OsmoMSC - Feature #2397

let osmo-msc record location area from location update for LAC wide paging

07/24/2017 08:13 PM - dexter

Status:	Resolved	**Start date:**	07/24/2017
Priority:	Normal	**Due date:**	
Assignee:	neels	**% Done:**	100%
Category:			
Target version:			
Resolution:			

Description

The MSC is responsible to issue the paging commands to the BSS (BSC), it should know which BSC has which location area under its control in order to support LAC wide paging at some point. Since we automatically discover the BSCs we also need to discover automatically which location are is behind which MSC. The location area becomes known with the first location update a random phone is doing. We could just update a list with location areas we have got location area updates from. This list can than be used to find out which BSC, handles which location area.

Related issues:

Related to OsmoMSC - Feature #2289: implement AoverIP (OsmoMSC side)	**Closed**	**05/24/2017**	
Blocks OsmoMSC - Feature #1609: Inter-BSC hand-over is missing (MSC side)	**Resolved**	**11/21/2016**	**11/21/2016**

History

#1 - 07/25/2017 08:02 AM - dexter

- Related to Feature #2289: implement AoverIP (OsmoMSC side) added

#2 - 08/07/2017 09:28 PM - dexter

neels: You solved a similar problem for 3G. Is the described idea above coherent to your solution?

#3 - 08/10/2017 04:00 PM - neels

In 3G we pass a subscriber's currently known LAC (in RAM in the VLR) to ranap_iu_page_cs(), which calls iu_page():
https://git.osmocom.org/osmo-iuh/tree/src/iu_client.c#n611
This iterates the connected RNCs (like BSC but for 3G) and transmits a paging request to its sccp address.

Each RNC's LAC is indeed picked up from a UE's InitialUE message, i.e. when a subscriber first shows up.
https://git.osmocom.org/osmo-iuh/tree/src/iu_client.c#n293

You'll see in iu_rnc_register() that we're so far undecided what to do if a UE sends a LAC that mismatches the LAC we've previously recorded for the same RNC:
https://git.osmocom.org/osmo-iuh/tree/src/iu_client.c#n134

If we're following 3G, a similar way could be implemented for A.

I believe there has once been talk about a common table of LACs across 2G BSCs and 3G RNCs, but the way the iu_client.c (recently moved to osmo-iuh.git) looks, it would be rather intricate to combine the two sensibly. We'd have to move the RNC lookup somehow back to osmo-msc.git, while an osmo-msc without Iu support can't really even know the rnc data type... Plus the SGSN uses the same code to page on IuPS, so then again we can't really move it out of osmo-iuh without creating code dup... I guess not worth the trouble to combine.

In an aside, with 3G, once we've sent the paging to an RNC with a given LAC, that RNC is practically our single OsmoHNBGW, which again needs to know which Iuh link corresponds to which LAC. OsmoHNBGW so far doesn't collect LACs at all, so in 3G we anyway end up paging on all Iuh links ;) (For OsmoHNBGW, the standard procedure is to page everywhere when the first LAC didn't work, so it's not really that harmful I guess).

#4 - 12/10/2017 07:55 PM - laforge

- Project changed from OpenBSC to OsmoMSC

- Category deleted (OpenBSC)

#5 - 12/10/2017 07:55 PM - laforge

- Priority changed from Low to Normal

#6 - 03/29/2018 02:37 PM - neels

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

#7 - 04/03/2018 04:56 PM - dexter

- Status changed from New to In Progress

- % Done changed from 0 to 20

I made some draft implementation that collects lac information from layer3compl messages. The context information is collected in a list in struct gsm_network. The reason why it is separate from the bsc_context information is because the same list also should be used by the Iu part in the future as well. I also added A vty command to introspect the collected information. The table does not necessarily have to be filled by the layer3compl messages. It also can be pre-filled manually using the VTY.

See also: pmaier/lac

#8 - 04/03/2018 05:40 PM - laforge

On Tue, Apr 03, 2018 at 04:56:09PM +0000, dexter [REDMINE] wrote:

> The reason why it is separate from the bsc_context information is
> because the same list also should be used by the Iu part in the future
> as well.

I'm really not sure it's worth adding yet another list, sorry :/

The rationale you are giving is one for further merging the A and Iu code. There is really nothing different from collecting bsc_contexts to collecting the analogous for Iu. The MSC also needs a list of RNCs with their LAC/RAC, just like it needs it for A.

#9 - 04/09/2018 08:15 PM - dexter

We can change the way of storing the LAC information of course, but I suggest to discuss the problem tomorrow together with neels.

Neels and me worked out the following requirements:

- Collect LAC / Pointcode tuples automatically during L3 COMPL (and the Iu equivalent)
- Allow making manual entries via VTY
- Allow deleting manual (and automatically) created entries via VTY
- Safe only entries that were entered manually

Probably we can have it all in one list. However, we need some more flags, but in general that is a good idea. I think the way to go here is to first unify the structs that handle A and Iu and then we add unify the lists.

#10 - 05/28/2018 09:06 AM - dexter

- Status changed from In Progress to Stalled

#11 - 06/18/2018 07:28 AM - dexter

- Status changed from Stalled to Feedback

- Assignee changed from dexter to neels

neels: Whats your opinion about this?

See also: pmaier/lac

#12 - 07/02/2018 11:40 AM - neels

- Assignee changed from neels to dexter

There is unfortunately a practical obstacle to freely changing the code:
So far we manage the RNCs for Iu in iu_client.c, which is linked from libosmo-ranap.
We're using the same code in osmo-sgsn, so if we want to continue to share code there the sgsn needs to come along.

iu_client.c is quite useful in composing and receiving RANAP messages. That's a bunch that should remain shared. But from today's point of view the list management of RNCs is out of scope and getting in the way.

So maybe we can separate the list-of-RNCs part out of there. It seems the touching points are very few:

- On InitialUE: alloc an RNC entry
- On Paging: look up an RNC entry

We use the talloc'd struct ranap_iu_rnc as talloc ctx for the struct ue_conn_ctx. That's not really necessary, we could just parent all ue_conn_ctx to the talloc_iu_ctx (see ranap_iu_init()).

The only time we actually need anything from the struct ranap_iu_rnc is on paging, and all that's needed for actual paging is the SCCP address.
So far we feed the LAC+RAC and the lookup happens in iu_client.c;
it would be easy to just look up the SCCP address "in our own list" and pass that to iu_page() directly.

The SCCP address of the RNC is the same that we store in the ue_conn_ctx.
The global_iu_recv_cb() (implemented by MSC and SGSN) already receives the ue_conn_ctx as msg->dst, and the ra_id as arg, so it can already record the LAC-RAC <-> SCCP address mapping.

The remaining question would be the RNC Id. We don't seem to be using that anywhere at all (besides for sanity checking of incoming requests, which is optional).
For sanity / logging, I guess we should still remember the RNC Id somewhere, and could pass it on to global_iu_recv_cb(). Simplest would be as a member of the same ue_conn_ctx, then MSC/SGSN can copy it to their own lists of RNCs.

So I would:

- completely drop struct ranap_iu_rnc and rnc_list from iu_client.c.
- in ue_conn_ctx, replace the *rnc with just the rnc_id (ABI change!?).
- drop wrappers ranap_iu_page_{cs,ps}(), instead use iu_page() directly with arguments iu_page2(imsi, tmsi_or_ptmsi, sccp_addr).
- In the SGSN and MSC, absorb the functionality of rnc_list: record LAC-RAC <-> RNCId-SCCPaddr mappings, lookup LAC-RAC for paging.

Then the MSC can freely store the RNC information in any way it chooses, and the iu_client.c is limited to useful bitshifting to receive and compose RANAP messages, which it should probably have been from the start.

What to do about API/ABI stability? When the list is ripped out of iu_client.c, the logical consequence would be complete non-backwards-compatibility of iu_paging_*, since the old API expects a list of RNCs that is no longer there. If we want to still allow deprecated use of iu_client.h, we could jump through hoops and allow a client to choose between an own list management or the RNC list within iu_client (SGSN could then remain unchanged). ranap_iu_init2() could be added to indicate use of an external list ... but the code could be clearer if we could just completely drop the list part and break compat. It seems very tempting to sacrifice "just" the two paging functions to get rid of the list code completely, but it would be a breaking point, preventing building new/old MSC,SGSN against new/old libosmo-ranap. The change of struct ue_conn_ctx is another aspect; in practice, MSC and SGSN don't access the ue_ctx->rnc member anywhere, so we at Osmocom wouldn't care much about that; would anyone else?

laforge, does all this make sense to you? What do you think about compat?

#13 - 03/28/2019 03:04 PM - neels

- Assignee changed from dexter to neels

- % Done changed from 20 to 90

In the ongoing refactoring of osmo-msc for inter-BSC Handover, the handling of RAN connections over SCCP has been basically rewritten.
It no longer uses the iu_client from libosmo-ranap at all. Both BSSAP and RANAP are handled by the same RAN peer FSM, plugged with distinct RAN implementations.
So all questions about ranap_iu_rnc, ue_conn_ctx and paging are moot; also the SGSN remains unchanged, so these questions are also moot.

In upcoming osmo-msc patches, each RAN peer automatically records the cells seen on it from Complete Layer 3 messages.
Also we feature an explicit "neighbor" configuration naming known BSC<->cell id relations.

A separate issue is that on Iu, there are actually more cell identity elements that we currently ignore, not really relevant here.

This issue will be resolved as soon as the osmo-msc inter-BSC HO patches are merged.

#14 - 05/09/2019 12:13 AM - neels

- Status changed from Feedback to Resolved

- % Done changed from 90 to 100

LAC recording is now merged to osmo-msc master