OsmoPCU - Feature #1527

Set correct values of the puncturing scheme and the retransmission flag

02/22/2016 09:00 PM - zecke

Status:	New	**Start date:**	02/22/2016
Priority:	Low	**Due date:**	
Assignee:	pespin	**% Done:**	0%
Category:			
Target version:			
Spec Reference:			

Description

The puncturing scheme and the RBS flag must be set according to certain rules in TS 44.060 (9.1.3.2, 9.3.2.1, ...). Currently they puncturing scheme 1 is always used and RBS is always 0. It must be updated and remembered per BSN. The RBS flag must be set, if at least one BSN data block in the RLC block is being retransmitted.

History

#1 - 03/01/2018 11:16 PM - laforge

- Assignee deleted (msuraev)

#2 - 03/03/2018 09:45 PM - laforge

- Assignee set to sysmocom

#3 - 02/06/2021 01:28 PM - laforge

- Assignee changed from sysmocom to pespin

#4 - 02/06/2021 01:29 PM - laforge

probably implemented in

```
commit 7952282b78867e53ab8fd9cf819d1f9fdece58ba
Author: Aravind Sirsikar <arvind.sirsikar@radisys.com>
Date:   Wed Mar 23 18:29:47 2016 +0530

    Support puncturing scheme selection for EGPRS DL

    Adds support to find the puncturing scheme for retransmission
    with MCS change, retransmission with no MCS change, transmission
    case. Puncturing scheme selection for retransmission case with
    MCS change is aligned with TS 44.060 9.3.2.1. Puncturing scheme
    selection for retransmission without MCS change, fresh transmission
    is aligned with TS 44.060 10.4.8a.3.1, 10.4.8a.2.1, 10.4.8a.1.1
```

#5 - 02/08/2021 07:10 PM - pespin

laforge I think the commit you referenced there is non related.

The commit you shared is about puncturing for EGPRS DL **DATA** blocks.

This ticket is about updating DL **CONTROL** blocks (CS-1) where they are somehow sent together in the same RLCMAC block (I'd need to check when exactly how's that actually possible?)

See TS 44.060 "10.4.12a Reduced Block Sequence Number (RBSN) bit".
Also "9.1.1a Control send state variable V(CS)":

```
The network RLC endpoint transmitter shall have one instance of an associated control send state variable V(CS
) for each parallel control transaction identified by the RTI field of the RLC/MAC control block header. V(CS)
 denotes the sequence number of the next in-sequence RLC/MAC control block to be transmitted for the control t
ransaction.
V(CS) can take on the values 0 or 1 when RLC/MAC control message segmentation into two RLC/MAC control blocks
is used, and the values 0 to 8 when extended RLC/MAC control message segmentation is used (see sub-clause 9.1.
```

```
12a). V(CS) shall be set to the value 0 prior to the transmission of each RLC/MAC control block that contains
the first octet of an RLC/MAC control message of the control transaction and the value of V(CS) shall be set t
o 1 after the transmission of the RLC/MAC control block with RBSN = 0. The value of V(CS) shall then be increm
ented by 1, when extended RLC/MAC control message segmentation is used, after the transmission of the next in-
sequence RLC/MAC control block and so on.
```

I don't think we are doing any kind of "RLC/MAC control message segmentation" at all and I'm not sure yet how that works, so I guess we are fine using RBSN=0 always....

I think we are indeed never setting the RBSN bit to 1. I think we never set it at all (so probably by default is 0 in memory).
Only place where the bit is written afaict is in encode_gsm_rlcmac_downlink(), but the ->RBSN field there is never set. Same goes for ->RTI.
That function is called for instance from gprs_rlcmac_tbf::create_dl_ass() AND gprs_rlcmac_tbf::create_ul_ass() (PACCH).

So all these features seem to be optional (optional octets in DL Control block, see "Figure 10.3.1.1: Downlink RLC/MAC control block together with its MAC header") and are only used when "PAYLOAD_TYPE == PAYLOAD_TYPE_CTRL_OPT_OCTET" according to our code, which AFAICT we never really use to transmit.

So, in conclusion, it's optional features we are currently not implemented. AFAIU The benefit of implementing this would be being able to send several CTRL messages segmented in CTRL blocks, which means could improve the amount of DL CTRL throughput we can sent to the MS under some circumstances.It could be too that it's possible to include those CS-1/MCS-0 CTRL messages inside an EGPRS data rlcmac/block, hence improving the throughput by not wasting an entire block to send CTRL data?

Last, but not least, if we ever want to implement this, it seems we also need take into consideration what we receive in PKT CTRL ACK with regards to what was acked when we use RBSN!=0, see "Table 11.2.2.2: PACKET CONTROL ACKNOWLEDGEMENT"