

### 3 Proposed solution

The current implementation does not fully employ the sliding-window nature of a RTP jitter buffer, where packets relevant for decoding are found only within a certain time window from the head of the stream. Let's use this feature in the new solution.

If we imagine this as a window over an infinite tape with boxes, each box being one sequence number, and the tape is moving from right to left, i.e. the window is sliding over the tape from left to right, few other things can be observed too:

- Packets can be received (appear in the window) in any order, but only the leftmost packet is always removed
- This also means that packets can randomly appear in the window but cannot randomly disappear
- For equidistant packets the window moves to the right by one field every *packet-duration* milliseconds, unless **rtpjitterbuffer** introduces its own jitter
- This also means that a packet has the *window-duration* time to arrive and then can be hard considered lost (otherwise the window cannot move right and introduces an extra delay in the pipeline)
- This also means that we can look at the left-most item at the time when the window is supposed to move to the right – if there is a hole that seqno was lost, otherwise the packet is delivered
- In the ideal case without packet loss we need memory to store all packets in the configured *buffer-delay* time period.

#### 3.1 Data structure

The main data structure is not an array or linked list but a circular buffer. I.e. an array of a fixed size, with a start and end index. The packets are stored between the start and end index, with a possible wrap-around.

In contrary to the current implementation the same data structure is used to hold the actually received packets and "holes". So this data structure is used to store the information currently spread among packets, gap\_packets and timers.

Retry timer is stored in every item. On top of the circular buffer a linked list is created over holes, in which the holes are sorted by time.

An item contains:

- type: packet or hole (other like EOS can be added...)

- the packet
- timer
- other information like retry counters, etc.

## 3.2 Operations

### Insert packet

From the packet seqno the index in the circular buffer is calculated and the item is directly accessed. If there is already a packet stored at that index the new packet is a duplicate, otherwise the packet is stored at that position.

Complexity:  $O(1)$

### Append packet

Here we describe how to insert a packet behind the end-index. The operation is identical to insert, with the difference that prior to insert the end-index is advanced by the respective number of items. At this point a check is performed a) if the overall duration isn't too long, b) if the physical size of the circular buffer is not exhausted. In the latter case the circular buffer must be carefully reallocated (i.e. the sliding window resized). Some reallocs will happen in the buffering phase when the packet duration isn't yet known. However reallocs should be minimized since they are expensive.

Complexity:

- $O(1)$  in normal case
- $O(\text{realloc})$  if sliding window is resized

### Remove packet (i.e. slide window)

If the leftmost item is a packet:

1. Push out the packet
2. Mark the item as hole
3. Set the timer to now + packet\_duration
4. Increment start-index

In case this was the last packet to remove and the circular is empty the jitter buffer is reinitialized.

If the leftmost item is a hole:

1. Report the seqno as lost

2. Search to the right to find the first packet, divide the gap equally by number of items and set the timer accordingly
3. Increment start-index

Index (seqno) of the first (non-hole) packet is remembered (and updated on possible insert into the gap) as long as the leftmost item is a hole. This optimizes large gaps. Also the initial search in point 2 can be stopped after the maximum allowed gap number of steps.

Complexity:

- $O(1)$  if the leftmost item is a packet
- aggregated  $O(1)$  on a definitely lost gap
- $O(n)$  on a gap in the worst case (packets being added in the least favourable pattern)

## Timers

In case retransmits and lost packet notifications are disabled the above operations suffice and no timers are needed. However in order to manage retransmits we do need timers. The operations slightly change as follows.

On every remove (window slide to the right) a hole is appended and a timer in the hole is set. The timer is sorted into the linked list:  $O(n)$

## Initialization

The circular buffer is preallocated to certain minimum size, e.g. based on the overall latency. All items are initialized as holes.