# libosmo-netif - Bug #3180

## Osmux: Improve scheduling generated RTP packets from received osmux frames

04/16/2018 05:09 PM - pespin

| | | | |
|---|---|---|---|
| **Status:** | Resolved | **Start date:** | 04/16/2018 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | pespin | **% Done:** | 100% |
| **Category:** | | | |
| **Target version:** | | | |
| **Spec Reference:** | | | |

**Description**

Nowadays, the typical procedure for a user of the Osmux API which reads osmux frames is to convert them into rtp message buffers and schedule them this way:

```
    while((osmuxh = osmux_xfrm_output_pull(msg)) != NULL) {
        osmux_xfrm_output(osmuxh, &h_output, &list);
        osmux_tx_sched(&list, tx_cb, NULL);
    }
```

So, in first line we get the parsed osmux frame from the UDP packet (there can be several osmux frames for different CIDs in one UDP packet).
On the second line, osmux_xfrm_output loops over the AMR payloads and created a msgb for each of them, and stores them in a list.
The third one takes the list and for each elemnt in the list it removes them and schedules the packets (one timer per packet) to be sent to the callback function in spaces of 20ms.

This current implementation has a major issue, specially when jitter is high. In that scenario, it may happen that two consecutive osmux frames A1 and A2 arrive to the receiver almost at the same time T0 (let's take the worst case).

As a result, the current code will schedule the AMR payloads (array indices) in the following way (we assume a batch factor of 4 here):
(A1 is received:)
$A1^0$ -> T0 (rtp seq X)
$A1^1$ -> T0+20 (rtp seq X+1)
$A1^2$ -> T0+40 (rtp seq X+2)
$A1^3$ -> T0+60 (rtp seq X+3)
(immediately after A2 is received:)
$A2^0$ -> T0 (rtp seq X+4)
$A2^1$ -> T0+20 (rtp seq X+5)
$A2^2$ -> T0+40 (rtp seq X+6)
$A2^3$ -> T0+60 (rtp seq X+7)

which means the RTP receiver ends up seeing:
T0: seq=X, seq=X+4
T0+20: seq=X+1, seq=X+5
T0+40: seq=X+2, seq=X+6
T0+40: seq=X+3, seq=X+7

So you get a really interleaved and mess up list of packets.

This can easily avoided/improved by not scheduling the packets with its own timer independently, but keeping them in the queue and using only 1 timer per circuit, and associating a timeout value to each packet, which may be tweaked to fit better a resulting output list. When the circuit timer fires, it rearms itself using the value from the first element in the list.

For instance, in the scenario explained before, when A1 arrives, we just put all the generated rtp message buffers in the list, and we schedule the 1st one to be send immediately. When we receive A2, we still have most packets in the list, so we can insert them in

order in the list based on seq number, and tweak the previous packet timing values to send them quickly as we anyway have new values (which means the first packet arrived late, and we don't want to accumulate delay).

## History

#### #1 - 04/18/2018 05:26 PM - pespin

*- Status changed from New to Feedback*

*- % Done changed from 0 to 90*

Should be fixed by https://gerrit.osmocom.org/7869 && tested by https://gerrit.osmocom.org/7870

#### #2 - 04/18/2018 05:29 PM - pespin

*- Status changed from Feedback to In Progress*

*- % Done changed from 90 to 80*

I still need to integrate the new APIs into openbsc/osmo-bsc and osmo-bsc-nat.

#### #3 - 05/03/2018 09:40 AM - pespin

Patches merged. Harald fixed a bug introduced with the commit in libosmo-netif ef1900151703e4d6b16cb434c3ae688fd7fddc55.

usage integration done in openbsc 36c51b4916ab6d321ec47360ef1cb047b4acc2bf

TODO:
- Test in osmo-bsc-nat setup
- Forward port to osmo-bsc.git/osmo-mgw.git

#### #4 - 06/22/2018 10:17 AM - pespin

*- Status changed from In Progress to Resolved*

*- % Done changed from 80 to 100*

Patches tested and being used nowadays in osmo-bsc-nat + osmo-bsc-sccplite.

Forward ports done, also applied in osmo-mgw.

Closing the task.