This patch [1] in practice reverted a previous patch [2] which was submitted a for a reason.

The reason is that we want to match the exact AMR Payload Type in first place, in order to avoid AMR transcoding OA<->BE if not needed.

This can happen for instance if connection A in the endpoint announces through SDP that it supports only let’s say:

- PT=100 AMR octet_aligned=1

And connection B, the other connection in the same endpoint, announces it supports both in SDP, let’s say:

- PT=102 AMR octet_aligned=0
- PT=103 AMR octet_aligned=1

After [1] is merged, the code, when forwarding RTP connA->connB, will select the PT of first AMR codec found in conn regardless of the octet_aligned, that is PT=102, which means it will end up later on having to transcode OA->BE. The ideal thing to do in this situation would be to pick PT=103, so that transcoding can be avoided when sending packets on that direction, hence freeing CPU use.

In order to do so, it’s quite easy, we just need to modify a bit patch [1] to incorporate the idea of [2]. So first we do a lookup as in [2], checking exact match, and if not found, do as in [1] (no AMR octet_aligned match).

This can be generalized as “first try a match not requiring transcoding, and if not found, try a match which requires transcoding”.

It can be implemented with something like this:

```c
diff --git a/src/libosmo-mgcp/mgcp_codec.c b/src/libosmo-mgcp/mgcp_codec.c
index 38aa0a79f..f6dfb66b 100644
--- a/src/libosmo-mgcp/mgcp_codec.c
+++ b/src/libosmo-mgcp/mgcp_codec.c
@@ -366,7 +366,7 @@
     bool mgcp_codec_amr_is_octet_aligned(const struct mgcp_rtp_codec *codec)

- static bool codecs_same(struct mgcp_rtp_codec *codec_a, struct mgcp_rtp_codec *codec_b)
+ static bool codecs_same(struct mgcp_rtp_codec *codec_a, struct mgcp_rtp_codec *codec_b, bool allow_transcoding)
{
     if (codec_a->rate != codec_b->rate)
         return false;
@@ -378,10 +378,10 @@
     else
         return false;
     
-     /* Note: AMR allows to set the RTP payload format to octet-aligned or bandwidth-efficient (octet-aligned=0)
-        * via SDP. This difference concerns payload format only, but not the actual codec. It is not a difference
-        * within the meaning of this function. */
+     /* Note: AMR allows to set the RTP payload format to octet-aligned or bandwidth-efficient (octet-aligned=0)
+        * via SDP. This difference concerns payload format only, but not the actual codec. It is not a difference
+        * within the meaning of this function. */
```
We should add TTCN3/unit tests validating scenarios like the one described above, and make sure we do the intelligent thing.

[1] https://gerrit.osmocom.org/c/osmo-mgw/+/27223

History

#1 - 02/23/2022 12:13 PM - neels
Sounds good to me, thanks for the explanation, Pau!

#2 - 03/03/2022 02:15 PM - dexter
- Status changed from New to In Progress
- % Done changed from 0 to 50

To ensure that we do not run again in the bwe<->oa conversion problem we have seen recently I have increased the test coverage in the TTCN3 tests. When I revert my previous patch, then TC_amr_oa_bwe_rtp_conversion fails, which shows that it works.

https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/27384 MGCP Test: ensure PT translation works when converting AMR bwe/oa

I have now changed the mgcp Codec pt_translate() and the related unit tests. The behavior is only slightly changed, but I don't think that this is of any practical relevance.

https://gerrit.osmocom.org/c/osmo-mgw/+/27391 Revert "mgcp Codec: do not differentiate between oa and bwe when comparing ...
https://gerrit.osmocom.org/c/osmo-mgw/+/27392 mgcp Codec: fix oa/bwe comparison in mgcp Codec pt_translate()

We probably might investigate though what really happens when one side has for example octet-aligned and the other side has both payload formats assigned. I doubt that this will work with the current code. For that kind of case we will need a TTCN3 test and probably make the choice of the egress codec a little more dynamic. Apart from that I think this fix is restoring the intended behavior as it was.

When I am understanding the issue correctly, then the concern was that a PBX on the SIP side may assign two AMR modes with two different PT numbers for compatibility reasons.
So a concrete scenario would be:

On the osmo-cn side use e.g. PT 112 for AMR in oa.

On the SIP side we use PT 113 for AMR in oa and PT 114 for AMR in bwe.

===> The MGW will notice that 112<->113 is requiring no format conversion, it will send out the packets to the SIP side under PT 113, while sending no packets under PT 114 at all. Any bwe packet that it receives under PT 114 will be converted to oe and forwarded to the osmo-cn side under PT 112. Any packet that osmo-mgw receives under PT 113 will just forwarded unchanged to the osmo-cn side under PT 112 as it is already oa formatted.

Yes correct, that's the scenario I had in mind.

So a concrete scenario would be:

On the osmo-cn side use e.g. PT 112 for AMR in oa.

You probably mean the osmo-bss side here, since the one handling SIP is the MSC (which is the CN).

On the SIP side we use PT 113 for AMR in oa and PT 114 for AMR in bwe.

===> The MGW will notice that 112<->113 is requiring no format conversion, it will send out the packets to the SIP side under PT 113, while sending no packets under PT 114 at all. Any bwe packet that it receives under PT 114 will be converted to oe and forwarded to the osmo-cn side under PT 112. Any packet that osmo-mgw receives under PT 113 will just forwarded unchanged to the osmo-cn side under PT 112 as it is already oa formatted.

Yes, that's the ideal solution IMHO.

I think it should be fairly easy to write a TTCN3 test validating that exact scenario, and then we make sure we have no regressions there, which may be difficult to spot.