While investigating #4586, I noticed that osmo-bts-trx never sends TRXC HANOVER command. It looks like TRXC NOHANDOVER is being sent twice.

The purpose of these TRXC commands is to control handover detection in transceiver. By default, handover detection is enabled on all inactive channels. As soon as the BSC activates a logical channel, osmo-bts-trx needs to send TRXC NOHANDOVER to the transceiver, so handover detection is disabled for that channel. As soon as a logical channel is deactivated, osmo-bts-trx needs to send TRXC HANOVER to the transceiver, so handover detection is on again.

I quickly checked the source code, and indeed there is a bug:

```c
/* setting all logical channels given attributes to active/inactive */
int trx_sched_set_lchan(struct l1sched_trx *l1t, uint8_t chan_nr, uint8_t link_id, bool active)
{
    /* Skipped code here... */

    /* disable handover detection (on deactivation) */
    if (!active)
    {
        _sched_act_rach_det(l1t, tn, ss, 0);
    }

    return rc;
}
```

Even the comment near the 'if' statement is wrong.

Related issues:
Related to OsmoBTS - Bug #4586: osmo-bts-trx leaks memory

Associated revisions
Revision f896d64e - 06/12/2020 06:55 AM - fixeria
osmo-bts-trx/scheduler: properly handle NOPE.ind during handover

Looking at GSMTAP during handover, I've noticed many packets on RACH looking pretty much like false positives, all with RA=0x00. I correlated GSMTAP traces with TRXD traces, and figure out that they all are triggered by NOPE indications from osmo-trx.
Since a NOPE.ind carries no valid burst, all its bits are set to zero. Funny enough, this sequence is still decoded just fine as a valid RACH, so that's why we see it on GSMTAP. Later on it gets rejected by L1SAP due to bad RSSI, ToA, and/or C/I ratio.

The is a side effect of [1]. In order to ensure proper Uplink measurement reporting during handover, including the time before the handover RACH is received, let's treat and handle NOPE indications as Normal Bursts.

[1] Ice45d5986610d9bcef2a7e41f0a395ec779e3928
Change-Id: Ic69f3bc2b776a23374c28a6884080a54bc16ef5f
Related: OS#4592

I quickly checked the source code, and indeed there is a bug:

Should be fixed by:

https://gerrit.osmocom.org/c/osmo-bts/+/18708 scheduler: fix_trx_sched_set_lchan(): send TRXC HANDOVER

We should not close this ticket until we have a proper (TTCN-3) testing coverage. IIRC, ttcn3-bts-test has some handover RACH test cases, but since fake_trx.py does not handle TRXC [NO]HANDOVER commands, we cannot be sure if it works with osmo-trx.

Not sure if I should be spending time on this though. laforge please assign to me if so.

#3 - 06/07/2020 02:23 PM - ipse

It's been a while since I looked into handover but shouldn't the logic be rather:
1) BSC allocates a channel to receive handover
2) BTS enables HANDOVER on this channel (i.e. is waiting for RACH)
3) Once the RACH is received and confirmed correct, BTS disables HANDOVER on this channel.

I.e. there should be HANDOVER should be enabled only for a short period of time. Otherwise, we're wasting CPU resources trying to decode RACH requests which are ignored anyway.

#4 - 06/07/2020 03:26 PM - fixeria
Well, this is how it's implemented. I am just trying to fix what we currently have. What you described would work fine for synchronized handover, but in case of non-synchronized handover you don't know on which time-/sub-slot you would get an Access Burst. Correct me if I am wrong.

#5 - 06/07/2020 06:10 PM - laforge
On Sun, Jun 07, 2020 at 09:21:35AM +0000, redmine@lists.osmocom.org wrote:

Not sure if I should be spending time on this though. laforge please assign to me if so.

How much time would you estimate?

#6 - 06/07/2020 06:50 PM - laforge
On Sun, Jun 07, 2020 at 08:18:07AM +0000, fixeria [REDMINE] wrote:

By default, handover detection is enabled on all inactive channels.

That doesn't seem to make sense to me. Why would we want to have handover detection on a channel that's not active? Doesn't that waste tons of RACH correlation resources all the time?

As soon as the BSC activates a logical channel, osmo-bts-trx needs to send TRXC NOHANDOVER to the transceiver, so handover detection is disabled for that channel. As soon as a logical channel is deactivated, osmo-bts-trx needs to send TRXC HANDOVER to the transceiver, so handover detection is on again.

Why would we want to re-enable it?

When a lchan is allocated from the Abis side, the RSL CHAN ACT carries an IE that tells us if this channel activation is handover-related or not. Based on this information, we should activate channels in OsmoTRX, and we should only enable handover detection (rach correlation) if the RSL CHAN ACT was for cause/reason == handover.

#7 - 06/07/2020 06:50 PM - laforge
On Sun, Jun 07, 2020 at 03:26:16PM +0000, fixeria [REDMINE] wrote:

[...] but shouldn't the logic be rather

Well, this is how it's implemented. I am just trying to fix what we currently have. What you described would work fine for synchronized handover, but in case of non-synchronized handover you don't know on which time-/sub-slot you would get an Access Burst. Correct me if I am wrong.

Unfortunately I think you are. The "synchronized" part only relates to whether or not we know the TA of the MS in the new cell or not. If we know it, we can do the synchronized handover (where there is no need for Access Burst)). If we don't know it (standard in OsmoBSC), the MS synchronizes to the new cell based on FCCH+SCH detection, and will then send an access burst. That burst can be anywhere within the timeslot but not outside. This is due to the fact that the access burst is very short and it will be received before the end of the timeslot (assuming the UE is within 35km of the BTS).

[1] the MS will actually have established "synchronization" to the TDMA clock of the target BTS a long time earlier during neighbor channel measurements. This cell sync state is kept around until the time we actually receive a related HO CMD, and then used when switchin to the new cell.

So I really don't think there is much point in fixing "what we have" in any other way than to align osmo-bts-trx with how we do it for the other BTSs (how it should be done)

#8 - 06/07/2020 08:40 PM - fixeria
...Correct me if I am wrong.

Unfortunately I think you are. [...]

Thanks a lot for detailed explanation and sorry for confusion.

By default, handover detection is enabled on all inactive channels.

That doesn't seem to make sense to me. [...]

I just checked the source code, and indeed it is disabled by default in osmo-trx:

```c
Transceiver::Transceiver(...) {
    /* ... */
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++)
            mHandover[i][j] = false;
    }
}
```

So I really don't think there is much point in fixing "what we have" [...]

It was looking suspicious to me that TRXC NOHANDOVER is sent so many times without prior handover activation, so I did this wrong assumption and tried to fix a problem that is not problem at all. I'll abandon that change.

Not sure if I should be spending time on this though. laforge please assign to me if so.

How much time would you estimate?

Just realized that running ttcn3-bts-test against a real BTS would (most likely) be enough to verify handover detection. As far as I can see [1], TC_ho_rach is failing. Probably because Calypso PHY does not support sending RACH on TN != 0, while trxcon does. 4h should be enough to implement the missing parts in the firmware and test against osmo-bts-trx + osmo-trx running on b210.


#9 - 06/08/2020 07:00 AM - laforge

ipse wrote:

'It's been a while since I looked into handover but shouldn't the logic be rather:
1) BSC allocates a channel to receive handover
2) BTS enables HANDOVER on this channel (i.e. is waiting for RACH)
3) Once the RACH is received and confirmed correct, BTS disables HANDOVER on this channel.

exactly. So the handover detection should only be enabled at time of RSL CHAN ACT, if the activation was handover related. And then once the dedicated channel is established (or the channel deactivated), deactivate handover detection again.

Without looking at the code, I'm quite sure this is exactly how the DSP based osmo-bts-sysmo,lc15,oc2g implement this.

#10 - 06/08/2020 07:01 AM - laforge

- Status changed from Feedback to In Progress
- Assignee set to fixeria

#11 - 06/08/2020 10:12 PM - fixeria

As it turned out, sending RACH.req to a phone in dedicated mode crashes the DSP:
I've managed to hack the firmware, so it is capable to send handover RACH on TS1..7 (not only on TS0) in idle mode, but this is still not a proper solution. Ideally we should get (at least some) changes from jolly/handover, or rather laforge/jolly_handover_rebased, merged. Unfortunately, I underestimated the time, but at least I found and fixed a regression in osmo-bts-trx:

https://gerrit.osmocom.org/c/osmo-bts/+/18734 scheduler: fix trx_sched_ul_burst(): ignore NOPE.ind during handover

#12 - 06/08/2020 10:15 PM - fixeria
- % Done changed from 30 to 60

I've managed to hack the firmware, so it is capable to send handover RACH on TS1..7 (not only on TS0) in idle mode […]

Forgot to mention that handover detection in osmo-trx + osmo-bts-trx seems to work fine.

#13 - 06/08/2020 11:11 PM - ipse

fixeria wrote:

I've managed to hack the firmware, so it is capable to send handover RACH on TS1..7 (not only on TS0) in idle mode […]

Forgot to mention that handover detection in osmo-trx + osmo-bts-trx seems to work fine.

Just curious - have you figured why are we sending duplicated NOHANDOVER as well?

#14 - 06/10/2020 02:16 PM - fixeria

ipse wrote:
Just curious - have you figured why are we sending duplicated NOHANDOVER as well?

Yep, I think I found the reason why it's being sent twice. The culprit is in src/osmo-bts-trx/l1_if.c, bts_model_l1sap_down(), where we first deactivate SACCH (so it triggers sending of NOHANDOVER), and then we deactivate DCCH, so it triggers sending of NOHANDOVER again.

Also, some interesting notes on why are we sending it regardless of the previous state:

```c
int trx_sched_set_mode(struct l1sched_trx *l1t, uint8_t chan_nr, uint8_t rsl_cmode,
                       uint8_t tch_mode, int codecs, uint8_t codec0, uint8_t codec1,
                       uint8_t codec2, uint8_t codec3, uint8_t initial_id, uint8_t handover)
{
    /* command rach detection
     * always enable handover, even if state is still set (due to loss
     * of transceiver link).
     * disable handover, if state is still set, since we might not know
     * the actual state of transceiver (due to loss of link) */
    _sched_act_rach_det(l1t, tn, ss, handover);

    return rc;
}
```

I compiled the firmware from laforge/jolly_handover_rebased, and it worked better than my hack. But still not without problems:

- sending handover RACH on TCH/F seems to work fine,
- sending handover RACH on TCH/H does not work (despite it should [1]),
- sending handover RACH on SDCCH/4 and SDCCH/8 does not work (not implemented).

[1] https://git.osmocom.org/osmocom-bb/commit/?h=laforge/jolly_handover_rebased&id=854e526de205de1493770b91c27e4d0a65a6173

I think this is mostly esoteric. The general opinion is that you don't do hand-over on a SDCCH, as it is very short lived and the MS can just as well establish a new SDCCH in the new cell after autonomous re-selection.

In OsmoBSC, we don't implement this either (for exactly that reason).

laforge wrote:

- sending handover RACH on TCH/H does not work (despite it should [1]),

this is sad and I think we do need it. Any ideas?

The way it's implemented in jolly/handover looks correct to me.
if (l1s.dedicated.type == GSM_DCHAN_TCH_F) {
    fn_sched = l1s.current_time.fn + offset;
    /* go next DCCH frame TCH/F channel */
    if (((fn_sched % 13) == 12))
        fn_sched++;
} else if (l1s.dedicated.type == GSM_DCHAN_TCH_H) {
    fn_sched = l1s.current_time.fn + offset;
    /* go next DCCH frame of TCH/H channel */
    if (((fn_sched % 13) == 12))
        fn_sched++;
    if ((l1s.dedicated.chan_n & 1) != ((fn_sched % 13) & 1))
        fn_sched++;
} else if (combined) { /* ... */ }

OsmoBSC has some special VTY commands to trigger handover manually, so I am going to test with a normal phone.

#18 - 06/11/2020 01:01 PM - fixeria
- Subject changed from osmo-bts-trx: handover detection control is broken to osmo-bts-trx: make sure that handover detection works

#19 - 06/11/2020 02:39 PM - ipse

laforge wrote:

On Wed, Jun 10, 2020 at 02:27:26PM +0000, fixeria [REDMINE] wrote:

  ● sending handover RACH on SDCCH/4 and SDCCH/8 does not work (not implemented).

I think this is mostly esoteric. The general opinion is that you don't do hand-over on a SDCCH, as it is very short lived and the MS can just as well establish a new SDCCH in the new cell after autonomous re-selection.

In OsmoBSC, we don't implement this either (for exactly that reason).

I think this is important for USSD menus which are going over SDCCH and might be quite long living. USSD menus are commonly used for customer self-service and mobile money transfer around the world. I personally used USSD menus to top up my balance while sitting in a taxi which would not be possible without handover. So I wouldn't call this use case esoteric.

#20 - 06/11/2020 03:07 PM - laforge

ipse wrote:

I think this is important for USSD menus which are going over SDCCH and might be quite long living. USSD menus are commonly used for customer self-service and mobile money transfer around the world. I personally used USSD menus to top up my balance while sitting in a taxi which would not be possible without handover. So I wouldn't call this use case esoteric.

So you are saying that you know of cellular networks that use handover for SDDCH in production?

I know those USSD applications exist, I just find many sources that claim handover for SDCCH is simply not enabled in production networks in general.

#21 - 06/11/2020 04:37 PM - fixeria
- % Done changed from 60 to 80

OsmoBSC has some special VTY commands to trigger handover manually, so I am going to test with a normal phone.

It took me a while to prepare the setup and properly configure osmo-bsc, but here are the good news:

  ● handover detection on TCH/H works
  ◆ sub-slot 0 -> sub-slot 1
  ◆ and vice versa
  ● handover detection on SDCCH/8 works

The testing algorithm is:
Make sure to configure handover for BTS#0:

```
network
  ... 
  bts 0
    ...
    handover 1
    handover algorithm 2
```

The bad news is that \textit{jolly/handover} needs more work before it can be used. Alternatively, automated testing can be done using an ofono supported modem, given that both silent call and handover is initiated by the network. I don't have time to work on this anymore, given that I already spent more than was estimated.

\textbf{#22 - 06/11/2020 06:09 PM - ipse}

ipse wrote:

```
ipse wrote:

  I think this is important for USSD menus which are going over SDCCH and might be quite long living. USSD menus are commonly used for customer self-service and mobile money transfer around the world. I personally used USSD menus to top up my balance while sitting in a taxi which would not be possible without handover. So I wouldn't call this use case esoteric.
```

So you are saying that you know of cellular networks that use handover for SDDCH in production?

I think I saw this but I can't be sure right now. I would need to go back there and test with some test tool :)

  I know those USSD applications exist, I just find many sources that claim handover for SDCCH is simply not enabled in production networks in general.

Is there any reason why it's disabled? Is it much more difficult than normal handover?

\textbf{#23 - 06/21/2020 03:10 PM - laforge}

- Priority changed from Normal to Low