libosmo-sccp + libosmo-sigtran - Bug #3871

osmo_scu_prim conn_id should be scoped per-user and direction, and should not correlate with the local-reference spoken on the SCCP wire

03/28/2019 07:58 PM - neels

Status:	In Progress	**Start date:**	03/28/2019
Priority:	Urgent	**Due date:**	
Assignee:	osmith	**% Done:**	80%
Category:			
Target version:			
Spec Reference:			

Description

In summary:

- separate osmo_scu_prim.*.conn_id from osmo_sccp_inst->next_id
- separate each osmo_sccp_user's conn_ids number spaces
- separate incoming from outgoing osmo_scu_prim.*.conn_id number spaces

Details:

The osmo_sccp_instance.next_id is responsible for local-reference IDs on the SCCP wire.
Currently, this same next_id number space is also used towards the SCU user, i.e. as osmo_scu_prim.*.conn_id.
Not only should this conn_id be scoped separately from the osmo_sccp_instance.next_id, but also be a distinct number space per SCU user.

struct sccp_connection in sccp_scoc.c is the place where an SCCP local-reference is correlated to an osmo_sccp_user.conn_id, it must store these IDs separately.
The number space for the osmo_scu_prim.*.conn_id should be guarded by some "next_id" counter in osmo_sccp_user.
Incoming and outgoing conn_ids must be made sure to not collide, best divide the number space in two for incoming and outgoing conn_ids.

For example, in osmo-msc, there may be two SCCP users connected to the same osmo_sccp_instance: one for OSMO_SCCP_SSN_BSSAP (= GERAN-A = 2G) and one for OSMO_SCCP_SSN_RANAP (= UTRAN-Iu = 3G).
Implementations using the SCU API might be completely separate, and it would be a "scoping violation" if these distinct users have to negotiate with each other about unused osmo_scu_prim conn_ids.

Another aspect is that there may be incoming SCCP connections and outgoing SCCP connections.
As an example:

From the perspective of osmo-msc, usually the BSC connects to the MSC:

```
   BSC               MSC AS                                  SCCP-User
    | ---N_CONNECT--> | ---osmo_scu_prim.connect.conn_id--> | BSSAP implementation stores new conn
_id.
    |                 |                                     |
    | <--N_DATA-----> | <---osmo_scu_prim.data.conn_id----> |
```

But in case of an inter-BSC HO, the MSC initiates a connection to the BSC instead:

```
   BSC               MSC AS                                  SCCP-User
    | <--N_CONNECT--- | <--osmo_scu_prim.connect.conn_id--- | BSSAP implementation *invents* new c
onn_id.
    |                 |                                     |
    | <--N_DATA-----> | <---osmo_scu_prim.data.conn_id----> |
```

(Note, this is only relevant for Connection-Oriented Initial messages, i.e. Layer 3 Complete (incoming to MSC) and Handover Request (outgoing from MSC).
For example, Paging is done by a connection-less message that has no conn_id at all, after which the BSC establishes a connection when the MS has responded.)

To keep sane layer separation, it is important to maintain the osmo_scu_prim struct as the **only** communication interface between

the SCCP-User and the SCCP instance.
Hence we cannot iterate all existing conn_ids and pick an unused one: an incoming SCCP connection might asynchronously pick the exact same conn_id.
In practice, this cannot happen right now, but the scoping should be such that it would be possible to place the SCCP User in a separate process.

So, a proposal is that for incoming connections, the libosmo-sccp creates new osmo_scu_prim.*.conn_id in the number space 0..0x7fffffff for incoming N-CONNECT,
while for outgoing N-CONNECT, the SCCP-User implementation creates new osmo_scu_prim.*.conn_id in the number space 0x80000000..0xffffffff. (Or rather, use the lowest bit to separate for shorter logging)
There should be osmo_sccp_user_* API to fetch an unused outgoing SCCP-User conn_id, so SCCP-Users don't need to implement their own.

Related issues:

Related to OsmoMSC - Feature #3618: Inter-MSC hand-over support	**In Progress**	**10/02/2018**	
Related to OsmoMSC - Feature #1609: Inter-BSC hand-over is missing (MSC side)	**Resolved**	**11/21/2016**	**11/21/2016**
Blocks OsmoBSC - Feature #3872: osmo_scu_prim.conn_id should be picked from a...	**New**	**03/28/2019**	

History

#1 - 03/28/2019 07:59 PM - neels

- Related to Feature #3618: Inter-MSC hand-over support added

#2 - 03/28/2019 07:59 PM - neels

- Related to Feature #1609: Inter-BSC hand-over is missing (MSC side) added

#3 - 03/28/2019 08:02 PM - neels

(a tweak could be to use the lowest bit as incoming/outgoing separation, so that conn_ids take up less space in logging: GERAN-A-0x80000001 is a lot longer than GERAN-A-0x2)

#4 - 03/28/2019 08:06 PM - neels

- Blocks Feature #3872: osmo_scu_prim.conn_id should be picked from a distinct number space for outgoing and incoming N-CONNECT added

#5 - 03/28/2019 08:07 PM - neels

- Subject changed from osmo_scu_prim conn_id should be scoper per-user and should not correlate with the local-reference spoken on the SCCP wire to osmo_scu_prim conn_id should be scoped per-user and direction, and should not correlate with the local-reference spoken on the SCCP wire

#6 - 03/28/2019 08:10 PM - neels

- Description updated

#7 - 03/29/2019 10:10 AM - laforge

On Thu, Mar 28, 2019 at 08:02:50PM +0000, neels [REDMINE] wrote:

> (a tweak could be to use the lowest bit as incoming/outgoing separation, so that conn_ids take up less space in logging: GERAN-A-0x80000001 is a lot longer than GERAN-A-0x2)

you can also simply adjust the logging function to say I1 for Inbound 1 and O1 for Outbound 1.

#8 - 03/29/2019 10:10 AM - laforge

The description looks very much correct, thanks a lot, Neels.

Some minor additions:

- please put safe-guards about the incoming/outgoing ID range splits at all related places. A program wrongly using a connection ID from the "incoming" range in an outgoing N-CONNECT.req primitive should fail explicitly, rather than "succeeding by accident in most of the cases".
- feel free to have a look at ITU-T Q.711 Section 6.1.1 for a description of the SCCP-SAP primitives, in case there's any doubt about how they work in general.

#9 - 03/29/2019 10:39 AM - osmith

- Status changed from New to In Progress

#10 - 03/29/2019 05:18 PM - neels

osmith have updated the WIP patch proposal again after our call

laforge Please confirm that this is the right approach: https://gerrit.osmocom.org/c/libosmo-sccp/+/13466

#11 - 03/30/2019 10:40 AM - laforge

> Please confirm that this is the right approach: https://gerrit.osmocom.org/c/libosmo-sccp/+/13466

I added review comments to gerrit. I think some parts are right, but others are not. I don't think

- there should be any renaming of the existing conn_id to include "user", nor any related functions. Connection IDs are always between SAP provider and SAP user, by very definition.
- libosmo-sccp can ever be used to allocate "user side" (sap_down) connection identifiers reliably. It must check for wraps/duplicates, and it can only do so by accessing whatever state structures the user application has, such as e.g. the list of subscriber_conns or whatever is attached to the SCCP connection.

Regards,
Harald

#12 - 04/03/2019 01:59 PM - osmith

I've updated the patch with my current understanding.

laforge: can you take another look at it?

https://gerrit.osmocom.org/#/q/topic:conn-id-scoping

#13 - 04/04/2019 07:13 AM - osmith

- % Done changed from 0 to 80

So the approach is right. :)

I'm testing the patch currently, and it does not work out of the box (OsmoBSC says "No unused local reference available for user msc-0"). Working on an update.

#14 - 04/04/2019 08:25 AM - osmith

Further testing reveals a problem with the current patch.

Incoming messages are not resolved by the conn_id anymore, like this snippet from master:

```
        conn_id = xua_msg_get_u32(xua, SUA_IEI_DEST_REF);
        conn = conn_find_by_id(inst, conn_id);
```

Instead they are resolved from the local_ref and user now.

```
static struct sccp_connection *sccp_find_connection(struct osmo_sccp_instance *inst, struct xua_msg *xua)
{
    uint32_t local_ref = xua_msg_get_u32(xua, SUA_IEI_DEST_REF);
    struct osmo_sccp_user *user = sccp_find_user(inst, xua);

    if (user)
        return conn_find_by_local_ref(user, local_ref);
    return NULL;
}
...
conn = sccp_find_connection(inst, xua);
```

sccp_find_user() needs SUA_IEI_DEST_ADDR to be set:

```
/* Find a SCCP user for given SUA message (based on SUA_IEI_DEST_ADDR */
static struct osmo_sccp_user *sccp_find_user(struct osmo_sccp_instance *inst,
                        struct xua_msg *xua)
{
    int rc;
    struct osmo_sccp_addr called_addr;

    rc = sua_addr_parse(&called_addr, xua, SUA_IEI_DEST_ADDR);
    if (rc < 0) {
        LOGP(DLSCCP, LOGL_ERROR, "Cannot find SCCP User for XUA "
            "Message %s without valid DEST_ADDR\n",
            xua_hdr_dump(xua, &xua_dialect_sua));
        return NULL;
    }
...
```

But we don't set it for all messages, and so it fails here:

```
OsmoMSC:
Cannot find SCCP User for XUA Message CO:RELRE without valid DEST_ADDR
Cannot find connection for local reference 4

OsmoBSC:
Cannot find SCCP User for XUA Message CO:RELCO without valid DEST_ADDR
Cannot find connection for local reference 4
```

IIRC we can't simply start sending SUA_IEI_DEST_ADDR in RELRE and RELCO, because then it is not backwards compatible.

laforge, how to proceed here?

#15 - 04/04/2019 08:36 AM - laforge

On Thu, Apr 04, 2019 at 08:25:56AM +0000, redmine@lists.osmocom.org wrote:

> Further testing reveals a problem with the current patch.

:/

> sccp_find_user() needs SUA_IEI_DEST_ADDR to be set:

This is unfortuantely not possible, as the Destination Address is not present
in a number of messages, as you found out.

> IIRC we can't simply start sending SUA_IEI_DEST_ADDR in RELRE and RELCO, because then it is not backwards compatible.

It's not about "backwards compatibility" but it's about how SCCP (and SUA) are specified. We are
interoperating with other implementations and we cannot change the protocol.

> laforge, how to proceed here?

I don't know off my head. I'll try to investigate but certainly won't have time before
Sunday or Monday, sorry.