According to 3GPP TS 45.002, section 6.3.2.2.2, and 3GPP TS 45.010, section 3.3.4.2:

- **PTCCH/D** (Packet Timing advance control channel, Downlink) is used to transmit **Timing Advance updates** for several MS. One PTCCH/D is paired with several PTCCH/U's. PTCCH/D frames are encoded using CS-1.
- **PTCCH/U** (Packet Timing advance control channel, Uplink) is used to transmit **random access bursts** to allow estimation of the Timing Advance for one MS in packet transfer mode. PTCCH/U has 16 sub-channels (0..15) assigned to different MS using TAI.

Therefore, neither we properly handle Downlink PTCCH nor Uplink PTCCH in osmo-bts-trx:

https://git.osmocom.org/osmo-bts/tree/src/common/scheduler.c#n536

```c
[TRXC_PTCCH] = {  
  .name = "PTCCH", /* 3GPP TS 05.02, section 3.3.4.2 */  
  .desc = "Packet Timing advance control channel",  
  .chan_nr = RSL_CHAN_OSMO_PDCCH,  
  /* Same as for TRXC_BCCH (xCCH), see above. */  
  .flags = TRX_CHAN_FLAG_PDCCH,  
  .rts_fn = rts_data_fn,  
  .dl_fn = tx_data_fn, /* FIXME: we need tx_pdtch_fn() here! */  
  .ul_fn = rx_data_fn, /* FIXME: we need rx_rach_fn() here! */  
},
```

**Related issues:**

Related to OsmoPCU - Feature #1545: continuous timing advance loop using PTCCH

<table>
<thead>
<tr>
<th>Revision</th>
<th>Author</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df5b813c</td>
<td>Vadim Yanitskiy</td>
<td>10/04/2019</td>
<td>scheduler: fix handling of PTCCH/U and PTCCH/D logical channels</td>
</tr>
</tbody>
</table>

According to 3GPP TS 45.010, section 5.6.2, for packet-switched channels the BTS shall monitor the delay of the Access Bursts sent by the MS on PTCCH and respond with timing advance values for all MS performing the procedure on that PDCCH.

According to 3GPP TS 45.002, section 3.3.4.2, PTCCH (Packet Timing advance control channel) is a packet dedicated channel, that is used for continuous Timing Advance control (mentioned above).

There are two sub-types of that logical channel:

- **PTCCH/U** (Uplink): used to transmit **random access bursts** to allow estimation of the Timing Advance for one MS in packet transfer mode.
- **PTCCH/D** (Downlink): used by the network to transmit **Timing Advance updates** for several MS.

As per 3GPP TS 45.003, section 5.2, the coding scheme used for PTCCH/U is the same as for PRACH as specified in subclause 5.3, while the coding scheme used for PTCCH/D is the same as for...
CS-1 as specified in subclause 5.1.1.

The way we used to handle both PTCCH/U and PTCCH/D is absolutely wrong - they have nothing to do with xCCH coding. Instead, we need to use tx_pdtch_fn() for Downlink and rx_rach_fn() for Uplink.

In l1sap_ph_rach_ind() we need to check if an Access Burst was received on PTCCH/U and forward it to OsmoPCU with proper SAPI value (PCU_IF_SAPI_PTCCH). To be able to specify a SAPI, a new parameter is introduced to pcu_tx_rach_ind().

Change-Id: i232e5f514fbad2c51daaa59ff516004aba97c8a3
Related: OS#4102

Revision a070e863 - 11/22/2019 05:38 PM - Vadim Yanitsky
pcuif_proto.h: extend RACH.ind with TRX and timeslot number fields

Since there can be multiple PDCH channels configured on different timeslots, different TRXes, and BTSes, the PTCCH/U handling code in OsmoPCU needs to know the exact origin of a given RACH.ind.

Otherwise, it is not known which subscriber originated a given PTCCH/U indication, and hence it is impossible to send PTCCH/D Timing Advance notification properly.

Fortunately, we can extend the RACH.ind message without even bumping the protocol version, because every single PDU has a fixed size defined by the largest message - INFO.ind. In case if the actual message payload is smaller, the rest is filled with a constant padding byte (0x00).

Older versions of OsmoPCU will consider the new fields as padding, while the messages from older OsmoBTS versions will always have both fields set to 0x00. Since C0/TS0 cannot be configured to PDCH, this can be easily detected on the other end.

Change-Id: iff38934a108b6b1cd298e69834263a7d5296c3f6
Related: OS#4102, OS#1545

History
#1 - 07/11/2019 05:32 PM - fixeria
- Related to Feature #1545: continuous timing advance loop using PTCCH added

#2 - 07/23/2019 06:02 PM - pespin
Consequence of not having this implemented: rx_data_fn is used to decode the PTCCH bursts, and then these errors appear all the time with osmo-bts-trx connected to an SC5 TRX:

```
20190723193916529 DTRX <000b> trx_if.c:120 Clock indication: fn=79572
20190723193916529 DSTR <0006> scheduler_trx.c:1722 TRX Clock Ind: elapsed_us=1061562, elapsed_fn=230, error_us = +112
20190723193916529 DLTR <0006> scheduler_trx.c:1790 GSM clock jitter: -4601us (elapsed_fn=0)
20190723193916900 DLP <0007> scheduler_trx.c:7981 079650/60/12/39/34 (bts=0,trx=0,ts=6) PTCCH: Received bad data (79572/12)
20190723193916901 DLP <0007> scheduler_trx.c:7981 079650/60/12/39/34 (bts=0,trx=0,ts=7) PTCCH: Received bad data (79572/12)
20190723193917381 DLP <0007> scheduler_trx.c:7981 079754/60/12/41/34 (bts=0,trx=0,ts=6) PTCCH: Received bad data (79572/12)
20190723193917382 DLP <0007> scheduler_trx.c:7981 079754/60/12/41/34 (bts=0,trx=0,ts=7) PTCCH: Received bad data (79572/12)
20190723193917591 DTRX <000b> trx_if.c:120 Clock indication: fn=79802
20190723193917591 DLTR <0006> scheduler_trx.c:1772 TRX Clock Ind: elapsed_us=1061499, elapsed_fn=230, error_us = +49
20190723193917591 DLTR <0006> scheduler_trx.c:1790 GSM clock jitter: -4584us (elapsed_fn=0)
20190723193917861 DLP <0007> scheduler_trx.c:7981 079858/60/12/43/34 (bts=0,trx=0,ts=6) PTCCH: Received bad data (79780/12)
20190723193917862 DLP <0007> scheduler_trx.c:7981 079858/60/12/43/34 (bts=0,trx=0,ts=7) PTCCH: Received bad data (79780/12)
20190723193918341 DLP <0007> scheduler_trx.c:7981 079962/60/12/45/38 (bts=0,trx=0,ts=6) PTCCH: Received bad data (79884/12)
```

03/09/2022
20190723193918341 DL1P <0007> scheduler_trx.c:981 079962/60/12/45/38 (bts=0,trx=0,ts=7) PTCCH: Received bad data (79884/12)

20190723193918652 DTRX <000b> trx_if.c:120 Clock indication: fn=80032

20190723193918652 DL1C <0006> scheduler_trx.c:1772 TRX Clock Ind: elapsed_us=1061489, elapsed_fn=230, error_us = +39

201907231939191852080066/60/12/47/38 (bts=0,trx=0,ts=6) PTCCH: Received bad data (79988/12)

201907231939191852080066/60/12/47/38 (bts=0,trx=0,ts=7) PTCCH: Received bad data (79988/12)

201907231939193002080170/60/12/49/42 (bts=0,trx=0,ts=6) PTCCH: Received bad data (80092/12)

201907231939193002080170/60/12/49/42 (bts=0,trx=0,ts=7) PTCCH: Received bad data (80092/12)

#3 - 07/23/2019 06:20 PM - fixeria

[... ] and then these errors appear all the time with osmo-bts-trx connected to an SC5 TRX [...]

I would really love to see a capture of the TRXD interface. We don't have PS Timing Advance control implemented at the moment, so I would expect nothing on PTCCH, excluding false-positive detections of course.

$ cd osmocom-bb/src/target/trx_toolkit/
$ sudo ./trx_sniff.py --direction L1 --timeslot 7 -p 5700 -o /tmp/ul_bursts.dump

#4 - 07/23/2019 06:52 PM - pespin

Running the command in that scenario provides no L1 log lines and a .dump file of 0 bytes (I checked the tool works fine on another scenario with osmo-trx).

#5 - 07/23/2019 07:01 PM - pespin

Sorry, it actually finds the packets, I was not passing the correct interface to sniff on. Once I set that correctly, .dump file is still 0 bytes, but I get lots of these messages:

[WARNING] trx_sniff.py:98 Failed to parse message, dropping...

#6 - 07/23/2019 07:06 PM - pespin

- File ptcch_errors_sc5_2.pcapng.gz added

I attach a raw pcap dump in case it's useful.

#7 - 07/23/2019 09:37 PM - fixeria

Here is a part of what I've managed to decode from your capture:

$ ./trx_sniff.py -r /tmp/ptcch_errors_sc5_2.pcap --direction L1 --timeslot 6 --frame-count 52
As far as I can see, your SC5 TRX detects pretty much everything on PDCH - just look at TDMA frame numbers. RSSI is quite low (looks like noise), while ToA256 values are not that bad...

You don't see PDTCH decoding errors because unlike rx_data_fn(), rx_pdtch_fn() uses LOGL_DEBUG:

LOGL1S(DL1P, LOGL_DEBUG, l1t, bi->tn, chan, bi->fn, "Received bad PDTCH (%u/%u)\n", ...);

LOGL1S(DL1P, LOGL_NOTICE, l1t, bi->tn, chan, bi->fn, "Received bad data (%u/%u)\n", ...);

Looks like a problem of SC5. Feel free to investigate and fill a separate bug report.

As far as I can see, your SC5 TRX detects pretty much everything on PDCH - just look at TDMA frame numbers. RSSI is quite low (looks like noise), while ToA256 values are not that bad...

You don't see PDTCH decoding errors because unlike rx_data_fn(), rx_pdtch_fn() uses LOGL_DEBUG:

LOGL1S(DL1P, LOGL_DEBUG, l1t, bi->tn, chan, bi->fn, "Received bad PDTCH (%u/%u)\n", ...);

LOGL1S(DL1P, LOGL_NOTICE, l1t, bi->tn, chan, bi->fn, "Received bad data (%u/%u)\n", ...);

Looks like a problem of SC5. Feel free to investigate and fill a separate bug report.

#8 - 10/02/2019 02:45 PM - pespin

Related patches for trxcon:
https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/15637
https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/15638

Something similar needs to be done in osmo-bts-trx.
Please see: [https://gerrit.osmocom.org/c/osmo-bts/+/15656](https://gerrit.osmocom.org/c/osmo-bts/+/15656) scheduler: fix handling of PTCCH/U and PTCCH/D logical channels. Now we need a TTCN-3 test case to verify PTCCH operation in both directions (changes for trxcon need to be merged first).


Merged.

**Files**

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Date</th>
<th>Username</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptcch_errors_sc5_2.pcapng.gz</td>
<td>440 KB</td>
<td>07/23/2019</td>
<td>pespin</td>
</tr>
</tbody>
</table>