The final aim of this task is to check if we run into CPU limitations of the Raspi CM3 of the LimeNet-micro when maximizing the channel load even of a single TRX.

Say, for example, 14 concurrent TCH/H with AMR inside on the 7 TS. It would be very interesting to see if that works, and if there is any margin on the CPU left, etc.

Quick way to test manually: Use osmo-bsc connected to osmo-bts-trx and use VTY command "bts <0-255> trx <0-255> timeslot <0-7> sub-slot <0-7> (activate|deactivate) (hr|fr|efr|amr) [<0-7>]"

"Automatic" testing: Use TTCN3, add test to BTS_Tests.ttcn:

As we even have that RTP source and sink you could even send some (random payload) RTP messages if you'd want, they just need to match in size and in terms of the RTP payload type. We don't care about the content of the RTP at all here.

Make sure the config/vty is configured for 7 timeslots at TCH/H and unlimited radio link timeout and then send the 14x CHAN ACT with the right channel mode (and if you want to add RTP, the IPA CRCX/MDCX).

Related issues:
Related to OsmoTRX - Bug #4366: Create testsuite to test osmo-bts-trx+osmo-trx under high channel load (w/ RACH correlation) added

I started some work in osmo-ttcn3-hacks.git and docker-playground.git branch "pespin/bts-perf".

https://git.osmocom.org/osmo-ttcn3-hacks/commit/?h=pespin/bts-perf
https://git.osmocom.org/docker-playground/commit/?h=pespin/bts-perf

In there I have a passing test which uses an osmo-bsc.cfg with all channels 1..7 set to TCH/H and the TC_pespin test activates speech AMR on all of them in the BTS, waits 10 seconds, then deactivates them.

Next step is to launch the test against osmo-bts-trx+osmo-trx-lms running on my own laptop to see if there's any issue. Later, run it against same setup but running in LimeNet-micro.

Note, that in normal operation, osmo-bts-trx also consumes a considerable amount of CPU. Not as much as osmo-trx but still quite noticeable. But this only happens when osmo-trx produces bursts for osmo-bts-trx to process. If you just enable channels, osmo-trx will produce idle bursts (or no bursts, depending on the version and configuration), which means osmo-bts-trx won't have anything to process.

Also note that osmo-bts-trx CPU usage might depend on the uplink signal quality because it involves Viterbi decoding which iterates over a signal. I personally haven't tested this but this is the theory.

I have a ttcn3 test opening 14 TCH/H channels from BSC side towards a bts-trx->LimeNet-micro all orchestrated with osmo-gsm-tester, and it seems...
to be working fine
In LimeNet-micro, top shows a load average of: 0.77, 0.78, 0.51
That's with osmo-bts-trx running outside of LimeNet-micro and without RTP processing though, but looks like there's enough load to keep it going anyway.

In osmo-gsm-tester prod main unit: jenkins ~/test/:

```
./run_locally.sh -s ttcn3_bts_tests:trx-lms-limenet-micro -l dbg -T -t highchanload_tchh
```

It worked fine for a while (20 mins) with no overruns or similar, while I could monitor it though VTY, top, etc.
I then tried to run stress-ng in the RPI CM3 while osmo-trx was running, and osmo-bts-trx quickly dropped the connection probably due to CLOCK IND arriving to late, despite osmo-trx-lms running with rt-prio 18. That's expected I'd say though because I did really put the whole system into a crazy stress which shouldn't happen in usual ways. I'll try do run with more realistic stress next times. that's what I used so far:

```
stress-ng --vm 4 --hdd 4 --cpu 4 --open 4 --sem 4 --sock 4 --io 2 --timer 2
```

https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/17098 bts: Introduce new module for performance tests

Next steps:
- Support running osmo-bts-trx remotely by osmo-gsm-tester into RPI CM3 module (same as we do for osmo-trx-lms, run the packaged binary directly).
- Add an RTP source/sink to the TTCN3 code to make sure some data is being sent so that osmo-bts-trx does some work (first in docker setup branch pespin/bts-perf, then in real HW with osmo-gsm-tester).

The problem is that we don't have an easy way to feed the uplink in this case afaiu, only the downlink.

#5 - 02/20/2020 12:32 PM - dexter

I have the testcase BTS_Tests_perf.TC_highchanload_tchh now running on my side. I am currently testing with faketrx and trxcon. I have implemented Sending of the IPA CRCX and now I get the IP/Port from the BTS back. I wonder if sending RTP packets possible immediately or if I have to go through the complete procedure.

#6 - 02/21/2020 02:02 PM - dexter

I have now added the RTP emulation and I can send RTP packets to the BTS. However, it still fails some where but I can already see the packets in WireShark.

#7 - 02/24/2020 09:17 PM - dexter

Sending RTP packets to all channels of the BTS works now. Now I will try to receive RTP packets. My first test sending an MDCX to to BTS looked promising. I indeed get packets from the BTS, however it still needs some more integration.

#8 - 02/27/2020 10:18 AM - dexter

- % Done changed from 50 to 60

The RTP side of the BTS test is now done. We have now RTP streams in both directions, next step is to also work with the traffic from the UM side.

See also: https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/17285 bts: Add RTP payload testing to performance tests

#9 - 03/09/2020 12:15 PM - dexter

I am currently working on the UM side. Unfortunately I am having problems with fake_trx.py. I can clearly see that frames from the BTS are arriving inside faketrx but they get stuck in burst_fwd.py. The reason is from what I can see that in method forward_msg() trx == src_trx is true. I do not understand what this is about. There are two TRX one for MS and one for BTS, shouldn't there be only one TRX for the BTS or is the TRX for MS the interface to trxcon?

#10 - 03/09/2020 01:25 PM - fixeria

There are two TRX one for MS and one for BTS, shouldn't there be only one TRX for the BTS or is the TRX for MS the interface to trxcon?

Yes, one Transceiver is for the MS (trxcon), another is for the BTS (osmo-bts-trx). Each Transceiver has its own TRX interface (TRXC and TRXD
sockets).

The reason is from what I can see that in method forward_msg() trx src_trx is true. I do not understand what this is about.

This is rather a cosmetic check. We're not supposed to deliver a burst to the same Transceiver that originated it, right? You can even drop it, and it would still work as expected because (given that trx src_trx) trx.rx_freq != trx.tx_freq.

#11 - 03/09/2020 01:55 PM - fixeria

You basically need to make sure that trxcon expects to receive bursts on TCH timeslots. If no logical channels are activated, then no SETSLOT command is sent to fake_trx.py, so the corresponding timeslot(s) are not enabled and the bursts are dropped. Your test case should send L1CTL_EST_REQ for each logical channel (i.e. 14 times). I guess that's the problem.

P.S. There may be some troubles with activating multiple logical channels in trxcon. In particular, it deactivates all other logical channels on receipt of L1CTL_EST_REQ. This little hack should make it work:
https://git.osmocom.org/osmocom-bb/commit/?h=fixeria/gprs&id=3d2bd1fdcbfb4864e30f35705321ecd57c88bb0b.

#12 - 03/11/2020 03:46 PM - pespin

Assignee changed from pespin to dexter

Assigning to dexter since he's taking care for the RTP part. Please-reassign it to me once you have the docker setup working. I can then make sure it's working fine in the osmo-gsm-tester setup running against LimeNET-micro.

#13 - 03/30/2020 04:30 PM - dexter

I have done some experiments today. It looks like the problems with TCH data not forwarded correctly inside of faketrx are now gone. I can follow the TCH data now from the BTS up to the l1ctl_link.c. I can even see frames arriving on TTCN3 level but with limitations. It looks like I only receive the FACCH yet but no TCH data. The reason for this is because the TCH data gets stuck in trxcon because the decoding (CRC8) of the frames fail. I also had trouble with a wrong TSC, but the errors also do not vanish after I have corrected that. I need to investigate what's wrong here.

A possible cause for the problem might be the fact that I currently only activate the channel, but I do not do any assignment, maybe that causes the BTS sending dummy frames or invalid frames?

#14 - 04/06/2020 09:34 PM - dexter

I have now found the cause for the non decoding RTP packets. The problem here was that the RTP emulation sends a 4 byte RTP packet by default. This is of course a made up payload and has nothing to do with a GSM V1 voice RTP packet. However, the BTS accepts those packets but it tosses the payload deeper in the stack in scheduler_trx.c:tx_tch_common() where the voice payload is expected to be 15 bytes. I have reconfigured the RTP emulation now so that it sends 15 bytes payload and now trxcon decodes the voice payload it gets from the BTS just fine (CRC matches)

Next step is to make sure the payload is received on the TTCN3 side and then we will loop it back.

#15 - 04/07/2020 04:49 PM - dexter

- % Done changed from 60 to 70

On TTCN3 level we now receive the traffic payload. An altstep receives the incoming traffic indications and sends them back as traffic requests to trxcon. This now works for a single TCH/H channel. When I increase the channel count to the possible maximum of 14 channels I get problems on the RTP side:

18:41:04.621682 16 RTP_Emulation.ttcn:531 setverdict(fail): none -> fail reason: "Connection refused (unexpected)", new component reason: "Connection refused (unexpected)"

I wonder what that could be as I did not see this when the TRX side was not handled.

#16 - 04/20/2020 09:36 AM - dexter

I have implemented some checks that check if the actual payload we pass around remains intact. For the first channel this is indeed the case, but for all subsequent the payload that comes back is all zero. It looks like if all subsequent channels do not receive data from trxcon. All payload that is passed on the L1CTL interface looks good, however I still need to make sure that I even receive payload of any subsequent channels on L1CTL, the chain may be interrupted even earlier. I also made sure that https://git.osmocom.org/osmocom-bb/commit/?h=fixeria/gprs&id=3d2bd1fdcbfb4864e30f35705321ecd57c88bb0b is in place, so this can not be the problem.

#17 - 05/04/2020 07:30 PM - dexter
I tried to locate the problem. Unfortunately I was not able to narrow it down completely to something specific yet. The setslot command seems to be applied correctly. The timeslots are indeed available but I get wired behavior. When I work with one TS, one SS only everything works fine but once I use multiple TS it only works for the second SS, which is confusing.

#18 - 05/05/2020 01:47 PM - dexter

I have traced the problem all the way up to TRXCON now. It seems that when the L1CTL_DM_EST_REQ is sent from TTCN3 it resets the TS, which means that it deactivates all lchans on that slot, and since each TS is used twice, the last L1CTL_DM_EST_REQ wins. That's why we see only TCH/H1 working but no TCH/H0. So, we need to work around the reset, we must not reset the entire TS, but only the SS we really mean.

#19 - 05/07/2020 03:55 PM - dexter

The test now runs fine with multiple channels. I have also added checks to make sure that the traffic is really passed around as expected and not replaced by dummy frames or even silence. On my machine I can simulate up to 8 channels before the load becomes a problem. However this does not mean anything, just because I use a very old laptop and its all still on faketrx/trxcon.

I think its time now to try it out with real hardware. I think I will use the USRP B200 on a separate laptop as a BTS and the motorola phone + ttcn3 on my laptop.

See also:
https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/17285 bts: Add RTP payload testing to performance tests

#20 - 05/07/2020 03:55 PM - dexter

- % Done changed from 70 to 80