libosmo-netif - Bug #2439

Osmux: overwrite duplicated RTP packets instead of dropping them

08/14/2017 05:10 PM - pespin

Status:	New	**Start date:**	08/14/2017
Priority:	Normal	**Due date:**	
Assignee:	pespin	**% Done:**	0%
Category:			
Target version:			
Spec Reference:			

Description

- There's a check in osmux.c osmux_batch_add() which checks if the RTP packet arriving is already in the Osmux batch (by comparing seq num), and in case it's found, the newly arrived packet is discarded.
- In osmux there code too to re-create lost RTP packets by detecting gaps and cloning the last received RTP packet.

Both statements don't play well together, because this means a lot of times we have unordered packet, eg. A, B, C, D:

1. A received -> added to the batch.
2. D received (out of order) -> a gap is detected, so A is cloned into B and C.
3. B received -> it is found in the batch, so it's discarded.
4. C received -> it is found in the batch, so it's discarded.

Which means we end up with payload AAAD instead of ABCD we could easily have by overwriting the received payload in the batch, eg. same scenario:

1. A received -> added to the batch.
2. D received (out of order) -> a gap is detected, so A is cloned into B and C.
3. B received -> it is found in the batch, so previous B (cloned A) is freed and new B payload is put instead in its position.
4. C received -> it is found in the batch, so previous C (cloned A) is freed and new C payload is put instead in its position.

I can see this in an environment I have set up with osmo-bsc_mgcp and osmo-bsc_nat:

```
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:513 adding cloned RTP
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:513 adding cloned RTP
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:513 adding cloned RTP
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:513 adding cloned RTP
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:619 already exists message with seq=
36463, skip it
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:619 already exists message with seq=
17, skip it
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:226 too many messages for this RTP s
src=37798033
...
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:513 adding cloned RTP
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:513 adding cloned RTP
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:619 already exists message with seq=
22128, skip it
<0023> /home/data/pespin/bin/../git/libosmo-netif/src/osmux.c:619 already exists message with seq=
51217, skip it
```

I plan to work on this once I am done with a change to improve the code handling lost packets (patch is done but I'm facing some issues).

Fixing this issue I think would improve audio noise during the calls.

Related issues:

Blocked by libosmo-netif - Bug #2440: Osmux: Improve lost packet recreation m...	New	08/14/2017

History

#1 - 08/14/2017 05:17 PM - pespin

- Description updated

#2 - 08/14/2017 05:27 PM - pespin

- Blocked by Bug #2440: Osmux: Improve lost packet recreation mechanism added

#3 - 08/14/2017 05:28 PM - pespin

#4 - 08/14/2017 07:45 PM - ipse

I'm curious why do you even need to duplicate packets to cover lost packets? Normally it's a function of a PLC (packet loss concealment) algo inside of the end device. And proper implementations of codecs like GSM and even more AMR should have quite complicated algorithms to make sure concealment is as much un-noticed as possible. Duplicating packets is the most basic packet concealment algorithm and i would expect at least that to be implemented even in cheapest Chinese phones, so why do that manually in osmux, complicating code and introducing issues like this?

#5 - 08/16/2017 01:48 PM - pespin

That's due to the way Osmux works, because we group several RTP packets in one Osmux frame, and we drop related information from all the RTP packets (we only keep the AMR payload) and we try to keep up with only 1 osmux header instead.

That means if you have a batch factor of X, you drop X RTP seq and timestamps and in exchange only use 1 Osmux seq to identify them all. On the receiver side, the Osmux code maintains internal state of seq + timestamp counters to use when extracting the AMR payload and re-generating the RTP packets. So currently there's no way in Osmux frames to account for dropped packets or "audio gaps". As the receiver cannot know, it will continue assigning seq numbers and timestamps as if there was no lost but instead the packet arrived late.

So, we need to ensure we don't create gaps inside Osmux frames.

The situation could be improved by adding an extra byte in the header which contains a bitmask stating whether AMR payload in position N can be found in the Osmux frame or it was lost/dropped for any reason. A byte is enough because ctr is only 3 bits and thus max batch factor is 8. This way we can avoid copying AMR payloads and the receiver can just increase RTP seqnum + timestamp counters and keep going as if the RTP packet was lost. However, adding this byte would break protocol compatibility and I am not entirely sure if we want to do that.

#6 - 08/16/2017 10:06 PM - ipse

pespin wrote:

> That's due to the way Osmux works, because we group several RTP packets in one Osmux frame, and we drop related information from all the RTP packets (we only keep the AMR payload) and we try to keep up with only 1 osmux header instead.

Looks like a classical RTP header compression scheme. I'm curious - have you looked at ROHC? It already solves a lot of problems you're trying to resolve here and a free lib is available.

https://en.wikipedia.org/wiki/Robust_Header_Compression

https://tools.ietf.org/html/rfc3095#section-5.7
https://rohc-lib.org/

> That means if you have a batch factor of X, you drop X RTP seq and timestamps and in exchange only use 1 Osmux seq to identify them all. On the receiver side, the Osmux code maintains internal state of seq + timestamp counters to use when extracting the AMR payload and re-generating the RTP packets. So currently there's no way in Osmux frames to account for dropped packets or "audio gaps". As the receiver cannot know, it will continue assigning seq numbers and timestamps as if there was no lost but instead the packet arrived late.

Just memset(0) the frame and check for all 0's on the receive side. Voice frames can't be all-zero.

#7 - 08/16/2017 10:21 PM - ipse

re: ROHC
Have you looked at e.g. https://rohc-lib.org/wiki/doku.php?id=iprohc-overview ?

#8 - 08/16/2017 11:15 PM - laforge

On Wed, Aug 16, 2017 at 10:06:29PM +0000, ipse [REDMINE] wrote:

> Looks like a classical RTP header compression scheme. I'm curious - have you looked at ROHC? It already solves a lot of problems you're trying to resolve here and a free lib is available.

ROHC does not aggregate frames of multiple flows in one packet, does it?o

Also, ROHC is a link layer mechanism for a single link, and not something that can
still be routed over the internet.

#9 - 08/17/2017 12:09 AM - ipse

laforge wrote:

> On Wed, Aug 16, 2017 at 10:06:29PM +0000, ipse [REDMINE] wrote:
>
> > Looks like a classical RTP header compression scheme. I'm curious - have you looked at ROHC? It already solves a lot of problems you're trying to resolve here and a free lib is available.
>
> ROHC does not aggregate frames of multiple flows in one packet, does it?o

https://rohc-lib.org/wiki/doku.php?id=iprohc-overview
"In addition, frame packing is implemented: several ROHC packets are put together in one single IP header to reduce the tunnel overhead even more."

It's not exactly what you mean, but with 1 byte per most packets, it already gives you a very good compression ratio without increasing latency and without introducing large gaps in case an aggregated packets loss. This is assuming you have more than 1 simultaneous voice stream. In VoIP terms it's RTP trunking - like what IAX is doing.

If you still do want to pack more than one voice frame from the same stream into a single packet, it's normally handled by the sender itself through a "ptime" SDP setting. So ROHC and other VoIP people assume you don't really need to do that on a higher level.

Btw, you may want to send odd and even voice frames in different packets to improve voice quality in case of packet loss.

If you want to do a single voice stream voice frame aggregation in presence of incoming packet loss, I think it's wiser to introduce variable length packets. Then you pack as many frames as you want until you hit a jump in a timestamp (doesn't matter whether this is due to DTX or packet loss). Then you transmit this packet, wait for the next voice frame to arrive and start sending again. Considering that a single voice frame is multi-byte, you will most likely save bandwidth even if you need to increase a header size for such short packets.

> Also, ROHC is a link layer mechanism for a single link, and not something that can
> still be routed over the internet.

ROHC per se is not a protocol and can be sent over IP or UDP or pigeons if you decide so. It just provide you means to compress headers and describes how to handle packet loss situations with maximum efficiency for a given scenario (profile). Ideally it should be implemented inside a VSAT modem (and some vendors do that actually) which then gives maximum efficiency - all headers, from Ethernet to RTP can be compressed into 4 bytes.

#10 - 08/17/2017 12:11 AM - ipse

Btw, it's debatable, but my initial thought would be that seq num should be kept continuous for voice packets lost on the BTS' air interface. I.e. they should be treated the same way as DTX.

#11 - 08/17/2017 07:10 AM - laforge

Dear Alexander,

what is "normal" in your VoIP world and what is "normal" in the GSM world, particularly the "early IP based GSM world" compatible with what ip.access did are two different things. There's little point in discussing this.

OSmux is needed, for example also in combination with ip.access nanoBTSs which support no ROHC or having multiple codec frames of a single stream in one UDP packet.

This ticket is not about arguing whether or not OSmux is needed. It is needed and it exists for many years.

So for sure, strategies of helping resolving certain bugs are welcome, but anything else is unfortunately a distraction.