OsmoSGSN - Bug #3956

SIGSEGV in rate_ctr_group_free()

04/24/2019 11:27 AM - keith

Status:	Resolved	**Start date:**	04/24/2019
Priority:	Normal	**Due date:**	
Assignee:	pespin	**% Done:**	100%
Category:	GTP interface		
Target version:			
Spec Reference:			

Description

Limited context: all i got :(

```
<0023> gtp.c:2653 Packet from 192.168.100.2:2123, length: 14 content: 32 15 00 06 00 00 00 00 59 4
7 00 00 01 c0 : Unknown PDP context: 0 (expected if gtp_delete_context_
req is used)
<000e> sgsn_libgtp.c:627 libgtp EOF (type=20, pdp=(nil), cbp=0x555555cfeca0)
<0012> gprs_sndcp.c:514 SNSM-DEACTIVATE.ind for non-existing TLLI=f38f648c SAPI=3 NSAPI=6

Program received signal SIGSEGV, Segmentation fault.
```

backtrace:

```
#0  0x00007ffff7313531 in rate_ctr_group_free () from /usr/lib/x86_64-linux-gnu/libosmocore.so.12
#1  0x00005555555651ae in sgsn_pdp_ctx_free (pdp=pdp@entry=0x555555cfeca0) at gprs_sgsn.c:471
#2  0x0000555555570363 in delete_pdp_conf (pdp=0x0, cause=-1, cbp=0x555555cfeca0) at sgsn_libgtp.c
:574
#3  cb_conf (type=20, cause=-1, pdp=0x0, cbp=0x555555cfeca0) at sgsn_libgtp.c:636
#4  0x00007ffff66a6401 in gtp_delete_pdp_conf () from /usr/lib/x86_64-linux-gnu/libgtp.so.4
#5  0x00007ffff66a921c in gtp_decaps1c () from /usr/lib/x86_64-linux-gnu/libgtp.so.4
#6  0x00007ffff7308dd4 in osmo_select_main () from /usr/lib/x86_64-linux-gnu/libosmocore.so.12
#7  0x000055555555b9e7 in main (argc=1, argv=0x7fffffffe668) at sgsn_main.c:524
```

pdp in sgsn_pdp_ctx_free():

```
(gdb) print *pdp
$6 = {list = {next = 0x100100, prev = 0x200200}, g_list = {next = 0x100100,
    prev = 0x200200}, mm = 0x555555d1c7d0, destroy_ggsn = 0, ggsn = 0x0,
  ggsn_list = {next = 0x100100, prev = 0x200200}, ctrg = 0x555555ce9ad0,
  lib = 0x0, state = PDP_STATE_CR_REQ, type = PDP_TYPE_NONE, address = 0,
  apn_subscribed = 0x0, nsapi = 6, sapi = 3, ti = 0 '\000', vplmn_allowed = 0,
  qos_profile_subscr = 0, radio_prio = 0 '\000', timer = {node = {
      rb_parent_color = 0, rb_right = 0x0, rb_left = 0x0}, list = {next = 0x0,
      prev = 0x0}, timeout = {tv_sec = 0, tv_usec = 0}, active = 0, cb = 0x0,
    data = 0x0}, T = 0, num_T_exp = 0, cdr_timer = {node = {
      rb_parent_color = 93824998219265, rb_right = 0x0, rb_left = 0x0}, list = {
      next = 0x555555cfed98, prev = 0x555555cfed98}, timeout = {
      tv_sec = 1556101766, tv_usec = 498923}, active = 0,
    cb = 0x555555577650 <cdr_pdp_timeout>, data = 0x555555cfeca0}, cdr_start = {
    tv_sec = 55042, tv_nsec = 956618094}, cdr_bytes_in = 59, cdr_bytes_out = 318,
  cdr_charging_id = 448}
```

pdp->ctrg:

```
desc = 0x555555790240 <pdpctx_ctrg_desc>, idx = 109, ctr = 0x555555ce9af0}

(gdb) print &pdp->ctrg->list
$51 = (struct llist_head *) 0x555555ce9ad0
(gdb) print *&pdp->ctrg->list
$52 = {next = 0x100100, prev = 0x200200}
(gdb)
```

```
I'm a bit lost in the llist things.
```

History

#1 - 04/24/2019 01:13 PM - laforge

#2 - 04/24/2019 01:20 PM - laforge

On Wed, Apr 24, 2019 at 11:27:54AM +0000, keith [REDMINE] wrote:

```
> desc = 0x555555790240 <pdpctx_ctrg_desc>, idx = 109, ctr = 0x555555ce9af0}
>
> (gdb) print &pdp->ctrg->list
> $51 = (struct llist_head *) 0x555555ce9ad0
> (gdb) print *&pdp->ctrg->list
> $52 = {next = 0x100100, prev = 0x200200}
> (gdb)
>
> I'm a bit lost in the llist things.

the "next = 0x100100, prev = 0x200200" refer to the LLIST_POISON1 / LLIST_POISON2 values
which we use when a llist_head (entry) is deleted from a list.

So in the above example, the pdp->ctrg->list had already been
llist_del()'ed [removed from the list of counter groups within this
class) before, and we just tried to delete it again.

I believe there was a related bug some year[s] ago, so it would be
interesting to hear how current the libosmocore / libgtp / osmo-sgsn
version was.
```

#3 - 04/24/2019 01:29 PM - laforge

On Wed, Apr 24, 2019 at 11:27:54AM +0000, keith [REDMINE] wrote:

> <0023> gtp.c:2653 Packet from 192.168.100.2:2123, length: 14 content: 32 15 00 06 00 00 00 00 59 47 00 00 01 c0 : Unknown PDP context: 0 (expected if gtp_delete_context_req is used)

the library already complains that there's a GTP delete context but we don't know it (wrong TEID?)

> <000e> sgsn_libgtp.c:627 libgtp EOF (type=20, pdp=(nil), cbp=0x555555cfeca0)

Here we see that "struct pdp_t *pdp == NULL" (no pdp context pointer in libgtp). However, we do
have a cbp (callback pointer) which in this case points to a memory location of a "struct sgsn_pdp_ctx *"

> <0012> gprs_sndcp.c:514 SNSM-DEACTIVATE.ind for non-existing TLLI=f38f648c SAPI=3 NSAPI=6

here again it states the TLLI for which we get the pdp delete is not even existing anymore

> pdp in sgsn_pdp_ctx_free():

```
> (gdb) print *pdp
> $6 = {list = {next = 0x100100, prev = 0x200200}, g_list = {next = 0x100100,
>     prev = 0x200200}, mm = 0x555555d1c7d0, destroy_ggsn = 0, ggsn = 0x0,
>   ggsn_list = {next = 0x100100, prev = 0x200200}, ctrg = 0x555555ce9ad0,
>   lib = 0x0, state = PDP_STATE_CR_REQ, type = PDP_TYPE_NONE, address = 0,
>   apn_subscribed = 0x0, nsapi = 6, sapi = 3, ti = 0 '\000', vplmn_allowed = 0,
>   qos_profile_subscr = 0, radio_prio = 0 '\000', timer = {node = {
>       rb_parent_color = 0, rb_right = 0x0, rb_left = 0x0}, list = {next = 0x0,
>       prev = 0x0}, timeout = {tv_sec = 0, tv_usec = 0}, active = 0, cb = 0x0,
>     data = 0x0}, T = 0, num_T_exp = 0, cdr_timer = {node = {
>       rb_parent_color = 93824998219265, rb_right = 0x0, rb_left = 0x0}, list = {
>       next = 0x555555cfed98, prev = 0x555555cfed98}, timeout = {
>       tv_sec = 1556101766, tv_usec = 498923}, active = 0,
>     cb = 0x555555577650 <cdr_pdp_timeout>, data = 0x555555cfeca0}, cdr_start = {
>     tv_sec = 55042, tv_nsec = 956618094}, cdr_bytes_in = 59, cdr_bytes_out = 318,
>   cdr_charging_id = 448}
>
```

From this we learn:

- it has already been deleted from 'list' and 'g_list', as well as ggsn_list
- it doesn't have a libgtp context any more (lib = 0x0)
- the last state was PDP_STATE_CR_REQ (create request), so it never really was fully established

So somehow a PDP context activate was requested (but never completed), and then a delete was requested on the GTP side to the GGSN. And before the GGSN could respond, the data structure about this pdp context was completely removed from all lists, so when the "GGSN has deleted confirmation" message came in, we tried to delete a second time -> boom.

#4 - 04/24/2019 05:23 PM - keith

laforge wrote:

> so it would be
> interesting to hear how current the libosmocore / libgtp / osmo-sgsn
> version was.

osmo-sgsn is compiled this morning from master + https://gerrit.osmocom.org/#/c/osmo-sgsn/+/13764/ which was not yet merged at the time.

$ apt-cache policy libosmocore
libosmocore:
Installed: 1.0.1.127.9838c
Candidate: 1.0.1.127.9838c

$ apt-cache policy libgtp4
libgtp4:
Installed: 1.3.0
Candidate: 1.3.0.2.ffa2

Thanks!

#5 - 05/08/2019 03:46 PM - keith

Lead up to this crash:

```
DMM INFO <0002> gprs_gmm.c:2718 MM(334020508916502/d350687a) -> DEACTIVATE PDP CONTEXT REQ (cause: Regular dea
ctivation)
DGPRS INFO <000e> sgsn_libgtp.c:315 PDP(334020508916502/0) Delete PDP Context
DBSSGP DEBUG <0010> gprs_bssgp.c:392 BSSGP TLLI=0xd350687a Rx UPLINK-UNITDATA
DLLC DEBUG <0011> gprs_llc_parse.c:81 LLC SAPI=1 C   U GEA0 IOV-UI=0x000000 FCS=0x94ef89 CMD=UI DATA
DMM INFO <0002> gprs_gmm.c:1497 MM(334020508916502/d350687a) -> GMM DETACH REQUEST TLLI=0xd350687a type=GPRS d
etach Power-off
DMM INFO <0002> gprs_gmm.c:319 MM(334020508916502/d350687a) Cleaning MM context due to GPRS DETACH REQUEST
DMM NOTICE <0002> gprs_sgsn.c:336 MM(334020508916502/d350687a) Dropping PDP context for NSAPI=5
DGPRS INFO <000e> gprs_sgsn.c:441 PDP(334020508916502/0) Forcing release of PDP context
DSNDCP INFO <0012> gprs_sndcp.c:508 SNSM-DEACTIVATE.ind (lle=0xbe57c8, TLLI=d350687a, SAPI=3, NSAPI=5)
DGPRS ERROR <000e> gprs_gmm.c:2240 PDP(---/0) Stopping PDP timer 3395 but 0 is running
DGPRS INFO <000e> sgsn_libgtp.c:315 PDP(---/0) Delete PDP Context
DMM DEBUG <0002> fsm.c:535 GMM_ATTACH_REQ_FSM(gb_gmm_req)[0xb9e410]{Init}: Deallocated
DGPRS DEBUG <000e> sgsn_libgtp.c:623 libgtp cb_conf(type=20, cause=128, pdp=0x7ffff68b7200, cbp=0x8cee40)
DGPRS INFO <000e> sgsn_libgtp.c:547 PDP(---/0) Received DELETE PDP CTX CONF, cause=128(Request accepted)
DGPRS NOTICE <000e> sgsn_libgtp.c:571 PDP(---/0) Not deactivating SNDCP layer since the MM context is not avai
lable
DLGTP DEBUG <0023> pdp.c:255 Begin pdp_tiddel tid = 5205619805020433
DLGTP DEBUG <0023> pdp.c:262 End pdp_tiddel: PDP found
DLGTP NOTICE <0023> gtp.c:2653 Packet from 192.168.100.2:2123, length: 14 content: 32 15 00 06 00 00 00 00 b0
e7 00 00 01 c0 : Unknown PDP context: 0 (expected if gtp_delet
e_context_req is used)
DGPRS DEBUG <000e> sgsn_libgtp.c:623 libgtp cb_conf(type=20, cause=-1, pdp=(nil), cbp=0x8cee40)
DGPRS ERROR <000e> sgsn_libgtp.c:627 libgtp EOF (type=20, pdp=(nil), cbp=0x8cee40)
DGPRS INFO <000e> sgsn_libgtp.c:547 PDP(---/0) Received DELETE PDP CTX CONF, cause=-1(unknown 0xffffffff)
DGPRS NOTICE <000e> sgsn_libgtp.c:571 PDP(---/0) Not deactivating SNDCP layer since the MM context is not avai
lable

Program received signal SIGSEGV, Segmentation fault.
```

#6 - 05/08/2019 03:56 PM - keith

This is a normal procedure in my lab:

```
20190508154150569 DMM INFO MM(262420000000111/c2d5861b) -> DEACTIVATE PDP CONTEXT REQ (cause: Regular deactiva
tion) (gprs_gmm.c:2718)
20190508154150569 DGPRS INFO PDP(262420000000111/0) Delete PDP Context (sgsn_libgtp.c:315)
20190508154150570 DGPRS DEBUG libgtp cb_conf(type=20, cause=128, pdp=0x7fcf4d5d42e0, cbp=0x5591429aadb0) (sgsn
_libgtp.c:623)
20190508154150570 DGPRS INFO PDP(262420000000111/0) Received DELETE PDP CTX CONF, cause=128(Request accepted)
(sgsn_libgtp.c:547)
20190508154150570 DSNDCP INFO SNSM-DEACTIVATE.ind (lle=0x5591429a8578, TLLI=c2d5861b, SAPI=3, NSAPI=5) (gprs_s
ndcp.c:508)
20190508154150570 DMM INFO MM(262420000000111/c2d5861b) <- DEACTIVATE PDP CONTEXT ACK (gprs_gmm.c:2382)
20190508154150571 DLGTP DEBUG Begin pdp_tiddel tid = 5111000000024262 (pdp.c:255)
20190508154150571 DLGTP DEBUG End pdp_tiddel: PDP found (pdp.c:262)
20190508154151880 DMM INFO MM(262420000000111/c2d5861b) -> GMM DETACH REQUEST TLLI=0xc2d5861b type=GPRS detach
 Power-off (gprs_gmm.c:1497)
20190508154151880 DMM INFO MM(262420000000111/c2d5861b) Cleaning MM context due to GPRS DETACH REQUEST (gprs_g
mm.c:319)
20190508154151881 DGPRS INFO SUBSCR(262420000000111) purging MS subscriber (gprs_subscriber.c:786)
20190508154151881 DGPRS INFO SUBSCR(262420000000111) Sending GSUP, will send: 0c 01 08 62 42 02 00 00 00 11 f1
 09 00 28 01 01  (gprs_subscriber.c:213)
20190508154151881 DMM DEBUG GMM_ATTACH_REQ_FSM(gb_gmm_req)[0x5591429af070]{Init}: Deallocated (fsm.c:535)
20190508154151891 DGPRS INFO GSUP(262420000000111) Completing purge MS (gprs_subscriber.c:580)
```

So it seems like we got (and processed) the MM DETACH before the PDP DEACTIVATE was processed?

#7 - 05/08/2019 04:27 PM - keith

Indeed, we "delete all existing PDP contexts for this MS" when dealing with GMM DETACH.
http://git.osmocom.org/osmo-sgsn/tree/src/gprs/gprs_sgsn.c#n333

#8 - 05/08/2019 05:54 PM - keith

https://gerrit.osmocom.org/#/c/osmo-sgsn/+/13929/

#9 - 05/09/2019 08:19 AM - keith

Running with the above patch, now there is something else revealed;

A double DEACTIVATE PDP CONTEXT REQ:

```
20190508221322398 DMM INFO <0002> MM(334020330861687/c04b2b93) -> DEACTIVATE PDP CONTEXT REQ (cause: Regular d
eactivation) (gprs_gmm.c:2718)
20190508221322399 DGPRS INFO <000e> PDP(334020330861687/0) Delete PDP Context (sgsn_libgtp.c:315)
20190508221322399 DBSSGP DEBUG <0010> BSSGP TLLI=0xc04b2b93 Rx UPLINK-UNITDATA (gprs_bssgp.c:392)
20190508221322400 DLLC DEBUG <0011> LLC SAPI=1 C   U GEA0 IOV-UI=0x000000 FCS=0x58ac1b CMD=UI DATA  (gprs_llc_
parse.c:89)
20190508221322400 DMM INFO <0002> MM(334020330861687/c04b2b93) -> DEACTIVATE PDP CONTEXT REQ (cause: Regular d
eactivation) (gprs_gmm.c:2718)
20190508221322400 DGPRS INFO <000e> PDP(334020330861687/0) Delete PDP Context (sgsn_libgtp.c:315)
20190508221322400 DGPRS DEBUG <000e> libgtp cb_conf(type=20, cause=128, pdp=0x7ffff68ffca0, cbp=0x55555634a8d0
) (sgsn_libgtp.c:622)
20190508221322400 DGPRS INFO <000e> PDP(334020330861687/0) Received DELETE PDP CTX CONF, cause=128(Request acc
epted) (sgsn_libgtp.c:547)
20190508221322400 DSNDCP INFO <0012> SNSM-DEACTIVATE.ind (lle=0x5555565da998, TLLI=c04b2b93, SAPI=3, NSAPI=5)
(gprs_sndcp.c:508)
20190508221322400 DMM INFO <0002> MM(334020330861687/c04b2b93) <- DEACTIVATE PDP CONTEXT ACK (gprs_gmm.c:2382)
20190508221322400 DLGTP DEBUG <0023> Begin pdp_tiddel tid = 5786168033020433 (pdp.c:255)
20190508221322400 DLGTP DEBUG <0023> End pdp_tiddel: PDP found (pdp.c:262)
20190508221322400 DLGTP NOTICE <0023> Packet from 192.168.100.2:2123, length: 14 content: 32 15 00 06 00 00 00
 00 cb 7d 00 00 01 c0 : Unknown PDP context: 0 (expected i$
 gtp_delete_context_req is used) (gtp.c:2653)
20190508221322400 DGPRS DEBUG <000e> libgtp cb_conf(type=20, cause=-1, pdp=(nil), cbp=0x55555634a8d0) (sgsn_li
bgtp.c:622)
20190508221322401 DGPRS ERROR <000e> libgtp EOF (type=20, pdp=(nil), cbp=0x55555634a8d0) (sgsn_libgtp.c:626)
20190508221322401 DGPRS INFO <000e> PDP(334020330861687/0) Received DELETE PDP CTX CONF, cause=-1(unknown 0xff
ffffff) (sgsn_libgtp.c:547)
```

```
20190508221322401 DSNDCP INFO <0012> SNSM-DEACTIVATE.ind (lle=0x5555565da998, TLLI=c04b2b93, SAPI=3, NSAPI=5)
(gprs_sndcp.c:508)
20190508221322401 DSNDCP ERROR <0012> SNSM-DEACTIVATE.ind for non-existing TLLI=c04b2b93 SAPI=3 NSAPI=5 (gprs_
sndcp.c:514)
20190508221322401 DMM INFO <0002> MM(334020330861687/c04b2b93) <- DEACTIVATE PDP CONTEXT ACK (gprs_gmm.c:2382)

Program received signal SIGSEGV, Segmentation fault.
```

#10 - 05/10/2019 10:46 AM - keith

After quite some hours, same thing again, different phone.

```
20190509191724159 DMM INFO <0002> MM(334020160307475/f05e7363) -> DEACTIVATE PDP CONTEXT REQ (cause: Regular d
eactivation) (gprs_gmm.c:2718)
20190509191724159 DGPRS INFO <000e> PDP(334020160307475/0) Delete PDP Context (sgsn_libgtp.c:315)
20190509191724159 DBSSGP DEBUG <0010> BSSGP TLLI=0xf05e7363 Rx UPLINK-UNITDATA (gprs_bssgp.c:392)
20190509191724161 DLLC DEBUG <0011> LLC SAPI=1 C   U GEA0 IOV-UI=0x000000 FCS=0xe057bd CMD=UI DATA  (gprs_llc_
parse.c:89)
20190509191724161 DMM INFO <0002> MM(334020160307475/f05e7363) -> DEACTIVATE PDP CONTEXT REQ (cause: Regular d
eactivation) (gprs_gmm.c:2718)
20190509191724162 DGPRS INFO <000e> PDP(334020160307475/0) Delete PDP Context (sgsn_libgtp.c:315)
20190509191724162 DGPRS DEBUG <000e> libgtp cb_conf(type=20, cause=128, pdp=0x7ffff6abe020, cbp=0x555556dcb460
) (sgsn_libgtp.c:622)
20190509191724164 DGPRS INFO <000e> PDP(334020160307475/0) Received DELETE PDP CTX CONF, cause=128(Request acc
epted) (sgsn_libgtp.c:547)
20190509191724164 DSNDCP INFO <0012> SNSM-DEACTIVATE.ind (lle=0x555555cf30c8, TLLI=f05e7363, SAPI=3, NSAPI=5)
(gprs_sndcp.c:508)
20190509191724164 DMM INFO <0002> MM(334020160307475/f05e7363) <- DEACTIVATE PDP CONTEXT ACK (gprs_gmm.c:2382)
20190509191724164 DLGTP DEBUG <0023> Begin pdp_tiddel tid = 5574703061020433 (pdp.c:255)
20190509191724164 DLGTP DEBUG <0023> End pdp_tiddel: PDP found (pdp.c:262)
20190509191724164 DLGTP NOTICE <0023> Packet from 192.168.100.2:2123, length: 14 content: 32 15 00 06 00 00 00
 00 dc c2 00 00 01 c0 : Unknown PDP context: 0 (expected if gtp_delete_context_req is used) (gtp.c:2653)
20190509191724164 DGPRS DEBUG <000e> libgtp cb_conf(type=20, cause=-1, pdp=(nil), cbp=0x555556dcb460) (sgsn_li
bgtp.c:622)
20190509191724164 DGPRS ERROR <000e> libgtp EOF (type=20, pdp=(nil), cbp=0x555556dcb460) (sgsn_libgtp.c:626)
20190509191724165 DGPRS INFO <000e> PDP(334020160307475/0) Received DELETE PDP CTX CONF, cause=-1(unknown 0xff
ffffff) (sgsn_libgtp.c:547)
20190509191724165 DSNDCP INFO <0012> SNSM-DEACTIVATE.ind (lle=0x555555cf30c8, TLLI=f05e7363, SAPI=3, NSAPI=5)
(gprs_sndcp.c:508)
20190509191724165 DSNDCP ERROR <0012> SNSM-DEACTIVATE.ind for non-existing TLLI=f05e7363 SAPI=3 NSAPI=5 (gprs_
sndcp.c:514)
20190509191724165 DMM INFO <0002> MM(334020160307475/f05e7363) <- DEACTIVATE PDP CONTEXT ACK (gprs_gmm.c:2382)

Program received signal SIGSEGV, Segmentation fault.
```

#11 - 05/18/2019 11:08 AM - keith

In osmo-ggsn gtp.c:gtp_delete_pdp_conf()

```
    /* Find the context in question. It may not be available if gtp_delete_context_req
     * was used and as a result the PDP ctx was already freed */
    if (pdp_getgtp1(&pdp, get_tei(pack))) {
        gsn->err_unknownpdp++;
        GTP_LOGPKG(LOGL_NOTICE, peer, pack, len,
                "Unknown PDP context: %u (expected if gtp_delete_context_req is used)\n",
                 get_tei(pack));
        if (gsn->cb_conf)
            gsn->cb_conf(type, EOF, NULL, cbp);
        return EOF;
    }
```

In the case of this bug, get_tei(pack) = 0 as already shown in the log above, the condition returns EOF and the code execs

So is this wrong? Should we call the cb_conf with NULL pdp at all?

#12 - 08/15/2019 10:56 AM - pespin

- Category set to GTP interface

- Status changed from New to In Progress

- Assignee set to pespin

Hi,

I'm sorry to say I missed this issue until now.

According to your latest log messages:

```
20190509191724159 DMM INFO <0002> MM(334020160307475/f05e7363) -> DEACTIVATE PDP CONTEXT REQ (cause: Regular d
eactivation) (gprs_gmm.c:2718)
20190509191724159 DGPRS INFO <000e> PDP(334020160307475/0) Delete PDP Context (sgsn_libgtp.c:315)
```

The delete PDP Context message is level INFO, which means gtp_delete_context_req2() is already being used (and pdp free should be delayed) (03dc773e081c608524a0907cb1d867f792156157).

Then it receives again a DEACTIVE PDP ctx and goes through the gtp_delete_context_req2() again (ideally should do nothing because the message is already in queue, need to check that):

```
20190509191724161 DMM INFO <0002> MM(334020160307475/f05e7363) -> DEACTIVATE PDP CONTEXT REQ (cause: Regular d
eactivation) (gprs_gmm.c:2718)
20190509191724162 DGPRS INFO <000e> PDP(334020160307475/0) Delete PDP Context (sgsn_libgtp.c:315)
```

Then first DELETE CONFIRMATIOn arrives and everything's freed fine:

```
20190509191724162 DGPRS DEBUG <000e> libgtp cb_conf(type=20, cause=128, pdp=0x7ffff6abe020, cbp=0x555556dcb460
) (sgsn_libgtp.c:622)
20190509191724164 DGPRS INFO <000e> PDP(334020160307475/0) Received DELETE PDP CTX CONF, cause=128(Request acc
epted) (sgsn_libgtp.c:547)
20190509191724164 DSNDCP INFO <0012> SNSM-DEACTIVATE.ind (lle=0x555555cf30c8, TLLI=f05e7363, SAPI=3, NSAPI=5)
(gprs_sndcp.c:508)
20190509191724164 DMM INFO <0002> MM(334020160307475/f05e7363) <- DEACTIVATE PDP CONTEXT ACK (gprs_gmm.c:2382)
20190509191724164 DLGTP DEBUG <0023> Begin pdp_tiddel tid = 5574703061020433 (pdp.c:255)
20190509191724164 DLGTP DEBUG <0023> End pdp_tiddel: PDP found (pdp.c:262)
```

Then a second confirmation arrives (immediatelly after the first one), which means it's not a retransmission and osmo-sgsn sent 2 different Delete Requests and this should not happen, so first bug! It for sure sends 2 of them, probably with different seq number (or different wrong pdp ctx?), otherwise the receive conf queue would discard the 2nd one after the first one is handled. As a result, since cbp was already freed before (delete_pdp_conf() called sgsn_pdp_ctx_free()), the address still kept in the receive conf queue is pointing to already-freed memory, and will eventually crash when used by sgsn:

```
20190509191724164 DLGTP NOTICE <0023> Packet from 192.168.100.2:2123, length: 14 content: 32 15 00 06 00 00 00
 00 dc c2 00 00 01 c0 : Unknown PDP context: 0 (expected if gtp_delete_context_req is used) (gtp.c:2653)
20190509191724164 DGPRS DEBUG <000e> libgtp cb_conf(type=20, cause=-1, pdp=(nil), cbp=0x555556dcb460) (sgsn_li
bgtp.c:622)
20190509191724164 DGPRS ERROR <000e> libgtp EOF (type=20, pdp=(nil), cbp=0x555556dcb460) (sgsn_libgtp.c:626)
20190509191724165 DGPRS INFO <000e> PDP(334020160307475/0) Received DELETE PDP CTX CONF, cause=-1(unknown 0xff
ffffff) (sgsn_libgtp.c:547)
20190509191724165 DSNDCP INFO <0012> SNSM-DEACTIVATE.ind (lle=0x555555cf30c8, TLLI=f05e7363, SAPI=3, NSAPI=5)
(gprs_sndcp.c:508)
20190509191724165 DSNDCP ERROR <0012> SNSM-DEACTIVATE.ind for non-existing TLLI=f05e7363 SAPI=3 NSAPI=5 (gprs_
sndcp.c:514)
20190509191724165 DMM INFO <0002> MM(334020160307475/f05e7363) <- DEACTIVATE PDP CONTEXT ACK (gprs_gmm.c:2382)
Program received signal SIGSEGV, Segmentation fault.
```

Interestingly, new osmo-sgsn/libgtp already accounts for related possibility (pdp being NULL), in 1cde2c169162de3773ccc49b0408a330d61be3d7:

```
-                "Unknown PDP context: %u (expected if gtp_delete_context_req is used)\n",
+                "Unknown PDP context: %u (expected if gtp_delete_context_req is used or pdp ctx was freed man
ually before response)\n",
```

So the problem here is transmit queue is being filled twice (and somehow it doesn't complain) with pdp ctx delete request with same CBP, which should not happen.

It would be great if you can try to reproduce the issue with new releases osmo-sgsn (1.5.0) and osmo-ggsn (1.4.0), master is fine too, since I have been refactoring and fixing related code, like 1cde2c169162de3773ccc49b0408a330d61be3d7 (fixing tear down related bugs). I will try to add better debugging for this kind of situation too in master.

Extra Self-Reminder: I need to check if pdp_freepdp() (or similar) removes pending messages for that pdp from the message queue used by gtp_conf().

#13 - 08/15/2019 03:54 PM - pespin

Generic issue is basically:
1- SGSN enqueues X messages (no matter the message type) for pdp ctx A and are transmitted but not yet ACKED. They are queue in TX queue for retransmission if needed.
2- SGSN enqueues a DeleteCtxRequest for pdp ctx A
3- SGSN receives a DeleteCtxAccept fro pdp ctx A, and cbp (sgsn_pdp_ctx) associated to pdp ctx A is freed, but messages in Tx queue still keep that pointer.
4- SGSN receives any of the X messages for pdp ctx A -> libgtp will forward pdp=NULL cbp=already-freed-memory and osmo-sgsn will crash eventuall when using cbp.

Note this can happen even if the ggsn decides to send no ACK for the messages after receiving DeleteCtxRequest (due to packet reordering in the network).

So several problems:

- messages enqueued in Tx queue don't hold directly a reference to the pdp ctx, they instead look it up upon receival through gtp_pdp_getgtp1(get_tei(packet)).
- upon pdp_freepdp(), currently the Tx queue packets related to that pdp ctx are not removed -> it could be done, because the pdp contains the tei so we can iterate over the hashtable and check for packets containing same tei and drop them, but it's probably quite costly (all in-air packets need to be checked).
- Another approximation would be having a per-pdp queue...
- We could simply let those message be in there (and be eventually retransmitted and acked but without announcing it to upper layers), but then they could interfer with new pdp ctx which contain same tei and are created later on?

#14 - 08/15/2019 04:22 PM - pespin

Self-note: match of address from pdp against qmsg->peer can be done:

```
struct in_addr addr;
gsna2in_addr(&addr, &pdp->gsnrc);

struct sockaddr_in addr;
memset(&addr, 0, sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_addr = *inetaddr;
#if defined(__FreeBSD__) || defined(__APPLE__)
addr.sin_len = sizeof(addr);
#endif
if (ver == 0)
    addr.sin_port = htons(GTP0_PORT);
else if (ver == 1)
    addr.sin_port = htons(GTP1C_PORT);

memcpy(&qmsg->peer, &addr, sizeof(addr));
```

#15 - 08/26/2019 08:59 AM - pespin

- Status changed from In Progress to Feedback

- % Done changed from 0 to 90

Fix available here :
https://gerrit.osmocom.org/c/osmo-ggsn/+/15228

Once merged this ticket can be closed.

#16 - 08/27/2019 06:04 PM - pespin

- Status changed from Feedback to Resolved

- % Done changed from 90 to 100

Merged, closing.