TSG R2#6(99)713

3G TS RAN 25.322 V1.1.~~0~~1 (1999-0~~5~~7)

Technical Specification

3rd Generation Partnership I
Technical Specification Grc
Working Group 2 (WG2);

RLC Protocol Specification
(3G TS 25.322 version 1.1.~~0~~1)

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 (TS) has been produced by the 3rd Generation Partnership Project (3GPP).
The contents of this TS are subject to continuing work within 3GPP TSG-RAN and may change following formal TSG RAN approval.

1. Scope

1. 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
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 RLC Sublayer supports, for the RLC PDU Mechanism, the following features:*~~

~~*- Fixed Size RLC PDU with the possibility to adjust the number of PU per transmission time interval.*~~

~~*- Multiple Fixed Size RLC PDU with a RLC PDU Header Compression.*~~

~~*One of the two options can be chosen during the RLC Configuration Phase.]*~~

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 control data (e.g. resynchronisation PDUs and acknowledgements) and data PDUs on separate logical channels. More detailed descriptions of the different entities are given in subsections 4.2.1.1, 4.2.1.2,4.2.1.3.

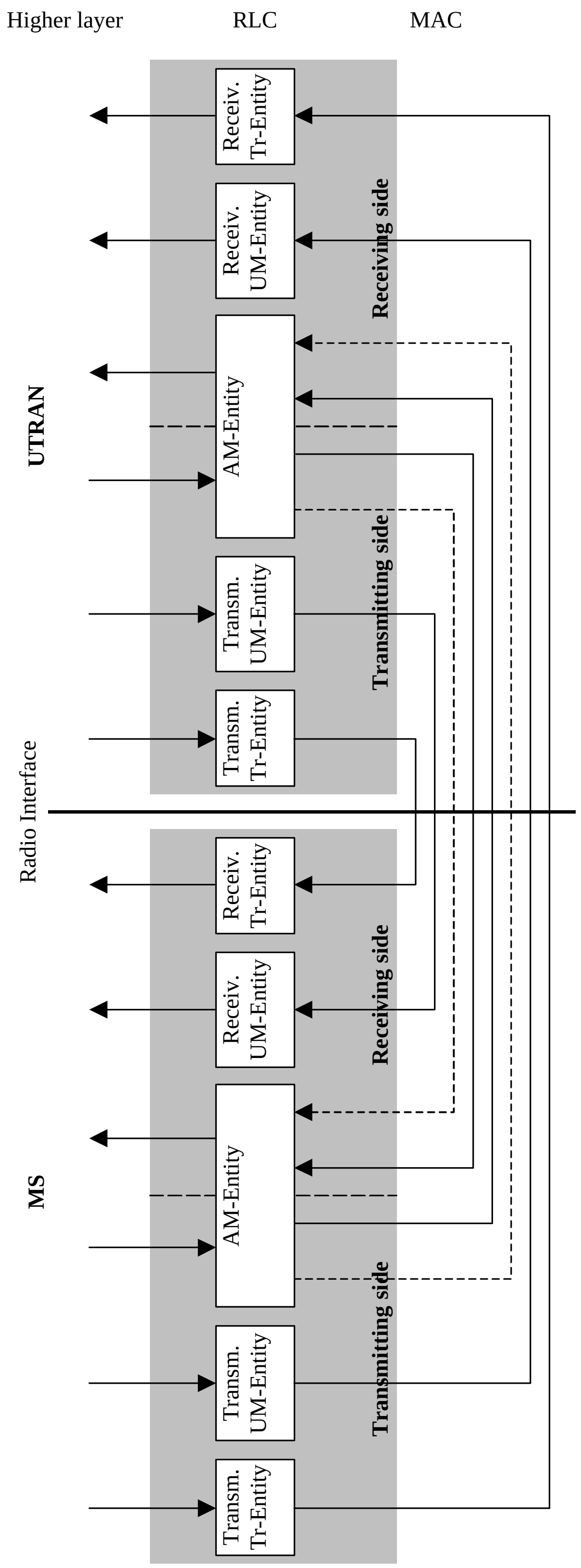

Figure 4-1 Overview model of RLC.

4.2.1.1. Transparent mode entities

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

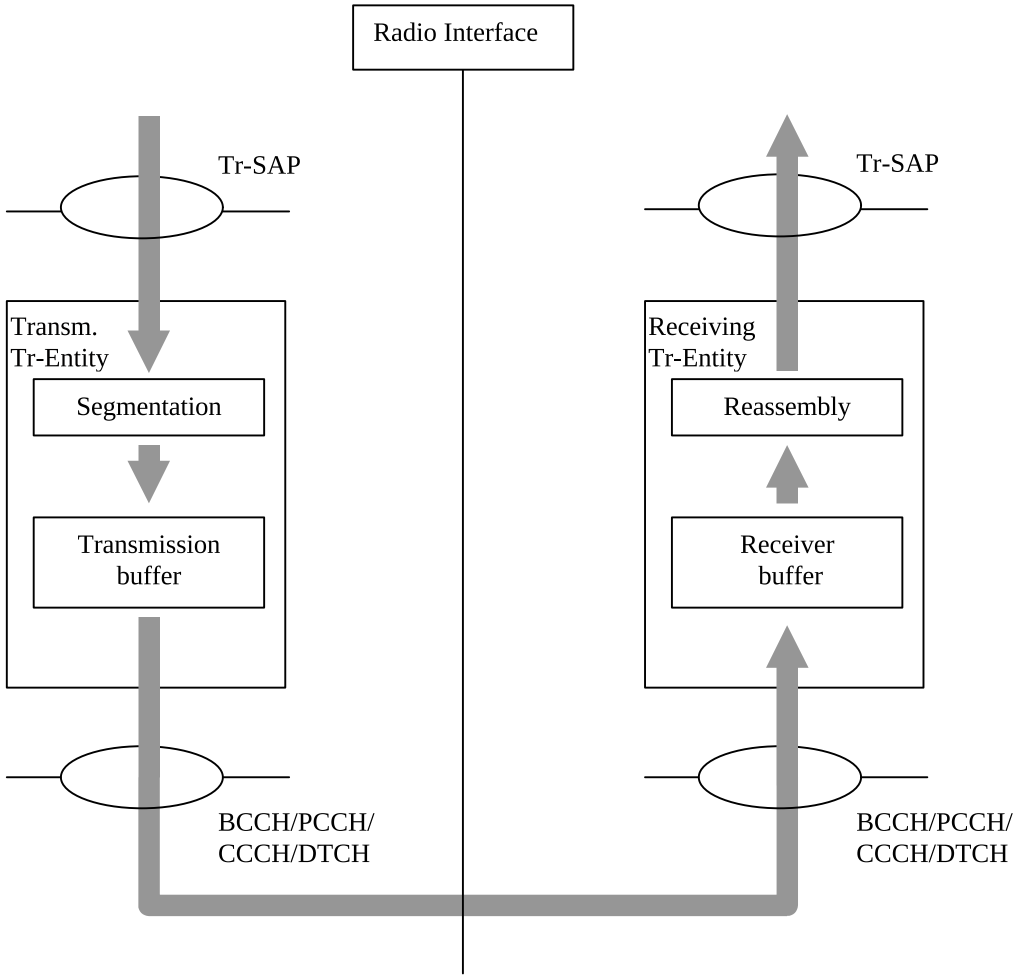

Figure 4-2 Model of two transparent mode peer entities.

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 or a DTCH. The delivery of RLC PDUs to MAC through CCCH is FFS. Which type of logical channel depends on if the higher layer is located in the control plane (BCCH, PCCH, CCCH) or user plane (DTCH).

The Tr-entity receives PDUs through from 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.

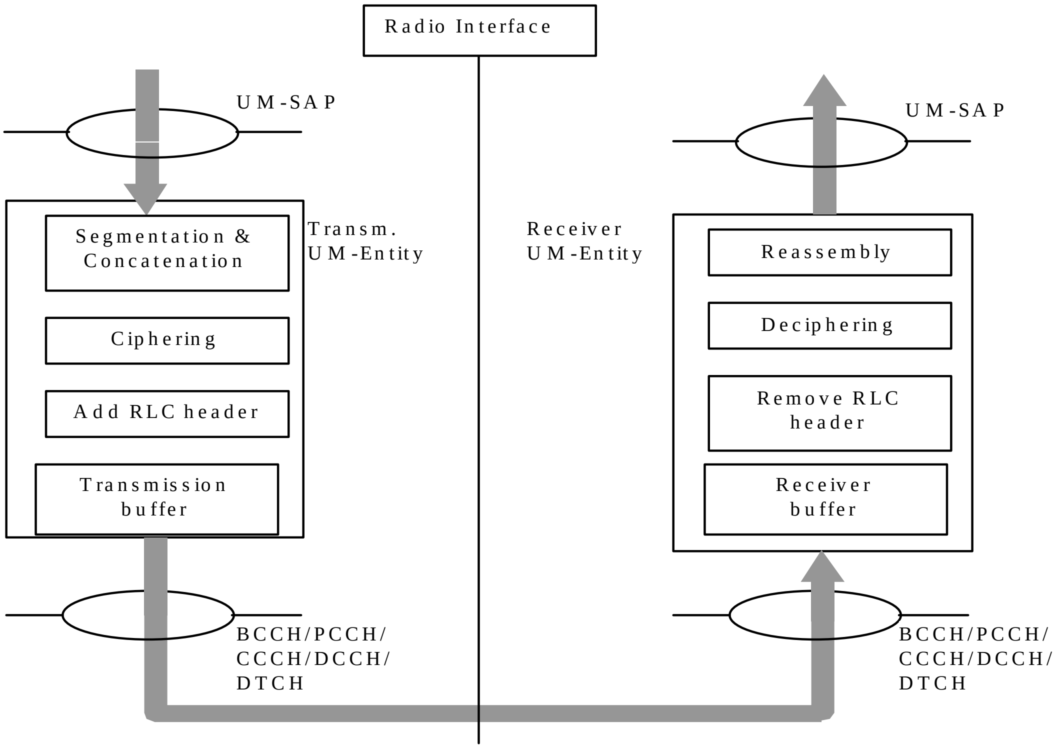

Figure 4-3 Model of two unacknowledged mode peer entities.

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 or a DTCH. The delivery of RLC PDU's to MAC through BCCH, PCCH, CCCH, is for FFS. Which type of logical channel depends on if the higher layer is located in the control plane (BCCH, PCCH, CCCH, DCCH) 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.

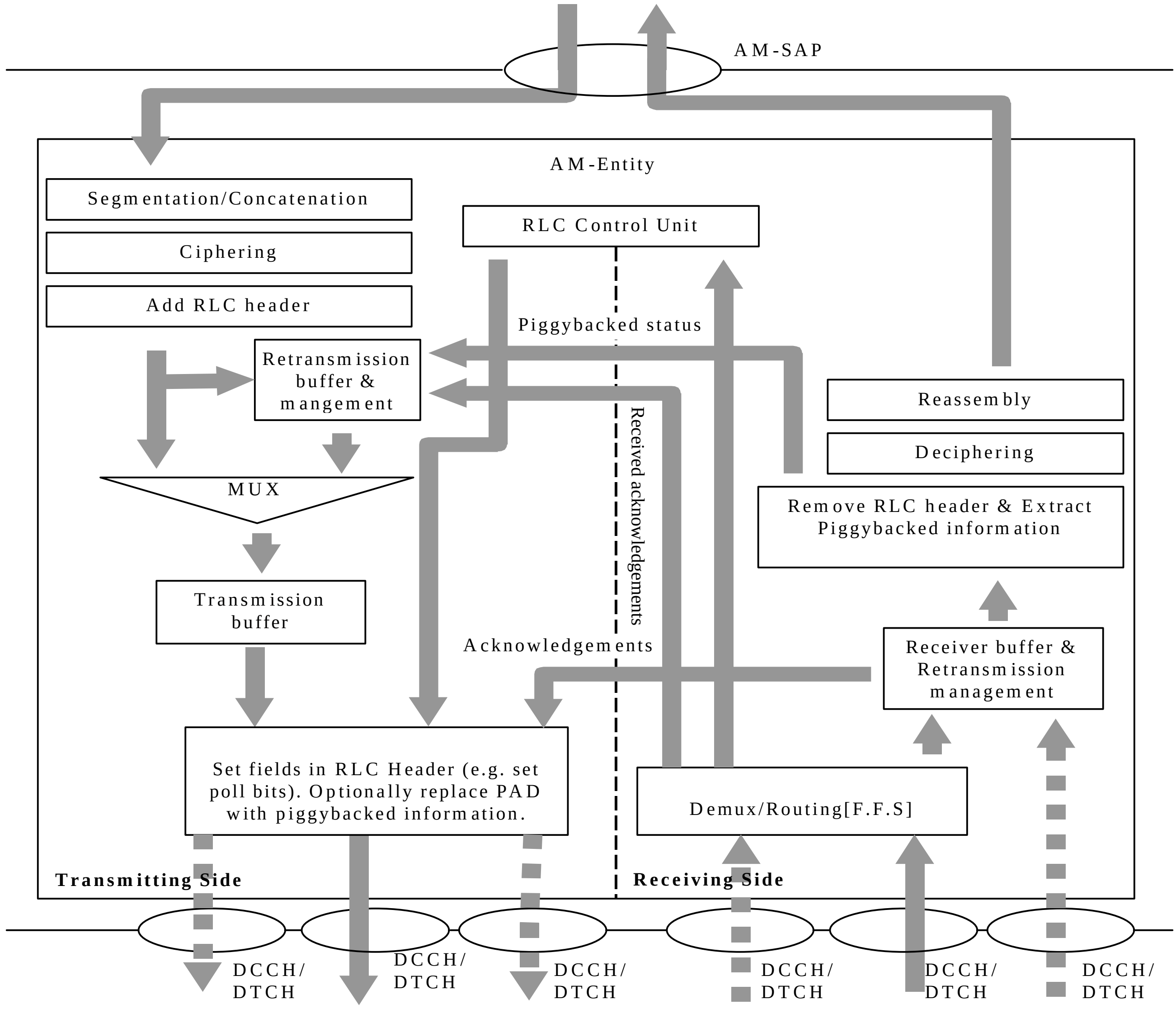

Figure 4-4 Model of a acknowledged mode entity.

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. RLC PDU is constructed from PU buffers.

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,*[The setting of other bits in the Header is [F.F.S]. and will be specified when the AMD PDU Structure will be defined]*, 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 mach 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

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;

~~- Quick repeat.~~

6. Services provided to upper layers

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

- **RLC connection establishment/release;**
- **Transparent data transfer Service**

  Following functions are needed to support transparent data transfer:

  - Segmentation and reassembly
  - Transfer of user data;

- **Unacknowledged data transfer Service**

  Following functions are needed to support unacknowledged data transfer:

  - Segmentation and reassembly
  - Concatenation
  - Transfer of user data;

- **Acknowledged data transfer Service**

  Following functions are needed to support acknowledged data transfer:

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

- **QoS setting**;
- **Notification of unrecoverable errors**.
- **Multicast delivery of higher layer messages. (FFS)**

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 6-16-1: RLC modes and functions in UE uplink side

Service	Functions	CCCH	DCCH	DTCH
Transparent Service	Applicability	+	-	+
	Segmentation	-	-	+
Unacknowledged Service	Applicability	FFS	+	+
	Segmentation	-	+	+
	Concatenation	-	+	+
Acknowledged Service	Applicability	-	+	+
	Segmentation	-	+	+
	Concatenation	-	+	+
	Flow Control	-	+	+
	Error Correction	-	+	+
	Protocol error correction & recovery	-	+	+

Table 6-36-2: RLC modes and functions in UE downlink side

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

Table 6-56-3: RLC modes and functions in UTRAN downlink side

Service	Functions	SCCH	BCCH	PCCH	CCCH	DCCH	DTCH
Transparent Service	Applicability	+	+	+	+	-	+
	Segmentation	+	+	+	-	-	+
Unacknowledged Service	Applicability	+	FFS	FFS	FFS	+	+
	Segmentation	+	+	+	-	+	+
	Concatenation	+	+	+	-	+	+
Acknowledged Service	Applicability	-	-	-	-	+	+
	Segmentation	-	-	-	-	+	+
	Concatenation	-	-	-	-	+	+
	Flow Control	-	-	-	-	+	+
	Error Correction	-	-	-	-	+	+
	Protocol error correction & recovery	-	-	-	-	+	+

Table 6-76-4: RLC modes and functions in UTRAN uplink sidef

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

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 8-18-1 : 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, ~~QR (ffs)~~	Data	Not Defined	Not Defined
RLC-TR-DATA	Data	Data	Not Defined	Not Defined
CRLC-CONFIG	E/R			
CRLC-STATUS	Not Defined	EVC	Not Defined	Not Defined

Each Primitive is defined as follows:

a) RLC-AM-DATA-Req./Ind/Conf.
It is used for acknowledged data transmission mode of point-to-point connection between the same level user entities.

b) RLC-UM-DATA-Req./Ind.
It is used for unacknowledged data transmission mode of point-to-point connection between the same level user entities.

c) RLC-TR-DATA-Req./Ind
It is used for trasparent data transmission mode of point-to-point connection between the same level user entities.

d) CRLC-CONFIG-Req
It is used for establishment and release of point-to-point connection between the same level user entities.

e) CRLC-STATUS-Ind
It is used by the RLC to send status information to the upper layers.

The parameter Data is mapped onto the Data field in a RLC PDU transparently in case of RLC-AM-DATA-Req. or RLC-UM-DATA-Req. Conversely the Data field of an RLC PDU received is mapped onto Data in case of RLC-AM-DATA-Ind. or RLC-UM-DATA_nd. transparently. The length of Data must be n octets (n is integer).
~~The Quick Repeat indicator (QR) indicates whether UMD PDU will be transmitted with Quick Repeat or not. It holds one of two values: "Yes" or "No". *(The need of this indicator is FFS)*~~
The parameter Confirmation request (CNF) indicates whether RLC-AM-DATA conf. should be necessary or not.
The parameter Message Unit Identifier (MUI) makes a relationship between message unit and confirm primitive.
The parameter E/R indicates whether establishment or release of RLC connection should be performed.
The parameter Event Code (EVC) indicates reason for the message (i.e unrecoverable errors such as data link layer loss or recoverable status events such as reset, etc.)

9. Elements for peer-to-peer communication

In unacknowledged transmission, only one type of unacknowledged data PDU is exchanged between peer RLC entities In acknowledged transmission, both (acknowledged) data PDUs and control PDUs are exchanged between peer RLC entities.

9.1. Protocol data units

[All the section shall be reviewed when the protocol is defined]

9.1.1. Data PDUs

a) 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 the RLC when it is in the acknowledged 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 the RLC when using the unacknowledged data transfer.

9.1.2. Control PDUs

a) STATUS PDU
The STATUS PDU is used to inform the transmitting entity about missing PUs at the receving entity either upon detection of a missing PU (unsolicited) or as a response to a polling request from the transmitting entity.

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~~9-1~~: RLC PDU names and descriptions

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

9.2. Formats and parameters

[All the section shall be reviewed when the protocol is defined]

9.2.1. Formats

AMD PDU

Transfers user data and piggybacked status information and requests status report by setting Poll bit.

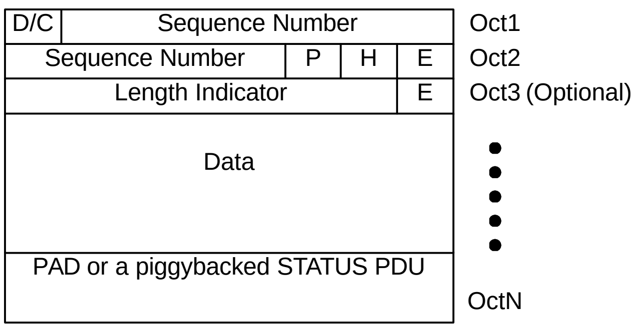

Figure 9-1. AMD PDU

<u>AMD PDU Extended Header</u>

If the H-flag of an AMD-PDU is set to 1, the first two bits of the next octet contain the Extended Header type field (EHType field) which define the length of the extension:

All the Extended Headers are listed for first then the PUs follow.

EHType	See EHType description	Oct1
		Oct2

Figure 9-2. AMD PDU Extended Header

UMD PDU

Sequence Number	E	Oct1
Length Indicator	E	Oct2 (Optional)
Data		• • • • • •
PAD		OctN

Figure 9-3. UMD PDU

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.

~~Reports the status of receiver to transmitter when AMD PDU with status report request is received, or to inform the transmitting entity about missing PUs.~~

[The message format will be reconsidered when the protocol will be defined]

D/C	PDU type	PA	$SUFI_1$	Octet 1
$SUFI_1$				Octet 2
~~$SUFI_1$~~				~~Octet 3~~
…				
$SUFI_K$				Octet N

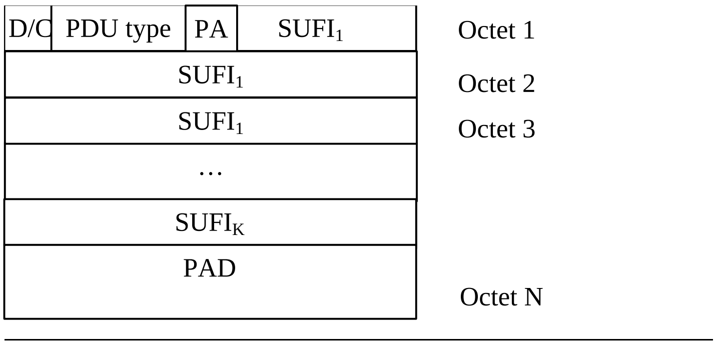

Figure 9-4. Status Information Control PDU (STATUS PDU)

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.

~~The inclusion of a credit value/retransmission window size in the STATUS PDU is FFS.~~
~~The maximum size of a STATUS PDU is bounded by the maximum RLC PDU size.~~

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.

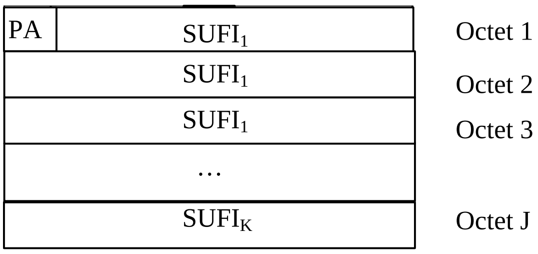

Figure 9-5. Piggybacked STATUS PDU

RESET, RESET ACK PDU

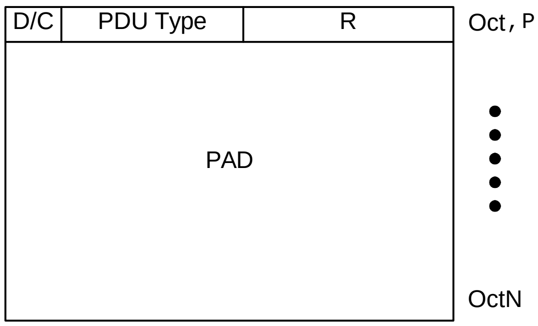

Figure 9-6. RESET, RESET ACK PDU

9.2.2. Parameters

~~The RLC PDU parameters are defined as follows:~~

D/C ~~-~~**field** ~~bit~~: 1bit
~~This~~ 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

—**PDU Type**: 3 bit length [FFS]
~~This~~ The PDU type field indicates the ~~type of~~ Control PDU type~~. They are indicated by the special values of sequence number field.~~

~~Bit~~	~~PDU Type~~	~~Bit~~	~~PDU Type~~
		~~001~~	~~STATUS~~
		~~FFS~~	~~RESET~~
		~~FFS~~	~~RESET ACK~~

Bit	PDU Type
000	STATUS
001	RESET
010	RESET ACK

—**Sequence Number (SN)**

This field indicates the sequence number of the payload unit. In a normal ~~acknowledged mode RLC~~AMD-PDU ~~header it is~~ the sequence number of the first PU in the PDU is indicated. If the PU~~:~~s 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

—**Polling bit (P)**: 1bit
This field is used to request a status report (STATUS PDU) from the receiver RLC.

Bit	Description
0	-
1	Request a status report

—**Extension bit (E)**: 1bit
This bit indicates whether the next octet will be header information (LI) or data.

Bit	Description
0	The next octet is data
1	The next octet is header information (LI)

—**Reserved (R)**:

One function of this field is to achieve octet alignment. Other functions are FFS. Where no functions are defined, this field shall be coded as zero. This field ignored by the receiver.

Header Extension flag (H): 1bit

Setting the header extension flag to 1 indicates the first two bits of the next octet contain the Extended Header type field (EHType field) in the AMD-PDU.

Extended Header Type (EHType): 2 bits

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

Value	Description
00	Extension consists of SN+H+E for additional PU sequence numbers
01	Reserved for future use
10	Reserved for future use
11	Reserved for future use

EHType 00:

This value is used when additional sequence numbers are needed to indicate PU:s that are not sequential within a PDU. The space to accommodate the extended header is provided by transmitting one PU less than normally with the same AMD PDU size. The decision to use this EHType is made by the transmitting RLC. The format is described in Figure 9-7.

0 0	Sequence Number			Oct1
Sequence Number		H	E	Oct2

Figure 9-7. AMD PDU Extended Header with EHType = 00

—**Length Indicator (LI)**: 7bit

This field is optional and is used if concatenation, padding or piggybacking takes place in RLC. It indicates the end of the last segment of a SDU. Some values are reserved for special purposes:

"0000000" indicates that the previous RLC PDU is exactly filled with the last segment of a RLC SDU;

"1111110" indicates that the rest part of the RLC PDU includes a piggybacked STATUS PDU;

"1111111" indicates that the rest part of the RLC PDU is padding.

—**Poll Answer (PA)**: 1bit

~~This~~ 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

—**SUFI** ~~(SUper FIeld)~~: variable number of bits

The SUFI (Super-Field) includes three sub-fields: type information (type of super-field, e.g. list, bitmap, ~~or~~ 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.

Type
Length
Value

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

—**Type:** ~~2~~3 bits (FFS)

~~Bit~~	~~Description~~
~~00~~	~~List (**LIST**)~~
~~01~~	~~Bitmap (**BITMAP**)~~
~~10~~	~~Acknowledgement (**ACK**)~~
~~11~~	~~No More Data (**NO_MORE**)~~

	Move Receiving Window **(MRW)**

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*

[Editor's note: the length of the "Type" field (2 bits) should be redefined in order to address all SUFI types defined by the RLC protocol]

—**Length:** depending on the super-field type
Gives the length of the variable size part of the following value field

—**Value:** variable number of bits given by the Type and the Length fields

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.

Type=**NO_MORE**

Figure 9-9. NO_MORE field in a STATUS PDU

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.

Type = **ACK**
LSN

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

LSN: 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. 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.

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

The List super-fieldSUFI for a List

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: 4 bits (FFS)

The number of (SN_i , L_i)-pairs in the super-field of type LIST.

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

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

The Bitmap super-fieldSUFI for a Bitmap

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

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

Bitmap: variable number of octets given by LENGTH
Status of the SNs in the interval [FSN, FSN + LENGTH*8 - 1] indicated in the bitmap where each position can have two different values (0 and 1) with the following meaning (bit_position$\in$[0,LENGTH*8 - 1]):

1: SN = (FSN + bit_position) has been correctly received

0: SN = (FSN + bit_position) has not been correctly received

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: 4 bits (FFS, could also be 5 to octet-align with 'Type')
The number of codewords (CW) in the super-field of type RLIST.

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

CW : 4 bits
The CW is interpreted as follows:

XXX 0 Next 3 bits of a number are XXX, the number continues in the next CW.

XXX 1 The number is terminated, the most significant bits are XXX.

By default, the CWs represent a distance from the previous indicated 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.

SUFI for an Acknowledgment

The Ack Super-Field consists of a type identifier field (ACK) and a sequence number (FSN) as shown in Figure 9-18 below:

Type = **ACK**
LSN

Figure 9-18. The ACK fields in a STATUS PDU

LSN: 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.

SUFI for aThe 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 9-16. The MRW fields in a STATUS PDU

~~**Type:** 3 bits~~
~~Bit combination 110 can be used for Move Receiving Window (MRW) command~~

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

~~**SUFI for No More Data (FFS)**~~
~~The 'No More Data' Super-Field indicates the end of the data part of a STATUS PDU and is shown in Figure 9-21 below:~~

Type=**NO_MORE**

~~**Figure 9-21. NO_MORE field in a STATUS PDU**~~

~~- N(R): 12bit~~
~~VR(R) is mapped to N(R) whenever a STAT or USTAT PDU is generated.~~
~~VR(MR) is mapped to N(R) whenever a STAT, USTAT, BGN, or BGAK PDU is generated. This is the basis for credit granting by the receiver.~~
~~- Header extension flag (H): 1bit~~
~~The header extension flag indicates that the following two octets contain an extended header (SN+H+E) in the AMD PDU. The use of this flag is *[F.F.S.]*~~

—**Data**:
In this field data from higher layer PDUs is mapped.

9.3. Protocol states

9.3.1. State model for transparent mode entities

Figure 9-18illustrates 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.

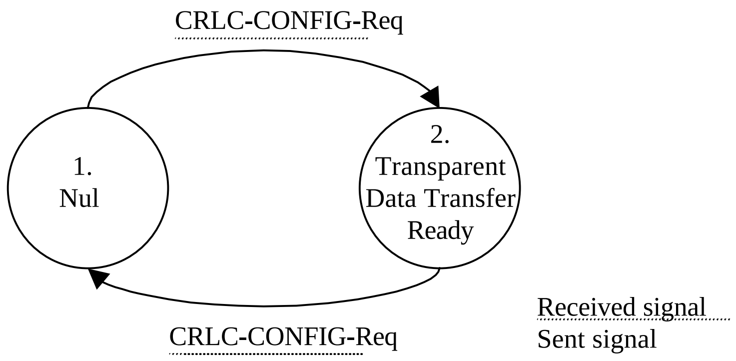

Figure 9-18. The state model for transparent mode entities.

9.3.2. State model for unacknowledged mode entities

Figure 9-19 illustrates the state model for unacknowledged mode RLC entities. 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, unacknowldged 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.

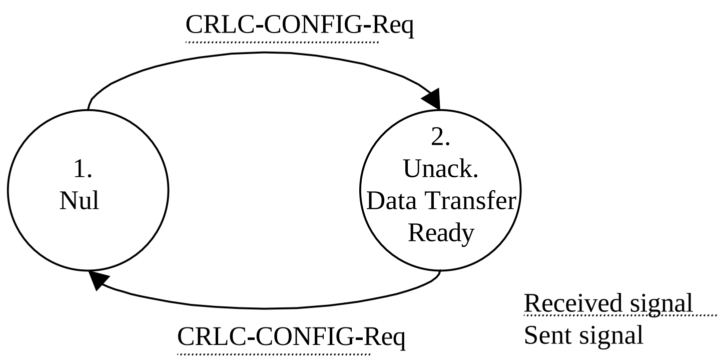

Figure 9-19. The state model for unacknoledged mode entities.

9.3.3. State model for acknowledged mode entities

Figure 9-20 illustrates the state model for the acknowledged mode RLC entity. 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, acknowledged mode data can be exchanged between the entities. Upon reception of an CRLC-CONFI-Rreq 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.

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.

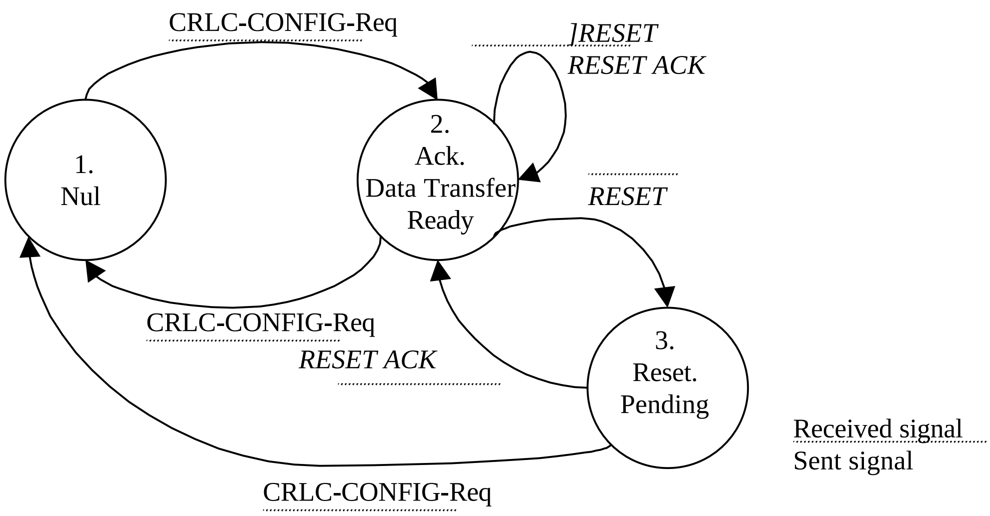

Figure 9-20. The state model for the acknoledged mode entities.

The messages that cause the transition between the "Ack. Data Transfer Ready" state and the "Recovery Pending" State, are [F.F.S.].

9.4. State variables

[All the section shall be reviewed when the protocol is defined]

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} and the sequence numbers cycle through the entire range, 0 through $2^{12} - 1$. All arithmetic operations on the following state variables and sequence numbers contained in this Recommendation are affected by the modulus: VT(S), VT(A), VT(MS), VR(R), VR(H), and VR(MR). 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. In addition, the state variables VT(SQ) and VR(SQ) use modulo 2 arithmetic and VT(US) and VT(UR) use modulo 48.

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). Incremented after transmission of a PU for the first time (i.e. excluding retransmission).

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~~upon acknowledgment of in-sequence PUs~~.

c) VT(DAT)
This state variable is used to count the retransmission number of each PU. VT(DAT) is incremented by sending PU.

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]. This value represents the upper edge of the transmit window. The transmitter shall not transmit a new PU if VT(S) ≥= 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 means new sequence number of UMD-PDU which will send next. After new UMD-PDU is sent, VT(US) will be incremented.

~~f) VT(QR) - Quick repeat state variable (FFS)~~
~~This state variable is used to count the retransmission number when UMD-PDU is sent by quick repeat scheme. It is incremented after UMD-PDU is sent and quick repeat will be continued until VT(QR) becomes to equal MaxQR.~~

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.

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.

h) VT(RST) - Reset state variable
It is used to count the retransmission number of RESET PDU. VT(RST) is incremented by sending RESET PDU. VT(RST) is reset upon the reception of a RESET ACK PDU.

-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. Incremented upon receipt of the next in-sequence PU.

b) VR(H) - Highest expected state variable
The sequence number of the next highest expected PU. This state variable is updated whenever a new PU is received.

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]. The receiver shall discard PUs with N(S) ≥= VR(MR), (in one case, such a PU may cause the transmission of an unsolicited STATUS PDU). Updating VR(MR) is implementation dependent, but VR(MR) should not be set to a value < VR(H).

d) VR(US) - Receiver Send Sequence state variable

The sequence number of the latest UMD PDU to be received. It is used to check the duplication receive. When new UMD PDU is received, VR(US) is compared with N(US). If VR(US) is equal to N(US), this PDU is quashed because duplication receive happens. And if not, N(US) is substituted for VR(US).

e) VR(EP) – Estimated PDU Counter state variable (FFS)
The number of PUs that should have been received after the latest STATUS PDU was sent. In acknowledged mode, this state variable is updated at the end of each transmission time interval. It is incremented by the number of PUs that should have been received during the transmission time interval. If VR(EP) is equal to the number of requested PUs in the latest STATUD PDU, then check if all PUs requested for retransmission have been received.

9.5. Timers

[All the section shall be reviewed when the protocol is defined]

a) Timer_~~STATUS~~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 with the PA bit set. The value of the timer is signalled by RRC.

If the timer expires and no STATUS PDU with the PA bit set has been received the receiver is polled once more and the timer is restarted.

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

~~It is used to detect the loss of the response from receiver side. This timer is set when transmitted AMD PDU requests status report (i.e. P bit is set to “1”). And it will be stopped when the transmitter receives an Acknowledgement of the PUs in that AMD PDU by STATUS PDU or Non Acknowledgment (Nack) by the STATUS PDU. When this timer is over, the PUs of the oldest unconfirmed AMD PDU should be retransmitted with requesting status report, and this timer is set again. If polling is taken place during this timer is active the timer will be stopped and set again.~~

b) Timer_Poll_Prohibit

This timer is only used when the poll prohibit function is used. It is used to prohibit transmission of ~~polling message~~ polls within a certain period. If polling is taken place during this timer is active, it will be once stopped and set again. This timer will not be stopped by ~~Ack or Nack~~a STATUS PDU. When this timer expires no action is performed. The value of the timer is signalled by RRC. ~~*[the values recommended for this timer are [FFS]*~~

~~c)Timer_QR (FFS)~~

~~Transmission interval of quick repeat for UMD PDU.~~

~~d)~~c) -Timer_EPC ~~(FFS)~~

This timer is only used when the EPC function is used and it ~~This timer~~ 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.~~The value of Timer_EPC is heavily based on the transmission time interval (corresponding to the Layer 1 interleaving depth). When changing the transmission time interval, then the value of the EPC timer also needs to be changed.~~

~~e)~~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

[All the section shall be reviewed when the protocol is defined]

The value of each RLC protocol parameter is application specific and may be defined in another Recommendation which references this Recommendation.

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)MaxQR (FFS)~~
~~Maximum successive transmission number of UMD PDU. This parameter is an upper limit for counter VT(QR).~~

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 $\frac{VT(MS)-VT(S)}{VT(MS)-VT(A)} < Poll\_Window$.

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.

9.7. Specific functions

[All the section shall be reviewed when the protocol is defined]
[The Hybrid ARQ (Type II/III) mechanism scheme is considered for the downlink only; in particular an incremental effort of protocol implementation will be followed. In this way it is possible to estimate the real performance of this scheme but also the impact that the introduction of such a scheme has on the protocol implementation. It is [FFS] if Hybrid ARQ mechanism for the downlink is mandatory for the UE (for both TDD and FDD)]

9.7.1. Retransmission Scheme

9.7.1.1. Basic Concept

1) Type of retransmission
   - Selective retransmission

2) Acknowledgement Confirmation
   - Receiver Status Report in response to the Transmitter Polling Request;
   - Unsolicited status report from the receiver caused by detecting the latest loss of PUs.

3) The Retransmission takes place when:
   - A Status Report (STATUS (PA=Yes)), transmitted by the Receiver, is received;
   - An Unsolicited Status Report (STATUS (PA=No)) is received;
   - Retransmission timer expires.

4) Timing of polling
   Basically to confirm acknowledgement every RLC SDU can reduce extra overhead and improve the throughput. But if small RLC SDUs are given continuously, many status reports will be transmitted and it will cause overhead. To solve the problem, this retransmission scheme uses the timer which prohibits excessive polling. This timer only prohibits polling for every RLC SDU.

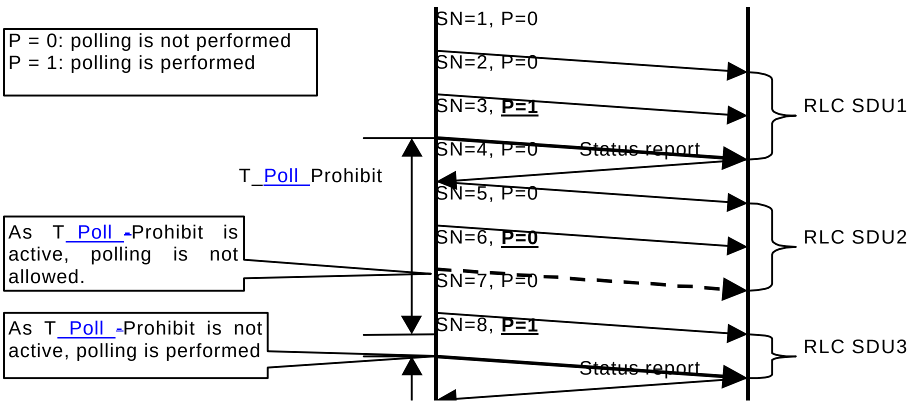

Figure 9-21 : Reduction of excessive polling

9.7.1.2. Outline of proposed retransmission scheme

[In this section it is described the outline of the retransmission scheme presented. The list of the PDU and timers used in the retransmission scheme described are reported in this section, also if a complete description of them is presented in the proper sections of this specification. In chapter 12 are also reported the SDL diagrams of this scheme that actually are [FFS]]

9.7.1.3. PDUs used for retransmission

Following PDUs are necessary for this retransmission scheme.

- AMD PDU:

Transfers user data and requests status report by setting Poll bit.

- STATUS PDU:

A STATUS PDU (PA=Yes) reports the status of receiver to transmitter when AMD PDU with status report request is received.The STATUS PDU (PA=No) is transmitted upon detection of an erroneous transmission of one or more data PUs. It is used to inform the transmitter side about missing PUs at the receiver RLC.

9.7.1.4. Timers used for retransmission

Following two timers are necessary for this retransmission scheme.

- Timer_~~STATUS~~Poll:

This timer is set when AMD PDU with polling (i.e. P bit is set to "1") is transmitted. And it will be stopped when the transmitter receives Ack or Nack for the AMD PDU with polling. If polling is taken place during this timer is active, it will be once stopped and set again.

- Timer_Poll_Prohibit:

This timer is set when AMD PDU with polling is transmitted. If polling is taken place during this timer is active, it will be once stopped and set again. This timer will not be stopped by Ack or Nack. When this timer expires no action is performed.

9.7.1.5. Trigger of Polling

Polling message is transmitted when;the last segment of AMD SDU is transmitted (every RLC SDU) and Timer_Poll_Prohibit is not active
the retransmission timer (Timer_~~STATUS~~Poll) expires
the last PU in the transmission queue is transmitted
the transmitter window has to move
the last PU among those which are requested to be retransmitted by a STATUS (PA=Yes) is transmitted.

9.7.1.6. Flow of retransmission

Case1) In case STATUS (PA=Yes) -is received:

- Polling is performed from the transmitter comply with the polling trigger.
- At this moment, the transmitter sets Timer_~~STATUS~~ Poll and Timer_Poll_Prohibit.
- The receiver transmits STATUS (PA=Yes) which requests retransmission for SN = 2, 3, 4 of PU in response to the polling.
- The transmitter receives the STATUS and stops Timer_~~STATUS~~Poll. Then the PUs (SN = 2, 3, 4) requested by the STATUS are retransmitted.
- When the PU whose SN =4 is transmitted, poll bit is set to 1 (polling is performed).
- At this moment, Timer_~~STATUS~~ Poll is set and Timer_Poll_Prohibit is reset.
- When the transmitter receives STATUS (PA=Yes) from the receiver, Timer_~~STATUS~~ Poll is stopped.

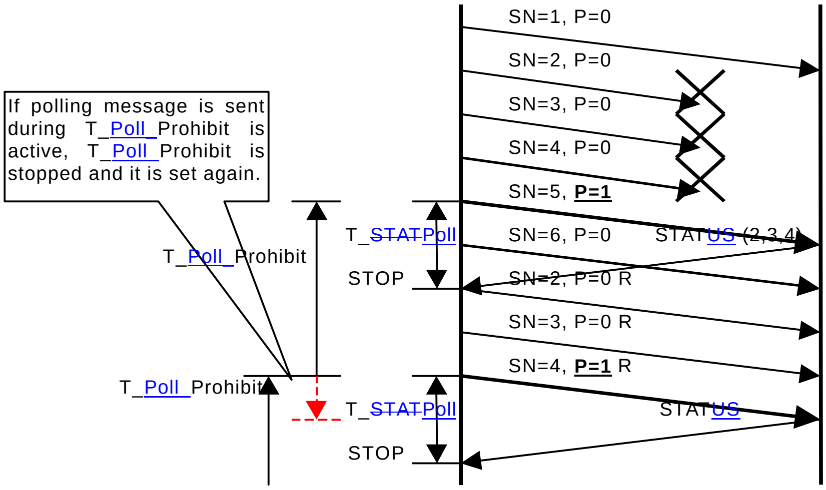

Figure 9-22 : Retransmission Scheme Behaviour when STATUS (PA=Yes) is received

- Case2) In case STATUS (PA=No) is received:
- If the receiver detects new loss of PUs, it transmits STATUS (PA=No) which requests retransmission of SN = 2, 3 of PU to the transmitter.
- The transmitter receives the STATUS and retransmits the requested PUs (SN = 2,3).
- In this case polling is not performed when the PU whose SN =3 is retransmitted.

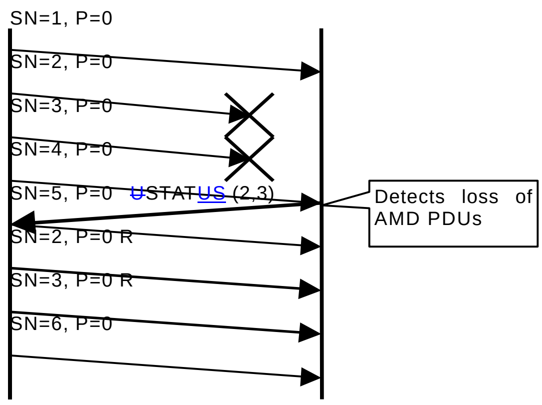

Figure 9-23 : Retransmission Scheme Behaviour USTATUS (PA=No) is received

Case3) In case Timer_STATUS Poll expires:

- If Timer_STATUS Poll expires due to loss of the AMD PDU with polling or STATUS, the transmitter retransmits the AMD PDU with polling.
- At this moment, Timer_STATUS Poll is set and Timer_Poll_Prohibit is reset.
- When the transmitter receives STATUS from the receiver, Timer_STATUS Poll is stopped.

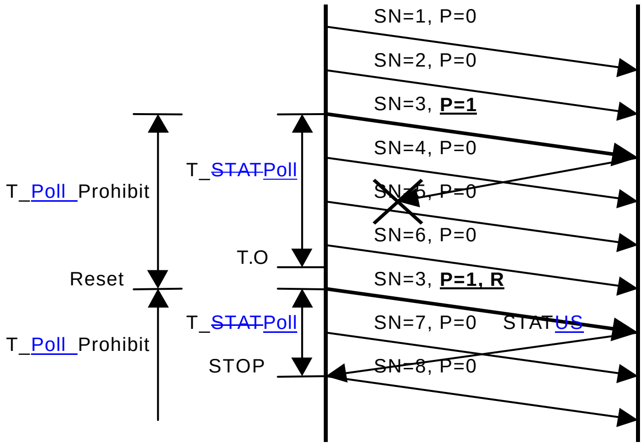

Figure 9-24 : Retransmission Scheme Behaviour when Timer_STAT Poll expires

[The Timer_Poll_ T Prohibit is related only to the UE; there's no need for "Timer_Poll__Prohibit Timer" in the network as it is an implementation aspect. The values for Timer_Poll__Prohibit Timer are [F.F.S]]

9.7.2. 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 preliminary list of operation modes for the RLC SDU discard function.

Table 9-39 3. 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 X number of retransmissions	Network controlled

9.7.2.1. Timer based discard, with explicit signalling

This alternative uses a timer based triggering of SDU 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.

9.7.2.2. Timer based discard, without explicit signalling

This alternative uses the same timer based trigger for SDU discard as the one described in the section 9.7.2.1. 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.

9.7.2.3. SDU discard after X 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.

9.7.3. The Estimated PDU Counter

The Estimated PDU Counter is one of the possible mechanisms, which can be defined for scheduling the retransmissio of status reports in the receiver side. With this mechanism, the receiver will send a new Status PDU in which it requests for ~~PDUs~~ 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 ~~PDUs~~ PUs that should have been transmitted during that transmission time interval.When the receiver detects that the PDUs are missing it generates and sends a Status PDU to the transmitter and sets the EPC equal to the number of requested PUs~~PDUs~~.
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 ~~DATA~~ AMD-PDU is received. When the EPC exceeds the number of outstanding PUs ~~PDUs~~ (i.e. the PUs ~~PDUs~~ which were requested to be retransmitted) and not all of these requested PUs ~~PDUs~~ have been received correctly, a new Status PDU will be transmitted and the EPC will be reset to zero. The EPC timer will be started once more.
The EPC is based on the estimation of the number of PUs ~~PDUs~~ 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 ~~PDUs~~ 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.

9.7.4. 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 9-25 shows some examples where the RLC AMD PDU size has been set to 320 bits.

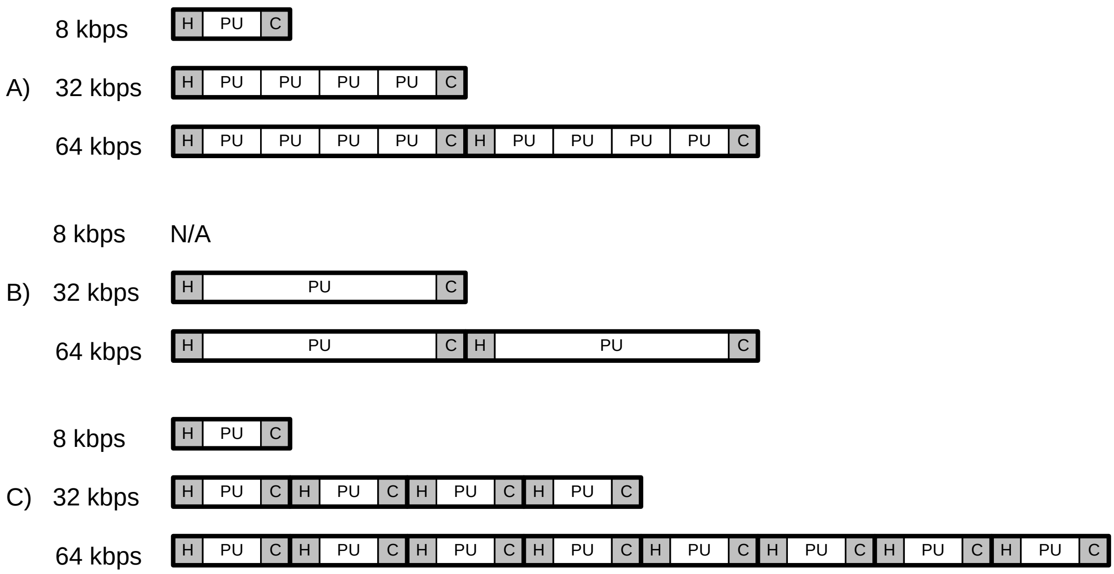

Figure 9-25. Example cases of payload units and PDUs at different transmission rates

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).

9.7.4.9.7.5.Credit and peer-to-peer flow control

Credit is granted by the RLC receiver to allow the peer RLC transmitter to transmit new AMD PDUs. The process by which a receiver entity determines credit is not subject to standardization, but is related to the buffer availability and the bandwidth/delay of the connection.

Details of the usage of Crediting is FFS.

9.7.5.9.7.6. Local flow control

RLC events, such as reception of PDUs and external and internal signals, are normally processed in the order in which they occurred. However, events pertaining to the exchange of RLC link status information have priority over data transfer.

An implementation may detect congestion (for example, a long queuing delay) in its lower protocol layers. If so, data transfer should be temporarily suspended in order to give priority to connection control messages. The means by which an RLC entity decides whether or not it is congested depends on the protocol environment, including protocol timer values, and is not subject to standardization.

If a RLC entity detects local congestion ("lower layer busy" in the SDL specification), it can elect to suspend the servicing of RLC-AM-DATA.request, RLC-UM-DATA.request It can also suspend the retransmission of requested AMD PDUs. The data transfer procedures allow this to occur without causing protocol errors.

Therefore, in terms of transmitting PDUs to the peer receiver, all types of PDUs except AMD PDU and UMD PDU are given highest priority. The AMD PDUs and UMD PDUs have equal priority. Among the AMD PDUs, retransmission have priority over new transmission if both types are pending. These priorities are only internal to RLC.

9.8. RLC Toolbox concept

The RLC toolbox concept specifies a number of basic functions. These functions can then be combined in different ways in order to get a complete and functional protocol. How to combine the different functions is signalled by RRC, before setting up a new RLC entity. The toobox concept may be applied to both transparent and non-transparent mode entities.

9.8.1. Toolbox concept for acknowledged mode RLC entities.

This section describes the different functions included in the RLC toolbox for acknowledged mode. This section concentrates on functions for ARQ mechanism, but other function can be added to the toolbox. The functions have been divided into two groups, transmitting side functions and receiving side functions. It is for FFS if all functions have to be supported by the UE. The presence field rather indicates if the function is always supported by the acknowledged mode entity or wheter its implementation is network controlled. No explicit signalling is needed if the function is always supported.

9.8.1.1.Transmitting side functions

9.8.1.1.1.When to poll

It is optional to apply a polling mechanism on transmitter side. If a polling mechanism is applied, Table 9-4 below summerizes the functions that control when the transmitter should poll the peer entity for a status report.

Table 9-4 List of functions that control when to poll the receiver for a status report.

Trigger	*Presence*
Last PU in buffer.	Always
Poll timer.	Always
Every X PU.	Network controlled
Every X SDU.	Network controlled
Last PU in retransmission buffer.	Network controlled

X% of transmission window.	Network controlled
Timer based.	Network controlled
$T_{prohibit}$	Network controlled

[The list of parameters shall be reviewed during the evolution of the prodocol definition]

- **Last PU in buffer**

The transmitting side polls the peer entity for a status report, when the last PDU in the transmission buffer is transmitted. This function is mandatory for the transmitting side, if polling should be applied.

- **Poll timer**

The poll timer is started when a poll is transmitted to the peer entity and if no status report has been received before the poll timer expires a new poll is transmitted to the receiver. The value of the timer is signalled by RRC. This function is mandatory for the transmitting side, if polling should be applied.

- **Every X PU**

The transmitting side polls the peer entity for a status report every X PU. The value of X is signalled by RRC. This function is optional for the transmitting side.

- **Every X SDU**

The transmitting side polls the peer entity for a status report every X SDU. The value of X is signalled by RRC. This function is optional for the transmitting side.

- **Last PU in retransmission buffer**

The transmitting side polls the peer entity for a status report at transmission of the last PDU in the retransmission buffer.

- **X% of transmission window**

The transmitting side polls the peer entity for a status report when it has reached X % of the transmission window. The value of X is signalled by RRC. This function is optional for the transmitting side.

- **Timer based**

The transmitting side polls the peer entity for a status report periodically. The value of the time period is signalled by RRC. The function is optional for the transmitting side.

- **$T_{prohibit}$**

This function controls how often the transmitting side is allowed to poll the peer entity. The $T_{prohibit}$ is started when a poll is transmitted to the peer entity. As long as the timer is running the transmitting side is not allowed to poll the peer entity. The value of the timer is signalled by RRC. This function is optional for the transmitting side.

9.8.1.1.2.How to react upon a status report

Table 9-6 below summerizes the functions that control how to react upon a status report.

Table 9-6 List of functions that control how to react upon a status report.

Trigger	*Presence*
Adjust transmission window.	Always
Retransmit PUs.	Always
Plausibility check.	Network controlled

- **Adjust transmission window**

The transmission window should be updated according to the received status report. It is mandatory for the transmitting side to support this function.

- **Retransmit AM PUs**

This function retransmits the AM PUs that are requested by the status report. If no plausibility check is applied, the PUs shall be retransmitted immediately and have higher priority than new AM PUs. This function is mandatory for the transmitting side.

- **Plausibility check**

This function checks whether the contents of a status report is reasonable or not. It can prohibit or delay retransmissions requested by a status report. For example, the status report could contain negative acknowledgements of PUs which may not had arrived at the receiver before the status report was transmitted. The transmitter should not retransmit these PUs. This function is optional for the transmitting side.

9.8.1.2.Receiving side functions

9.8.1.2.1.How to react upon a poll

The receiving side should send a status report if it receives a poll. It should send the status report immediately. This function is mandatory for the receiving side.

9.8.1.2.2.When to send a status report

Table 9-8 below summerizes the functions that control when to send a status report.

Table 9-8 List of functions that control when to send a status report.

Trigger	*Presence*
Reception of poll.	Always
EPC	Network controlled
Detection of missing PU(s).	Network controlled
Every X SDU.	Network controlled
Every X PU.	Network controlled
X% of receiving window.	Network controlled
Timer based.	Network controlled
$T_{prohibit}$	Network controlled

- **Reception of poll**

The receiving side sends a status report to the peer entity upon reception of a poll, see section 9.8.1.2.1. The status report should be transmitted immediately. This function is mandatory for the receiving side.

- **EPC**

The EPC is started when a status report is transmitted to the peer entity. If not all AM PDUs requested for retransmission have been received before the EPC has expired a new status report is transmitted to the peer entity. A more detailed description of the EPC mechanism is given in section 9.7.3.

- **Detection of missing PU(s)**

The receiving side sends a status report to the peer entity upon detection of missing AM PU(s). The status report should be transmitted immediately. This function is optional for the receiving side.

- **Every X SDU**

The receiving side sends a status report to its peer entity every X SDU. The value of X is signalled by RRC. This function is optional for the receiving side.

- **Every X PU**

The receiving side sends a status report to its peer entity every X PU. The value of X is signalled by RRC. This function is optional for the receiving side.

- **X% of receiving window**

The receiving side sends a status report when X % of the transmission window has been reached. The value of X is signalled by RRC. This function is optional for the receiving side.

- **Timer based**

The receiving side sends a status report periodically to the peer entity. The value of the time period is signalled by RRC. The function is optional for the receiving side.

- **$T_{prohibit}$**

This function controls how often the receiving side is allowed to send status reports the peer entity. The $T_{prohibit}$ is started when a status report is transmitted to the peer entity. As long as the timer is running the receiving side is not allowed to send a status reports to the peer entity. The value of the timer is signalled by RRC. This function is optional for the receiving side.

10. Handling of unknown, unforeseen and erroneous protocol data

A preliminary list of possible error cases is reported below:

a) Inconsistent state variables

If the RLC entity receives a PDU including "erroneous Sequence Number", state variables between peer entities may be inconsistent. Following shows "erroneous Sequence Number" examples;

- Sequence Number of an AMD PDU is bigger than "Highest expected state variable (VR(H))",

- Each Sequence Number of missing PU informed by SUFI LIST or BITMAP parameter is not within the value between "Acknowledge state variable(VT(A))" and "Send state variable(VT(S))", and

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

b) Repeated failed retransmission over MaxDAT times

It is the case when the number of failed retransmission is over the maximum defined number of retransmission (MaxDAT). The repeated failed retransmission over MaxDat times is an unrecoverable error.

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

The transparent mode data transfer procedure is used for transferring of data between two RLC peer entities, which are operating in transparent mode. The procedure may be initiated either by the UE or by the UTRAN. Figure 11-1 below illustrates the elementary procedure for transparent mode data transfer.

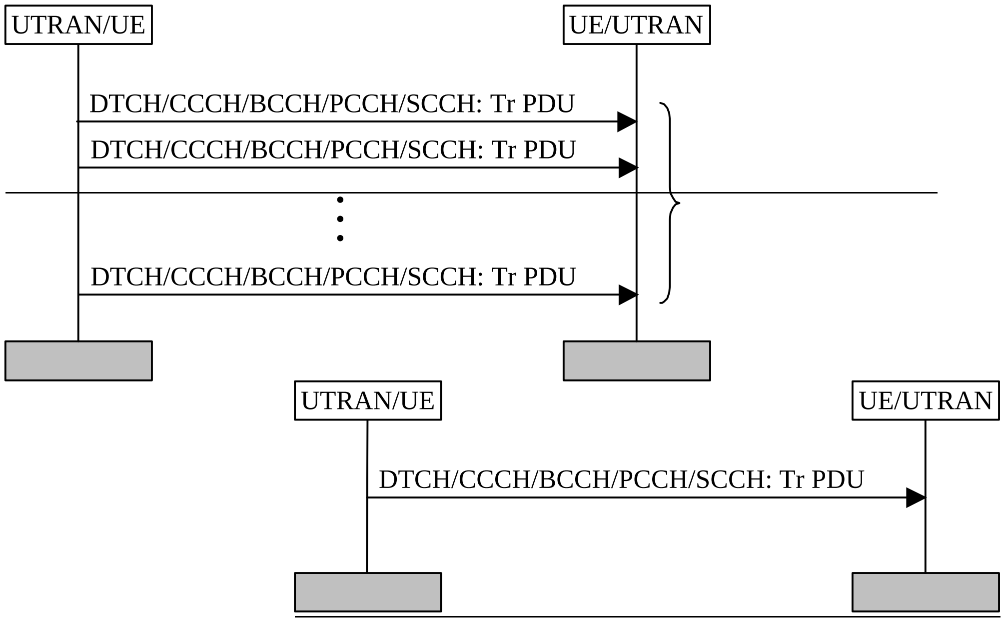

Figure 11-1. Transparent mode data transfer procedure.

The UTRAN/UE sends one or several Tr PDUs in one transmission time interval on one of the logical channels DTCH, CCCH, BCCH, PCCH or SCCH to the UE/UTRAN. The number of Tr PDUs depends on the rate of the logical channel and 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/PCCH,SCCH). The Tr PDU includes a complete or a segment of a ~~higher layer PDU~~SDU.

[Note: There is no PDU specified for transparent mode data transfer in section 9]

11.2.Unacknowledged mode data transfer procedure

The unacknowledged mode data transfer procedure is used for transferring of data between two RLC peer entities, which are operating in unacknowledged mode. The procedure may be initiated either by the UE or by the UTRAN. Figure 11-2 below illustrates the elementary procedure for unacknowledged mode data transfer.

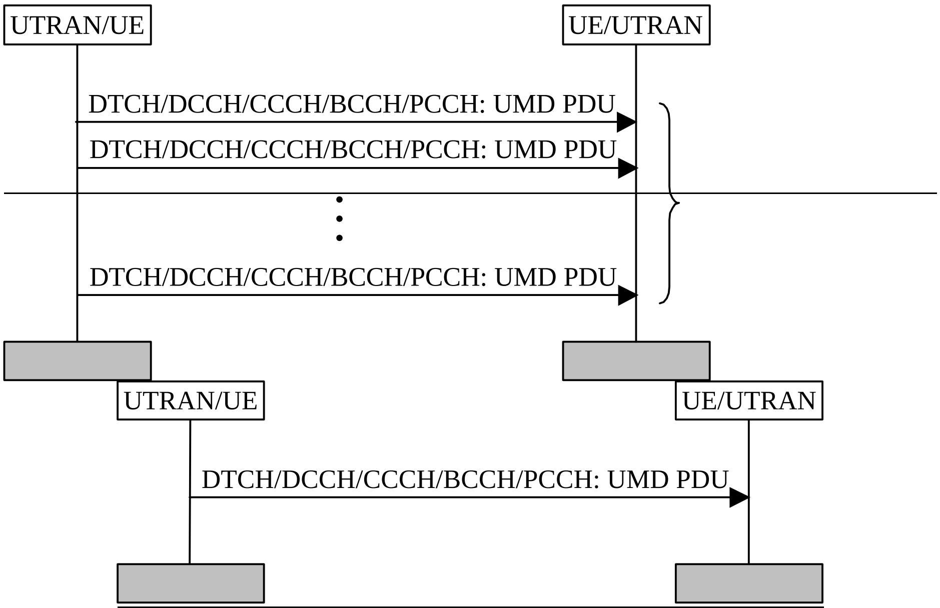

Figure 11-2. Unacknowledged mode data transfer procedure.

The UTRAN/UE sends one or several UMD PDUs in one transmission time interval on one of the logical channels DTCH, DCCH, CCCH, BCCH or PCCH to the UTRAN/UE. The number of UMD PDUs depends on the rate of the logical channel and 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/BCCH/PCCH). The UMD PDU includes a segment of one or several ~~higher layer PDU~~SDUs. It also includes a sequence number and one or several length indicator fields.

11.3.Acknowledged mode data transfer procedure

The acknowledged mode data transfer procedure is used for transferring of data between two RLC peer entities, which are operating in acknowledged mode. The procedure may be initiated either by the UE or by the UTRAN. Figure 11-3 below illustrates the elementary procedure for acknowledged mode data transfer.

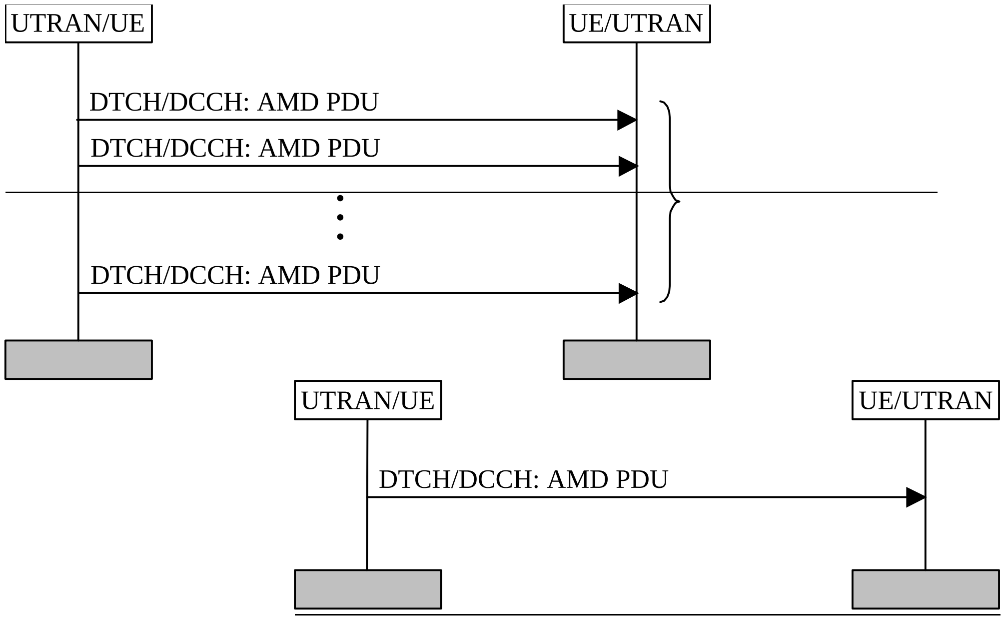

Figure 11-3. Acknowledged mode data transfer procedure.

The UTRAN/UE sends one or several AMD PDUs in one transmission time interval on either the DTCH or the DCCH logical channel to the receiver UE/UTRAN. The number of AMD PDUs depends on the rate of the logical channel and the type of logical channel depends on if the RLC entity is located in the user plane (DTCH) or in the control plane (DCCH). The AMD PDU includes a segment of one or several ~~higher layer PDU~~SDUs. It also includes a D/C field (which indicates that it is data PDU), a sequence number, polling bit, header extension bit and one or several length indicator fields.

11.4.RLC reset procedure

The RLC reset procedure is used to reset two RLC peer entities, which are operating in acknowledged mode. It is triggered when a protocol error occurs in RLC and it may be initiated either by the UE or by the UTRAN. Figure 11-4 below illustrates the elementary procedure for a RLC reset.

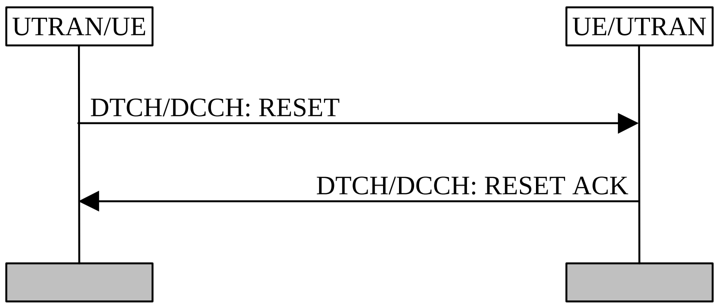

Figure 11-4. RLC reset procedure.

The UTRAN/UE sends a RESET PDU on a DTCH or a DCCH logical channel to the receiver UE/UTRAN. The type of logical channel depends on if the RLC entity is located in the user plane (DTCH) or in the control plane (DCCH). The RESET PDU includes the RLC parameters needed to perform the reset.

Upon reception of the RESET PDU, the receiver responds with a RESET ACK PDU.

11.5.STATUS PDU transfer procedure

The STATUS PDU transfer procedure is used for transferring of status information between two RLC peer entities, which are operating in acknowledged mode. The procedure may be initiated either by the UE or by the UTRAN. Figure 11-5 below illustrates the elementary procedure for STATUS PDU transfer procedure~~acknowledged mode data transfer~~.

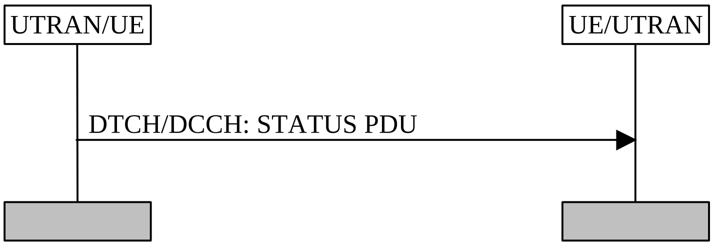

Figure 11-5. STATUS PDU transfer procedure.

~~The procedure is triggered when e.g. a missing AMD PDU is detected or a poll has been received.~~ There are several triggers of this procedure, see section 11.5.1.The originator UTRAN/UE sends STATUS PDUs on either the DTCH or the DCCH logical channel to the receiver UE/UTRAN. The type of logical channel depends on if the RLC entity is located in the user plane (DTCH) or in the control plane (DCCH). The STATUS PDU includes D/C field, PDU type field and ~~information about received data PDUs~~one or several SUFIs. This procedure may trigger retransmission of lost PU.
The receiver of the STATUS PDU may react in several ways, which are described in section 11.5.2.

11.5.1. Triggers of the STATUS report procedure

The status report procedure can be triggered in many different ways. Some triggers are always used and some are controlled by the network. RRC signals during setup of the RLC entity which, network controlled triggers that should be used. Table 11-1 below summerizes the triggers of the STATUS report procedure. It is FFS if all triggers have to be supported by the UE.

Table 11-1: List of triggers of the STATUS report procedure.

Trigger	*Presence*
Reception of poll.	Always
EPC	Network controlled
Detection of missing PU(s).	Network controlled
Timer based.	Network controlled
STATUS PDU prohibit	Network controlled
SDU discard	Network controlled

- **Reception of poll**

The receiving side sends a STATUS PDU, which at least contains information about received and known missing PUs within the receiving window, to the peer entity upon reception of a poll. The STATUS PDU should be transmitted immediately, except when the EPC or EPC timer is running. This trigger should always be present.

- **EPC**

The EPC is started when a STATUS PDU, which at least contains information about received and known missing PUs within the receiving window, 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, which at least contains information about all received and all known missing PUs within the receiving window, is transmitted to the peer entity. A more detailed description of the EPC mechanism is given in section 9.7.3. The network controls if this trigger should be used.

- **Detection of missing PU(s)**

The receiving side sends a STATUS PDU, which at least contains information about received and known missing PUs within the receiving window, to the peer entity upon detection of missing PU(s). The STATUS PDU should be transmitted immediately. The network controls if this trigger should be used.

- **Timer based**

The receiving side sends STATUS PDUs , which at least contains information about received and known missing PUs within the receiving window, periodically to the peer entity. The time period is controlled by the timer Timer_Status_Periodic and the value of the timer is signalled by RRC. The network controls if this trigger should be used.

- **STATUS PDU prohibit**

This is not a trigger, but rather a function that controls how often the receiving side is allowed to send STATUS PDUs to the peer entity. 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 value of the timer is signalled by RRC. The network controls if this function should be used.

- **SDU discard**

The transmitting side triggers a STATUS report procedure when performing a SDU discard with explicit signalling. The STATUS PDU should contain at least a MRW SUFI. This trigger is not prohibited by the STATUS PDU prohibit function. The network controls if this trigger should be used.

11.5.2. How to act upon reception of a STATUS PDU

Table 11-2 below summerizes the different actions of a receiver of a STATUS PDU. These actions should always be performed when receiving a STATUS PDU. RRC signals during setup of the RLC entity, which network controlled actions that should be used.

Table 11-2: List of functions that control how to react upon a STATUS PDU.

Action	*Presence*
Adjust transmission window.	Always
Retransmit PUs.	Always

- **Adjust transmission window**

The transmission window should be updated according to the received STATUS PDU.

- **Retransmit PUs**

This action retransmits the PUs that are requested by the STATUS PDU. The PUs shall be retransmitted immediately and have higher priority than new PUs.

11.6.Poll procedure

The poll procedure is used by an acknowledged mode RLC entity for requesting status information from its peer entity. The procedure may be initiated either by the UE or by the UTRAN. Figure 11-6 below illustrates the elementary procedure for polling.

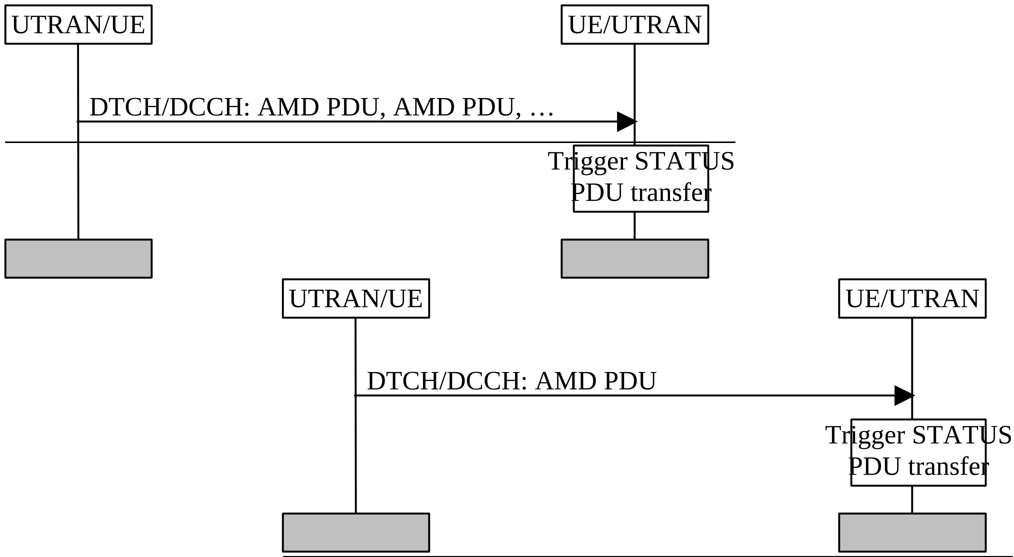

Figure 11-6. Poll procedure.

There are several triggers of this procedure, see section 11.6.1.~~The procedure is triggered when e.g. the last PU in the transmission buffer is transmitted.~~ The UTRAN/UE sends one or several AMD PDUs in one transmission time interval on either the DTCH or the DCCH logical channel to the receiver UE/UTRAN. The number of PDUs depends on the rate of the logical channel and the type of logical channel depends on if the RLC entity is located in the user plane (DTCH) or in the control plane (DCCH). It is FFS if the poll bit is set in all PDUs transmitted in the same transmission time interval.
Upon reception of the polls, the receiver triggers a STATUS PDU transfer procedure, see section 11.5.1.

11.6.1. Triggers of the poll procedure

The poll procedure can be triggered in many different ways. The network controls which triggers that should be used. RRC signals during setup of the RLC entity, which network controlled triggers that should be used. below summerizes the triggers of the STATUS report procedure. It is for FFS if all triggers indicated in Table 11-3 have to be supported by the UE.

Table 11-3: List of triggers of the poll procedure.

Trigger	*Presence*
Last PU in buffer.	Network controlled
Last PU in retransmission buffer.	Network controlled
Poll timer.	Network controlled
Every Poll_PU PU.	Network controlled
Every Poll_SDU SDU.	Network controlled
Poll_Window% of transmission window.	Network controlled
Timer based.	Network controlled
Poll prohibit	Network controlled

- **Last PU in buffer**

The transmitting side polls the peer entity for a STATUS PDU, when the last PDU in the transmission buffer is transmitted.

- **Last PU in retransmission buffer**

The transmitting side polls the peer entity for a STATUS PDU at transmission of the last PU in the retransmission buffer.

- **Poll timer**

The Timer_Poll is started when a poll is transmitted to the peer entity and if no STATUS PDU has been received before the Timer_Poll expires a new poll is transmitted to the receiver. The value of the timer is signalled by RRC.

- **Every Poll_PU PU**

The transmitting side polls the peer entity for a STATUS PDU every Poll_PU PU. All PUs should be counted, also retransmitted PUs. The value of Poll_PU is signalled by RRC.

- **Every Poll_SDU SDU**

The transmitting side polls the peer entity for a STATUS PDU every Poll_SDU SDU. The poll bit should be set in the PU that contains the last segment of the SDU. The value of Poll_SDU is signalled by RRC.

- **Poll_Window% of transmission window**

The transmitting side polls the peer entity for a STATUS PDU when it has reached Poll_Window % of the transmission window. The value of Poll_Window is signalled by RRC.

- **Timer based**

The transmitting side polls the peer entity for a STATUS PDU periodically. The time period is controlled by the timer Timer_Poll_Periodic and the value of the timer is signalled by RRC.

- **Poll prohibit**

This is not a trigger, but rather a function that controls how often the transmitting side is allowed to poll the peer entity. The timer Timer_Poll_Prohibit is started when a poll is transmitted to the peer entity. As long as the timer is running the transmitting side is not allowed to poll the peer entity.The value of Timer_Poll_Prohibit is signalled by RRC.

12. SDL diagrams

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

The resultant SDL diagrams (Timer_Prohibit scheme) are shown below:

Estimated PDU Counter (EPC) scheme (receiving side) (FFS)

1. Send a status report (STATUS (PA=Yes)), requesting for the retransmission of K number of missing PDUs.
2. Start Timer_EPC. This timer accounts for the roundtrip delay, i.e. the time when the first retransmitted PDU should be received.

3. When the timer expires, start counting the received PDUs, or rather the PDUs that should have been received using the state variable VT(EP)
4. If VT(EP) = K, then check if all PDUs (requested in the status report in step 1) have been received.
   a) If some of the previously missing PDUs are still missing, then repeat the procedure from step 1 for the PDUs that are still missing.
   b)If none of the previously missing PDUs are still missing, then no status report needs to be sent, unless a poll had been transmitted or a new missing PDU has been detected. In case of a poll or a new missing PDU, then repeat the procedure from step 1.

Every poll received during the time when the Timer_EPC is active and VT(EP) < K will be discarded by the receiving side, i.e. STATUS PDU will not be sent from the receiving side during this time.

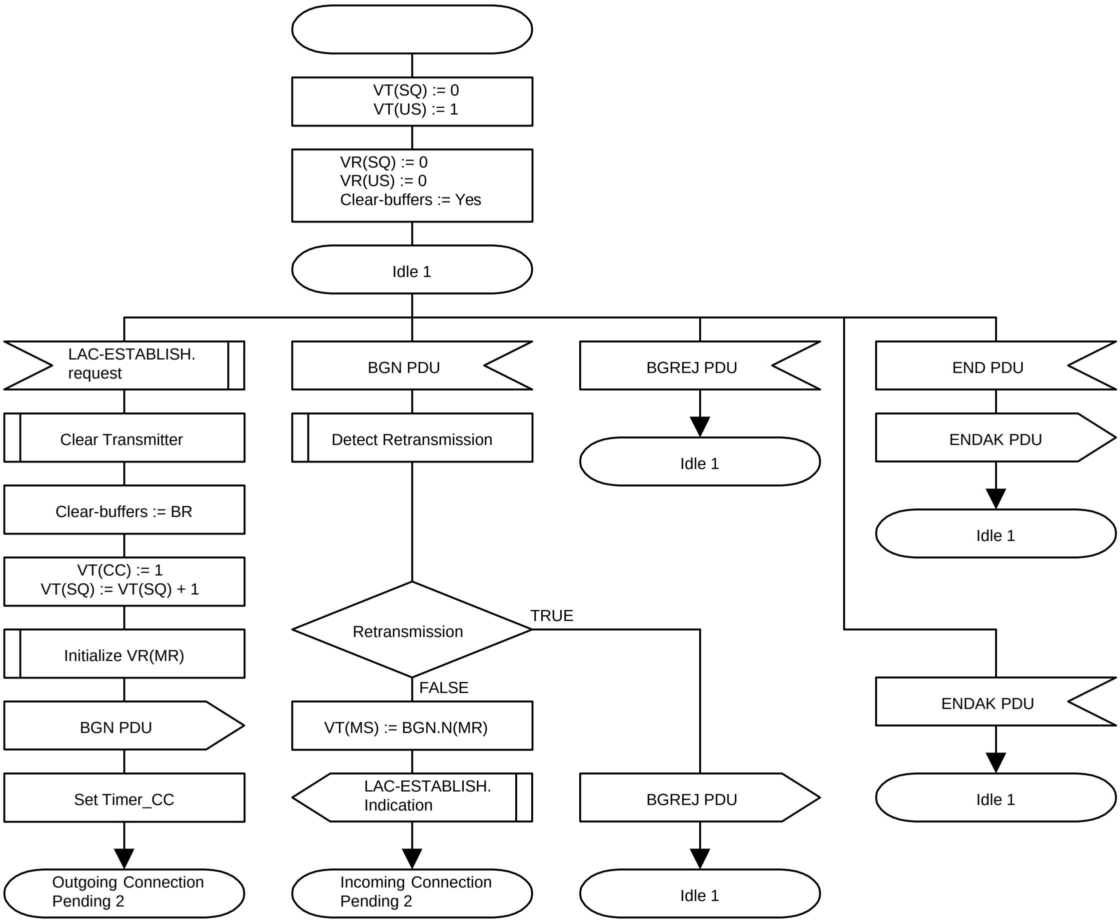

Figure 12-1

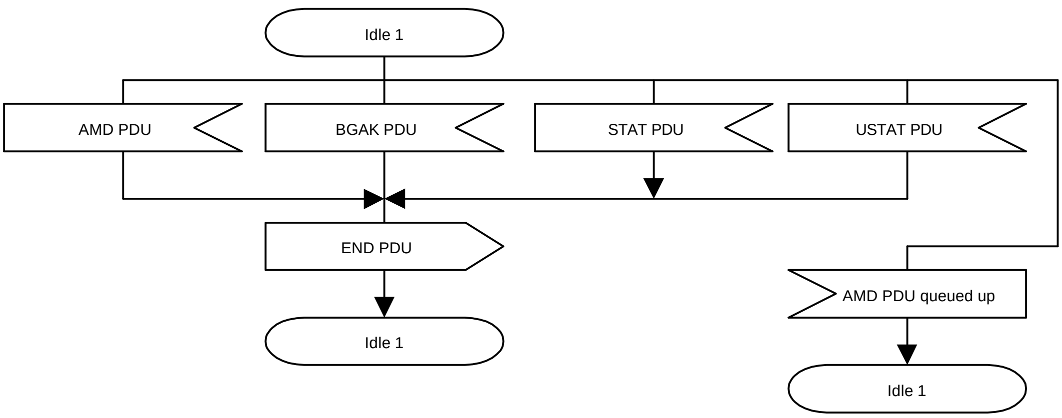

Figure 12-2

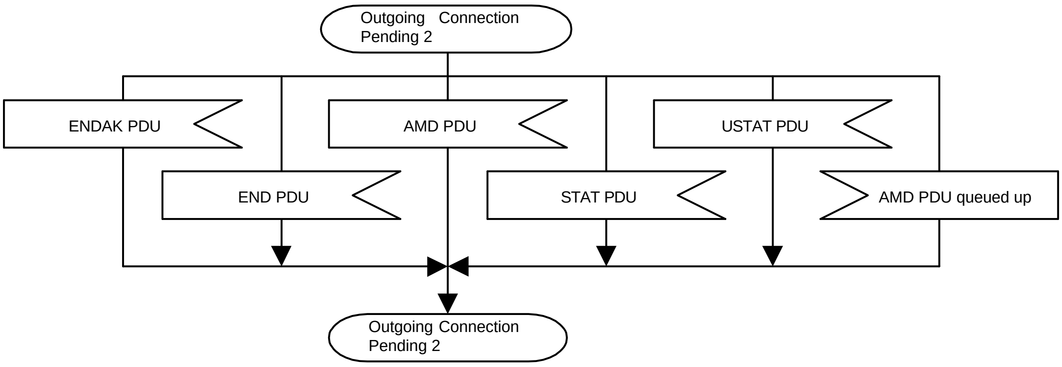

Figure 12-3

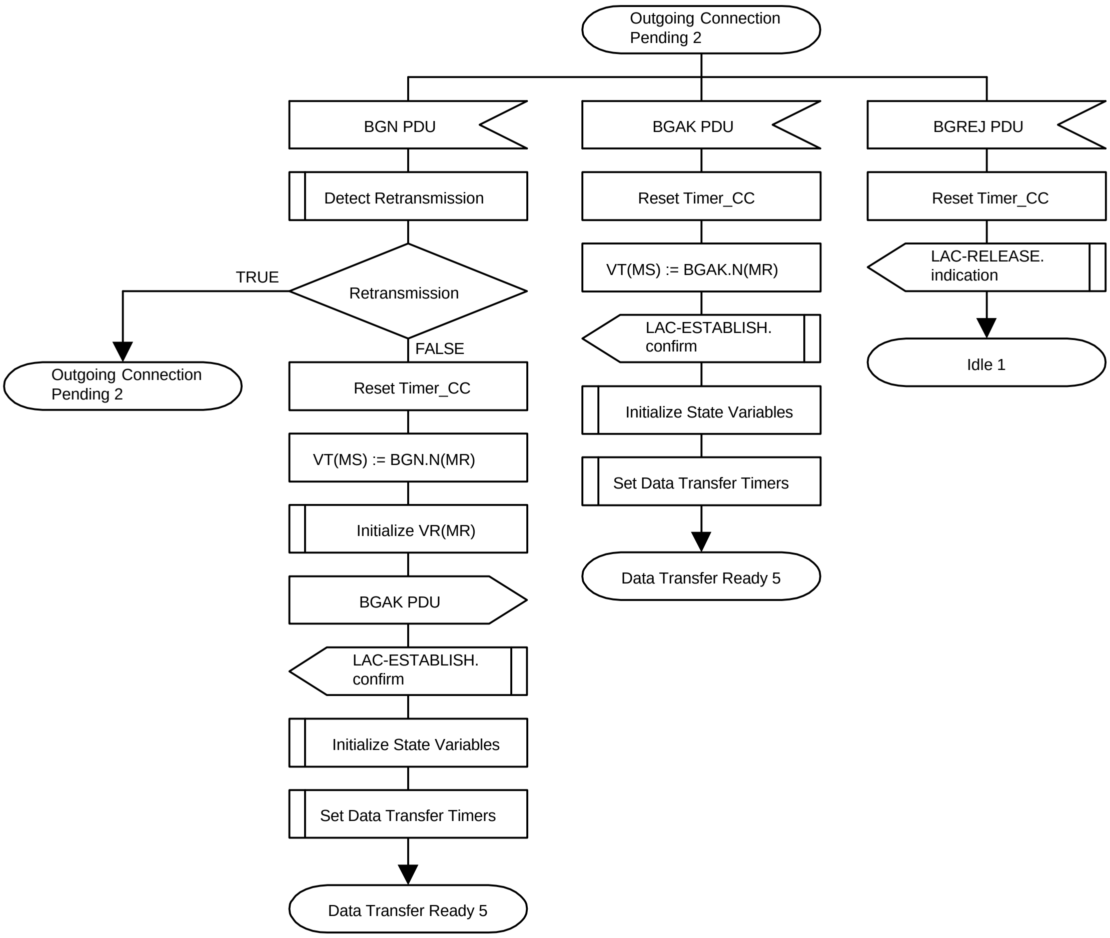

Figure 12-4

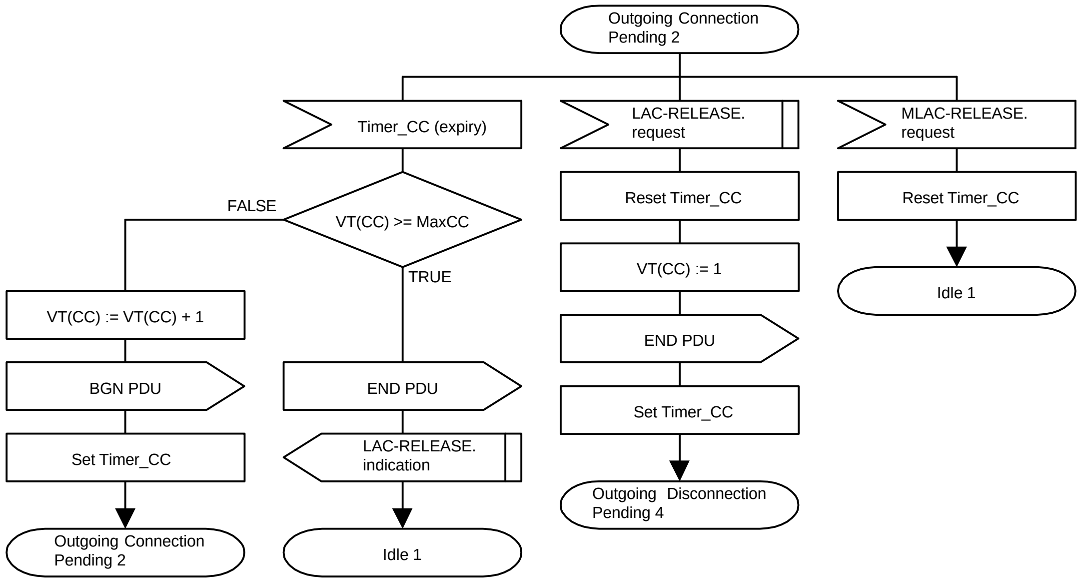

Figure 12-5

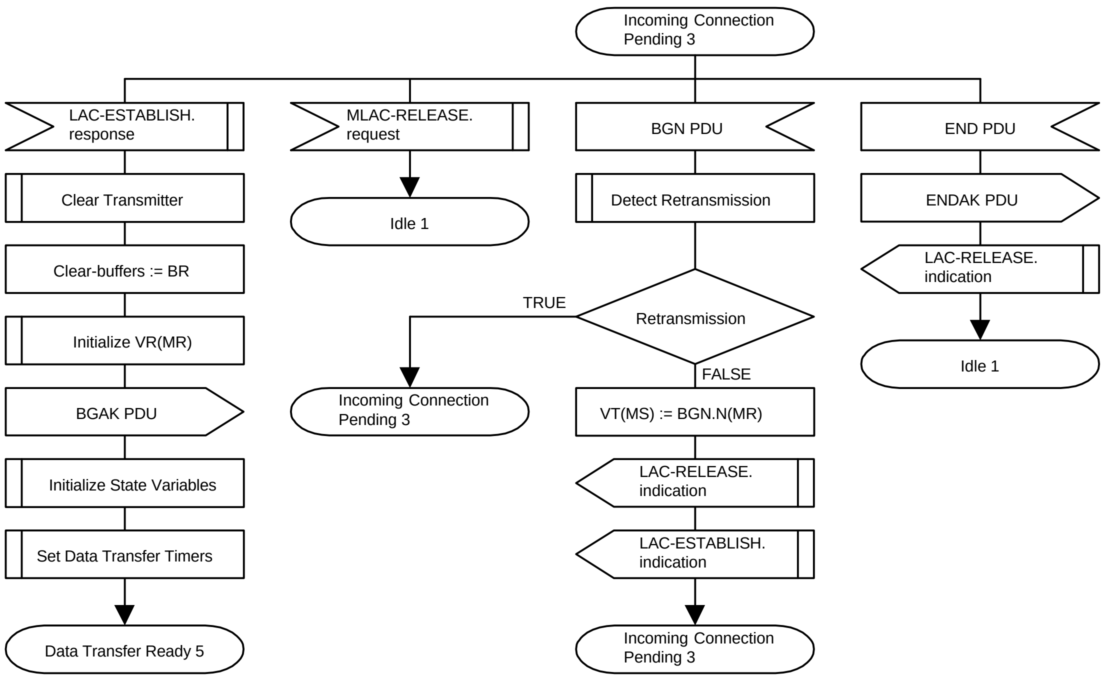

Figure 12-6

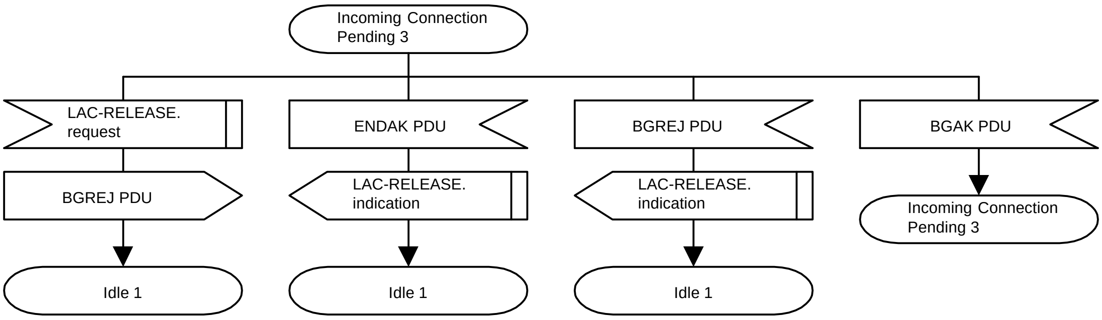

Figure 12-7

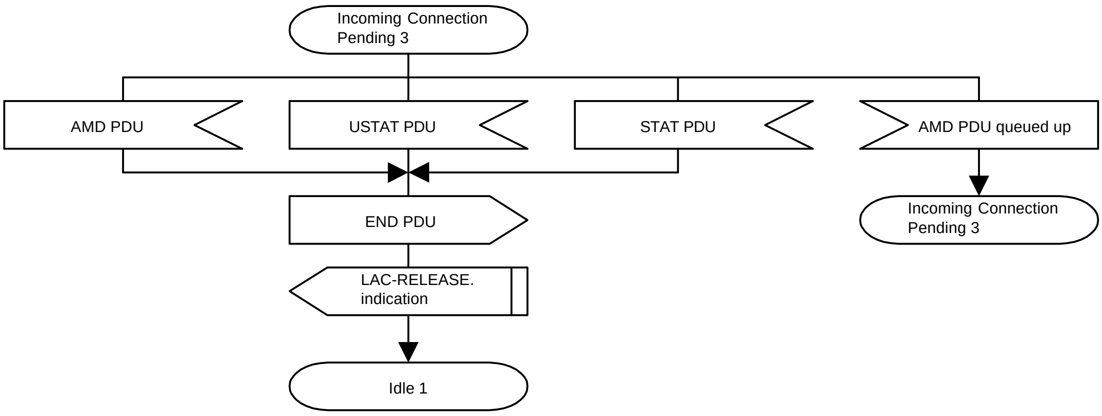

Figure 12-8

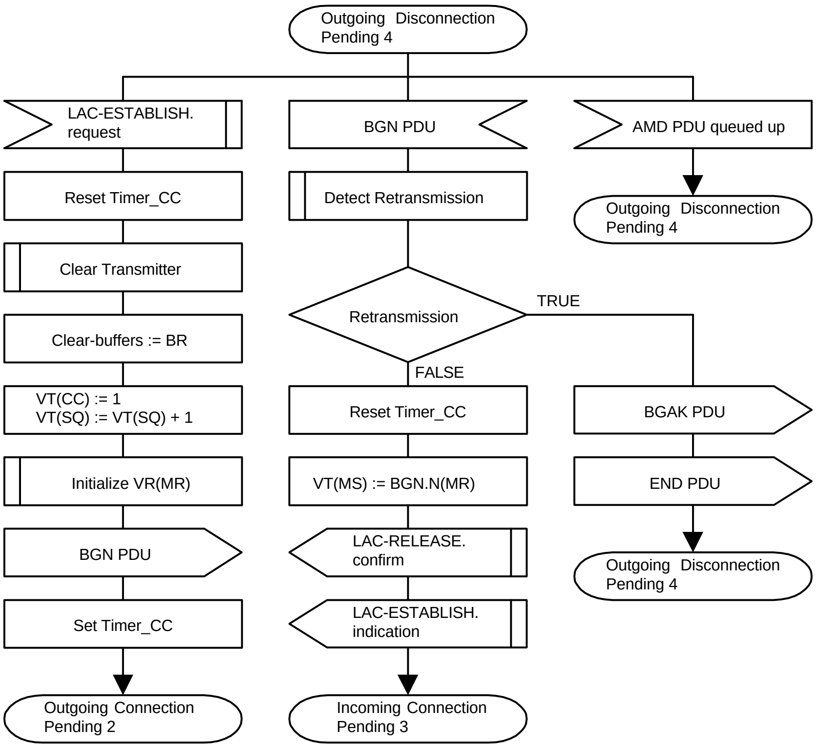

Figure 12-9

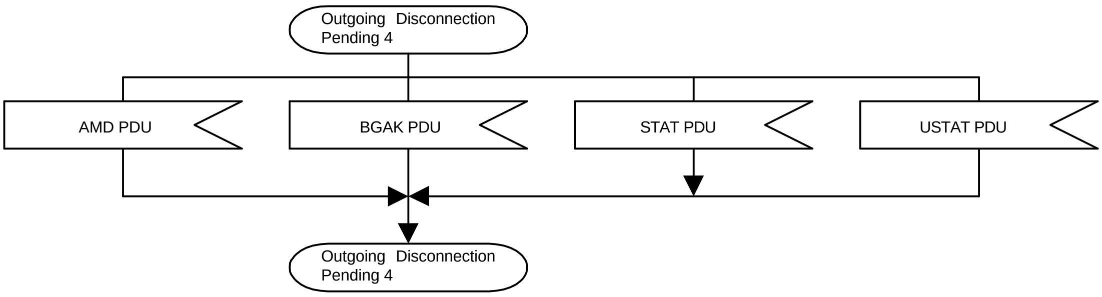

Figure 12-10

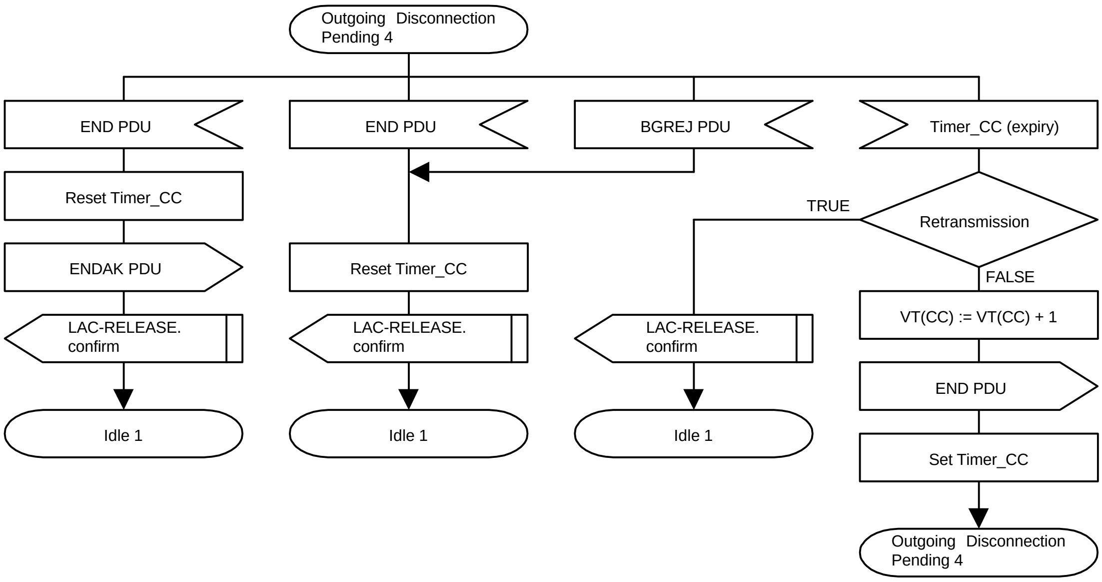

Figure 12-11

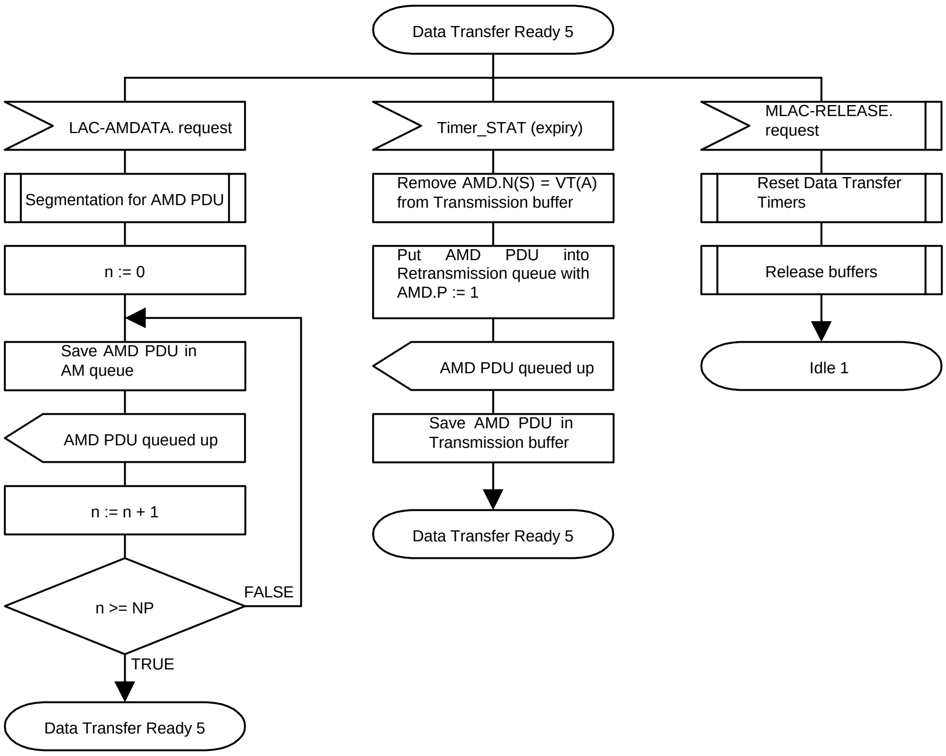

Figure 12-12

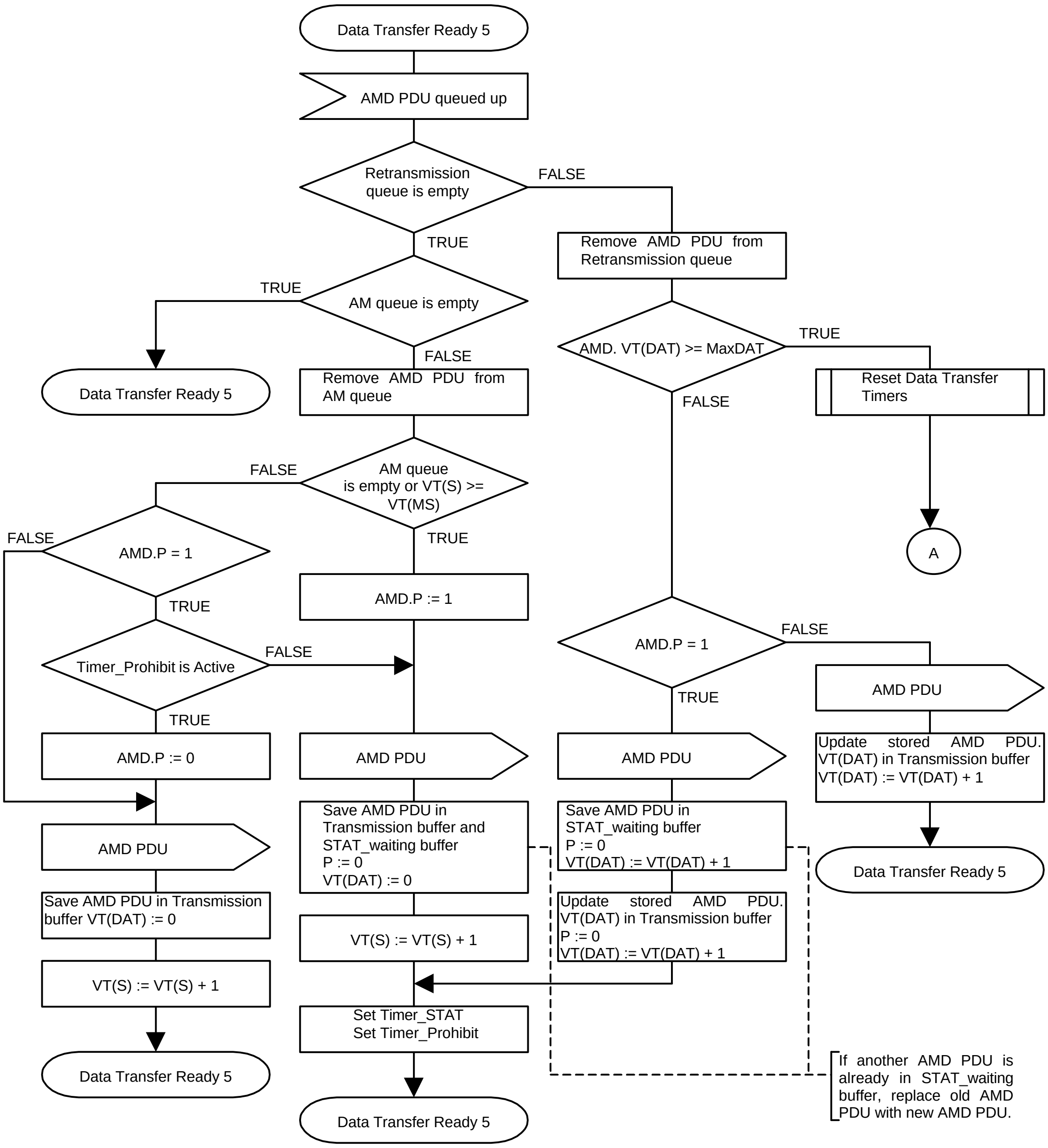
Data Transfer Ready 5
AMD PDU queued up
Retransmission queue is empty
FALSE
TRUE
Remove AMD PDU from Retransmission queue
AM queue is empty
TRUE
FALSE
AMD. VT(DAT) >= MaxDAT
TRUE
FALSE
Data Transfer Ready 5
Remove AMD PDU from AM queue
Reset Data Transfer Timers
AM queue is empty or VT(S) >= VT(MS)
FALSE
TRUE
AMD.P = 1
FALSE
TRUE
A
AMD.P := 1
Timer_Prohibit is Active
FALSE
TRUE
AMD.P = 1
FALSE
TRUE
AMD PDU
AMD.P := 0
AMD PDU
AMD PDU
Update stored AMD PDU. VT(DAT) in Transmission buffer VT(DAT) := VT(DAT) + 1
AMD PDU
Save AMD PDU in Transmission buffer and STAT_waiting buffer P := 0 VT(DAT) := 0
Save AMD PDU in STAT_waiting buffer P := 0 VT(DAT) := VT(DAT) + 1
Data Transfer Ready 5
Save AMD PDU in Transmission buffer VT(DAT) := 0
VT(S) := VT(S) + 1
Update stored AMD PDU. VT(DAT) in Transmission buffer P := 0 VT(DAT) := VT(DAT) + 1
VT(S) := VT(S) + 1
Set Timer_STAT Set Timer_Prohibit
Data Transfer Ready 5
Data Transfer Ready 5
If another AMD PDU is already in STAT_waiting buffer, replace old AMD PDU with new AMD PDU.

Figure 12-13

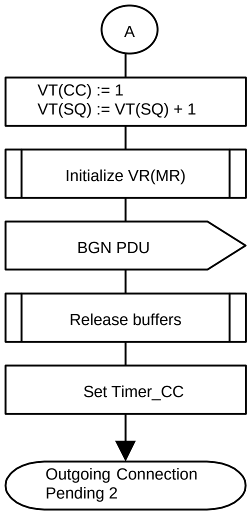
A
VT(CC) := 1
VT(SQ) := VT(SQ) + 1
Initialize VR(MR)
BGN PDU
Release buffers
Set Timer_CC
Outgoing Connection
Pending 2

Figure 12-14

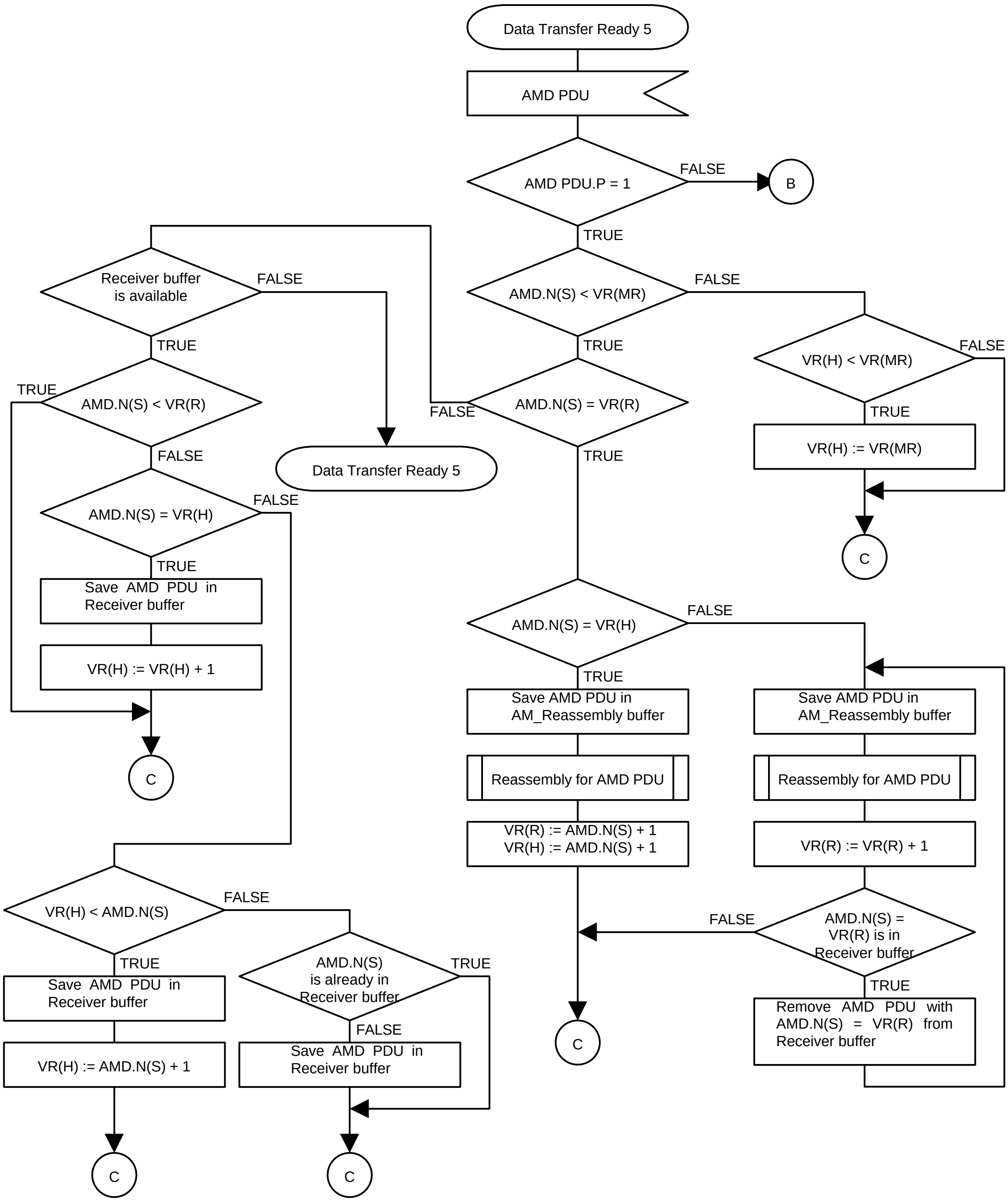
Data Transfer Ready 5
AMD PDU
AMD PDU.P = 1
FALSE
B
TRUE
AMD.N(S) < VR(MR)
FALSE
TRUE
VR(H) < VR(MR)
FALSE
TRUE
VR(H) := VR(MR)
C
Receiver buffer is available
FALSE
TRUE
AMD.N(S) = VR(R)
FALSE
TRUE
AMD.N(S) < VR(R)
TRUE
FALSE
Data Transfer Ready 5
AMD.N(S) = VR(H)
FALSE
TRUE
Save AMD PDU in Receiver buffer
VR(H) := VR(H) + 1
C
AMD.N(S) = VR(H)
FALSE
TRUE
Save AMD PDU in AM_Reassembly buffer
Reassembly for AMD PDU
VR(R) := AMD.N(S) + 1
VR(H) := AMD.N(S) + 1
C
Save AMD PDU in AM_Reassembly buffer
Reassembly for AMD PDU
VR(R) := VR(R) + 1
AMD.N(S) = VR(R) is in Receiver buffer
FALSE
TRUE
Remove AMD PDU with AMD.N(S) = VR(R) from Receiver buffer
VR(H) < AMD.N(S)
FALSE
TRUE
Save AMD PDU in Receiver buffer
VR(H) := AMD.N(S) + 1
C
AMD.N(S) is already in Receiver buffer
TRUE
FALSE
Save AMD PDU in Receiver buffer
C

Figure 12-15

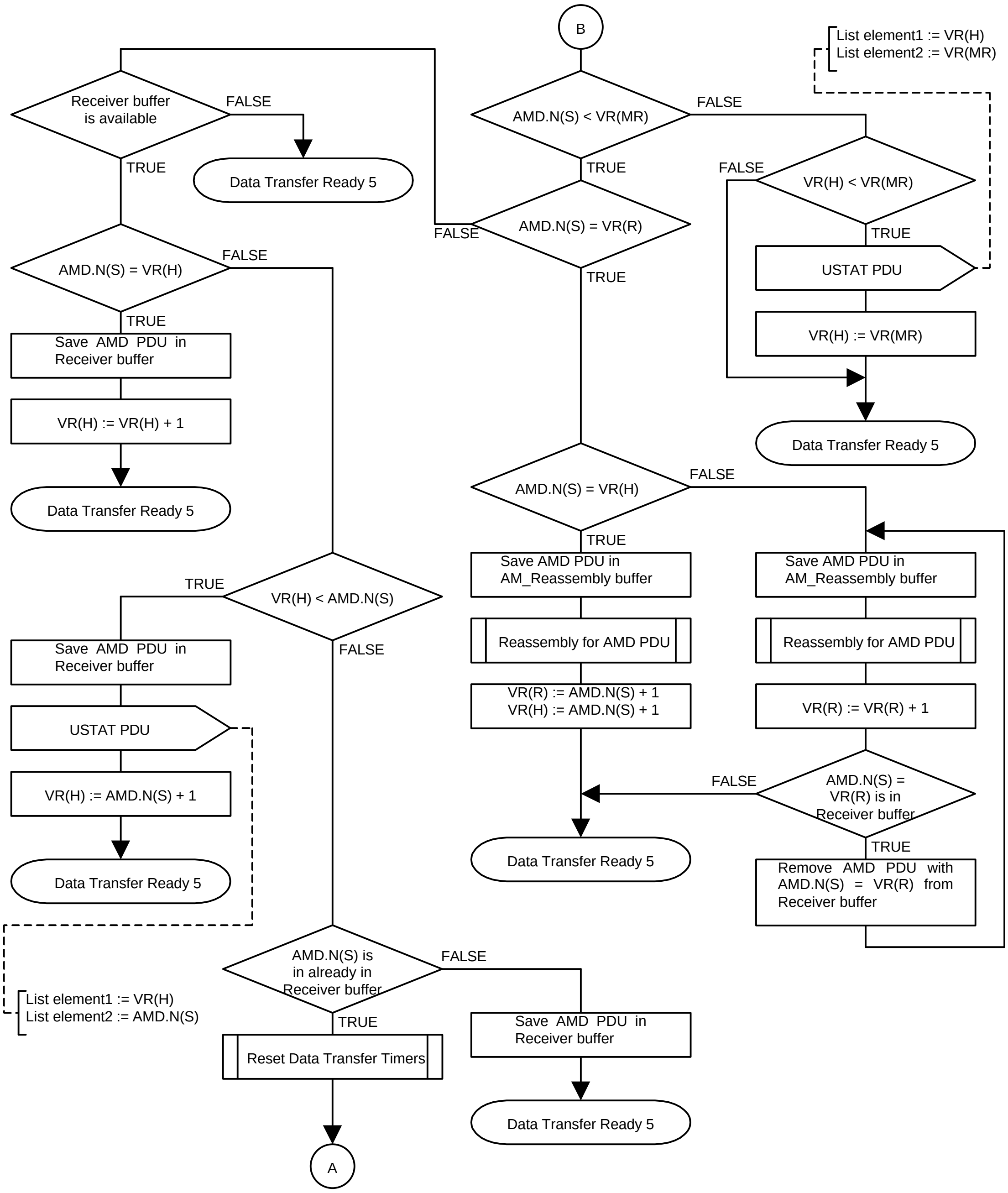
B
List element1 := VR(H)
List element2 := VR(MR)
Receiver buffer is available
FALSE
TRUE
Data Transfer Ready 5
AMD.N(S) < VR(MR)
FALSE
TRUE
VR(H) < VR(MR)
FALSE
TRUE
AMD.N(S) = VR(R)
FALSE
TRUE
USTAT PDU
VR(H) := VR(MR)
AMD.N(S) = VR(H)
FALSE
TRUE
Save AMD PDU in Receiver buffer
VR(H) := VR(H) + 1
Data Transfer Ready 5
Data Transfer Ready 5
AMD.N(S) = VR(H)
FALSE
TRUE
Save AMD PDU in AM_Reassembly buffer
Reassembly for AMD PDU
VR(R) := AMD.N(S) + 1
VR(H) := AMD.N(S) + 1
Save AMD PDU in AM_Reassembly buffer
Reassembly for AMD PDU
VR(R) := VR(R) + 1
AMD.N(S) = VR(R) is in Receiver buffer
FALSE
TRUE
Remove AMD PDU with AMD.N(S) = VR(R) from Receiver buffer
Data Transfer Ready 5
VR(H) < AMD.N(S)
TRUE
FALSE
Save AMD PDU in Receiver buffer
USTAT PDU
VR(H) := AMD.N(S) + 1
Data Transfer Ready 5
List element1 := VR(H)
List element2 := AMD.N(S)
AMD.N(S) is in already in Receiver buffer
FALSE
TRUE
Reset Data Transfer Timers
Save AMD PDU in Receiver buffer
Data Transfer Ready 5
A

Figure 12-16

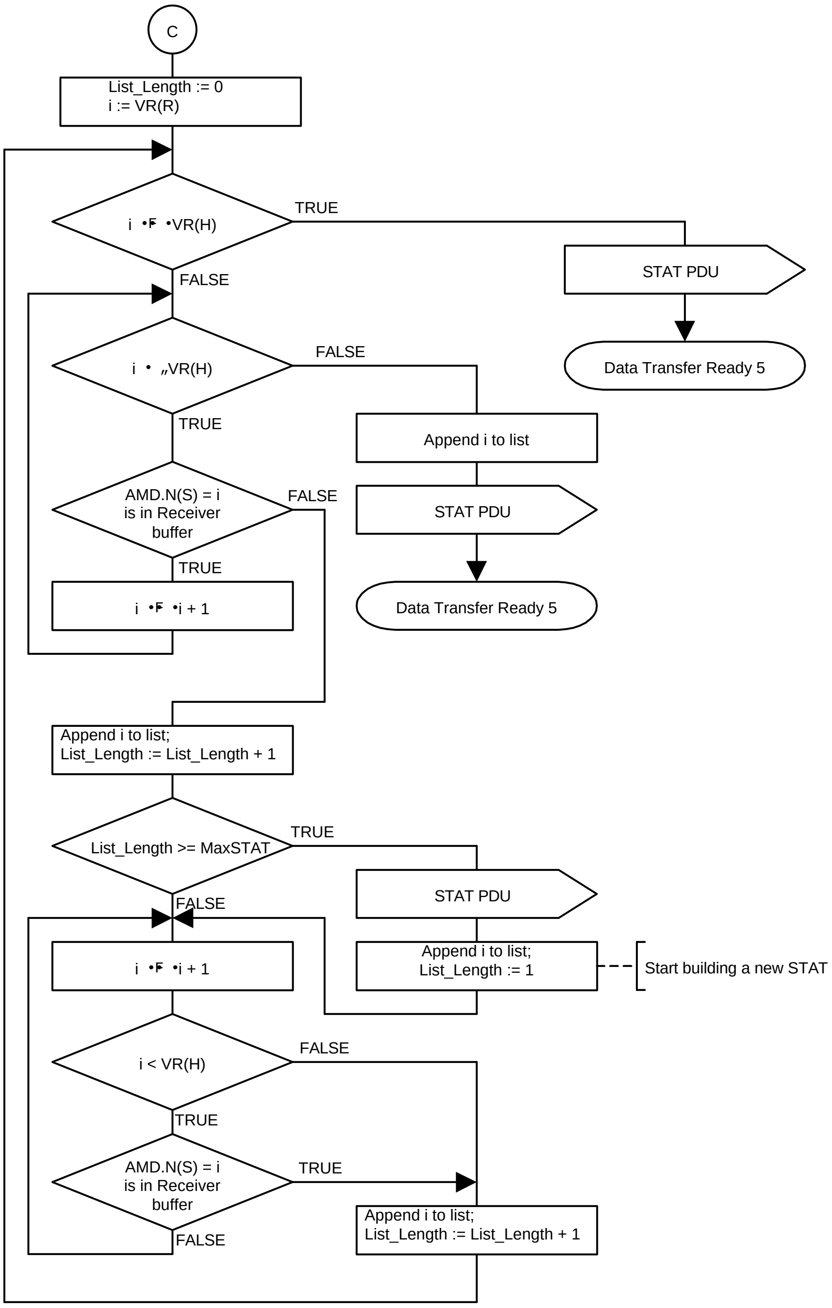
C
List_Length := 0
i := VR(R)
i •F •VR(H)
TRUE
FALSE
STAT PDU
Data Transfer Ready 5
i • „VR(H)
FALSE
TRUE
Append i to list
AMD.N(S) = i
is in Receiver
buffer
FALSE
STAT PDU
TRUE
i •F •i + 1
Data Transfer Ready 5
Append i to list;
List_Length := List_Length + 1
List_Length >= MaxSTAT
TRUE
FALSE
STAT PDU
i •F •i + 1
Append i to list;
List_Length := 1
Start building a new STAT
i < VR(H)
FALSE
TRUE
AMD.N(S) = i
is in Receiver
buffer
TRUE
FALSE
Append i to list;
List_Length := List_Length + 1

Figure 12-17

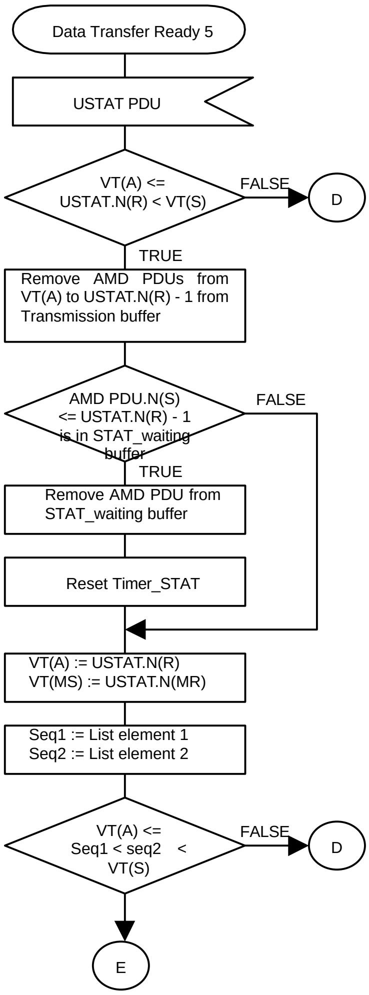
Data Transfer Ready 5
USTAT PDU
VT(A) <= USTAT.N(R) < VT(S)
FALSE
D
TRUE
Remove AMD PDUs from VT(A) to USTAT.N(R) - 1 from Transmission buffer
AMD PDU.N(S) <= USTAT.N(R) - 1 is in STAT_waiting buffer
FALSE
TRUE
Remove AMD PDU from STAT_waiting buffer
Reset Timer_STAT
VT(A) := USTAT.N(R)
VT(MS) := USTAT.N(MR)
Seq1 := List element 1
Seq2 := List element 2
VT(A) <= Seq1 < seq2 < VT(S)
FALSE
D
E

Figure 12-18

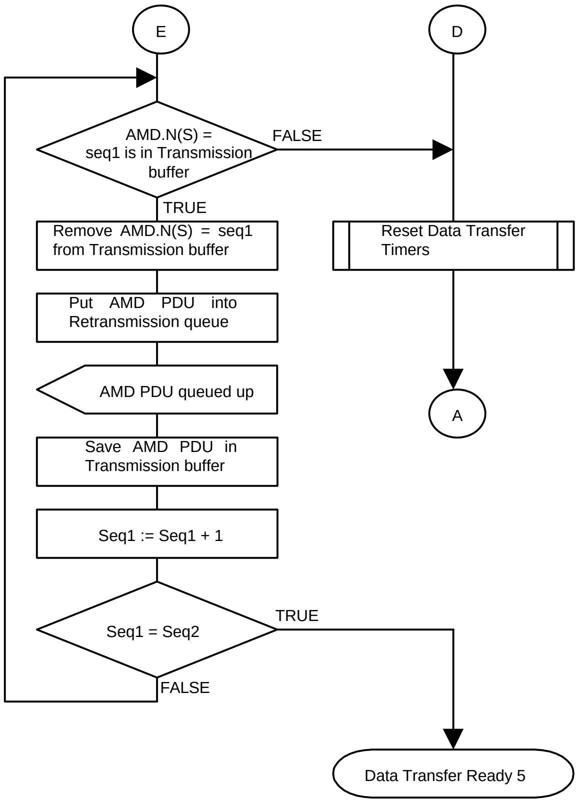
E
D
AMD.N(S) =
seq1 is in Transmission
buffer
FALSE
TRUE
Remove AMD.N(S) = seq1
from Transmission buffer
Reset Data Transfer
Timers
Put AMD PDU into
Retransmission queue
AMD PDU queued up
A
Save AMD PDU in
Transmission buffer
Seq1 := Seq1 + 1
Seq1 = Seq2
TRUE
FALSE
Data Transfer Ready 5

Figure 12-19

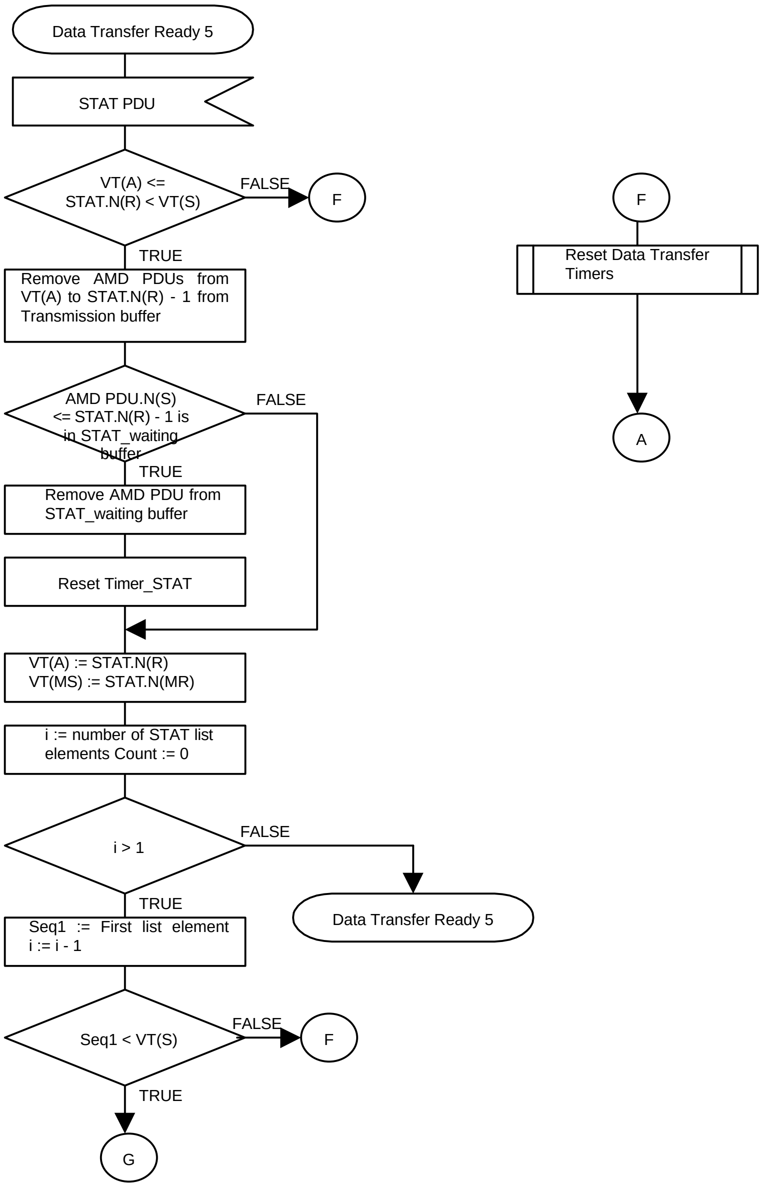
Data Transfer Ready 5
STAT PDU
VT(A) <= STAT.N(R) < VT(S)
FALSE
F
TRUE
Remove AMD PDUs from VT(A) to STAT.N(R) - 1 from Transmission buffer
AMD PDU.N(S) <= STAT.N(R) - 1 is in STAT_waiting buffer
FALSE
TRUE
Remove AMD PDU from STAT_waiting buffer
Reset Timer_STAT
VT(A) := STAT.N(R)
VT(MS) := STAT.N(MR)
i := number of STAT list elements Count := 0
i > 1
FALSE
TRUE
Data Transfer Ready 5
Seq1 := First list element
i := i - 1
Seq1 < VT(S)
FALSE
F
TRUE
G
F
Reset Data Transfer Timers
A

Figure 12-20

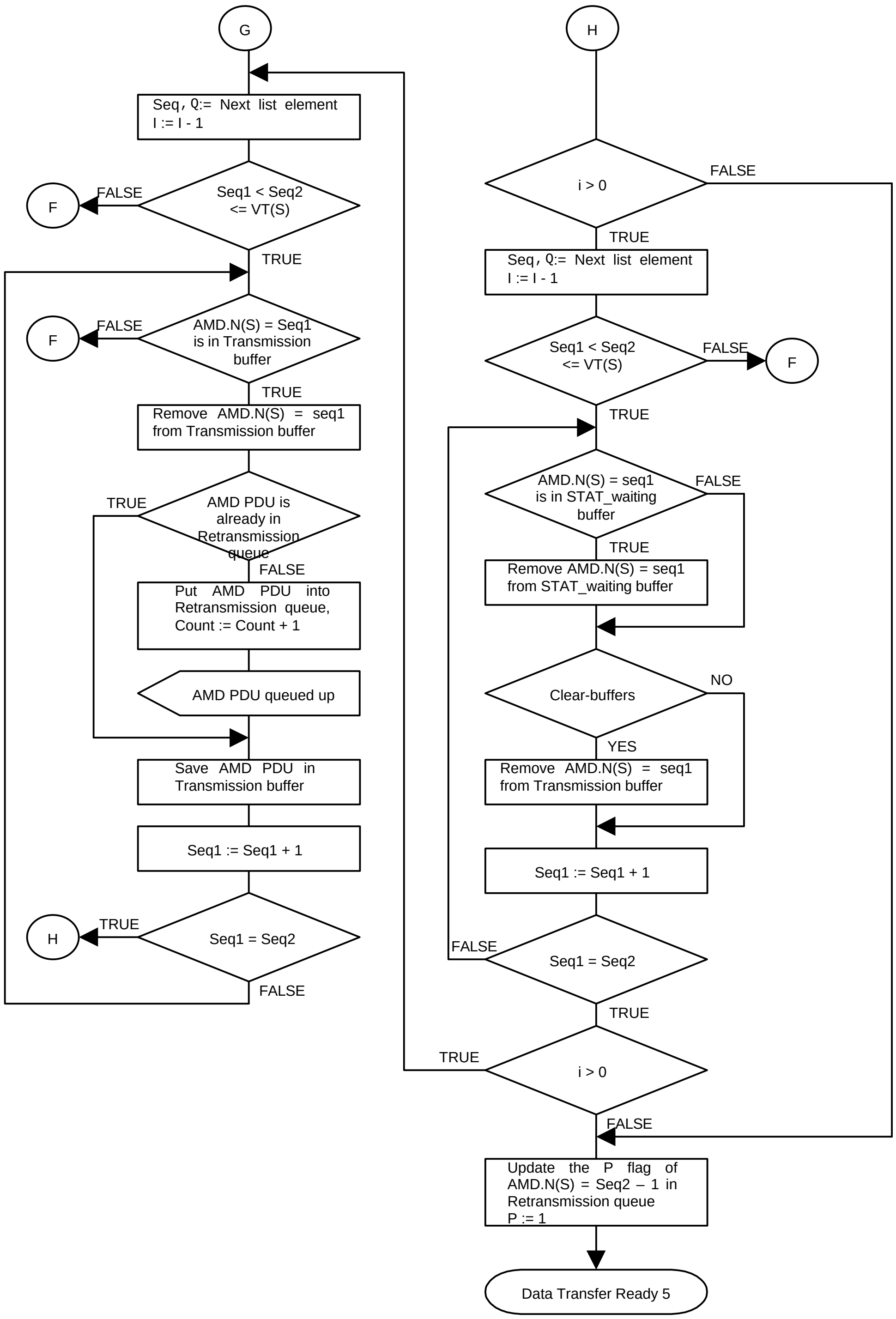
G
H
Seq, Q:= Next list element
I := I - 1
Seq1 < Seq2
<= VT(S)
FALSE
F
TRUE
AMD.N(S) = Seq1
is in Transmission
buffer
Remove AMD.N(S) = seq1
from Transmission buffer
AMD PDU is
already in
Retransmission
queue
Put AMD PDU into
Retransmission queue,
Count := Count + 1
AMD PDU queued up
Save AMD PDU in
Transmission buffer
Seq1 := Seq1 + 1
Seq1 = Seq2
i > 0
AMD.N(S) = seq1
is in STAT_waiting
buffer
Remove AMD.N(S) = seq1
from STAT_waiting buffer
Clear-buffers
NO
YES
Update the P flag of
AMD.N(S) = Seq2 – 1 in
Retransmission queue
P := 1
Data Transfer Ready 5

Figure 12-21

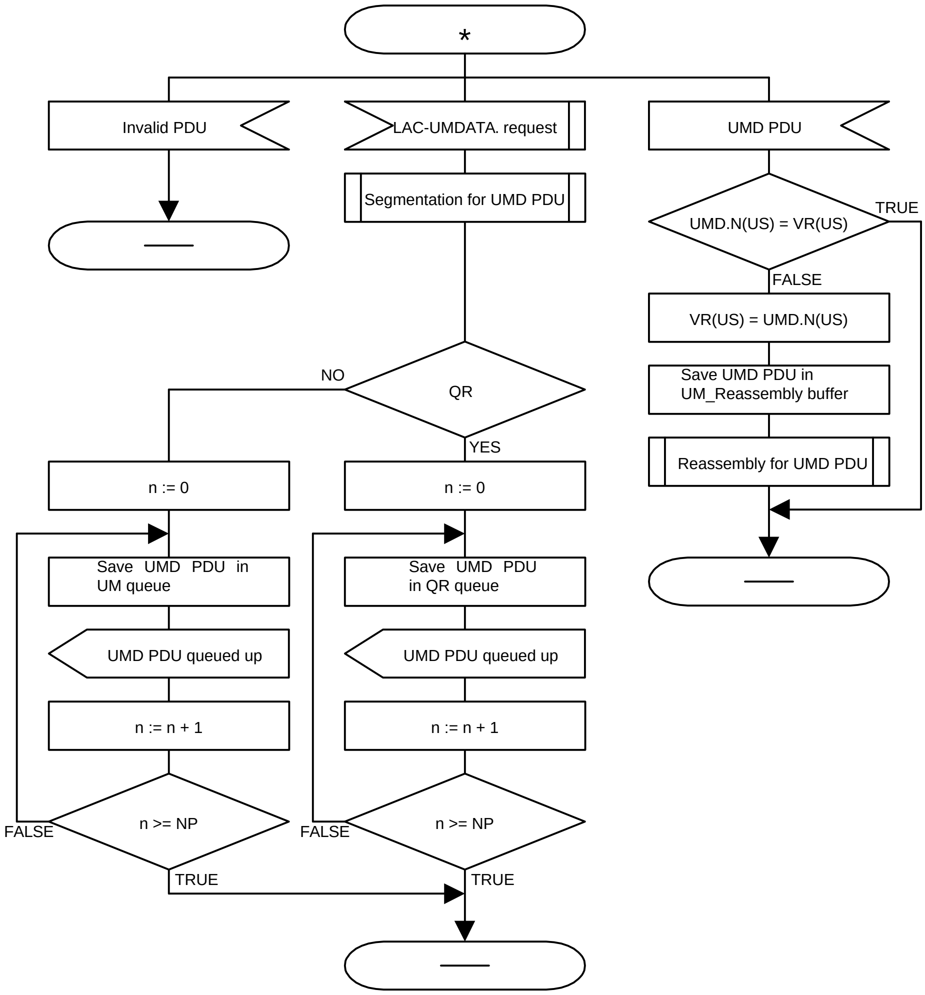
*
Invalid PDU
LAC-UMDATA. request
UMD PDU
Segmentation for UMD PDU
UMD.N(US) = VR(US)
TRUE
FALSE
VR(US) = UMD.N(US)
Save UMD PDU in UM_Reassembly buffer
Reassembly for UMD PDU
NO
QR
YES
n := 0
n := 0
Save UMD PDU in UM queue
Save UMD PDU in QR queue
UMD PDU queued up
UMD PDU queued up
n := n + 1
n := n + 1
FALSE
n >= NP
FALSE
n >= NP
TRUE
TRUE

Figure 12-22

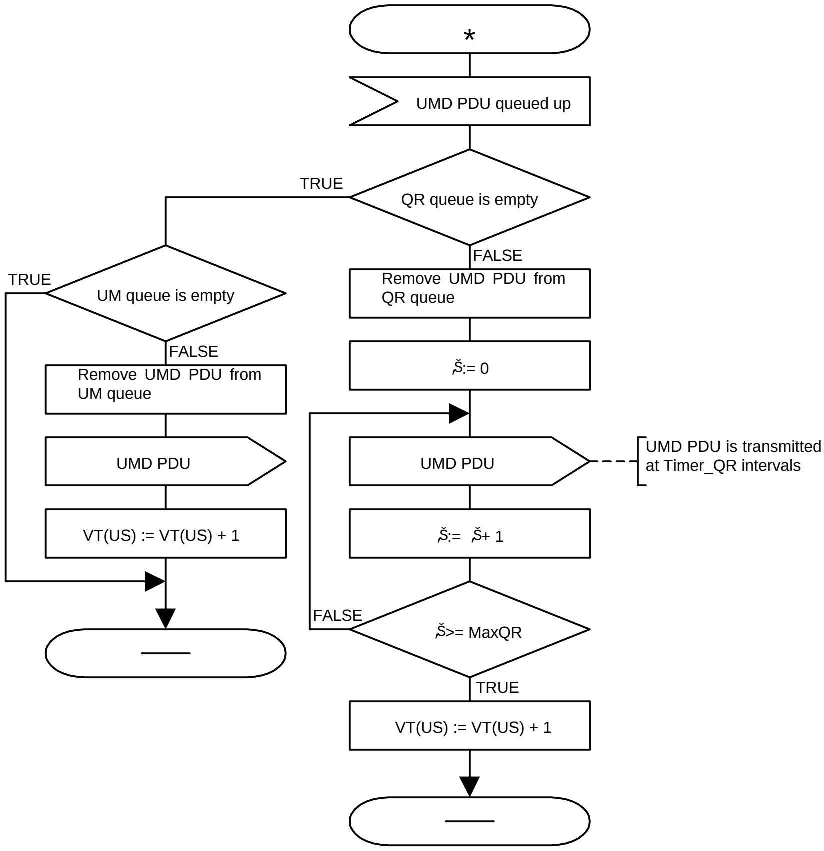

Figure 12-23

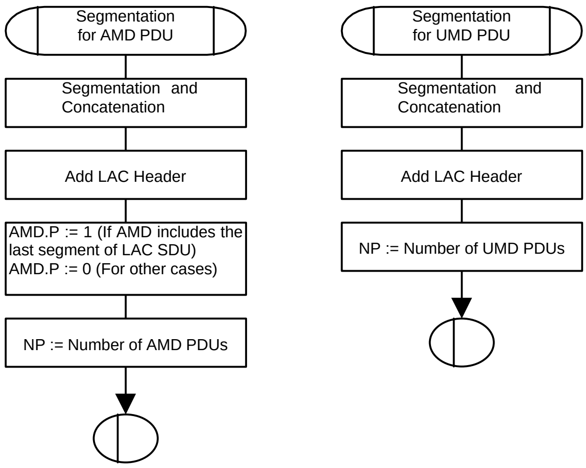

Figure 12-24

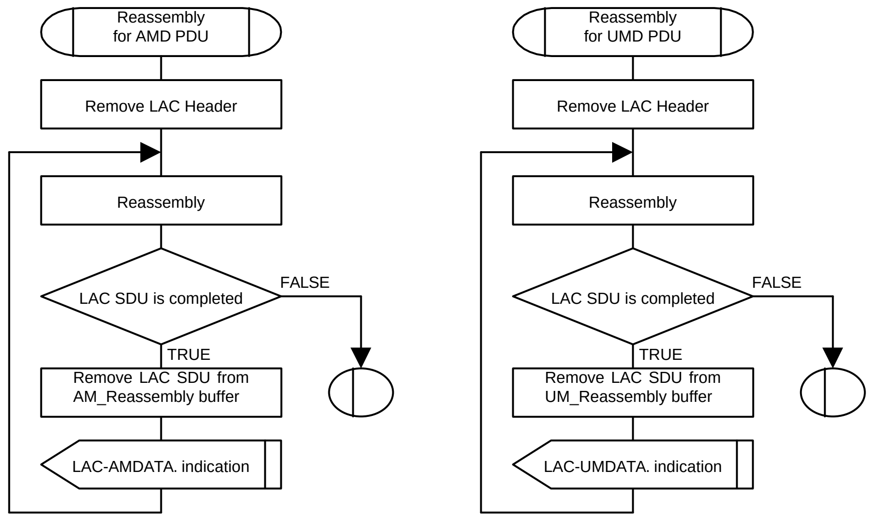

Figure 12-25

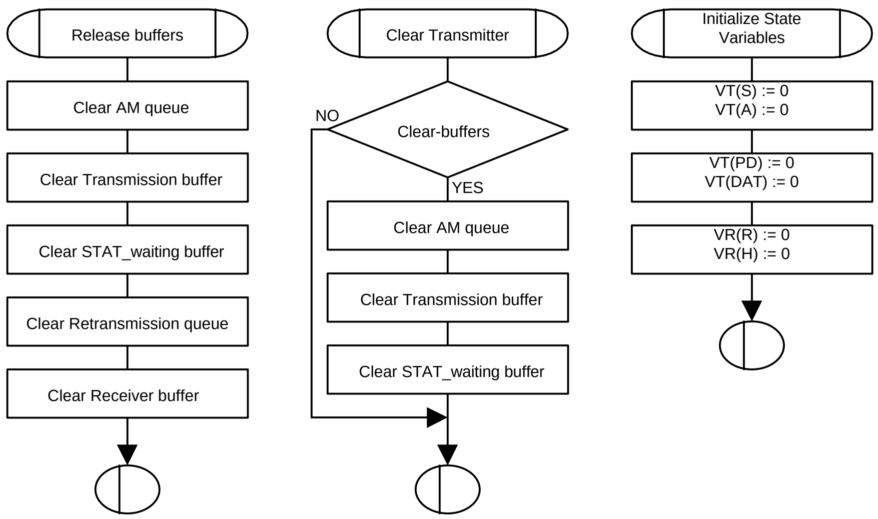

Figure 12-26

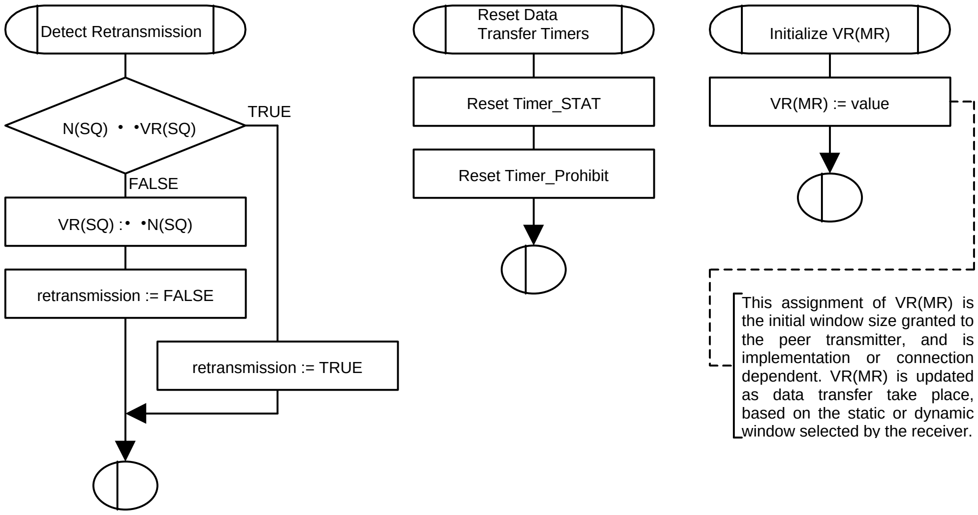
Detect Retransmission
N(SQ) • •VR(SQ)
TRUE
FALSE
VR(SQ) :• •N(SQ)
retransmission := FALSE
retransmission := TRUE
Reset Data
Transfer Timers
Reset Timer_STAT
Reset Timer_Prohibit
Initialize VR(MR)
VR(MR) := value
This assignment of VR(MR) is the initial window size granted to the peer transmitter, and is implementation or connection dependent. VR(MR) is updated as data transfer take place, based on the static or dynamic window selected by the receiver.

Figure 12-27

Appendix

1. Recommended values

1.1 PDU length

The length of the data field in AMD / UMD PDUs is k (>=0) octets.

4

1.3 MaxDAT

[FFS]

1.4 MaxQR

[FFS]

1.5 MaxSTATUS

This parameter should be an odd integer greater than or equal to 3.

1.6 Timer_STATUS

[FFS]

1.7 Timer_Prohibit

[FFS]

1.9 Timer_QR

[FFS]

13. Annex A 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 */
```

14. History

Document history		
Date	**Version**	**Comment**
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: -Sec.8:the CRLC_STATUS primitive has been included -Sec. 9.2: general text rewording with the inclusion of two new super-field types (WINDOW SIZE, RLIST). The extended header description has been included as well. -Secs.9.4, 9.5, 9.6: part of the text in has been updated, and a number of new state variables, timers and parameters, have been included. -Sec.9.7.1: text and figures have been updates according the new names adopted for the timers in Sec. 9.5. -Sec.9.7.3: some editoral corrections concerning the replacement of PDU with PU have been included. -Sec. 9.7.4: a more appropriate description of the header compression concept, has been included. -Sec. 10: text on RLC error handling has been included. -Sec 11: part of the text coming from Sec. 9.8 relative to the toolbox functions has been added inside, plus minor editorial changes. Sec. 8 has been removed. - the Quick Repeat function and related parameters have been removed.
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 S2.22 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 6.0c/95.