baseband frequency hopping support for osmo-bts-trx

The idea of baseband frequency hopping is to "commutate" the bursts among the available TRX. So we have a BTS with e.g. 4 TRX. TRX0 carries the CCCH and hence cannot do hopping. However, TRX1 to TRX3 can implement baseband hopping. This works by having each of the TRX on a fixed ARFCN (no tuning or actual hopping of the carrier frequency). The trick is to then assign dedicated channels (TCH + SDCCH) not on one ARFCN but on a hopping sequence that "hops" between those three ARFCN, basically jumping to a different ARFCN for every burst. This feature would be implemented in osmo-bts. It has four separate streams of bursts to/from the four TRX in the example above, and it needs to re-order them according to the hopping sequences before passing them into the convolutional decoder.

Revision 63bee454 - 05/15/2020 05:56 PM - tnt
libosmogsm: import hopping sequence generation code

This implementation is taken from OsmocomBB, in particular from:

target/firmware/layer1/rfch.c

Change return type to uint16_t, because neither ARFCN, nor MAI can be negative. Add prefix 'gsm0502_' to the function's name.

Change-Id: I8aba1578cc9d1bd89d4f5d33a6e8f6dc8bea789a
Related: OS#4546

Revision 830b8dbb - 05/15/2020 05:56 PM - laforge
libosmogsm: add Doxygen docs for gsm0502_hop_seq_gen()

This is basically I16d5190b3cddc97c5609b52d41203f10264b017c.

Change-Id: Ife660f00d7a6d3b874ab2e59e34dca2109fa82d8
Related: OS#4546

Revision c5f1a673 - 05/28/2020 10:31 AM - fixeria
ttcn3-bts-test: enable 3 additional transceivers for BTS#0

This is needed for the forthcoming frequency hopping test cases.

Change-Id: lb5ad31388ae25399ad09739aa3f3f0cb0b3a1f78b
Depends: I86afb55ecc6703ce7a229aaa6262239f93314778
Related: OS#4546

Revision 842e4fab - 06/29/2020 08:51 AM - fixeria
A-bis/OML: handle hopping params in Set Channel Attributes
3GPP TS 12.21 V8.0.0 (OML)

8.6.2 Set Radio Carrier Attribute -> contains ARFCN list (param info forwarded to TS 05.05)
8.6.3 Set Channel Attributes -> contains HSN + MAIO + ARFCN list (param info forwarded to TS 05.02)

3GPP TS 05.02 V8.11.0

Most related required information can be found in this document.

5.4 Radio frequency channel sequence

The radio frequency channel sequence is determined by a function that, in a given cell, with a given set of general parameters, (see subclause 5.6.2), with a given timeslot number (TN), a given mobile radio frequency channel allocation (MA) and a given mobile allocation index offset (MAIO), maps the TDMA frame number (FN) to a radio frequency channel.

In a given cell there is therefore, for a physical channel assigned to a particular mobile, a unique correspondence between radio frequency channel and TDMA frame number.

The detailed hopping generation algorithm is given in subclause 6.2.

5.6 Parameters for channel definition and assignment

summary: Per TimeSlot we need to add this information (Already in osmo-bsc VTY. it goes BSC->BTS through OML):

- TSC (we already have it, "training_sequence_code")
- MA (mobile radio frequency channel allocation) (the ARFCN list afaiu, TS 05.02 "6.2.2 Parameters" says "The MA contains N radio frequency channels, where 1<=N<=64". "hopping arfcn add"
- MAIO (mobile allocation index offset) "hopping maio"
- HSN (hopping sequence number) "hopping sequence-number"

6 Mapping of logical channels onto physical channels

This explains all the mappings that we need to add to osmo-bts scheduler as a new abstraction layer level.

6.2.3 "Hopping sequence generation"

This section explains how to get the ARFCN based on input params (MAI, N=|MA|, FN, MAIO, HSN. (see "Figure 6" for graphical input/output params).

6.2.4 Specific cases

frequency hopping is not permitted on any timeslot supporting a BCCH
according to table 3 of clause 7. A non-hopping radio frequency channel sequence is characterized by a mobile allocation consisting of only one radio frequency channel, i.e. with N=1, MAIO=0. In this instance sequence generation is unaffected by the value of the value HSN.

"Figure 1" is quite useful to understand the mapping in different layer levels.
"Figure 4", "Figure 5", "Figure 6" shows hoping in action.

Current Status Summary:

- Configuration of hopping params supported in osmo-bsc, as well as sending values over OML (afaui because we supported that in the past with an older non-osmo-bts BTS)
- Only initial minimal retrieving implemented in osmo-bts (oml.c), but hopping not supported in general.

Steps:

- osmo-bts:
  - It lacks most of the hopping support. OML messages take the params but do nothing (eg oml.c oml_rx_set_radio_attr and
Towards osmo-trx, we probably need to enable HANDOVER on all ARFCNs in MA group (ARFCN list) for that TS? it also means we need some ref counting structure since several channels may require it to be set.

Also in osmo-trx, we specifically set each TS with its type. TODO: checked what needs to be done there regarding this.

Update in lots of placed (grep "hopping") that hopping is not implemented in osmo-bts.

**scheduler:**
- Downlink (Tx): trx_sched_fn in scheduler_trx.c calls sched_rts() and sched_dl_burst() for each timeslot, but when using frequency hoping then this would be wrong, we need to somehow convert arfcn + fn => ts, and so far AFAIU the spec only provides functions to go the opposite path...
  We could perhaps go this way and simply end up sending on a different arfcn for which it was polled?

**TODO:**
- Channel assignment parameters (RSL? 08.58? already done by BSC?)
- Find out how to infer TS from arfcn + fn (UL or RTS), and other scheduler required changes.
- Pass information to PCU

#2 - 05/12/2020 01:26 PM - pespin

**Regarding operational sample configuration:**

The idea is to set, for the same TS number on all TRXs [len(TRXs)=N], the same HSN (Hopping sequence Number) and a different MAIO (0..N-1)

So for instance this would be possible:

```plaintext
trx 0
  timeslot 0
    phys_chan_config TCH/H
    hopping enabled 1
    hopping sequence-number 1
    hopping maio 1
    hopping arfcn add 870
    hopping arfcn add 872

  timeslot 1
    phys_chan_config TCH/H
    hopping enabled 1
    hopping sequence-number 2
    hopping maio 1
    hopping arfcn add 870
    hopping arfcn add 872

  timeslot 2
    phys_chan_config TCH/H
    hopping enabled 1
    hopping sequence-number 3
    hopping maio 1
    hopping arfcn add 870
    hopping arfcn add 872

  timeslot 3
    phys_chan_config TCH/F
    hopping enabled 1
    hopping sequence-number 4
    hopping maio 1
    hopping arfcn add 870
    hopping arfcn add 872

  timeslot 4
    phys_chan_config TCH/F
    hopping enabled 1
    hopping sequence-number 5
    hopping maio 1
    hopping arfcn add 870
    hopping arfcn add 872

  timeslot 5
    phys_chan_config TCH/F
    hopping enabled 1
    hopping sequence-number 6
    hopping maio 1
    hopping arfcn add 870
    hopping arfcn add 872

  timeslot 6
    phys_chan_config PDCH
    hopping enabled 1
    hopping sequence-number 7
```
hopping maio 1
hopping arfcn add 870
hopping arfcn add 872
timeslot 7
phys_chan_config PDCH
hopping enabled 1
hopping sequence-number 8
hopping maio 1
hopping arfcn add 870
hopping arfcn add 872
trx 1
timeslot 0
phys_chan_config TCH/H
hopping enabled 1
hopping sequence-number 1
hopping maio 2
hopping arfcn add 870
hopping arfcn add 872
timeslot 1
phys_chan_config TCH/H
hopping enabled 1
hopping sequence-number 2
hopping maio 2
hopping arfcn add 870
hopping arfcn add 872
timeslot 2
phys_chan_config TCH/H
hopping enabled 1
hopping sequence-number 3
hopping maio 2
hopping arfcn add 870
hopping arfcn add 872
timeslot 3
phys_chan_config TCH/F
hopping enabled 1
hopping sequence-number 4
hopping maio 2
hopping arfcn add 870
hopping arfcn add 872
timeslot 4
phys_chan_config TCH/F
hopping enabled 1
hopping sequence-number 5
hopping maio 2
hopping arfcn add 870
hopping arfcn add 872
timeslot 5
phys_chan_config TCH/F
hopping enabled 1
hopping sequence-number 6
hopping maio 2
hopping arfcn add 870
hopping arfcn add 872
timeslot 6
phys_chan_config PDCH
hopping enabled 1
hopping sequence-number 7
hopping maio 2
hopping arfcn add 870
hopping arfcn add 872
timeslot 7
phys_chan_config PDCH
hopping enabled 1
hopping sequence-number 8
hopping maio 2
hopping arfcn add 870
hopping arfcn add 872

Regarding RSL CHan ACT

Check TS 08.58 sec "9.3.5 Channel Identification" IE, where it points to TS 0408:

04.08 "Channel Description" *
04.08 "Mobile Allocation" *
A * denotes that the whole of the 04.08 element including the element identifier and length should be included. The 04.08 "Mobile Allocation" shall for compatibility reasons be included but empty, i.e. the length shall be zero.

From there, TS 04.08 sec "10.5.2.5 Channel Description" explains how to either encode the ARFCN or the MAIO + HSN. Similarly, "10.5.2.21 Mobile Allocation" IE encodes an ARFCN List as a bitmap of 1 and 0. It, however, must be always empty according to spec TS 08.58.

This is already implemented in osmo-bssc.git abis_rsl.c rsl_tx_chan_activ() as pointed out by Harald. On the other hand, in osmo-bts.git, we are not even parsing RSL_IE_CHAN_IDENT in rsl.c rsl_rx_chan_activ(), which means nowadays we are not even verifying that the ARFCN in there fine (we probably assume the ARFCN for the trx associated to the rsl connection is used).

We also need to make sure we send the CA (list of arfcns) in BCCH in the BTS:

If frequency hopping is applied, the cell allocation if present in the message is used to decode the mobile all ocation. If the cell allocation is not included, the mobile station uses its current cell allocation, the current cell is the last CA received on the BCCH. Afterward, the current CA may be changed by some messages sent on the main signalling link containing a CA (the possible messages are: ASSIGNMENT COMMAND, HANDOVER COMMAND and FREQUENCY REDEFINITION). Note that there are cases in which the current CA is undefined, see sub-clause 3.4.3.3.

Regarding PCU

Taking over from what Harald did look at, on osmo-pcu.git we need to:

- pass Mobile Allocation bitmap over the PCU socket (only if it turns out we need to implement "Frequency Parameters" for indirect mode in "Immediate assignment", see further below). We actually don't need to add this anyway because it's already being sent to BSC->BTS->PCU through System Information 13 element "GPRS Mobile Allocation : GPRS Mobile Allocation IE > } -- Defined in 3GPP TS 04.6012.10a GPRS Mobile Allocation"
- Pass per-pdch TS (MAIO, HSN) parameters over the PCU socket so the PCU knows about them: MAIO, HSN. We could try encoding it inside 16bit arfcn (we need 6 bit for MAIO and 6 bit for HSN, so 12 of 16 total, plus some way to differentiate between hopping enabled or disabled). It needs to be done at least for info_ind, when in osmo-pcu arfcn is copied and later used. Other usages look more like gsmtap related.
- encode the hopping parameters in any kind of RLC/MAC messages, specifically
  - Encoding::write_immediate_assignment()
  - Encoding::write_packet_uplink_assignment()
  - Encoding::write_packet_downlink_assignment()

TS 04.08:

- "10.5.2.16 IA Rest Octet" Included in "IMMEDIATE ASSIGNMENT". Contains struct "Frequency Parameters", which in turn contains MAIO and Mobile Allocation. We don't seem to need this though, since we only send Downlink and Uplink options which don't contain those.
- "10.5.2.25a Packet Channel Description": Included in "IMMEDIATE ASSIGNMENT". It has a bit to encode either ARFCN or MAIO + HSN. In there, there seems to be "Direct" vs "Indirect" "encoding of hopping RF channel configuration", the first providing MAIO+HSN and the other one MAIO+MA_NUMBER_IND. We probably want to use direct since we have MAIO + HSN from the BTS anyway).

Summary of steps:

- Move osmocom-bb hopping functions to libosmocore
- osmo-bts.git: Rework scheduler-trx (plus generic scheduler structures) to decouple 1 trx = 1 specific rfch. Use libosmocore rfch functions in in schedueler-trxto convert rfch<->trx and pass logical trx to generic scheduler. (we may want to add some generic code in the scheduler to do so).
- pcuif: Find way to include MAIO+HSN (12bit) into uint16_t to avoid changing protocol. Otherwise, change structs to accomodate MAIO+HSN. TRX in pcuinf continues to work the same since they are considered logical TRXs.
- osmo-pcu.git: Write MAIO+HSN in write_assignment if hopping is enabled
- faketrx: We may have to add support for hopping in faketrx to test with TTCN3

#3 - 05/12/2020 01:26 PM - laforge

pespin: Let me illustrate what happens in the following tables.

Assumptions:

- 5-TRX BTS (C0..C4 each represent one ARFCN)
- BCCH on TRX0/TS0
- sequential hopping sequence (for illustration, works for any other sequence)
- b = BCCH/CCCH
- A..E = logical timeslots (think of TCH/F or PDCH)

TS0
I would like to follow test-driven development approach here, so my current plan is:

1. Implement freq. hopping support for the MS side of ttcn3-bts-test:
   - trxcon remains unchanged and still maintains a single TRXC/TRXD connection
   - on TRXC we already have an MS specific command: `SETFH <HSN> <MAIO> <CH1> <CH2> [... <CHN>]`
   - `fake_trx.py` gets the actual implementation of freq. hopping
     - hopping sequence is generated from parameters indicated by `SETFH`
     - in Transceiver/FakeTRX every burst coming from the BurstForwarder triggers freq. change
   - unit tests!
   - `BTS_Tests.f_est_dchan()` must be extended:
     - to be able to send `RSL_MT_IMMEDIATE_ASSGN_CMD` with hopping parameters
     - to be able to send `L1CTL DM_EST.req` with hopping parameters

2. Implement test cases as a part of ttcn3-bts-test:
   - 2.0. Investigate what needs to be changed or refactored in order to indicate freq. hopping parameters to osmo-bts-trx (IUT)
     - osmo-bts.cfg in docker-playground needs to be changed, so osmo-bts-trx knows about all additional transceivers
     - osmo-bsc.cfg in docker-playground needs to be changed, so osmo-bsc can OML-bootstrap all transceivers for us
     - `fake_trx.py` needs to know about all additional transceivers too (by default we have 1 MS + 1 BTS only)
   - 2.1. Make sure that L2 frames on different kinds of logical channel (SDCCH, TCH, PDCH) can be sent and received when freq. hopping is in use
   - 2.2. Make sure that several channels with and without freq. hopping can co-exist together, not interfering with each other
   - 2.3. Make sure that BCCH on C0/TRX0 can still be decoded when at least one dedicated channel is hopping
   - 2.4. Make sure that CBCH on SDCCH/8 can still be decoded when SI4 indicates that it
     - `FIXME: record GSM_RR_Types.Mai0Hsn` is empty o_O
   - 2.5. Make sure that L1CTL and L2 frames can still be sent and received when freq. hopping is in use

3. Finally, implement baseband freq. hopping support in osmo-bts-trx:
   - 3.0. Illustrate the current bts/trx/ts/lchan dependency diagram, investigate what needs to be changed
     - obviously we need to decouple both logical channel state and tx/ts state
     - logical channel handlers (tx_data_fn, tx_tchf_fn, etc.) shall have no access to anything else than their associated state
     - `this is how it's (almost) done in txcon, but still: *tx is needed to know TSC for Tx bursts, *ts provides multiframe layout info...`
     - we can just store what's needed in the logical channel state
   - 3.1. Implement handling of the hopping related parameters in the OML messages
   - 3.2. Implement the actual code for dispatching bursts to and from the transceivers
   - 3.3. Indicate support of freq. hopping to the BSC among with the other features
Some more additional notes:

- Implement freq. hopping support for the MS side of ttcn3-bts-test:
  - Calypso PHY does not require any changes and can be also used in the testing setup
- Finally, implement baseband freq. hopping support in osmo-bts-trx:
  - We already have hopping sequence generation code in OsmocomBB, makes sense to move it to libosmocore

3.5. We already have hopping sequence generation code in OsmocomBB, makes sense to move it to libosmocore

https://gerrit.osmocom.org/c/libosmocore/+/18244 libosmogsm: import hopping sequence generation code
https://gerrit.osmocom.org/c/libosmocore/+/18245 libosmogsm: add Doxygen docs for gsm0502_hop_seq_gen()

fake_trx.py gets the actual implementation of freq. hopping

This part is pretty much done, but unit tests are still missing (low priority).

hopping sequence is generated from parameters indicated by SETFH

https://gerrit.osmocom.org/c/osmocom-bb/+/18261/  trx_toolkit/gsm_shared.py: implement hopping sequence generation

on TRXC we already have an MS specific command: SETFH <HSN> <MAIO> <CH1> <CH2> [... <CHN>]

I had to change the format of this command, so we send a list of frequencies instead of ARFCNs:

https://gerrit.osmocom.org/c/osmocom-bb/+/18319/  trxcon: refactor trx_if_cmd_setfh(): send Rx/Tx frequencies

in Transceiver/FakeTRX every burst coming from the BurstForwarder triggers freq. change

https://gerrit.osmocom.org/c/osmocom-bb/+/18262/  trx_toolkit/transceiver.py: add frequency hopping support

I am starting to work on TTCN-3 test cases now.

- % Done changed from 0 to 10

All changes mentioned above have been merged.

I am starting to work on TTCN-3 test cases now.

I have prepared a multi-TRX configuration for ttcn3-bts-test:
https://git.osmocom.org/docker-playground/commit/?h=fixeria/mtrx&id=8dd2bf5cedeab3d1f9c35388918bf1eba8136ef41. Unfortunately, the existing test cases fail when running against the new setup.

The problem is that in the new multi-TRX setup we have more than one IPA/RSL connection, because every transceiver establishes it's own RSL link to the same IP address / port of the test case (virtual BSC). This works with osmo-bsc, but not with our IPA/RSL emulation components. As it turns out, IPA_Emulation_CT was not designed to serve more than one IPA connection, so it overwrites its g_ipa_conn_id on every incoming connection. In the end, all messages are getting sent to TRX3 instead of TRX0, because it was the last connection.
The main question is how do we map one of multiple IPA/RSL connections to the exact trx number? Some options: a) the transceivers always connect one after another, starting from trx#0, so we can assume that every new connection belongs to trx # (N+1)
b) the RSL_Emulation component can be modified to ask IPA UnitID of each transceiver, so we can parse trx#N from response: _/_/N
c) get rid of osmo-bsc and do the OML bootstrapping from the test case (probably, a lot of work)
d) get rid of osmo-bsc and do the OML bootstrapping from the test case (probably, a lot of work)

I implemented b), and finally got the existing TTCN-3 test cases working against a multi-trx setup. Yay '/-. All changes have been submitted to Gerrit, some of them have already been merged. The most important ones are:

- library/RSL_Emulation: fixup: handle RSL messages from any TRX
- library/IPA_Emulation: server mode: handle multiple client connections
- library/IPA_Emulation: handle optional conn_id in ASP_RSL_Unitdata
- library/IPA_Emulation: server mode: also request IPA UnitID
- library/IPA_Emulation: server mode: expose IPA IDENTITY RESPONSE
- library/RSL_Emulation: server mode: handle multiple transceivers

P.S. This also means that we can write test cases for multi-trx setup, not only for frequency hopping.

FIXME: record GSM_RR_Types.MaioHsn is empty o_O

BTS_Tests.f_est_dchan() must be extended:

- to be able to send RSL MT_IMMEDIATE_ASSIGN_CMD with hopping parameters
- to be able to send L1CTL DM_EST.req with hopping parameters

2.0. Investigate what needs to be changed or refactored in order to indicate freq. hopping parameters to osmo-bts-trx (IUT)

- osmo-bts.cfg in docker-playground needs to be changed, so osmo-bts-trx knows about all additional transceivers
- osmo-bsc.cfg in docker-playground needs to be changed, so osmo-bsc can OML-bootstraps all transceivers for us

- fake_trx.py needs to know about all additional transceivers too (by default we have 1 MS + 1 BTS only)

- ttcn3-bts-test: enable 3 additional transceivers for BTS#0
While working on the first test case, I found and fixed a bug in trxcon:

https://gerrit.osmocom.org/c/osmocom-bb/+/18519/  trxcon: fix l1ctl_proc_est_req_h0(): convert to host byte order

so here is a "demo" test case, and of course it fails at the moment (as expected):


In general, I think we can introduce "hopping" variants of the existing test cases, at least some of them.

#15 - 05/29/2020 06:30 PM - fixeria

I just noticed that osmo-bsc does not actually distinguish different osmo-bts-{trx,sysmo,...} variants. Instead, they all are treated as "sysmobts", so they all share same set of features in osmo-bsc. We're about to introduce frequency hopping support to osmo-bts-trx, but the other variants will unlikely get it any time soon, probably never. I think we need to implement a proper handling of osmo-bts-{trx,sysmo,...} variants in osmo-bsc, so they all can have separate feature vectors.

While reading the code, I also discovered that the "public" osmo-bts features that are reported via OML were mixed together with the internal implementation specific features, so they are reported to the BSC all together. This is definitely a bad idea, so I decided to separate both feature sets:

https://gerrit.osmocom.org/c/osmo-bts/+/18591  Do not mix public and private BTS features, use libosmocore's API

and found a minor bug:

https://gerrit.osmocom.org/c/osmo-bts/+/18592  oml: fix TL16V length calculation in add_bts_feat()

#16 - 06/06/2020 12:14 PM - fixeria

- File osmo-bts-trx.dot added

So this is the actual dependency diagram for the current code:

  "/usr/src/redmine/plugins/wiki_graphviz_plugin/app/views"  *
  "/usr/src/redmine/plugins/redmine_tags/app/views"  *
  "/usr/src/redmine/plugins/redmine_openid_provider/app/views"  *
  "/usr/src/redmine/plugins/redmine_mentions/app/views"  *
  "/usr/src/redmine/plugins/redmine_lightbox2/app/views"  *
  "/usr/src/redmine/plugins/redmine_checklists/app/views"  *
  "/usr/src/redmine/plugins/redmine_banner/app/views"  *
  "/usr/src/redmine/plugins/clipboard_image_paste/app/views"  *
  "/usr/local/bundle/gems/redmine_crm-0.0.55/app/views"

As can be seen, osmo-bts-trx specific structures live separately from generic osmo-bts structures. Only l1sched_trx has a pointer to gsm_bts_trx. Another important point is that osmo-bts-trx maintains individual schedulers for each transceiver. This is critical for the upcoming frequency hopping implementation. Ideally we should have one scheduler per gsm_bts that would serve all child transceivers. I'll propose another graph for the new architecture soon.

#17 - 06/06/2020 07:36 PM - laforge

Thanks for the diagram!

My naive thought would have been that all of this can stay the same? After all, the state, including the burst bufer[s] and the scheduler are referring to a logical TRX. The actual permutation of bursts between radio carriers and logical TRX would have to be even one level lower, right?

#18 - 06/07/2020 06:52 PM - pespin

Agree with laforge , that was also my conclusion after looking at the code.

#19 - 06/11/2020 05:16 PM - fixeria

My naive thought would have been that all of this can stay the same?

Mostly, yes, I am not planning to refactor everything ;) However we still need some important changes:

- There should be one scheduler per BTS, not per TRX. All child transceivers of a BTS share the same clock source anyway. Otherwise every 'per TRX scheduler' would try to dequeue packets from per timeslot Downlink queues.

03/08/2022
- Logical channel handlers should not operate on TRX (struct l1sched_trx), but on the associated logical channel state (struct l1sched_chan_state).

I am also splitting logical channel handlers into separate files, because currently it's hard to find e.g. xCCH handler in a huge file.

#20 - 06/29/2020 05:46 AM - fixeria

- Checklist item [x] handle/parse hopping channels on A-bis OML + RSL (common part) set to Done
- Checklist item [x] commutator between "gsm_bts_trx" and "radio carrier" doing the actual hopping set to Done
- % Done changed from 30 to 80

During the last couple of weeks I was distracted by some unrelated tasks (see #4619, #3032, #2978), but now I finally got enough time to move on.

There should be one scheduler per BTS, not per TRX. All child transceivers of a BTS share the same clock source anyway. Otherwise every ‘per TRX scheduler’ would try to dequeue packets from per timeslot Downlink queues.

As it turned out, I was wrong. Sorry for confusion. We actually have one scheduler per BTS instance.

Logical channel handlers should not operate on TRX (struct l1sched_trx), but on the associated logical channel state (struct l1sched_chan_state).

I decided to keep this as is. Having access to struct l1sched_trx in the logical channel handlers allows them to print more context information in the logging messages.

Last week I continued to work on this task, found an interesting problem in the scheduler (already merged):

https://gerrit.osmocom.org/c/osmo-bts/+/19009
osmo-bts-trx: fix trx_sched_fn(): properly advance frame number

and finally submitted the actual frequency hopping implementation to Gerrit:

https://gerrit.osmocom.org/c/osmo-bts/+/19028
https://gerrit.osmocom.org/c/osmo-bts/+/19029
https://gerrit.osmocom.org/c/osmo-bts/+/19030
osmo-bts-trx/scheduler: cosmetic: move trx_if_powered() check
osmo-bts-trx/scheduler: refactor dummy burst scheduling
osmo-bts-trx/scheduler: implement baseband frequency hopping

I did some testing with the real hardware (USRP B210 in multi-AFCN mode, 2 TRX) and a Nokia phone:

```
trx 0
    arfcn 801
    ...
    timeslot 0
        phys_chan_config CCCH
        hopping enabled 0
    timeslot 1
        phys_chan_config SDCCH8
        hopping enabled 1
        hopping sequence-number 1
        hopping maio 0
        hopping arfcn add 801
        hopping arfcn add 805
    timeslot 2
        phys_chan_config TCH/F
        hopping enabled 1
        hopping sequence-number 0
        hopping maio 0
        hopping arfcn add 801
        hopping arfcn add 805
    timeslot 3
        phys_chan_config TCH/F
        hopping enabled 1
        hopping sequence-number 0
        hopping maio 0
        hopping arfcn add 801
        hopping arfcn add 805
    ...

trx 1
    arfcn 805
    ...
    timeslot 0
        phys_chan_config TCH/H
        hopping enabled 0
    timeslot 1
```
Hopping SDCCH/8 on TS1 works fine (tested Location Updating, USSD, SMS, Silent Call). Hopping TCH/F and TCH/H on TS2..7 still to be tested. Unfortunately, OsmoBSC fails to encode *RR Channel Assignment* properly, so the phone rejects this message by sending *RR Status* (cause=Conditional IE Error).

Since TRX0 is C0 (ARFCN 801), I also checked the spectrum to make sure that dummy bursts are sent as expected and there are no gaps. Everything looks fine.

#21 - 06/29/2020 05:55 AM - fixeria
- Checklist item deleted (separate logical TRX from "radio carriers" in our data structures)

[] separate logical TRX from "radio carriers" in our data structures

I don't think we really need this. The existing object hierarchy in osmo-bts-trx is flexible enough. Burst "routing" in the proposed implementation is done by the scheduler itself.

#22 - 07/08/2020 11:50 AM - fixeria
- Blocked by Bug #4658: Wrong burst order in a multi-trx setup added

#23 - 07/14/2020 10:51 AM - fixeria
- Checklist item [x] automatic TTCN3 testing of frequency hopping configurations set to Done
- % Done changed from 80 to 90

I have submitted a set of changes:

https://gerrit.osmocom.org/c/docker-playground/+/19159  ttcn3-bts-test/osmo-bsc.cfg: change pchan config for TRX#1..3
https://gerrit.osmocom.org/c/docker-playground/+/19160  ttcn3-bts-test: add configuration with frequency hopping parameters

https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/19244  BTS_Tests: introduce and use helper f_l1ctl_est_dchan()
https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/19245  BTS_Tests: implement optional frequency hopping support

The existing test cases implicitly cover the following scenarios:

2.1. Make sure that L2 frames on different kinds of logical channel (SDCCH, TCH, PDCH) can be sent and received when freq. hopping is in use
2.2. Make sure that several channels with and without freq. hopping can co-exist together, not interfering with each other
2.3. Make sure that BCCH on C0/TRX0 can still be decoded when at least one dedicated channel is hopping

excluding this one (we don't seems to have any test cases for SDCCH8+CBCH at all):

2.4. Make sure that CBCH on SDCCH/8 can still be decoded when SI4 indicates that it
Running the existing test cases against a multi-trx setup with frequency hopping enabled locally shows the following regressions:

- pass->FAIL BTS_Tests.TC_meas_res_sign_tchf
- pass->FAIL BTS_Tests.TC_meas_res_sign_sdcch8
- pass->FAIL BTS_Tests.TC_rsl_ms_pwr_ctrl
- pass->FAIL BTS_Tests.TC_rsl_ms_pwr_dyn_up
- pass->FAIL BTS_Tests.TC_paging_lmsi_200percent
- pass->FAIL BTS_Tests.TC_paging_lmsi_200percent
- pass->FAIL BTS_Tests.TC_SI_sched_2Quater
- pass->FAIL BTS_Tests.TC_SI_sched_13_2bis_2ter_2quater
- pass->FAIL BTS_Tests.TC_pcu_time_ind
- pass->FAIL BTS_Tests.TC_pcu_ptcch
- pass->FAIL BTS_Tests.SMSCB.TC_sms_cb_cmd_sdcch4_1block
- pass->FAIL BTS_Tests.SMSCB.TC_sms_cb_cmd_sdcch4_2block
- pass->FAIL BTS_Tests.SMSCB.TC_sms_cb_cmd_sdcch4_3block
- pass->FAIL BTS_Tests.SMSCB.TC_sms_cb_cmd_sdcch4_4block
- pass->FAIL BTS_Tests.SMSCB.TC_sms_cb_cmd_sdcch4_multi
- pass->FAIL BTS_Tests.SMSCB.TC_sms_cb_cmd_sdcch4_schedule
- pass->FAIL BTS_Tests.SMSCB.TC_sms_cb_cmd_sdcch4_default_and_normal
- pass->FAIL BTS_Tests.SMSCB.TC_cbc_sdcch4_load_overload
- pass->FAIL BTS_Tests.LAPDm.TC_sabm_ua_dcch_sapi0
- pass->FAIL BTS_Tests.LAPDm.TC_sabm_retransmit
- pass->FAIL BTS_Tests.LAPDm.TC_sabm_invalid_resp
- pass->FAIL BTS_Tests.LAPDm.TC_sabm_dm
- pass->FAIL BTS_Tests.LAPDm.TC_establish_ign_first_sabm
- pass->FAIL BTS_Tests.LAPDm.TC_iframe_seq_and_ack
- pass->FAIL BTS_Tests.LAPDm.TC_iframe_timer_recovery
- pass->FAIL BTS_Tests.LAPDm.TC_rec_invalid_frame
- pass->FAIL BTS_Tests.LAPDm.TC_segm_concat_dcch
- pass->FAIL BTS_Tests.LAPDm.TC_segm_concat_sacch
- pass->FAIL BTS_Tests.LAPDm.TC_rr_response_frame_loss
- pass->FAIL BTS_Tests.LAPDm.TC_incorrect_cr
- pass->FAIL BTS_Tests.LAPDm.TC_sabm_incorrect_c

Some of them related to the recent changes to fake_trx.py (see #4658), so let's filter them out:

- pass->FAIL BTS_Tests.TC_meas_res_sign_tchf
- pass->FAIL BTS_Tests.TC_meas_res_sign_sdcch8
- pass->FAIL BTS_Tests.TC_SI_sched_2Quater  <-- unrelated regression (see OS#4662)
- pass->FAIL BTS_Tests.TC_SI_sched_13_2bis_2ter_2quater  <-- unrelated regression (see OS#4662)
- pass->FAIL BTS_Tests.TC_pcu_time_ind
- pass->FAIL BTS_Tests.LAPDm.TC_absm_invalid_sabm
- pass->FAIL BTS_Tests.LAPDm.TC_absm_dm
- pass->FAIL BTS_Tests.LAPDm.TC_establish_ign_first_sabm
- pass->FAIL BTS_Tests.LAPDm.TC_iframe_timer_recovery
- pass->FAIL BTS_Tests.LAPDm.TC_segm_concat_dcch
- pass->FAIL BTS_Tests.LAPDm.TC_rr_response_frame_loss
- pass->FAIL BTS_Tests.LAPDm.TC_incorrect_cr
- pass->FAIL BTS_Tests.LAPDm.TC_absm_incorrect_c

Both measurement related test case failures (TC_meas_res_sign_tchf and TC_meas_res_sign_sdcch8) seem to be caused by the fact that in fake_trx.py both FAKE_TOA and FAKE_RSSI commands apply to a transceiver they are sent to, while in this setup we deal with several transceivers. The easiest approach would be to send FAKE_* commands to all transceivers.

#25 - 07/14/2020 01:40 PM - fixeria

This change solves the LAPDm test case failures:

https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/19253 BTS_Tests_LAPDm: consider frequency hopping parameters

#26 - 07/14/2020 02:21 PM - fixeria

This change makes BTS_Tests.TC_pcu_ptcch pass:

https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/19254 BTS_Tests: fix PDCH tests: consider frequency hopping

#27 - 08/02/2020 08:54 PM - fixeria
- Checklist item [x] pass the hopping parameters (MAIO, HSN, ...) over the PCU socket so the PCU knows about them (common part) set to Done
- Status changed from In Progress to Feedback
- % Done changed from 90 to 100

https://gerrit.osmocom.org/c/osmo-bts/+/19510
https://gerrit.osmocom.org/c/osmo-bts/+/19511
https://gerrit.osmocom.org/c/osmo-bts/+/19513

pcu_sock: warn about maximum transceiver number constraints
pcu_sock: separate trx / ts filling from pcu_tx_info_ind()
pcuf_proto: version 10: add frequency hopping parameters

All changes submitted to Gerrit, waiting for code review.

#28 - 08/03/2020 09:04 AM - fixeria

I also submitted a TTCN-3 test case:

https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/19517 BTS_Tests: compose the MA bitmask in f_resolve_fh_params()
https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/19518 BTS_Tests: verify hopping parameters in the INFO.ind message

and it reveals a problem with the Mobile Allocation bitmask sent by osmo-bts-trx. To be fixed.

#29 - 08/11/2020 06:42 AM - fixeria

- Blocked by deleted (Bug #4658: Wrong burst order in a multi-trx setup)

#30 - 08/11/2020 06:42 AM - fixeria

- Related to Bug #4658: Wrong burst order in a multi-trx setup added

#31 - 08/11/2020 11:37 AM - fixeria

- Status changed from Feedback to Resolved

All changes have been merged.

#32 - 08/13/2020 11:38 AM - fixeria

All hopping test cases were failing on Jenkins two times in a row. I found and fixed the problems causing this:

https://gerrit.osmocom.org/c/docker-playground/+/19623 ttcn3-bts-test/jenkins.sh: use bts-tester-{generic,oml,virtphy,hopping}
https://gerrit.osmocom.org/c/docker-playground/+/19624 fixup ttcn3-bts-test/jenkins.sh: enable frequency hopping test cases

#33 - 08/15/2020 12:07 PM - fixeria

- Related to Bug #4708: Various SMSCB test cases fail with frequency hopping enabled added