OsmoMGW - Feature #1936

Implementation of IuUP SMpSDU mode

02/02/2017 01:07 PM - laforge

Status:	Stalled	**Start date:**	02/02/2017
Priority:	Normal	**Due date:**	
Assignee:	laforge	**% Done:**	20%
Category:			
Target version:			

Description

We'll need an implementation of the IuUP protocol described in 3GPP TS 25.415 and TS 25.414 for the RTP encapsulation.

Particularly the SMpSDU mode is important here.

The code should come as a library so it can be used by multiple different applications. It could implement the primitives described in Section 7 of TS 25.415 towards the user (RNL).

Related issues:

Precedes OsmoMGW - Feature #1937: Implement way how to handle IuUP on RTP end...	**New**	**02/03/2017**	**02/03/2017**

History

#1 - 02/02/2017 03:40 PM - neels

- Related to Feature #1937: Implement way how to handle IuUP on RTP endpoints added

#2 - 02/02/2017 07:44 PM - laforge

- Assignee changed from Osmocom CNI Developers to laforge

- % Done changed from 0 to 10

#3 - 10/11/2017 12:53 AM - laforge

- Project changed from Cellular Network Infrastructure to OsmoMGW

- Assignee changed from laforge to dexter

#4 - 10/11/2017 12:56 AM - laforge

The laforge/iu_up branch of libosmocore contains some initial untested code for the CRC generation as well as for the primitives and a related FSM. This should be used as a base for the library code. This library code can then be used from osmo-mgw.

#5 - 11/18/2017 08:27 PM - laforge

- Related to deleted (Feature #1937: Implement way how to handle IuUP on RTP endpoints)

#6 - 11/18/2017 08:27 PM - laforge

- Precedes Feature #1937: Implement way how to handle IuUP on RTP endpoints added

#7 - 12/23/2017 07:34 PM - laforge

- Assignee changed from dexter to laforge

- Priority changed from Normal to Urgent

#8 - 12/23/2017 08:53 PM - laforge

- Status changed from New to In Progress

- % Done changed from 10 to 20

Implemented an encoder/decoder for IuUP accordint to TS 25.415 Chapter 6.6 in TTCN-3 for implementation of a test fixture. Next will be that test fixture with some inspection if wireshark decodes the payload correctly. After that, the fixture can be used towards testing the upcoming osmocom implementation.

#9 - 05/30/2018 02:53 PM - laforge

- Tags set to 3G

#10 - 07/04/2018 12:44 PM - laforge

- Status changed from In Progress to Stalled

#11 - 10/10/2018 03:21 PM - laforge

see also http://lists.osmocom.org/pipermail/openbsc/2018-October/012255.html:

```
On Wed, Oct 10, 2018 at 01:18:39PM +0200, Neels Hofmeyr wrote:
> So I think there's still some fundamental concept that I'm lacking. Is there
> anyone more familiar with AMR and/or the way 3G encodes audio, and whether
> there's a simple way to make them match? Where should I read on?

The fundamental part that you're lacking is the fact that the codec payload
for 3G and 2G is completely different.  You will need lots of bit re-shuffling
in order to make this work.

>From the MGW point of view, IuUP and the 3G-style codec payload should be seen
as a different codec, and an endpoint with two connections of two different codec
requires "transcoding".

The only difference here is that in case of AMR on both the IuUP and the 2G side,
you don't actually transcode to PCM and re-encode, but you're really just shuffling
around the bits.

> Would a call router like we use in the C3 POC be able to transcode when the
> IuUP is stripped? (I'm not even sure what the POC side is doing to connect SIP
> with GSM.)

Nobody outside a 3G network will have any idea what IuUP is.  Even more so, I think
nobody on the planet will be able to process codec payload that is in IuUP payload
format without the IuUP header.  Basically either

a) you have IuUP and the related payload format, or

b) you have no IuUP and native RTP [AMR] payload format

Going in between by just removing the IuUP header you would create yet another
derivative that doesn't exist so far.

> I've noticed the laforge/iu_up branch in libosmocore only later,

It would have been great to first catch up about this, before investing time
on an implementation.  I had mentioned this branch in
http://osmocom.org/issues/1936 when I wrote it close to a year ago.

> which includes an FSM that apparently does only state transitions so far.

It does a bit more, such as both header and payload CRC computation /
verification and should actually implement pretty much everything
between the TNL and the RNL SAP and implement all primitives on both
sides.  It hasn't been tested yet, though.  To do that, I first
implemented RTP source/sink in TTCN3 and then IuUP in TTCN3, but then
got distracted before putting the full setup together and write actual
test cases against the libosmocore iu_up impementation.

You should be able to

* feed primitives from the transport layer (RTP) up into it using
  osmo_iuup_tnl_prim_up()
* feed primitives from the upper layer down into it using
  osmo_iuup_rnl_prim_down()

The transformations are done inside the FSM using tnl_to_rnl_data() as
well as rnl_to_tnl_data().  Only SMpSDU mode is implemented, which is
what's used in our case.

> My patch has no FSM yet, since all I see on the wire is Init->InitAck,
> then Data PDUs, and maybe an occasional error report. Do those FSM
> states convey a secret of making 3G encoding readable by 2G?
```

```
The state machines are required as soon as you actually have changes between the
AMR codec rates, i.e. the actual *adaptive* part of AMR.

Also, once you are actually re-ordering ("transcoding") the payload
bits, you will have to recompute the CRC.  I do think the FSM should be
used, rather than some other approach.
```

as well as http://lists.osmocom.org/pipermail/openbsc/2018-October/012256.html:

```
in terms of the actual payload, I believe (IIRC), that AMR over IuUP
used three sub-flows.  They are encoded after each other, see
TS 25.415 6.6.3.27 / Figure 27a for an illustration.

Thse SDUs of each sub-flow contain the bits of one class.

The rationale for teh above lies in the structure of the RAB and the
channel bundles on the Iu radio layer.  Both on the radio and on the
IuFP side between NodeB and RNC, there are actually three independent
streams of bits, one for each of the classes.

For IuUP, it seems they are then re-combined in the RNC into one IuUP
payload, but still the three separate chunks accordign to their class.

In order to get the regular RTP-AMR payload, they need to be
mixed/mangled into the order specified by whatever RFC specifies the AMR
payload format, I think it as https://tools.ietf.org/html/rfc3267 -
whcih then refers to the IF1 format as specified in 3GPP TS 26.101,
where IF1 is described in Section 4.

The order of the fields here is "logical", so if there's a given
parameter that has 6 bits, then those 6 bits are consecutive.  In the
air interface, you may want to transmit the first two (MSB) bits in
class A, the middle two in class B and the last two bits in class C.
This is so the most improtant bits for speech recovery are given higher
amount of forward error correction than the less important bits.

I believe the tables in Annex B of 26.101 is what's needed in terms of
re-ordering.

Still, there's plenty of things that can go wrong in terms of bit-order
within a byte, byte ordering, ..., and hence I think it's good to start
with writing functions and unit tests for this first.  GAPK might be a
good place for prototyping, as it already understands the concept of
different representations/bit-orders/formats for one given codec, and it
has support for playback via alsa, ...
```

#12 - 10/11/2018 11:23 AM - neels

- File signature.asc added

On Wed, Oct 10, 2018 at 03:21:13PM +0000, laforge [REDMINE] wrote:

> b) you have no IuUP and native RTP [AMR] payload format

^ that's the aim.

> > I've noticed the laforge/iu_up branch in libosmocore only later,
>
> It would have been great to first catch up about this, before investing time
> on an implementation. I had mentioned this branch in
> http://osmocom.org/issues/1936 when I wrote it close to a year ago.

agreed, but my patch mostly deals with osmo-mgw and being able to change
the payload "before" receiving / "after" sending.

I did "re-implement" the checksums part, by copying a CRC table from
wireshark sources (with attribution...). Am so far not familiar with the
osmo_crc API, but I wonder whether crc16 is possible to be re-used; I had
to do special handling for calculating the remainder, because the CRC
processing has to stop after 10 bits in the end...

Anyway, incorporating the FSM in there shouldn't pose much of a conflict / re-implementation.

> used three sub-flows.

If I read the pcaps correctly I don't see a lot of complexity in the IuUP headers, i.e. nothing that would indicate multiple flows... (not sure though).

> In order to get the regular RTP-AMR payload, they need to be mixed/mangled into the order specified by whatever RFC specifies the AMR payload format, I think it as https://tools.ietf.org/html/rfc3267 - whcih then refers to the IF1 format as specified in 3GPP TS 26.101, where IF1 is described in Section 4.
> [...]
> I believe the tables in Annex B of 26.101 is what's needed in terms of re-ordering.

ok, maybe I'll find some time to hack on this, maybe only in the days leading up to congress...

All of this is not really urgent to me, just for fun so far, to distract a bit from all the serious handover code.

~N

#13 - 10/11/2018 05:37 PM - laforge

neels wrote:

> > used three sub-flows.
>
> If I read the pcaps correctly I don't see a lot of complexity in the IuUP headers, i.e. nothing that would indicate multiple flows... (not sure though).

I seriously doubt that. I think in AMR you have a minimum of thwo sub-flows, and typically you have three sub-flows. See the below example from a nano3g:

```
example from nano3G:

160051673C012700008200000017100001 00

16: TI=1, 3 sub-flows per RFCI, chain=0

00: LRI=0(not last), LI=0(8bit), RFCI=0
51: length of sub-flow 1: 81 bits
67: length of sub-flow 2: 103 bits
3C: length of sub-flow 3: 60 bits

01: LRI=0(not last), LI=0(8bit), RFCI=1
27: length of sub-flow 1: 39 bits
00: length of sub-flow 2: 0 bits
00: length of sub-flow 3: 0 bits

82: LRI=1(last), LI=0(8bit), RFCI=2
00: length of sub-flow 1: 0 bits
00: length of sub-flow 2: 0 bits
00: length of sub-flow 3: 0 bits

17: IPTI-RFCI-1: 1, IPTI-RFCI-2: 7
10: IPTI-RFCI-3: 1, padding
0001: IuUP versions supportd: only '2'
00: data PDU type 0
```

#14 - 10/11/2018 05:41 PM - laforge

to interpret the above nan3G example:

- you have three different AMR configurations:
  - RFCI0 with 81A-103B-60C bits
    - compare that with 26.101 table 7, this is amr frame type 7 (12.2 kBps)
  - RFCI1 with 39A-0B-0C bits
    - compare that with 26.101 table 7, this is amr frame type 8 (SID UPDATE/FIRST/BAD)
- RFCI2 with 0-0-0 bits (basically muted?)

So this means that the AMR codec configuration for this call only permits 12.2k rate or SID (or nothing?). The codec configuration expressed in IuUP must of course follow/match the RAB assignment on IuCS.

#15 - 10/12/2018 09:37 AM - neels

On Thu, Oct 11, 2018 at 05:37:06PM +0000, laforge [REDMINE] wrote:

> > If I read the pcaps correctly I don't see a lot of complexity in the IuUP
> > headers, i.e. nothing that would indicate multiple flows... (not sure
> > though).
>
> I seriously doubt that. I think in AMR you have a minimum of thwo sub-flows, and typically you have three sub-flows.

Oh, right -- for some reason I thought I had seen only one RFCI and pretty much
zero for the rest -- but I see them now. My mistake.

~N

#16 - 10/17/2018 10:14 AM - laforge

- Priority changed from Urgent to Normal

Files

signature.asc	833 Bytes	10/11/2018	neels