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

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

```
Related issues:
Related to OsmoBTS - Bug #2987: OsmoBTS RxQual/RxLev averaging broken if bursts... Resolved 02/23/2018
```

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

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.

#8 - 08/30/2018 11:26 AM - dexter
- 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

#9 - 09/03/2018 07:42 AM - dexter
- Status changed from In Progress to Stalled

#10 - 09/03/2018 07:44 AM - dexter

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

#11 - 01/23/2020 01:26 PM - dexter

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:

```
2014e959f35fdfe5e9667ffbc088818088010100000101
```

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.

#12 - 02/03/2020 01:39 PM - dexter
- Status changed from Stalled to In Progress

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.

#13 - 02/04/2020 02:23 PM - dexter
- % 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
For the last two I am not sure if I have done it right since I have no real world sample of those, but to me it looks correct.

The next problem to solve would be to make sure that the incoming frames are correctly tagged and handed upwards. Also we do not have any deciding yet. I do not know what the coded inband data is for but I think we need to be able to decode the SID_UPDATE frames since those contain comfort noise updates. The SID_UPDATE has a 1/4 convolutional coder in between and is CRC protected.

For the last two I am not sure if I have done it right since I have no real world sample of those, but to me it looks correct.

The next problem to solve would be to make sure that the incoming frames are correctly tagged and handed upwards. Also we do not have any deciding yet. I do not know what the coded inband data is for but I think we need to be able to decode the SID_UPDATE frames since those contain comfort noise updates. The SID_UPDATE has a 1/4 convolutional coder in between and is CRC protected.

#14 - 02/05/2020 11:50 AM - laforge

Hi dexter,

On Tue, Feb 04, 2020 at 02:23:34PM +0000, dexter [REDMINE] wrote:

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:

this is great progress, I'm really happy to read about it!

The next problem to solve would be to make sure that the incoming frames are correctly tagged and handed upwards. Also we do not have any deciding yet.

I'm not sure what you are trying to express here: "do not have any deciding yet"?

I do not know what the coded inband data is for but I think we need to be able to decode the SID_UPDATE frames since those contain comfort noise updates. The SID_UPDATE has a 1/4 convolutional coder in between and is CRC protected.

#15 - 02/06/2020 01:42 PM - dexter

There was a typo in the last message "deciding" => "decoding"

The current code that is used to detect the DTX frames is now in gerrit, however I think it is not perfect yet. [https://gerrit.osmocom.org/c/libosmocore/+/17095](https://gerrit.osmocom.org/c/libosmocore/+/17095) dtx: add functions to determine DTX frame types

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-symo does the DTX frame handling in tch.c11f_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.
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. There we face 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 it's 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

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?).

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.

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

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).

#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% ), RSSI-FULL(- 83dBm), BER-SUB( 0.00%), RSSI-SUB(- 48dBm)

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:

======FN_CALL_WITH_RSSI=> tn=1, rssi=-98

05/16/2020 5/8
"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 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 would not 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 source code 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 the 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 from the burst that concludes the block, but the data is from the block before? And why is looking everything like if there were marker and CN data in the same block when observing the RSSIs while this seems not to be the case when looking at the data (marker in one block, CN data in the following block).

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 what's 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=2060b5b7cc3b63b64e651d3cda5ed50b44593a05.

#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

<table>
<thead>
<tr>
<th>Files</th>
<th>Size</th>
<th>Date</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>amr_dtx_with_omso-trx.pcapng</td>
<td>926 KB</td>
<td>08/30/2018</td>
<td>dexter</td>
</tr>
<tr>
<td>amr_dtx_with_sysmo-bts.pcapng</td>
<td>935 KB</td>
<td>08/30/2018</td>
<td>dexter</td>
</tr>
</tbody>
</table>