We need to select codecs. There were multiple mails on the OpenBSC list. For GSM MO call we need to check the bearer calls and the availability of channels and then can offer different codecs. The FR vs. HR decision needs to be done early but then codec can be changed.

The MNCC protocol needs to change in many still to be defined ways. This includes:

- NITB needs to inform which TCH/F, TCH/H are available
- RTP_CREATE/RTP_CONNECT needs to include AMR mode-sets
- Need to test switching codec
- BTS needs to support Payload2 to allow asymmetric PayloadType for the same codec!

Related issues:
- Related to OsmoMSC - Bug #2854: OsmoMSC never updates bearer capability from MNCC
  - Resolved 01/22/2018
- Related to OsmoMSC - Feature #2832: MNCC doesn't include field for "supported AMR mode-sets"
  - New 01/16/2018
- Related to OsmoMGW - Bug #3114: libosmo-mgcp doesn't properly deal with SDP codec_set...
  - Resolved 03/26/2018
- Related to OsmoMGW - Bug #3115: libosmo-mgcp_client isn't able to properly encrypt AH
  - Resolved 03/26/2018
- Related to OsmoMSC - Bug #3650: MSC is not sending a payload_type to MNCC?
  - Closed 10/12/2018
- Related to osmo-sip-connector - Bug #5177: respect dynamic RTP payload types...
  - New 06/12/2021

Associated revisions

Revision d5b6a78b - 03/26/2018 09:03 PM - laforge
message sequence charts for GSM <-> SIP codec selection

Those diagrams represent the current design of the way how GSM <-> SIP codec selection should work in the future.

Related: OS#1683
Change-Id: If92379b41d859bd7be4b7689dc0bbca7cb6e682

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 10:00 PM - laforge
- Priority changed from Normal to High

#4 - 01/22/2018 01:03 AM - laforge
- Related to Bug #2854: OsmoMSC never updates bearer capability from MNCC added

#5 - 02/19/2018 09:19 AM - laforge
- Assignee changed from zecke to sysmocom
The high-level goals are as follows:

- respect the codec capabilities of all elements, i.e.
  - the external SIP world (as in the SDP)
  - the MS (bearer capabilities)
  - the BTS+BSC (codec lists in BSSMAP ASSIGNMENT related messages)
- prefer a non-transcoding situation to any that involves transcoding
- if transcoding is required, it should happen at the bsc-located OsmoMGW

**MT call**

1. INVITE with SDP is received
2. INVITE is translated to MNCC_SETUP_REQ
   - if the SDP included any GSM related codecs (FR/HR/EF R/AMR), then those codecs should be translated to the MNCC bearer capability field. Otherwise, MNCC bearer capability is omitted (equals any codec/bearer)
   - bearer capabilities should indicate a preference for those (in the same order as the SDP, as it also is ordered by preference). All other codecs shall be listed last, i.e. with lower preference
3. OsmoMSC triggers paging towards BSC/BTS/...
4. OsmoMSC sends the CC SETUP to the MT, including CC bearer capabilities translated from the MNCC bearer capabilities
5. the MT MS responds, possibly with negotiated bearer capabilities, based on its own capabilities
6. OsmoMSC at some point sends a BSSMAP ASSIGNMENT CMD listing the negotiated codecs
7. OsmoBSC responds with a BSSMAP ASSIGNMENT COMPLETE, indicating the exact channel type + codec that was chosen (based on available channels, the BTS model, local policy, ...)
8. the MT MS responds at some point with a CONNECT (when the user picks up)

All in all, I think the ladder diagram should look like this:
**#8 - 03/26/2018 03:48 PM - zecke**

A pre-selection could already be made at the time of paging? GSM 08.08 holds "Channel Needed" to indicate if a Full rate or Dual rate channel should be required. This makes it more complicated but maybe it still fits in the design? Otherwise this looks great!

**#9 - 03/26/2018 07:53 PM - laforge**

- Related to Bug #3114: libosmo-mgcp doesn't properly deal with SDP containing multiple codecs added

**#10 - 03/26/2018 07:55 PM - laforge**

- Related to Bug #3115: libosmo-mgcp_client isn’t able to properly encode SDP with multiple codec options added

**#11 - 03/26/2018 08:00 PM - laforge**

After working on this for another two days, I have identified the following problems:

**Policy**

The question is what is the operator policy / optimization target here? It could be

1. perform as little transcoding as possible. This would possibly mean using more FR (TCH/F) than AMR/HR (TCH/H) assuming that the SIP peer supports FR natively but not AMR
2. use radio resources as efficiently as possible. In this case we’d prefer AMR/HR over anything else and would then end up transcoding potentially every call if the SIP peer doesn’t support AMR.
I guess both cases have their usefulness, so the policy would have to be configurable:

- case a (avoid transcoding) is useful if the machine running OsmoMGW is cpu constrained (like a deeply embedded system)
- case b (optimize radio resources) makes sense in all cases where sufficient transcoding capacity is available

**MO call: SDP from SIP side will arrive late ("200 OK" to INVITE)**

- in classic SIP, the offer of codecs is made in INVITE, and the response only in "200 OK"
  - this means for MO we couldn’t select the least common denominator between GSM and SIP side (to avoid transcoding) until the SIP UA has already picked up the phone.
  - RFC3262 offers the option to send the response SDP in "Reliable 18x" responses. The question is: Who supports this? If support is wide-spread this would be a solution. We would then request reliable 18x responses and wait with the BSSMAP Assignment until we have the "180 Ringing" from the SIP side, at which point we can try to assign a channel with the codec supported by the SIP side to avoid transcoding.

**osmo-mgw / libosmo-mgcp[-client] shortcomings**

- libosmo-mgcp_client doesn’t seem capable of properly encoding SDP with multiple codec options (rtpmap) :(
- libosmo-mgcp seems capable of properly parsing SDP with many codec options (rtpmap) but then only passes up to two from the parser upwards :-( see mgcp_rtp_end.codec and mgcp_rtp_end.alt_codec #3114

**complete MNCC breakage in OsmoMSC**

osmo-msc contains funny bits of code like hard-coding the RTP payload type in the MNCC_RTP_CREATE to "0", rather than using any of the actual codecs, see e.g.

```c
/*FIXME: This has to be set to some meaningful value,
 * before the MSC-Split, this value was pulled from
 * lchan->abis_ip.rtp_payload */
uint32_t payload_type = 0;
```

This is absolutely not acceptable and basically breaks functionality of MNCC beyond expectations. In my testing, osmo-sip-connector is never able to find a common codec between the SIP and GSM side, as the SIP side of course is right to accept any re-definition of a non-dynamic RTP payload type "0".

#12 - 03/26/2018 08:37 PM - laforge

- File msc_mncc_mo_sip_rel18x_mgcp.png added
- File msc_mncc_mo_sip_mgcp.png added

**proposed MO call with reliable 18x option**
proposed MO call without reliable 18x option
On Mon, Mar 26, 2018 at 03:48:21PM +0000, zecke [REDMINE] wrote:

A pre-selection could already be made at the time of paging? GSM 08.08 holds "Channel Needed" to indicate if a Full rate or Dual rate channel should be required.

OsmoBSC will to my knowledge ignore that, as we always use late assignment these days. So it will use the "smallest available" channel to handle the signaling until the call goes into alerting state, when the BSSMAP assignment is used.

#14 - 10/02/2018 03:38 PM - laforge

#17 - 11/19/2018 12:53 PM - keith

Related:

GSM 02.40:
Supervisory Tones, indicating primarily ringing, engaged and unobtainable numbers, may be generated by both the PLMN and PSTN.

Except for ring tone, all tones indicating call progress to a Mobile Station user shall be generated in the MS, on the basis of signals from the network where available, and are according to the standard defined in this specification.

Tones sent to a caller to a mobile station will be generated in the network, generally local to the caller, and will be to the standard of his local exchange, except for mobile to mobile calls, where the tones will be generated in the calling MS. For mobile terminated calls, the ring tone will be generated in the called MSC.

This means that if we are using a SIP UA connected to the GSM network, we should have it behave in a compatible way. That is to say, we should not send 180 Ringing when we are the "CALLED MSC", but rather we should send 183 with SDP and then start sending ring-back media.

For a SIP call terminating in our MSC we should be generating ring-back in the MSC and our SIP UA should be configured to expect this.

---

This means that if we are using a SIP UA connected to the GSM network, we should have it behave in a compatible way. That is to say, we should not send 180 Ringing when we are the "CALLED MSC", but rather we should send 183 with SDP and then start sending ring-back media.

For a SIP call terminating in our MSC we should be generating ring-back in the MSC and our SIP UA should be configured to expect this.

#18 - 07/18/2019 05:11 AM - laforge
- Priority changed from High to Urgent

#19 - 07/25/2019 07:32 AM - laforge
- Status changed from New to Feedback
- Assignee changed from sysmocom to neels

Hi neels, I thought this might be a suitable challenge for you to resolve at some point in the near future, maybe after the last part of your summer holidays?

The downside is that there's currently no customer or support project to which we can bill this, so any related effort would be entirely sysmocom-funded (and our own resources are limited). So the question is also how much time do you expect to need to get this resolved?

IMHO, this one of the most critical and long-standing issues to "complete" the voice support.

#20 - 07/25/2019 09:33 AM - keith

aaah

:-/

I started on some of this. that is, I setup lubuhtrace, did some analysis, had a look at the comms between LCR and osmo-nitb... then got distracted with rhizomatica instalations.

My 1st intended TODO was to at least have the osmo-sip-connector fill in the bearer caps fields based on the codec list in the incoming SIP.

I think both neels and I would like progress on this before CCC camp.

#21 - 07/25/2019 09:58 AM - keith

Looking through comments here:

re: #1683-7 I will try to to the above early next week (the bearer caps in sip connector part)

re: #1683-11 and "Who supports 183 aka Early Media?" - Everyone. All of our PSTN providers send 183 with early media, I mean.. it's the standard UX these days.. "Ihr Gesprächspartner ist zur Zeit leider nicht erreichbar", right? :)

AFAIK, you cannot request a 183 as such, however I'm not sure it makes any difference. Whether you get a 183 or a 200, the point is you then should expect and also handle the media stream at that time, no?

re: #1683-17 Am I wrong here? Reading this again, damn it is ambiguous. The last sentence seems to contradict the first.

#22 - 07/29/2019 12:33 PM - neels

laforge re "this might be a suitable challenge for you": an interesting task indeed and it makes a lot of sense to solve this. Though I shall refrain from completely refactoring the sipcon...

I'll coordinate with keith and try to gauge the effort required.

#23 - 08/05/2019 09:26 PM - neels
IIUC, the (verbally communicated) plan is to incorporate SDP in the MNCC protocol. I haven't yet figured out whether that makes the bearer capabilities obsolete, but currently I expect it does.

I've once written up a patch about that "funny" bit of code mentioned in #1683-11
Now pushed it as https://gerrit.osmocom.org/c/osmo-msc/+/15060 -- haven't properly tested yet, but thought it best to share.

Open questions I need to figure out...

1. A lot of detailed codec choice sits in osmo-bsc.
   If the MT side selects one specific codec (after Assignment), it seems likely that the MO BSS doesn't like that specific codec, while there would have been a match if all info had been available.
   Solution: the Assignment Complete contains both a Speech Codec (Chosen) as well as a Codec List (BSS Supported).
   So I assume that even though a specific codec got chosen on Assignment, the SDP should still contain all the other BSS Supported codecs.
   Then, if the other side can't serve the specific Speech Codec (Chosen), it can select from the others and send SDP back to indicate a change.
   Then that side can in turn decide to do another Assignment Command to select a different codec (or to transcode).

2. Only osmo-bsc.cfg has AMR modes config, setting which up-to-4 rates should be allowed on AMR.
   The selected AMR rates must ultimately also end up in the SDP towards the other call leg, right?
   I assume we should add an 'a=fmtp:112 mode-set=0,4,6,7' to indicate the permitted AMR modes towards PBX.
   The AMR modes are apparently also contained in the Assignment Complete, in the s0..15 bits for both Speech Codec (Chosen) and Codec List (BSS Supported).
   Again, if the two Chosen codecs on MO and MT do not match, the MSC might choose to do another Assignment Command to the BSS, or to transcode.

So my first idea for propagating a codec match between the two BSS looks something like...

Alternative idea: wait for the MO Assignment Complete before sending INVITE to the MT side, that gives us the MO BSS' Codec List (BSS Supported) already in the first INVITE's SDP towards MT BSS.
Then the Ringing back from MT can already contain the specific MT's Speech Codec (Chosen), and the MO side could decide whether to re-Assign or transcode.
To clarify the above a bit more, I am currently thinking of a scenario where a Kamailio simply forwards the SDP between the MO and MT side. The PBX in the middle could of course also decide to transcoded and/or offer only a specific codec in SDP, but my first aim is to make the MO and MT sides of osmo-msc negotiate SDP with each other successfully.

Open questions I need to figure out...

(1) A lot of detailed codec choice sits in osmo-bsc. If the MT side selects one specific codec (after Assignment), it seems likely that the MO BSS doesn’t like that specific codec, while there would have been a match if all info had been available. Solution: the Assignment Complete contains both a Speech Codec (Chosen) as well as a Codec List (BSS Supported).

(2) Only osmo-bsc.cfg has AMR modes config, setting which up-to-4 rates should be allowed on AMR.

please note that the current ‘active AMR modeset’ may very well be changed during a call. I’m not sure if we fully implement that today, but at least all the 3GPP interfaces allow for that, AFAIR. So we should preferably make assumptions on “how GSM works as per spec” and not make any specific assumptions about today’s implementation constraints.

Sure, for some very specific use cases / users we allow to constrain/restrict the AMR codec modes to a sub-set, but from a spec and real-world equipment side, every AMR implementation supports all modes, and should be able to change them without doing any assignment.
This looks like you're only using the bearer capabilities of the UE to determine the initial list of codecs. However, at this point it should already be clear

- what kind of codecs the serving BTS supports at all (frequently some devices don't support a given codec like nanoBTS not supporting HRv1)
- what kind of policy the BSC has configured in terms of which codecs and/or rates to use

The problem is probably that all this knowledge is in the BSS, and it's not propagate to the MSC until there's an actual assignment happening. One could of course think of vendor-specific TLVs, at least as a hypothetical option? I don't like deviating from the spec too much, and if we do, we always must make it an option that can be disabled for interop with other implementations (like osmux), but if we need that information sooner, why not add the Codec List (BSS Supported) already to an earlier BSSMAP message, maybe even to the Complete L3? The code for encoding and parsing that IE exists already, so it doesn't look like a lot of extra code. Your call (no pun intended)!

Alternative idea: wait for the MO Assignment Complete before sending INVITE to the MT side, that gives us the MO BSS Codec List (BSS Supported) already in the first INVITE's SDP towards MT BSS. Then the Ringing back from MT can already contain the specific MT's Speech Codec (Chosen), and the MO side could decide whether to re-assign or transcode:
This would basically mean we fall back to the "very early assignment" paradigm implemented in OsmoNITB. Real-world networks don't use this scheme in practise as only a fraction of all call attempts actually ever is established (B subscriber not picking up, ...) and using very early assignment would increase the TCH load with no gain in terms of calls that could actually be billed.

#28 - 08/14/2019 03:06 PM - neels

laforge wrote:

please note that the current 'active AMR modeset' may very well be changed during a call

Ok, but I don't fully understand HR3: AMR on TCH/H definitely cannot support bitrates above 6k7, so I assumed we should tell the other call leg about that limitation?

This looks like you're only using the bearer capabilities of the UE to determine the initial list of codecs.

The problem is probably that all this knowledge is in the BSS, and it's not propagate to the MSC until there's an actual assignment happening.

Exactly, that is precisely the problem I am facing.

One could of course think of vendor-specific TLVs,

I assumed there would be a sensible way to do this that the writers of the GSM specs intended? Is the official paradigm just to get each MS its favorite codec and always transcode on mismatch? No ambition in the specs to find a matching codec and avoid transcoding?

I don't like deviating from the spec too much, and if we do, we always must make it an option that can be disabled for interop with other implementations.

So there should be a code path that doesn't use osmo-specific deviations. So I would first focus on not using osmo-specific messages, even if that is suboptimal, and we can add deviations to achieve optimisation later, if we desire.

Decisions made during network planning and placing codec configurations to favor a match could in some setups render optimisations unnecessary, we'll see how it plays out.
This would basically mean we fall back to the "very early assignment" paradigm implemented in OsmoNITB.

I understood "early assignment" to mean that the first Chan Request from the MS already gets a TCH/F or TCH/H assigned; and that "late assignment" means assigning a SDCCH first, then moving to a TCH/* after CC Setup.

In this idea we would still wait for a CC Setup, but then get the BSS' Codec List from Assignment on MO first.

(B subscriber not picking up, ...)

Not picking up would mean that a ringtone was playing on an active TCH/*...
Did you mean that the B subscriber does not exist or is not attached to any network?

... 

Actually sufficient would be a message to ask the BSS for the supported codec list at a given point in time.
The BSSMAP Resource Request message looked promising but has no Codec List.

Another idea would be to remember a BSS' Codec List (supported) from earlier voice calls.

osmo-msc keeps state for each connected BSC and could easily store the last received codec list for each, maybe even for each individual Cell Identifier, to remember codecs per BTS even.

But available lchans on a BTS can of course change, so maybe this could be a sort of collection of everything we've ever seen from a cell, just to identify those that never show up as supported...

But it's obviously a hacky idea.

#29 - 08/15/2019 04:01 AM - neels
- File codecs3.png added
- File codecs4.png added

Interesting twist: looking closely at the msc-sipcon-pbx communication, I notice that indeed the current implementation follows this order:

( the places marked "SDP" in this chart are still mostly hypothetical, but the actual order is accurate)

So in fact the sipcon first instructs to RTP_CREATE, the MO Assignment happens and the RTP_CREATE is "acked" towards sipcon, and only then does sipcon send the INVITE to the MT side, i.e. when the RTP IP address and port from the MO side are known.

I came to notice this because of course the SDP contains the RTP IP address and port, which isn't known until the lchan is setup. So I was wondering what the sipcon sent as IP:port before knowing them, and noticed that it instead waits until it does know them.

Now, if we keep this order, then the drawback is: the MO side first already decides for a codec all on its own, before even knowing what the MT side
might support.
It is in fact possible to change this: we can instruct the MGW to create a port towards the PBX before starting the Assignment, hence knowing the RTP IP:port before MO Assignment.
We answer the sipcon's RTP_CREATE with full IP:port and codecs before having started Assignment on the MO side.
Then the MT side knows its own Bearer Cap and the MO side’s codecs, keeps only the intersection of both, and then MT is the first to do Assignment.
Thus we should always end up with a codec that both sides support (if such exists).

And then we would also only request a TCH/* on the MO side if the MT side is a valid subscriber that is attached and reachable.

#30 - 08/15/2019 04:09 AM - neels

To clarify, the above is able to get both MO and MT sides' MS bearer caps into consideration. But it is still not solved how to get the BSS’ knowledge of available channels and codec config into the equation.
In this last suggestion, the MT BSS gets to choose a codec knowing the MO MS bearer caps, but still doesn't know the MO BSS' preferences.
The ideas here:
- Osmocom-specific protocol extension
- Let MO again choose the codec, and if it mismatches ricochet that codec back to the MT side and re-Assign there to match.
- Yet another idea would be to make osmo-bsc re-encode the bearer caps and remove those speech_ver that aren't doable.
  That's a layer violation in that osmo-bsc tampers with the DTAP, but we would remove unavailable codecs right from the start.

#31 - 08/15/2019 08:39 AM - laforge
On Wed, Aug 14, 2019 at 03:06:56PM +0000, neels [REDMINE] wrote:

    please note that the current ‘active AMR modeset’ may very well be changed during a call

Ok, but I don't fully understand HR3: AMR on TCH/H definitely cannot support bitrates above 6k7, so I assumed we should tell the other call leg about that limitation?

makes sense.

The problem is probably that all this knowledge is in the BSS, and it's not propagate to the MSC
until there's an actual assignment happening.

Exactly, that is precisely the problem I am facing.

One could of course think of vendor-specific TLVs,

I assumed there would be a sensible way to do this that the writers of the GSM specs intended?
Is the official paradigm just to get each MS its favorite codec and always transcode on mismatch?
No ambition in the specs to find a matching codec and avoid transcoding?

The default is to do transcoding. Remember, this was designed when
a) the majority of calls were between mobile phones and land-line/PSTN
b) all the connections between the MSCs were ISDN circuits with uncompressed G.711 audio
anyway, so you must transcode anyway. In most cases actually you'd already transcode at
the BSC level (BSC-colocated TRAU) as the circuits between BSC and MSC are ISDN circuits.

The 3GPP solution for later optimization is called "TFO" or "TrFO" where two transcoders can detect
each others presence via in-band-signalling hidden in the voice stream and then synchronously
switch themselves off while the call is ongoing.

So there should be a code path that doesn't use osmo-specific deviations.
So I would first focus on not using osmo-specific messages, even if that is suboptimal,
and we can add deviations to achieve optimisation later, if we desire.

If you want to use that paradigm to full extent, you would implement transcoding in the MGW. Everything else
is an optimization which inherently is Osmocom specific.

This would basically mean we fall back to the "very early assignment" paradigm implemented in OsmoNITB.

I understood "early assignment" to mean that the first Chan Request from the MS already gets a TCH/F or TCH/H assigned;

that's "Very Early Assignment".

and that "late assignment" means assigning a SDCCH first, then moving to a TCH/" after CC Setup.
In this idea we would still wait for a CC Setup, but then get the BSS' Codec List from Assignment on MO first.

I think there's "Early" assignment (SETUP stage) and "Late Assignment" (Answer Stage).

Actually sufficient would be a message to ask the BSS for the supported codec list at a given point in time.
The BSSMAP Resource Request message looked promising but has no Codec List.

Another idea would be to remember a BSS' Codec List (supported) from earlier voice calls.
osmo-msc keeps state for each connected BSC and could easily store the last received codec list for each,
maybe even for each individual Cell Identifier, to remember codecs per BTS even.

But then you won't catch reconfigurations at the BSC, or changes/upgrades of BTS equipment. I think the
easiest approach is to add those capabilities to some random early BSSMAP message.

#32 - 08/28/2019 03:48 PM - neels
- Subject changed from osmo-sip-connector: Implement codec selection to osmo-sip-connector: Implement codec selection / move codec selection to osmo-msc
- Status changed from Feedback to In Progress
- % Done changed from 10 to 50

By now I have incorporated SDP in the MNCC protocol, and in my patches the SDP now merely gets forwarded 1:1 between MSC and PBX.
So this issue description should rather be "move codec selection to osmo-msc".

Source code wise it is interesting to note that osmo-sip-connector so far has a lot of code to parse and convert the SDP to/from payload type number,
IP addr+port etc, and is still missing most of the the CC Bearer Capabilities handling. By forwarding the SDP directly through to osmo-msc, most of
that code becomes unnecessary: osmo-msc already has a bunch of functions converting SDP to/from Bearer Capabilities Speech Versoin, Permitted
Speech, IP:port. We can end up ignoring most of the MNCC message contents.

There is one limitation, which is the recently added call-hold support in sipcon by whytek: I still need to check whether this can work with only
forwarding SDP, or whether osmo-sip-connector still needs to parse and overwrite the a=sendrecv|sendonly|recvonly|inactive attribute to make call holding work. In most places, it actually parses this from SDP and puts it back in SDP, so the chances are pretty good. Probably, osmo-msc would need to parse these SDP sendrecv attributes and translate to CC messages, possibly replacing the MNCC_HOLD_* message types? Could be simpler than editing SDP in sipcon.

This is only for later.

In the end, it seems the only thing osmo-sip-connector ends up doing is to compose/decode SIP messages and pasting SDP into/from it. It seems to me like it would make sense to just add SIP messaging to osmo-msc entirely, do away with the separate MNCC socket and separate program. We can see about that later.

Adopting above assignments naming:

- osmo-nitb used to do Very Early Assignment in the Channel Request stage;
- osmo-msc currently does Early Assignment in the SETUP stage;
- I think I want to move to Late Assignment in the Answer stage, to best benefit from both sides' codec selection.

From above this seems desirable in general: to not assign TCH before it is clear that the remote side is available. Should be easy enough with the new FSMs, just tweak a few events and transitions.

Same image as above again, this is my goal:

...and to complete above picture, I guess adding a "custom" IE to an early BSSMAP message indicating the BSS capabilities, to be enabled by osmo-bsc.cfg switch, would indeed be desirable.

For now still testing and exploring what SDP-in-MNCC can do without rearranging the BSS / MSC.
The final goal for avoiding transcoding and Late Assignment would look like this:

On Wed, Aug 28, 2019 at 03:48:52PM +0000, neels [REDMINE] wrote:

In the end, it seems the only thing osmo-sip-connector ends up doing is to compose/decode SIP messages and pasting SDP into/from it. It seems to me like it would make sense to just add SIP messaging to osmo-msc entirely, do away with the separate MNCC socket and separate program. We can see about that later.

MNCC is implemented in LCR, OsmocomBB and possibly other programs. We should ensure all speak the same dialect. Translating "externally" from MCNC to SIP ensures that e.g. we could build MNCC to BICC or MNCC to ISUP translators, should the need arise.

#36 - 09/02/2019 03:25 AM - neels

- % Done changed from 50 to 70
I am getting the first voice calls established by forwarding SDP between MSC and SIP. SDP replaces these MNCC message parts: ip, port, bearer_cap, payload_type, payload_msg_type. I've changed osmo-msc to do an MGW CRCX to CN first thing, so that the SDP already reflects a valid IP from the start.

With this patch, se should already be able to configure a Yate or FreeSwitch transcoder to accept both GSM FR and AMR, and dynamically tell it which one the RAN assigned after evaluating the MS Bearer Cap.

We are not yet able to

- negotiate around limitations imposed by cells (like GSM FR vs. GSM HR depending on available lchans) or
- negotiate a matching codec if the MT leg doesn't support the MO leg's favorite codec.

Currently settling the patch a bit: removing wild debug logging, grooming code.

After that I could explore, in no particular order:

- make osmo-bts advertise the cell's Codec List (BSS supported) early, during or before CC Setup.
- move Assignment on the MO side to after MNCC_ALERT_REQ

both of which should increase the chances of getting a matching codec negotiated.

We can actually also settle for the current result and keep the more versatile codec matching for a future exercise. At least on our Congress installation, we would simply ensure that the PBX supports all codecs a RAN might select, and need only tell the PBX about it. We can then dynamically support old non-AMR phones on 2G.

Since the POC guys anyway always transcode to PCMU (or was it PCMA?), none of the mentioned additional efforts would affect Congress.

#37 - 09/03/2019 01:53 AM - neels

How did I not see that: the BSSMAP Complete Layer 3 Information message actually contains a Codec List (BSS Supported) IE. Plus osmo-bts already goes at great lengths to compose this IE according to osmo-bsc.cfg and BTS lchan configurations. I've also already made provision for filtering on cell supported codecs from the start; looks like all I need to do now is plug the speech codec list from L3 Complete, nice.

#38 - 09/04/2019 09:32 AM - laforge

- Parent task deleted (#1604)

#39 - 09/12/2019 12:32 AM - neels

I'm facing one unsolved problem in mixed TCH/F vs TCH/H situations:

- there is an ambiguous mapping between the codec representations in CC and SDP:
  HR3 and FR3 are both represented as "AMR" in SDP. All other codecs imply TCH/F or TCH/H, but "AMR" may match a TCH/F with FR3 with a TCH/H using HR3.

With my limited knowledge of AMR I would have assumed that somehow these two types of AMR are able to agree on a common rate and just work. However, using a sysmobTS, when one side has FR3 and the other HR3, I can observe an actual crash (presumably of the sysmobTS DSP) similar to when sending non-AMR audio to an AMR configured lchan.

The symptoms are: Assignment succeeds, but the lchan fails hard before the CC Connect Acknowledge is sent. Shortly after, the BTS disconnects and reconnects to the BSC.

(This sounds similar: https://osmocom.org/issues/3746 -- I remember having seen this before, but can't find an issue on it that I created)

Remains to be seen whether we can sort this discrepancy out by composing accurate SDP fmtp parameters to reflect the audio rates supported by each side, and translate those to the permitted speech values in the BSSMAP Assignment Request.

So far in my patches I haven't gotten as far as accurately representing AMR rates in SDP, so that only the subset of rates that both sides support end up being used.

Might want to move that to a separate issue, because this only appears in mixed TCH/F vs TCH/H situations which we don't support well before this anyway: it's not a regression but can be seen a further improvement for later.

I'm first concentrating on verifying that both the ttcn3 tests and internal MNCC still work after my patches.

#40 - 09/12/2019 01:30 AM - neels

I was too curious and did a little test series. Indeed it is fairly easy to get FR3 and HR3 to work nicely, all that is required is at least one AMR rate that both sides support.

In above test, I had only tch-f modes 7 (FR3 only 12k2) and tch-h modes 3 (HR3 only 6k7), i.e. no similar modes allowed on the two TCH kinds. In fact, this was the only setting I tested that managed to crash the sysmobTS.

When I configured amr tch-f modes 5 7 and amr tch-h modes 0 3 (also no overlap, but different numbers), the lchans remained open and there was simply no audio.

The conclusion is that it is fine to treat both FR3 and HR3 as AMR in SDP, as long as the BSCs are configured so that FR3 and HR3 have at least one common AMR rate.

In a network where all BSCs are controlled, it is not even necessary to communicate the individual rates via SDP: apparently AMR does indeed sort
itself out nicely as long as both sides' rate support have overlaps.

We might want to still communicate the allowed rates in SDP fmtp parameters for PBX transcoders or somesuch, but even that could turn out unnecessary.

So, the good news is that we can sensibly support voice calls on fully dynamic timeslots with AMR as long as tch-f and tch-h have at least one similar rate configured.

With overlapping rates, audio works even if one side has TCH/H with HR3 and the other TCH/F with FR3.

This makes me curious whether we can also get 3G (so far 12k2) to work against TCH/H HR3, possibly with lower rates configured in the IuCS RAB Assignment.

---

**#41 - 09/12/2019 11:13 AM - laforge**

On Thu, Sep 12, 2019 at 01:30:57AM +0000, neels [REDMINE] wrote:

In fact, this was the only setting I tested that managed to crash the sysmoBTS.

osmo-bts should receive a corresponding fix that drops such AMR packets before feeding them into the DSP. I think the fix can be made in the generic part as it doesn't ever make sense to accept RTP frames of invalid AMR modes which are not part of the current active set.

This makes me curious whether we can also get 3G (so far 12k2) to work against TCH/H HR3, possibly with lower rates configured in the IuCS RAB Assignment.

I don't see any reason why not, it's just a matter of choosing overlapping codec modes.

---

**#42 - 09/12/2019 01:49 PM - keith**

neels wrote:

In a network where all BSCs are controlled, it is not even necessary to communicate the individual rates via SDP: apparently AMR does indeed sort itself out nicely as long as both sides' rate support have overlaps.

Sure, but It might be desirable.. well.. if it's desirable..

See 3GPP TS 26.103 [28] for preferred AMR and AMR-WB configurations for operation in GSM and 3GPP UMTS networks. In gateway scenarios, encoders can be requested through the "mode-set" parameter to use a limited mode-set that is supported by the link beyond the gateway.


:)
Related: I would also like to change osmo-msc to late assignment (MO Assignment after CC Alerting), which I think would be fairly quick for me to implement in osmo-msc.

But I am discouraged by what that means for the test suite; particularly if I want ttcn to pass both the old and the new osmo-msc, as I am currently trying to do in this ttcn pass, to avoid fallout in the "latest" runs and so forth.

It's a bit disappointing to omit a nice feature just because it is hard to adjust the test suite, or that there is no time left because of that...

#44 - 09/19/2019 07:22 AM - laforge

On Wed, Sep 18, 2019 at 11:50:57PM +0000, neels [REDMINE] wrote:

I don't currently see any ways to shorten the dev cycle with ttcn or magically make things easier to modify...

The TTCN-3 functional tests exists as automatic test suite to catch regressions. Their benefit is in continuously testing things to ensure things don't break. Therefore, the benefits primarily are long-term, as you can ensure a certain behavior doesn't change months or years from now - especially at a time where the original developer[s] involved might not be around, available, ...

It's a bit disappointing to omit a nice feature just because it is hard to adjust the test suite, or that there is no time left because of that...

The same effort would have to be invested for any test suite, whenever suddenly the expected behavior changes significantly from how it used to be. It doesn't matter in which programming language the tests are implemented.

The tests are typically written rather tightly to ensure the expected behavior is verified fully. If of course you change all of the expectations all of a sudden, you need to rewrite the tests accordingly.

As I mentioned I don't know how many times before, the proper process here is the other way around: You first adjust the test suite to the new expectations, and then you change the actual implementation. The benefit is that the changes in the implementation are then verified with very quick turn-around times, much faster than you could ever do with manual testing.

I'm somewhat puzzled that we still appear to see people doing it in the inverse order: First changing the implementation with manual testing (trials), and then changing the test suite. That seems completely weird to me, and I would argue it maximizes the amount of time you first spend on changing the code, i.e. you don't get the benefit of fast-turnaround time of iterations while doing the actual development. Hence, it is a waste of time.

#45 - 09/26/2019 07:43 PM - neels

laforge wrote:

be. It doesn't matter in which programming language the tests are implemented.

Of course that matters, each testing technology has different dev cycle time and complexity to adjust.

If of course you change all of the expectations all of a sudden, you need to rewrite the tests accordingly.

The frustrating bit is that I merely moved a single CRCX message to an earlier point in time. The change seems very trivial, and has surprisingly large impact on the test suite.

On the up-side, while trying to adjust the CRCX, I managed to catch hidden errors in our MGCP handling: because I tried to take a fast track, an incomplete MGCP message managed to crash osmo-msc. I was able to fix it, as will soon be seen in gerrit..

As I mentioned I don't know how many times before, the proper process here is the other way around: You first adjust the test suite to the new expectations, and then you change the actual implementation.
[manual tests are] a waste of time.

I accept that test driven development is a good practice, and for sufficiently bounded problems, that works well. But it definitely is not the holy grail solution for every problem. Before I can just sit down and write tests, I need to find out what actually works and decide for a good solution.

I acknowledge the value of the ttcn tests, it is probably the best solution for long term stability. I merely would like to inform that I am taking longer than expected because of the ttcn tests structure, which turns out to be more complex than the osmo-msc changes themselves.

#46 - 10/04/2019 08:47 AM - keith
- Related to Bug #3650: MSC is not sending a payload_type to MNCC? added

#47 - 11/07/2019 11:56 PM - neels
- % Done changed from 70 to 80

I long overdue status update: the patches are done and I believe stable, not I am slowly getting them merged. I am already busy working on Distributed GSM, so I am checking on review and getting these patches merged as it comes up in-between other things.

One last thing is still open in osmo-sip-connector, where the recently added parsing/setting of the a:sendrecv / sendedonly / recvonly modes still needs to be applied to SDP that is otherwise now just forwarded between SIP and MSC. I think in the end I'll just again fully parse and re-compose the SDP instead of just forwarding the string as-is (the way the patch does it now and hence drops sendrecv modes).

#48 - 01/06/2020 04:34 PM - neels
- Status changed from In Progress to Stalled

osmo-msc with the new codec negotiation patches was running successfully at 36c3. We even got report of someone getting "HD" voice call quality on 3G, I guess AMR-WB was negotiated between phone and PBX? I should have grabbed a pcap of it, which I didn't, unfortunately.
(There was a bad OsmoMSC crash due to incoming invalid Mobile Identity, but in a very old code path and unrelated to the codecs patches.)

36c3 operation uncovered a problem in the osmo-sip-connector part, i.e. also unrelated to OsmoMSC, which was successfully worked around at congress; a proper fix is identified (basically due to a not fully finished patch there) and I still need to implement that, in osmo-sip-connector.

A problem for merging this effort is patch review: the patches are complex and need substantial effort by reviewers to make sense of. I hope that 36c3 operation gives more confidence that the OsmoMSC patches do the right thing...

#49 - 01/06/2020 06:03 PM - laforge
On Mon, Jan 06, 2020 at 04:34:42PM +0000, neels [REDMINE] wrote:

In theory, this should be relatively easy to reproduce by using a SIP UA [only] offering 16bit codecs?

#50 - 01/07/2020 03:06 PM - neels

(In my own tests things work out, but I would have liked to verify what the phone saying "HD" in the display actually got for codec...)

#51 - 01/07/2020 04:52 PM - keith

laforge wrote:

I'm pretty sure that the The B-leg (MT) of the call originating from the POC going via kamailio/rtpengine is offered with AMR only, not AMR-WB.

that's not to say that whatever signalling is going on might cause the a handset to display "HD" on the screen, but the codec coming from Kamailio
was always AMR/8000/1/12.20k (at least it was last year, and as far as I am aware nobody touched the kamailio box)

Anyway, that's of minor importance I think. I am going to try to find time and wherewithall to catch up, or should I say start up, with my knowledge and understanding of MSC MGW, BSC channel allocations and all this signalling, hopefully can help out somewhat with the code review.

#52 - 03/15/2020 02:49 PM - laforge

what is the status here? Unless there are pending known issues, can we please get the patches rebased + re-pushed to gerrit?

#53 - 03/23/2020 03:18 PM - neels

Remaining issues:

- biggest show stopper: need CR+2 on https://gerrit.osmocom.org/c/osmo-msc/+/15953
  This is the large patch that I am not planning to split up, but which doesn't get any positive review because of that.
  The discussion goes: "please split up, I can't read that much" vs. "it takes huge amount of time and inventing intermediate fixes of problems that
  no-one will benefit from in the end, please rather spend the time to read it as a whole".
  Maybe it could make sense to split into intermediate patches that don't have to work? Just as a reader's guide?

- The sipcon side of the patch passes SDP thru to the PBX 1:1, which cuts out the call-hold patches that keith added relatively recently. That's a
  fairly trivial part to fix.

I just realize that we already merged the MNCC version bump, so I can indeed merge the sipcon patch without needing to wait for the MSC part.

Question is: do we currently have sysmocom hours available for this topic?

#54 - 06/08/2020 03:11 PM - neels

- Priority changed from Urgent to Low

#55 - 09/23/2020 08:47 AM - keith

I pushed a patch to https://gerrit.osmocom.org/#/c/osmo-sip-connector/+/16222/ for the sipcon side.

#56 - 09/23/2020 02:02 PM - neels

On Wed, Sep 23, 2020 at 08:47:54AM +0000, keith [REDMINE] wrote:

I pushed a patch to https://gerrit.osmocom.org/#/c/osmo-sip-connector/+/16222/ for the sipcon side.

please note, I also keep a version of this patch on a branch in
osmo-sip-connector.git that is being rebased onto master regularly, and might
even have changes not shown there.

#57 - 06/12/2021 07:32 AM - laforge

- Related to Bug #5177: respect dynamic RTP payload types of external SIP added

Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>msc_mncc_mt_sip_mgcp.png</td>
<td>120 KB</td>
<td>03/24/2018</td>
<td>laforge</td>
</tr>
<tr>
<td>msc_mncc_mo_sip_rel18x_mgcp.png</td>
<td>149 KB</td>
<td>03/26/2018</td>
<td>laforge</td>
</tr>
<tr>
<td>msc_mncc_mo_sip_mgcp.png</td>
<td>142 KB</td>
<td>03/26/2018</td>
<td>laforge</td>
</tr>
<tr>
<td>codecs.png</td>
<td>53.9 KB</td>
<td>08/14/2019</td>
<td>neels</td>
</tr>
<tr>
<td>codecs2.png</td>
<td>46.2 KB</td>
<td>08/14/2019</td>
<td>neels</td>
</tr>
<tr>
<td>codecs3.png</td>
<td>56.9 KB</td>
<td>08/15/2019</td>
<td>neels</td>
</tr>
<tr>
<td>codecs4.png</td>
<td>65.1 KB</td>
<td>08/15/2019</td>
<td>neels</td>
</tr>
<tr>
<td>codecs5.png</td>
<td>132 KB</td>
<td>08/28/2019</td>
<td>neels</td>
</tr>
</tbody>
</table>