OsmoBTS - Bug #2978

OsmoBTS rxlev/rxqual SUB computation completely broken [AMR DTX]

<table>
<thead>
<tr>
<th>Status:</th>
<th>Resolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>High</td>
</tr>
<tr>
<td>Assignee:</td>
<td>fixeria</td>
</tr>
<tr>
<td>Category:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>Spec Reference:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

In the osmo-bts codebase, I can only see readers of bts_ul_meas.is_sub, but not see any code that sets this:

```
src/common/measurement.c: if (m->is_sub) {
```

**Related issues**

Related to OsmoBTS - Bug #2987: OsmoBTS RxQual/RxLev averaging broken if burs...

**Revision 232f53ad** - 02/23/2018 02:05 PM - laforge

osmo-bts-virtual: Make sure PRIM_INFO_MEAS have non-zero frame number

Measurement reports fed into L1SAP so far had their frame number always set to zero, resulting in higher-layer common code above L1SAP to never detect the end of the measurement period, which in turn caused no RSL MEAS REP to be sent.

Related: OS#2978
Change-Id: I67837d1951ea335614928570c12dd5027104c6b

**Revision a38dee5** - 02/23/2018 02:27 PM - laforge

scheduler.c: Factor out find_sched_mframe_idx() function

Let's split the look-up of the multiframe scheduler from the assignment to a given l1s in trx_sched_set_pchan.

Related: OS#2978
Change-Id: I7954ab25aae647ce993a9d3c771d22b08cb1c74

**Revision b82b1b2** - 02/23/2018 02:27 PM - laforge

scheduler: add trx_sched_is_sacch_fn() function

For proper measurement processing of RX{LEV,QUAL}-SUB, we will need this information.

Related: OS#2978
Change-Id: I768fde62452a74dce471ebf946e56eb1e4de1abc

**Revision 1effad10** - 02/23/2018 06:57 PM - laforge

measurement.c: Hand Frame Number into measurement computation

This is currently only used for logging, but will be needed for proper RX{LEV,QUAL}-SUB reporting in upcoming patches.

Related: OS#2978
Change-Id: I07fd06e8a379cab7c0c2eb111c3f560037d3c9e

**Revision aa9ce821** - 02/23/2018 06:57 PM - laforge

l1sap: Pass is_sub from L1 primitive into the Uplink Measurement info_meas_ind on the L1SAP always allowed the lower layers to pass

05/08/2021 1/13
in whether a given measurement is part of the "SUB", or not.

However, the existing l1sap code before this patch simply drops this
information, despite the measurement.c code also having "is_sub" state.

Let's make sure this state is passed from L1SAP into measurement
processing as intended.

Fact is, none of our current lower-layers actually set this is_sub flag
for their primitives passed up in L1SAP, but at least now if they
would set that flag, the measurement code would process it as intended.

Related: OS#2978
Change-Id: lbed2e8d7563b471c6c65b2d21ac4765caf31ed2a

**Revision 63819006 - 02/26/2018 02:01 PM - laforge**

split scheduler_mframe.c from scheduler.c

There are use cases for the multiframe scheduler tables outside the
context of the entire scheduler. Let's prepare for that.

Related: OS#2978
Change-Id: l6a501e66c47809ae3cde55bef2cb63900e096b1

**Revision 3a80a56b - 02/26/2018 02:01 PM - laforge**

measurement: Compute RX{LEV,QUAL}-SUB for SDCCH and non-AMR TCH

The rules on how to compute RX{LEV,QUAL}-SUB are rather convoluted, and
depend on the detailed channel type and mode.

For SDCCH and TCH/H in signalling mode, it's easy: No DTX is allowed,
and all measurements are used in SUB.

For non-AMR (TCH/F and TCH/H in non-signalling mode), we need to count
the SACCH block measurements, as well as any
SID/SID_UPDATE/L3_FILL/DUMMY blocks received in the blocks of table
8.3 of TS 45.008.

Only AMR (TCH/AFS + TCH/AHS) are more difficult, as there are no fixed
blocks/bursts/frames that always contain uplink messages, but the L1
will have to determine if a valid SID_UPDATE was received or not.

This patch implements the above rules (except AMR related) in the common
part of OsmoBTS. The AMR specific bits will have to follow as a later
patch, likely in a BTS specific way, i.e. separate changes to
sysmo/lc15/octphy/trx code.

Related: OS#2978
Change-Id: 116eb3747a1c23df935a4c50dafe46abce512a474

**Revision 1c46d19e - 03/09/2020 12:40 PM - dexter**

parity: add amr crc14 definition

AMR not only specifies a 6 bit CRC for regular voice information. It also
specifies a 14 bit CRC to protect the comfort noise updates contained in
the SID_UPDATE frames.

Change-Id: 55cfd8ca806aba8d42cb9787f669605cea7de6e900
Related: OS#2978

**Revision 31c29ea0 - 03/09/2020 12:40 PM - dexter**

conv: add convolutional coder for AMR SID UPDATE frames

AMR SID update frames are protected using an 1/4 convolutional coder,
which is similar to the one used with 6,7 kbit voice frames. Except that
there is no puncturing and the length is different.

Change-Id: la35ed4178a7f0d816052b7e5d6478b93a1d9744f
Related: OS#2978

**Revision 3a5917bd - 03/13/2020 09:52 AM - dexter**
Fix amr speech bit length table

The table `amr_len_by_ft` represents the length of the raw AMR speech bits in bytes. The table is based on the Table found in RFC 4867 §3.6, Table 1, Column “Total speech bits”. The number of bits is divided by 8 and then rounded up to get the number of consumed octets.

An AMR SID frame (including STI and MI) takes up 39 bits, this will result in 5 octets, not in 7. Let's correct this.

Related: OS#2978
Change-Id: lc1f330450981b32be5d1cee5b10aa92bac4cb72f5

Revision 898c9c6a - 04/14/2020 01:22 PM - dexter
dtx: add decoding for AMR-DTX frames

gsm0503_coding contains AMR decoder functions for HR and FR. Those can only decode AMR payload frames but not AMR DTX frames. Let's add functionality to detect DTX frames. Also let's add decoding for SID_UPDATE frames as well as error checking for the SID frame recognition patterns.

Related: OS#2978
Change-Id: l2bbdb39ea20461ca08b2e6f1a33532cb55cd5195

Revision 906f2d7d - 04/22/2020 01:47 PM - dexter
dtx: add detection of AMR DTX frames for osmo-bts-trx

Currently we do not detect any of the DTX frames (SID_FIRST, SID_UPDATE etc.) Detecting and tagging those frames as is_sub is important for measurement processing. Also the RTP marker bit must be set on each ONSET frame.

- Add detection of DTX frames
  - Tag DTX frames as is_sub and set frame type to AMR_SID
  - Set RTP marker bit when ONSET frames are received

Change-Id: l5afe730ff2fa3199a5913b0de4f5c7b23a39f31
Depends: libosmocore l2bbdb39ea20461ca08b2e6f1a33532cb55cd5195
Related: OS#2978

Revision 58b9f118 - 04/25/2020 02:03 PM - dexter
dtx: add detection of AMR DTX frames for osmo-bts-trx

Currently we do not detect any of the DTX frames (SID_FIRST, SID_UPDATE etc.) Detecting and tagging those frames as is_sub is important for measurement processing. Also the RTP marker bit must be set on each ONSET frame.

- Add detection of DTX frames
  - Tag DTX frames as is_sub and set frame type to AMR_SID
  - Set RTP marker bit when ONSET frames are received

Change-Id: l5afe730ff2fa3199a5913b0de4f5c7b23a39f31
Depends: libosmocore l2bbdb39ea20461ca08b2e6f1a33532cb55cd5195
Related: OS#2978

Revision 78a1d3a3 - 04/25/2020 02:03 PM - dexter
measurement: remove unecessary is_amr_sid_update parameter

The function `ts45008_83_is_sub` roughly decides if a frame is a SUB frame or not. This works by checking the frame number against against lookup tables. This works fine for codecs where the occurrence of SUB frames is fixed. However for AMR this is not the case as the DTX periods are dynamic. Here it is the responsibility of the lower layers (phy. frame decoding) to tag SUB frames early since making the decision later based on the frame number is not possible.

The parameter `is_amr_sid_update` was probably added as a placeholder. It is set to false by the callers of the function. Let's remove this parameter as a late decision if an AMR frame is a SUB frame will never work.
Revision ee320d5b - 05/06/2020 07:39 PM - dexter
measurement: make measurements more debugable

A debug line is printed whenever a measurement is added and it also displays if the added measurement is a sub frame or not, lets display all the other properties too, especially inv_rssi would help a lot as it shows us the rssi values which are reported by the lower layers

Revision 3c0a87ca - 05/12/2020 06:13 PM - dexter
gsm0505_amr_dtx: add missing value strings

The value string array that explain the type of the AMR DTX / SID frames is incomplete, lets add the missing strings.

Revision 41b94e9 - 10/03/2020 07:05 AM - dexter
osmo-bts-trx/scheduler: fix measurement handling for SUB frames

Make sure that we pick the correct UL measurements from the history when we deal with AMR SID frames (SUB frames).

History

#1 - 02/23/2018 12:58 PM - laforge
- Status changed from New to In Progress
- % Done changed from 0 to 70

#2 - 02/23/2018 01:14 PM - laforge
- Subject changed from OsmoBTS rxlev/rxqual SUB computation likely broken to OsmoBTS rxlev/rxqual SUB computation completely broken

See following patch series that fix it for all non-AMR use cases:

https://gerrit.osmocom.org/#/c/6678
https://gerrit.osmocom.org/#/c/6850
https://gerrit.osmocom.org/#/c/6851
https://gerrit.osmocom.org/#/c/6852
https://gerrit.osmocom.org/#/c/6853
https://gerrit.osmocom.org/#/c/6854
https://gerrit.osmocom.org/#/c/6856
https://gerrit.osmocom.org/#/c/6857

#3 - 02/23/2018 03:03 PM - laforge
- Related to Bug #2987: OsmoBTS RxQual/RxLev averaging broken if bursts are misign added

#4 - 02/26/2018 02:11 PM - laforge
- Checklist item [ ] [RxQual,RxLev]-SUB for AMR DTX added
- Status changed from In Progress to Stalled

#5 - 03/21/2018 08:24 AM - laforge
- Subject changed from OsmoBTS rxlev/rxqual SUB computation completely broken to OsmoBTS rxlev/rxqual SUB computation completely broken [AMR DTX]

#6 - 08/08/2018 09:48 AM - laforge
- Assignee changed from laforge to dexter
measurement.c expects the lower layers to set the is_sub flag correctly. We need a small FSM that looks into the packets and switches between silent periods and non silent periods so that we know when to set the is_sub flag and when not. The SID_UPDATE frames should come in a fixed interval, so for those we know when to expect them.

While we tag the measurements in the lower layers we also must record some information about the silent periods. We need to keep a record of the exact number of SUB measurements we expected and this number must then be reported to measurement.c. If we don't do this we would have correctly tagged SUB measurements but we still wouldn't be able say anything about lost is-sub frames.

I need to try a few more things out. I now have a setup where AMR with DTX is turned on. I can see and hear that it is enabled. Next we need some code to look into each voice frame (RTP) in order to detect the SID_FIRST and ONSET frames.

File amr_dtx_with_omso-trx.pcapng added
- File amr_dtx_with_sysmo-bts.pcapng added

I have tried to detect when SID frames. The plan is (as described above) to make an FSM similar to dtx_dl_amr_fsm.c that observes the incoming RTP stream. For my experiment I have tapped the incoming frames directly in l1sap.c:l1sap_tch_ind(). Then I used osmo_amr_rtp_dec() to decode the frames. This gives me the frame type (ft) and in theory I should be able to check with osmo_amr_is_speech(ft) if I see a voice frame or a SID frame.

In my experiment this unfortunately did not work. I don't think that there is something wrong with the utilities. When I look at the trace I can see that there are no SID frames in it, only voice frames are visible. Also all the marks are missing. I think osmo-trx has some problems here, which we need to fix first.

When comparing to a trace from a sysmo-bts things look different. Here the marks are visible and there are also some comfort noise AMR-SID frames. However, I would expect to see some more SID frames. I miss ONSET and SID_FIRST frames. Presumably there are also some lower level problems here as well.

Attached one finds the two traces I made

I have set this to stalled. We will pick this up again when all remaining measurement and frame number calculation/scheduling problems are resolved.

I now have a bit of an understanding what to expect but I am unable to get smart out of the frames we actually receive. A SID frame should be distributed over 4 sub frames, where one sub frame should be one MAC-Block. The spec says that the RX SCR determines the frame type to determine what the frame actually is. (3GPP TS 26.093, Chapter E.2, Table 8)

in rx_tchf_fn() in the switchcase GSM48_CMODE_SPEECH_AMR, I get Macblocks like this:

2014e959f35fdfe5e9667ffbc0888180880101000101

02 is the header I also can see in Wireshark. 1 = 0001 ==> 000, which should mean "SPEECH_GOOD". I can not see any other frame, I would expect if there are SID frames that they are transmitted in four consecutive mac blocks and that they are marked with 101 = "POST1" (Correct SID update frame).

My Idea is to match on the SID frames on block level. I wonder if this is even possible like I thought it out. I think no. Because the spec mentions a deframing unit which does the tagging. Any hints would be very much appreciated.

I came a bit further with this. I now have some code to detect the SID_FIRST frame and I am now working on the SID_UPDATE detection. The problem why I wasn't able to detect the marker patterns was because there is an interleaving in between. The docs aren't all too clear about this.

- % Done changed from 70 to 80
I now got my head around the interleaving. I have programmed some detector functions that can detect the various different types of SID frames by their identification pattern or by their coded inband data repetition sequence (depends on the sid frame type). We can now detect the following sid frames:

AFS_SID_FIRST
AFS_SID_UPDATE
AFS_ONSET
AHs_SID_UPDATE
AHs_SID_FIRST_P1
AHs_SID_FIRST_P2
AHs_ONSET
AHs_SID_FIRST_INH
AHs_SID_UPDATE_INH

AHs SID_FIRST_INH

(coded like an onset frame since it also occurs on sub-block 3 and 4)

xBxBxBxBxBxBxBxBxBxBxBxBxBxBxBxBx1x0x1x1x0x0x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0x1x0x0x0x0
The experimental code that detects the DTX frames in osmo-bts-trx can be found here: pmaier/dtxtest Currently I only print the frame type. Next I will make sure that the is_sub flag is set whenever a DTX frame is received.

I had a look on how osmo-bts-sysmo does the DTX frame handling in tch.c:l1if_tch_rx(). There it seems not do do anything with the information other than setting the rtp_tx_marker to true for ONSET and the INH frames. I wonder if we should find a way here to set the is_sub flag here too.

(also helpful: https://ftp.osmocom.org/docs/latest/rtp-amr.pdf)

#16 - 02/17/2020 09:16 AM - dexter

The next step here is to tag the incoming frames as correctly "is_sub". For osmo-bts-trx this is now easy, but it should be made working for osmo-bts-sysmo as well. We have two problems. The first is that measurement data and payload data are sent separately through l1sap. This has to be resolved first. The other is that the phy only identifies SID_FIRST and ONSET for us, the SID_UPDATE frames are not identified. I am not sure what the best way to resolve this. A state flag could be enough but I think its better when we somehow identify the frames, this should be possible.

Also the marker bit for AMR_ONSET must be set as well. Osmo-bts-sysmo does this by setting lchan->rtp_tx_marker to true. I have copied this method to osmo-bts-trx

#17 - 02/28/2020 03:58 PM - dexter

Found out that in osmo-bts-sysmo also the SID_UPDATE is not identified. I think those frames are then IF2 formatted and we need to look into the frames to check if they are SID frames.

However, we still need to decode the SID_UPDATE frames we receive in osmo-bts-trx. I have done some experiments and I can now extract the data from SID_UPDATE frames. I can also apply the convolutional decoder and the CRC check on them for the two frames I experimented with the CRC check is successful, that means the data is decoded correctly. Now I have to integrate that somehow in libosmocore. I also think we also will have to mangle the BITs a bit so that we get a valid AMR TRAU frame (in IF2 format?).

#18 - 02/28/2020 04:12 PM - laforge

On Fri, Feb 28, 2020 at 03:58:43PM +0000, dexter [REDMINE] wrote:

   Found out that in osmo-bts-sysmo also the SID_UPDATE is not identified. I think those frames are then IF2 formatted and we need to look into the frames to check if they are SID frames.

In general, one selects either RTP or IF2 mode, and I would be surprised if even in RTP mode some would be reported as IF2.

   However, we still need to decode the SID_UPDATE frames we receive in osmo-bts-trx. I have done some experiments and I can now extract the data from SID_UPDATE frames. I can also apply the convolutional decoder and the CRC check on them for the two frames I experimented with the CRC check is successful, that means the data is decoded correctly.

great.

   Now I have to integrate that somehow in libosmocore. I also think we also will have to mangle the BITs a bit so that we get a valid AMR TRAU frame (in IF2 format?).

Where do we need the TRAU format? That format is only used between the BTS and BSC over a circuit-switched E1/T1 Abis interface (16k sub-slots with TRAU frames). So that's only relevant to Nokia/Siemens/Ericsson BTSs, but not to the IP based BTS models such as osmo-bts-

#19 - 03/06/2020 09:16 AM - dexter

I see TRAU format is only relevant for E1/T1, I have confused this with the RTP format.

I have done some integration now in osmo-bts-trx. In FR AMR the sid update decodes fine but in HR AMR I get a lot of CRC errors. Only a few frames decode properly. Since there is no direct connection between BTS and MS, this could be due to interference. The CRC errors get less when I shorten the distance between BTS and MS. However I still have the feeling that something might be wrong here.

There is also another problem I see. For the convolutional encoded section of the SID_UPDATE i get an n_bis_total and an n_errors value but for the other SID frames i do not determine those values. I could fix this. All i need to do is to count out the deviations to the marker pattern. But then the question is what to do with frames other than SID_UPDATE and SID_FIRST. Those won't generate an RTP packet but they would generate a measurement. Presumably I would need to pass an AMR_BAD frame or something up to the higher layers.

#20 - 03/09/2020 09:39 PM - dexter
I have worked out the integration of the SID_FIRST and SID_UPDATE decoding into osmo-bts-trx to the point that the decoder in libosmocore outputs a frame that should be suitable to be sent via RTP. However, re-reading the spec revealed that there needs to be a frame type prepended to the 35 decoded bits. This is very confusing, i first thought that those 4 bit frame types were already the RTP header, but apparently the CMR and TOC data is something independent. After all we never really see a proper IF2 frame in our chain, its basically the RTP format without header that leaves the decoder. I also investigated if there is any reordering specified for the comfort noise bits themselves, fortunately there is no reordering. The original order of the bits is preserved.

It also looks like if there were a solution for the problem that we can not pass SID frames up that would not translate into any RTP frame. There is a frame type 15 which basically means that the frame does not contain any useful data, I think the SID frames that terminate at the BTS should generate a type 15 frame and send it to the higher layer. There then the measurement data is processed, but no RTP packet is emitted.

Also we still need to think about counting out the error bits. At the moment the decoder for the SID frames only counts out the errors roughly and does not report them back. This needs to be extended at least to the marker patterns so that we can deliver a value for n_bits_total and n_errors.

#21 - 03/27/2020 11:26 AM - dexter

The error bits are now counted out. However I was noticing that we can not just accept any pattern of coded inband data. The spec defines 4 distinctive patterns. I have changed the checks now. Fortunately since the coded inband data can now only consist of 4 well defined patterns and not just random bits checking becomes much easier. I also hope that checking for ONSET frames now becomes more reliable.

I was also noticing that there were a lot of bad frames in the RTP stream which only consisted of 0x00 bytes. I saw that the phy based BTSs call lchan_set_marker() allow for skipping of the not sent DTX frames. We did not call this function before, presumably because we did not have SID frame detection but now we have. However, I can see the amount of RTP frames is vastly reduced during the silence periods. I need to do a couple more tests, I also did not test HR yet.

#22 - 03/30/2020 04:49 PM - dexter

While testing I noticed that on FR none of the SID_UPDATE frame could be decoded anymore. The reason turned out to be quite surprising. Actually we have now the same behavior as we are experiencing with HR. I did not get my head around it yet but it makes sense together with the fact that we have two times coded-inband-data inside the SID_UPDATE. The sudden change of the behaviour is due to fact that we now also pass nope indications for the tch channels. I think that we were seeing SID_UPDATE + Marker pattern in one frame was just because some blocks were counted wrongly. I now need to fix this so that it works the same as it does on HR. I ran a quick test on HR today and to my surprise also the decoding there has been improved (could also be due to more correct configs for osmo-bts-trx and osmo-trx-uhd).

#23 - 04/06/2020 02:00 PM - dexter

- % Done changed from 80 to 90

Change Ice45d5986610d9bcef2a7e4110a395ec779e3928 has fixed some timing/clocking issues so that some of the behavior during DTX looked different, what seemed to be correct really was not correct. I have revisited the DTX changes now and also fixed the remaining problems. One major problem was that during a silent period osmo-bts-trx continued to emit bad RTP packets. Also the first (ONSET) frame of the beginning of the voice spurt was always bad. This now fixed and I feel confident that we can now review and merge it.

#24 - 04/23/2020 04:22 PM - dexter

The following patches are currently in review:

https://gerrit.osmocom.org/c/osmo-bts/+/17221 dtx: add detection of AMR DTX frames for osmo-bts-trx
https://gerrit.osmocom.org/c/osmo-bts/+/17928 measurement: remove unnecessary is_amr_sid_update parameter
https://gerrit.osmocom.org/c/osmo-bts/+/17929 measurement: expect at least 1 SUB frame for AMR

While the mechanics look good (we receive plausible DTX frames and we can decode them), I am noticing that the measurement results (RSSI-VALUES) are actually looking very strange:

The following is for FR-AMR:

Without DTX enable I get the following:

Thu Apr 23 12:49:06 2020 <0004> measurement.c:706 (bts=0,trx=0,ts=1,ss=0) Computed TA256( 7) BER-FULL( 0.00%), RSSI-FULL(- 33dBm), BER-SUB( 0.00%), RSSI-SUB(- 33dBm)
Thu Apr 23 12:49:06 2020 <0004> measurement.c:706 (bts=0,trx=0,ts=2,ss=0) Computed TA256( 27) BER-FULL( 0.00%), RSSI-FULL(- 42dBm), BER-SUB( 0.00%), RSSI-SUB(- 42dBm)

This looks normal to me but when I enable DTX I get much lower RSSI values, this is strange because during the whole test I did not change the position of the mobile phone, the external condition are the same, I assume tat I get values that are at least in the ballpark of the values I get when DTX is off.

Thu Apr 23 12:52:57 2020 <0004> measurement.c:706 (bts=0,trx=0,ts=1,ss=0) Computed TA256( 0) BER-FULL(72.00%), RSSI-FULL(- 87dBm), BER-SUB( 0.00%), RSSI-SUB(- 60dBm)
Thu Apr 23 12:52:57 2020 <0004> measurement.c:706 (bts=0,trx=0,ts=2,ss=0) Computed TA256( -4) BER-FULL(72.00%)
I found the cause of the problem, however I do not get why it is programmed that way but first lets look at the following log:

```plaintext
====FN_CALL_WITH_RSSI=> tn=1, rssi=-98
====FN_CALL_WITH_RSSI=> tn=2, rssi=-98
====FN_CALL_WITH_RSSI=> tn=1, rssi=-96
====FN_CALL_WITH_RSSI=> tn=2, rssi=-98
====FN_CALL_WITH_RSSI=> tn=1, rssi=-98
====FN_CALL_WITH_RSSI=> tn=2, rssi=-97
Thu Apr 23 17:49:33 2020 <0007> scheduler_trx.c:1283 1684972/1270/16/34/28 (bts=0,trx=0,ts=1) TCH/F: Received bad data (68/104)

=================> ts=1 -97, 0
Thu Apr 23 17:49:33 2020 <0004> measurement.c:345 1684964/1270/08/26/20 (bts=0,trx=0,ts=1,ss=0) adding measurement (is_sub=0), num_ul_meas=15, fn_mod=60
====FN_CALL_WITH_RSSI=> tn=2, rssi=-98
Thu Apr 23 17:49:33 2020 <0007> scheduler_trx.c:1283 1684972/1270/16/34/28 (bts=0,trx=0,ts=2) TCH/F: Received bad data (68/104)

=================> ts=2 -98, 0
Thu Apr 23 17:49:33 2020 <0004> measurement.c:345 1684964/1270/08/26/20 (bts=0,trx=0,ts=2,ss=0) adding measurement (is_sub=0), num_ul_meas=12, fn_mod=60
====FN_CALL_WITH_RSSI=> tn=1, rssi=-97
====FN_CALL_WITH_RSSI=> tn=2, rssi=-49
====FN_CALL_WITH_RSSI=> tn=1, rssi=-97
====FN_CALL_WITH_RSSI=> tn=2, rssi=-50
====FN_CALL_WITH_RSSI=> tn=1, rssi=-99
====FN_CALL_WITH_RSSI=> tn=2, rssi=-49
====FN_CALL_WITH_RSSI=> tn=1, rssi=-97
Thu Apr 23 17:49:33 2020 <0007> scheduler_trx.c:1283 1684972/1270/16/34/28 (bts=0,trx=0,ts=1) TCH/F: Received bad data (72/104)

=================> ts=1 -97, 0
Thu Apr 23 17:49:33 2020 <0004> measurement.c:345 1684969/1270/13/31/25 (bts=0,trx=0,ts=1,ss=0) adding measurement (is_sub=0), num_ul_meas=16, fn_mod=65
====FN_CALL_WITH_RSSI=> tn=2, rssi=-49
====RSSI-FOR-SID=> tn=2, rssi=-49
Thu Apr 23 17:49:33 2020 <0007> scheduler_trx.c:1283 1684972/1270/16/34/28 (bts=0,trx=0,ts=2) TCH/F: Received bad data (72/104) with invalid codec mode 3

=================> ts=2 -49, 1
Thu Apr 23 17:49:33 2020 <0004> measurement.c:345 1684969/1270/13/31/25 (bts=0,trx=0,ts=2,ss=0) adding measurement (is_sub=1), num_ul_meas=13, fn_mod=65
====FN_CALL_WITH_RSSI=> tn=1, rssi=-98
====FN_CALL_WITH_RSSI=> tn=2, rssi=-97
====FN_CALL_WITH_RSSI=> tn=1, rssi=-98
====FN_CALL_WITH_RSSI=> tn=2, rssi=-98
====FN_CALL_WITH_RSSI=> tn=1, rssi=-98
====FN_CALL_WITH_RSSI=> tn=2, rssi=-98
Thu Apr 23 17:49:33 2020 <0007> scheduler_trx.c:1283 1684972/1270/16/34/28 (bts=0,trx=0,ts=1) TCH/F: Received bad data (76/104)

=================> ts=1 -97, 0
Thu Apr 23 17:49:33 2020 <0004> measurement.c:345 1684973/1270/17/35/29 (bts=0,trx=0,ts=1,ss=0) adding measurement (is_sub=0), num_ul_meas=17, fn_mod=69
====FN_CALL_WITH_RSSI=> tn=2, rssi=-98
====RSSI-FOR-SID=> tn=2, rssi=-98
Thu Apr 23 17:49:33 2020 <0007> scheduler_trx.c:1283 1684972/1270/16/34/28 (bts=0,trx=0,ts=2) TCH/F: Received bad data (72/104) with invalid codec mode 3

```
"FN_CALL_WITH_RSSI" is just a marker that tells me that rx_tchf_fn() was called and what the RSSI of the burst was that came in. "RSSI-FOR-SID" is the point that marks that the DTX block is complete and which RSSI the block has. We see that we receive the AFS_SID_UPDATE (marker) with an RSSI of -49, this is plausible but already problematic since the the code only uses the RSSI value of the last burst in the block. All other values are ignored. I do not know if this is a bug or if we just assume that the value wouldn't change much anyway, so taking the last one is enough. So far everything is still fine but when the AFS_SID_UPDATE_CN (audio) is received we get an RSSI value of -98, which can not be.

In the sourcecode I can see that the burst is first built up at the high side of the buffer (see line 1166) but when the burst is complete the code runs on the beginning of the buffer. Once the code is through with its work on the buffer it copies the high side of the buffer to the beginning. I do not really get what the purpose of this shifting, I also don't see that the high side of the buffer is used somewhere until it gets copied to the low side. This partially explains why the RSSI of the CN data is messed up. However I do not fully understand this yet. The problem is caused by the fact that when the burst data is processed it uses the RSSI f
I will investigate this further, but I think we need to find a way to somehow lock the RSSI values that arrive on burst level onto the actual data so that we are sure that the RSSI levels actually belong the the block that we pass up to the higher layers.

#25 - 04/24/2020 06:53 AM - laforge

On Thu, Apr 23, 2020 at 04:22:07PM +0000, dexter [REDMINE] wrote:

```
problematic sind the the code only uses the RSSI value of the last burst in the block. All other values are ignored. I do not know if this is a bug or if we just assume that the value wouldn't change much anyway, so taking the last one is enough.

This is a bug. It should be the average of those bursts, IMHO

In the sourcecode I can see that the burst is first built up at the high side of the buffer (see line 1166) but when the burst is complete the code runs on the beginning of the buffer. Once the code is through with its work on the buffer it copies the high side of the buffer to the beginning. I do not really get whats the purpose of this shifting,

It relates to the spreading / interleaving of one voice frame across 8 bursts. Every four bursts one voice frame is complete, but you still need 8 bursts to decode it.

I also don't see that the high side of the buffer is used somewhere until it gets copied to the low side.

it is used when the next voice frame is to be decoded...

I will investigate this further, but I think we need to find a way to somehow lock the RSSI values that arrive on burst level onto the actual data so that we are sure that the RSSI levels actually belong the the block that we pass up to the higher layers.

I you want to do it correct and the same way as the data bits, you would have to store the per-burst RSSI values also in an array with eight entries, and apply the same shifting logic. Then every time you report a voice frame upwards, you take the average of those eight values.
```

#26 - 04/24/2020 08:37 AM - fixeria

Hi Harald, Philipp,

This is a bug. It should be the average of those bursts, IMHO

ACK.

I you want to do it correct and the same way as the data bits, you would have to store the per-burst RSSI values also in an array with eight entries, and apply the same shifting logic. Then every time you report a voice frame upwards, you take the average of those eight values.

This is exactly how it's implemented in trxcon, using a stack allocated ring-buffer. I am planning to port my changes to osmo-bts-trx as soon as I find some time for that. See https://git.osmocom.org/osmocom-bb/commit/?id=2060b5b7cc3b63b64e651d3eda5ed50b44593a05.

#27 - 04/24/2020 09:06 AM - dexter

I see, gsm0503_tch_ahs_decode_dtx() is accessing the far end of the buffer as well, now its clear to me.

eight entries, and apply the same shifting logic.

I had the same idea last night, thanks for confirming. I will implement it that way.

#28 - 05/04/2020 06:58 PM - dexter
I have now fixed the remaining problems with the sub frames. We now get meaningful RSSI values for the SUB frames and we also average the RSSI values that we pass up to higher layers. I have also tested the patch with all possible voice codecs.

The following patches are now in review:
https://gerrit.osmocom.org/c/osmo-bts/+/18034 measurement: make measurements more debugable
https://gerrit.osmocom.org/c/osmo-bts/+/18035 scheduler_trx: fix RSSI calculation for SUB frames
https://gerrit.osmocom.org/c/osmo-bts/+/17929 measurement: expect at least 1 SUB frame for AMR

#29 - 06/15/2020 04:27 PM - dexter

Patch https://gerrit.osmocom.org/c/osmo-bts/+/18035 is still in review, unfortunately it suffers from merge conflicts. A quick inspection of the situation reveals that there were drastic changes in scheduler_trx.c recently.

#30 - 06/15/2020 04:30 PM - dexter
- % Done changed from 90 to 50

#31 - 06/22/2020 09:18 AM - dexter
- Assignee changed from dexter to fixeria

#32 - 06/23/2020 08:51 AM - fixeria
- Status changed from In Progress to Feedback
- Assignee changed from fixeria to dexter
- % Done changed from 50 to 80

dexter wrote:
Patch https://gerrit.osmocom.org/c/osmo-bts/+/18035 is still in review, unfortunately it suffers from merge conflicts. A quick inspection of the situation reveals that there were drastic changes in scheduler_trx.c recently.

I implemented measurement averaging using a simple ring buffer, so now not only RSSI is handled, but also ToA256 and C/I. Change by dexter has been rebased on top of it (thus on top of the current master) and now waiting for review in Gerrit.

I would still want to look at the related parts of the spec., because right now this ("4 low bursts of 8" or "2 middle bursts of 6") looks like a magic to me, and I guess would look so for other people reading the code. dexter any hints? Thanks!

#33 - 06/23/2020 05:40 PM - laforge

On Tue, Jun 23, 2020 at 08:51:07AM +0000, fixeria [REDMINE] wrote:
I would still want to look at the related parts of the spec., because right now this ("4 low bursts of 8" or "2 middle bursts of 6") looks like a magic to me, and I guess would look so for other people reading the code. dexter any hints? Thanks!

I don't think it matters much. Measurements are aggregated / averaged over time at various different levels. So as long as all of the bursts we are using are from the same logical channel, we can't really do anything wrong.

#34 - 06/24/2020 08:41 PM - fixeria

Measurements are aggregated / averaged over time at various different levels.

Ok, that's clear. So far it worked even with measurements of the last burst of a block, and (almost) nobody complained ;)

So as long as all of the bursts we are using are from the same logical channel, we can't really do anything wrong.

Unless we deal with DTX, where some bursts are intentionally skipped. If we pick wrong entries in the measurement history, we get wrong measurements (basically noise measurements). That's why I am curious. After all, I believe that dexter took those "4 first bursts of 8" or "2 middle bursts of 6" averaging modes from some document, and they are not just random guesses.

#35 - 07/04/2020 09:50 AM - fixeria
After all, I believe that Philipp took those "4 first bursts of 8" or "2 middle bursts of 6" averaging modes from some document, and they are not just random guesses.

_dexter_ ping?

#36 - 07/06/2020 05:05 PM - dexter

I found the burst combination through experimenting. While in non-dmx mode the selection of the bursts does not have much effect, however I think averaging over 8 bursts instead of 4 is not all too good because of the diagonal interleaving the the measurement results are valid for the current and for the adjacent burst at the same time. Taking four should be ok, especially when the results are getting averaged by the higher layers it should not be all too important if the computation was done over 4 or over 8 bursts (4 or 2 for HR) - but that is not the problem here.

When using DTX there may be blocks where only one half of bursts is used to transmit the DTX frame. So I decided to pick the bursts I use for averaging in a way so that when a DTX frame that announces the begin of a silence interval is received the 4 (2) bursts with actual information are picked instead of the ones that contain silence. It is difficult to detect this situation in advance.

You can make an experiment: Enable AMR DTX, make a call and trigger between silence periods and voice spurts by blowing into the mic of the phone. When you print the RSSI for each burst individually you will see what I mean.

#37 - 07/06/2020 08:23 PM - fixeria

Thank you, Philipp!

I found the burst combination through experimenting.

Nice! I'll then add comments to the code stating that those magic measurement modes were found experimentally.

When using DTX there may be blocks where only one half of bursts is used to transmit the DTX frame.

Let's just make sure that we understand "only one half" correctly. You cannot just send half of a burst, it must always be two halves (57 bit each, 114 payload bits in total) together, plus a training sequence for demodulator. In context of the block-diagonal interleaving, one half means either odd bits of a burst, or even bits of a burst. So a L2 block is **equally distributed** over N consecutive bursts. That's why I believe that we should be calculating AVG for the whole distribution period. But as was mentioned, averaging in the higher layers makes this aspect less critical.

When using DTX there may be blocks where only one half of bursts is used to transmit the DTX frame.

So in case of e.g. TCH/F, where each L2 frame is interleaved over 8 consecutive bursts,

- unused halves (odd numbered bits) of the first 4 bursts are set to zero or simply reflect the old content of the buffers;
- unused halves (even numbered bits) of the last 4 bursts are filled according to 3GPP TS 45.002, section A.2.2.

So here is an extract from section A.2.2 named "Half burst filling":

For downlink DTX, when a given traffic frame is scheduled for transmission and one of its adjacent traffic frames is not scheduled for transmission, half of the "encrypted bits" belonging to the normal bursts associated with the scheduled traffic frame need to be filled. These bits are referred to as "half burst filling bits". These half bursts filling bits contain either:

- A.2.2.1 Partial SID information from any associated SID frame; or
- A.2.2.2 The mixed bits of the dummy bursts (encrypted or not encrypted).

So the specs. say *half of the "encrypted bits"*, not half of the normal burst itself. Regarding the interleaving/mapping of these partial SID frames, we should refer to sections 3.9 and 3.10 of 3GPP 45.003.

So I decided to pick the bursts I use for averaging in a way so that when a DTX frame that announces the begin of a silence interval is received the 4 (2) bursts with actual information are picked instead of the ones that contain silence. It is difficult to detect this situation in advance.

Ah, I think I understand your idea now. So we're talking about *SID_FIRST* (TCH/AFS) and *SID_FIRST\{P1,P2\}*_ (TCH/AHS).

*SID-FIRST* (TCH/AFS) is interleaved in the same way as a regular full-rate speech or FACCH/F frame - over 8 consecutive bursts:

**3.9.2.3 Interleaving**

The interleaving is done as specified for the TCH/FS in subclause 3.1.3.
3.9.2.4 Mapping on a Burst

The mapping is done as specified for the TCH/FS in subclause 3.1.4. The last 4 bursts shall not be transmitted unless the SID_FIRST frame is immediately followed by a speech frame.

SID_FIRST[P1,P2] is interleaved in the same way as a regular half-rate speech frame - over 4 consecutive bursts.

3.10.3.3 Interleaving

The interleaving is done as specified for the TCH/HS in subclause 3.2.3.

3.10.3.4 Mapping on a Burst

The mapping is done as specified for the TCH/HS in subclause 3.2.4.

It looks like we should handle them in the same way as regular speech frames, not like "first 2 bursts of 6" or "first 4 bursts of 8". Correct me if I am wrong. Other SID frames are interleaved in a different way, I didn't look further.

Another interesting question is how do we handle the measurements triggered by NOPE.ind after reception of SID_FIRST (on TCH/AFS) or SID_FIRST[P1,P2] (on TCH/AHS). The reported RSSI values will be equal to the noise levels, so we should make sure that we ignore them during the intervals of silence. Not sure if and where it's implemented.

Best regards,
Vadim.

#38 - 09/03/2020 10:29 AM - dexter
- Status changed from Feedback to Stalled
- Assignee changed from dexter to fixeria

It is not possible to handle the SID frames like voice frames as they do not fill up the whole time period of a voice frame. They are more like half-frames. There is some printed documentation available in the office.

As far as I know we get RSSI levels on a per burst level. We also can examine the bit errors in the SID frames, so it is possible to pass up RSSI and bit error levels there.

#39 - 10/04/2020 11:54 AM - fixeria
- Checklist item [x] {RxQual,RxLev}-SUB for AMR DTX set to Done
- Status changed from Stalled to Feedback
- Assignee changed from fixeria to dexter
- % Done changed from 80 to 100

Patch has been merged (yay!).

[x] {RxQual,RxLev}-SUB for AMR DTX

Thanks to unified measurement processing API [1], we now handle both RxQual (BER + C/I) and RxLev (UL RSSI).

dexter let me know if anything is still missing, otherwise let's close this ticket.


#40 - 10/05/2020 02:07 PM - dexter
- Assignee changed from dexter to fixeria

Thanks for taking care of this. I think this can be closed now.

#41 - 10/05/2020 03:56 PM - fixeria
- Status changed from Feedback to Resolved

Files
amr_dtx_with_omso-trx.pcapng 926 KB 08/30/2018 dexter
amr_dtx_with_sysmo-bts.pcapng 935 KB 08/30/2018 dexter