OsmoTRX - Bug #3339

osmo-trx-lms "expect ... got ... diff ff0" error message

06/13/2018 01:34 PM - laforge

Status:	Feedback	**Start date:**	06/13/2018
Priority:	Urgent	**Due date:**	
Assignee:	pespin	**% Done:**	90%
Category:	LimeSDR		
Target version:			
Spec Reference:			

Description

After running osmo-trx-lms for ~10mins on a LimeSDR-mini, I am getting:

```
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e11d0 got 939e21c0 (939e21c0) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e1b94 got 939e2b84 (939e2b84) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e2558 got 939e3548 (939e3548) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e2f1c got 939e3f0c (939e3f0c) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e38e0 got 939e48d0 (939e48d0) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e42a4 got 939e5294 (939e5294) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e4c68 got 939e5c58 (939e5c58) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e562c got 939e661c (939e661c) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e5ff0 got 939e6fe0 (939e6fe0) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e69b4 got 939e79a4 (939e79a4) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e7378 got 939e8368 (939e8368) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e7d3c got 939e8d2c (939e8d2c) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e8700 got 939e96f0 (939e96f0) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e90c4 got 939ea0b4 (939ea0b4) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939e9a88 got 939eaa78 (939eaa78) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939ea44c got 939eb43c (939eb43c) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939eae10 got 939ebe00 (939ebe00) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939eb7d4 got 939ec7c4 (939ec7c4) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939ec198 got 939ed188 (939ed188) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939ecb5c got 939edb4c (939edb4c) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939ed520 got 939ee510 (939ee510) diff=ff0
Wed Jun 13 15:32:45 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139695550179072] chan 0 recv buffer o
f len 2500 expect 939edee4 got 939eeed4 (939eeed4) diff=ff0
```

Related issues:

Related to OsmoTRX - Feature #2919: Native LimeSDR support	**Resolved**	**02/09/2018**
Related to OsmoTRX - Bug #3971: osmo-trx-lms: makes kernel eat all system mem...	**New**	**05/03/2019**

History

#1 - 06/13/2018 02:17 PM - laforge

the problem is reproducible here. I'm trying to figure out exactly how long osmo-btx-trx needs to run with the LimeSDR-mini, but it it will happen sooner or later (about ~1min to ~10min after start on a completely unloaded quite high-end 64bit intel system). It doesn't happen at all with LimeSDR non-mini ?!?

#2 - 06/13/2018 02:26 PM - laforge

Another attempt, problem appeared 52 seconds after start:

```
Wed Jun 13 16:17:52 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=140396877776640] chan 0 recv buffer of len 2500 e
xpect 30c2e90 got 30c328c (30c328c) diff=3fc
```

Yet another attempt, problem appeared after 7mins 42 secs:

```
Wed Jun 13 16:25:42 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=140409076295424] chan 0 recv buffer of len 2500 e
xpect 1cdfb364 got 1cdfd344 (1cdfd344) diff=1fe0
```

#3 - 06/13/2018 03:29 PM - laforge

there is no other log messages before which might be able to help us what's happening:

```
Wed Jun 13 16:28:52 2018 DMAIN <0000> LMSDevice.cpp:305 [tid=139743110162176] Setting TX gain to 73 dB.
Wed Jun 13 16:32:43 2018 DMAIN <0000> LMSDevice.cpp:507 [tid=139743110088448] chan 0 recv buffer of len 2500 e
xpect ef3ef2c got ef3f328 (ef3f328) diff=3fc
```

The "Setting TX gain" is part of the power ramping at start-up, and almost 4 minutes later the next message is already the error message.

#4 - 06/13/2018 08:23 PM - laforge

bad news: I saw this once happen with LimeSDR-USB, too. Very hard to reproduce, though.

#5 - 06/14/2018 07:02 AM - laforge

- Related to Feature #2919: Native LimeSDR support added

#6 - 06/28/2018 07:42 PM - rlehm

Using LimeSDR-USB w/ version 2 revision 12, LimeUtil version 2 revision 17, and newest versions of osmo-trx-lms, osmo-nitb, and omso-bts-trx this bug happens 50% of the time. If it doesn't occur on it's own within the first minute or so, trying to register on the network with a phone has a good chance of triggering it.

```
Thu Jun 28 12:29:22 2018 DMAIN <0000> Transceiver.cpp:1018 [tid=140309812614912] ClockInterface: sending IND C
LOCK 1379288
Thu Jun 28 12:29:23 2018 DMAIN <0000> Transceiver.cpp:709 [tid=140309812680448] command is NOHANDOVER 0 0
Thu Jun 28 12:29:23 2018 DMAIN <0000> Transceiver.cpp:1018 [tid=140309812614912] ClockInterface: sending IND C
LOCK 1379505
Thu Jun 28 12:29:23 2018 DMAIN <0000> LMSDevice.cpp:515 [tid=140309812614912] chan 0 recv buffer of len 2500 e
xpect 2b7c79c got 2b7d78c (2b7d78c) diff=ff0
Thu Jun 28 12:29:23 2018 DMAIN <0000> LMSDevice.cpp:515 [tid=140309812614912] chan 0 recv buffer of len 2500 e
xpect 2b7d160 got 2b7e150 (2b7e150) diff=ff0
Thu Jun 28 12:29:23 2018 DMAIN <0000> LMSDevice.cpp:515 [tid=140309812614912] chan 0 recv buffer of len 2500 e
xpect 2b7db24 got 2b7eb14 (2b7eb14) diff=ff0
Thu Jun 28 12:29:23 2018 DMAIN <0000> LMSDevice.cpp:515 [tid=140309812614912] chan 0 recv buffer of len 2500 e
xpect 2b7e4e8 got 2b7f4d8 (2b7f4d8) diff=ff0
```

#7 - 08/16/2018 02:53 PM - cswiger

Looks like a case of just getting behind delivering data - I took the "if" conditional out so osmo-trx-lms prints all "got" and "expect" even if they are

the same, and collected the timestamps around when it goes from diff=0 to non-zero, and there's a gap of 10660 instead of 2500, and from then on all packets are from the future and thus out of sync.

expect hex	got hex	expect dec	got dec	got - expect	got - previous
99f091c	99f091c	161417500	161417500	0	2500
99f12e0	99f12e0	161420000	161420000	0	2500
99f1ca4	99f1ca4	161422500	161422500	0	2500
99f2668	99f4648	161425000	161433160	8160	10660 <--*
99f302c	99f500c	161427500	161435660	8160	2500
99f39f0	99f59d0	161430000	161438160	8160	2500
99f43b4	99f6394	161432500	161440660	8160	2500

#8 - 08/17/2018 01:51 PM - laforge

- Priority changed from Normal to Urgent

#9 - 08/20/2018 06:40 AM - odunboye

I also got the following:
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad3b08 got ac659d0 (ac659d0) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad44cc got ac66394 (ac66394) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad4e90 got ac66d58 (ac66d58) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad5854 got ac6771c (ac6771c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad6218 got ac680e0 (ac680e0) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad6bdc got ac68aa4 (ac68aa4) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad75a0 got ac69468 (ac69468) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad7f64 got ac69e2c (ac69e2c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad8928 got ac6a7f0 (ac6a7f0) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad92ec got ac6b1b4 (ac6b1b4) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aad9cb0 got ac6bb78 (ac6bb78) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aada674 got ac6c53c (ac6c53c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aadb038 got ac6cf00 (ac6cf00) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aadb9fc got ac6d8c4 (ac6d8c4) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aadc3c0 got ac6e288 (ac6e288) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aadcd84 got ac6ec4c (ac6ec4c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aadd748 got ac6f610 (ac6f610) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aade10c got ac6ffd4 (ac6ffd4) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aadead0 got ac70998 (ac70998) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aadf494 got ac7135c (ac7135c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aadfe58 got ac71d20 (ac71d20) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae081c got ac726e4 (ac726e4) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae11e0 got ac730a8 (ac730a8) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae1ba4 got ac73a6c (ac73a6c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae2568 got ac74430 (ac74430) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae2f2c got ac74df4 (ac74df4) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae38f0 got ac757b8 (ac757b8) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae42b4 got ac7617c

(ac7617c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae4c78 got ac76b40 (ac76b40) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae563c got ac77504 (ac77504) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae6000 got ac77ec8 (ac77ec8) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae69c4 got ac7888c (ac7888c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae7388 got ac79250 (ac79250) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae7d4c got ac79c14 (ac79c14) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae8710 got ac7a5d8 (ac7a5d8) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae90d4 got ac7af9c (ac7af9c) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aae9a98 got ac7b960 (ac7b960) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aaea45c got ac7c324 (ac7c324) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aaeae20 got ac7cce8 (ac7cce8) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aaeb7e4 got ac7d6ac (ac7d6ac) diff=191ec8
Mon Aug 20 07:11:17 2018 DDEV <0001> LMSDevice.cpp:515 [tid=139690793129728] chan 0 recv buffer of len 2500 expect aaec1a8 got ac7e070 (ac7e070) diff=191ec8

#10 - 11/12/2018 01:46 PM - fixeria

- Subject changed from osmo-bts-lms "expect ... got ... diff ff0" error message to osmo-trx-lms "expect ... got ... diff ff0" error message

#11 - 11/26/2018 03:13 PM - laforge

Irrespective of the root cause that makes a given setup loose samples on read, the UHD code (smpl_buf class) is re-synchronizing after a loss and only prints a short error message and recovers, while the LMS code in OsmoTRX fails indefinitely from the first lost sample buffer.

So in order to level the playing field and do better comparison, we should implement the same re-sync logic. Next step then is to see if the number of lost sample messages is different between e.g. a USRP device and a LMS device on the same system/os/hardware.

#12 - 11/29/2018 12:31 PM - laforge

#13 - 12/04/2018 06:29 PM - pespin

Since a few days ago I can reproduce this issue again in laptop, I couldn't reproduce it so far with apparently same setup and could run LimeSDR successfully for lots of minutes without issue. Only related change I can think of is kernel/libusb/cpufreq upgrade. Anyway, since recently I get this issue after a few seconds of osmo-trx + osmo-bts-trx running.

Interestingly, my system totally freezes for 2-5 seconds before/during the time osmo-trx starts failing reading/writing. My XServer blocs and music playing from a youtube video on the background also either stops or plays in a 1 sec loop. When I recover control of my system, I can see in the logs of osmo-trx the read/write failure:

```
Tue Dec  4 19:02:26 2018 DMAIN <0000> Transceiver.cpp:1039 [tid=139625388013312] ClockInterface: sending IND C
LOCK 1000817
Tue Dec  4 19:02:27 2018 DMAIN <0000> Transceiver.cpp:1039 [tid=139625388013312] ClockInterface: sending IND C
LOCK 1001034
Tue Dec  4 19:02:28 2018 DMAIN <0000> Transceiver.cpp:1039 [tid=139625388013312] ClockInterface: sending IND C
LOCK 1001250
Tue Dec  4 19:02:28 2018 DMAIN <0000> Transceiver.cpp:391 [tid=139625388107520] chan 0 dumping STALE burst in
TRX->USRP interface (0:1001248 vs 3:1001249), retrans=0
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:391 [tid=139625388107520] chan 0 dumping STALE burst in
TRX->USRP interface (1:1001248 vs 3:1001249), retrans=0
Tue Dec  4 19:02:34 2018 DDEV <0002> LMSDevice.cpp:525 [tid=139625388013312] LMS: Device receive timed out (14
00 vs exp 2500).
Tue Dec  4 19:02:34 2018 DLMS <0003> LMSDevice.cpp:83 [tid=139625325979392] popping from TX, samples popped 56
0/1020
Tue Dec  4 19:02:34 2018 DCTRL <0001> Transceiver.cpp:724 [tid=139625388414720] chan 0: command is 'POWEROFF'
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:391 [tid=139625388107520] chan 0 dumping STALE burst in
TRX->USRP interface (2:1001248 vs 3:1001249), retrans=0
Tue Dec  4 19:02:34 2018 DMAIN <0000> radioInterface.cpp:330 [tid=139625388013312] Receive error -1
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:391 [tid=139625388107520] chan 0 dumping STALE burst in
```

```
TRX->USRP interface (3:1001248 vs 3:1001249), retrans=0
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:307 [tid=139625388414720] Stopping the transceiver
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:908 [tid=139625388013312] radio Interface receive failed
, requesting stop.
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:391 [tid=139625388107520] chan 0 dumping STALE burst in
TRX->USRP interface (4:1001248 vs 3:1001249), retrans=0
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:391 [tid=139625388107520] chan 0 dumping STALE burst in
TRX->USRP interface (5:1001248 vs 3:1001249), retrans=0
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:391 [tid=139625388107520] chan 0 dumping STALE burst in
TRX->USRP interface (6:1001248 vs 3:1001249), retrans=0
Tue Dec  4 19:02:34 2018 DMAIN <0000> Transceiver.cpp:391 [tid=139625388107520] chan 0 dumping STALE burst in
TRX->USRP interface (7:1001248 vs 3:1001249), retrans=0
Tue Dec  4 19:02:34 2018 DDEV <0002> LMSDevice.cpp:530 [tid=139625388013312] chan 0 recv buffer of len 2500 ex
pect 27b4cb8 got 27b5230 (27b5230) diff=578
Tue Dec  4 19:02:34 2018 DDEV <0002> LMSDevice.cpp:530 [tid=139625388013312] chan 0 recv buffer of len 2500 ex
pect 27b567c got 27b5bf4 (27b5bf4) diff=578
Tue Dec  4 19:02:34 2018 DDEV <0002> LMSDevice.cpp:530 [tid=139625388013312] chan 0 recv buffer of len 2500 ex
pect 27b6040 got 27b65b8 (27b65b8) diff=578
```

Immediately after, it can be seen that osmo-bts-trx has requested it to stop (through 'POWEROFF' command). That poweroff command is sent by BTS because it lost the clock:

```
20181204190226785 DTRX <000b> trx_if.c:124 Clock indication: fn=1000817
20181204190226785 DL1C <0006> scheduler_trx.c:1578 TRX Clock Ind: elapsed_us=1001696, elapsed_fn=217, error_us
= +241
20181204190226785 DL1C <0006> scheduler_trx.c:1596 GSM clock jitter: -3946us (elapsed_fn=0)
20181204190227785 DTRX <000b> trx_if.c:124 Clock indication: fn=1001034
20181204190227785 DL1C <0006> scheduler_trx.c:1578 TRX Clock Ind: elapsed_us=1000353, elapsed_fn=217, error_us
=-1102
20181204190227785 DL1C <0006> scheduler_trx.c:1596 GSM clock jitter: -2746us (elapsed_fn=0)
20181204190228629 DL1C <0006> scheduler_trx.c:1463 FN timer expire_count=3: We missed 2 timers
20181204190228842 DL1C <0006> scheduler_trx.c:1463 FN timer expire_count=33: We missed 32 timers
20181204190228965 DL1C <0006> scheduler_trx.c:1485 PC clock skew: elapsed_us=272900, error_us=268285
20181204190229462 DOML <0001> bts.c:250 Shutting down BTS 0, Reason No clock from osmo-trx
20181204190229467 DL1C <0006> scheduler.c:242 Exit scheduler for trx=0
```

The clock is lost most probably due to recv() timing out and thus the CLOCK in osmo-trx not advancing.

"FN timer expire_count=33: We missed 32 timers", so really osmo-bts-trx neither had time to process timers. That's strange given that I'm running it on a i7-8550U with 8 logical processors.

#14 - 12/04/2018 07:01 PM - pespin

Tx path may be somehow blocking Rx path:

```
Tue Dec  4 19:57:31 2018 DDEV <0002> LMSDevice.cpp:513 [tid=140520674105088] chan 0: underrun 0, overrun 0
Tue Dec  4 19:57:31 2018 DDEV <0002> LMSDevice.cpp:545 [tid=140520674105088] chan 0 recv buffer of len 2500 ex
pect 31a72e8 got 31a72e8 (31a72e8) diff=0
Tue Dec  4 19:57:31 2018 DDEV <0002> LMSDevice.cpp:513 [tid=140520674105088] chan 0: underrun 0, overrun 0
Tue Dec  4 19:57:31 2018 DDEV <0002> LMSDevice.cpp:545 [tid=140520674105088] chan 0 recv buffer of len 2500 ex
pect 31a7cac got 31a7cac (31a7cac) diff=0
Tue Dec  4 19:57:36 2018 DLMS <0003> LMSDevice.cpp:83 [tid=140520614848256] popping from TX, samples popped 88
0/1020                                  <-- My system freezes around this time (without seeing it). After
that, Rx no longer matches expectancy.
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:513 [tid=140520674105088] chan 0: underrun 0, overrun 0
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:545 [tid=140520674105088] chan 0 recv buffer of len 2500 ex
pect 31a8670 got 360306c (360306c) diff=45a9fc
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:513 [tid=140520674105088] chan 0: underrun 0, overrun 0
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:545 [tid=140520674105088] chan 0 recv buffer of len 2500 ex
pect 31a9034 got 3603a30 (3603a30) diff=45a9fc
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:513 [tid=140520674105088] chan 0: underrun 0, overrun 0
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:545 [tid=140520674105088] chan 0 recv buffer of len 2500 ex
pect 31a99f8 got 36043f4 (36043f4) diff=45a9fc
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:513 [tid=140520674105088] chan 0: underrun 0, overrun 0
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:545 [tid=140520674105088] chan 0 recv buffer of len 2500 ex
pect 31aa3bc got 3604db8 (3604db8) diff=45a9fc
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:513 [tid=140520674105088] chan 0: underrun 0, overrun 0
Tue Dec  4 19:57:36 2018 DDEV <0002> LMSDevice.cpp:545 [tid=140520674105088] chan 0 recv buffer of len 2500 ex
pect 31aad80 got 360577c (360577c) diff=45a9fc
```

(log prints underrun and overrun values from LMS_GetStreamStatus() immediatelly after checking rc from LMS_RecvStream).

#15 - 12/04/2018 07:26 PM - pespin

In my case I just spotted through htop that while running osmo-trx + osmo-bts-trx, memory suddenly grows until filling my 16GB, and then is when my system freezes and osmo-trx starts failing.

However, so far I cannot see this memory is attached to the osmo-trx or osmo-bts-trx processes, I still need to investigate where it goes.

EDIT: after investigation, it seems memory is allocated by the kernel. I checked it with watch smem -twk

memory is allocated even without having a running osmo-bts-trx (and thus without calling readSamples() in osmo-trx, only LMS_Open and LMS_Init).

```
$ smem -twk
Area                           Used      Cache   Noncache
firmware/hardware                 0          0          0
kernel image                      0          0          0
kernel dynamic memory          1.2G     988.1M     281.6M
userspace memory               4.1G     550.8M       3.6G
free memory                   10.1G      10.1G          0
------------------------------------------------------------
                              15.4G      11.6G       3.8G
```

After a few seconds of running osmo-trx-lms, my memory is already full (according to htop) and the same command shows:

```
$ smem -twk
Area                           Used      Cache   Noncache
firmware/hardware                 0          0          0
kernel image                      0          0          0
kernel dynamic memory         10.2G    1009.3M       9.2G  <-----!!!!!!!
userspace memory               4.1G     566.3M       3.6G
free memory                    1.1G       1.1G          0
------------------------------------------------------------
                              15.4G       2.6G      12.8G
```

So it seems launching osmo-trx-lms with LimeSuite allocates 9.2GB of kernel memory.

Following mem usage through htop mem bar, I can see it eventually allocates all memory and my system freezes for 1-2 seconds, then the kernel frees some (around 1GB probably the cache), then it continues growing again until it fills it again, then the kernel deallocates all that memory to the initial status when osmo-trx-lms was started, but then it continues allocating it once again.

#16 - 12/04/2018 07:31 PM - pespin

I'm now using sudo slabtop -s c to monitor which kind of memory is growing.

It turns out the filp and kmalloc-64 slabs are growing like crazy following the general MEM usage I see through htop. When memory is full and kernel frees it, I see these 2 lines disapearing from slabtop. Same when I kill osmo-trx-lms.

```
 Active / Total Objects (% used)    : 52614248 / 52704560 (99.8%)
 Active / Total Slabs (% used)      : 1239371 / 1239371 (100.0%)
 Active / Total Caches (% used)     : 103 / 137 (75.2%)
 Active / Total Size (% used)       : 8279108.19K / 8309975.23K (99.6%)
 Minimum / Average / Maximum Object : 0.01K / 0.16K / 23.19K

  OBJS ACTIVE  USE OBJ SIZE  SLABS OBJ/SLAB CACHE SIZE NAME
26117200 26114592  99%    0.25K 816487       32   6531896K filp            <---!!!!!!!
26120640 26118794  99%    0.06K 408135       64   1632540K kmalloc-64      <---!!!!!!!
 31248  14895  47%    0.57K   1116       28     17856K radix_tree_node
 11592   5311  45%    1.11K    414       28     13248K btrfs_inode
 19350  17632  91%    0.59K    719       27     11504K inode_cache
  1216   1172  96%    7.69K    304        4      9728K task_struct
 47565  28390  59%    0.19K   2265       21      9060K dentry
```

#17 - 12/04/2018 07:41 PM - pespin

Interestingly, if I strace the osmo-trx-lms I don't see this kind of issue, but it's true too that the CPU consumption drops a lot too.

#18 - 12/10/2018 05:04 PM - pespin

If I ctrl+z (SIGSTOP) the osmo-trx-lms, the kernel stops acquiring memory (and releases most of it). Once I use "fg" to SIGCONTINUE the process, it

continues acquiring memory like crazy.

#19 - 12/10/2018 05:13 PM - pespin

Same issue with memory allocation if connecting the LimeSDR through a USB2 cable (and showing up as media=USB 2.0 in LimeUtil --find). (I usually run it through USB3).

#20 - 12/10/2018 05:48 PM - pespin

I see the issues stated aboce using following versions:

Lenovo Thinkpad x280 (Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz)

kernel:

```
Linux pespin-thinkpad 4.19.4-arch1-1-ARCH #1 SMP PREEMPT Fri Nov 23 09:06:58 UTC 2018 x86_64 GNU/Linux
```

LimeSuite:

```
Version information:
  Library version:      v18.10.0-18.10.0.r0.59d51d5
  Build timestamp:      2018-10-04
  Interface version:    v2018.10.0
  Binary interface:     18.10-1
```

libusb:

```
libusb 1.0.22-1
```

#21 - 12/14/2018 05:22 PM - roh

hi

i tried to replicate this with a -nightly build (freshly updated)

versions:
system is debian stable
limesdr-usb connected via usb3 to apu2

```
kernel        4.9.0-8-amd64
libusb        1.0.21-1
limesuite     18.10.0-1
osmo-trx-lms  0.4.0.121.25185

LimeUtil --make
Make device
  Device name: LimeSDR-USB
  Expansion name: UNSUPPORTED
  Firmware version: 4
  Hardware version: 4
  Protocol version: 1
  Gateware version: 2
  Gateware revision: 18
  Gateware target: LimeSDR-USB
  Serial number: 0x9060b00472227
  Free connection... OK
```

before starting osmo-trx-lms:

```
smem -twk
Area                           Used      Cache   Noncache
firmware/hardware                 0          0          0
kernel image                      0          0          0
kernel dynamic memory         128.0M      83.0M      45.0M
userspace memory               64.7M      25.2M      39.5M
free memory                     3.7G       3.7G          0
```

```
-------------------------------------------------------------
                               3.8G       3.8G      84.5M
```

after starting osmo-trx-lms:

```
smem -twk
Area                           Used      Cache   Noncache
firmware/hardware                 0          0          0
kernel image                      0          0          0
kernel dynamic memory         129.6M      82.7M      46.9M
userspace memory               85.5M      28.2M      57.3M
free memory                     3.6G       3.6G          0
-------------------------------------------------------------
                               3.8G       3.7G     104.2M
```

after some minutes of uptime (no fail/buffer desync):

```
smem -twk
Area                           Used      Cache   Noncache
firmware/hardware                 0          0          0
kernel image                      0          0          0
kernel dynamic memory         132.9M      85.7M      47.1M
userspace memory               91.5M      34.3M      57.3M
free memory                     3.6G       3.6G          0
-------------------------------------------------------------
                               3.8G       3.7G     104.4M

smem -twk
Area                           Used      Cache   Noncache
firmware/hardware                 0          0          0
kernel image                      0          0          0
kernel dynamic memory         132.5M      85.8M      46.7M
userspace memory               91.8M      34.4M      57.4M
free memory                     3.6G       3.6G          0
-------------------------------------------------------------
                               3.8G       3.7G     104.1M
```

so i can not replicate the kernelside allocation problem. maybe try and see if this is a bug in recent kernel(s)?

#22 - 04/27/2019 10:57 PM - pespin

Regarding out of sync issue:
It was discussed during OsmoDevCon2019 that in LimeSDR osmo-trx driver we miss a buffer to push "ahead-of-time" reads from SDR, like it is done in UHD one.
So the idea is to have a buffer quite bigger than what we read during readSamples(), and then if we get out of sync, we expect to read timestamp 123 but we get 678, we store 678 into this temporary buffer in buff[678] and onwards, and return always buf[0] with len passed. During start of readSamples(), buffer content is moved left so that buffer[0] is the timestamp we want to read. This way during out-of-sync we'll return a few garbage bursts but since we are still storing future bursts, when we arrive there we'll still be sending valid data to upper layers.

We should actually move this buffer management to generic code instead of having it duplicated in every driver. This way driver doesn't have an expectation on timestamp, it simply reads len bytes and returns timestamp + buffer.

#23 - 05/03/2019 01:58 PM - pespin

I was going to implement and test the buffering solution, but in my host I still hit the issue mentioned above about the kernel eating the whole memory space when osmo-trx-lms is started, which prevents me from testing or running osmo-trx-lms.

I still see the same issue using latest osmo-trx-lms, limesuite master and archlinux kernel (quite new):

kernel:

```
5.0.9-arch1-1-ARCH #1 SMP PREEMPT Sat Apr 20 15:00:46 UTC 2019 x86_64 GNU/Linux
```

limesuite:

```
Version information:
  Library version:      v19.01.0-19.01.0.r22.d0fdb112
  Build timestamp:      2019-05-03
```

```
  Interface version:    v2019.1.0
  Binary interface:     19.01-1
```

libusb:

```
1.0.22-1
```

I can reproduce the issue with both LimeSDR-USB and LimeSDR-mini.

#24 - 05/03/2019 03:19 PM - pespin

- Related to Bug #3971: osmo-trx-lms: makes kernel eat all system memory when run under realtime priority added

#25 - 05/03/2019 03:19 PM - pespin

Continuing investigation of the memleak in a separate issue: #3971

#26 - 05/03/2019 03:26 PM - pespin

#27 - 05/03/2019 06:14 PM - pespin

- Status changed from New to Feedback

Should be fixed/workarounded by:
https://gerrit.osmocom.org/#/c/osmo-trx/+/13872 lms: Use smpl_buf to recover from timestamp jumps

Once that commit is merged I think we can close this ticket.
Once someone else tests it we probably want to notify LimeSDR somehow about it. Maybe also osmocom mailinglist.

#28 - 05/04/2019 05:42 AM - laforge

while it's good that the workaround is now in place, this only fixes the bug that we
don't recover.

We should still have a redmine issue (this or another one) to
investigate what exactly is stalling the system long enough to cause
those overruns in the first place. Any real base station system should
not silently have overruns.

Also, as indicated in the discussion, there should be a counter for
those kind of events (in the generic part of osmo-bts-trx) that can be
inspected to trace the frequency of such events without having to have a
log file or resorting to long-term log file analysis.

Furthermore, even within this bug workaround issue, it makes sense to
have a ?configurable? "rate cap" on these events, so that you can say
something like "I accept one overrun per hour, but if there's more than
that, I'd actually want to restart. Ideally probably two thresholds:

1) an "alerting" threshold at which point we would send OML ALERT to the BSC
2) a "terminating" threshold at which point we would want to kill
osmo-bts-trx and re-spawn (and also send a [different] OML alert about
it).

Let's please make sure we address the counter + thresholds + alerting in
a timely manner, and create a new issue about the fact that overuns
occur at all despite RT_PRIO.

Regards,
Harald

#29 - 05/07/2019 09:05 AM - pespin

I'm not sure I'm entirely following you here.

We can for sure catch this kind of events in osmo-trx radioDevice implementation and add some API to gather/submit events to have some counters in osmo-trx. Maybe also send a TRAP through CTRL iface if the counter changes. But I don't think we can send that can of information in current TRX protocol towards osmo-bts-trx. When this kind of overruns happen, smpl_buf saves us so from "data" point of view, osmo-bts-trx simply receives garbage instead of valid payload, but that's all. No way to know from osmo-bts-trx whether that's simply noise or random data generated by an overrun afaiu.

fixeria may want to add some comment here or take that into account when thinking about the new protocol between osmo-trx and osmo-bts-trx.

EDIT: task to track overruns created here: https://osmocom.org/issues/3981

#30 - 05/07/2019 09:13 AM - pespin

- Related to Bug #3981: osmo-trx-lms: Overruns appear from time to time added

#31 - 05/07/2019 12:28 PM - fixeria

Hi all,

> [...] But I don't think we can send that can of information in current TRX protocol towards osmo-bts-trx. When this kind of overruns happen, smpl_buf saves us so from "data" point of view, osmo-bts-trx simply receives garbage instead of valid payload, but that's all. No way to know from osmo-bts-trx whether that's simply noise or random data generated by an overrun afaiu.
> fixeria may want to add some comment here or take that into account when thinking about the new protocol between osmo-trx and osmo-bts-trx.

Yep, unfortunately there is no way to inform osmo-bts-trx about such kind of events using the current TRXC (control) protocol. It is similar to HTTP: a client initiates a request, the server responds. The server does not sent anything without a request. I don't think we can easily modify TRXC to have such notifications, this would break the request-response logic.

> [...] osmo-bts-trx simply receives garbage instead of valid payload [...]

Could you please clarify, what do you mean by garbage? Would osmo-trx detect and demodulate any kind of burst from the smpl_buf in such cases? IMHO, it makes sense to send an empty burst (i.e. just a header) on TRXD, since we are going to send such notifications anyway in the near feature, e.g. when nothing is received.

#32 - 05/07/2019 01:12 PM - pespin

fixeria wrote:

> Yep, unfortunately there is no way to inform osmo-bts-trx about such kind of events using the current TRXC (control) protocol. It is similar to HTTP: a client initiates a request, the server responds. The server does not sent anything without a request. I don't think we can easily modify TRXC to have such notifications, this would break the request-response logic.

We could modify osmo-bts-trx to have a timer and request that information (counters) over TRXC every X seconds and then decide what to do.

> Could you please clarify, what do you mean by garbage? Would osmo-trx detect and demodulate any kind of burst from the smpl_buf in such cases? IMHO, it makes sense to send an empty burst (i.e. just a header) on TRXD, since we are going to send such notifications anyway in the near feature, e.g. when nothing is received.

Yes, it will try to demodulate it but it will find random stuff there and probably handle it as if there was a lot of noise. It does whatever osmo-trx does in that case. I guess it simply sends the burst with noise to osmo-bts-trx?

#33 - 05/24/2019 06:28 PM - pespin

Related to last discussions, I submitted a fix for LMS I found, and possibility to store/manage those error counters through rate_ctr in VTY:
remote: https://gerrit.osmocom.org/#/c/osmo-trx/+/14166 lms: Fix stream_stats checks with droppedPackets
remote: https://gerrit.osmocom.org/#/c/osmo-trx/+/14167 Add rate_ctr support to store/retrieve SDR errors through VTY

Example:

```
OsmoTRX> show rate-counters
osmo-trx statistics 0:
   device:rx_underruns:          0 (0/s 0/m 0/h 0/d) Number of Rx underruns
    device:rx_overruns:          0 (0/s 0/m 0/h 0/d) Number of Rx overruns
   device:tx_underruns:          0 (0/s 0/m 0/h 0/d) Number of Tx underruns
 device:rx_drop_events:          4 (0/s 2/m 3/h 0/d) Number of times Rx samples were dropped by HW
device:rx_drop_samples:        513 (0/s 196/m 425/h 0/d) Number of Rx samples dropped by HW
```

and also (WIP, almost there) you can specify thresholds at which point osmo-trx restarts (thus restarting BTS and letting to know up to BSC):
https://gerrit.osmocom.org/#/c/osmo-trx/+/14168 WIP: Add VTY commands to set error ctr thresholds [WIP]
For instance:

```
ctr-error-threshold rx_drop_events 2 minute
```

that means if rate_ctr at some point shows rx_drop_events >= 2m, osmo-trx will die with an error.

Non-related: I updated nightly Limesuite deb to 19.04.0: https://gerrit.osmocom.org/#/c/osmo-ci/+/14153/
TODO: Once it's merged and packages are available, make sure osmo-gsm-tester Prod and RnD are updated with those, as well as gateware firmware flashed to LimeSDR device (LimeUtil --update).

#34 - 06/06/2019 10:14 AM - pespin

- % Done changed from 0 to 90

I updated osmo-gsm-tester Prod with latest LimeSuite stuff.

I also have a bunch of patches waiting for review. Once merged we can close the ticket:

remote: https://gerrit.osmocom.org/c/osmo-trx/+/14382 doc: vty: Update trx_vty_reference.xml
remote: https://gerrit.osmocom.org/c/osmo-trx/+/14167 Add rate_ctr support to store/retrieve SDR errors through VTY
remote: https://gerrit.osmocom.org/c/osmo-trx/+/14168 Add VTY commands to set error ctr thresholds [WIP]
remote: https://gerrit.osmocom.org/c/osmo-trx/+/14366 Rename and move STOP signal from Transceiver to main
remote: https://gerrit.osmocom.org/c/osmo-trx/+/14367 lms: Drop unusued variable masterClockRate
remote: https://gerrit.osmocom.org/c/osmo-trx/+/14368 lms: Fix stream_stats checks with overrun/underrun