osmo-sip-connector - Feature #1679

Bug # 1604 (New): Easier / more direct SIP integration

osmo-sip-connector: Add stats for call handling

03/31/2016 06:34 PM - zecke

Status:	Stalled	**Start date:**	03/31/2016
Priority:	High	**Due date:**	
Assignee:	osmith	**% Done:**	10%
Category:			
Target version:			
Resolution:			

Description

Use the rate ctr and other stats for call handling. This should measure:

- Initiated calls
- Failed calls
- Connected calls
- Released calls

Related issues:

Related to OsmoMSC - Feature #3636: detailed call statistics (with rate count...	**New**	**10/08/2018**

History

#1 - 05/02/2016 10:59 PM - laforge

- Category set to osmo-sip-connector

#2 - 03/17/2017 09:59 PM - laforge

- Project changed from OsmoNITB to osmo-sip-connector

- Category deleted (osmo-sip-connector)

#3 - 11/07/2017 10:01 PM - laforge

#4 - 03/27/2018 01:43 PM - laforge

- Assignee changed from zecke to sysmocom

#5 - 09/01/2018 08:40 PM - laforge

- Assignee changed from sysmocom to osmith

#6 - 09/20/2018 01:28 PM - laforge

- Priority changed from Normal to High

#7 - 09/20/2018 04:14 PM - neels

There are various counters and stats frameworks in Osmocom for historical reasons.
(libosmocore/src/vty/stats_vty.c gives a kind of overview what we have in terms of stats/counters.)

The latest greatest is apparently libosmocore/include/osmocom/core/rate_ctr.h.
This issue could be more specific on which stats kind is desired, I'm assuming rate_ctr.

For example, see

- osmo-bsc/include/osmocom/bsc/gsm_data.h

```
enum {
        BSC_CTR_ASSIGNMENT_ATTEMPTED,
```

```
        BSC_CTR_ASSIGNMENT_COMPLETED,
        BSC_CTR_ASSIGNMENT_STOPPED,
[...]
};

static const struct rate_ctr_desc bsc_ctr_description[] = {
        [BSC_CTR_ASSIGNMENT_ATTEMPTED] =        {"assignment:attempted", "Assignment attempts."},
        [BSC_CTR_ASSIGNMENT_COMPLETED] =        {"assignment:completed", "Assignment completed."},
        [BSC_CTR_ASSIGNMENT_STOPPED] =          {"assignment:stopped", "Connection ended during Assignment."},
[...]

};

static const struct rate_ctr_group_desc bsc_ctrg_desc = {
        "bsc",
        "base station controller",
        OSMO_STATS_CLASS_GLOBAL,
        ARRAY_SIZE(bsc_ctr_description),
        bsc_ctr_description,
};
```

- osmo-bsc/src/osmo-bsc/bsc_init.c

```
        /* init statistics */
        net->bsc_ctrs = rate_ctr_group_alloc(net, &bsc_ctrg_desc, 0);
        if (!net->bsc_ctrs) {
                talloc_free(net);
                return NULL;
        }
```

and in bsc_vty.c bsc_vty_init()

```
osmo_stats_vty_add_cmds();
```

The rate_ctr are also exposed on the CTRL interface besides the VTY, by libosmocore/src/ctrl/control_if.c under CTRL command 'rate_ctr *' (see get_rate_ctr()).

There should also be a rate_ctr_init(), just now I notice osmo-bsc doesn't call that!? Bug!!

You can also look for examples in MSC, SGSN,...

#8 - 10/01/2018 12:12 PM - osmith

- Status changed from New to In Progress

Thanks for the detailed intro neels!

To make sure this is not already implemented, I've looked at how it is behaving in the current state first. When connecting to the VTY, there are show stats and show rate-counters commands. They don't output anything useful and it does not change when making calls.

```
> show stats
Ungrouped counters:
> show rate-counters
>
```

#9 - 10/01/2018 03:02 PM - osmith

- % Done changed from 0 to 10

WIP in branch osmith/statistics.

#10 - 10/02/2018 03:39 PM - laforge

#13 - 10/04/2018 09:48 AM - osmith

useful feedback from whytek in IRC:

> From my point of view of actual use, Probably it would be more interesting to see stats on failure causes than just "failed"
> also, I'm not sure what "released calls" would really describe,
> all calls (that are connected) will eventually be released.
> so i guess, that any difference between connected calls and released calls would mean "calls still in connected state", or "something went wrong"
> I think you might be able to stay out of sip.c, and rather work within the call control functions in call.c

#14 - 10/04/2018 11:14 AM - osmith

> also, I'm not sure what "released calls" would really describe,
> all calls (that are connected) will eventually be released.
> so i guess, that any difference between connected calls and released calls would mean "calls still in connected state", or "something went wrong"

Sounds reasonable, I'll leave that out unless zecke or laforge have a reason why that is needed.

From my point of view of actual use, Probably it would be more interesting to see stats on failure causes than just "failed"

That sounds more useful indeed. But which failures are actually relevant? If we would only catch the ones coming from call.c as you have suggested, then we have the following:

```
call.c:         LOGP(DAPP, LOGL_ERROR, "call(%u) with unknown leg(%p/%d)\n",
call.c:         LOGP(DCALL, LOGL_ERROR, "Failed to allocate memory for call\n");
call.c:         LOGP(DCALL, LOGL_ERROR, "Failed to allocate MNCC leg\n");
call.c:         LOGP(DCALL, LOGL_ERROR, "Failed to allocate memory for call\n");
call.c:         LOGP(DCALL, LOGL_ERROR, "Failed to allocate SIP leg\n");
```

we can not count the errors raised from sip.c (which may also be interesting):

```
sip.c:          LOGP(DSIP, LOGL_ERROR, "leg(%p) connected but leg gone\n", leg);
sip.c:          LOGP(DSIP, LOGL_ERROR, "leg(%p) incompatible audio, releasing\n", leg);
sip.c:          LOGP(DSIP, LOGL_ERROR, "No supported codec.\n");
sip.c:          LOGP(DSIP, LOGL_ERROR, "No supported codec.\n");
sip.c:          LOGP(DSIP, LOGL_ERROR, "Unknown from/to for invite.\n");
sip.c:          LOGP(DSIP, LOGL_ERROR, "leg(%p) no audio, releasing\n", leg);
sip.c:                  LOGP(DSIP, LOGL_ERROR, "leg(%p) unknown SIP status(%d), releasing.\n", leg, status);
sip.c:          LOGP(DSIP, LOGL_ERROR, "leg(%p) got bye, releasing.\n", leg);
sip.c:          LOGP(DSIP, LOGL_ERROR, "Canceled on leg(%p)\n", hmagic);
sip.c:  LOGP(DSIP, LOGL_ERROR, "%s(): Cause(%s) not found in map.\n", __func__, gsm48_cc_cause_name(cause));
sip.c:          LOGP(DSIP, LOGL_ERROR, "Failed to allocate leg for call(%u)\n",
sip.c:          LOGP(DSIP, LOGL_ERROR, "Failed to allocate nua for call(%u)\n",
```

Besides, there are also errors from other files.

keith: So does it make sense to only look at call.c then, or should we go through all error messages and decide which ones are useful to count and which ones are not?

#15 - 10/04/2018 11:20 AM - laforge

from my point of view, the following errors from your list sound worth counting
(in distinct/separate counters):

```
* call.c:          LOGP(DAPP, LOGL_ERROR, "call(%u) with unknown leg(%p/%d)\n",

* call.c:          LOGP(DCALL, LOGL_ERROR, "Failed to allocate memory for call\n");
call.c:          LOGP(DCALL, LOGL_ERROR, "Failed to allocate memory for call\n");
[both in one shared counter]

* call.c:          LOGP(DCALL, LOGL_ERROR, "Failed to allocate MNCC leg\n");
call.c:          LOGP(DCALL, LOGL_ERROR, "Failed to allocate SIP leg\n");
sip.c:           LOGP(DSIP, LOGL_ERROR, "Failed to allocate leg for call(%u)\n",
sip.c:           LOGP(DSIP, LOGL_ERROR, "Failed to allocate nua for call(%u)\n",
[all in one shared counter, making sure we don't double-count]

* sip.c:           LOGP(DSIP, LOGL_ERROR, "leg(%p) incompatible audio, releasing\n", leg);

* sip.c:           LOGP(DSIP, LOGL_ERROR, "No supported codec.\n");
sip.c:           LOGP(DSIP, LOGL_ERROR, "No supported codec.\n");
[both in one shared counter]

* sip.c:  LOGP(DSIP, LOGL_ERROR, "%s(): Cause(%s) not found in map.\n", __func__, gsm48_cc_cause_name(cause));

* sip.c:                  LOGP(DSIP, LOGL_ERROR, "leg(%p) unknown SIP status(%d), releasing.\n", leg, status);
```

But in any case, keith should probably have the best practical feedback on this.

#16 - 10/05/2018 07:20 PM - keith

Hi.
I would not be so interested in the errors that are "caught" and logged in the code, although this might be useful to a programmer, it is of little interest to the end user. Still, if you're going to do counters we should so them for everyone, so I concur with laforge.
All the same, most if not every one of those LOGP lines that you quoted means something went kind of (badly) wrong that should not have. I would not expect to see them, ever, in normal usage.
I'm not sure if that is a reason to not count them, or a very good reason TO count them! :)

example:

```
LOGP(DCALL, LOGL_ERROR, "Failed to allocate memory for call\n");
```

I've never seen this, and I guess if it happened, you are in trouble anyway!

What I might be more interested in is:
How many calls were;

- busy
- rang out (unanswered)
- unreachable,
- etc, etc...

This is all available in the MNCC cause and SIP status, but there's no if's and LOGing for them in the code.
You might need to write a switch statement or some such and increase the counters based on the cause values.

#17 - 10/05/2018 07:50 PM - laforge

keith: One of the questions is if the osmo-sip-connector is the best place to collect such [not straight sip relatd] stats, or if they shouldn't rather be in OsmoMSC.

Having them on the MSC side will make the stats available to everyone who usees it, whether with built-in MNCC handler or with osmo-sip-connector, or with LCR (assuming compatibility issues are sorted out) or whatever other handler/translator we may have, e.g. an ISUP converter.

Also, the MSC has a bit more knowledge on what's happening in detail, i.e. whether there was some failure talking to the MGW, or why exactly the radio channel was closed, ...

#18 - 10/05/2018 07:55 PM - keith

osmith,

http://git.osmocom.org/osmo-sip-connector/tree/src/call.h#n47

This is going to contain the reason for call termination^^^^^

You could print out stats of call terminations, based on this:
https://freeswitch.org/confluence/display/FREESWITCH/Hangup+Cause+Code+Table

So I guess you'ld see something like:

OsmoSIPcon> show stats

Call Termination Reasons:
xxx UNALLOCATED_NUMBER
xxx USER_BUSY
xxx NO_ANSWER

something like that..
We could extract the "Reason" and get call termination reason on the BYE as well. (it's going to be nearly always 16) I don't think we do that right now. Then maybe divide into "successful" and "unsuccessful" call terminations.

Although I learned recently while looking at some early media issues, that in the SIP world, "success" is not what a human might imagine. a busy call for example, is "successful"., i.e without error. :))

#19 - 10/05/2018 07:57 PM - keith

laforge Yep.. You're absolutely correct.

All the same, if Oliver wants to go ahead and do it here too, I guess that's OK ?

#20 - 10/08/2018 09:31 AM - osmith

keith wrote:

> All the same, if Oliver wants to go ahead and do it here too, I guess that's OK ?

I only want to go ahead with this if it is actually useful :)

keith: laforge mentioned, that you have the best practical feedback on this. So you can probably tell me if that makes sense.

> What I might be more interested in is:
> How many calls were;
>
> - busy
> - rang out (unanswered)
> - unreachable,
> - etc, etc...
>
> One of the questions is if the osmo-sip-connector is the best place to collect such
> [not straight sip relatd] stats, or if they shouldn't rather be in OsmoMSC.
>
> Yep.. You're absolutely correct.

keith, laforge: how about I open a new issue, about adding these stats to OsmoMSC and start working on that?

#21 - 10/08/2018 09:50 AM - laforge

IMHO, osmo-sip-connector should only receive those statistics that are not possible to gather on the OsmoMSC side (if there ara any such statistics).

On Mon, Oct 08, 2018 at 09:31:54AM +0000, redmine@lists.osmocom.org wrote:

> keith, laforge: how about I open a new issue, about adding these stats to OsmoMSC and start working on that?

At least open the new issue. Whether or not to work on it right ahead depends on the priority of other issues in your list.

#22 - 10/08/2018 11:15 AM - osmith

- Related to Feature #3636: detailed call statistics (with rate counters) added

#23 - 10/12/2018 12:11 PM - osmith

New issue #3636 opened for the osmo-msc counters.

keith, laforge: should I reject this issue now?

#24 - 10/16/2018 01:57 PM - osmith

- Status changed from In Progress to Stalled