libosmo-ranap - Bug #3420

ranap_transp_assoc_decode() decodes X.213 NSAP address headers as IP address

07/26/2018 01:51 PM - neels

Status:	Stalled	**Start date:**	07/26/2018
Priority:	Normal	**Due date:**	
Assignee:	osmith	**% Done:**	30%
Category:			
Target version:			
Spec Reference:			

Description

The SysmoCell5000 series sends RAB-AssignmentResponse with X.213 NSAP addresses, which get parsed by ranap_transp_assoc_decode().
These contain a three byte header followed by the IP address, typically it is 0x350001 followed by the Ipv4 address in hex.

ranap_transp_assoc_decode() however starts at the header and decodes as 53.0.1.N where N is the first octet of the actual IPv4 address, e.g. 192.
Hence switching RTP streams to the cell fails, since obviously 53.0.1.192 is not the correct IP address.

a) fix the check for the address length, should be len == 7, not len > 7.

b) generally be stricter about the lengths.

History

#1 - 07/26/2018 01:53 PM - neels

osmo-iuh iu_helpers.c

```
/* decode a network address as used inside ASN.1 encoded Iu interface protocols */
int ranap_transp_layer_addr_decode(char *addr, unsigned int addr_len,
                                   const RANAP_TransportLayerAddress_t *trasp_layer_addr)
{
        unsigned char *buf;
        int len;
        const char *rc;

        buf = trasp_layer_addr->buf;
        len = trasp_layer_addr->size;

        if (buf[0] == 0x35 && len > 7)
                rc = inet_ntop(AF_INET, buf + 3, addr, addr_len);
        else if (len > 3)
                rc = inet_ntop(AF_INET, buf, addr, addr_len);
        else
                return -EINVAL;

        if (!rc)
                return -EINVAL;

        return 0;
}
```

This is so wrong on various levels.

#2 - 08/20/2018 03:34 PM - neels

- Checklist item [x] fix the check for the address length, should be len == 7, not len > 7 set to Done

the current failure is mended by trivial https://gerrit.osmocom.org/#/c/osmo-iuh/+/10545 ,
other decoding improvements remain pending.

#3 - 08/20/2018 03:34 PM - neels

- Status changed from New to Stalled

- % Done changed from 0 to 30

#4 - 09/30/2018 08:37 AM - laforge

- Assignee changed from neels to osmith

#5 - 09/30/2018 09:46 PM - neels

That fix which changed the condition to '== 7' actually broke nano3G voice calls.
To hackily accomodate both sysmocell and nano3G, it must be '>= 7' instead :/
https://gerrit.osmocom.org/#/c/osmo-iuh/+/11167

#6 - 10/25/2018 12:20 PM - osmith

- Status changed from Stalled to In Progress

#7 - 10/25/2018 02:37 PM - osmith

neels wrote in the commit message of the last patch:

> A proper X.213 NSAP decoding would still be more welcome than this patching back and forth, but this is (another) quick fix without spending too much time on it.

So I looked into what a proper X.213 NSAP decoding would be. The reference for having IPv4 encoded in NSAPA is defined in RFC4538 § 3.1.

It matches exactly what you have found, the 0x350001 followed by the IPv4 is the standard. The header for IPv6 is 0x350000.

=> We can update the code to check for 0x350001 now, not only for 0x35.

The length is always 20 bytes in the document. However, as the address is stored in Osmocom as transportLayerAddress, it gets cut off and that sort of implies what kind of address it has been. I did not find yet where it gets saved and cut up initially (will look into that tomorrow, that is probably the key to why we have different lengths for NSAP addresses that should have the same length), but I did find another piece of code that extracts the IPv4 from the transportLayerAddress in osmo-sgsn/src/gprs/sgsn_libgtp.c:sgsn_ranap_rab_ass_resp():

```
switch (item->transportLayerAddress->size) {
        case 7:
            /* It must be IPv4 inside a X213 NSAP */
            memcpy(pdp->lib->gsnlu.v, &item->transportLayerAddress->buf[3], 4);
            break;
        case 4:
            /* It must be a raw IPv4 address */
            memcpy(pdp->lib->gsnlu.v, item->transportLayerAddress->buf, 4);
            break;
        case 16:
            /* TODO: It must be a raw IPv6 address */
        case 19:
            /* TODO: It must be IPv6 inside a X213 NSAP */
        default:
            LOGP(DRANAP, LOGL_ERROR, "RAB Assignment Resp: Unknown "
                "transport layer address size %u\n",
                item->transportLayerAddress->size);
            return -1;
        }
```

#8 - 10/29/2018 12:43 PM - osmith

Summary

IPv4 Addresses get written into RANAP_TransportLayerAddress_t in various formats. The ranap_transp_layer_addr_decode() and sgsn_ranap_rab_ass_resp() functions interpret it as follows:

	Source	length	header bytes	bytes containing the IPv4
1.	(unknown)	4 (or more)	none	0-3
2.	nano3g	> 7 (probably 20)	0x350001	4-7
3.	sysmocell	7	0x350001	4-7

So this is a mixture of writing the real IPv4 address (1.) (possibly with unneeded bytes that follow) and full/cut-off versions of IPv4 Address Encoding in an NSAPA (2. and 3.). IPv6 addresses are not interpreted by both functions (there is a standard for embedding them in NSAP addresses).

A lot of the osmo-iuh C code is generated from ASN1 files. The relevant datatype seems to be defined in RANAP-IEs.asn:

```
TransportLayerAddress           ::= BIT STRING (SIZE (1..160, ...))
```

How to continue?

I thought the TransportLayerAddress would get written somewhere in the Osmocom code, and we could simply write the real address without a header and with a fixed length, and it would be fixed. But it seems that the addresses get sent from the nano3g, sysmocell or other devices instead. Does this mean, where the format does not match the spec, that this is a bug in these devices, and we can't really do anything about it instead of guess the address that has been sent like we are doing now?

Maybe we could move that guessing code into a central location instead of having it twice in both functions?

#9 - 10/29/2018 12:43 PM - osmith

- Status changed from In Progress to Stalled

#10 - 10/29/2018 02:28 PM - neels

We need to be able to parse various formats coming from various 3G cells.

We also need to encode either raw IPv4 or a valid X.213 depending on osmo-msc.cfg.

Parsing in one central location sounds good, I wasn't aware that we had two different places?

(Would be even nicer to autodetect which format we should send, but I think that's not easily possible; during RAB assignment, we first send an address and then get one back, so we can't mimick the cell's format)