Technical Specification Group, Radio Access Network
Meeting #5, Korea, 6 - 8 October 1999

TSGR#5(99)465

Source:

Title: **TS 25.322: Description of RLC protocol**

Document for:

Agenda Item: **6.3.3**

3G TS RAN 25.322 V1.3.0 (1999-09)

Technical Specification

3rd Generation Partnership Project (3GPP); Technical Specification Group (TSG) RAN; Working Group 2 (WG2);

RLC Protocol Specification (3G TS 25.322 version 1.3.0)

Reference

<Workitem> (<Shortfilename>.PDF)

Keywords

Digital cellular telecommunications system,
Universal Mobile Telecommunication System
(UMTS), UTRA, IMT-2000

3GPP

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present deliverable may have been declared to ETSI/3GPP. The information pertaining to these essential IPRs, if any, is publicly available for ETSI members and non-members, free of charge. This can be found in the latest version of the ETSI Technical Report: ETR 314: "Intellectual Property Rights (IPRs); Essential or potentially Essential, IPRs notified to ETSI in respect of ETSI standards". The most recent update of ETR 314, is available on the ETSI web server or on request from the Secretariat.

Pursuant to the ETSI Interim IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in the ETR 314, which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification has been produced by the 3GPP.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of this TS, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version 3.y.z
where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 Indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the specification;

1. Scope

The scope of this specification is to specify the RLC protocol.

2. References

[1] 3GPP TS 25.401: “RAN Overall Description
[2] 3GPP TR 25.945: “Vocabulary for the UTRAN”
[3] 3GPP TS 25.301: “Radio Interface Protocol Architecture”
[4] 3GPP TS 25.302: “Services Provided by the Physical Layer”
[5] 3GPP TS 25.303: “UE Functions and Inter-Layer Procedures in Connected Mode”
[6] 3GPP TS 25.304: “UE Procedures in Idle Mode”
[7] 3GPP TS 25.321: “MAC Protocol Specification”
[8] 3GPP TS.25.331: “RRC Protocol Specification”

3. Definitions and Abbreviations

ARQ	Automatic Repeat Request
BCCH	Broadcast Control Channel
BCH	Broadcast Channel
C-	Control-
CC	Call Control
CCCH	Common Control Channel
CCH	Control Channel
CCTrCH	Coded Composite Transport Channel
CN	Core Network

CRC	Cyclic Redundancy Check
DC	Dedicated Control (SAP)
DCCH	Dedicated Control Channel
DCH	Dedicated Channel
DL	Downlink
DSCH	Downlink Shared Channel
DTCH	Dedicated Traffic Channel
FACH	Forward Link Access Channel
FCS	Frame Check Sequence
FDD	Frequency Division Duplex
GC	General Control (SAP)
HO	Handover
ITU	International Telecommunication Union
kbps	kilo-bits per second
L1	Layer 1 (physical layer)
L2	Layer 2 (data link layer)
L3	Layer 3 (network layer)
MAC	Medium Access Control
MS	Mobile Station
MM	Mobility Management
Nt	Notification (SAP)
PCCH	Paging Control Channel
PCH	Paging Channel
PDU	Protocol Data Unit
PU	Payload Unit.
PHY	Physical layer
PhyCH	Physical Channels
RACH	Random Access Channel
RLC	Radio Link Control
RNTI	Radio Network Temporary Identity
RRC	Radio Resource Control
SAP	Service Access Point
SCCH	Synchronization Control Channel
SCH	Synchronization Channel
SDU	Service Data Unit
SHCCH	Shared Channel Control Channel
TCH	Traffic Channel
TDD	Time Division Duplex
TFI	Transport Format Indicator
TFCI	Transport Format Combination Indicator
TPC	Transmit Power Control
U-	User-
UE	User Equipment
UL	Uplink
UMTS	Universal Mobile Telecommunications System
URA	UTRAN Registration Area
UTRA	UMTS Terrestrial Radio Access
UTRAN	UMTS Terrestrial Radio Access Network

4. General

4.1.Objective

4.2. Overview on sublayer architecture

The model presented in this section is not for implementation purposes.

4.2.1.Model of RLC

Figure 4-1 gives an overview model of the RLC layer. The figure illustrates the different RLC peer entities. There is one transmitting and one receiving entity for the transparent mode service and the unacknowledged mode service and one

combined transmitting and receiving entity for the acknowledged mode service. The dashed lines between the AM-Entities illustrate the possibility to send the RLC PDUs on separate logical channels, e.g. control PDUs on one and data PDUs on the other. More detailed descriptions of the different entities are given in subsections 4.2.1.1, 4.2.1.2 and 4.2.1.3.

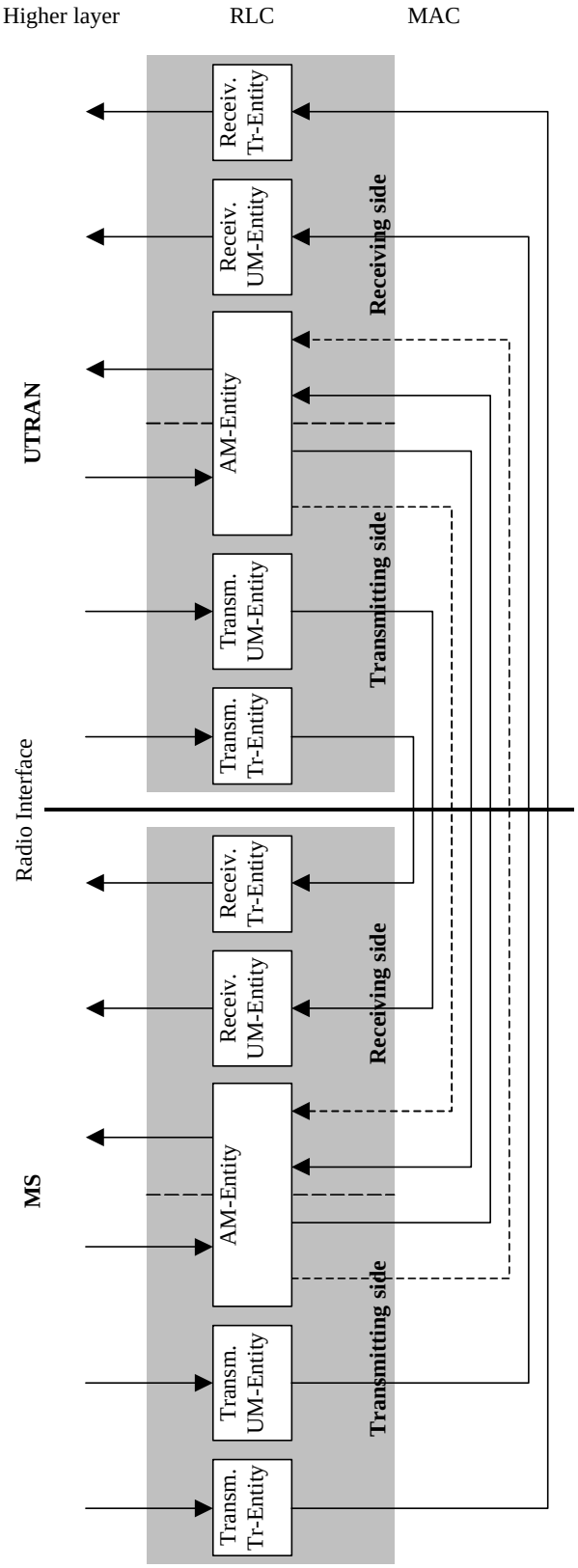


Figure 4-1 Overview model of RLC.

4.2.1.1. Transparent mode entities

Figure 4-2 below shows the model of two transparent mode peer entities.

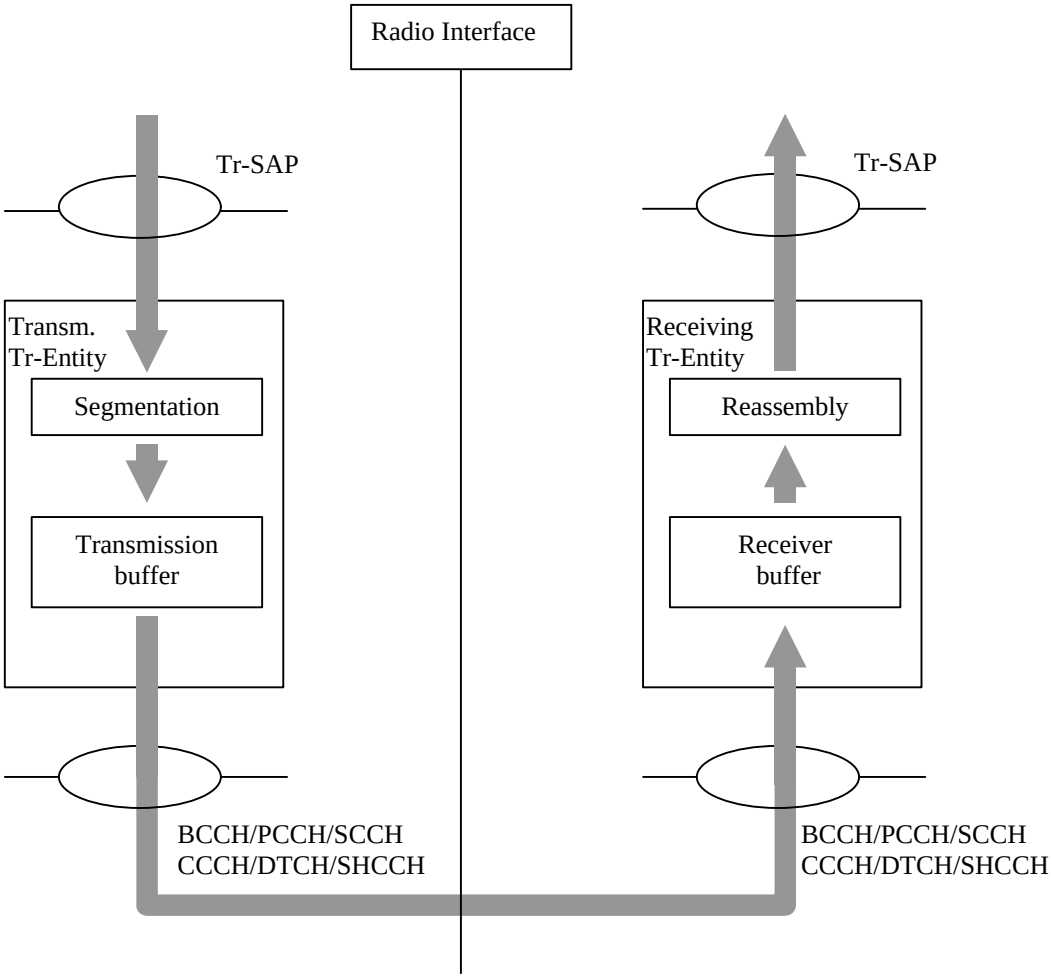


Figure 4-2 Model of two transparent mode peer entities.

The transmitting Tr-entity receives SDUs from the higher layers through the Tr-SAP. RLC might segment the SDUs into appropriate RLC PDUs without adding any overhead. How to perform the segmentation is decided upon when the service is established. RLC delivers the RLC PDUs to MAC through either a BCCH, PCCH, SHCCH, SCCH or a DTCH. The CCCH also uses transparent mode, but only for the uplink. Which type of logical channel depends on if the higher layer is located in the control plane (BCCH, PCCH, CCCH, SHCCH, SCCH (downlink only)) or user plane (DTCH).

The Tr-entity receives PDUs through one of the logical channels from the MAC sublayer. RLC reassembles (if segmentation has been performed) the PDUs into RLC SDUs. How to perform the reassembling is decided upon when the service is established. RLC delivers the RLC SDUs to the higher layer through the Tr-SAP.

4.2.1.2. Unacknowledged mode entities

Figure 4-3 below shows the model of two unacknowledged mode peer entities.

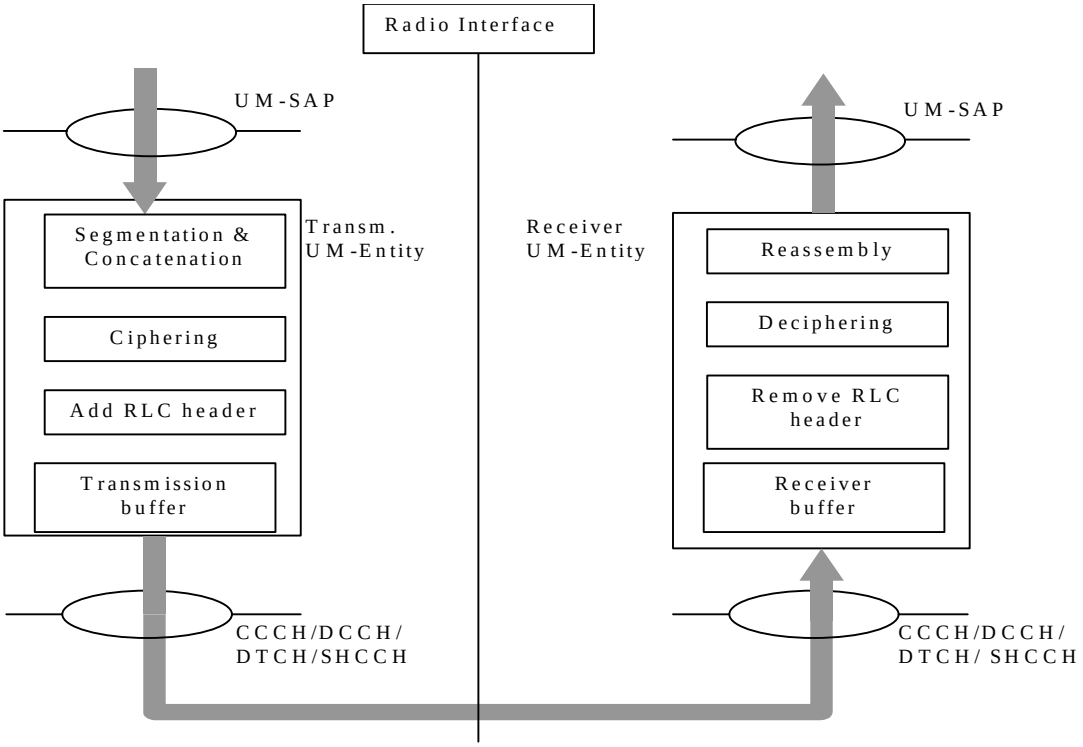


Figure 4-3 Model of two unacknowledged mode peer entities.

The transmitting UM-entity receives SDUs from the higher layers. If the SDU is very large it is segmented into RLC PDUs of appropriate size. The SDU might also be concatenated with other SDUs. RLC adds a header and the PDU is placed in the transmission buffer. RLC delivers the RLC PDUs to MAC through either a DCCH, a SHCCH (downlink only) or a DTCH. The CCCH also uses unacknowledged mode, but only for the downlink. Which type of logical channel depends on if the higher layer is located in the control plane (CCCH, DCCH, SHCCH) or user plane (DTCH).

The receiving UM-entity receives PDUs through one of the logical channels from the MAC sublayer. RLC removes header from the PDUs and reassembles the PDUs (if segmentation has been performed) into RLC SDUs. After that the SDUs are delivered to the higher layer.

4.2.1.3. Acknowledged mode entity

Figure 4-4 below shows the model of an acknowledged mode entity.

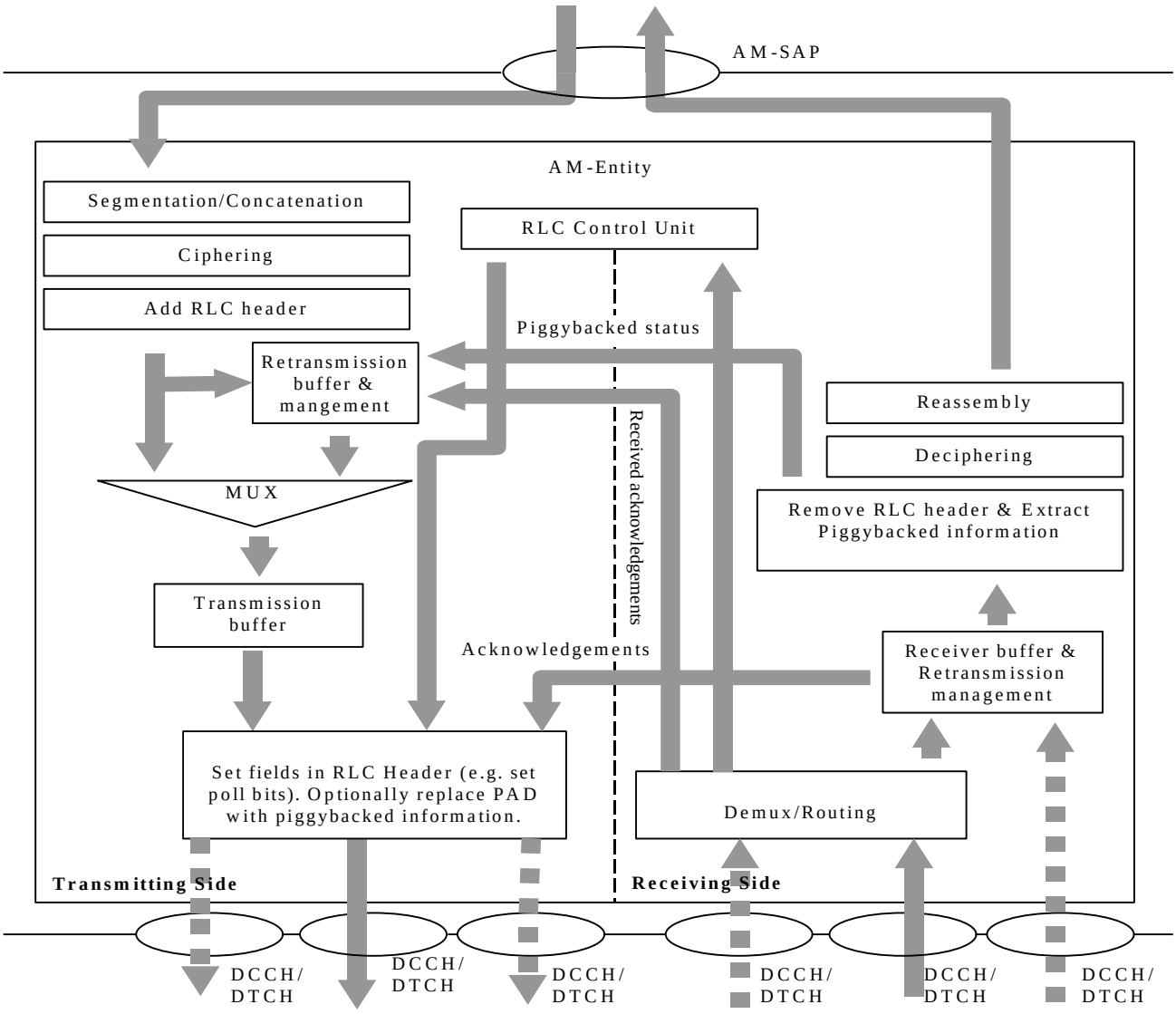


Figure 4-4 Model of a acknowledged mode entity.

The transmitting side of the AM-entity receives SDUs from the higher layers. The SDUs are segmented and/or concatenated to PUs of fixed length. PU length is a semi-static value that is decided in bearer setup and can only be changed through bearer reconfiguration by RRC.

For purposes of RLC buffering and retransmission handling, the operation is the same as if there would be one PU per PDU. For concatenation or padding purposes, bits of information on the length and extension, are inserted into the beginning of the last PU where data from an SDU is included. If several SDU:s fit into one PU, they are concatenated and the appropriate length indicators are inserted into the beginning of the PU. After that the PU:s are placed in the retransmission buffer and the transmission buffer. One or several PUs are included in one RLC PDU.

The MUX then decides which PDUs and when the PDUs are delivered to MAC, e.g. it could be useful to send RLC control PDUs on one logical channel and data PDUs on another logical channel. The PDUs are delivered via a function that completes the RLC-PDU header and potentially replaces padding with piggybacked status information. This includes setting the poll bit compressing subsequent PUs into one RLC-PDU or setting up the extended RLC-PDU header (PUs not in sequence) where applicable.

When Piggybacking mechanism is applied the padding is replaced by control information, in order to increase the transmission efficiency and making possible a faster message exchange between the peer to peer RLC entities. The piggybacked control information is not saved in any retransmission buffer. The piggybacked control information is contained in the piggybacked STATUS PDU which is in turn included into the AMD-PDU. The piggybacked STATUS PDUs will be of variable size in order to match with the amount of free space in the AMD PDU.

The dashed lines illustrate the case where AMD PDUs and control PDUs are transmitted on separate logical channels. The retransmission buffer also receives acknowledgements from the receiving side, which are used to indicate retransmissions of PUs and when to delete a PU from the retransmission buffer.

The Receiving Side of the AM-entity receives PDUs through one of the logical channels from the MAC sublayer. The RLC-PDUs are expanded into separate PUs and potential piggybacked status information are extracted. The PUs are placed in the receiver buffer until a complete SDU has been received. The receiver buffer requests retransmissions of PUs by sending negative acknowledgements to the peer entity. After that the headers are removed from the PDUs and the PDUs are reassembled into a SDU. Finally the SDU is delivered to the higher layer. The receiving side also receives acknowledgements from the peer entity. The acknowledgements are passed to the retransmission buffer on the transmitting side.

5. Functions

The following functions are supported by RLC.For a detailed description of the following functions see [3].

- Connection Control;
- Segmentation and reassembly;
- Header compression;
- Concatenation;
- Padding;
- Transfer of user data;
- Error correction;
- In-sequence delivery of higher layer PDUs;
- Duplicate Detection;
- Flow control;
- Sequence number check (Unacknowledged data transfer mode);
- Protocol error detection and recovery.
- Ciphering;

The following potential function(s) are regarded as further study items (FFS):

- Suspend/resume function;

6. Services provided to upper layers

This section describes the different services provided by RLC to higher layers. It also includes mapping of functions to different services.For a detailed description of the following functions see [3].

- **Transparent data transfer Service**

  The following functions are needed to support transparent data transfer:

  - Segmentation and reassembly
  - Transfer of user data;

- **Unacknowledged data transfer Service**

  The following functions are needed to support unacknowledged data transfer:

  - Segmentation and reassembly
  - Concatenation
  - Padding
  - Transfer of user data
  - Ciphering
  - Sequence number check;

- **Acknowledged data transfer Service**

  The following functions are needed to support acknowledged data transfer:

  - Segmentation and reassembly
  - Concatenation
  - Padding
  - Transfer of user data
  - Error correction
  - In-sequence delivery of higher layer PDUs
  - Duplicate detection
  - Flow Control
  - Protocol error detection and recovery
  - Ciphering;

- **QoS setting**
- **Notification of unrecoverable errors**

6.1.Mapping of services/functions onto logical channels

The following tables show the applicability of services and functions to the logical channels in UL/DL and UE/UTRAN. A '+' in a column denotes that the service/function is applicable for the logical channel in question whereas a '-' denotes that the service/function is not applicable.

Table 6-1: RLC modes and functions in UE uplink side

Service	Functions	CCCH	SHCCH	DCCH	DTCH
Transparent Service	Applicability	+	+	-	+
	Segmentation	-	-	-	+
	Transfer of user data	+	+	-	+
Unacknowledged Service	Applicability	-	-	+	+
	Segmentation	-	-	+	+
	Concatenation	-	-	+	+
	Padding	-	-	+	+
	Transfer of user data	-	-	+	+
	Ciphering	-	-	+	+
Acknowledged Service	Applicability	-	-	+	+
	Segmentation	-	-	+	+
	Concatenation	-	-	+	+
	Padding	-	-	+	+
	Transfer of user data	-	-	+	+
	Flow Control	-	-	+	+
	Error Correction	-	-	+	+
	Protocol error correction & recovery	-	-	+	+
	Ciphering	-	-	+	+

Table 6-2: RLC modes and functions in UE downlink side

Service	Functions	SCCH	BCCH	PCCH	SHCCH	CCCH	DCCH	DTCH	CTCH
Transparent Service	Applicability	+	+	+	+	-	-	+	+
	Reassembly	+	+	+	-	-	-	+	-
Unacknowledged Service	Applicability	-	-	-	+	+	+	+	+
	Reassembly	-	-	-	+	+	+	+	+
	Deciphering	-	-	-	-	-	+	+	-
	Sequence number check	-	-	-	+	+	+	+	+
Acknowledged Service	Applicability	-	-	-	-	-	+	+	-
	Reassembly	-	-	-	-	-	+	+	-
	Error correction	-	-	-	-	-	+	+	-
	Flow Control	-	-	-	-	-	+	+	-
	In sequence delivery	-	-	-	-	-	+	+	-
	Duplicate detection	-	-	-	-	-	+	+	-
	Protocol error correction & recovery	-	-	-	-	-	+	+	-
	Deciphering	-	-	-	-	-	+	+	-

Table 6-3: RLC modes and functions in UTRAN downlink side

Service	Functions	SCCH	BCCH	PCCH	CCCH	SHCCH	DCCH	DTCH	CTCH
Transparent Service	Applicability	+	+	+	-	+	-	+	+
	Segmentation	+	+	+	-	-	-	+	-
	Transfer of user data	+	+	+	-	+	-	+	+
Unacknowledged Service	Applicability	-	-	-	+	+	+	+	+
	Segmentation	-	-	-	+	+	+	+	+
	Concatenation	-	-	-	+	+	+	+	+
	Padding	-	-	-	+	+	+	+	+
	Ciphering	-	-	-	-	-	+	+	-
Acknowledged Service	Applicability	-	-	-	-	-	+	+	-
	Segmentation	-	-	-	-	-	+	+	-
	Concatenation	-	-	-	-	-	+	+	-
	Padding	-	-	-	-	-	+	+	-
	Transfer of user data	-	-	-	-	-	+	+	-
	Flow Control	-	-	-	-	-	+	+	-
	Error Correction	-	-	-	-	-	+	+	-
	Protocol error correction & recovery	-	-	-	-	-	+	+	-
	Ciphering	-	-	-	-	-	+	+	-

Table 6-4: RLC modes and functions in UTRAN uplink side

Service	Functions	CCCH	SHCCH	DCCH	DTCH
Transparent Service	Applicability	+	+	-	+
	Reassembly	-	-	-	+
Unacknowledged Service	Applicability	-	-	+	+
	Reassembly	-	-	+	+
	Deciphering	-	-	+	+
	Sequence number check	-	-	+	+
Acknowledged Service	Applicability	-	-	+	+
	Reassembly	-	-	+	+
	Error correction	-	-	+	+
	Flow Control	-	-	+	+
	In sequence delivery	-	-	+	+
	Duplicate detection	-	-	+	+
	Protocol error correction & recovery	-	-	+	+
	Deciphering	-	-	+	+

7. Services expected from MAC

For a detailed description of the following functions see [3].

- Data transfer;

8. Elements for layer-to-layer communication

8.1. Primitives between RLC and higher layers

The primitives between RLC and upper layers are shown in Table 8-1.

Table 8-1 : Primitives between RLC and upper layers

Generic Name	Parameter			
	Req.	Ind.	Resp.	Conf.
RLC-AM-DATA	Data, CFN, MUI	Data	Not Defined	MUI
RLC-UM-DATA	Data,	Data	Not Defined	Not Defined
RLC-TR-DATA	Data	Data	Not Defined	Not Defined
CRLC-CONFIG	E/R, Ciphering Elements (UM/AM only)	Not Defined	Not Defined	Not Defined
CRLC-STATUS	Not Defined	EVC	Not Defined	Not Defined

Each Primitive is defined as follows:

RLC-AM-DATA-Req/Ind/Conf

- RLC-AM-DATA-Req is used by higher layers to request transmission of a higher layer PDU in acknowledged mode.
- RLC-AM-DATA-Ind is used by RLC to deliver to higher layers RLC SDUs, that have been transmitted in acknowledged mode.

- RLC-AM-DATA-Conf is used by RLC to confirm to higher layers the transmission of a RLC SDU.

RLC-UM-DATA-Req/Ind

- RLC-UM-DATA-Req is used by higher layers to request transmission of a higher layer PDU in unacknowledged mode.
- RLC-UM-DATA-Ind is used by RLC to deliver to higher layers RLC SDUs, that have been transmitted in unacknowledged mode.

RLC-TR-DATA-Req/Ind

- RLC-TR-DATA-Req is used by higher layers to request transmission of a higher layer PDU in transparent mode.
- RLC-TR-DATA-Ind is used by RLC to deliver to higher layers RLC SDUs, that have been transmitted in transparent mode.

CRLC-CONFIG-Req

This primitive is used by RRC to establish, release or reconfigure the RLC. Ciphering elements are included for UM and AM operation.

CRLC-STATUS-Ind

It is used by the RLC to send status information to RRC.

Following parameters are used in the primitives:

1. The parameter Data is the RLC SDU that is mapped onto the Data field in RLC PDUs. The Data parameter may be divided over several RLC PDUs.In case of a RLC-AM-DATA or a RLC-UM-DATA primitive the length of the Data parameter shall be octet alligned.
2. The parameter Confirmation request (CNF) indicates whether the RLC needs to confirm the correct transmission of the RLC SDU.
3. The parameter Message Unit Identifier (MUI) is an identity of the RLC SDU, which is used to indicate which RLC SDU that is confirmed with the RLC-AM-DATA conf. primitive.
4. The parameter E/R indicates whether RLC should enter or exit the data transfer ready state.
5. The parameter Event Code (EVC) indicates the reason for the CRLC-STATUS-ind (i.e unrecoverable errors such as data link layer loss or recoverable status events such as reset, etc.).
6. The parameter ciphering elements are only applicable for UM and AM operation. These parameters are Ciphering Mode, Ciphering Key and Ciphering Sequence Number.

9. Elements for peer-to-peer communication

9.1.Protocol data units

9.1.1.Data PDUs

a) TrD PDU (Transparent Mode Data PDU)
The TrD PDU is used to convey RLC SDU data without adding any RLC overhead. The TrD PDU is used by RLC when it is in transparent mode.

b) UMD PDU (Unacknowledged Mode Data PDU)
The UMD PDU is used to convey sequentially numbered PDUs containing RLC SDU data. It is used by RLC when using unacknowledged data transfer.

c) AMD PDU (Acknowledged Mode Data PDU)
The AMD PDU is used to convey sequentially numbered PUs containing RLC SDU data. The AMD PDU is used by RLC when it is in acknowledged mode.

9.1.2. Control PDUs

a) STATUS PDU and Piggybacked STATUS PDU
The STATUS PDU and the Piggybacked STATUS PDU are used:

- by the receiving entity to inform the transmitting entity about missing PUs at the receving entity;
- by the receiving entity to inform the transmitting entity about the size of the allowed transmission window;
- and by the transmitting entity to request the receiving entity to move the receiving window.

b) RESET (Reset)
The RESET PDU is used in acknowledged mode to reset all protocol states, protocol variables and protocol timers of the peer RLC entity in order to synchronise the two peer entities.

c) RESET ACK (Reset Acknowledge)
The RESET ACK PDU is an acknowledgement to the RESET PDU.

Table 9-1: RLC PDU names and descriptions

Data Transfer Mode	PDU name	Description
Transparent	TrD	Transparent mode data
Unacknowledged	UMD	Sequenced unacknowledged mode data
Acknowledged	AMD	Sequenced acknowledged mode data
	STATUS	Solicited or Unsolicited Status Report
	Piggybacked STATUS	Piggybacked Solicited or Unsolicited Status Report
	RESET	Reset Command
	RESET ACK	Reset Acknowledgement

9.2.Formats and parameters

9.2.1.Formats

This section specifies the format of the RLC PDUs. The parameters of each PDU are explained in section 9.2.2.

9.2.1.1. TrD PDU

The TrD PDU transfers user data when RLC is operating in transparent mode. No overhead is added by RLC.

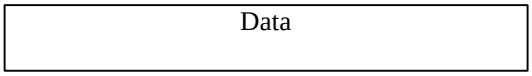


Figure 9-1. TrD PDU

9.2.1.2. UMD PDU

The UMD PDU transfers user data when RLC is operating in unacknowledged mode.

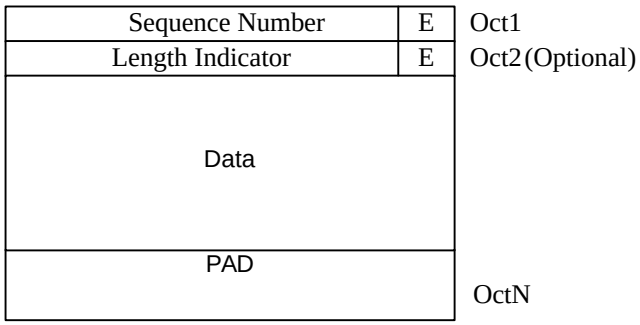


Figure 9-2. UMD PDU

9.2.1.3. AMD PDU

The AMD PDU transfers user data and piggybacked status information and requests status report by setting Poll bit when RLC is operating in acknowledged mode.

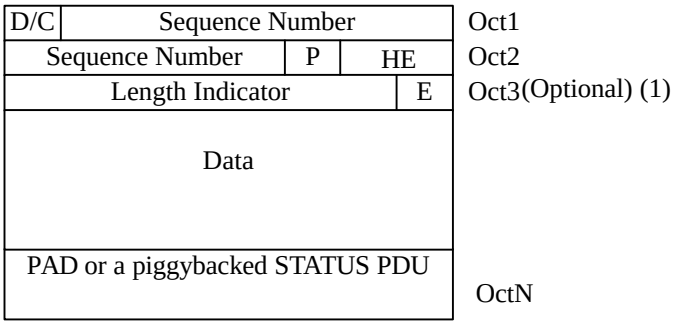


Figure 9-3. AMD PDU

(1) Note: The Length Indicator maybe 15bits.

9.2.1.4. STATUS PDU

The STATUS PDU is used to report the status between two RLC AM entities. Both receiver and transmitter status information may be included in the same STATUS PDU.

The format of the STATUS PDU is given in Figure 9-4 below.

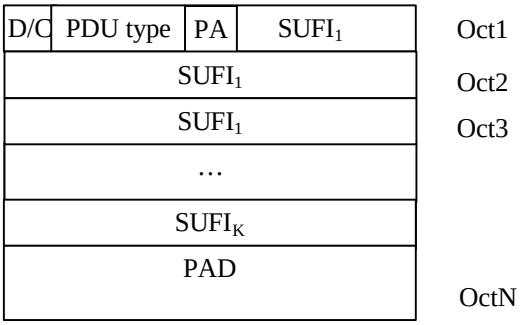


Figure 9-4. Status Information Control PDU (STATUS PDU)

Up to K different super-fields ($SUFI_1$-$SUFI_K$) can be included into one STATUS PDU. The size of a STATUS PDU is variable and upper bounded by the maximum RLC PDU size used by an RLC entity. Padding shall be included to exactly fit one of the PDU sizes used by the entity.

9.2.1.5. Piggybacked STATUS PDU

The format of the piggybacked STATUS PDU is the same as the ordinary STATUS PDU except that the D/C field and the PDU type field is omitted. This PDU can be used to piggyback STATUS PDU in a AMD PDU if the data does not fill the complete AMD PDU

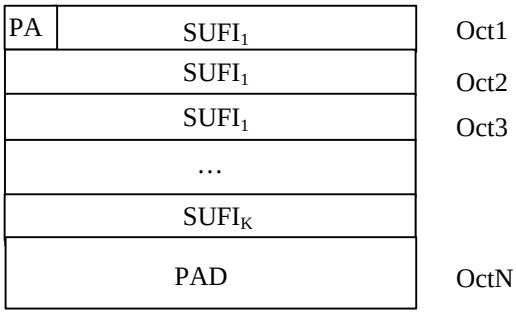


Figure 9-5. Piggybacked STATUS PDU

9.2.1.6. RESET, RESET ACK PDU

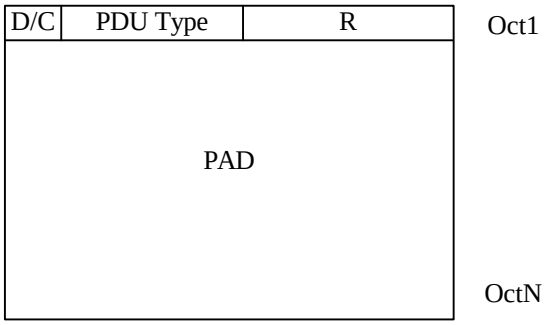


Figure 9-6. RESET, RESET ACK PDU

9.2.2. Parameters

9.2.2.1. D/C field

Length: 1bit

The D/C field indicates the type of an acknowledged mode PDU. It can be either data or control PDU.

Bit	Description
0	Control PDU
1	Acknowledged mode data PDU

9.2.2.2. PDU Type

Length: 3 bit

The PDU type field indicates the Control PDU type

Bit	PDU Type
000	STATUS
001	RESET
010	RESET ACK

9.2.2.3. Sequence Number (SN)

This field indicates the sequence number of the payload unit. In a normal AMD-PDU the sequence number of the first PU in the PDU is indicated. If the PUs are not in sequence, a sequence number is indicated separately for each PU in the extended header.

PDU type	Length	Notes
AMD PDU	12 bits	Used for retransmission and reassembly
UMD PDU	7 bits	Used for reassembly

9.2.2.4. Polling bit (P)

Length: 1bit

This field is used to request a status report (STATUS PDU) from the receiver RLC.

Bit	Description
0	-

1	Request a status report

9.2.2.5. Extension bit (E)

Length: 1bit

This bit indicates if the next octet will be a length indicator and E bit.

Bit	Description
0	The next field is data
1	The next field is Length Indicator and E bit

9.2.2.6. Reserved (R)

Length: 4 bits

This field is used to achieve octet alignment and for this purpose it is coded as 0000. Other functions of it are left for future releases.

9.2.2.7. Header Extension Type (HE)

Length: 2 bits

This two-bit field indicates the format of the extended header.

Value	Description
00	The succeeding octet contains data
01	The succeeding octet contains a 7bit length indicator and E bit
10	The succeeding octet contains an extended header field
11	The succeeding octet contains a 15bit length indicator and E bit

9.2.2.7.1. AMD PDU Extended Header

The Extended Header is used when additional sequence numbers are needed to indicate PUs that are not sequential within a PDU or when the rest of a PDU, which is not filled by PUs, is equal or larger than the size of a PU. A PDU that includes more than one sequence number shall include sequence numbers for all PUs in the PDU. The n^{th} sequence number in the PDU indicates the sequence number of the n^{th} PU in the PDU. The decision to use Extended Header is made by the transmitting RLC.

First all the Extended Headers are listed. Then all Length Indicators are listed. Finally the PUs follow.

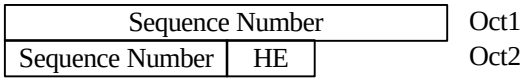


Figure 9-7. Format of the extended header

9.2.2.8. Length Indicator (LI)

This field is optional and is used if concatenation, padding or a piggybacked STATUS PDU takes place in a PU. It indicates the end of the last segment of a SDU. It points out the end of a segment by giving the number of octets between the end of the header fields (including the length indicator fields) and the end of the segment. The size of the Length Indicator may be either 7bits or 15bits. If the last segment of a SDU do not completely fill a PU either padding or a piggybacked STATUS PDU can be added. Predefined values of the length indicator are used to indicate this. The padding/piggybacked STATUS PDU predefined length indicators shall be added after the length indicator that indicates the end of the last SDU segment in the PU. The values that are reserved for special purposes are listed in the tables below depending on the size of the Length Indicator.

If a length indicator that indicates padding/piggybacked STATUS PDU refers to the last PU in the PDU it implicitly means that the rest of the PDU contains padding/piggybacked STATUS PDU. If the last PU in a PDU does not include padding or piggybacked STATUS PDU, but the PDU includes padding or a piggybacked STATUS PDU, an extra length indicator field shall be added as a normal length indicator to the last PU. This extra length indicator shall indicate either padding or a piggybacked STATUS PDU and shall be placed as the last length indicator in the PDU. The space needed for this length indicator shall not be taken from the data part in the PU, but from the padding or piggybacked STATUS PDU in the PDU. The receiving entity shall discard this length indicator.

If RLC PDUs always carry only one PU, 7bit indicators are used in a particular RLC PDU if the address space is sufficient to indicate all SDU segment borders. Otherwise 15bit Length Indicators are applied.

If RLC PDUs may carry more than one PUs the length of the Length Indicator only depends on the size of the largest RLC PDU and the size of the Length Indicator is always the same for all PUs.

Only one size of Length Indicators is used in one RLC PDU.

Length: 7bit

Bit	Description
0000000	The previous RLC PU was exactly filled with the last segment of a RLC SDU.
1111110	The rest part of the RLC PU includes a piggybacked STATUS PDU.
1111111	The rest part of the RLC PU is padding.

Length: 15bit

Bit	Description
000000000000000	The previous RLC PU was exactly filled with the last segment of a RLC SDU.
111111111111110	The rest part of the RLC PU includes a piggybacked STATUS PDU.
111111111111111	The rest part of the RLC PU is padding.

9.2.2.9. Data

RLC SDUs are mapped to this field. If a SDU is too large to fit into the data field it is segmented. If possible, the last segment of a SDU shall be concatenated with the first segment of the next SDU in order to fill the data field completely and avoid unnecessary padding. The length indicator field is used to point the borders between SDUs.

9.2.2.10. Padding (PAD)

Padding may have any value and the receiving entity shall disregard it.

9.2.2.11. Poll Answer (PA)

Length: 1bit

The PA (Poll Answer) field indicates whether the status report is the answer to a poll or not

Bit	Description
0	The status report is not the answer to a polling request
1	The status report is the answer to a polling request

9.2.2.12. SUFI

Length: variable number of bits

The SUFI (Super-Field) includes three sub-fields: type information (type of super-field, e.g. list, bitmap, acknowledgement, etc), length information (providing the length of a variable length field within the following value field) and a value. Figure 9-8 shows the structure of the super-field. The size of the type sub-field is non-zero but the size of the other sub-fields may be zero.

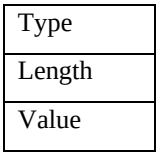


Figure 9-8. The Structure of a Super-Field

The length of the type field is 3 bits and it may have any of following values.

Bit	Description
000	No More Data (**NO_MORE**)
001	Window Size (**WINDOW**)
010	Acknowledgement (**ACK**)
011	List (**LIST**)
100	Bitmap (**BITMAP**)
101	Relative list (**Rlist)**
110	Move Receiving Window **(MRW)**
111	*Reserved for future super-field types*

The length sub-field gives the length of the variable size part of the following value sub-field and the length of it depends on the super-field type.The value sub-field includes the value of the super-field, e.g. the bitmap in case of a BITMAP super-field, and the length is given by the length or the type sub-field.

9.2.2.12.1. The No More Data super-field

The 'No More Data' super-field indicates the end of the data part of a STATUS PDU and is shown in Figure 9-9 below. It shall always be placed as the last SUFI if it is included in a STATUS PDU. All data after this SUFI shall be regarded as padding and shall be neglected.

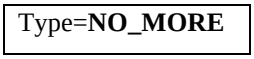


Figure 9-9. NO_MORE field in a STATUS PDU

9.2.2.12.2. The Acknowledgement super-field

The 'Acknowledgement' super-field consists of a type identifier field (ACK) and a sequence number (LSN) as shown in Figure 9-10 below. The acknowledgement super-field is also indicating the end of the data part of a STATUS PDU. Thus, no 'NO_MORE' super-field is needed in the STATUS PDU when the 'ACK' super-field is present. The ACK SUFI shall always be placed as the last SUFI if it is included in a STATUS PDU. All data after this SUFI shall be regarded as padding and shall be neglected.

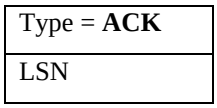


Figure 9-10. The ACK fields in a STATUS PDU

LSN
Length: 12 bits
Acknowledges the reception of all PUs with sequence numbers < LSN (Last Sequence Number) that are *not* indicated to be erroneous in earlier parts of the STATUS PDU. The LSN should not be set to a value ≥ VR(H). This means that if the LSN is set to a different value than VR(R) all erroneous PUs must be included in the same STATUS PDU and VT(A) will be updated according to the first error indicated in the STATUS PDU.

9.2.2.12.3. The Window Size super-field

The 'Window Size' super-field consists of a type identifier (WINDOW) and a window size number (WSN) as shown in Figure 9-11 below. The receiver is always allowed to change the window size during a connection.

Type = **WINDOW**
WSN

Figure 9-11. The WINDOW fields in a STATUS PDU

WSN
Length: 12 bits
The allowed window size to be used by the transmitter. The range of the window size is [0, 2^{12}-1]. The Window_Size parameter is set equal to WSN.

9.2.2.12.4. The List super-field

The List Super-Field consists of a type identifier field (LIST), a list length field (LENGTH) and a list of LENGTH number of pairs as shown in Figure 9-12 below:

Type = **LIST**
LENGTH
SN_1
L_1
SN_2
L_2
…
SN_{LENGTH}
L_{LENGTH}

Figure 9-12. The List fields in a STATUS PDU for a list

LENGTH
Length: 4 bits
The number of (SN_i , L_i)-pairs in the super-field of type LIST.

SN_i
Length: 12 bits
Sequence number of PU which was not correctly received.

L_i
Length: 4 bits
Number of consecutive PUs not correctly received following PU with sequence number SN_i.

9.2.2.12.5. The Bitmap super-field

The Bitmap Super-Field consists of a type identifier field (BITMAP), a bitmap length field (LENGTH), a first sequence number (FSN) and a bitmap as shown in Figure 9-13 below:

Type = **BITMAP**
LENGTH
FSN
Bitmap

Figure 9-13. The Bitmap fields in a STATUS PDU.

LENGTH
Length: 4 bits
The size of the bitmap in octets (maximum bitmap size: 2^4*8=128 bits).

FSN

Length: 12 bits
The sequence number for the first bit in the bitmap.

Bitmap
Length: Variable number of octets given by the LENGTH field.
Status of the SNs in the interval [FSN, FSN + LENGTH*8 - 1] indicated in the bitmap where each position (from left to right) can have two different values (0 and 1) with the following meaning (bit_position∈[0,LENGTH*8 - 1]):
1: SN = (FSN + bit_position) has been correctly received
0: SN = (FSN + bit_position) has not been correctly received

9.2.2.12.6. The Relative List super-field

The Relative List super-field consists of a type identifier field (RLIST), a list length field (LENGTH), the first sequence number (FSN) and a list of LENGTH number of codewords (CW) as shown in Figure 9-14 below.

Type = **RLIST**
LENGTH
FSN
CW_1
CW_2
…
CW_{LENGTH}

Figure 9-14. The RList fields in a STATUS PDU

LENGTH
Length: 4 bits
The number of codewords (CW) in the super-field of type RLIST.

FSN

Length: 12 bits
The sequence number for the first erroneous PU in the RLIST.

CW
Length: 4 bits
The CW consists of 4 bits where the three first bits are part of a number and the last bit is a status indicator and it shall be interpreted as follows.

Code Word	Description
$X_1X_2X_3$ 0	Next 3 bits of the number are $X_1X_2X_3$ and the number continues in the next CW. The most significant bit within this CW is X_1.
$X_1X_2X_3$ 1	Next 3 bits of the number are $X_1X_2X_3$ and the number is terminated. The most significant bit within this CW is X_1. This is the most significant CW within the number.

By default, the number given by the CWs represents a distance from the previous indicated erroneous PU to the next erroneous PU.

One special value of CW is defined:

000 1 'Error burst indicator'

The error burst indicator means that the next CW:s will represent the number of subsequent erroneous PU:s (not counting the already indicated error position). After the number of errors in a burst is terminated with XXX 1, the next codeword will again by default be the least significant bits (LSB) of the distance to the next error.

9.2.2.12.7. The Move Receiving Window super-field

The 'Move Receiving Window' super-field is used to request the RLC receiver to move its receiving window, as a result of a SDU discard in the RLC transmitter. The format is given in the figure below.

Type = **MRW**
SN

Figure 9-15. The MRW fields in a STATUS PDU

SN
Length: 12 bits
Requests the RLC receiver to discard all PUs with sequence number < SN, and to move the receiving window accordingly.

9.3.Protocol states

9.3.1.State model for transparent mode entities

Figure 9-16 illustrates the state model for transparent mode RLC entities (both transmitting and receiving). A transparent mode entity can be in one of following states.

9.3.1.1. Null State

In the null state the RLC entity does not exist and therefore it is not possible to transfer any data through it.

Upon reception of an CRLC-CONFIG-Req from higher layer the RLC entity is created and transparent data transfer ready state is entered.

9.3.1.2. Transparent Data Transfer Ready State

In the transparent data transfer ready, transparent mode data can be exchanged between the entities. Upon reception of an CRLC-CONFIG-Req from higher layer the RLC entity is terminated and the null state is entered.

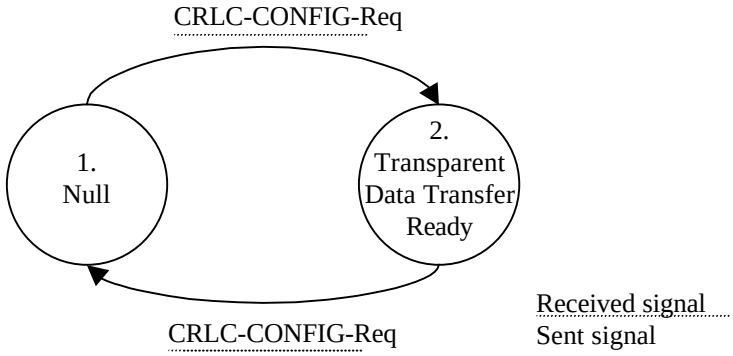


Figure 9-16. The state model for transparent mode entities.

9.3.2.State model for unacknowledged mode entities

Figure 9-17 illustrates the state model for unacknowledged mode RLC entities (both transmitting and receiving). An unacknowledged mode entity can be in one of following states.

9.3.2.1. Null State

In the null state the RLC entity does not exist and therefore it is not possible to transfer any data through it.

Upon reception of an CRLC-CONFIG-Req from higher layer the RLC entity is created and unacknowledged data transfer ready state is entered.

9.3.2.2. Unacknowledged Data Transfer Ready State

In the unacknowledged data transfer ready, unacknowledged mode data can be exchanged between the entities. Upon reception of an CRLC-CONFIG-Req from higher layer the RLC entity is terminated and the null state is entered.

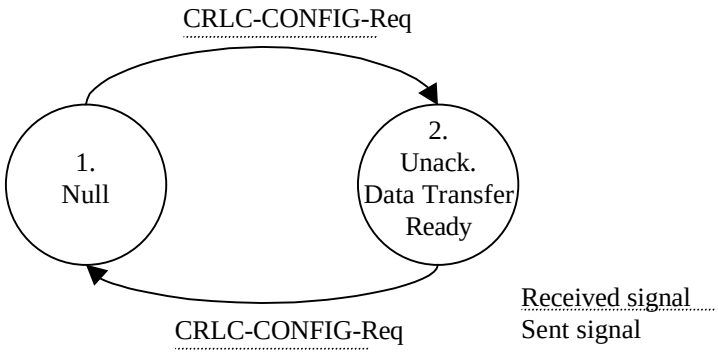


Figure 9-17. The state model for unacknoledged mode entities.

9.3.3. State model for acknowledged mode entities

Figure 9-18 illustrates the state model for the acknowledged mode RLC entity (both transmitting and receiving). An acknowledged mode entity can be in one of following states.

9.3.3.1. Null State

In the null state the RLC entity does not exist and therefore it is not possible to transfer any data through it.

Upon reception of an CRLC-CONFIG-Req from higher layer the RLC entity is created and acknowledged data transfer ready state is entered.

9.3.3.2. Acknowledged Data Transfer Ready State

In the acknowledged data transfer ready state, acknowledged mode data can be exchanged between the entities. Upon reception of a CRLC-CONFIG-Req from higher layer the RLC entity is terminated and the null state is entered.

Upon errors in the protocol, the RLC entity sends a RESET PDU to its peer and enters the reset pending state.

Upon reception of a RESET PDU, the RLC entity resets the protocol and responds to the peer entity with a RESET ACK PDU.

Upon reception of a RESET ACK PDU, the RLC takes no action.

9.3.3.3. *Reset* Pending State

In the reset pending state the entity waits for a response from its peer entity and no data can be exchanged between the entities. Upon reception of CRLC-CONFIG-Req from higher layer the RLC entity is terminated and the null state is entered.

Upon reception of a RESET ACK PDU, the RLC entity resets the protocol and enters the acknowledged data transfer ready state.

Upon reception of a RESET PDU, the RLC entity resets the protocol, send a RESET ACK PDU and enters the acknowledged data tranfer ready state.

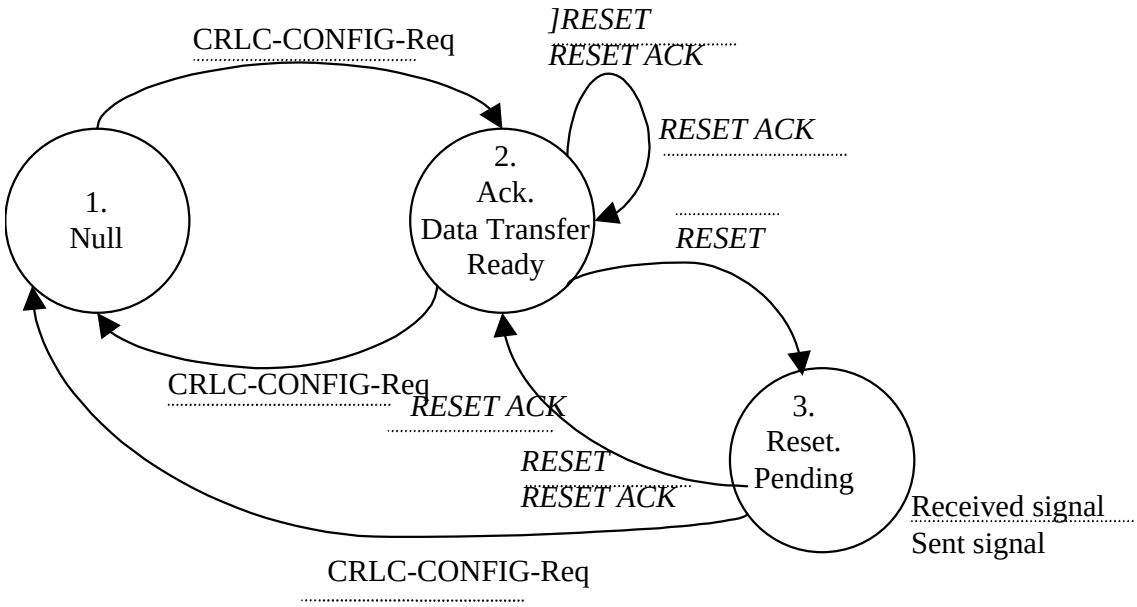


Figure 9-18. The state model for the acknoledged mode entities.

9.4.State variables

This sub-clause describes the state variables used in the specification of the peer-to-peer protocol. PUs are sequentially and independently numbered and may have the value 0 through n minus 1 (where n is the modulus of the sequence numbers). The modulus equals 2^{12} for AM and 2^{7} for UM; the sequence numbers cycle through the entire range: 0 through $2^{12} - 1$ for AM and 0 through $2^{7} - 1$ for UM. All arithmetic operations on the following state variables and sequence numbers contained in this specification are affected by the modulus: VT(S), VT(A), VT(MS), VR(R), VR(H), VR(MR), VT(US) and VR(US). When performing arithmetic comparisons of transmitter variables, VT(A) is assumed to be the base. When performing arithmetic comparisons of receiver variables, VR(R) is assumed to be the base.

The RLC maintains the following state variables at the transmitter.

a) VT(S) - Send state variable
The sequence number of the next PU to be transmitted for the first time (i.e. excluding retransmission). It is updated after transmission of a PDU which includes not earlier transmitted PUs. The initial value of this variable is 0.

b) VT(A) - Acknowledge state variable
The sequence number of the next in-sequence PU expected to be acknowledged, which forms the lower edge of the window of acceptable acknowledgments. VT(A) is updated based on receipt of a STATUS PDU including an ACK super-field. The initial value of this variable is 0.

c) VT(DAT)
This state variable counts the number of times a PU has been transmitted. There is one VT(DAT) for each PU and it is incremented each time the PU is transmitted. The initial value of this variable is 0.

d) VT(MS) - Maximum Send state variable
The sequence number of the first PU not allowed by the peer receiver [i.e. the receiver will allow up to VT(MS) – 1], VT(MS) = VT(A) + Window_Size. This value represents the upper edge of the transmit window. The transmitter shall not transmit a new PU if VT(S) $\geq$ VT(MS). VT(MS) is updated based on receipt of a STATUS PDU including an ACK and/or a WINDOW super-field.

e) VT(US) – UM data state variable
This state variable gives the sequence number of the next UMD PDU to be transmitted. It is updated each time a UMD PDU is transmitted. The initial value of this variable is 0.

f) VT(PU)
This state variable is used when the poll every Poll_PU PU function is used. It is incremented with 1 for each PU that is transmitted. It should be incremented for both new and retransmitted PUs. When it reaches Poll_PU a new poll is transmitted and the state variable is set to zero. The initial value of this variable is 0.

g) VT(SDU)

This state variable is used when the poll every Poll_SDU SDU function is used. It is incremented with 1 for each SDU that is transmitted. When it reaches Poll_SDU a new poll is transmitted and the state variable is set to zero. The poll bit should be set in the PU that contains the last segment of the SDU. The initial value of this variable is 0.

h) VT(RST) - Reset state variable
It is used to count the number of times a RESET PDU is transmitted. VT(RST) is incremented with 1 each time a RESET PDU is transmitted. VT(RST) is reset upon the reception of a RESET ACK PDU. The initial value of this variable is 0.

The RLC maintains the following state variables at the receiver:

a) VR(R) - Receive state variable
The sequence number of the next in-sequence PU expected to be received. It is updated upon receipt of the next in-sequence PU. The initial value of this variable is 0.

b) VR(H) - Highest expected state variable
The sequence number of the highest expected PU. This state variable is updated when a new PU is received with SN≥VR(H). The initial value of this variable is 0.

c) VR(MR) - Maximum acceptable Receive state variable
The sequence number of the first PU not allowed by the receiver [i.e. the receiver will allow up to VR(MR) – 1], VR(MR) = VR(R) + Window_Size. The receiver shall discard PUs with SN ≥ VR(MR), (in one case, such a PU may cause the transmission of an unsolicited STATUS PDU).

d) VR(US) - Receiver Send Sequence state variable
The sequence number of the next PDU to be received. It shall set equal to SN + 1 upon reception of a PDU. The initial value of this variable is 0.

e) VR(EP) – Estimated PDU Counter state variable
The number of PUs that should be received yet as a consequence of the transmission of the latest STATUS PDU. In acknowledged mode, this state variable is updated at the end of each transmission time interval. It is decremented by the number of PUs that should have been received during the transmission time interval. If VR(EP) is equal to zero, then check if all PUs requested for retransmission in the latest STATUS PDU have been received.

9.5.Timers

a) Timer_Poll
This timer is only used when the poll timer trigger is used. It is started when the transmitting side sends a poll to the peer entity. The timer is stopped when receiving a STATUS PDU that contains an acknowledgement or negative acknowledgement of the AMD PDU that triggered the timer. The value of the timer is signalled by RRC.

If the timer expires and no STATUS PDU containing an acknowledgement or negative acknowledgement of the AMD PDU that triggered the timer has been received, the receiver is polled once more (either by the transmission of a PDU which was not yet sent, or by a retransmission) and the timer is restarted.

If a new poll is sent when the timer is running it is restarted.

b) Timer_Poll_Prohibit
This timer is only used when the poll prohibit function is used. It is used to prohibit transmission of polls within a certain period. A poll shall be delayed until the timer expires if a poll is triggered when the timer is activeOnly one poll shall be transmitted when the timer expires even if several polls were triggered when the timer was active. This timer will not be stopped by a STATUS PDU. The value of the timer is signalled by RRC.

c) Timer_EPC
This timer is only used when the EPC function is used and it accounts for the roundtrip delay, i.e. the time when the first retransmitted PU should be received after a STATUS has been sent. The timer is started when a STATUS report is transmitted and when it expires EPC can start decrease (see section 9.7.3). The value of the timer is signalled by RRC.

d) Timer_Discard
This timer is used for the SDU discard function. In the transmitter, the timer is activated upon reception of a SDU from 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 between sequence numbers of received PDUs. The value of the timer is signalled by RRC.

e) Timer_Poll_Periodic
This timer is only used when the timer based polling is used. The timer is started when the RLC entity is created. Each time the timer expires a poll is transmitted and the timer is restarted. The value of the timer is signalled by RRC.

f) Timer_Status_Prohibit
This timer is only used when the STATUS PDU prohibit function is used. It prohibits the receiving side from sending STATUS PDUs. The timer is started when a STATUS PDU is transmitted and no new STATUS PDU can be transmitted before the timer has expired. The value of the timer is signalled by RRC.

g) Timer_Status_Periodic
This timer is only used when timer based STATUS PDU sending is used. The timer is started when the RLC entity is created. Each time the timer expires a STATUS PDU is transmitted and the timer is restarted. The value of the timer is signalled by RRC.

h) Timer_RST
It is used to detect the loss of RESET ACK PDU from the peer RLC entity. This timer is set when the RESET PDU is transmitted. And it will be stopped upon reception of RESET ACK PDU. If it expires, RESET PDU will be retransmitted.

9.6.Protocol Parameters

a) MaxDAT
It is the maximum value for the number of retransmissions of a PU. This parameter is an upper limit of counter VT(DAT). When the value of VT(DAT) comes to MaxDAT, error recovery procedure will be performed.

b) Poll_PU
This parameter indicates how often the transmitter should poll the receiver in case of polling every Poll_PU PU. This is an upper limit for the VT(PU) state variable, when VT(PU) reaches Poll_PU a poll is transmitted to the peer entity.

c) Poll_SDU
This parameter indicates how often the transmitter should poll the receiver in case of polling every Poll_SDU SDU. This is an upper limit for the VT(SDU) state variable, when VT(SDU) reaches Poll_SDU a poll is transmitted to the peer entity.

d) Poll_Window
This parameter indicates when the transmitter should poll the receiver in case of performing window based polling. A poll is transmitted when:

$$1 - \frac{(Window\_Size + VT(MS) - VT(S)) \bmod Window\_Size}{Window\_Size} > Poll\_Window.$$

e) MaxRST
It is the maximum value for the number of retransmission of RESET PDU. This parameter is an upper limit of counter VT(RST). When the value of VT(RST) comes to MaxRST, the higher layer (RRC) is notified.

f) Window_Size
The maximum allowed transmitter window size.

9.7.Specific functions

9.7.1.Polling function for acknowledged mode transfer

The transmitter of AMD PDUs may poll the receiver for a STATUS PDU. The Polling bit in the AMD PDU indicates the poll request. There are several triggers for setting the polling bit. The network (RRC) controls which triggers should be used for each RLC entity. Following triggers are possible:

1. Last PU in buffer
The sender transmits a poll when the last PU available for transmission is transmitted.

2. Last PU in retransmission buffer
The sender transmits a poll when the last PU to be retransmitted is transmitted.

3. Poll timer
The timer Timer_Poll is started when a poll is transmitted to the receiver and if no STATUS PDU has been received before the timer Timer_Poll expires a new poll is transmitted to the receiver.

4. Every Poll_PU PU
The sender polls the receiver every Poll_PU PU. Both retransmitted and new Pus shall be counted.

5. Every Poll_SDU SDU

The sender polls the receiver every Poll_SDU SDU.

6. Poll_Window% of transmission window
   The sender polls the receiver when it has reached Poll_Window% of the transmission window.

7. Timer based
   The sender polls the receiver periodically.

The network also controls if the poll prohibit function shall be used. The poll bit shall be set to 0 if the poll prohibit function is used and the timer Timer_Poll_Prohibit is active. This function has higher priority than any of the above mentioned triggers.

9.7.2. STATUS PDU transmission for acknowledged mode

The receiver of AMD PDUs transmits STATUS PDUs to the sender in order to inform about which PUs that have been received and not received. There are several triggers for sending a STATUS PDU. The network (RRC) controls which triggers should be used for each RLC entity, except for one, which is always present. The receiver shall always send a STATUS PDU when receiving a poll request. Except for that trigger following triggers are configurable:

1. Detection of missing PU(s).
   If the receiver detects one or several missing PUs it shall send a STATUS PDU to the sender.

2. Timer based STATUS PDU transfer
   The receiver transmits a STATUS PDU periodically to the sender. The time period is controlled by the timer Timer_Status_Periodic.

3. The EPC mechanism
   The EPC is started when a STATUS PDU is transmitted to the peer entity. If not all PUs requested for retransmission have been received before the EPC has expired a new STATUS PDU is transmitted to the peer entity. A more detailed description of the EPC mechanism is given in section 9.7.2.

There are two functions that can prohibit the receiver from sending a STATUS PDU. The network (RRC) controls which functions should be used for each RLC entity. If any of the following functions is used the sending of the STATUS PDU shall be delayed, even if any of the conditions above are fulfilled:

1. STATUS PDU prohibit
   The Timer_Status_Prohibit is started when a STATUS PDU is transmitted to the peer entity. As long as the timer is running the receiving side is not allowed to send a STATUS PDUs to the peer entity. The STATUS PDU is transmitted after the timer has expired. The receiver shall only send information about a PU once, even if there are several triggers when the timer running.

2. The EPC mechanism
   If the EPC mechanism is active and the sending of a STATUS PDU is triggered it shall be delayed until the EPC mechanism has ended. The receiver shall only send information about a PU once, even if there are several triggers when the timer is active or the counter is counting down.

9.7.3. SDU discard function

The SDU discard function allows to discharge RLC PDU from the buffer on the transmitter side, when the transmission of the RLC PDU does not success for a long time.The SDU discard function allows to avoid buffer overflow, in the case of non-transparent transmission mode. There will be several alternative operation modes of the RLC SDU discard function, and which discard function to use will be given by the QoS requirements of the Radio Access Bearer.
The following is a list of operation modes for the RLC SDU discard function.

Table 9-2. List of criteria's that control when to perform SDU discard.

Operation mode	Presence
Timer based discard, with explicit signalling	Network controlled
Timer based discard, without explicit signalling	Network controlled
SDU discard after MaxDAT number of retransmissions	Network controlled

9.7.3.1. Timer based discard, with explicit signalling

This alternative uses a timer based triggering of SDU discard (Timer_Discard). This makes the SDU discard function insensitive to variations in the channel rate and provides means for exact definition of maximum delay. However, the SDU loss rate of the connection is increased as SDUs are discarded.
For every SDU received from a higher layer, timer monitoring of the transmission time of the SDU is started. If the transmission time exceeds a predefined value for a SDU in acknowledged mode RLC, this SDU is discarded in the transmitter and a Move Receiving Window (MRW) command is sent to the receiver so that AMD PDUs carrying that SDU are discarded in the receiver and the receiver window is updated accordingly. Note that when the concatenation function is active, PDUs carrying segments of other SDUs that have not timed out shall not be discarded.
The MRW command is defined as a super-field in the RLC STATUS PDU (see section 9.2), and piggy backed to status information of transmissions in the opposite direction. Therefore, SDU discard variants requiring peer-to-peer signalling are only possible for full duplex connections.

9.7.3.2. Timer based discard, without explicit signalling

This alternative uses the same timer based trigger for SDU discard (Timer_Discard) as the one described in the section 9.7.3.1. The difference is that this discard method does not use any peer-to-peer signalling. For unacknowledged mode RLC, peer-to-peer signalling is never needed. The SDUs are simply discarded in the transmitter, once the transmission time is exceeded. For acknowledged mode RLC, peer-to-peer signalling can be avoided as long as SDU discard is always performed in the transmitter before it is performed in the receiver. As long as the corresponding SDU is eventually discarded in the receiver too, possible retransmission requests of PDU of discarded SDUs can be ignored by the transmitter. The bigger the time difference is between the triggering of the discard condition at the transmitter and the receiver, the bigger the unnecessary buffering need is at the receiver and the more bandwidth is lost on the reverse link due to unnecessary retransmission requests. On the other hand, forward link bandwidth is saved, as no explicit SDU discard signalling is needed.

9.7.3.3. SDU discard after MaxDAT number of retransmissions

This alternative uses the number of retransmissions as a trigger for SDU discard, and is therefore only applicable for acknowledged mode RLC. This makes the SDU discard function dependent of the channel rate. Also, this variant of the SDU discard function strives to keep the SDU loss rate constant for the connection, on the cost of a variable delay. SDU discard is triggered at the transmitter, and a MRW command is necessary to convey the discard information to the receiver, like in the timer based discard with explicit signalling.

9.7.4. The Estimated PDU Counter

The Estimated PDU Counter is a mechanism used for scheduling the retransmission of status reports in the receiver side. With this mechanism, the receiver will send a new Status PDU in which it requests for PUs not yet received. The time between two subsequent status report retransmissions is not fixed, but it is controlled by the Estimated PDU Counter (EPC), which adapt this time to the current bit rate, indicated in the TFI, in order to minimise the delay of the status report retransmission.
The EPC is a counter, which is decremented every transmission time interval with the estimated number of PUs that should have been transmitted during that transmission time interval. When the receiver detects that PDUs are missing it generates and sends a Status PDU to the transmitter and sets the EPC equal to the number of requested PUs.
A special timer, called EPC timer, controls the maximum time that the EPC needs to wait before it will start counting down. This timer starts immediately after a transmission of a retransmission request from the receiver (Status PDU). The EPC timer typically depends on the roundtrip delay, which consists of the propagation delay, processing time in the transmitter and receiver and the frame structure. This timer can also be implemented as a counter, which counts the number of 10 ms radio frames that could be expected to elapse before the first requested AMD PDU is received.
When the EPC is equal to zero and not all of these requested PUs have been received correctly, a new Status PDU will be transmitted and the EPC will be reset accordingly. The EPC timer will be started once more.
The EPC is based on the estimation of the number of PUs that should have been received during a transmission time interval. To estimate this number is easiest done by means of the TFI bits. However, if these bits are lost due to some reason or another, this estimation must be based on something else. A straightforward solution is to base the estimation on the number done in the previous transmission time interval. Only if the rate has changed this estimation is incorrect. Another method of estimating the number of PUs is based on the maximum allowable rate. The consequence of this is that if the estimation is incorrect, the Status PDU is sent too early. Alternatively, the estimation can be based on the lowest possible transmission rate. In this case, if the estimation is incorrect, the Status PDU will most likely be transmitted too late.

9.7.5. Multiple payload units and header compression

The possibility to include multiple payload units (PU) into one RLC AMD PDU provides a way to support variable bit rate services with lower overhead. The method is to be a part of the service capabilities of a UE capable to support user plane traffic in acknowledged mode.

A semi-static sized payload unit is the smallest unit that can be separately addressed for retransmission. The segmentation and concatenation of SDUs into the RLC transmitter buffer is always performed as if there would be only one PU in each AMD PDU. However one AMD PDU may contain also multiple PUs. The header compression is applied to incorporate several PUs effectively into the AMD PDU and it takes place when the transport block size for the next transmission time interval indicated by MAC can accommodate several PUs. The concept of having multiple PUs in one AMD PDU does not remove the need to still have several transport blocks (RLC PDUs) in the transport block set.

Determination of the configuration of PU and PDU sizes is based on the needed transmission rates. The size of the PU is derived directly from the lowest common block size that can be used to compose all the applicable data rates. To support faster transmission rates more payload units are combined into one AMD PDU with header compression function as long as the size of the PDU reaches the optimum. When higher transmission rates are used, several transport blocks (RLC PDUs) are delivered within one transport block set keeping the amount of PUs the same in one AMD PDU.

The method of multiple PUs in one PDU combined with the header compression is especially useful when the lowest needed transmission bit rate cannot be effectively supported with PDUs optimized for the highest transmission bit rate. Figure 3-1 presents a few examples illustrating the applicability of the method. To illustrate the method, Figure 9-19 shows some examples where the RLC AMD PDU size has been set to 320 bits.

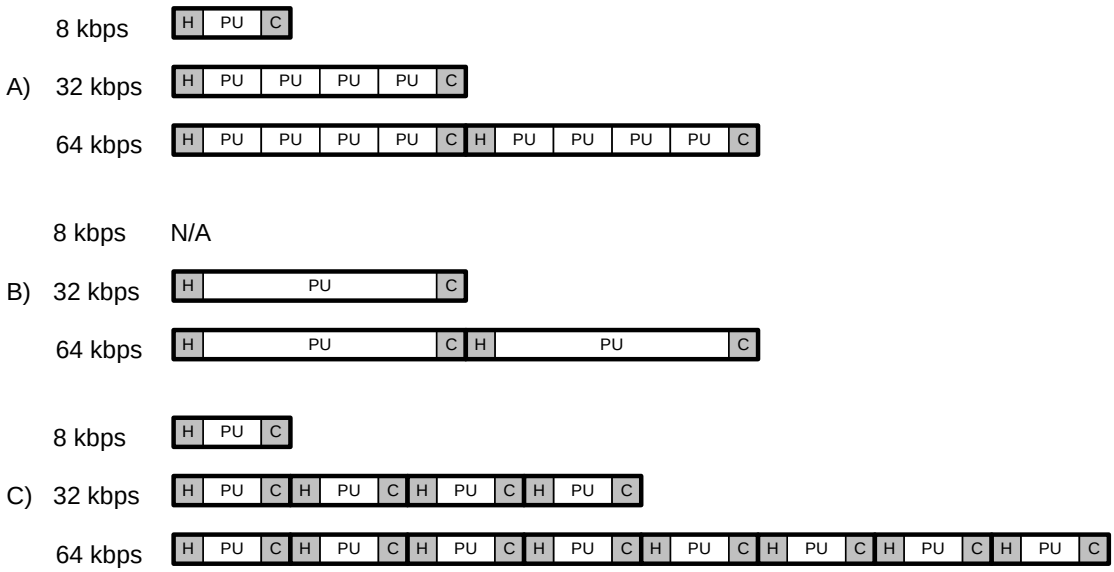


Figure 9-19. Example cases of payload units and PDUs at different transmission rates

Assuming there is a service having a need for variable transmission rate with the maximum above 32 kbps (e.g. 64 kbps) and the minimum of e.g. 8 kbps. With the PU size of 80 bits the flexibility for transmission rates of 8 kbps, 16 kbps or 32 kbps can be achieved by adjusting simply the amount of PUs in one RLC PDU. This also affects on the size of the RLC PDU. While having the optimum sized PDU at 32 kbps, faster transmission rates are constructed by sending several PDUs, each of them containing four PUs (case A).

In case B no data rates below 32 kbps are in the transport format set and the size of the PU is the same as the size of the payload in the RLC PDU. To support transmission bit rates above 32 kbps several PDUs (transport blocks) are sent within one transport block set.

Without the capability to append several PU:s in the PDU the overhead becomes rather high if several transport blocks are sent within one transport block set without header compression at a higher transmission rate (case C).

10. Handling of unknown, unforeseen and erroneous protocol data

A preliminary list of possible error cases is reported below:

and “Send state variable(VT(S)) , and

- LSN of SUFI ACK is not within the value between “Acknowledge state variable(VT(A)) variable(VT(S))”.

In case of error situations the following actions are foreseen:
1) RLC entity should use RESET procedure in case of an unrecoverable error
2) RLC entity should discard invalid PDU
3) RLC entity should notify upper layer of unrecoverable error occurrence in case of failed retransmission

11. Elementary procedures

11.1.Transparent mode data transfer procedure

11.1.1. Purpose

The transparent mode data transfer procedure is used for transferring of data between two RLC peer entities, which are operating in transparent mode. Figure 11-1 below illustrates the elementary procedure for transparent mode data transfer. The sender can be either the UE or the network and the receiver is either the network or the UE.

Figure 11-1. Transparent mode data transfer procedure.

11.1.2. Initiation

The sender initiates this procedure upon a request of transparent mode data transfer from higher layer. When the sender is in data transfer ready state it shall put the data received from the higher layer into TrD PDUs. If needed RLC shall perform segmentation.

Channels that can be used are DTCH, CCCH (uplink only), BCCH, PCCH, SHCCH and SCCH (downlink only). The type of logical channel depends on if the RLC entity is located in the user plane (DTCH) or in the control plane (CCCH/BCCH/SHCCH/PCCH, SCCH). One or several PDUs may be transmitted in each transmission time interval (TTI) and MAC decides how many PDUs shall be transmitted in each TTI.

11.1.2.1. TrD PDU contents to set

The TrD PDU includes a complete SDU or a segment of an SDU. How to perform the segmentation is decided upon when the service is established. No overhead or header is added.

11.1.3. Reception of TrD PDU

Upon reception of a TrD PDU, the receiving entity reassembles (if segmentation was performed) the PDUs into RLC SDUs. RLC delivers the RLC SDUs to the higher layer through the Tr-SAP.

11.1.4. Abnormal cases

11.1.4.1. Undefined SDU size at receiver

If the TrD PDUs are reassembled to a SDU which have a size that is not allowed the SDU shall be discarded.

11.2.Unacknowledged mode data transfer procedure

11.2.1. Purpose

The unacknowledged mode data transfer procedure is used for transferring data between two RLC peer entities, which are operating in unacknowledged mode. Figure 11-2 below illustrates the elementary procedure for unacknowledged mode data transfer. The sender can be either the UE or the network and the receiver is either the network or the UE.

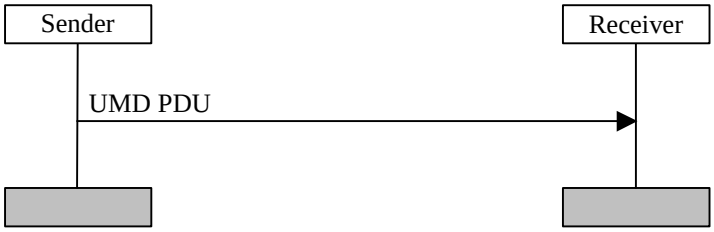


Figure 11-2. Unacknowledged mode data transfer procedure.

11.2.2. Initiation

The sender initiates this procedure upon a request of unacknowledged mode data transfer from higher layer.

When the sender is in data transfer ready state it shall segment the data received from the higher layer into PDUs.

Channels that can be used are DTCH, DCCH, CCCH (downlink only), SHCCH (downlink only). The type of logical channel depends on if the RLC entity is located in the user plane (DTCH) or in the control plane (DCCH/CCCH(downlink only)/SHCCH(downlink only)). One or several PDUs may be transmitted in each transmission time interval (TTI) and MAC decides how many PDUs shall be transmitted in each TTI.

The VT(US) state variable shall be updated for each UMD PDU that is transmitted.

11.2.2.1. UMD PDU contents to set

The Sequence Number field shall be set equal to VT(US).

The Extension bit shall be set to 1 if the next field is a length indicator field, otherwise it shall be set to zero.

One length indicator field shall be included for each end of a SDU that the PDU includes. The length indicator shall be set equal to the number octets between the end of the header fields and the end of the segment. If padding is needed another length indicator shall be added. If the PDU is exactly filled with the last segment of a SDU and there is no room for a length indicator field a length indicator field set to only 0’s shall be included in the next PDU.

11.2.3. Reception of UMD PDU

Upon reception of a UMD PDU the receiver shall update VR(US) state variable according to the received PDU(s).

The PDUs are reassembled into RLC SDUs. If a PDU with sequence number < VR(US) is missing then all SDUs that have segments in this PDU shall be discarded. RLC delivers the RLC SDUs to the higher layer through the UM-SAP.

11.2.4. Abnormal cases

11.2.4.1. Length Indicator value 1111110

Upon reception of an UMD PDU that contains Length Indicator value 1111110 or 111111111111110 ("piggybacked STATUS PDU", in case 7bit or 15 bit Length Indicator field is used, respectively) the receiver shall discard that UMD PDU. This Length Indicator value is not used in unacknowledged mode data transfer.

11.2.4.2. Invalid length indicator value

If the length indicator of a PDU has a value that is larger than the PDU size, the PDU shall be discarded and treated as a missing PDU.

11.3.Acknowledged mode data transfer procedure

11.3.1. Purpose

The acknowledged mode data transfer procedure is used for transferring of data between two RLC peer entities, which are operating in acknowledged mode. Figure 11-3 below illustrates the elementary procedure for acknowledged mode data transfer. The sender can be either the UE or the network and the receiver is either the network or the UE.

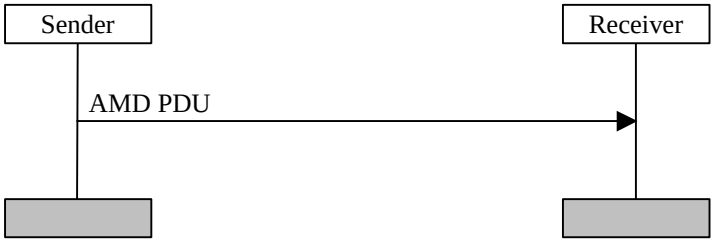


Figure 11-3. Acknowledged mode data transfer procedure.

11.3.2. Initiation

The sender initiates this procedure upon a request of acknowledged mode data transfer from higher layer or upon retransmission of PUs. Retransmitted PUs have higher priority than PUs transmitted for the first time.

The sender is only allowed to retransmit PUs that have been indicated missing by the receiver. There is one exception and that is the last PU that was transmitted can always be retransmitted.

RLC shall segment the data received from the higher layer into PUs. When the sender is in data transfer ready state one or several PUs are included in one AMD PDU, which is sent to the receiver. The PDUs shall be transmitted on the DCCH logical channel if the sender is located in the control plane and on the DTCH if it is located in the user plane. One or several PDUs may be transmitted in each transmission time interval (TTI) and MAC decides how many PDUs shall be transmitted in each TTI.

The VT(DAT) state variables shall be updated for each AMD PDU that is transmitted. The PDU shall not include any PU with Sequence Number $\geq$ VT(MS).

If the poll bit is set in any of the AMD PDUs and the timer Timer_Poll shall be used the sender shall start the timer Timer_Poll.

If timer based SDU discard is used the timer Timer_Discard shall be started when the first segment of a SDU is transmitted.

If the trigger for polling, "Every Poll_PU PU", is used the VT(PU) shall be increased by 1 for each PU that is transmitted.

If the trigger for polling, "Every Poll_SDU SDU", is used the VT(SDU) shall be increased by 1 for each SDU that is transmitted.

11.3.2.1. AMD PDU contents to set

If the PDU is transmitted for the first time, the Sequence Number field shall be set equal to VT(S) and VT(S) shall be updated. In case of multiple in-sequence PUs in PDU the Sequence Number field shows the Sequence Number of the first PU in that PDU.

The setting of the Polling bit is specified in section 11.3.2.1.1.

Extended Header field is needed when out-of-sequence PUs are placed in a PDU or when the rest of a PDU, which is not filled with PUs, is equal or larger than the size of a PU.

One length indicator field shall be included for each end of a SDU that the PDU includes. The length indicator shall be set equal to the number of octets between the end of the header fields and the end of the segment. If the PDU is exactly filled with the last segment of a SDU and there is no room for a length indicator field a length indicator field set to only 0's shall be included in the next PDU. How to perform the segmentation of a SDU is specified in subsection 11.3.2.1.2

11.3.2.1.1. Setting of the Polling bit

The Polling bit shall be set to 1 if any of following conditions are fulfilled except when the poll prohibit function is used and the timer Timer_Poll_Prohibit is active (the different triggers are described in 9.7.4):

1. Last PU in buffer is used and the last PU available for transmission is transmitted.

2. Last PU in retransmission buffer is used and the last PU to be retransmitted is transmitted.
3. Poll timer is used and timer Timer_Poll has expired.
4. Every Poll_PU PU is used and when VT(PU)=Poll_PU.
5. Every Poll_SDU is used and VT(SDU)=Poll_SDU and the PDU contains the last segment that SDU.
6. Poll_Window% of transmission window is used and
$$1 - \frac{(Window\_Size + VT(MS) - VT(S)) \bmod Window\_Size}{Window\_Size} > Poll\_Window.$$
7. Timer based polling is used and Timer_Poll_Periodic has expired.
8. Poll prohibit shall be used, the timer Timer_Poll_Prohibit has expired and one or several polls were prohibited during the time Timer_Poll_Prohibit was active.

11.3.2.1.2. Segmentation of a SDU

Upon reception of a SDU, RLC shall segment the SDU to fit into the fixed size of a PU. The segments are inserted in the data field of a PU. A length indicator shall be added to each PU that includes a border of a SDU, i.e. if a PU does not contain a length indicator the SDU continues in the next PU. The length indicator indicates where the border occurs in the PU. The data after the indicated border can be either a new SDU, padding or piggybacked information. If padding or piggybacking is added another length indicator shall be added, see section 9.2.2.8.

11.3.3. Reception of AMD PDU by the receiver

Upon reception of a AMD PDU the receiver shall update VR(R), VR(H) and VR(MR) state variables according to the received PU(s).

If any of the PUs include a Polling bit set to 1 the STATUS PDU transfer procedure shall be initiated.

If the detection of missing PU(s) shall be used and the receiver detects that a PU is missing the receiver shall initiate the STATUS PDU transfer procedure.

If timer based SDU discard without explicit signalling is used and a missing PU is detected the timer Timer_Discard is started.

11.3.4. Abnormal cases

11.3.4.1. Timer_Poll timeout

Upon expiry of the Timer_Poll the sender shall retransmit the poll. The poll can be retransmitted in either a new PDU or a retransmitted PDU.

11.3.4.2. Receiving a PU outside the receiving window

Upon reception of a PU with SN<VR(R) or SN≥VR(MR) the receiver shall discard the PU. The poll bit shall be considered even if a complete PDU is discarded.

11.3.4.3. Timer_Discard timeout

11.3.4.3.1. SDU discard with explicit signalling

Upon expiry of Timer_Discard the sender shall initiate the SDU discard with explicit signalling procedure.

11.3.4.3.2. SDU discard without explicit signalling

Upon expiry of the Timer_Discard on the sender side the sender shall discard all PUs that contains a segment of the associated SDU. If the concatenation function is active, PDUs carrying segments of other SDUs that have not timed out shall not be discarded. The state variable variables VT(A) and VT(MS) shall be updated.

Upon expiry of the Timer_Discard on the receiver side the receiver shall discard all PUs that contains a segment of the associated SDU. If the concatenation function is active, PDUs carrying segments of other SDUs that have not timed out shall not be discarded. The state variable variables VR(R), VR(H) and VR(MR) shall be updated.

11.3.4.4. VT(DAT) > MaxDAT

If SDU discard after MaxDAT number of retransmission is used and VT(DAT) > MaxDAT for any PU the sender shall initiate the SDU discard with explicit signalling procedure.

If the SDU discard is not used the sender shall initiate the RLC reset procedure when VT(DAT) > MaxDAT.

11.3.4.5. Invalid length indicator value

If the length indicator of a PU has a value that is larger than the PU size, the PU shall be discarded and treated as a missing PU.

11.4.RLC reset procedure

11.4.1. Purpose

The RLC reset procedure is used to reset two RLC peer entities, which are operating in acknowledged mode. Figure 11-4 below illustrates the elementary procedure for a RLC reset. The sender can be either the UE or the network and the receiver is either the network or the UE.

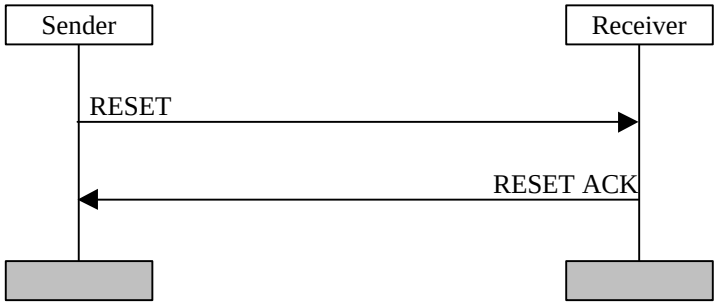


Figure 11-4. RLC reset procedure.

11.4.2. Initiation

The procedure shall be initiated when a protocol error occurs.

The sender sends the RESET PDU when it is in data transfer ready state and enters reset pending state. The sender shall start the timer Timer_RST and increase VT(RST) with 1. The RESET PDU shall be transmitted on the DCCH logical channel if the sender is located in the control plane and on the DTCH if it is located in the user plane.

The RESET PDU has higher priority than data PDUs.

11.4.2.1. RESET PDU contents to set

The size of the RESET PDU shall be equal to one of the allowed PDU sizes.

11.4.3. Reception of the RESET PDU by the receiver

Upon reception of a RESET PDU the receiver shall respond with a RESET ACK PDU.

The RESET ACK PDU shall be transmitted on the DCCH logical channel if the sender is located in the control plane and on the DTCH if it is located in the user plane.

The RESET ACK PDU has higher priority than data PDUs.

11.4.3.1. RESET ACK PDU contents to set

The size of the RESET ACK PDU shall be equal to one of the allowed PDU sizes.

11.4.4. Reception of the RESET ACK PDU by the sender

Upon reception of a RESET ACK the Timer_RST shall be stopped and VT(RST) shall be reset. The sender shall enter data transfer ready state.

11.4.5. Abnormal cases

11.4.5.1. Timer_RST timeout

Upon expiry of Timer_RST the sender shall retransmit the RESET PDU and increase VT(RST) with 1.

11.4.5.2. VT(RST) ≥ MaxRST

If VT(RST) becomes larger or equal to MaxRST the RRC layer shall be informed.

11.5.STATUS PDU transfer procedure

11.5.1. Purpose

The STATUS PDU transfer procedure is used for transferring of status information between two RLC peer entities, which are operating in acknowledged mode. Figure 11-5 below illustrates the elementary procedure for STATUS PDU transfer. The receiver is the receiver of AMD PDUs and it is either the UE or the network and the sender is the sender of AMD PDUs and it is either the network or the UE.

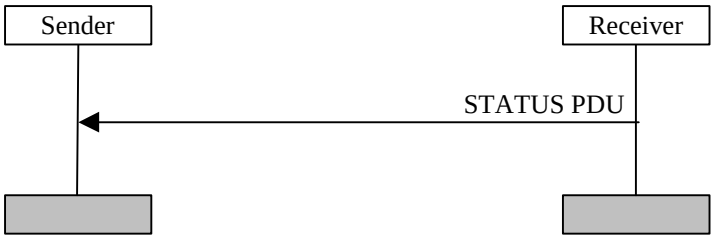


Figure 11-5. STATUS PDU transfer procedure.

11.5.2. Initiation

This procedure is initiated by the receiver in any of following cases:

1. The poll bit in a received AMD PDU is set to 1.
2. Detection of missing PUs is used and a missing PU is detected.
3. The timer based STATUS PDU transfer is used and the timer Timer_Status_Periodic has expired.

The receiver shall transmit a STATUS PDU on the DCCH logical channel if the receiver is located in the control plane and on the DTCH if it is located in the user plane.

The STATUS PDU has higher priority than data PDUs.

There are two functions that can prohibit the receiver from sending a STATUS PDU. If any of following conditions are fulfilled the sending of the STATUS PDU shall be delayed, even if any of the conditions above are fulfilled:

1. STATUS PDU prohibit is used and the timer Timer_Status_Prohibit is active.
   The STATUS PDU shall be transmitted after the Timer_Status_Prohibit has expired. The receiver shall send only one STATUS PDU, even if there are several triggers when the timer is running.
2. The EPC mechanism is used and the timer Timer_EPC is active or VR(EP) is counting down.
   The STATUS PDU shall be transmitted after the VR(EP) has reached 0. The receiver send only one STATUS PDU, even if there are several triggers when the timer is active or the counter is counting down.

If the timer based STATUS PDU transfer shall be used and the Timer_Status_Periodic has expired it shall be restarted.

If the EPC meachanism shall be used the timer Timer_EPC shall be started and the VR(EP) shall be set equal to the number PUs requested to be retransmitted.

11.5.2.1. Piggybacked STATUS PDU

It is possible to piggyback a STATUS PDU on an AMD PDU. If a PDU includes padding a piggybacked STATUS PDU can be inserted instead of the padding. The sending of such STATUS PDU does not have to be triggered by the triggers in section 11.5.2. It shall not be sent if any of the prohibit conditions are fulfilled.

11.5.2.2. STATUS PDU contents to set

The size of the STATUS PDU shall be equal to one of the allowed PDU sizes. The information that needs to be transmitted can be split into several STATUS PDUs if one STATUS PDU does not accommodate all the information.

Which SUFI fields to use is implementation dependent, but the STATUS PDU shall include information about which PUs have been received and not received.

Padding shall be inserted if the SUFI fields do not fill the entire STATUS PDU. If the PDU contains padding the last SUFI field shall be either an Acknowledgement super-field or a No More super-field.

11.5.3. Reception of the STATUS PDU by the sender

The sender shall upon reception of the STATUS PDU/piggybacked STATUS PDU update the state variables VT(A) and VT(MS) according to the received STATUS PDU/piggybacked STATUS PDU.

If the STATUS PDU includes negative acknowledged PUs the acknowledged data transfer procedure shall be initiated and the PUs shall be retransmitted. Retransmitted PUs have higher priority than new PUs.

11.5.4. Abnormal cases

11.5.4.1. EPC reaches zero and the requested PUs have not been received

If the EPC mechanism is used and VR(EP) has reached 0 and not all PUs requested for retransmission have been received the receiver shall:

Retransmit the STATUS PDU. The retransmitted STATUS PDU may contain new or different SUFI fields in order to indicate that some PUs have been received and that some new have been lost.

11.6.SDU discard with explicit signalling procedure

11.6.1. Purpose

An SDU can be discarded with explicit signalling when MaxDAT number of retransmissions is reached or the transmission time exceeds a predefined value (Timer_Discard) for a SDU in acknowledged mode RLC. Move Receiving Window (MRW) command is sent to the receiver so that AMD PDUs carrying that SDU are discarded in the receiver and the receiver window is updated accordingly. Note that when the concatenation function is active, PDUs carrying segments of other SDUs that have not timed out shall not be discarded.

The MRW command is defined as a super-field in the RLC STATUS PDU, and piggybacked to status information of transmissions in the opposite direction.

Figure 11-6 below illustrates the elementary procedure for SDU discard with explicit signalling. The sender is the sender of AMD PDUs and it is either the UE or the network and the receiver is the receiver of AMD PDUs and it is either the network or the UE.

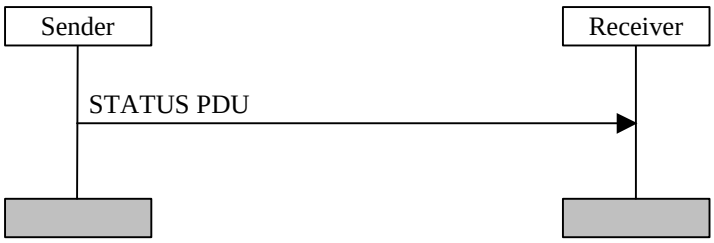


Figure 11-6. SDU discard with explicit signalling.

11.6.2. Initiation

This procedure is initiated by the sender when the following conditions are fulfilled:

1. SDU discard with explicit signalling is used.
2. MaxDAT number of retransmissions is reached or Timer_Discard expires for a SDU in acknowledged mode RLC.

The sender shall discard all PUs that contain a segment of the associated SDU. If the concatenation function is active, PDUs carrying segments of other SDUs that have not timed out shall not be discarded. The state variable variables VT(A) and VT(MS) shall be updated.

The sender shall transmit a STATUS PDU on the DCCH logical channel if the sender is located in the control plane and on the DTCH if it is located in the user plane.

This STATUS PDU is sent even if the 'STATUS PDU prohibit' is used and the timer 'Timer_Status_Prohibit' is active.

The STATUS PDU has higher priority than data PDUs.

11.6.2.1. Piggybacked STATUS PDU

It is possible to piggyback a STATUS PDU on an AMD PDU. If a PDU includes padding a piggybacked STATUS PDU can be inserted instead of the padding.

11.6.2.2. STATUS PDU contents to set

The size of the STATUS PDU shall be equal to one of the allowed PDU sizes. The information that needs to be transmitted can be split into several STATUS PDUs if one STATUS PDU does not accommodate all the information.

STATUS PDU shall include the MRW SUFI, other SUFI fields can be used additionally. MRW SUFI shall convey information about the discarded SDU(s) to the receiver.

Padding shall be inserted if the SUFI fields do not fill the entire STATUS PDU. If the PDU contains padding the last SUFI field shall be a No More Data super-field.

11.6.3. Reception of the STATUS PDU by the receiver

The receiver shall upon reception of the STATUS PDU/piggybacked STATUS PDU discard PUs and update the state variables VR(R), VR(H) and VR(MR) according to the received STATUS PDU/piggybacked STATUS PDU.

11.6.4. Abnormal cases

11.6.4.1. Obsolete/corrupted MRW command

If the MRW command contains outdated information about the receiver window (receiver window already moved further than MRW command is indicating) the MRW command shall be discarded.

12. Appendix

12.1.Annex A: SDL diagrams

This section contains the SDL diagrams. For Release'99, it is meant for informative purposes only.

*[All the section shall be reviewed when the protocol is defined;
all the SDL diagrams presented are [FFS]]*

Virtual Process Type Acknowledged_connection 1_Declarations(44)

;

```
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,


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.*/
```

Virtual Process Type Acknowledged_connection 2_Declarations(44)

;

```
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.*/




 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).*/


/*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.



/*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.*/
```

Virtual Process Type Acknowledged_connection 3_Declarations(44)

;

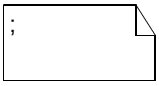
```
DCL

 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.*/


 /*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.*/
```

Virtual Process Type Acknowledged_connection 4_Declarations(44)

;

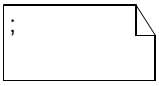
```
TIMER

 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.*/


 /*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)



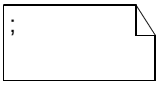
Sdu_am_segmentation	This procedure manages segmentation and concatenation of sdus. It applies pollingin accorance with the toolset functions applied by the higher layer protocols.
	This procedure manages transmission of RLC PDUs across the proper SAP.
Check_if_queue_empty	This procedure checks if there 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.
	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.
	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.

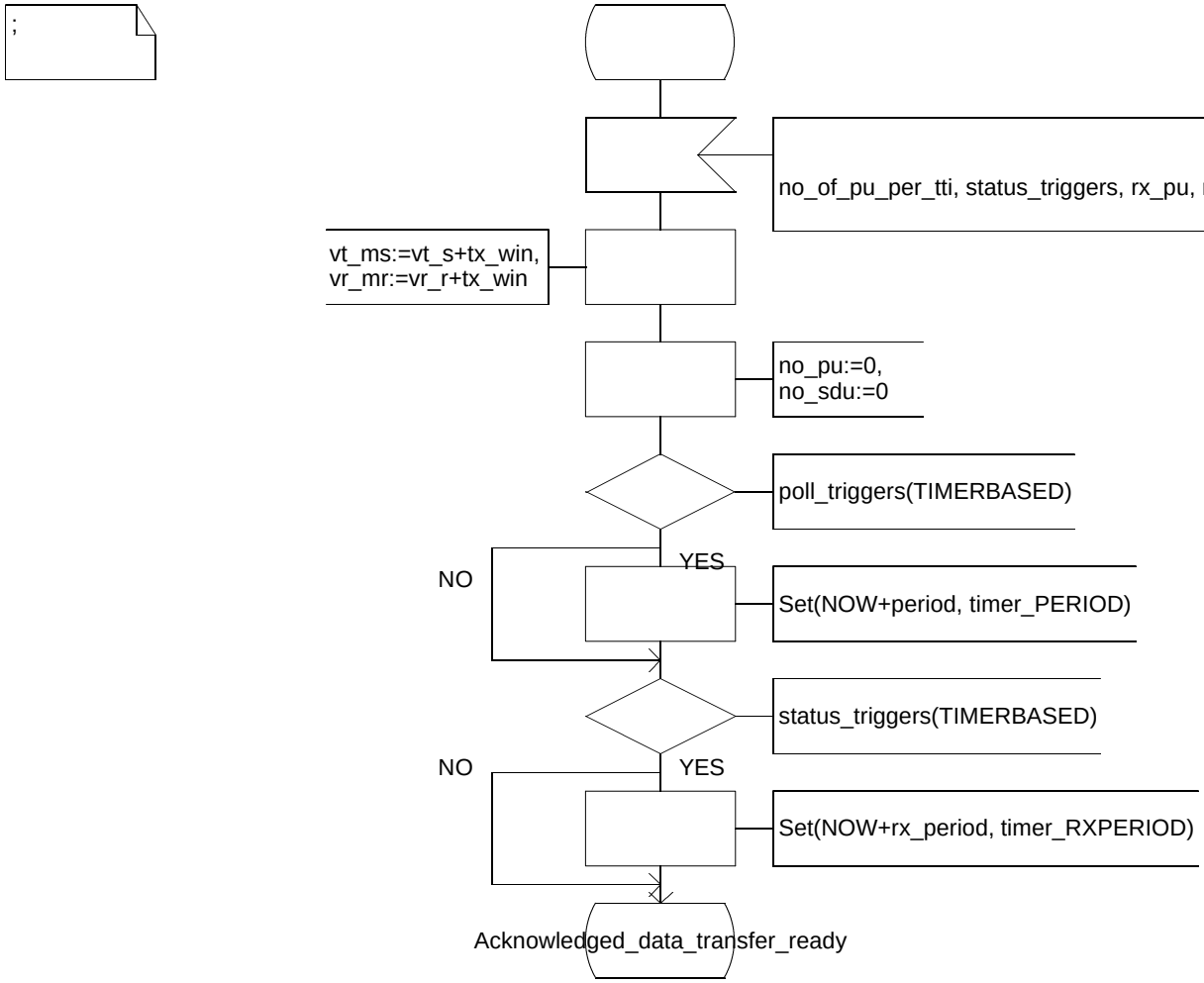
Virtual Process Type Acknowledged_connection 3_Procedures(44)



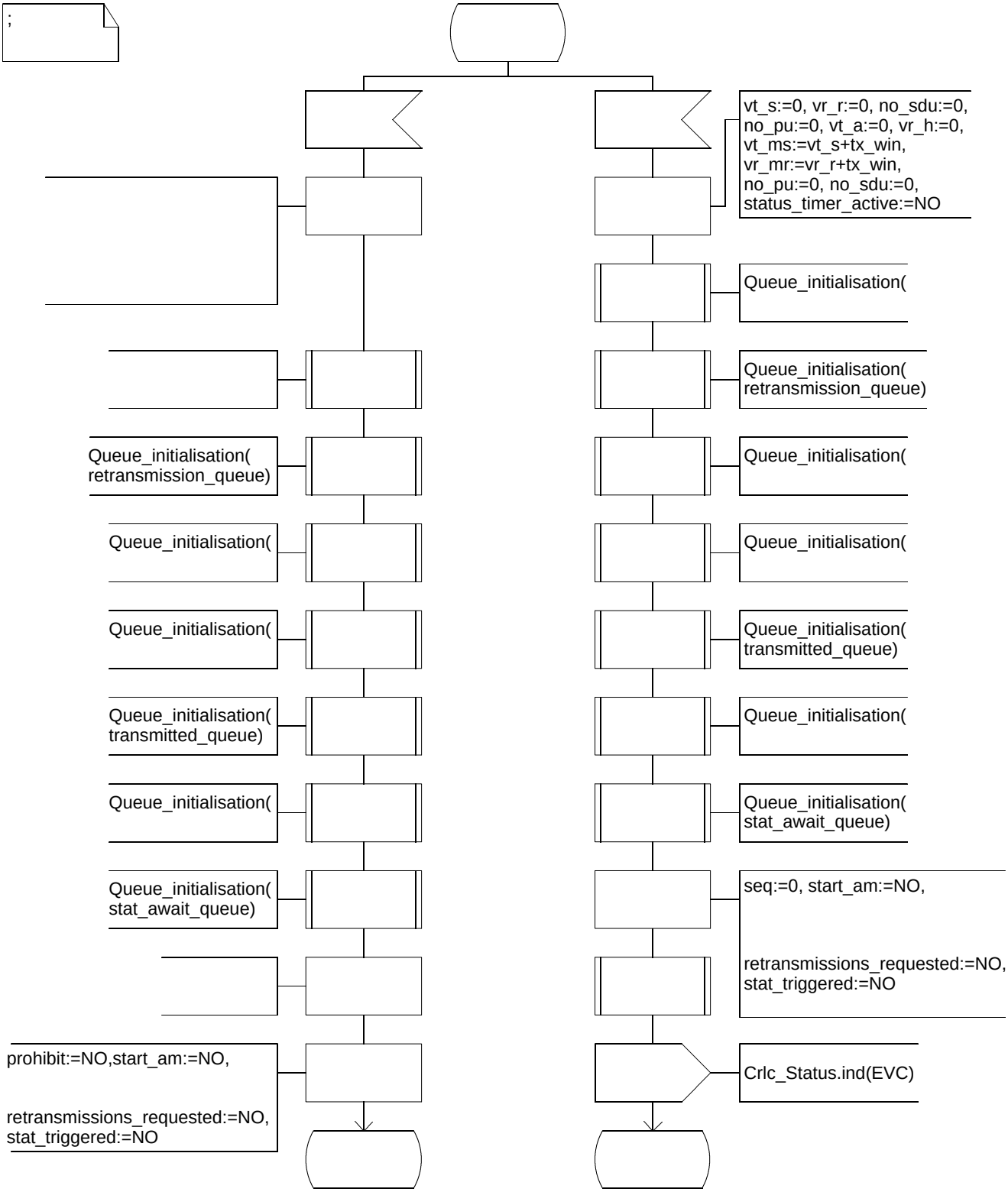
Remove_mui_from_queues	This procedure removes all pdus associated with a given mui from the transmitted_queue.
Remove_all_below_from_queues	This procedure removes all pdus below an identified sequence number from all receiver queues.
	This procedure causes the polling flag to be set in the first PDU 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)

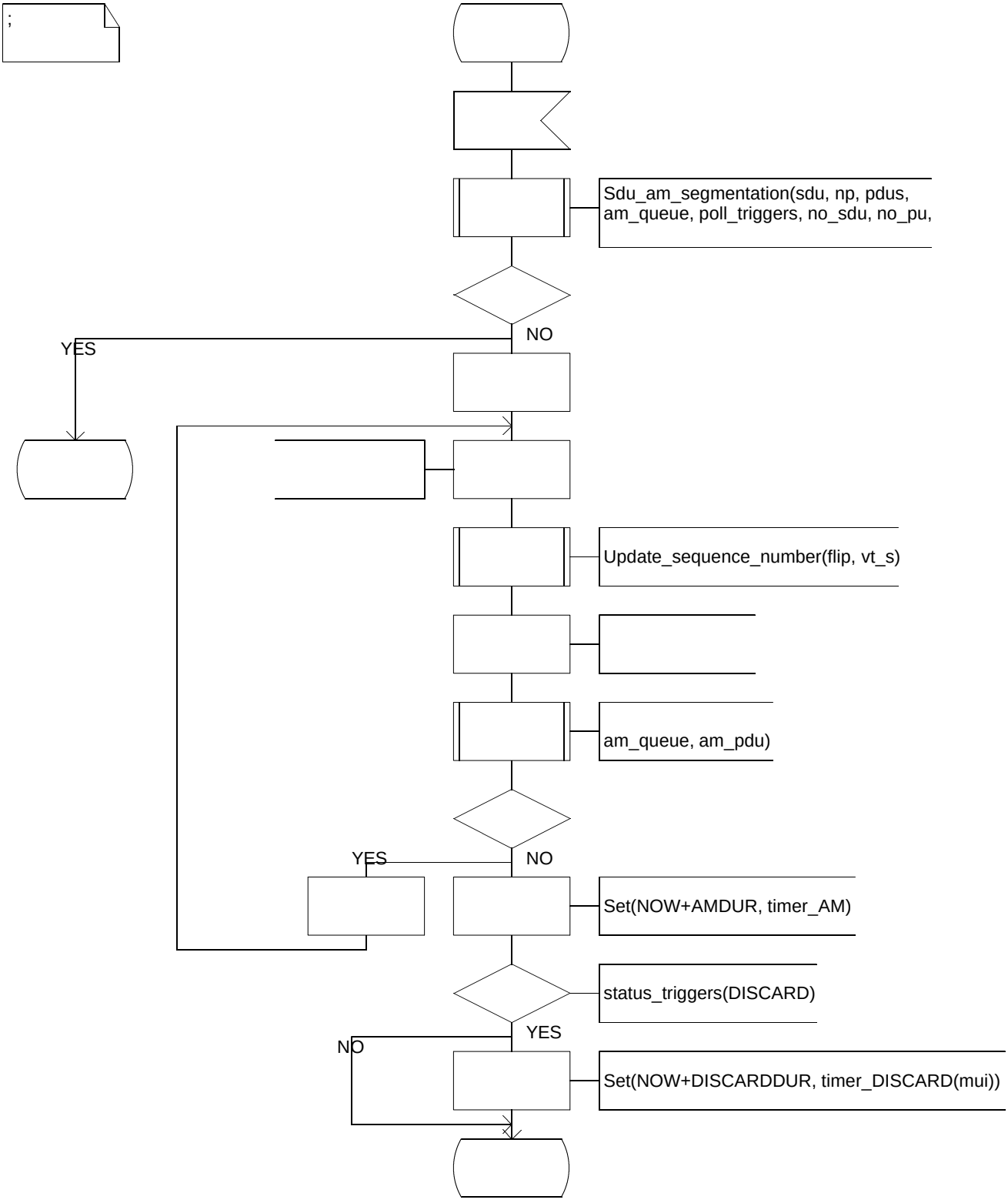
Virtual Process Type Acknowledged_connection 1_Nul(44)



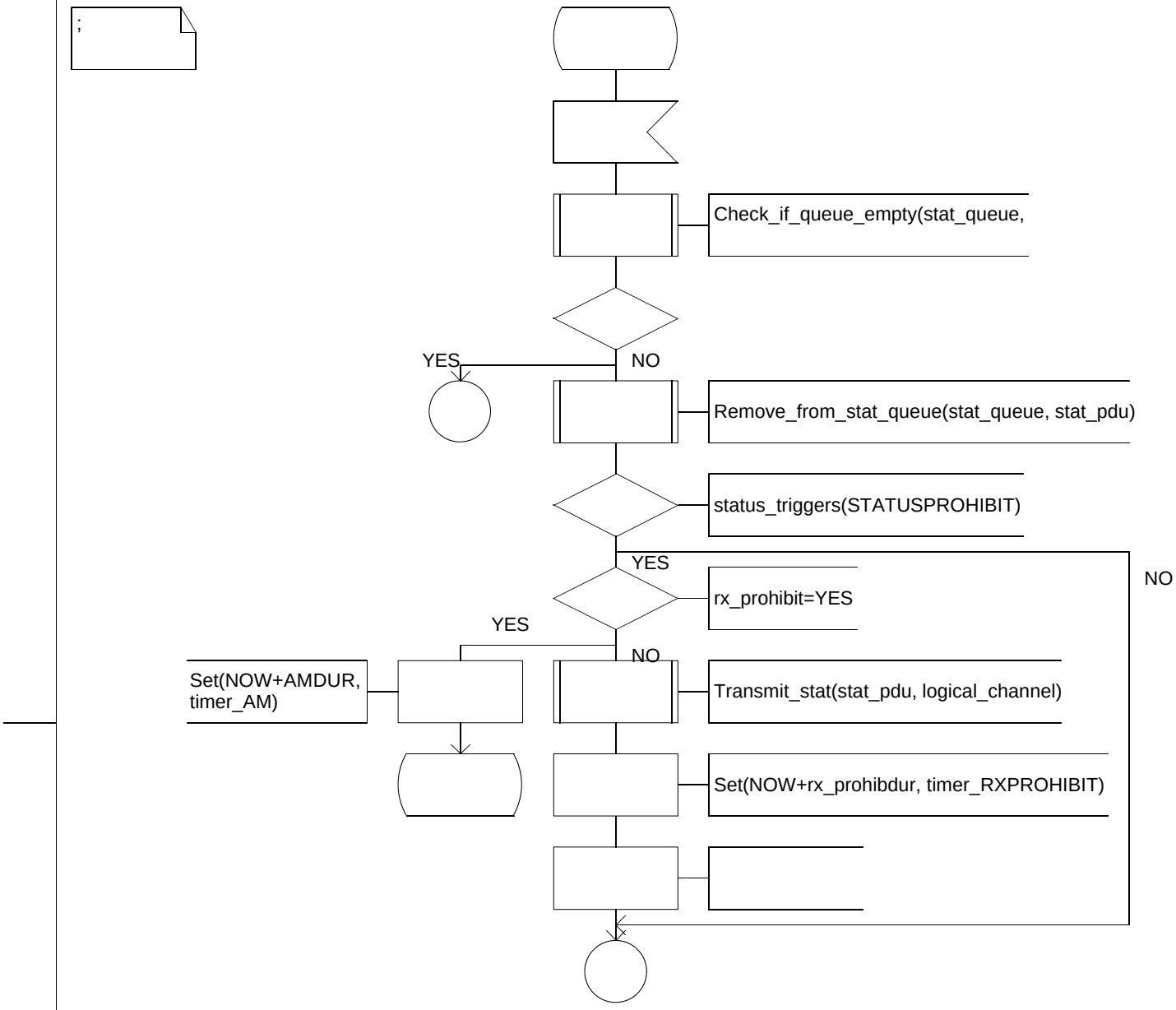
Virtual Process Type Acknowledged_connection 1_AcknowledgedDataTransferReady(44)



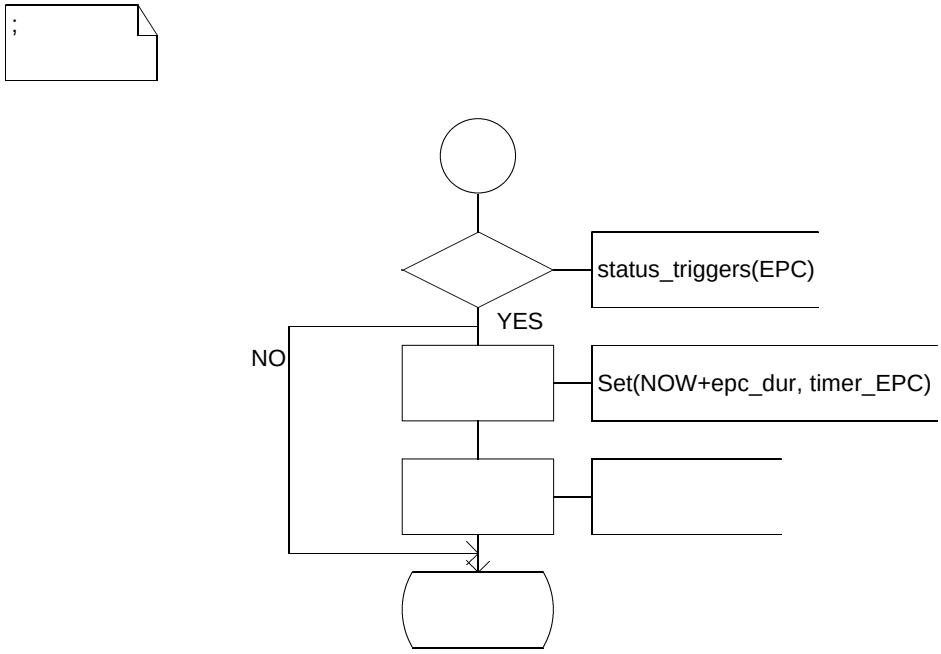
Virtual Process Type Acknowledged_conn1_AcknowledgedDataTransferReady_RlcAmDataReq(44)



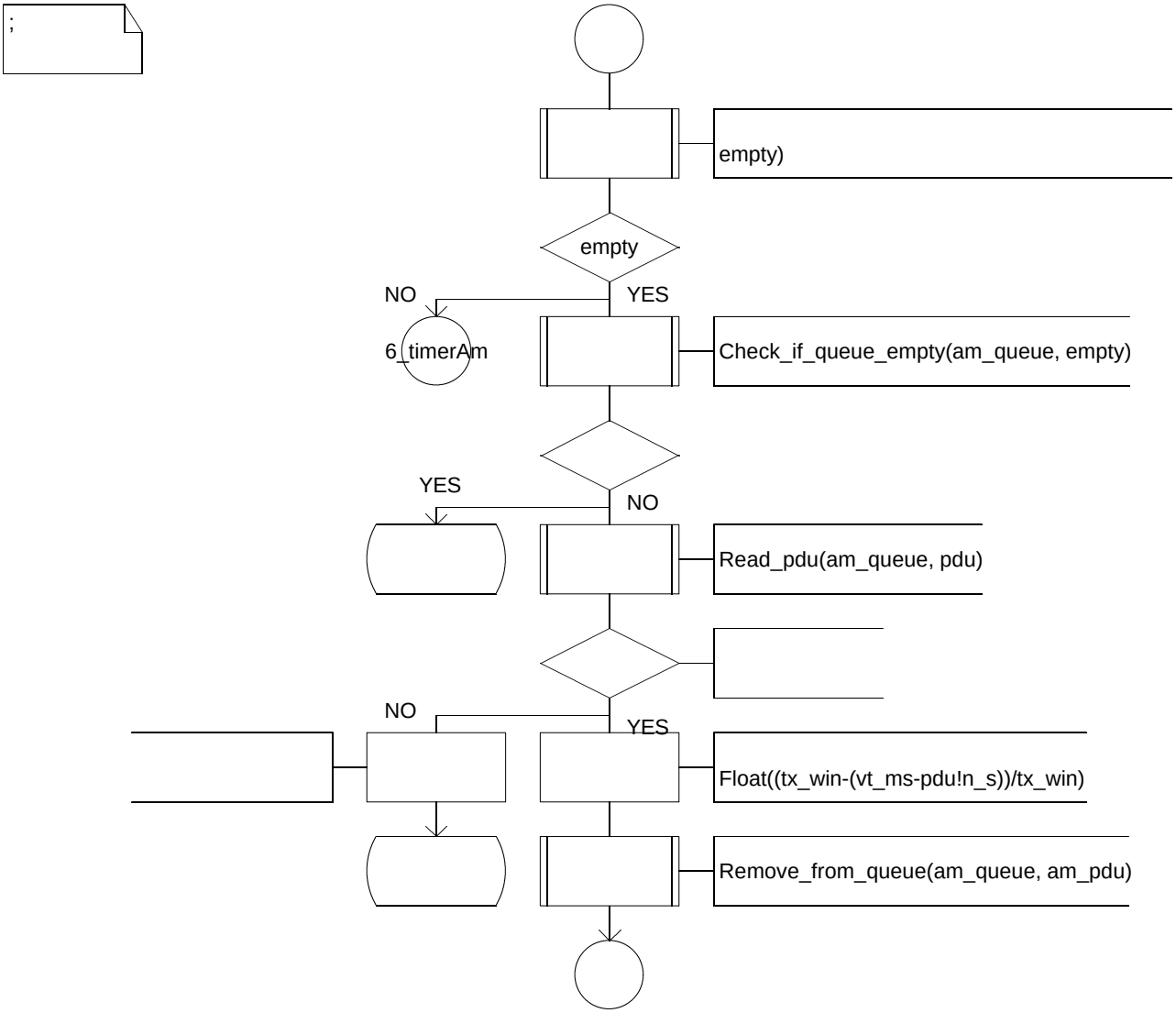
Virtual Process Type Acknowledged_connection 1_AcknowledgedDataTransferReady_timerAm(44)



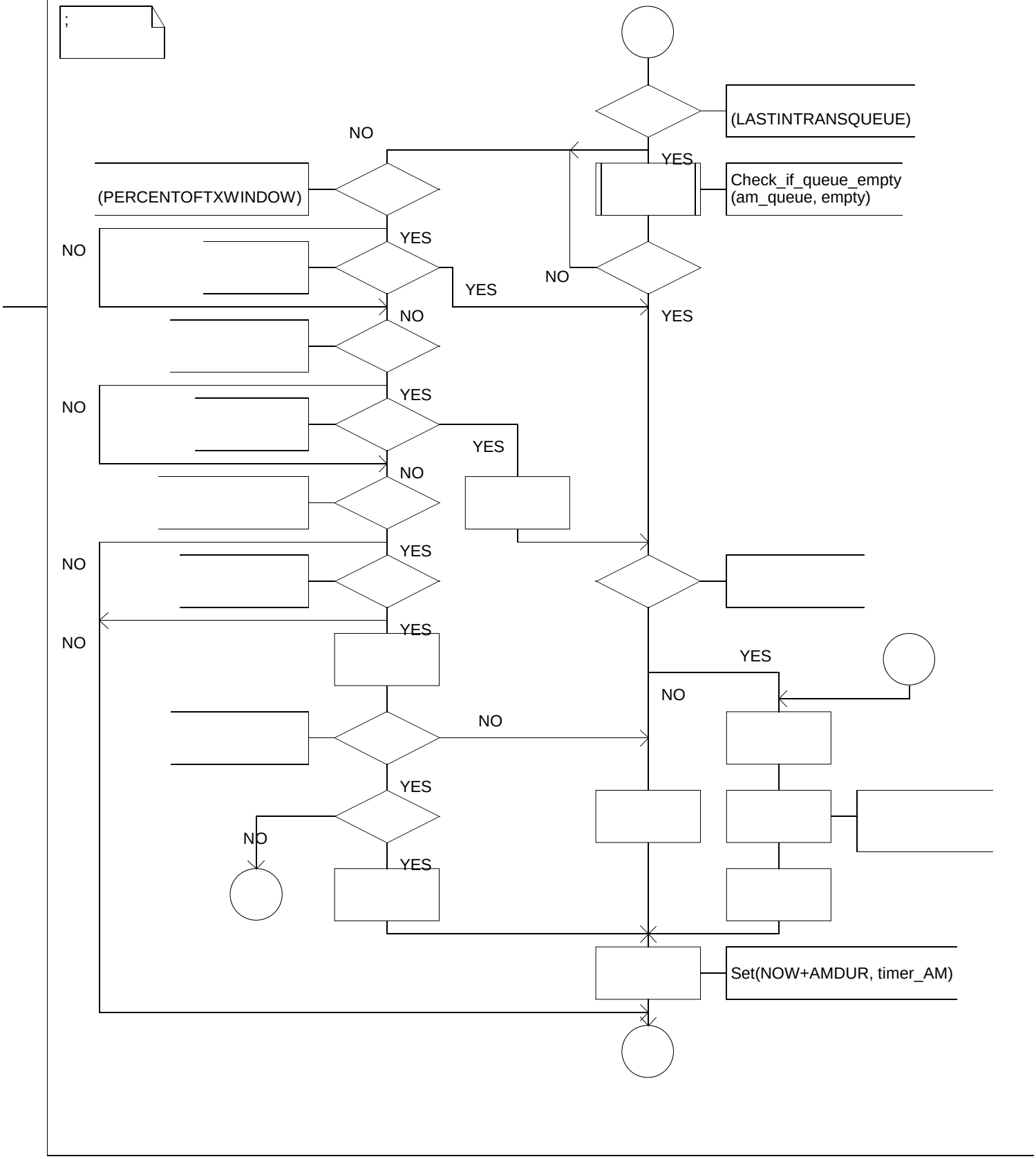
Virtual Process Type Acknowledged_connection 2_AcknowledgedDataTransferReady_timerAm(44)



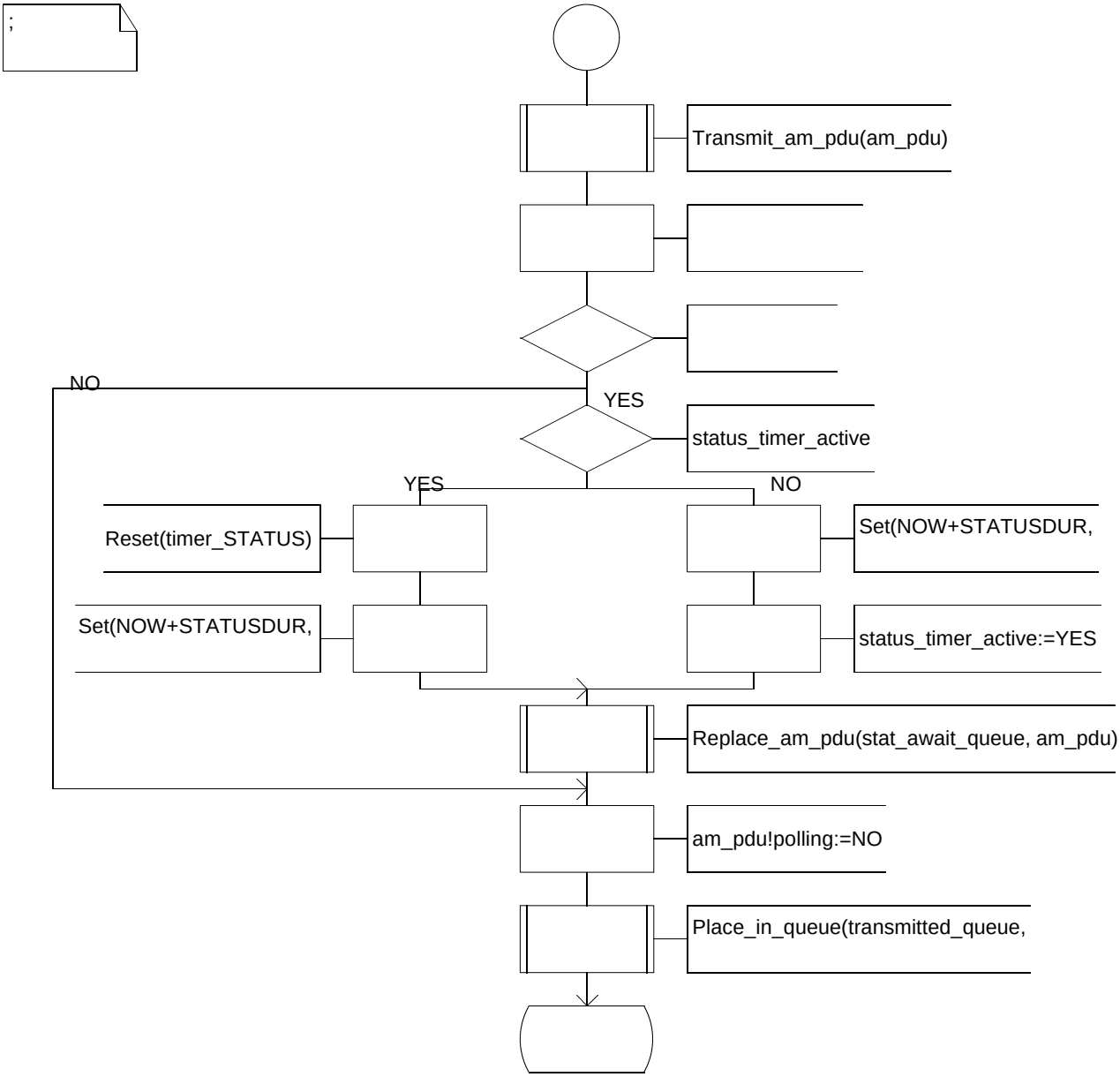
Virtual Process Type Acknowledged_connection 3_AcknowledgedDataTransferReady_timerAm(44)



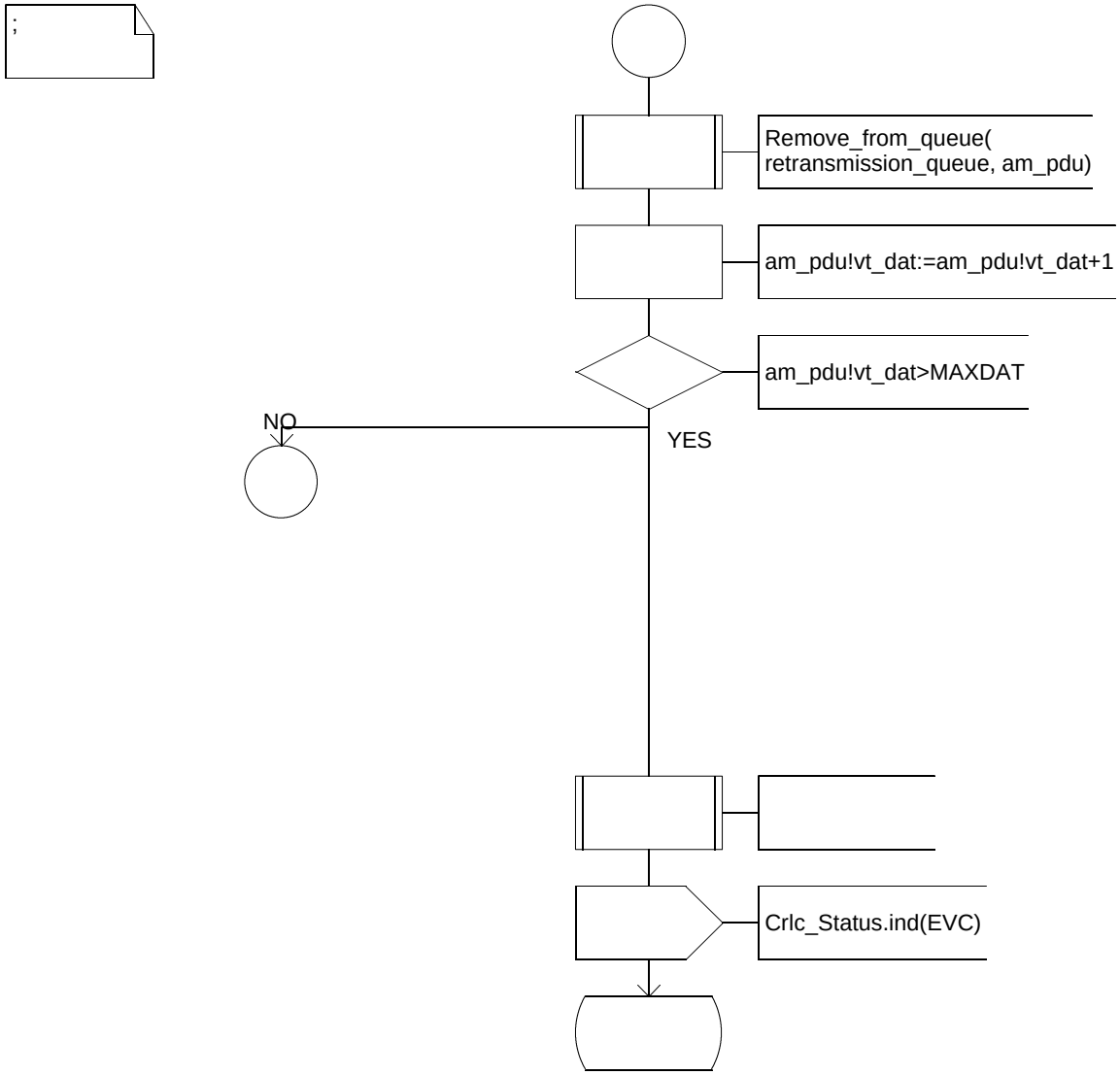
Virtual Process Type Acknowledged_connection 4_AcknowledgedDataTransferReady_timerAm(44)



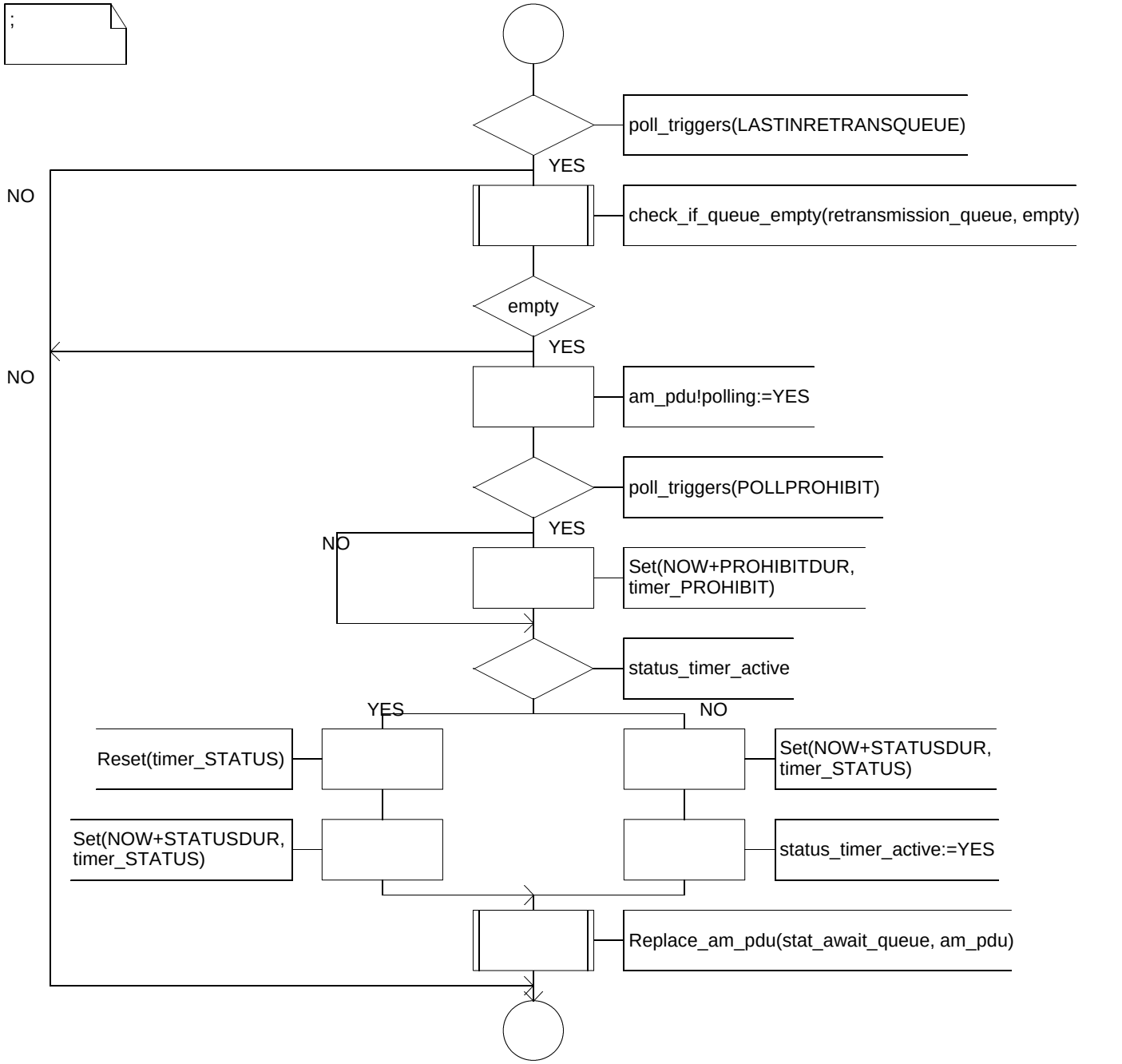
Virtual Process Type Acknowledged_connection 5_AcknowledgedDataTransferReady_timerAm(44)



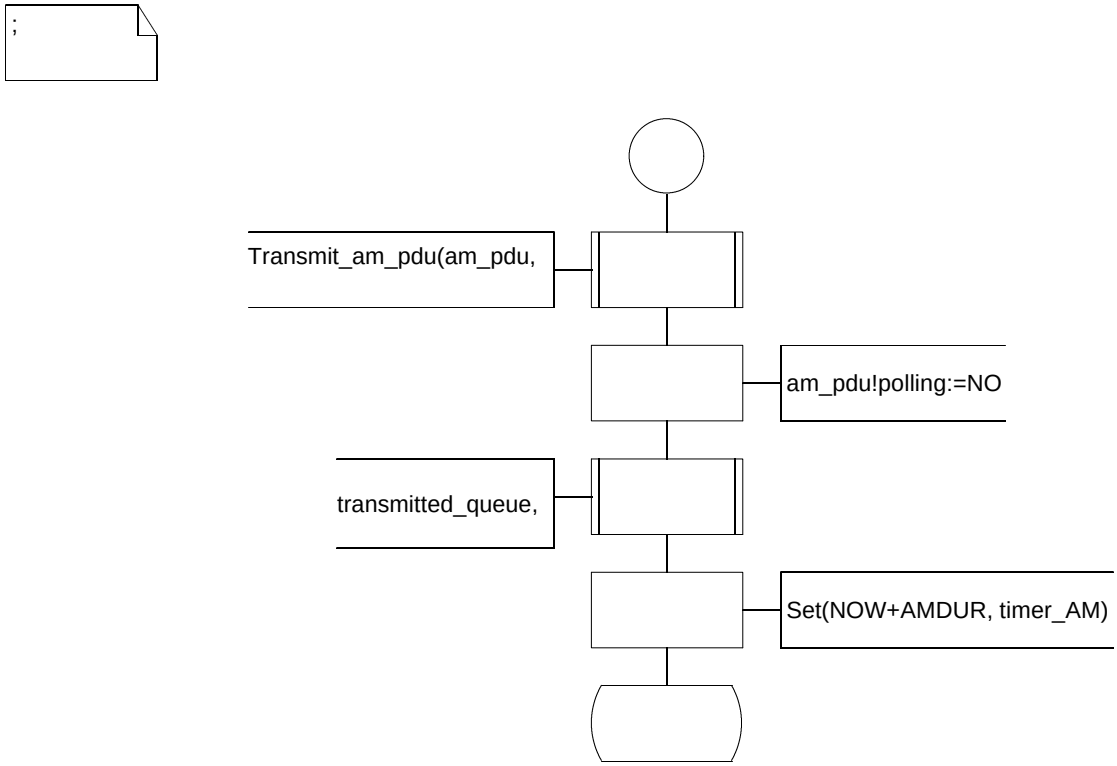
Virtual Process Type Acknowledged_connection 6_AcknowledgedDataTransferReady_timerAm(44)



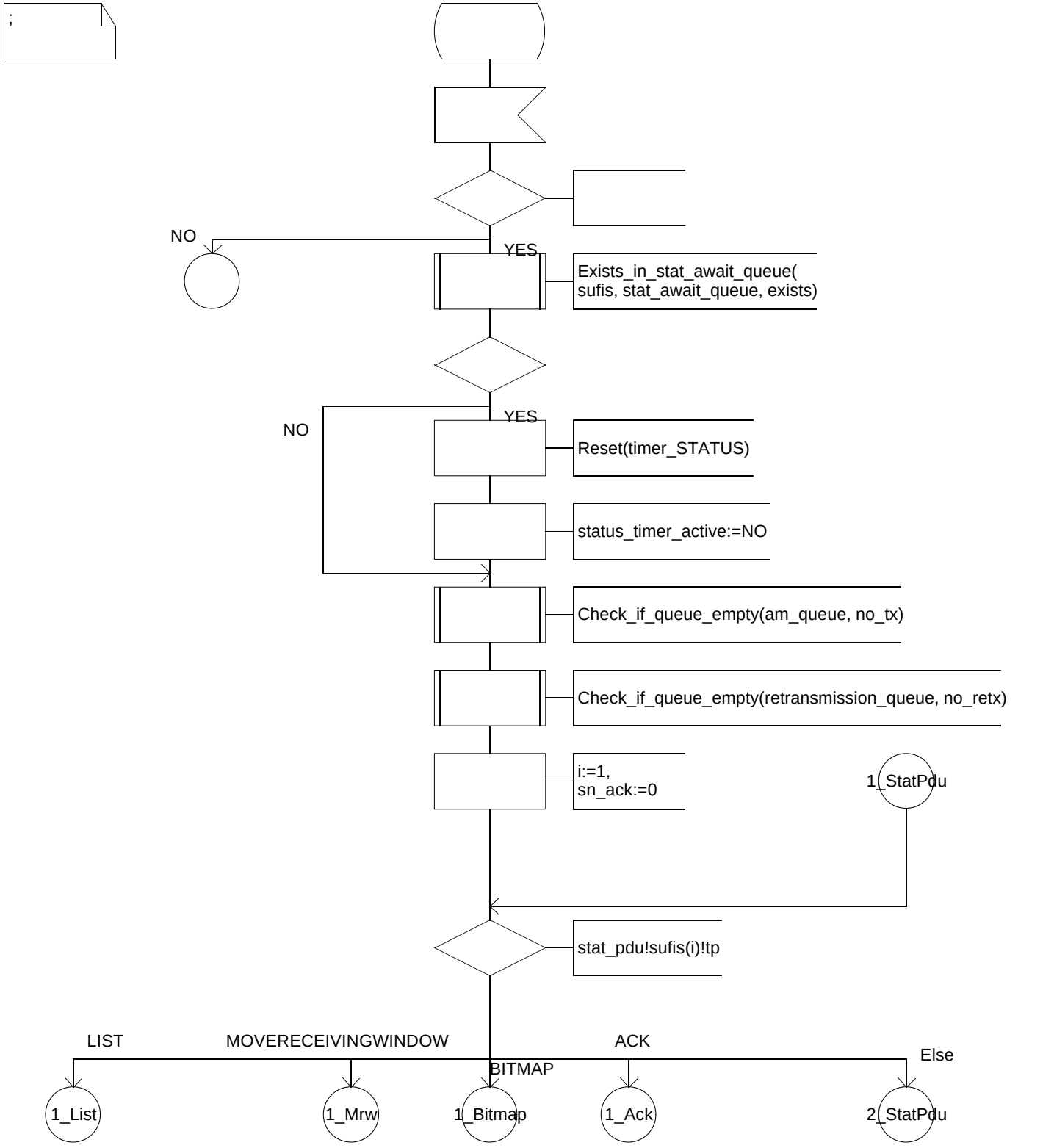
Virtual Process Type Acknowledged_connection 7_AcknowledgedDataTransferReady_timerAm(44)



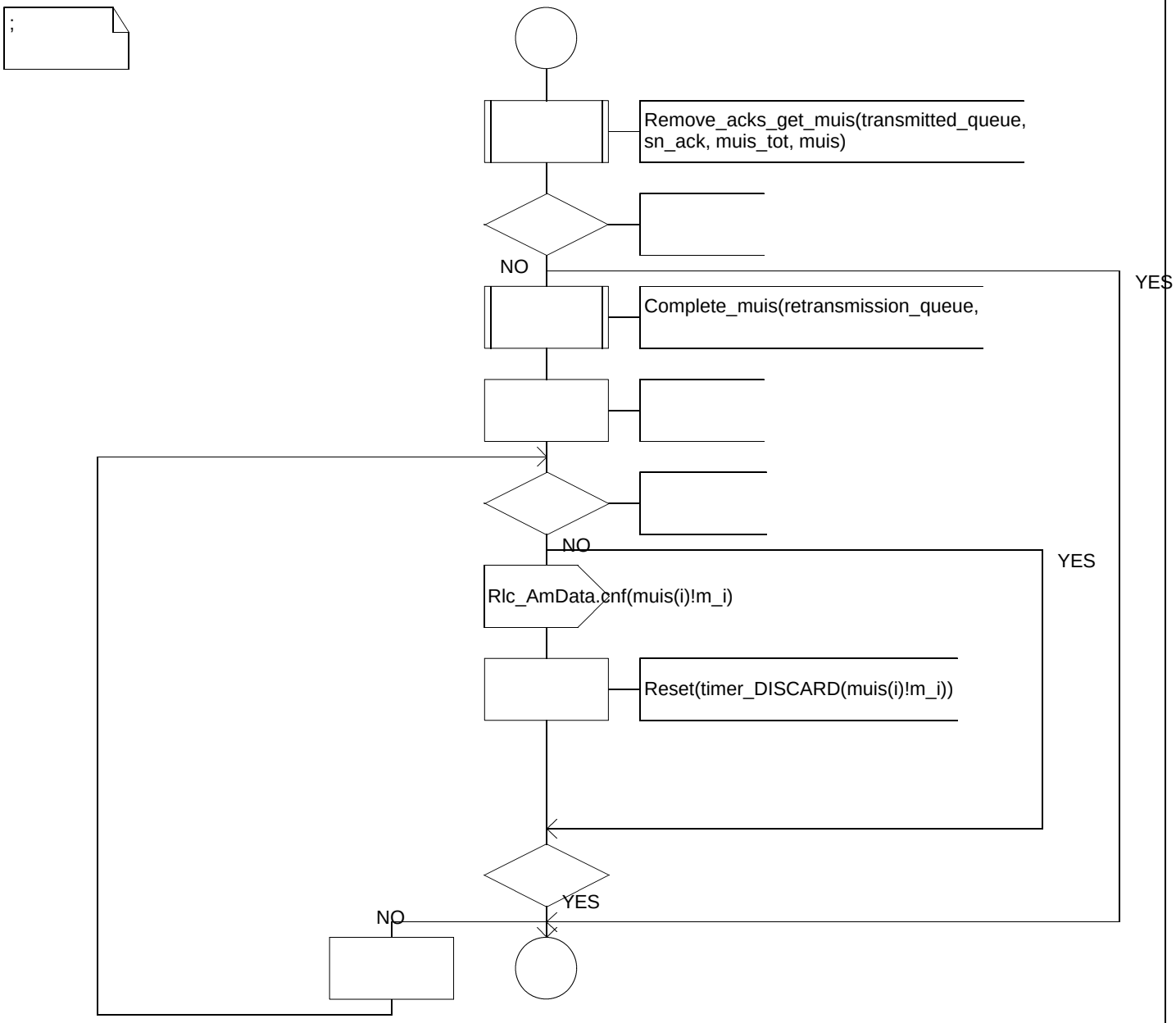
Virtual Process Type Acknowledged_connection 8_AcknowledgedDataTransferReady_timerAm(44)



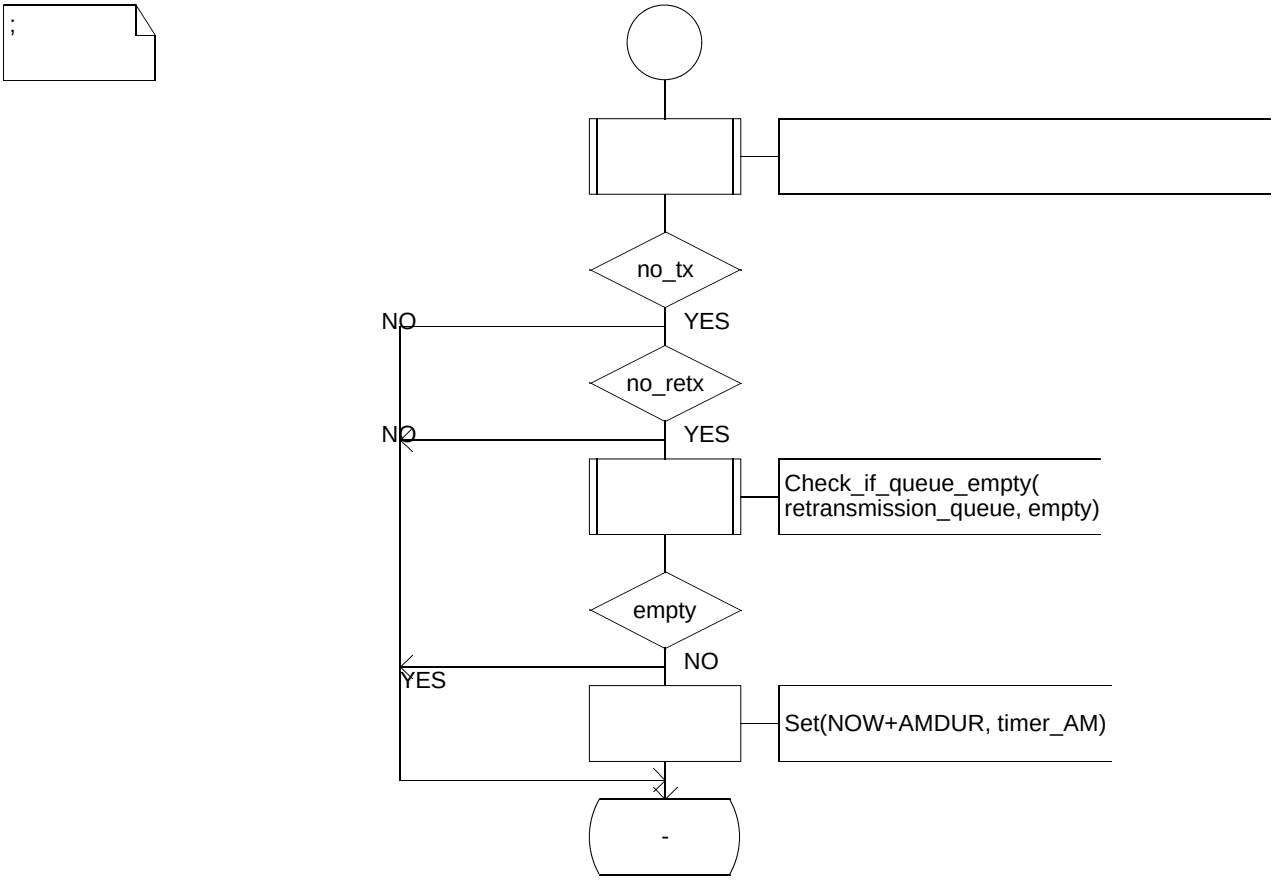
Virtual Process Type Acknowledged_connection 1_AcknowledgedDataTransferReady_StatPdu(44)



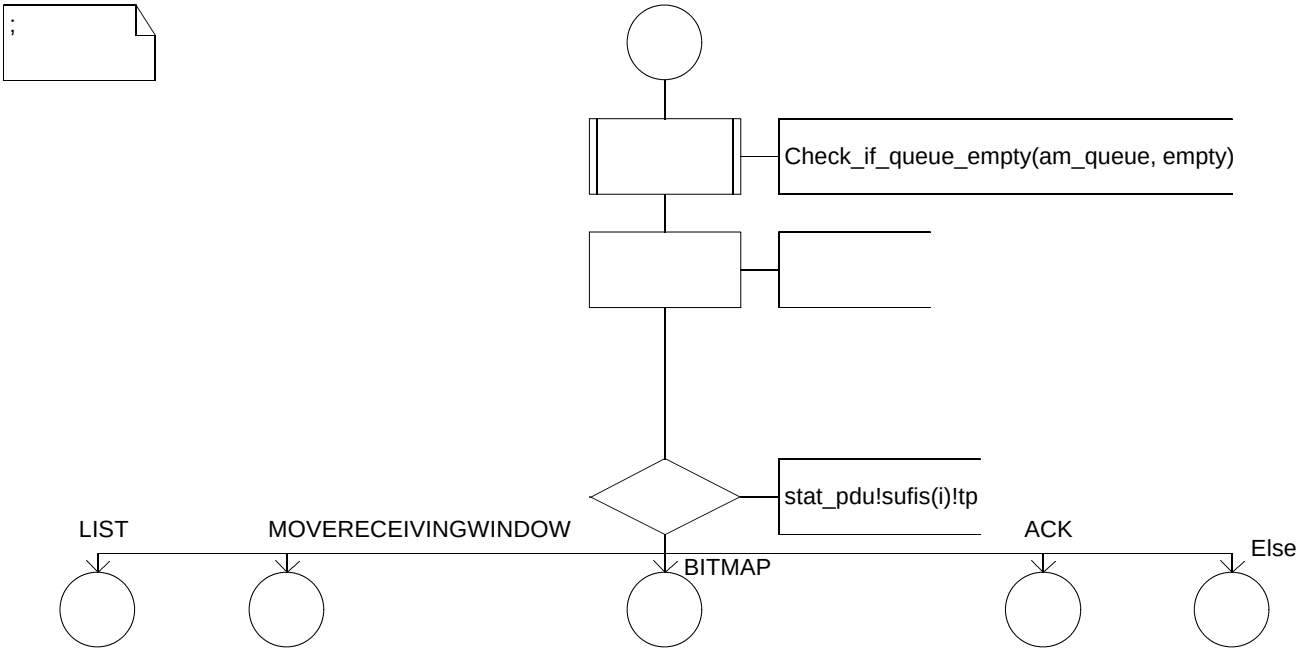
Virtual Process Type Acknowledged_connection 2_AcknowledgedDataTransferReady_StatPdu(44)



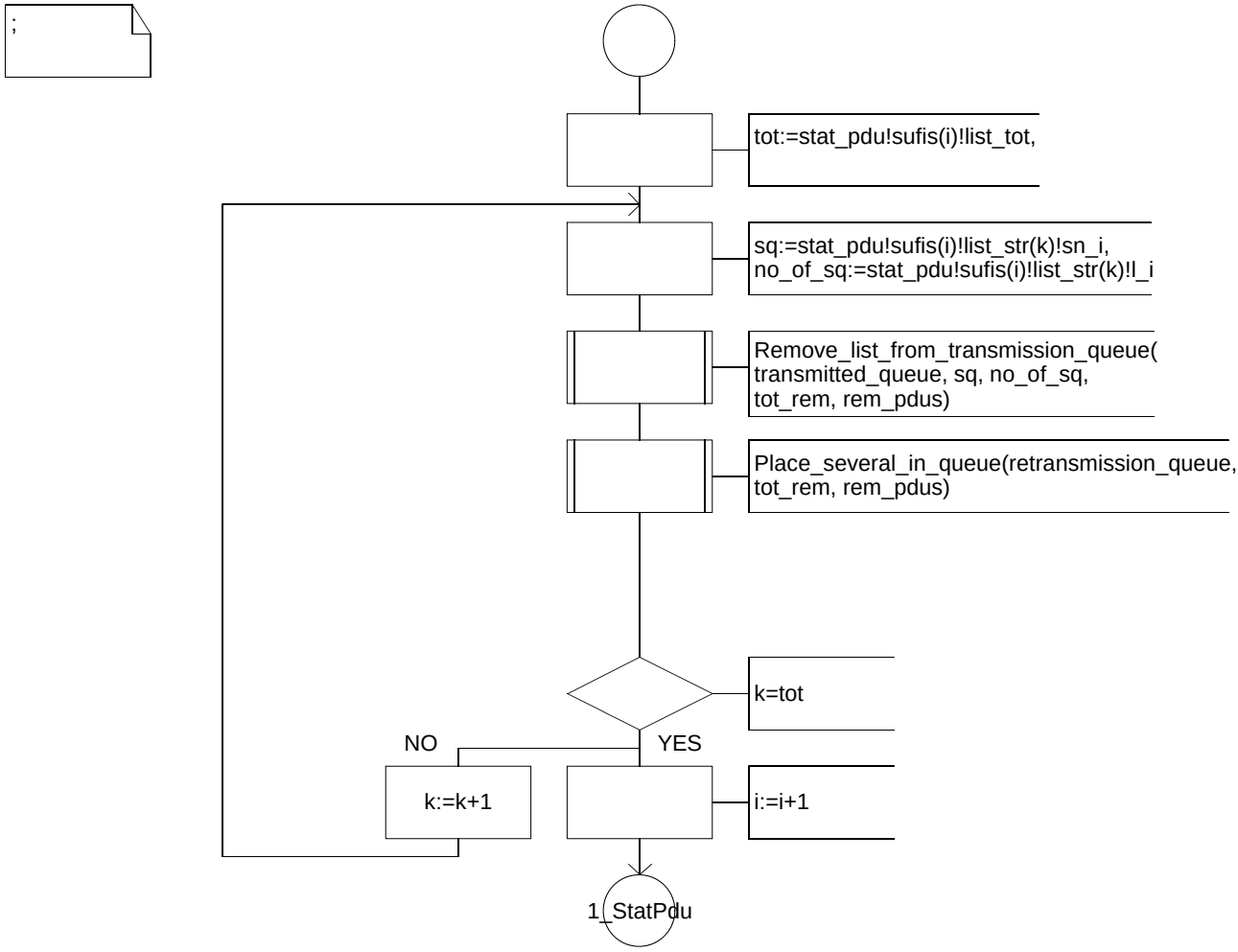
Virtual Process Type Acknowledged_connection 3_AcknowledgedDataTransferReady_StatPdu(44)



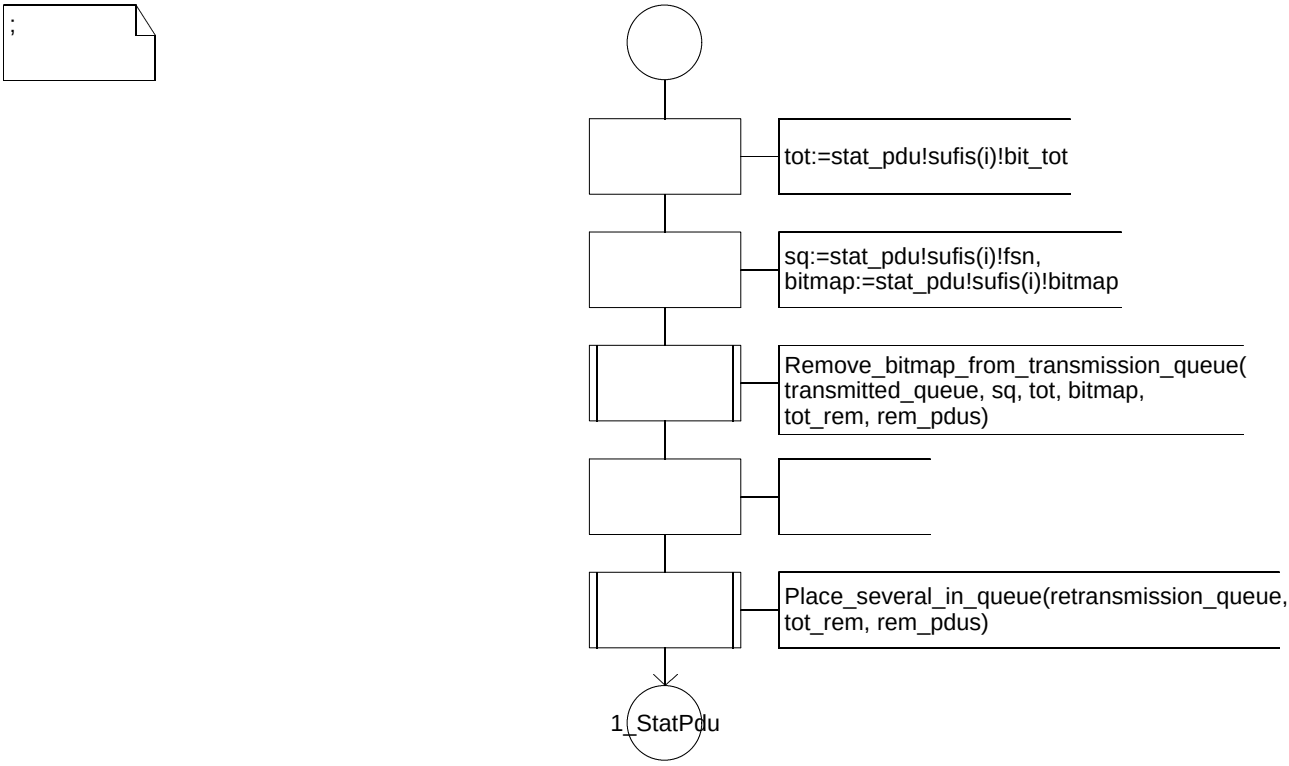
Virtual Process Type Acknowledged_connection 4_AcknowledgedDataTransferReady_StatPdu(44)



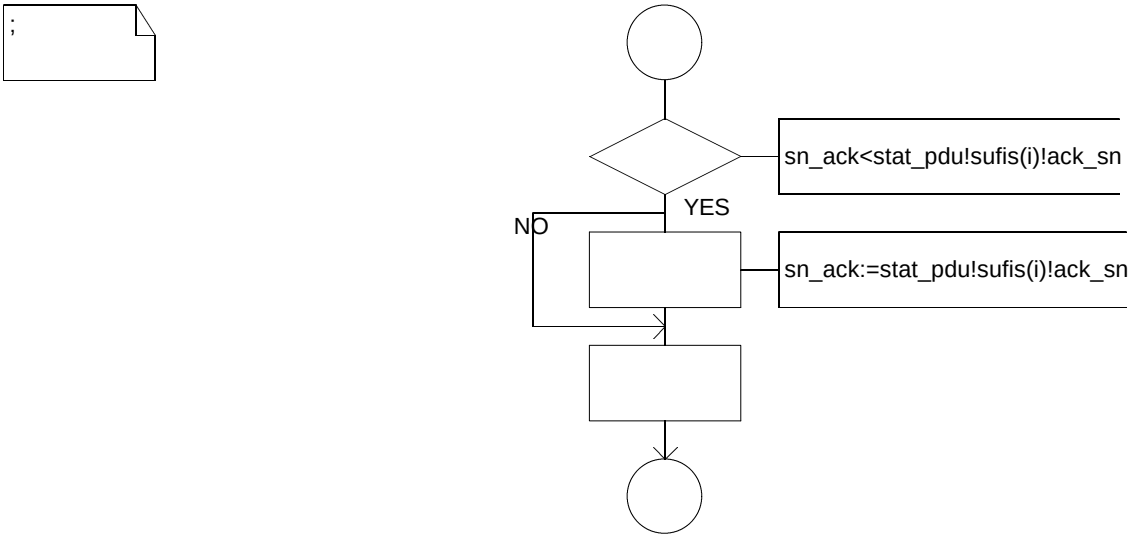
Virtual Process Type Acknowledged_connecti1_AcknowledgedDataTransferReady_StatPduList(44)



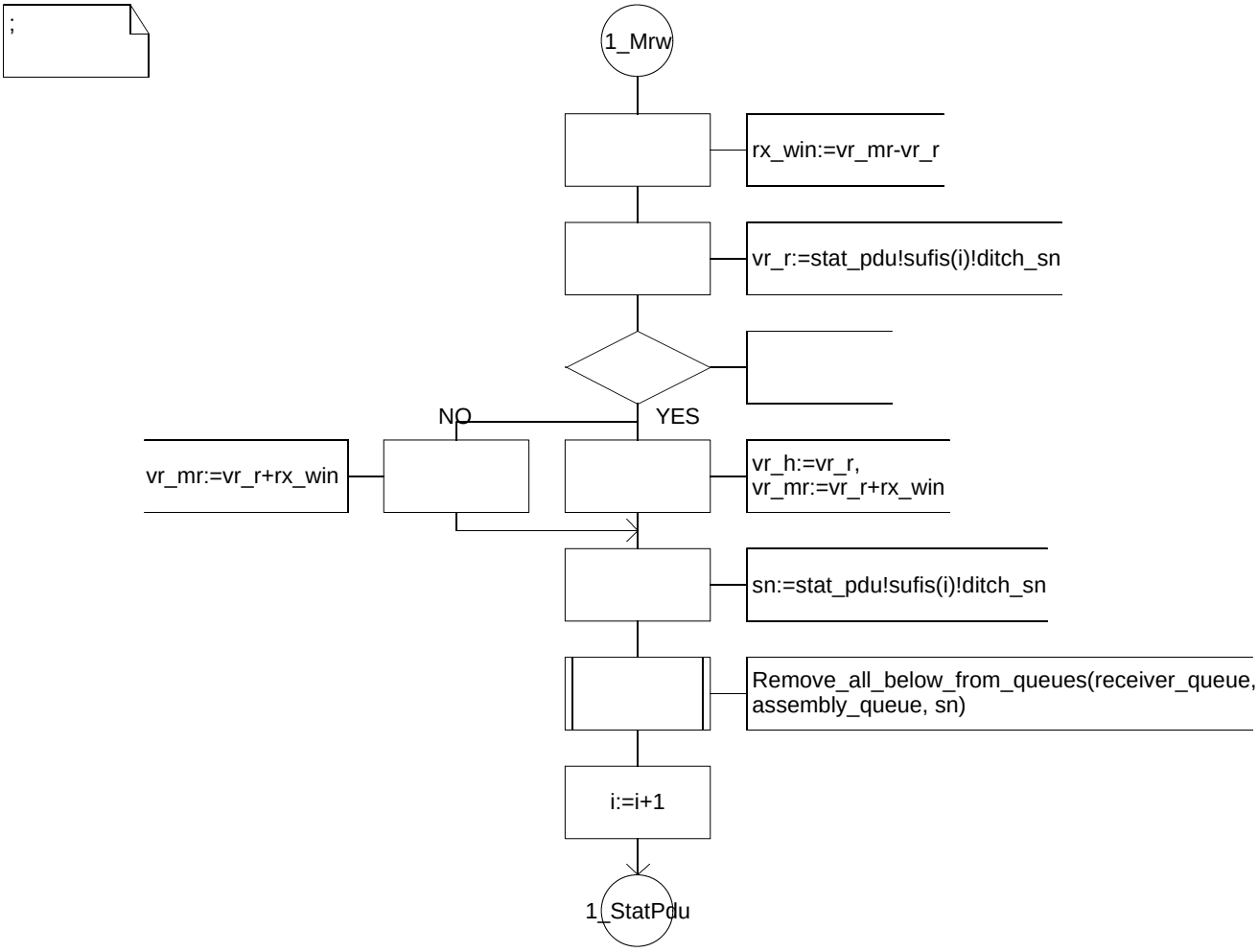
Virtual Process Type Acknowledged_conn1_AcknowledgedDataTransferReady_StatPduBitmap(44)



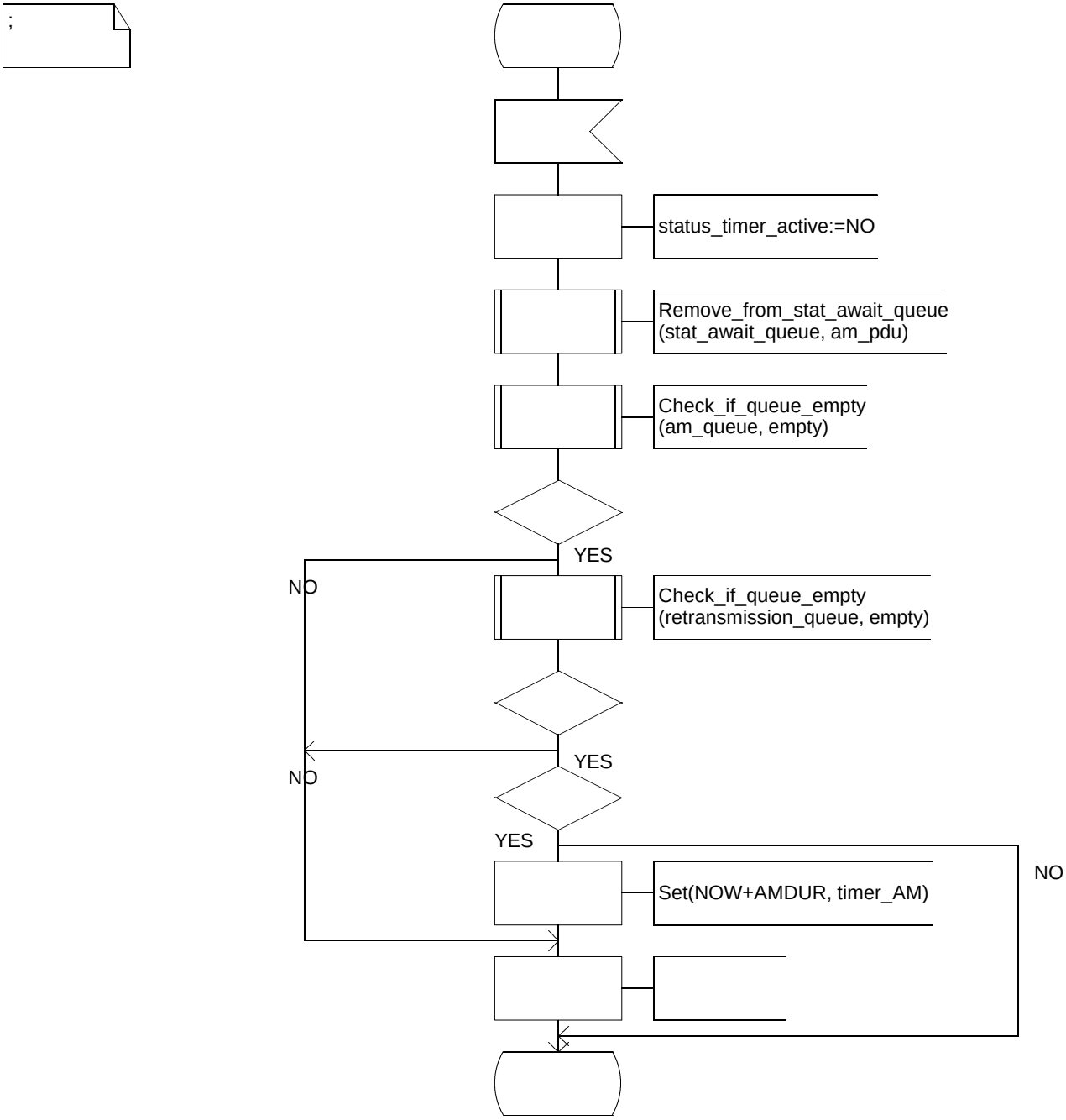
Virtual Process Type Acknowledged_connecti1_AcknowledgedDataTransferReady_StatPduAck(44)



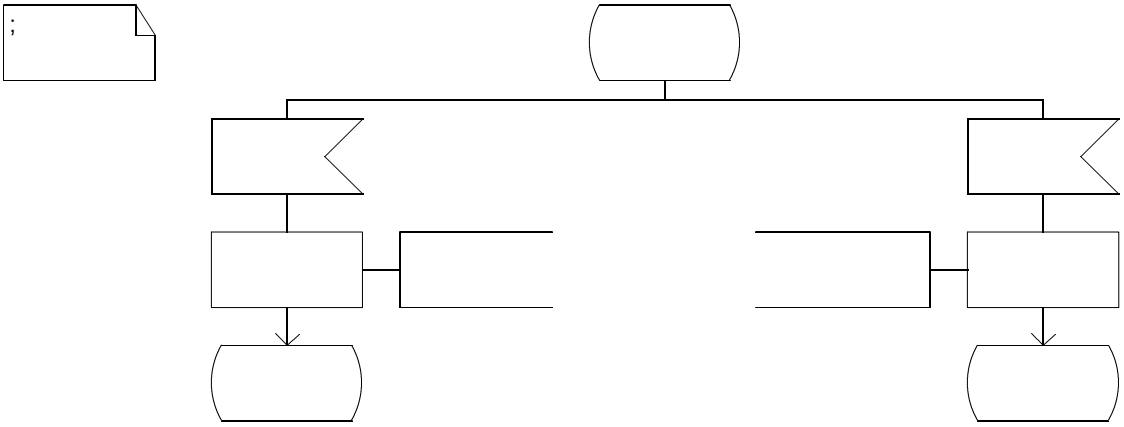
Virtual Process Type Acknowledged_connect1_AcknowledgedDataTransferReady_StatPduMrw(44)



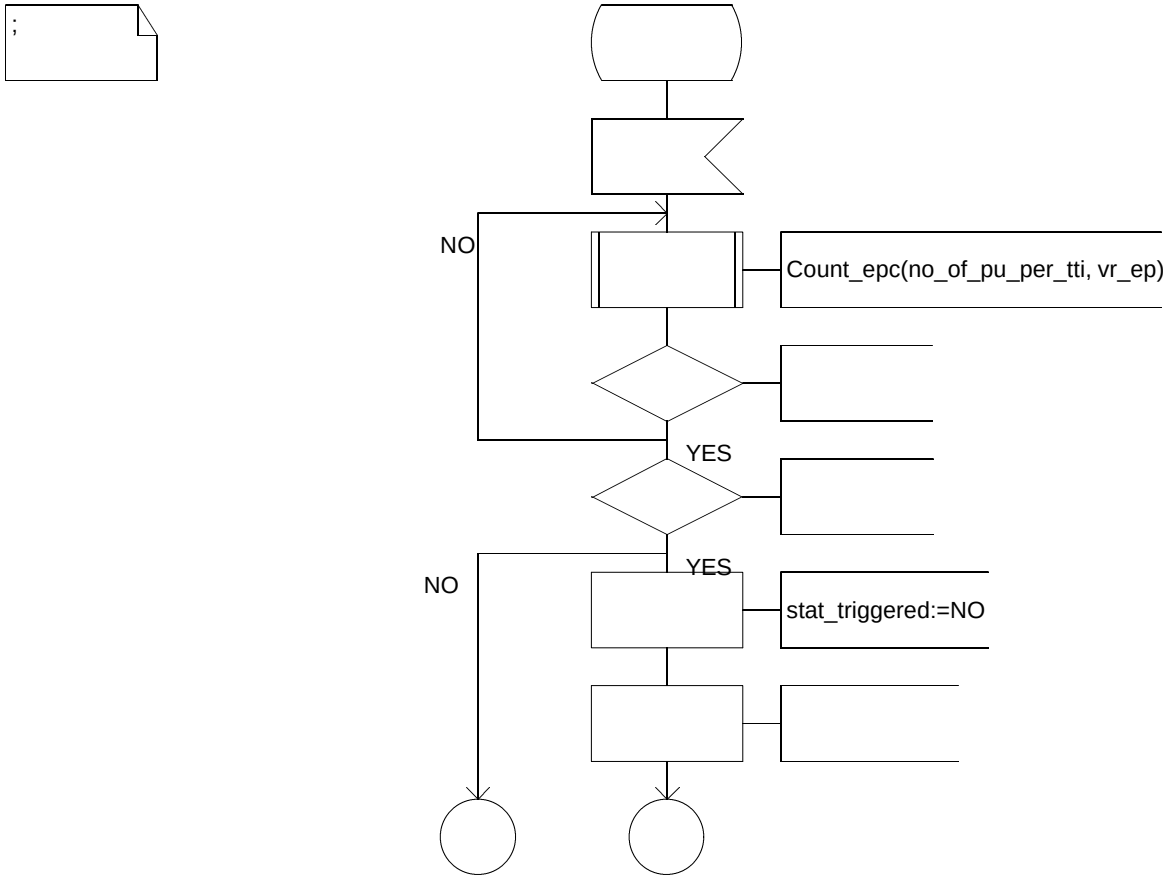
Virtual Process Type Acknowledged_connect1_AcknowledgedDataTransferReady_TimerStatus(44)



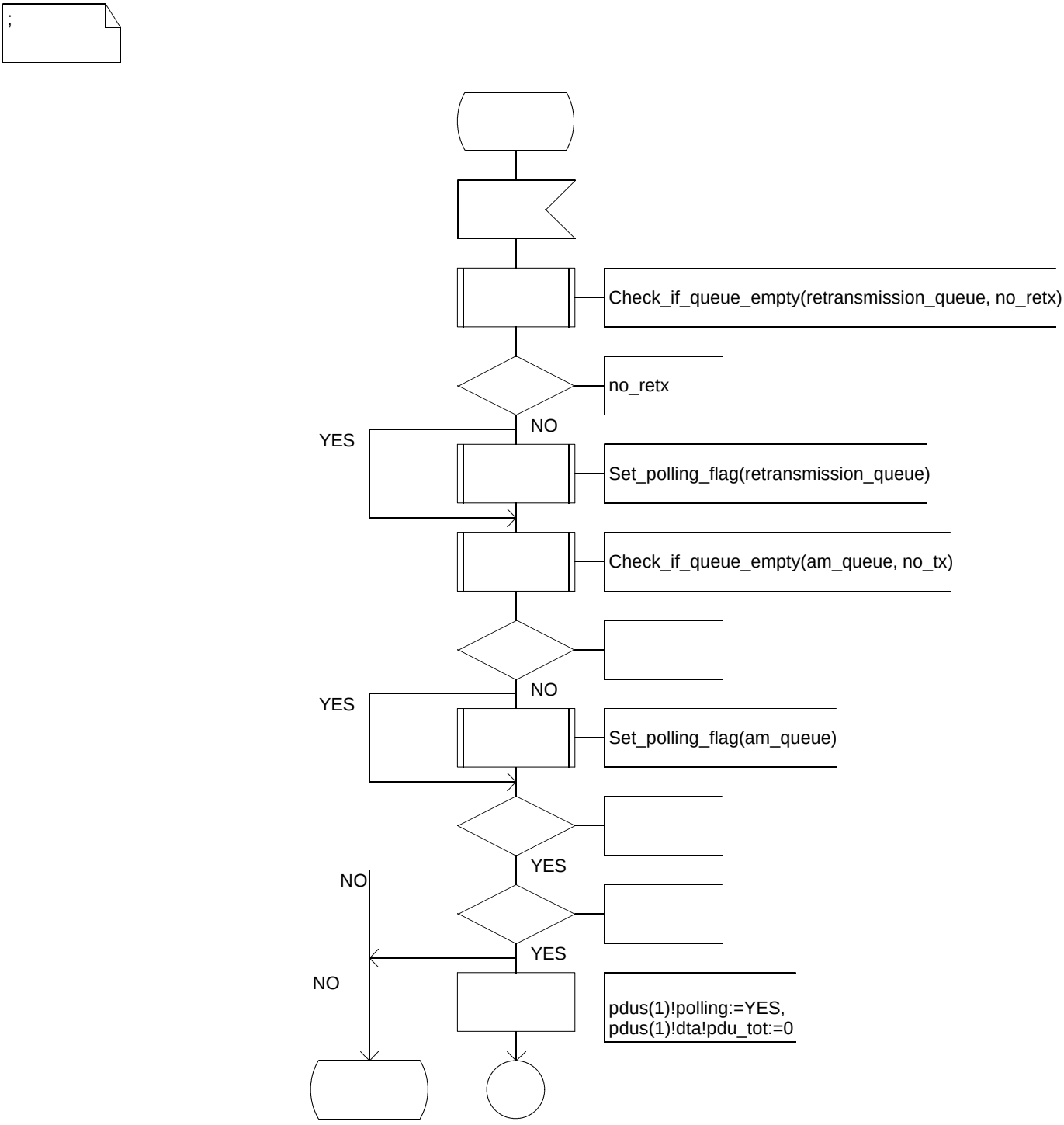
Virtual Process Type Acknowledged_conne1_AcknowledgedDataTransferReady_TimerProhibit(44)



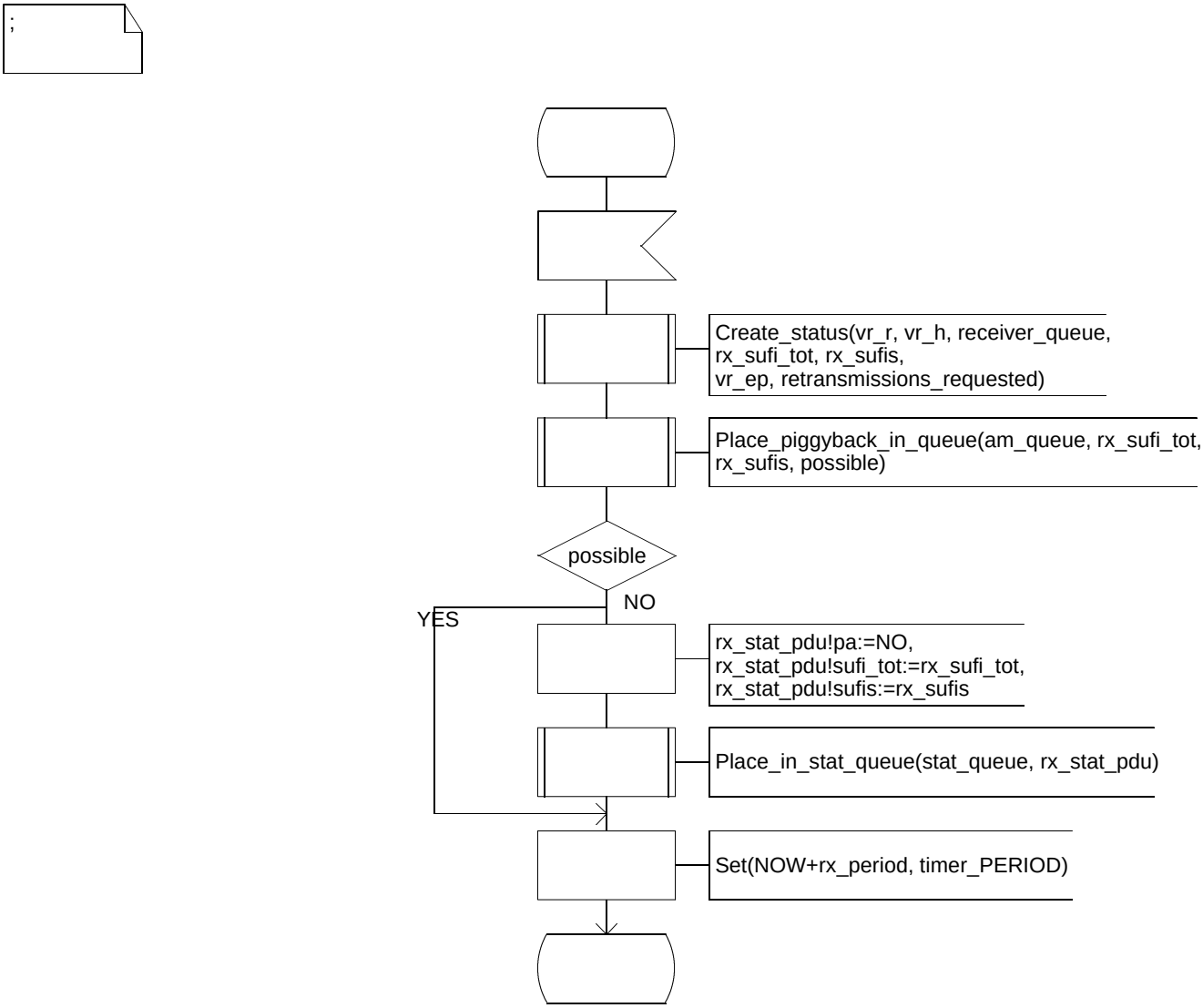
Virtual Process Type Acknowledged_connection1_AcknowledgedDataTransferReady_timerEpc(44)



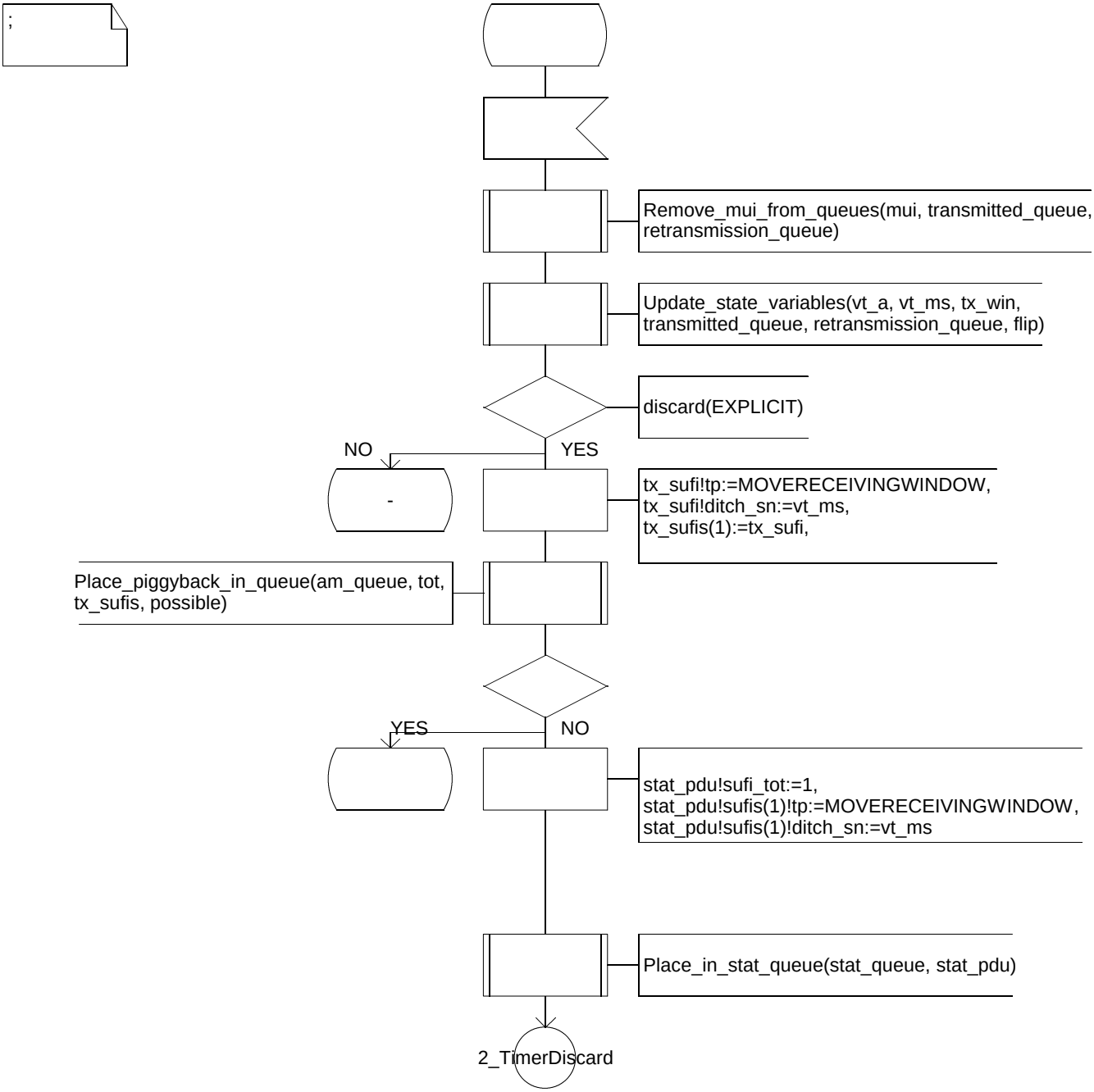
Virtual Process Type Acknowledged_connecti1_AcknowledgedDataTransferReady_timerPeriod(44)



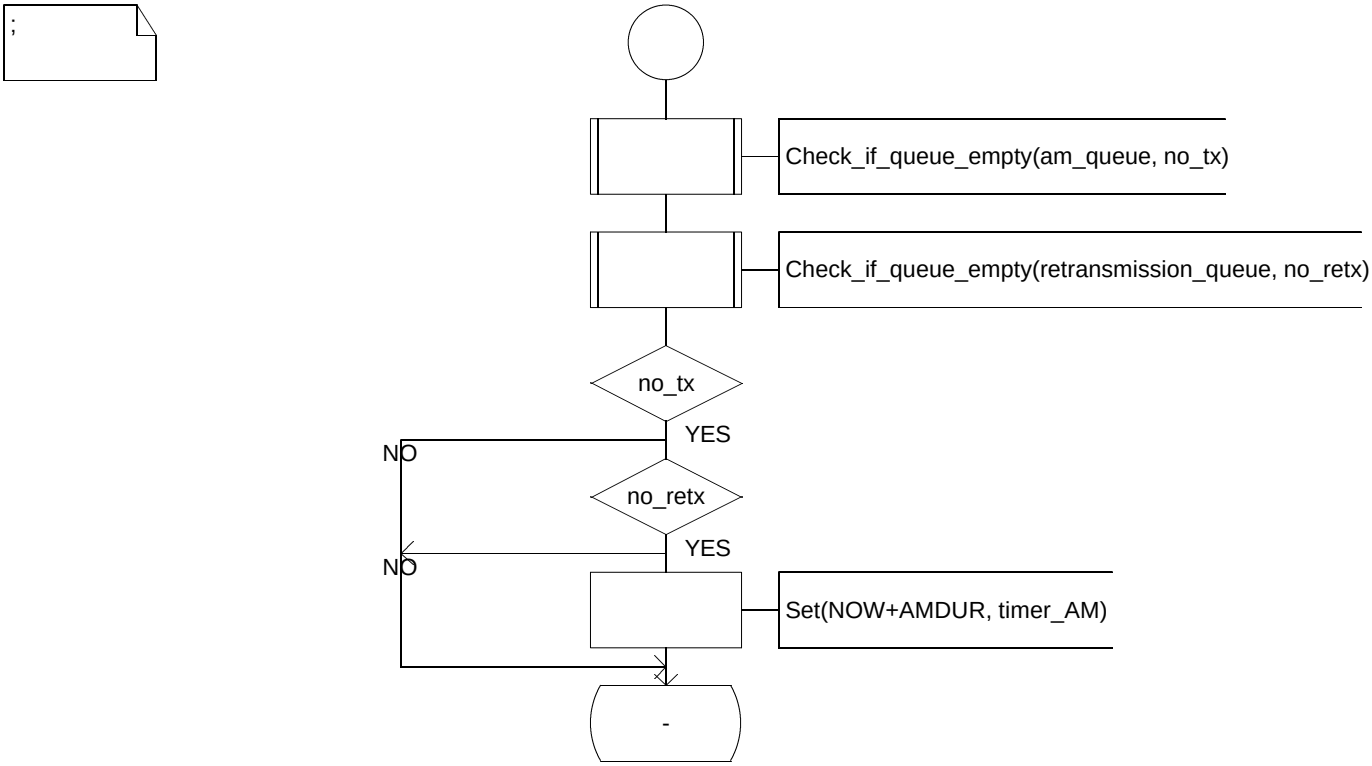
Virtual Process Type Acknowledged_conne1_AcknowledgedDataTransferReady_timerRxPeriod(44)



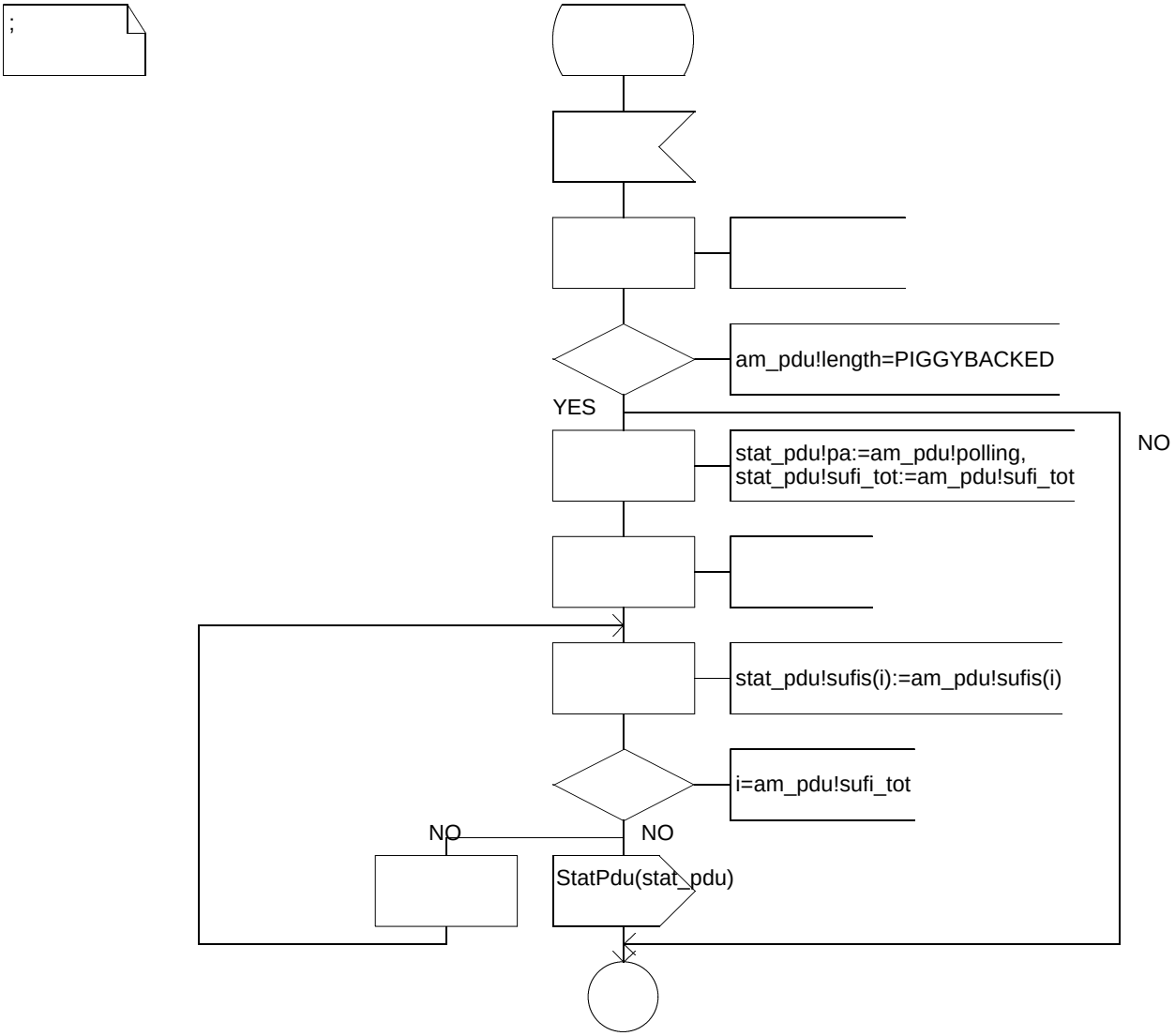
Virtual Process Type Acknowledged_conne1_AcknowledgedDataTransferReady_TimerDiscard(44)



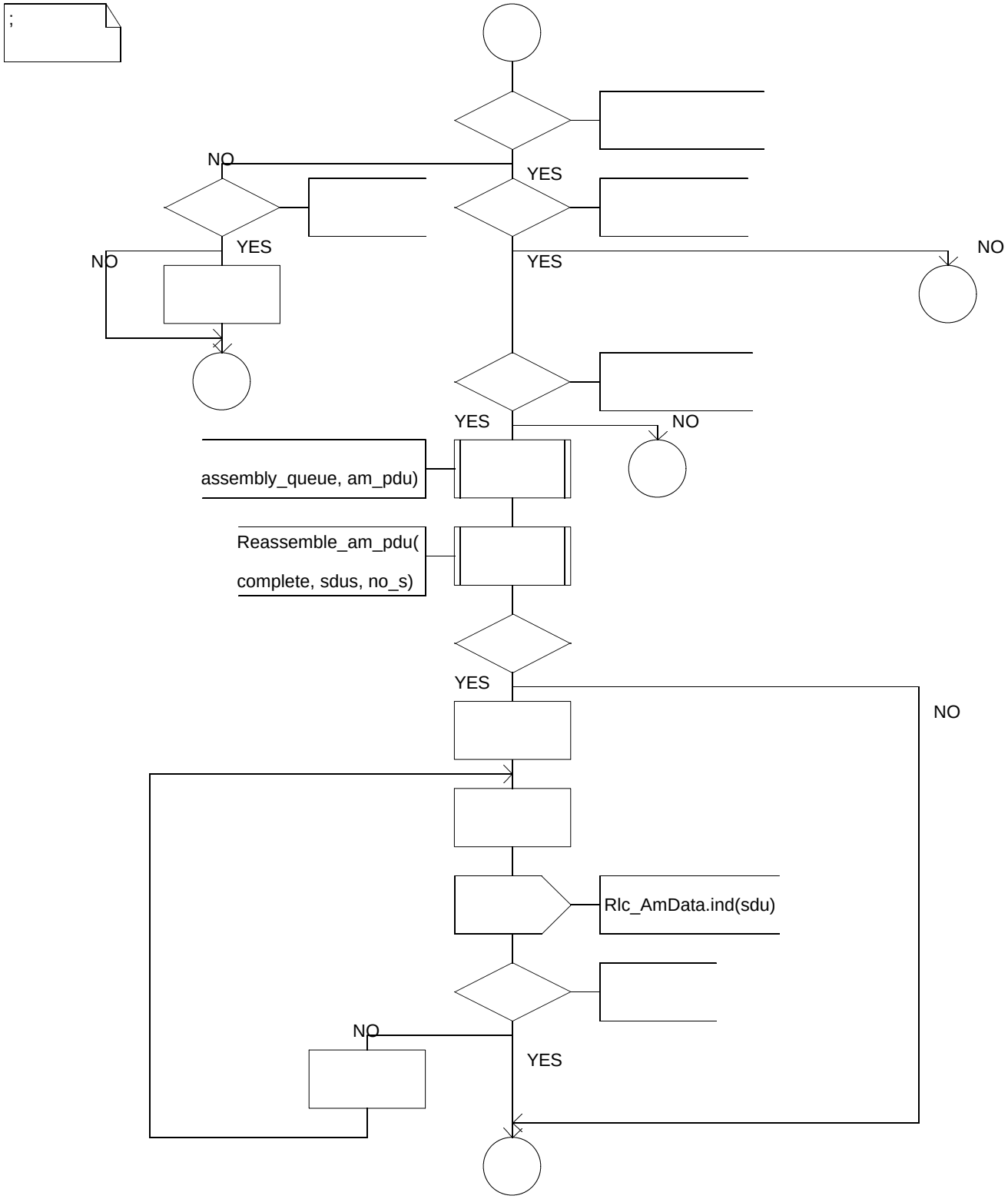
Virtual Process Type Acknowledged_connect_AcknowledgedDataTransferReady_TimerDiscard(44)



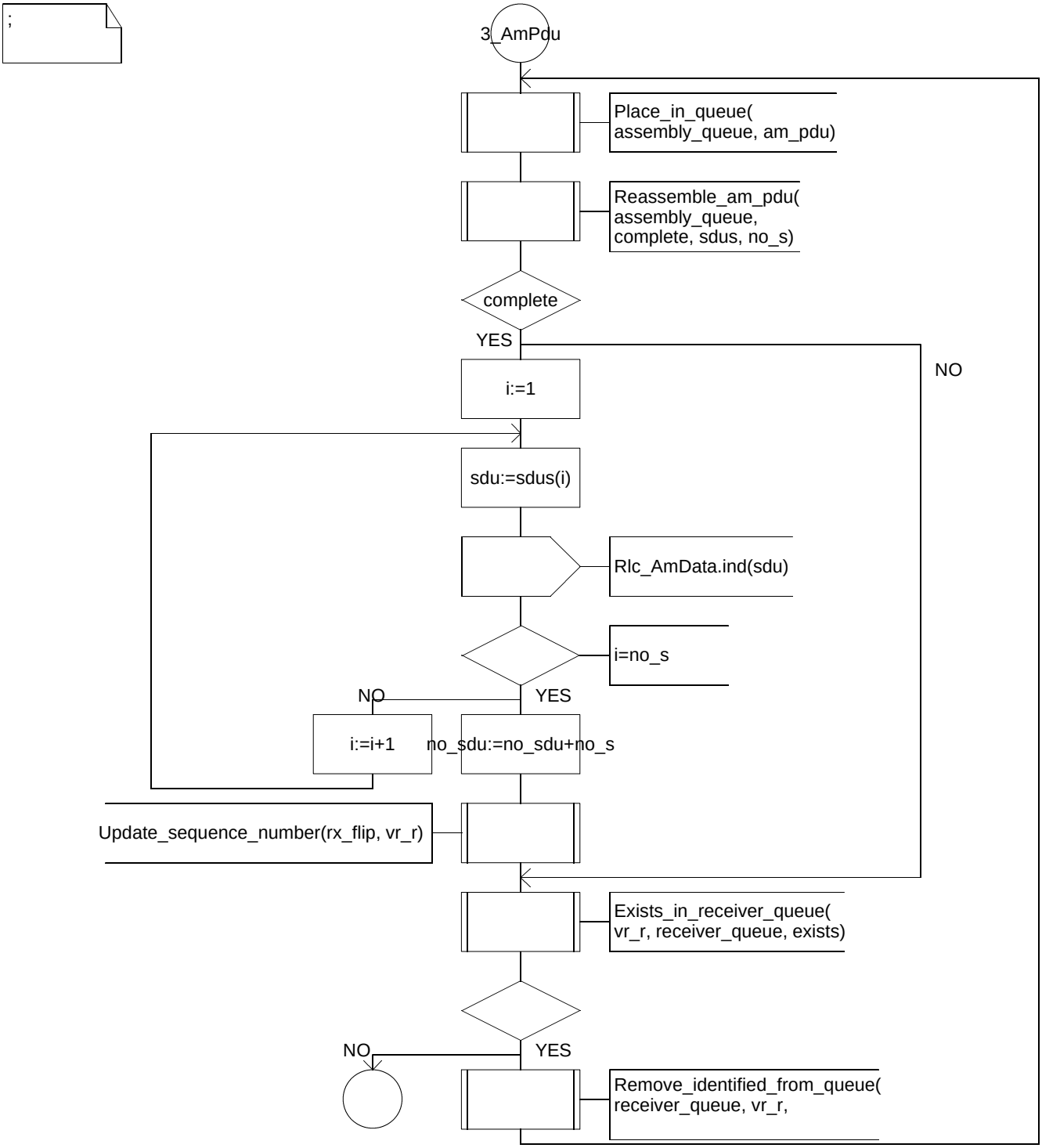
Virtual Process Type Acknowledged_connection 1_AcknowledgedDataTransferReady_AmPdu(44)



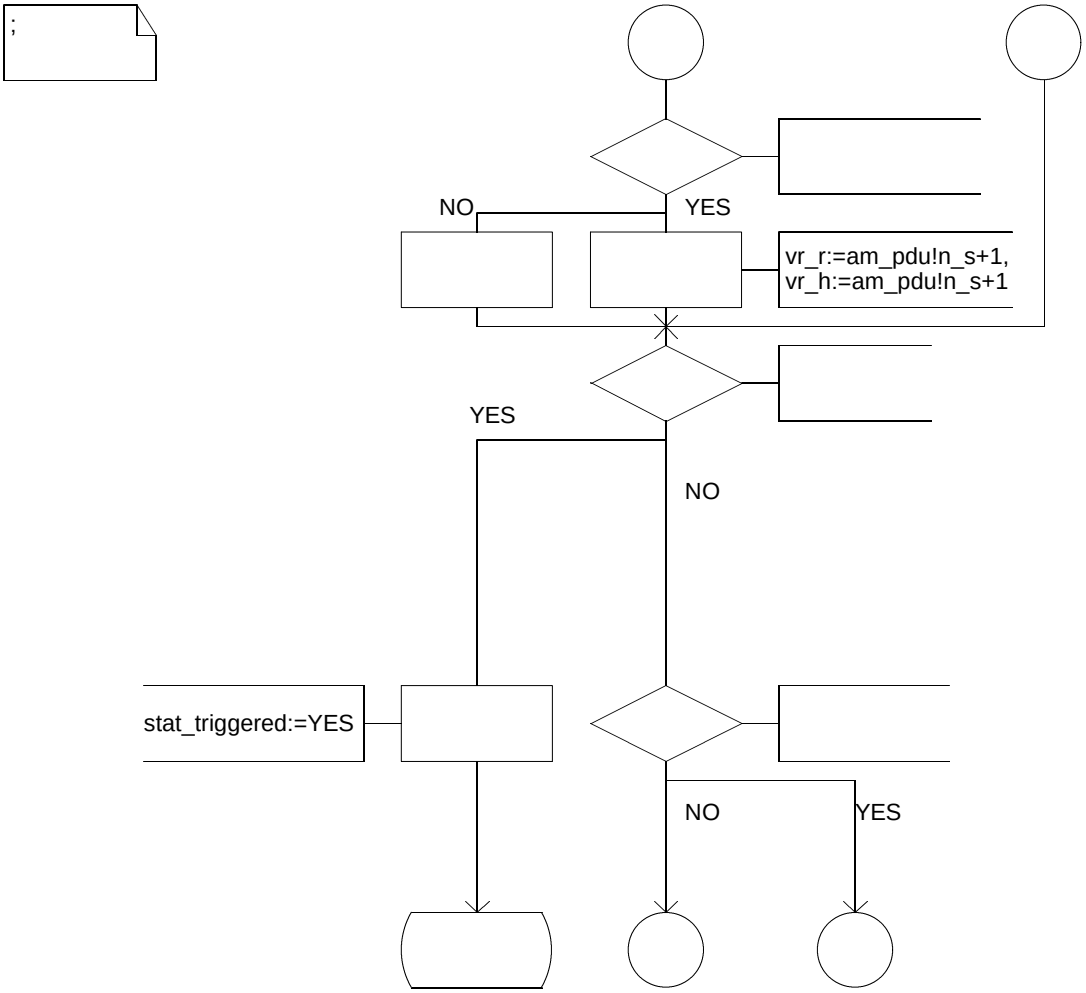
Virtual Process Type Acknowledged_connection 2_AcknowledgedDataTransferReady_AmPdu(44)



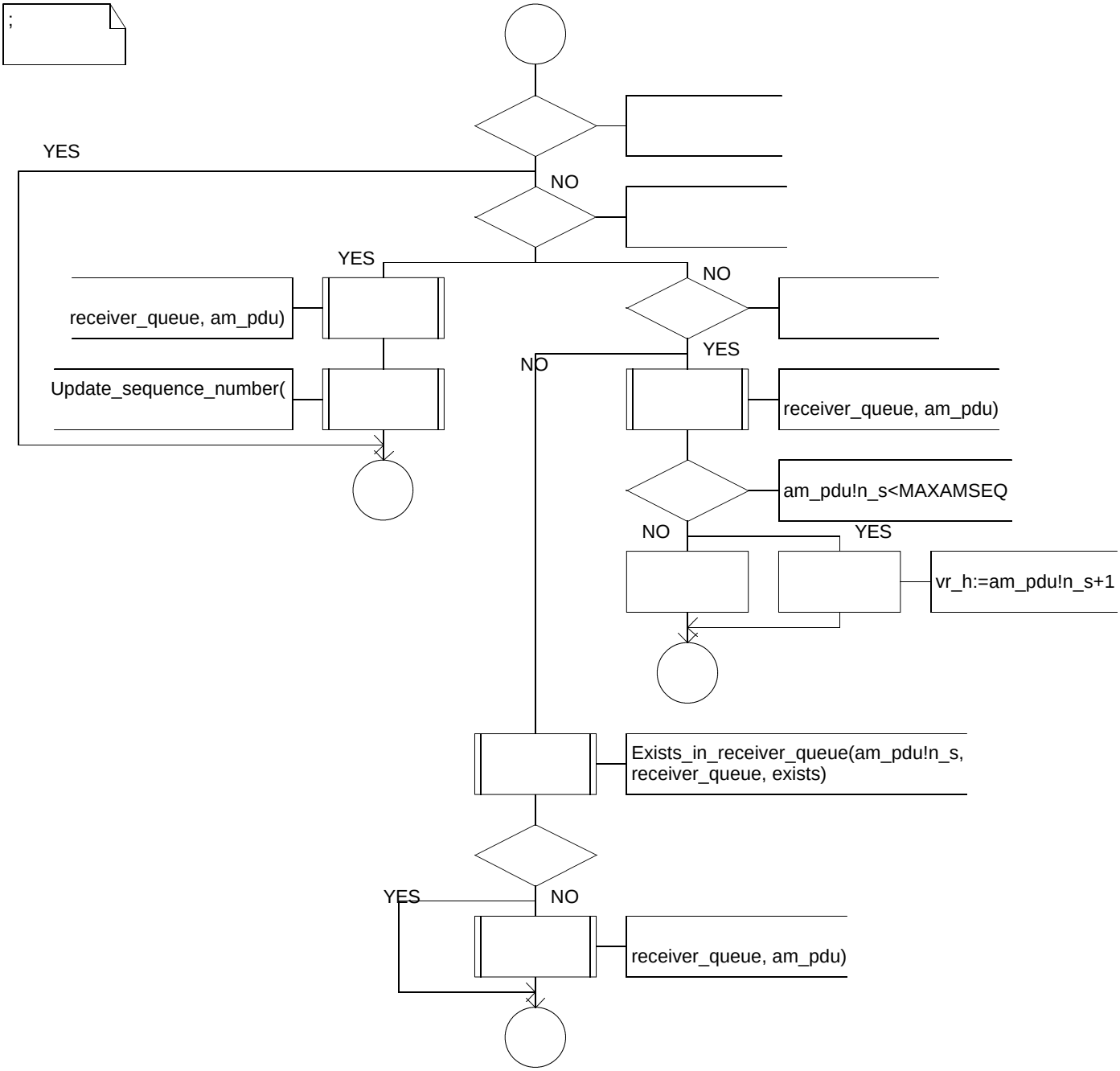
Virtual Process Type Acknowledged_connection 3_AcknowledgedDataTransferReady_AmPdu(44)



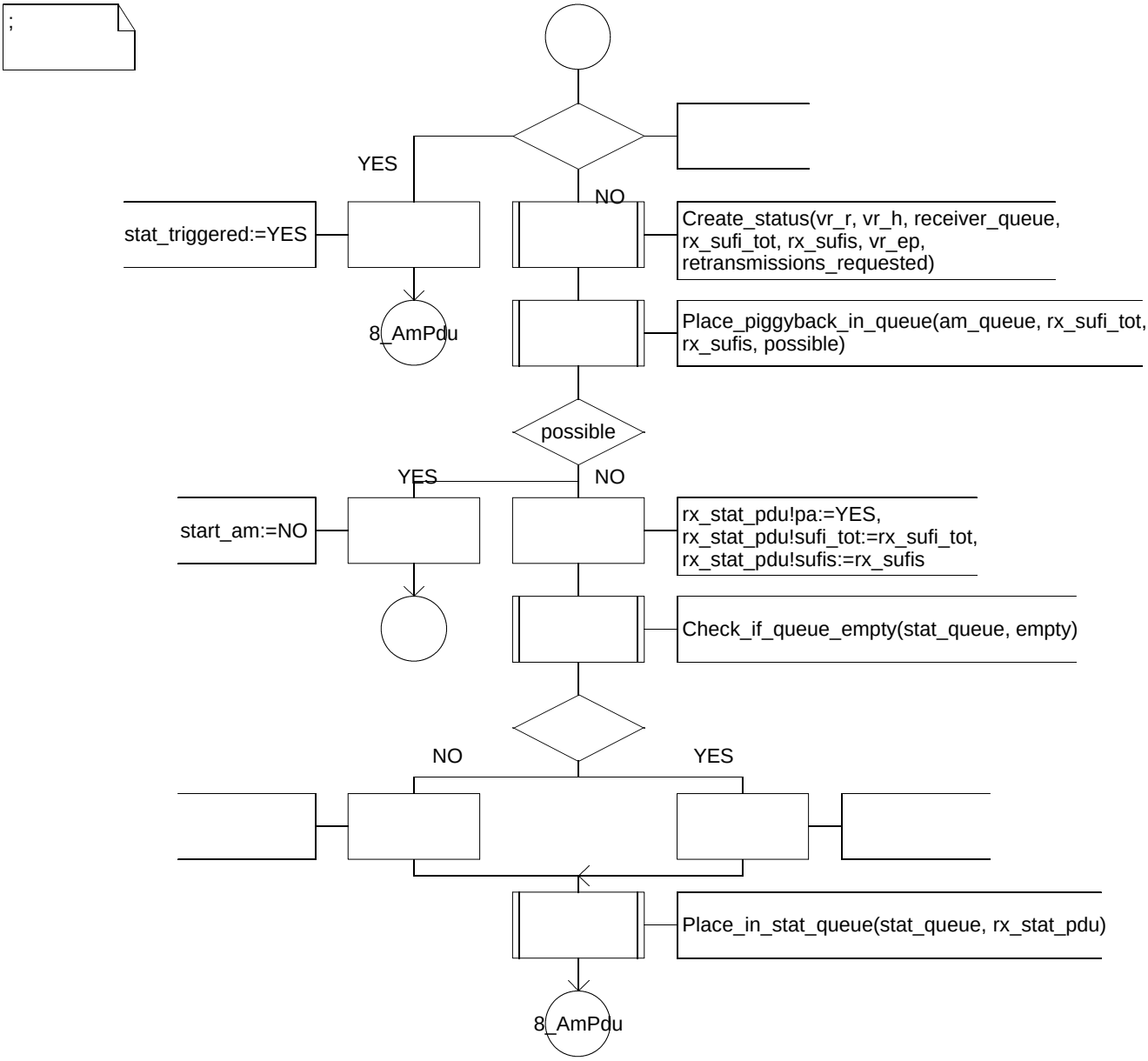
Virtual Process Type Acknowledged_connection 4_AcknowledgedDataTransferReady_AmPdu(44)



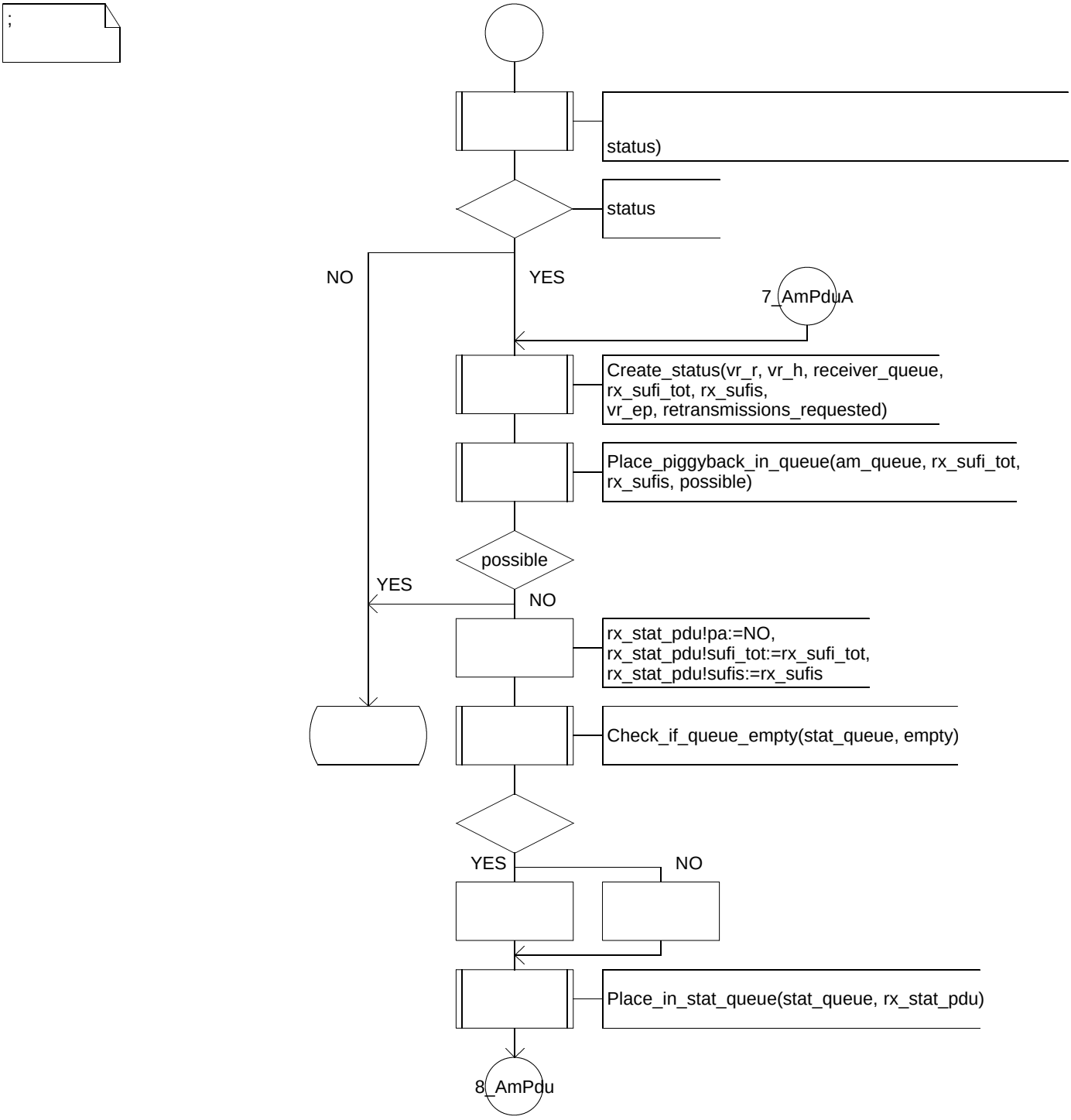
Virtual Process Type Acknowledged_connection 5_AcknowledgedDataTransferReady_AmPdu(44)



Virtual Process Type Acknowledged_connection 6_AcknowledgedDataTransferReady_AmPdu(44)

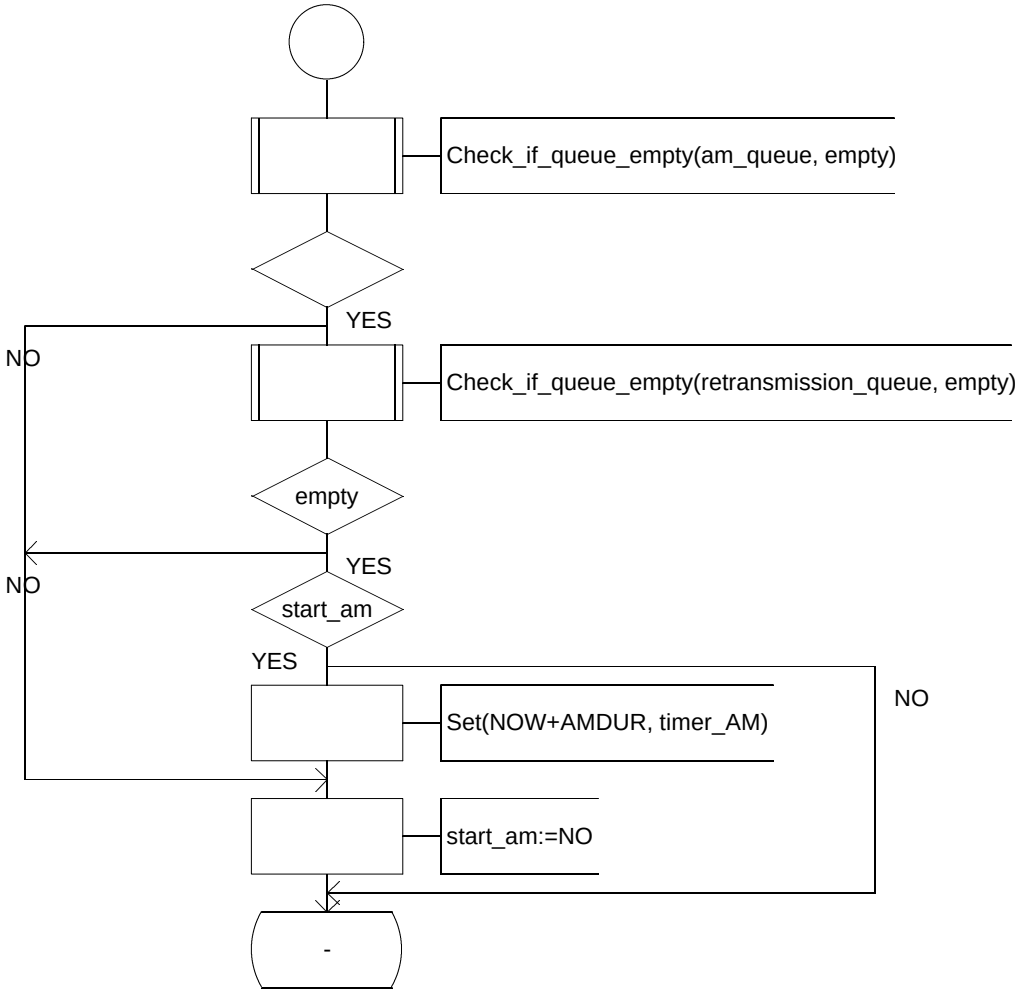


Virtual Process Type Acknowledged_connection 7_AcknowledgedDataTransferReady_AmPdu(44)

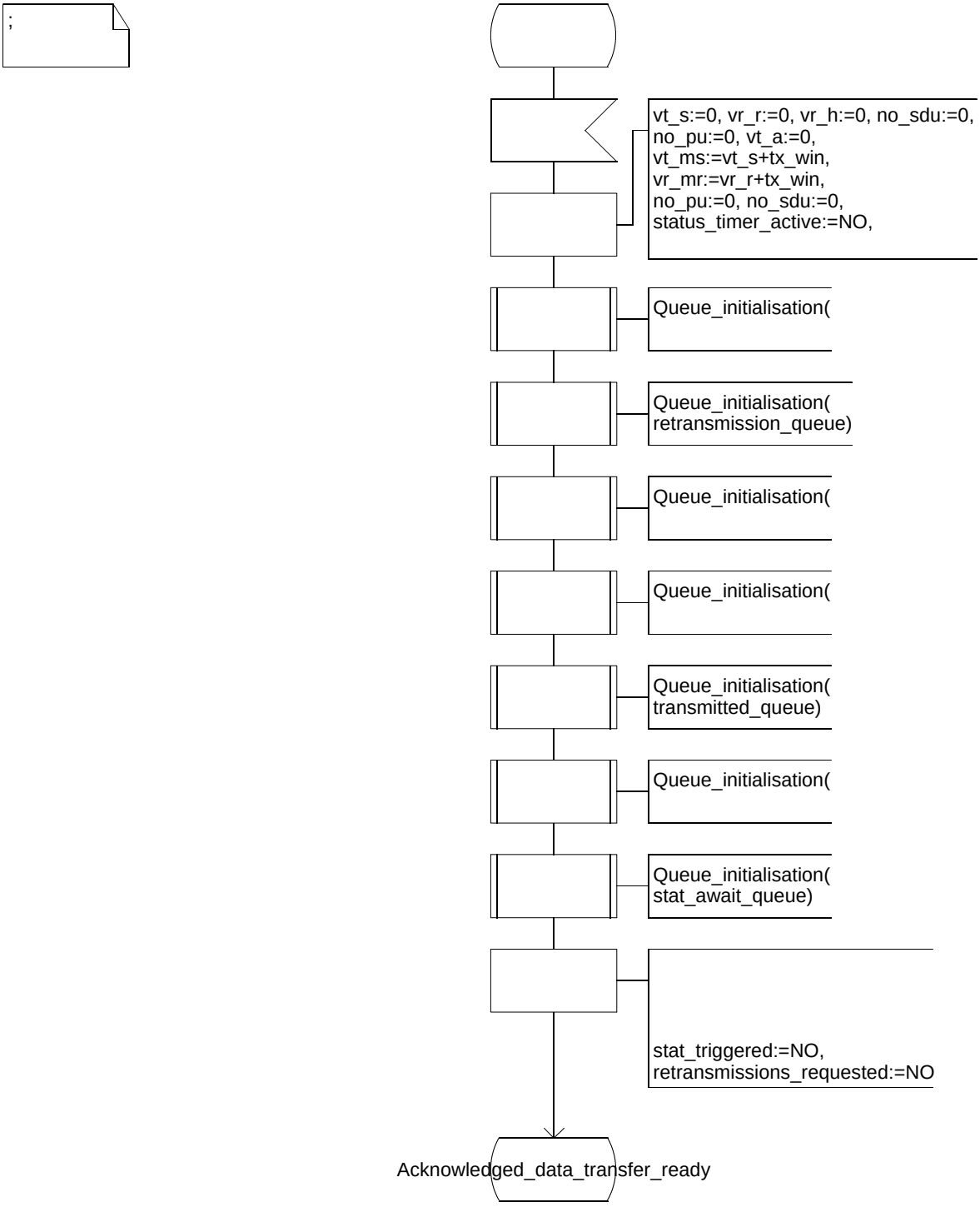


Virtual Process Type Acknowledged_connec8_AcknowledgedModeDataTransferReady_AmPdu(44)

;



Virtual Process Type Acknowledged_connection 1_ResetPending(44)



12.2.Annex B: Pseudo code describing AMD PDU header Compression

The following Pseudo-Code is an example of algorithm to describe the exact Header Compression Operation that takes place when several PUs are packed into one RLC PDU.

```
/* Prior to calling this procedure it must be checked that <pus_in_pdu> consecutive PU:s
   are to be transmitted (or there is padding in the end)*/

Compress_PDU (pus_in_pdu, pu_size) {

  li_addition = 0;                  // reset the variable that counts data in full pu:s

  Loop through pus_in_pdu {

    d_e_flag = E-flag for this PU;

    If (d_e_flag == FALSE) {

      Append PU data to PDU data;   // complete PU is SDU-data
      li_addition += pu_size;       // to be added to the next LI

    } else {                        // E-flag is TRUE, so LI-field(s) exist

      Previous E-flag in PDU = TRUE;   // Either in PDU header or pdu_li_vector;

      j = 0;                        // reset LI-counter for this PU
      pu_data_size = 0;             // reset data size counter for this PU

      Loop until (d_e_flag == FALSE) {

        d_li = next LI;             // in octet j of PU;
        d_e_flag = next E_FLAG;     // in octet j of PU;

        if (d_li is not PADDING) {

          pu_data_size += d_li;     // to keep track of data segment size in this PU);
          d_li += li_addition;      // to add data from previous PU:s to LI-value);
          li_addition = 0;          // reset li_addition;

        }

        Append (d_li + d_e_flag) to pdu_li_vector;

        j++;                        // go to next li_octet, if d_e_flag is TRUE);

      } /* end-of-loop (exit when d_e_flag is TRUE) */

      Append pu_data_size segments starting from j to RLC-PDU data;

    } /* end-of e-flag == TRUE */

  } /* end-of loop through PU:s in PDU */

} /* end-of Compress_PDU */
```

13. History

Document history		
Date	**Version**	**Comment**
September 1999	1.3.0	Inclusion of minor editorial changes. Approved for the submission to 3GPP TSG RAN#5

September 1999	1.2.1	Outcome of TSG/RAN/WG2#7: proposals in documents A80, C71(HE field and LI, merged with A80), A81 (STATUS PDU Tx and Polling function included in section 9.7 and revision of section 11), B62 (state model fro AM entities), C36 (inclusion of Ciphering elements on primitives), C38 (SHCCH), and C39 (revised definition of Timer_Poll) were included.
August 1999	1.2.0	Outcome of TSG/RAN/WG2#6: decision on Tdoc 946, 725 and 938 were incorporated.
August 1999	1.1.2	Editorial clarification and solution of some inconsistencies submitted to WG#6 as Tdoc 946
July 1999	1.1.1	This is the outcome of TSG/RAN/WG2#5 (Sophia Antipolis 5-9 July 99).The Tdocs 549,551,552,561,562, 576,577,605,611,656 have been included. The main changes concern as follows: - the CRLC_STATUS primitive has been included - text on RLC error handling has been included in Sec. 10. - general rewording of Sec. 9.2 with the inclusion of 2 new super-field types (WINDOW SIZE, RLIST). The extended header description has been included as well. - part of the text in Sec.9.4, 9.5, 9.6 has been updated, and a number of new state variables, timers and parameters, have been included. -part of the text relative to the toolbox functions has been moved from Sec. 9.8 to Sec 11. Sec. 8 has been removed. - the Quick Repeat function, and related parameters has been removed. - a more appropriate description of the header compression concept, has been included in sec. 9.7.4. - some editoral corrections concerning the replacement of PDU with PU have been included in Sec.9.7.3
June 1999	1.1.0	This version has been noted by the RAN plenary in Miami.
May 1999	1.0.1	The Tdocs 382,383,384,385,402,403,404,405,407, 488 have been included. The main changes concern as followes: - the removal of a number of control PDUs (BGN, BGACK,BGREJ, END, END ACK) - the inclusion of new PDUs: RESET, RESET ACK - the redefinition of RLC primitives and their parameters - the inclusion of new principles: Piggybacking, EPC, SDU discard., chiphering - the inclusion of RLC elementary procedures
May 1999	1.0.0	The old numbering S2.22 has been removed and replaced with new one 25.322. The document was noted by the TSG/RAN plenary (Yokohama 21-24 April) and the old version 0.1.0 has been upgraded to 1.0.0.
April 1999	0.1.0	The content of Tdoc 99/253 concerning the new STAUS PDU format has been included in section 9.2. The content of Tsoc 99/255 on the RLC toolbox has been included in section 9.8. Approved by WG2.

March 1999	0.0.2	The content of Td155 was included on section 9.7; the principle for the Multiple fixed size RLC PDU with RLC PDU Header compression expressed in td115, td116 were included and part of the proposed changes in td117 applied. The RLC Repetition Scheme proposed in Td 155 was included in Section 9.7. The changes to the RLC Model presented in Td 147 were included in Section 4.2.1. The RLC Protocol States presented in Td 148 were included in Section 9.3.
January 1999	0.0.1	Document created. Based on TSG RAN WG2 Tdoc 016/99, 006/99 and 021/99..

Rapporteur for 3GPP TSG RAN WG2 TS 25.322 is:

CSELT

Marco Mastroforti
Tel.: +39 011 228 7596
Fax: +39 011 228 7055

Email: marco.mastroforti@cselt.it

Daniele Franceschini
Tel: +39 011 228 5203
Fax: +39 011 228 7613

E-mail: daniele.franceschini@cselt.it

This document is written in Microsoft Word Version 97