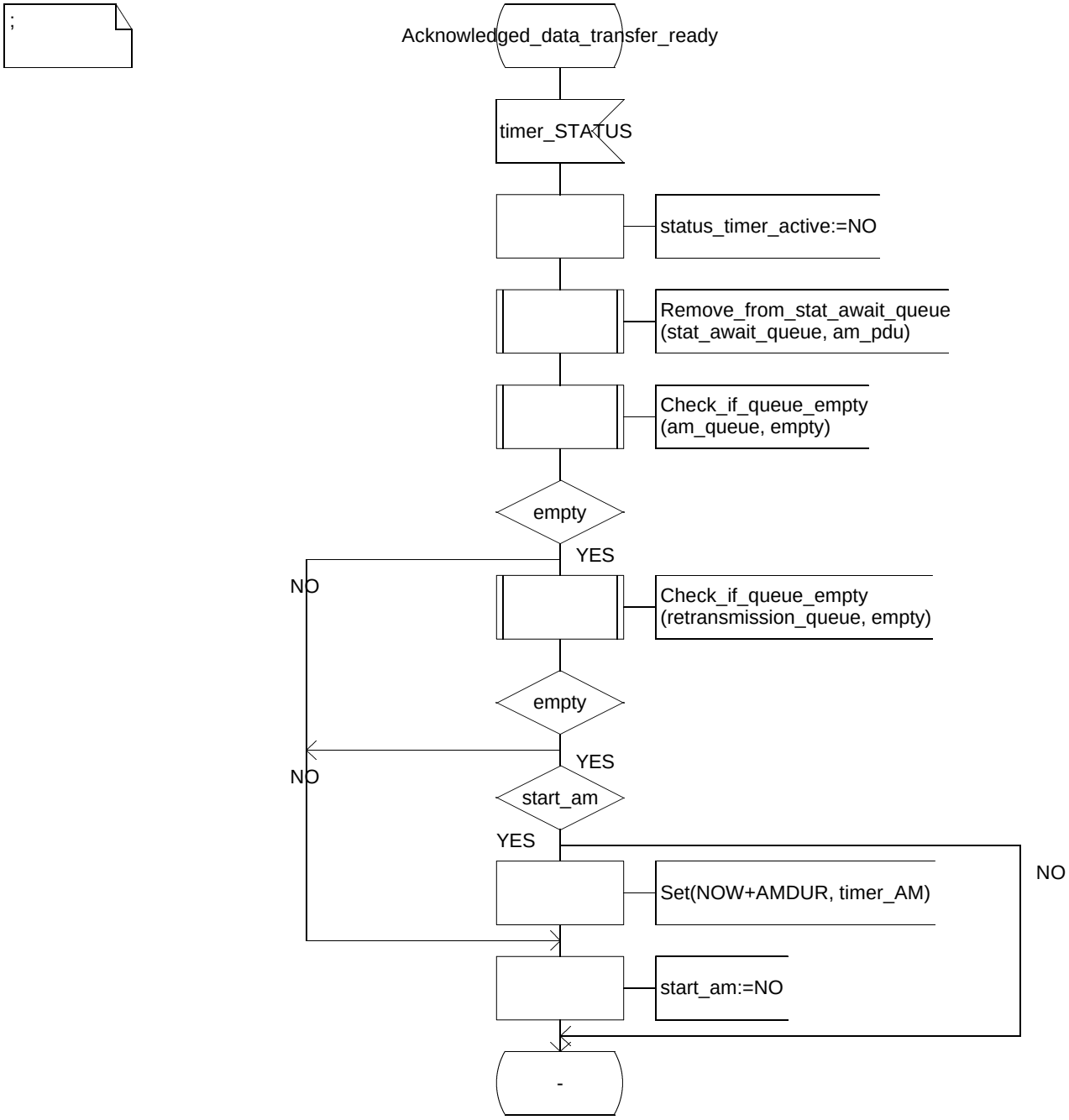
;



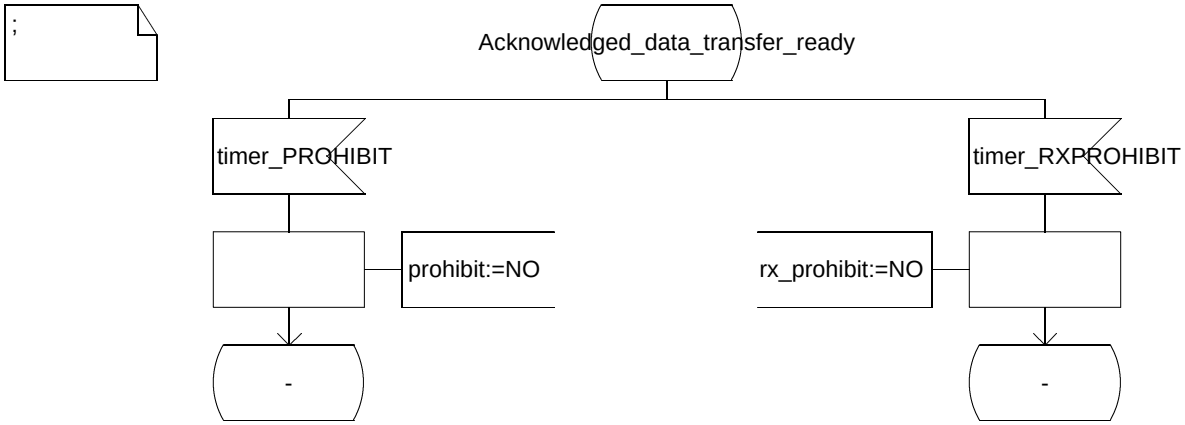
Virtual Process Type Acknowledged_connect1_AcknowledgedDataTransferReady_StatPduMrw(44)



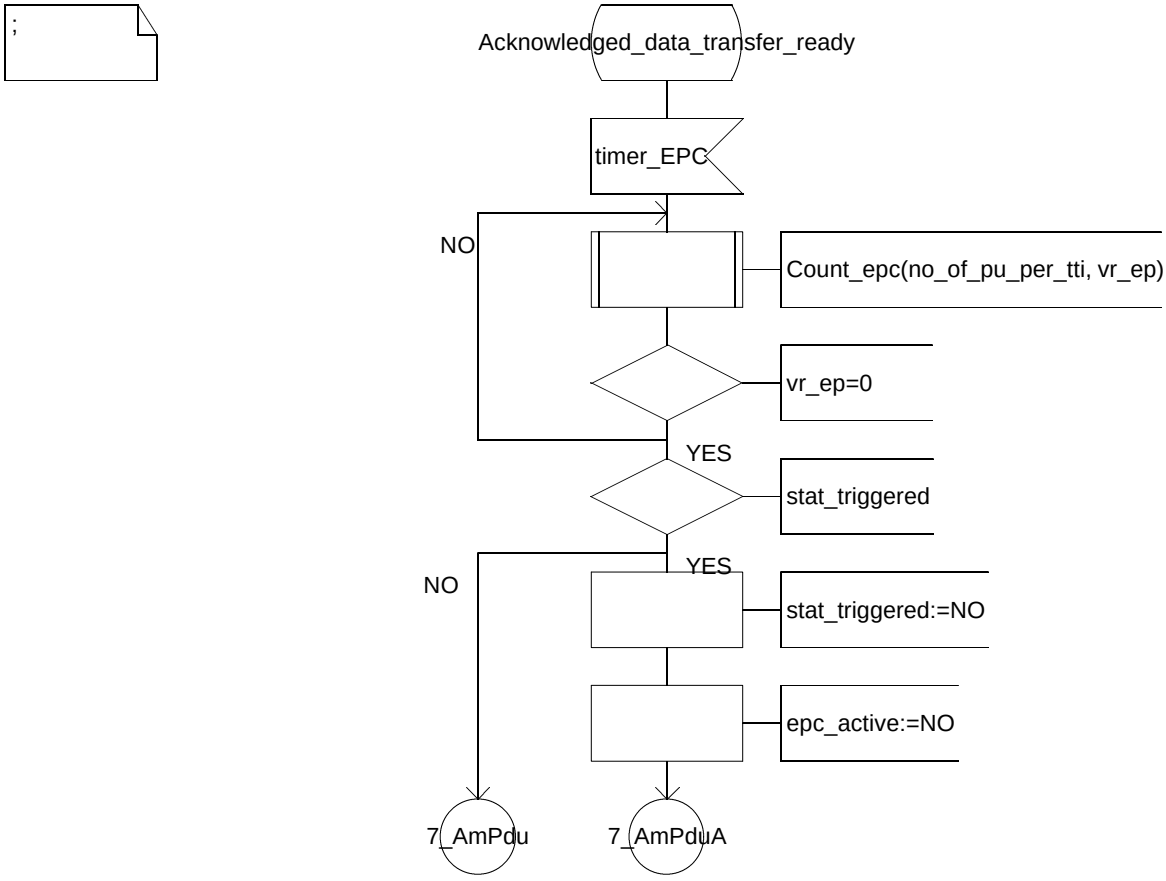
Virtual Process Type Acknowledged_connect1_AcknowledgedDataTransferReady_TimerStatus(44)



Virtual Process Type Acknowledged_conne1_AcknowledgedDataTransferReady_TimerProhibit(44)

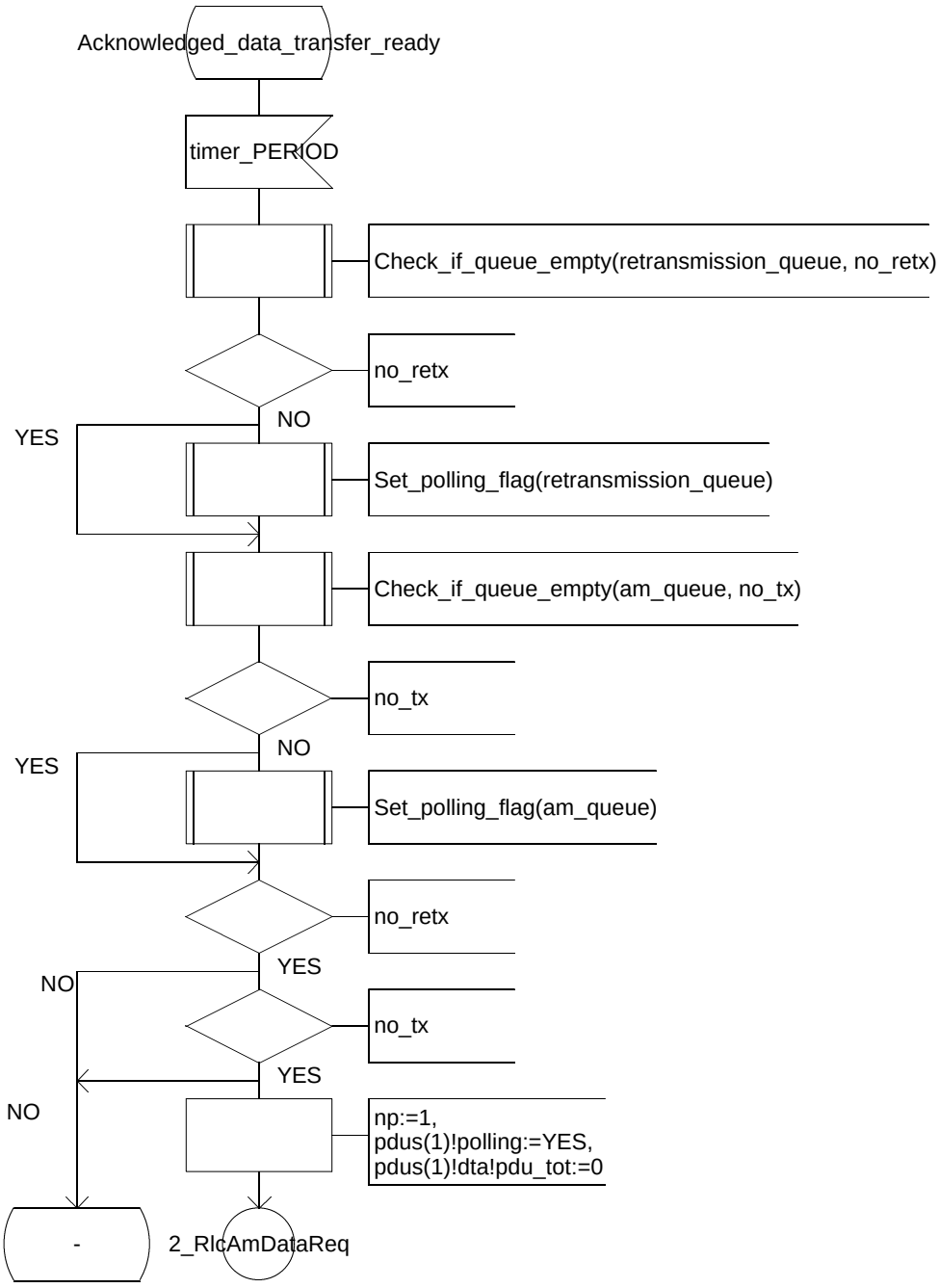


Virtual Process Type Acknowledged_connection1_AcknowledgedDataTransferReady_timerEpc(44)



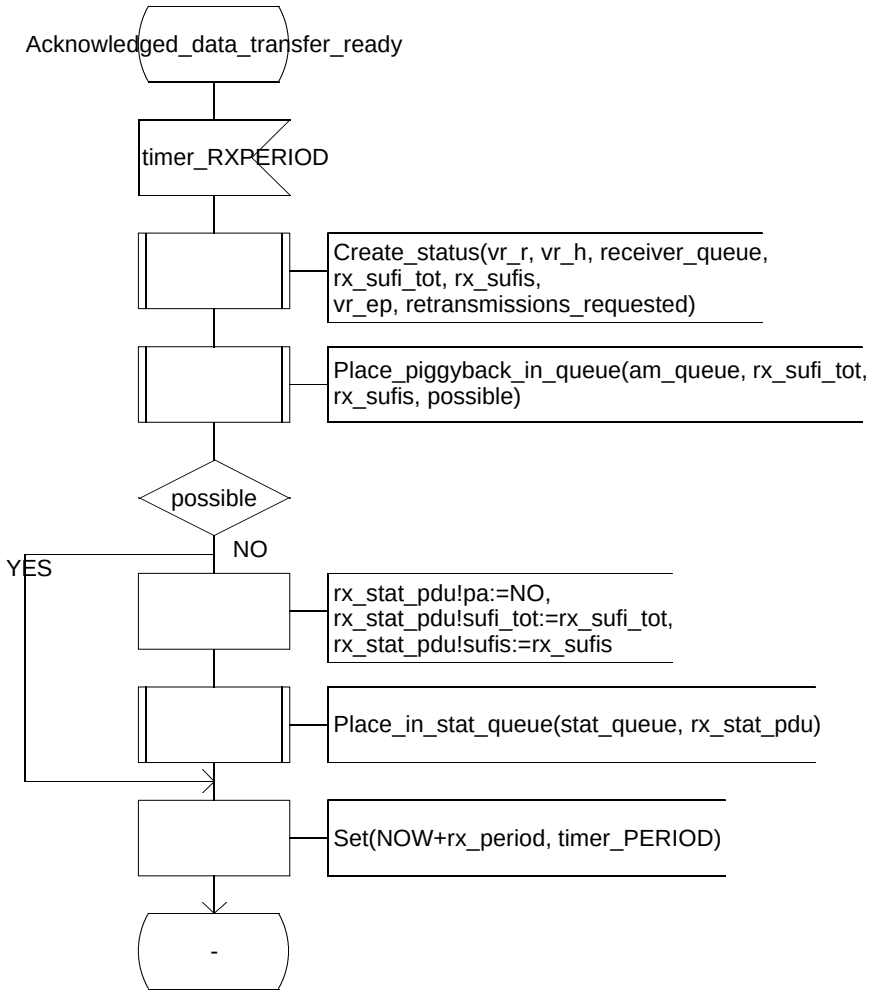
Virtual Process Type Acknowledged_connecti1_AcknowledgedDataTransferReady_timerPeriod(44)

;

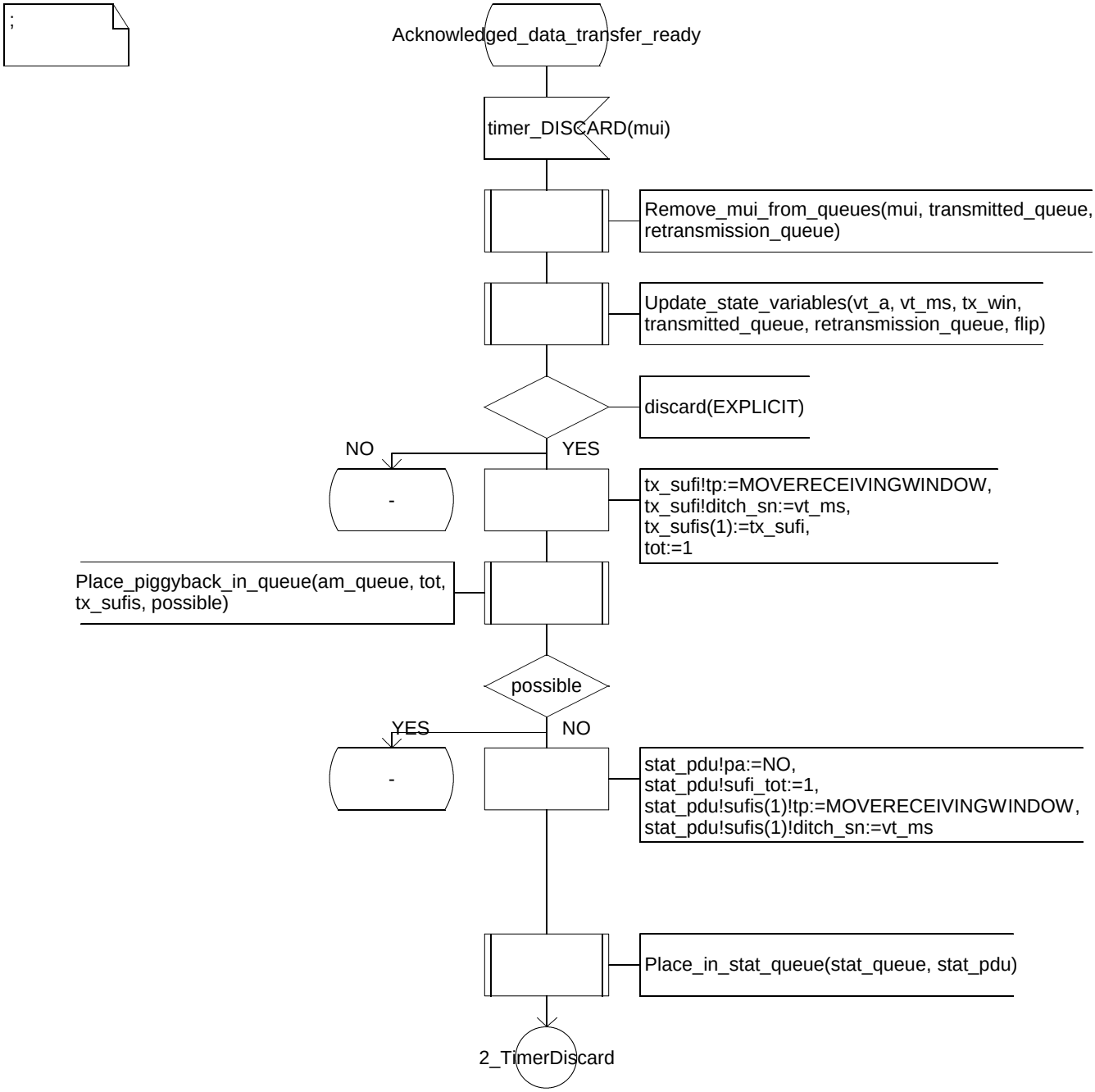


Virtual Process Type Acknowledged_conne1_AcknowledgedDataTransferReady_timerRxPeriod(44)

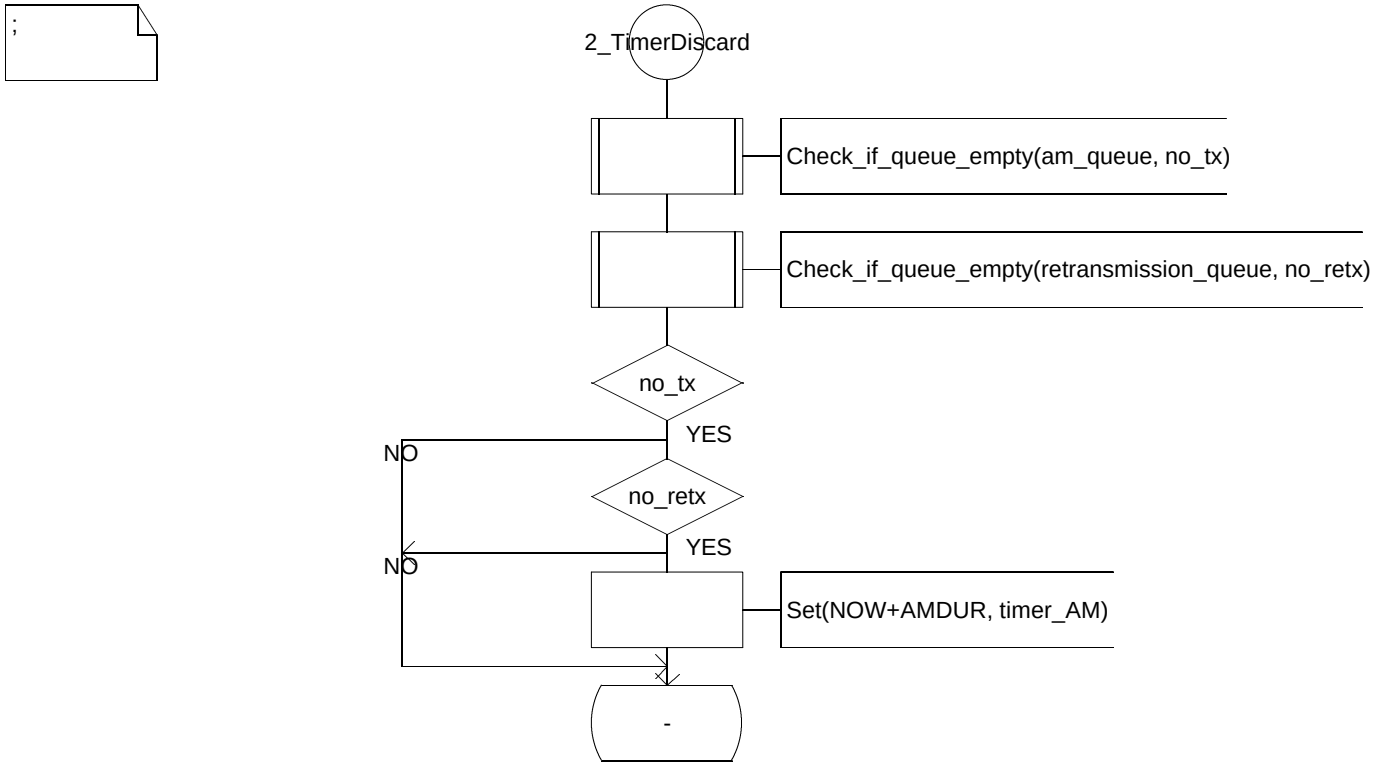
;



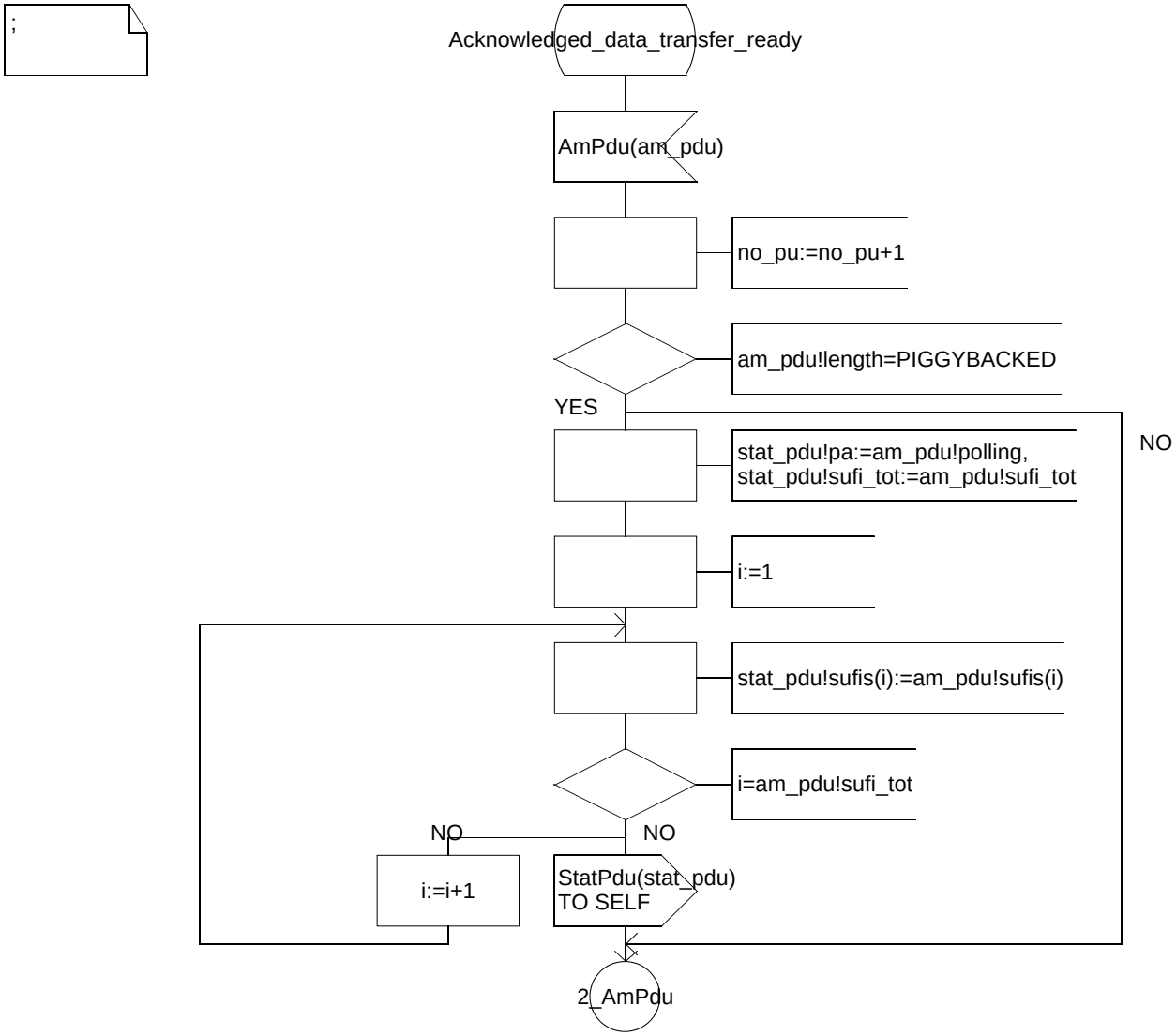
Virtual Process Type Acknowledged_conned_AcknowledgedDataTransferReady_TimerDiscard(44)



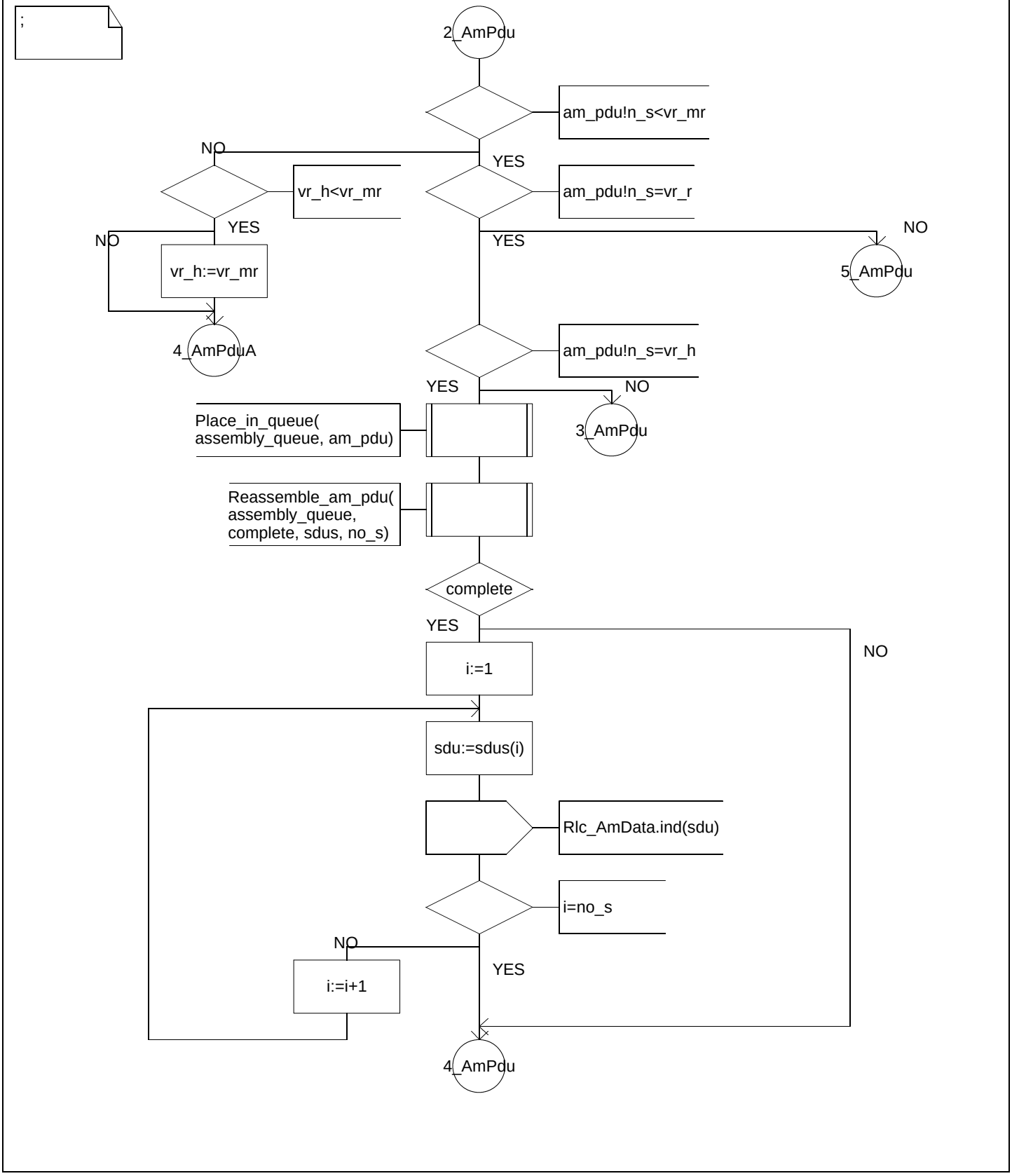
Virtual Process Type Acknowledged_conne2_AcknowledgedDataTransferReady_TimerDiscard(44)



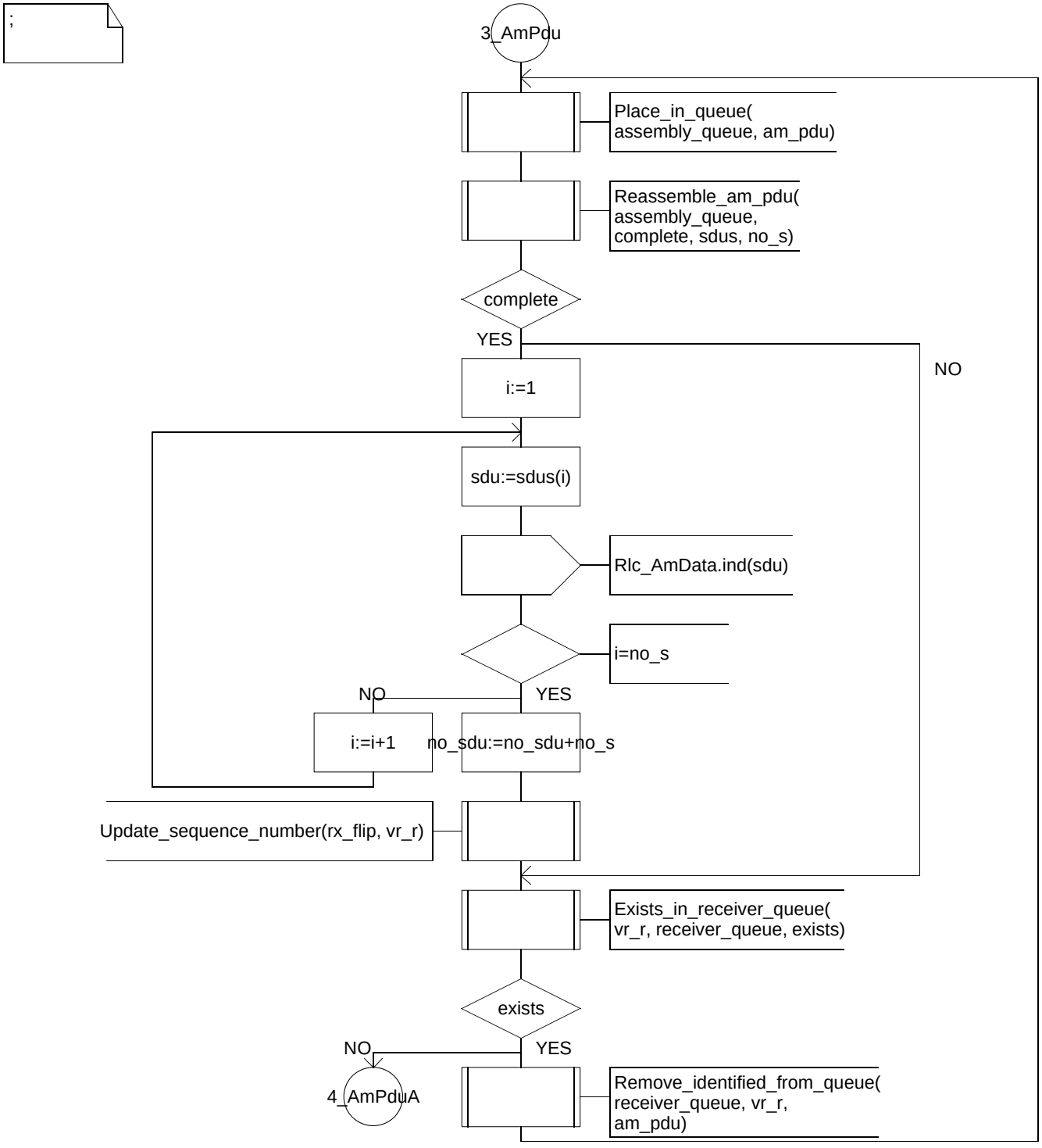
Virtual Process Type Acknowledged_connection 1_AcknowledgedDataTransferReady_AmPdu(44)



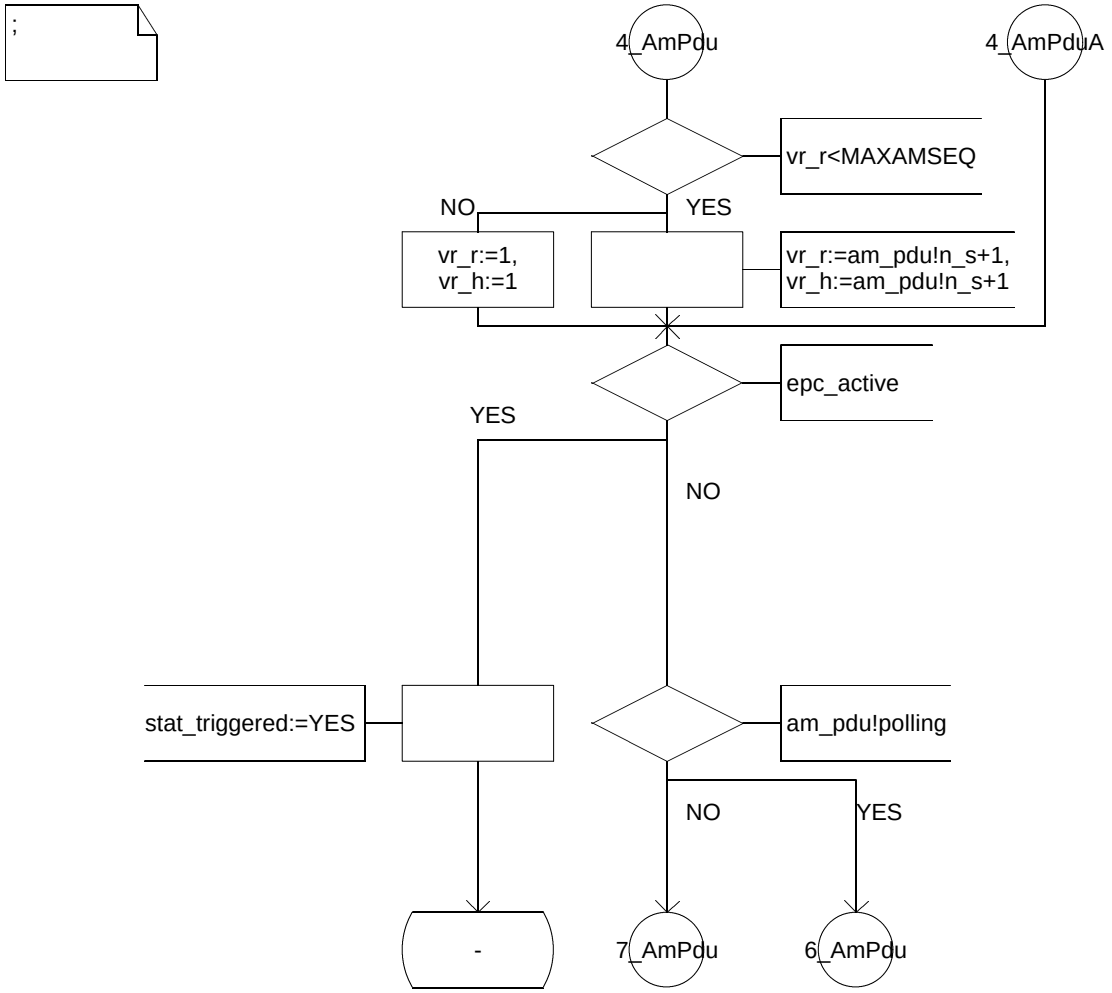
Virtual Process Type Acknowledged_connection 2_AcknowledgedDataTransferReady_AmPdu(44)



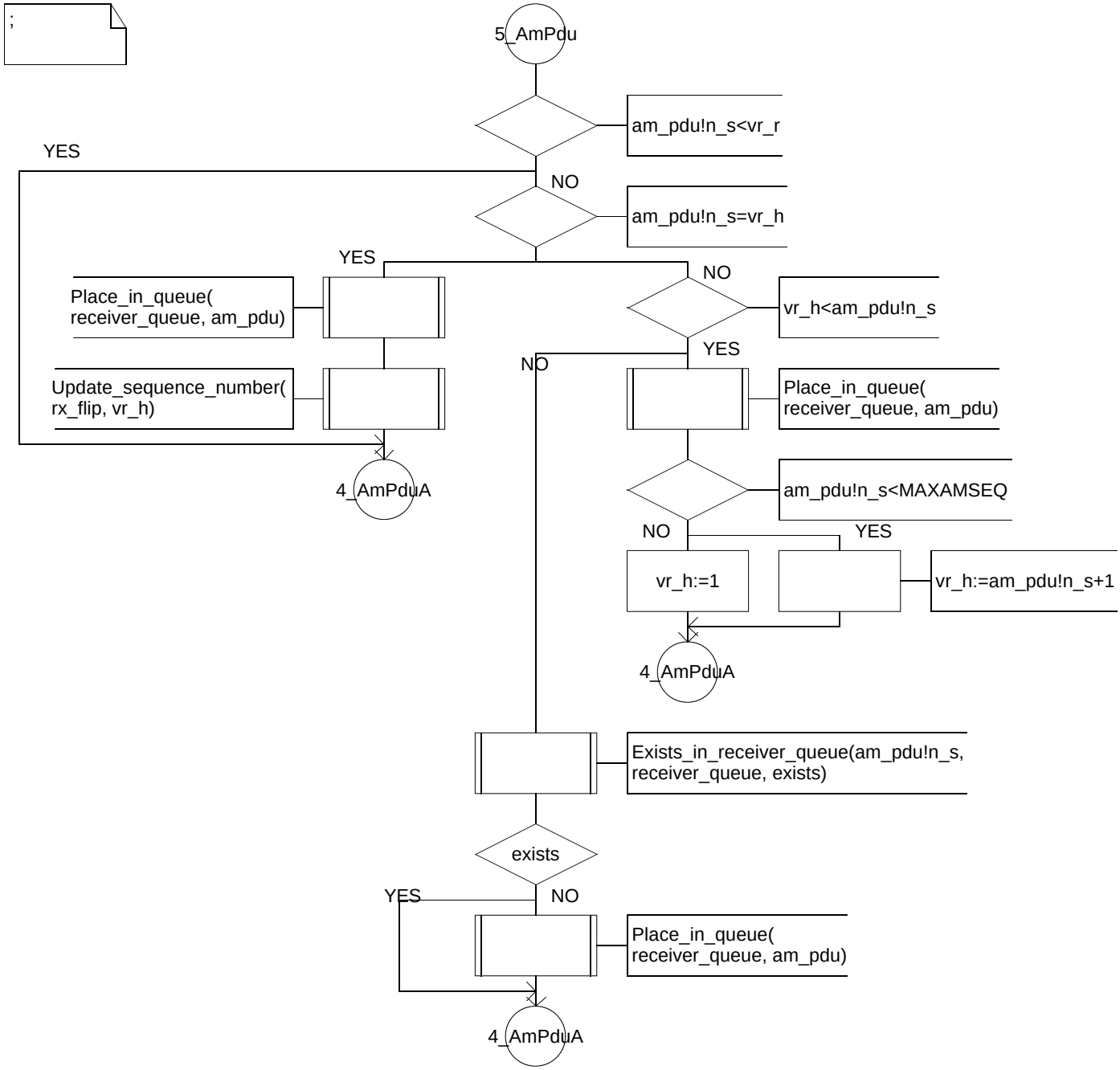
Virtual Process Type Acknowledged_connection 3_AcknowledgedDataTransferReady_AmPdu(44)



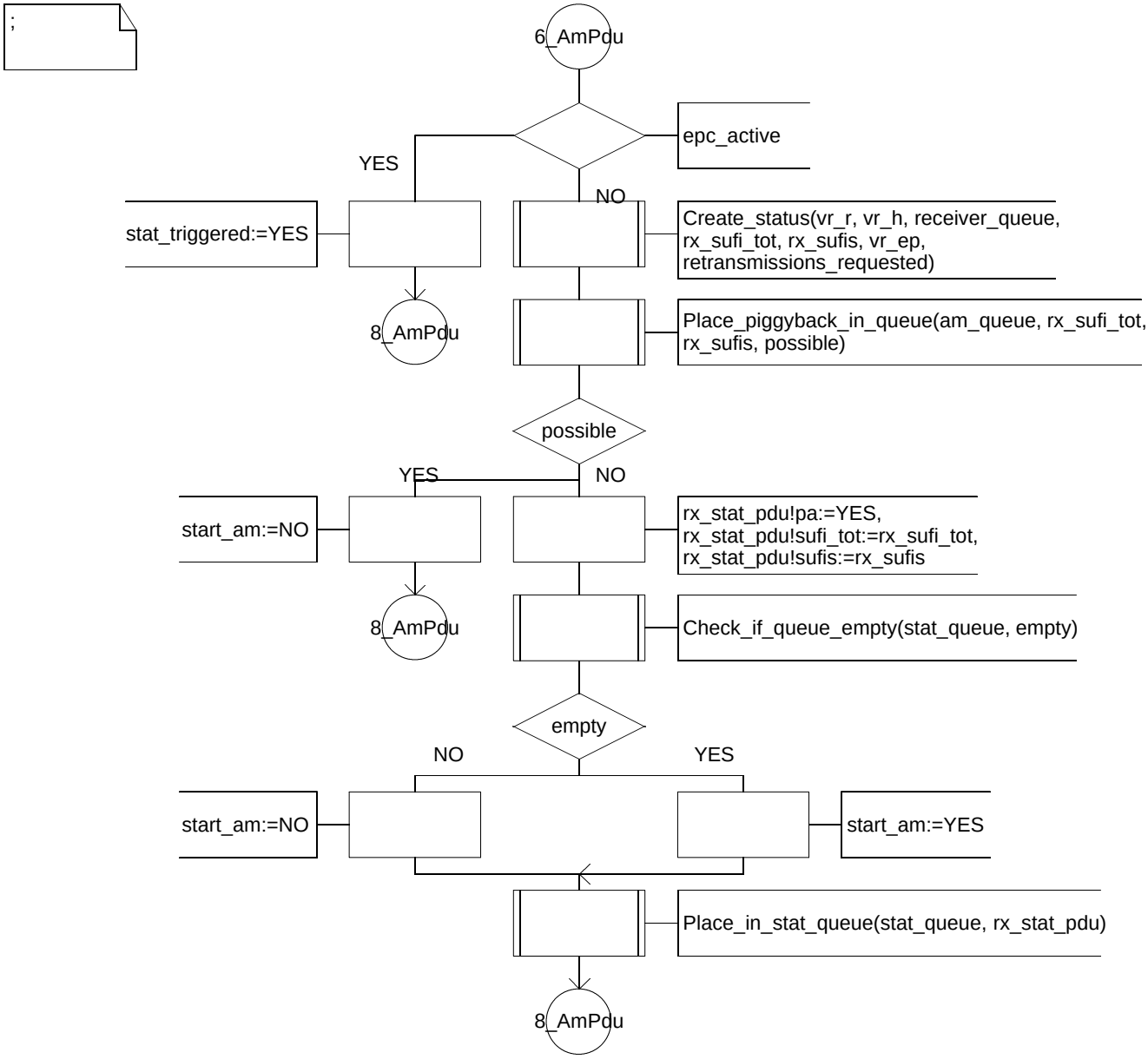
Virtual Process Type Acknowledged_connection 4_AcknowledgedDataTransferReady_AmPdu(44)



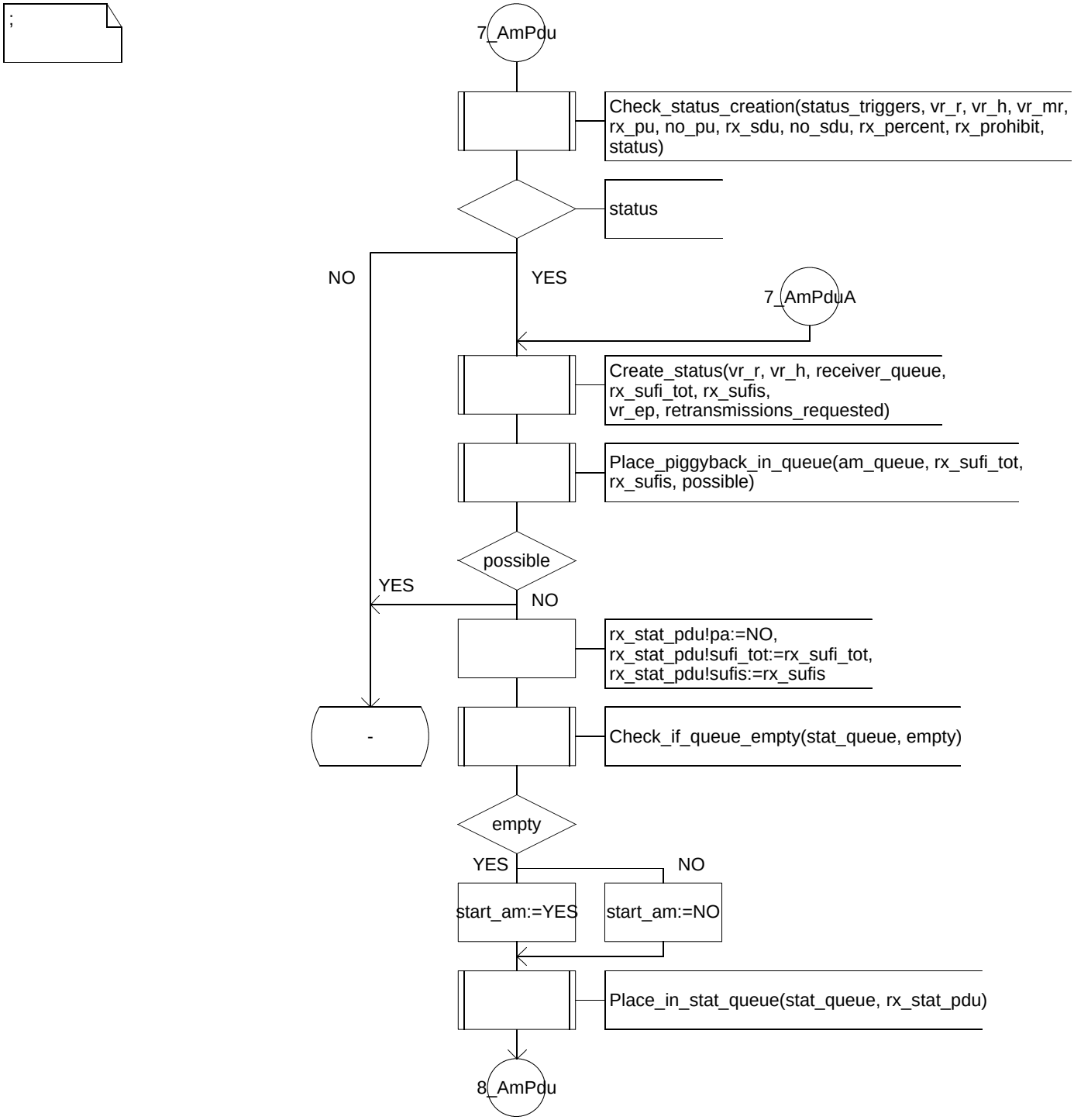
Virtual Process Type Acknowledged_connection 5_AcknowledgedDataTransferReady_AmPdu(44)



Virtual Process Type Acknowledged_connection 6_AcknowledgedDataTransferReady_AmPdu(44)

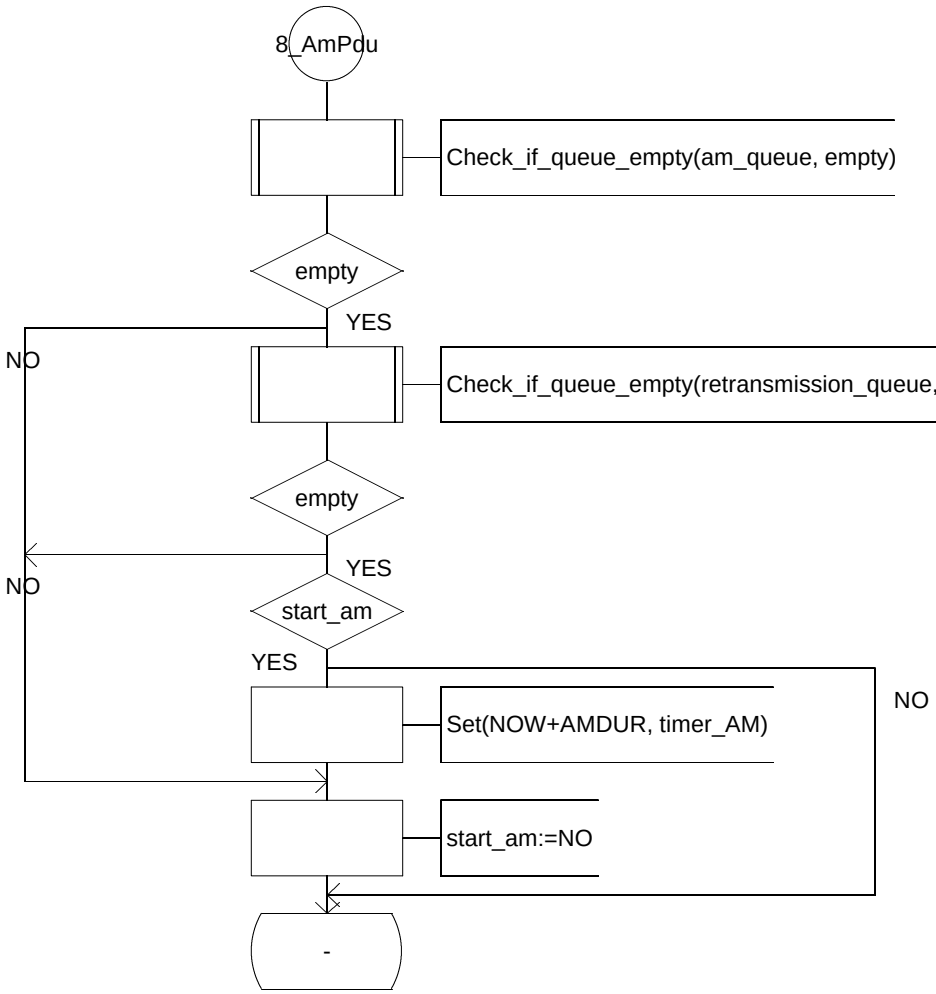


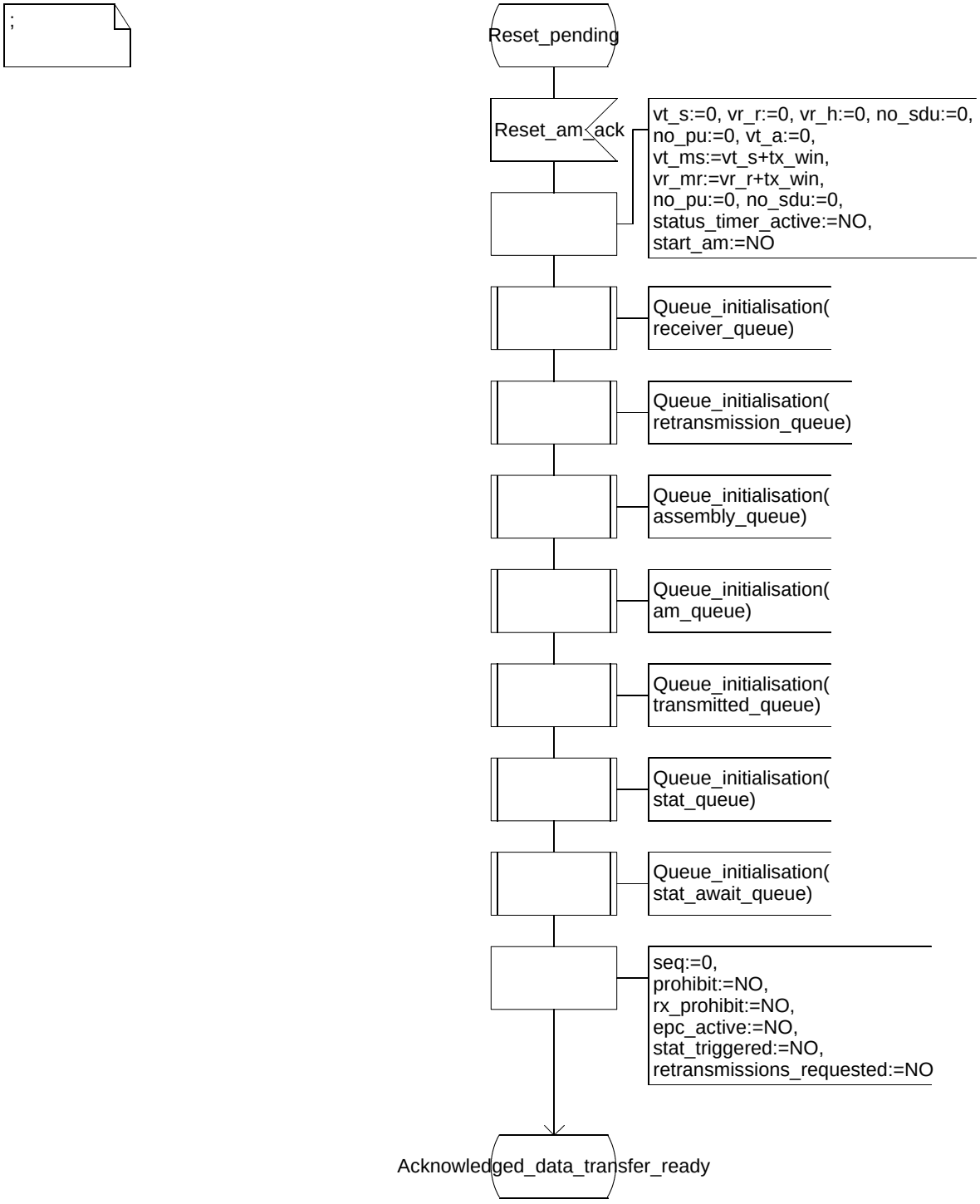
Virtual Process Type Acknowledged_connection 7_AcknowledgedDataTransferReady_AmPdu(44)



Virtual Process Type Acknowledged_connec8_AcknowledgedModeDataTransferReady_AmPdu(44)

;




;
Reset_pending
Reset_am_ack
vt_s:=0, vr_r:=0, vr_h:=0, no_sdu:=0,
no_pu:=0, vt_a:=0,
vt_ms:=vt_s+tx_win,
vr_mr:=vr_r+tx_win,
no_pu:=0, no_sdu:=0,
status_timer_active:=NO,
start_am:=NO
Queue_initialisation(
receiver_queue)
Queue_initialisation(
retransmission_queue)
Queue_initialisation(
assembly_queue)
Queue_initialisation(
am_queue)
Queue_initialisation(
transmitted_queue)
Queue_initialisation(
stat_queue)
Queue_initialisation(
stat_await_queue)
seq:=0,
prohibit:=NO,
rx_prohibit:=NO,
epc_active:=NO,
stat_triggered:=NO,
retransmissions_requested:=NO
Acknowledged_data_transfer_ready