OsmoBSC - Bug #3708

osmo-bsc: Wrong handling of PDCH DEACT NACK

11/26/2018 05:34 PM - pespin

Status:	Feedback	**Start date:**	11/26/2018
Priority:	Normal	**Due date:**	
Assignee:	pespin	**% Done:**	70%
Category:	A-bis RSL		
Target version:	Dynamic TCH/H TCH/F PDCH		
Spec Reference:			

Description

Due to bug fixed in osmo-bts-trx 133a3d96dc07ebda4dfc7899dab9c0d0c80c9fea, osmo-bts-trx was sending a PDCH DEACT NACK followed by an ACK (in osmo-gsm-tester test dynts\:trx-b200+mod-bts0-dynts67-ipa+cfg-codec-fr-any).

Let's see how osmo-bsc behaves in that case.

After gprs, a phone call between the 2 MS is places, so the dynts which was in PDCH is selected and needs to be moved to TCH/F:

```
20181123034306831 DAS <0012> fsm.c:299 assignment(conn5_IMSI901700000023847)[0x612000001720]{WAIT_
LCHAN_ACTIVE}: Allocated
20181123034306832 DAS <0012> fsm.c:329 assignment(conn5_IMSI901700000023847)[0x612000001720]{WAIT_
LCHAN_ACTIVE}: is child of SUBSCR_CONN(conn5_IMSI901700000023847)[0x612000001ba0]
20181123034306832 DRLL <0000> lchan_select.c:159 (bts=0) lchan_select_by_type(TCH_F)
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F: (bts=0,trx=0,ts=0,pchan=C
CCH+SDCCH4,state=IN_USE) is != TCH/F
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F: (bts=0,trx=0,ts=1,pchan=S
DCCH8,state=UNUSED) is != TCH/F
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F: (bts=0,trx=0,ts=2,pchan=S
DCCH8,state=UNUSED) is != TCH/F
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F: (bts=0,trx=0,ts=3,pchan=S
DCCH8,state=UNUSED) is != TCH/F
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F: (bts=0,trx=0,ts=4,pchan=S
DCCH8,state=UNUSED) is != TCH/F
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F: (bts=0,trx=0,ts=5,pchan=S
DCCH8,state=UNUSED) is != TCH/F
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F: (bts=0,trx=0,ts=7,pchan_o
n_init=TCH/F_PDCH,pchan=PDCH,state=PDCH) is != TCH/F
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/H: (bts=0,trx=0,ts=0,pchan=C
CCH+SDCCH4,state=IN_USE) is != TCH/H
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/H: (bts=0,trx=0,ts=1,pchan=S
DCCH8,state=UNUSED) is != TCH/H
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/H: (bts=0,trx=0,ts=2,pchan=S
DCCH8,state=UNUSED) is != TCH/H
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/H: (bts=0,trx=0,ts=3,pchan=S
DCCH8,state=UNUSED) is != TCH/H
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/H: (bts=0,trx=0,ts=4,pchan=S
DCCH8,state=UNUSED) is != TCH/H
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/H: (bts=0,trx=0,ts=5,pchan=S
DCCH8,state=UNUSED) is != TCH/H
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/H: (bts=0,trx=0,ts=7,pchan_o
n_init=TCH/F_PDCH,pchan=PDCH,state=PDCH) is != TCH/H
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F_PDCH as TCH/F: (bts=0,trx=
0,ts=0,pchan=CCCH+SDCCH4,state=IN_USE) is != TCH/F_PDCH
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F_PDCH as TCH/F: (bts=0,trx=
0,ts=1,pchan=SDCCH8,state=UNUSED) is != TCH/F_PDCH
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F_PDCH as TCH/F: (bts=0,trx=
0,ts=2,pchan=SDCCH8,state=UNUSED) is != TCH/F_PDCH
20181123034306832 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F_PDCH as TCH/F: (bts=0,trx=
0,ts=3,pchan=SDCCH8,state=UNUSED) is != TCH/F_PDCH
20181123034306833 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F_PDCH as TCH/F: (bts=0,trx=
```

```
0,ts=4,pchan=SDCCH8,state=UNUSED) is != TCH/F_PDCH
20181123034306833 DRLL <0000> lchan_select.c:71 looking for lchan TCH/F_PDCH as TCH/F: (bts=0,trx=
0,ts=5,pchan=SDCCH8,state=UNUSED) is != TCH/F_PDCH
20181123034306833 DRLL <0000> lchan_select.c:86 looking for lchan TCH/F_PDCH as TCH/F: (bts=0,trx=
0,ts=7,pchan_on_init=TCH/F_PDCH,pchan=PDCH,state=PDCH) ss=0 is available after dyn PC
HAN change
```

fsm state is changed to WAIT_PDCH_DEACT and PDCH DEACT is sent to osmo-bts-trx:

```
20181123034306833 DTS <0011> lchan_fsm.c:577 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{PDCH}: Rec
eived Event TS_EV_LCHAN_REQUESTED
20181123034306833 DTS <0011> timeslot_fsm.c:347 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{PDCH}:
state_chg to WAIT_PDCH_DEACT
20181123034306833 DTS <0011> abis_rsl.c:2124 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{WAIT_PDCH_
DEACT}: (pchan_is=PDCH) Tx: ip.access dyn TS: PDCH DEACT
```

Then the (buggy in this case) NACK is received, and we move the state into BORKEN state, but the channel is not released at that time, it just keeps in that state. It should have been released at this time!

```
20181123034306840 DCHAN <0010> abis_rsl.c:1124 lchan(0-0-7-TCH_F_PDCH-0)[0x612000003520]{WAIT_TS_R
EADY}: (type=TCH_F) Rx IPAC_PDCH_DEACT_NACK
20181123034306840 DTS <0011> abis_rsl.c:1098 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{WAIT_PDCH_
DEACT}: Received Event TS_EV_PDCH_DEACT_NACK
20181123034306840 DTS <0011> timeslot_fsm.c:172 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{WAIT_PD
CH_DEACT}: (pchan_is=PDCH) Received PDCH deactivation NACK
20181123034306840 DTS <0011> timeslot_fsm.c:176 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{WAIT_PD
CH_DEACT}: state_chg to BORKEN
```

Then the PDCH DEACT ACT arrives (due to bug mentioned in osmo-bts-trx). It is correctly handled as not permitted, since we already received the NACK. Atually, this FSM maybe should already be gone or should have started at list to close the channel by this time:

```
20181123034306880 DLMI <0017> input/ipaccess.c:251 RX 2: 08 4c 01 0f
20181123034306880 DCHAN <0010> abis_rsl.c:1124 lchan(0-0-7-TCH_F_PDCH-0)[0x612000003520]{WAIT_TS_R
EADY}: (type=TCH_F) Rx IPAC_PDCH_DEACT_ACK
20181123034306880 DTS <0011> abis_rsl.c:1098 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{BORKEN}: R
eceived Event TS_EV_PDCH_DEACT_ACK
20181123034306880 DTS <0011> abis_rsl.c:1098 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{BORKEN}: E
vent TS_EV_PDCH_DEACT_ACK not permitted
```

Finally, after some time, the timer T23001 fires and the conn is released. Why did it wait until timer to release it if we knew since the NACK that it was not possible to go on?! (This is actually TS6, but actually we see same behavior for TS7, see below).

```
20181123034308762 DCHAN <0010> fsm.c:189 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{WAIT_TS_READY}:
 Timeout of T23001
20181123034308762 DCHAN <0010> lchan_fsm.c:1297 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{WAIT_TS_
READY}: (type=TCH_F) Handling failure, will then transition to state UNUSED
20181123034308762 DCHAN <0010> lchan_fsm.c:78 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{WAIT_TS_RE
ADY}: (type=TCH_F) lchan allocation failed in state WAIT_TS_READY: Timeout
20181123034308762 DCHAN <0010> lchan_fsm.c:107 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{WAIT_TS_R
EADY}: (type=TCH_F) Signalling Assignment FSM of error (lchan allocation failed in state WAIT_TS_R
```

```
EADY: Timeout)
20181123034308762 DAS <0012> lchan_fsm.c:1297 assignment(conn4_0-0-6-TCH_F_PDCHasPDCH-0)[0x6120000
021a0]{WAIT_LCHAN_ACTIVE}: Received Event ASSIGNMENT_EV_LCHAN_ERROR
20181123034308762 DAS <0012> assignment_fsm.c:625 assignment(conn4_0-0-6-TCH_F_PDCHasPDCH-0)[0x612
0000021a0]{WAIT_LCHAN_ACTIVE}: (bts=0,trx=0,ts=6,ss=0) Assignment failed in state WAIT_LCHAN_ACTIV
E, cause RADIO INTERFACE MESSAGE FAILURE: Failed to activate lchan (bts=0,trx=0,ts=6,ss=0)
20181123034308762 DAS <0012> assignment_fsm.c:625 assignment(conn4_0-0-6-TCH_F_PDCHasPDCH-0)[0x612
0000021a0]{WAIT_LCHAN_ACTIVE}: (bts=0,trx=0,ts=6,ss=0) incrementing rate counter: assignment:error
 Assigment failed for other reason.
```

Then cascade teardown happens:

```
20181123034308763 DAS <0012> assignment_fsm.c:127 assignment(conn4_0-0-6-TCH_F_PDCHasPDCH-0)[0x612
0000021a0]{WAIT_LCHAN_ACTIVE}: (bts=0,trx=0,ts=6,ss=0) Assignment failed
20181123034308763 DAS <0012> assignment_fsm.c:128 assignment(conn4_0-0-6-TCH_F_PDCHasPDCH-0)[0x612
0000021a0]{WAIT_LCHAN_ACTIVE}: Terminating (cause = OSMO_FSM_TERM_ERROR)
20181123034308763 DAS <0012> assignment_fsm.c:128 assignment(conn4_0-0-6-TCH_F_PDCHasPDCH-0)[0x612
0000021a0]{WAIT_LCHAN_ACTIVE}: Removing from parent SUBSCR_CONN(conn4)[0x612000002620]
20181123034308763 DCHAN <0010> lchan_fsm.c:1324 lchan_rtp(0-0-6-TCH_F_PDCH-0)[0x612000002020]{WAIT
_LCHAN_READY}: Received Event LCHAN_RTP_EV_ROLLBACK
20181123034308763 DCHAN <0010> lchan_rtp_fsm.c:223 lchan_rtp(0-0-6-TCH_F_PDCH-0)[0x612000002020]{W
AIT_LCHAN_READY}: Terminating (cause = OSMO_FSM_TERM_REQUEST)
20181123034308764 DCHAN <0010> lchan_rtp_fsm.c:223 lchan_rtp(0-0-6-TCH_F_PDCH-0)[0x612000002020]{W
AIT_LCHAN_READY}: Removing from parent lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:441 mgw-endpoint(conn4)[0x612000001ea0]{IN_USE}:
(rtpbridge/0@mgw) CI[0] to-BTS CI=AA0C7641: DLCX :0: notify=NULL
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:482 mgw-endpoint(conn4)[0x612000001ea0]{IN_USE}:
(rtpbridge/0@mgw) CI[0] to-BTS CI=AA0C7641: DLCX :0: Scheduling
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:485 mgw-endpoint(conn4)[0x612000001ea0]{IN_USE}:
state_chg to WAIT_MGW_RESPONSE
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:530 mgw-endpoint(conn4)[0x612000001ea0]{WAIT_MGW_
RESPONSE}: (rtpbridge/0@mgw) CI[0] to-BTS CI=AA0C7641: Sending MGCP: DLCX AA0C7641
20181123034308764 DRLL <0000> mgcp_client_fsm.c:686 MGCP_CONN(to-BTS)[0x612000001d20]{ST_READY}: R
eceived Event EV_DLCX
20181123034308764 DLMGCP <0023> mgcp_client.c:953 Queued 52 bytes for MGCP GW
20181123034308764 DRLL <0000> mgcp_client_fsm.c:347 MGCP_CONN(to-BTS)[0x612000001d20]{ST_READY}: s
tate_chg to ST_DLCX_RESP
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:625 mgw-endpoint(conn4)[0x612000001ea0]{WAIT_MGW_
RESPONSE}: (rtpbridge/0@mgw) Sent messages: 1
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:598 mgw-endpoint(conn4)[0x612000001ea0]{WAIT_MGW_
RESPONSE}: (rtpbridge/0@mgw) CI in use: 0, waiting for response: 0
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:603 mgw-endpoint(conn4)[0x612000001ea0]{WAIT_MGW_
RESPONSE}: Terminating (cause = OSMO_FSM_TERM_REGULAR)
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:603 mgw-endpoint(conn4)[0x612000001ea0]{WAIT_MGW_
RESPONSE}: Removing from parent SUBSCR_CONN(conn4)[0x612000002620]
20181123034308764 DRSL <0003> mgw_endpoint_fsm.c:603 mgw-endpoint(conn4)[0x612000001ea0]{WAIT_MGW_
RESPONSE}: Freeing instance
20181123034308764 DRSL <0003> fsm.c:381 mgw-endpoint(conn4)[0x612000001ea0]{WAIT_MGW_RESPONSE}: De
allocated
20181123034308765 DMSC <0007> mgw_endpoint_fsm.c:603 SUBSCR_CONN(conn4)[0x612000002620]{ASSIGNMENT
}: Received Event FORGET_MGW_ENDPOINT
20181123034308765 DCHAN <0010> lchan_rtp_fsm.c:742 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{WAIT_
TS_READY}: Received Event LCHAN_EV_RTP_RELEASED
20181123034308765 DCHAN <0010> lchan_rtp_fsm.c:223 lchan_rtp(0-0-6-TCH_F_PDCH-0)[0x612000002020]{W
AIT_LCHAN_READY}: Freeing instance
20181123034308765 DCHAN <0010> fsm.c:381 lchan_rtp(0-0-6-TCH_F_PDCH-0)[0x612000002020]{WAIT_LCHAN_
READY}: Deallocated
20181123034308765 DCHAN <0010> lchan_rtp_fsm.c:223 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{WAIT_
TS_READY}: Received Event LCHAN_EV_RTP_RELEASED
20181123034308765 DCHAN <0010> lchan_fsm.c:1338 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{WAIT_TS_
READY}: transition to state WAIT_RLL_RTP_RELEASED not permitted!
20181123034308765 DAS <0012> assignment_fsm.c:128 assignment(conn4_0-0-6-TCH_F_PDCHasPDCH-0)[0x612
0000021a0]{WAIT_LCHAN_ACTIVE}: Freeing instance
```

```
20181123034308765 DAS <0012> fsm.c:381 assignment(conn4_0-0-6-TCH_F_PDCHasPDCH-0)[0x6120000021a0]{
WAIT_LCHAN_ACTIVE}: Deallocated
20181123034308765 DMSC <0007> assignment_fsm.c:128 SUBSCR_CONN(conn4)[0x612000002620]{ASSIGNMENT}:
 Received Event ASSIGNMENT_END
20181123034308765 DMSC <0007> bsc_subscr_conn_fsm.c:399 SUBSCR_CONN(conn4)[0x612000002620]{ASSIGNM
ENT}: state_chg to ACTIVE
20181123034308765 DCHAN <0010> lchan_fsm.c:1297 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{WAIT_TS_
READY}: state_chg to UNUSED
20181123034308765 DCHAN <0010> lchan_fsm.c:371 lchan(0-0-6-TCH_F_PDCH-0)[0x6120000036a0]{UNUSED}:
(type=TCH_F) Clearing lchan state
20181123034308765 DTS <0011> lchan_fsm.c:404 timeslot(0-0-6-TCH_F_PDCH)[0x6120000087a0]{BORKEN}: R
eceived Event TS_EV_LCHAN_UNUSED
20181123034308766 DLMGCP <0023> mgcp_client.c:749 Tx MGCP msg to MGCP GW: 'DLCX 3 rtpbridge/0@mgw
MGCP 1.0'
20181123034308766 DLMGCP <0023> mgcp_client.c:751 Sending msg to MGCP GW size: 52
20181123034308766 DLINP <0015> stream.c:279 connected write
20181123034308766 DLINP <0015> stream.c:204 sending data
20181123034308769 DRLL <0000> mgcp_client_fsm.c:423 MGCP_CONN(to-BTS)[0x612000001d20]{ST_DLCX_RESP
}: Received Event EV_DLCX_RESP
20181123034308769 DRLL <0000> mgcp_client_fsm.c:439 MGCP_CONN(to-BTS)[0x612000001d20]{ST_DLCX_RESP
}: Terminating (cause = OSMO_FSM_TERM_REGULAR)
20181123034308769 DRLL <0000> mgcp_client_fsm.c:439 MGCP_CONN(to-BTS)[0x612000001d20]{ST_DLCX_RESP
}: Freeing instance
20181123034308769 DRLL <0000> fsm.c:381 MGCP_CONN(to-BTS)[0x612000001d20]{ST_DLCX_RESP}: Deallocat
ed
```

Similarly and even later, the other conn (other leg of the call, actually this is the one which we know failed initially, TS 7) fails due to timer expiring, while it should have been torn down a lot before:

```
20181123034311833 DCHAN <0010> fsm.c:189 lchan(0-0-7-TCH_F_PDCH-0)[0x612000003520]{WAIT_TS_READY}:
 Timeout of T23001
20181123034311833 DCHAN <0010> lchan_fsm.c:1297 lchan(0-0-7-TCH_F_PDCH-0)[0x612000003520]{WAIT_TS_
READY}: (type=TCH_F) Handling failure, will then transition to state UNUSED
20181123034311834 DCHAN <0010> lchan_fsm.c:78 lchan(0-0-7-TCH_F_PDCH-0)[0x612000003520]{WAIT_TS_RE
ADY}: (type=TCH_F) lchan allocation failed in state WAIT_TS_READY: Timeout
20181123034311834 DCHAN <0010> lchan_fsm.c:107 lchan(0-0-7-TCH_F_PDCH-0)[0x612000003520]{WAIT_TS_R
EADY}: (type=TCH_F) Signalling Assignment FSM of error (lchan allocation failed in state WAIT_TS_R
EADY: Timeout)
20181123034311834 DAS <0012> lchan_fsm.c:1297 assignment(conn5_IMSI901700000023847_0-0-7-TCH_F_PDC
HasPDCH-0)[0x612000001720]{WAIT_LCHAN_ACTIVE}: Received Event ASSIGNMENT_EV_LCHAN_ERROR
20181123034311834 DAS <0012> assignment_fsm.c:625 assignment(conn5_IMSI901700000023847_0-0-7-TCH_F
_PDCHasPDCH-0)[0x612000001720]{WAIT_LCHAN_ACTIVE}: (bts=0,trx=0,ts=7,ss=0) Assignment failed in st
ate WAIT_LCHAN_ACTIVE, cause RADIO INTERFACE MESSAGE FAILURE: Failed to activate lchan (bts=0,trx=
0,ts=7,ss=0)
20181123034311834 DAS <0012> assignment_fsm.c:625 assignment(conn5_IMSI901700000023847_0-0-7-TCH_F
_PDCHasPDCH-0)[0x612000001720]{WAIT_LCHAN_ACTIVE}: (bts=0,trx=0,ts=7,ss=0) incrementing rate count
er: assignment:error Assigment failed for other reason.
20181123034311834 DMSC <0007> osmo_bsc_sigtran.c:367 Tx MSC: BSSMAP: ASSIGNMENT FAIL
....
```

Related issues:

Related to OsmoBTS - Bug #3706: osmo-bts-trx: PDCH DEACT ACK is sent before r...	**Resolved**	**11/23/2018**
Related to OsmoMSC - Feature #3236: Rx Assignment Failure from BSC does nothing	**Resolved**	**05/04/2018**

History

#1 - 11/26/2018 06:05 PM - pespin

Extra issue: When the T23001 finally fires and BSC sends a BSSAP Assignment Failure to the MSC, the MSC doesn't answer and does nothing.

After a big bunch of seconds, the MS sends a CC Disconnect -> BTS -> BSC -> MSC which finally makes the MSC relase everything.

So two bugs actually so far:

- OsmoBSC should send BSSAP Assignment Failure immediatelly after receiving the PDCH DEACT NACK
- OsmoMSC should release everything when receiving those BSSAP Assignment Failure

#2 - 11/26/2018 06:35 PM - pespin

- Related to Bug #3706: osmo-bts-trx: PDCH DEACT ACK is sent before receiving response from TRX added

#3 - 11/26/2018 07:14 PM - pespin

Regarding OsmoMSC, it seems that part of the code is not implemented.
When BSSAP Assignment Failure is received, here's osmo-msc log:

```
20181123034308770 DMSC <0006> a_iface_bssap.c:80 Looking for A subscriber: conn_id 3
20181123034308771 DBSSAP <0010> a_iface_bssap.c:88 (subscr MSISDN:5037, conn_id 3) Found A subscriber for conn
_id 3
20181123034308771 DBSSAP <0010> a_iface_bssap.c:643 (subscr MSISDN:5037, conn_id 3) Rx BSSMAP DT1 ASSIGNMENT F
AIL
20181123034308771 DBSSAP <0010> a_iface_bssap.c:461 (subscr MSISDN:5037, conn_id 3) Rx BSSMAP ASSIGNMENT FAILU
RE message
20181123034308771 DRR <0003> osmo_msc.c:144 MSC assign failure (do nothing).
```

And if we check osmo_msc.c:144:

```
/* Receive an ASSIGNMENT FAILURE from BSC */
void msc_assign_fail(struct gsm_subscriber_connection *conn,
             uint8_t cause, uint8_t *rr_cause)
{
    LOGP(DRR, LOGL_DEBUG, "MSC assign failure (do nothing).\n");
}
```

So clearly MSC is missing implemnting this part...

#4 - 11/26/2018 07:18 PM - pespin

- Related to Feature #3236: Rx Assignment Failure from BSC does nothing added

#5 - 11/26/2018 07:19 PM - pespin

MSC missing feature seems to be tracked in #3236.

#6 - 11/27/2018 12:12 AM - neels

pespin wrote:

> but the channel is not released at that time, it just keeps in that state. It should have been released at this time!

That is not quite clear. It is not well defined how to handle a NACK on Abis. We expect a BTS to **do what we want**!
As soon as a chan activ or a PDCH act/deact is NACKed, that lchan is basically FACKed (scnr).
IOW it is safest to assume this lchan is now broken.

We might come up with some scheme to try and send some Release messages, but what would you send?
As far as the BTS is concerned, a dyn TS in PDCH mode doesn't even exist (only to the PCU it does), the lchan is not connected, and hence cannot be released.
The process to "release the lchan" is a PDCH DEACT. And since that was NACKed, well, that's a good night until we completely re-connect the BTS.

We could try to send a PDCH DEACT some more times until it might succeed, but practically, there is no correct situation where this could be useful.

In another problem with lchans with super-high latency on the Abis, rhizomatica often saw a Chan Deact ACK coming far too late.
So there we enabled a code path where, if the BORKEN state receives a Chan Deact ACK, we then assume that it was just late, and re-enable the lchan.
We could, probably should do a similar thing for PDCH DEACT? (probably also should remember the BORKEN reason.)

But a NACK? That's pretty defnititely the end of it, placing the lchan in an undefined state.

- OsmoBSC should send BSSAP Assignment Failure immediatelly after receiving the PDCH DEACT NACK

OsmoBSC might/should try another lchan instead?

#7 - 11/27/2018 03:54 PM - pespin

I first though ts_fsm_wait_pdch_deact (state WAIT_PDCH_DEACT) was missing a lchan_dispatch(lchan, LCHAN_EV_TS_ERROR); in the case a NACK was received, but actually looking closer at the code it contains following line:

```
ts_fsm_error(fi, TS_ST_BORKEN, "Received PDCH deactivation NACK");
```

which in turn calls:

```
ts_lchans_dispatch(ts, LCHAN_ST_WAIT_TS_READY, LCHAN_EV_TS_ERROR);

static void ts_lchans_dispatch(struct gsm_bts_trx_ts *ts, int lchan_state, uint32_t lchan_ev)
{
    struct gsm_lchan *lchan;

    ts_for_each_lchan(lchan, ts) {
        if (lchan_state >= 0
            && !lchan_state_is(lchan, lchan_state))
            continue;
        lchan_dispatch(lchan, lchan_ev);
    }
}
```

And looking at the log:

```
20181123034306840 DLMI <0017> input/ipaccess.c:251 RX 2: 08 4d 01 0f 1a 01 0f
20181123034306840 DCHAN <0010> abis_rsl.c:1124 lchan(0-0-7-TCH_F_PDCH-0)[0x612000003520]{WAIT_TS_READY}: (type
=TCH_F) Rx IPAC_PDCH_DEACT_NACK
20181123034306840 DTS <0011> abis_rsl.c:1098 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{WAIT_PDCH_DEACT}: Rece
ived Event TS_EV_PDCH_DEACT_NACK
20181123034306840 DTS <0011> timeslot_fsm.c:172 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{WAIT_PDCH_DEACT}: (
pchan_is=PDCH) Received PDCH deactivation NACK
20181123034306840 DTS <0011> timeslot_fsm.c:176 timeslot(0-0-7-TCH_F_PDCH)[0x612000008620]{WAIT_PDCH_DEACT}: s
tate_chg to BORKEN
20181123034306880 DLMI <0017> input/ipaccess.c:251 RX 2: 08 4c 01 0f
```

It clearly goes through ts_fsm_error and ts_lchans_dispatch, but doesn't print anything from the lchan_fsm.c, so it's not reaching it for some reason. state for the lchan is clearly LCHAN_ST_WAIT_TS_READY as can be seen in logs lines before and after this chunk. So only reason I can find is that ts_for_each_lchan(lchan, ts) doesn't contain it for whatever reason.

#8 - 11/27/2018 04:01 PM - pespin

If the lchan event was sent successfuly, we should see this being called:

```
void lchan_fsm_allstate_action(struct osmo_fsm_inst *fi, uint32_t event, void *data)
{
    switch (event) {

    case LCHAN_EV_TS_ERROR:
        lchan_fail_to(LCHAN_ST_UNUSED, "LCHAN_EV_TS_ERROR");
        return;

    default:
        return;
    }
}
```

Now that I notice, though, I see that log messages in lchan_fail_to() are marked as DEBUG, and for this FSM it's category DCHAN. But we are using "logging level set-all debug" in osmo-gsm-tester osmo-bsc.cfg, so we should see a message like "Handling failure, will then transition to stat..." in there.

#9 - 11/27/2018 04:06 PM - pespin

Furthermore, lchan_set_last_error() prints with ERROR level, so definetly something that should be seen.

#10 - 11/27/2018 05:13 PM - pespin

As far as I understand the issue lies in ts_for_each_lchan, which will bypass the lchan requesting the deact:

```
/* usage:
 * struct gsm_lchan *lchan;
 * struct gsm_bts_trx_ts *ts = get_some_timeslot();
 * ts_for_each_lchan(lchan, ts) {
 *     LOGPLCHAN(DMAIN, LOGL_DEBUG, "hello world\n");
 * }
 * Iterate only those lchans that have an FSM allocated. */
#define ts_for_each_lchan(lchan, ts) ts_as_pchan_for_each_lchan(lchan, ts, (ts)->pchan_is)

/* Same as ts_for_each_lchan() but with an explicit pchan kind (GSM_PCHAN_* constant).
 * Iterate only those lchans that have an FSM allocated. */
#define ts_as_pchan_for_each_lchan(lchan, ts, as_pchan) \
    for (lchan = (ts)->lchan; \
         ((lchan - (ts)->lchan) < ARRAY_SIZE((ts)->lchan)) \
         && lchan->fi \
```

```
        && lchan->nr < pchan_subslots(as_pchan); \
        lchan++)
```

In case we are in WAIT_PDCH_DEACT (ts_fsm_wait_pdch_deact), ts->pchan_is is GSM_PCHAN_PDCH. pchan_sublots(GSM_PCHAN_PDCH) = 0, while pchan_subslots(GSM_PCHAN_TCH_F) = 1. As a result, ts_as_pchan_for_each_lchan(lchan, GSM_PCHAN_PDCH) is a void loop, and the TCH/F lchan in ts->lchan[0] (lchan->nr==0) cannot be accessed. because lchan->nr < 0 is false.

#11 - 11/27/2018 08:06 PM - pespin

- Status changed from New to In Progress

- % Done changed from 0 to 70

Pushed related commits to gerrit as WIP. I still need to do some testing with them, probably by temporarily reverting osmo-bts 133a3d96dc07ebda4dfc7899dab9c0d0c80c9fea.

```
remote: New Changes:
remote:   https://gerrit.osmocom.org/#/c/osmo-bsc/+/11955 cosmetic: bsc: timeslot_fsm: Clean uneeded scope bra
ckets
remote:   https://gerrit.osmocom.org/#/c/osmo-bsc/+/11956 bsc: trx_count_free_ts: Fix counting of GSM_PCHAN_PD
CH
remote:   https://gerrit.osmocom.org/#/c/osmo-bsc/+/11957 Fix ts_for_each_lchan iteration for dynamic TS in pr
ocess of being PDCH ...
remote:   https://gerrit.osmocom.org/#/c/osmo-bsc/+/11958 WIP: bsc: timeslot_fsm: Set TS state as unused inste
ad of broken on PDCH ...
remote:
remote:
remote: Updated Changes:
remote:   https://gerrit.osmocom.org/#/c/osmo-bsc/+/11954 bsc: timeslot_fsm: ts_is_pchan_switching: Set correc
t target pchan for IPA dynts
```

#12 - 11/29/2018 06:07 PM - pespin

- Status changed from In Progress to Feedback

#13 - 12/05/2018 04:19 PM - pespin

Patches merged, waiting until next osmo-gsm-tester complete run to see how it behaves before closing the issue.

Files

merge.pcap	1.46 MB	11/26/2018	pespin