OsmoBSC - Feature #3697

Make "MSC Release" in SI3 configurable

11/19/2018 10:09 AM - laforge

Status:	Rejected	**Start date:**	11/19/2018
Priority:	High	**Due date:**	
Assignee:	osmith	**% Done:**	0%
Category:			
Target version:			
Spec Reference:			

Description

Right now we unconditionally advertise MSCR as 1 (equals R99). Let's make this setting (vty-) configurable so that people can play with advertising different 3GPP release support to phones.

I think the single bit only defines R98 / R99+. It's a follow-up question where we can indicate capabilities for higher releases.

Related issues:

Related to OsmoBSC - Feature #3698: Make "SGSN Release" in SI3 configurable	**Rejected**	**11/19/2018**

History

#1 - 11/19/2018 10:11 AM - laforge

- Related to Feature #3698: Make "SGSN Release" in SI3 configurable added

#2 - 11/19/2018 10:52 AM - ant

I looked at MSCR just now on my QualiPOC and it seemed to be declaring MSCR as R98 or older.

Might be worth validating that first...

//Ant

-----Original Message-----
From: laforge [REDMINE] <redmine@lists.osmocom.org>
Sent: 19 November 2018 10:10
Subject: [OsmoBSC - Feature #3697] (New) Make "MSC Release" in SI3 configurable

Issue #3697 has been reported by laforge.

#3 - 11/22/2018 12:51 PM - osmith

- Status changed from New to In Progress

#4 - 11/22/2018 01:21 PM - osmith

> I looked at MSCR just now on my QualiPOC and it seemed to be declaring MSCR as R98 or older.

This is weird, we have 1 hardcoded in osmo_bsc_main.c right now:

```
/* Indicate R99 MSC in SI3 */
  bts->si_common.chan_desc.mscr = 1;
```

I'll implement that it can be changed with the vty configuration.

#5 - 11/22/2018 01:30 PM - ant

I guess we could still be using an older osmorelease... a major pain vector of mine

Thank you for making MSCR changeable though. That will hopefully help, and if it doesn't I'll ask you to make a test change on the SI2quat which should liven up the UE actually using the information contained within it (which it currently ignores).

//Ant

#6 - 11/22/2018 05:21 PM - neels

Pondering about this:

Since it's the MSC's release indicator, I somehow expected that the MSC tells the BSC about it over BSSMAP, but that doesn't seem to be the case. So then the BSC has to "guess"/know what the MSC is capable of.

osmo-bsc has an 'msc' section in its config, I would intuitively place this config in there, see osmo_bsc_vty.c. I would name the config option:

```
msc
   r99
   no r99
```

or maybe

```
r99_or_later
   r98_or_earlier
```

(I think the first is nicer)

The question that remains for me is, does it make sense to set it per BTS instead? After all it is a system information setting, and assuming that all MSCs we encounter in the field will anyway be R99+ capable (?), the user might instead want to configure individual BTSes to advertise R98 for hacking reasons?

Assuming this is useful, my deeper question is: does it make sense at all to advertise an MSC as R98 when it is in fact R99, and without the MSC knowing about that? Would the MSC still encode things in R99 ways and the subscriber might be stunned by that?

- Authentication: as soon as there are UMTS AKA tokens in osmo-hlr, osmo-msc will send an AUTN challenge. So a BTS might advertise R98 or earlier in SI, but the MSC still sends AUTN -- weird. Notably, since the MSC has no way to tell what the BSC's config for R99-ness is, the MSC can't switch that off per BTS.
- I remember this bit of code: "#define gsm48_hdr_msg_type gsm48_hdr_msg_type_r99" in libosmocore/include/osmocom/gsm/protocol/gsm_04_08.h. Funnily enough both gsm48_hdr_msg_type_r99() and gsm48_hdr_msg_type_r98() appear to do the exact same thing, even though it looks slightly different on first sight. Whatever happened there...

But considering authentication, to me it makes sense to only set the bit according to what the MSC actually **is** capable of.
So that brings me to conclude: add this configuration in the 'msc' section of osmo-bsc.cfg.

More questions asked by osmith in the WIP gerrit patch:

'How do I set the default to "1" instead of "0"?'

You have full control over the default config setting and whether/what the VTY outputs during 'write'.

The pattern for any default value would be, in pseudocode:

```
#define FOO_DEFAULT_VAL tomato

void cfg_init(){
  cfg.foo = FOO_DEFAULT_VAL;
}

DEFUN("foo (tomato|potato)")
{
  cfg.foo = argv[0];
}

void foo_vty_write(){
  if (cfg.foo != FOO_DEFAULT_VAL)
    vty_out("foo %s%s", cfg.foo, VTY_NEWLINE);
}
```

So you see that you can choose to always output the vty config, or to choose "potato" as the default, or whatever.
There are also various ways to shoot yourself in the foot, e.g. typos in vty_write() or outputting options that aren't parseable.
That's why we have VTY tests that try to ensure we write config back in a parseable way. Point being that **anything** is possible.

#7 - 11/22/2018 05:46 PM - neels

Argh, it is even more complex.

Techincally in the BSC we still are able to maintain several different MSCs, and would select the MSC per subscriber.

However, the SI bits get sent to the BTS on OML bootstrapping; we could technically update the SI information also after the BTS has connected. But obviously we don't send different SI bits per subscriber, but per cell. That doesn't fit at all. So it doesn't even make sense to have separate MSCs with

separate R99 capability, because there is no way to communicate that in a useful way.

So, on the one hand, it doesn't make sense to say "R98" in the SI when the MSC is in fact R99 -- i.e. set R99 per cell doesn't make sense.
It also doesn't make sense to keep a per-MSC "R99" flag, because should we ever have differing MSCs on the same cell, that would be impossible to match.

The only thing I see making sense now is one single global R99 flag for the entire BSC.

So now I'm at

```
network
 r98
 r99
```

or maybe

```
network
 release (r98|r99)
```

In the meantime laforge said "I think (R98|R99+) would be more user friendly" but the VTY will trigger on the capital R to interpret as a free-form value instead of a fixed token. So it has to be lower case, and not sure whether the + is allowed. If it is, then feel free to add it, like "release (r98-|r99+)"

So after all I'm saying to add a cfg_net_release command in bsc_vty.c.

#8 - 11/22/2018 05:47 PM - neels

s/small caps/lower case

#9 - 11/22/2018 05:50 PM - neels

osmith, I hope I'm making sense here, maybe run my conclusion by laforge if he has the time. (In practice I don't really understand why we need to advertise R98- at all.)

#10 - 11/22/2018 06:47 PM - ant

From my perspective I am 100% happy to have NO option if we are definitely advertising as R99+. This whole line of enquiry started because of two things:

1) on my QualiPOC, the SI3 decoded message shows our osmoBTS as MSCR of R98. It is possible that the QualiPOC is wrong but on a commercial GSM cell it shows MSCR of R99 onwards..

2) we need SI2quat to function correctly and not one single UE we have ever tested honours the message - which means that:

- the message is formatted wrongly (although it displays correctly on the QualiPOC).
- one of the parameters is wrong so the message is never executed.
- the MSCR release is such that the UE doesn't process the message because the system release is too old to support such functionality.

Given that my reference test handset tells me we're R98, it seemed that the latter was a definite possibility. I am happy to re-validate later this evening and will send some versions etc. as I am keen not to waste anyone's time.

/Ant

#11 - 11/23/2018 10:03 AM - osmith

ant wrote:

> I guess we could still be using an older osmorelease... a major pain vector of mine

It seems, that the R99+ advertising was added as part of #1593 in March of 2017.

neels wrote:

> I hope I'm making sense here, maybe run my conclusion by laforge if he has the time. (In practice I don't really understand why we need to advertise R98- at all.)

Yes, it makes sense. Thank you very much for the detailed explanations.

laforge: do you still want this feature? what do you think about implementing the network-level setting that neels described above?

#12 - 11/23/2018 11:07 AM - ant

I should be able to find out what we're using build-wise in the next two hours. Will update then.

#13 - 11/23/2018 02:40 PM - osmith

ant wrote:

> I should be able to find out what we're using build-wise in the next two hours. Will update then.

any results? :)

#14 - 11/23/2018 04:57 PM - ant

For fear of sounding stupid (and I apologise, building osmo sources isn't part of my remit), is there an easy way of determining the build revision either from CLI or in the binary collection.

I was hoping to get this from the live network, but if not I'll go source hunting.

/Anthony Timson

#15 - 11/23/2018 09:50 PM - laforge

"show version"

#16 - 12/04/2018 10:37 AM - osmith

ant: did you have any success with querying the version?

#17 - 01/29/2019 01:51 PM - osmith

- Status changed from In Progress to Rejected

Closing, as the usefulness of having this configurable is highly questionable as Neels stated.

It doesn't solve Anthony's problem either, upgrading to a new osmo-bsc version that advertises R99+ support would be the solution.

#18 - 01/29/2019 02:00 PM - ant

I patched our version to R99+ when I got our SDK so am now able to make changes as needed.

Thanks - and agree with closing this out.

/Ant