Potential race condition issue of GStreamer

[Test URL & settings]:

http://media-dcp.otvs.tv/storage/tears-of-steel/dash/live.main.isml/.mpd

[Issue]:

Wrong codec settings leads to mosaic (ex, use the codec setting of high bit-rate representation to decode data of low bit-rate representation).

[Root cause]:

1. Race condition between threads.
2. Slow start of gst_mpd_client_setup_streaming() when updating manifest.

[Background]

Generally we have three kinds of threads within adaptive demuxer for a live (type = dynamical) streaming; they are listed below.

1. Download thread: (as fig 1)

The flow controller to update fragment info, wait until the target fragment is available, create base source thread to download, wait for completeness, check EOS.

2. Update loop thread: (as fig 2)

Update manifest according "minimumUpdatePeriod".

3. Base source thread: (as fig 3)

Actually download bit-stream of each fragment (segment).

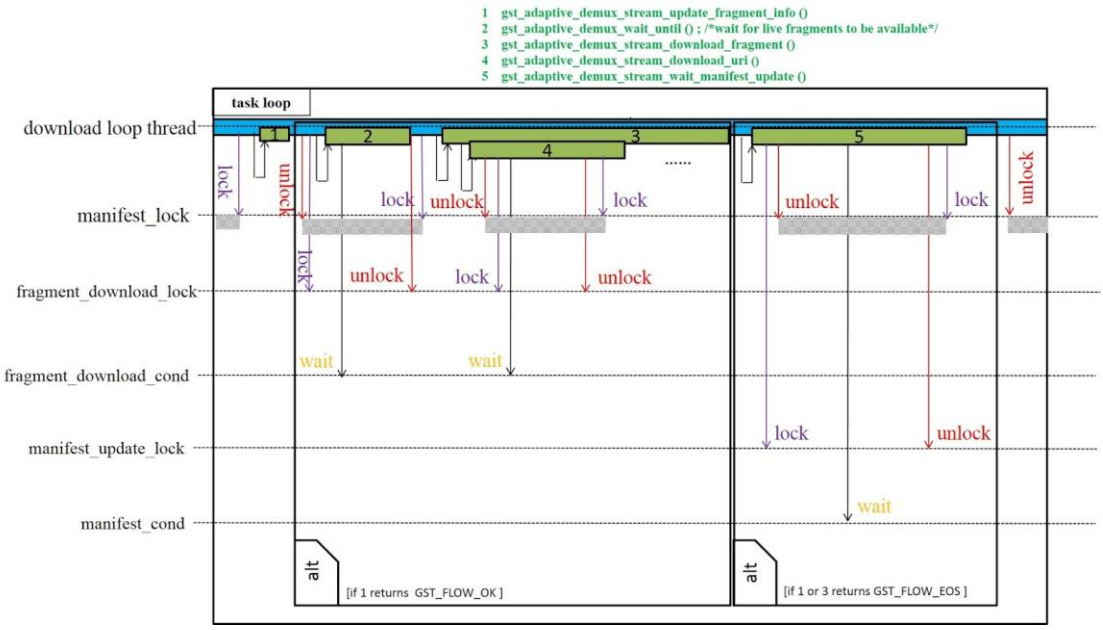

Fig 1: download loop thread

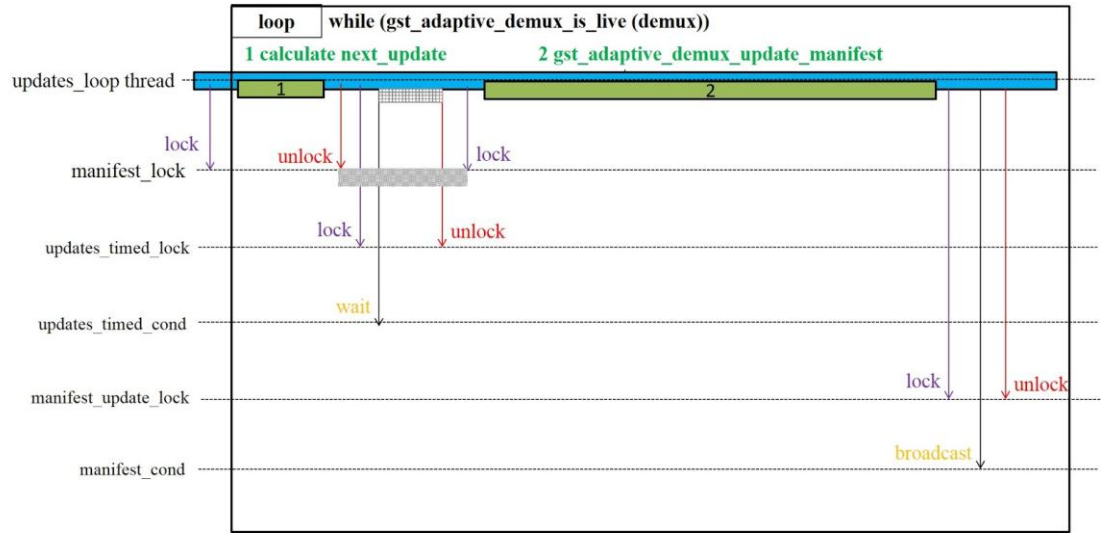

Fig 2: Update loop thread

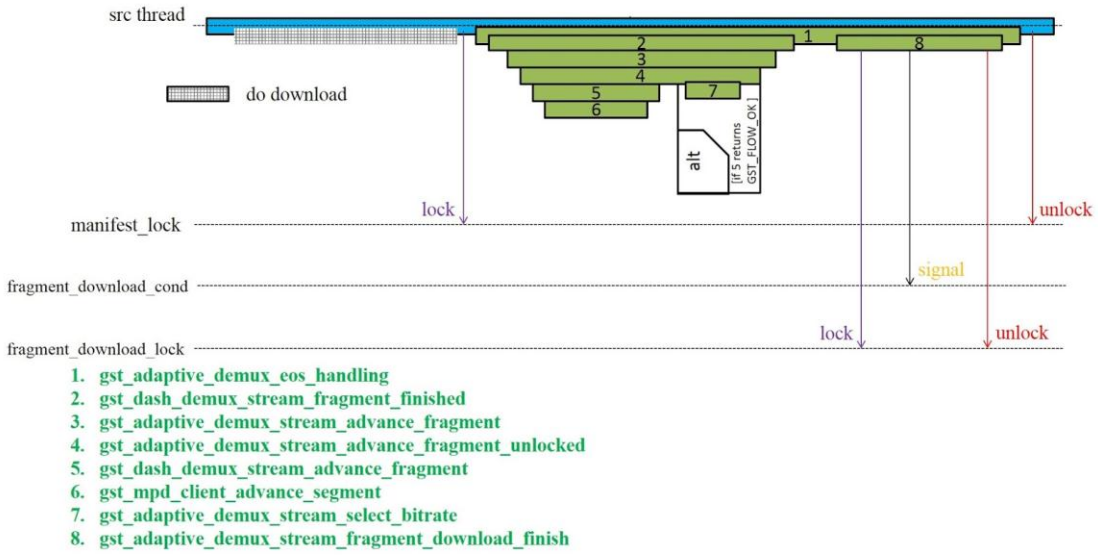

Fig 3: Base source thread

[Detailed description]:

Please refer to fig 4, an issued case is illustrated to understand where the problem arises.

Originally the representation is of 250000 bits/sec. Then the update loop thread locks the manifest_lock and by default launches from slow start (the lowest bit-stream) upon the updated manifest. It makes cur_representation to be of 125000 bits/sec.

After update done, manifest_lock is unlocked.
Since the download loop thread is still waiting for the signal from sec thread to inform the completeness of download, the next thread which will get manifest_lock is sec thread.

At _src_event when src thread has completed download,
gst_adaptive_demux_eos_handling() is executed with manifest_lock locked.

At B.4 gst_adaptive_demux_stream_advance_fragment_unlocked() is executed & the next download bit-rate is set to 125000 bits/sec. At
B.7 gst_adaptive_demux_stream_select_bitrate() (in
fact, gst_dash_demux_stream_select_bitrate ()) the check of "if (new_index != active_stream->representation_idx)" is **false**. It is because that the slow start of update

manifest has changed active_stream->representation_idx to the lowest one. **As the result, the new caps as well as the Boolean variable need_header will NOT be set.** **It makes the switch of bit-rate without re-passing necessary codec data.**

Finally, it leads to mosaic by applying wrong codec data (of 250000 bits/sec) to decode 125000 bits/sec. As at C1 where the next URL is composed of the lowest bit-rate = 125000 but we do ***NOT*** pass codec data & header down for this bit-rate switch.

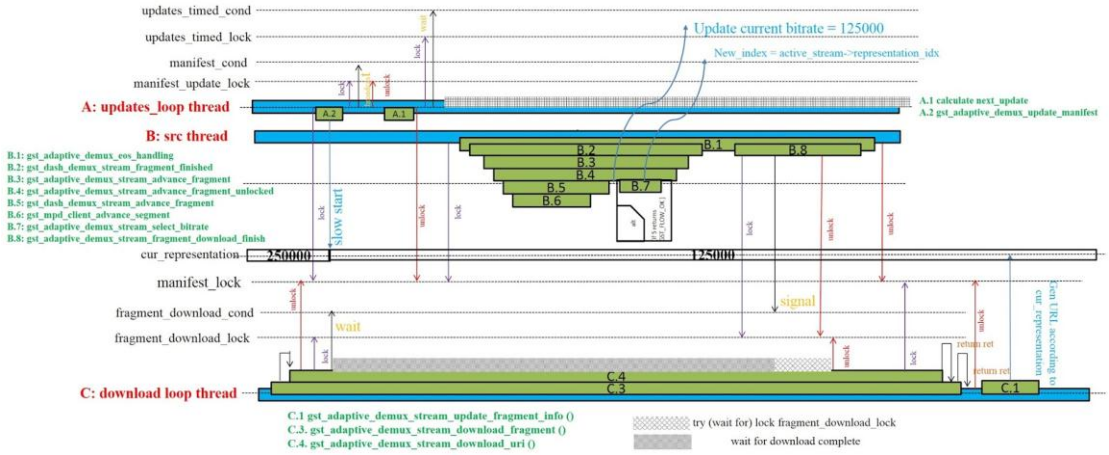

Fig 4: An issued case

[Proposed solution]:

As fig 1 & fig 5, to download next fragment, at first download loop thread will update fragment info (by gst_adaptive_demux_stream_update_fragment_info()). It results in the update to stream->fragment of the type GstAdaptiveDemuxStreamFragment which includes the URI info.

Once GstAdaptiveDemuxStreamFragment has been updated, it will NOT be changed. To take use of this fact, we keep the bit-rate we have downloaded previously and compare it to current target. If they are different, we pass the info of header & caps down.

To avoid passing redundant header & caps, we only do the check if
1. It is a live (type = dynamic) streaming.
2. If the "stream->need_header == FALSE" is TRUE.

Finally as figure 6, within gst_dash_demux_stream_update_fragment_info(), the bit-rate of current chosen representation could be known from: **dashstream->active_stream->cur_representation->bandwidth.**

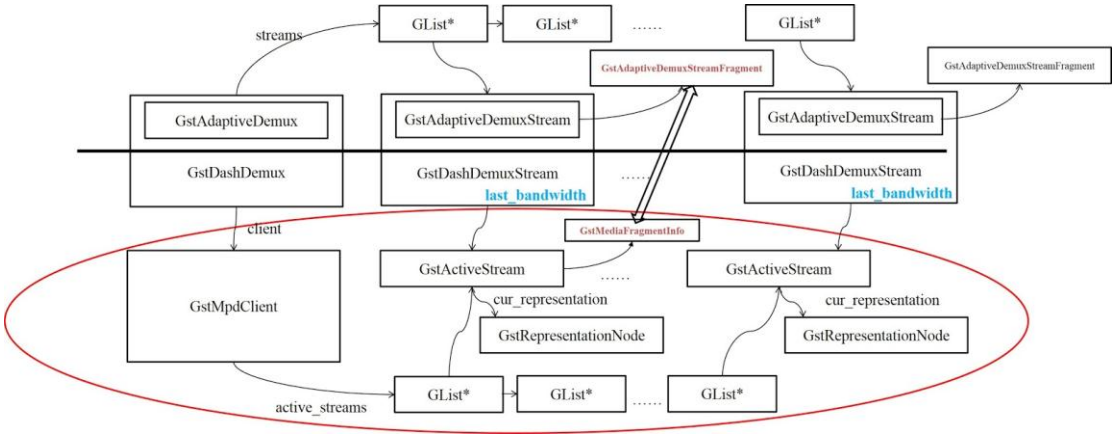

Fig 5: Relationship of GstAdaptiveDemuxStream, GstActiveStream & GstRepresentationNode.

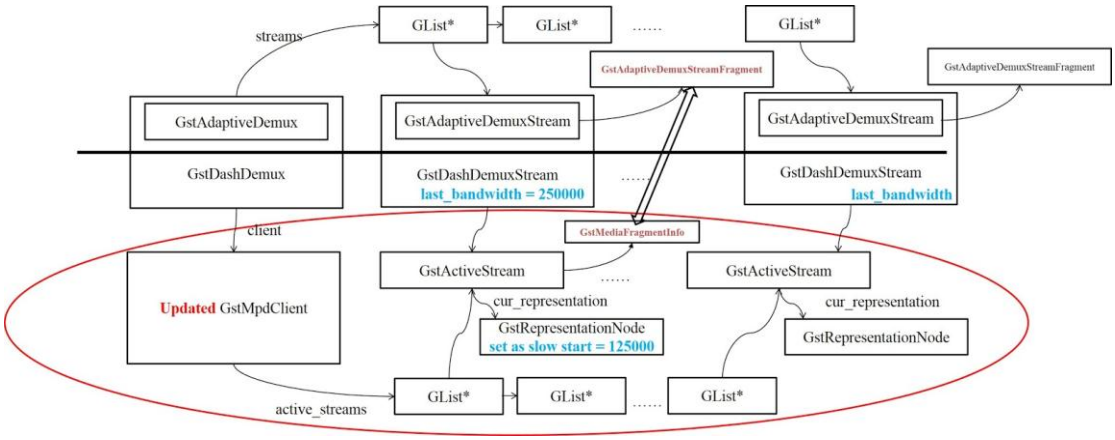

Fig 6: How could we get bit-rate of current target fragment within gst_dash_demux_stream_update_fragment_info()