Changes in RTP Multi-stream
draft-ietf-avtcore-rtp-multi-stream-07

Magnus Westerlund (Ericsson)
Jonathan Lennox (Vidyo)
Colin Perkins (University of Glasgow)
Qin Wu (Huawei)
Outline

› Changes
  – Scheduling algorithm change
  – Limit for transmission of initial RTCP compound packets
  – Recommendation for mitigating legacy avg_rtcp_size calculation
  – Piggybacking Feedback Packets on other SSRCs’ transmission
  – Rules for determining point to point behavior vs. multiparty
  – Intend no example configurations

› Next Step
Variables as used in RFC3550 and RFC4585
- $tp$: Time of previous RTCP transmission
- $tc$: Current Time
- $tn$: Time of next scheduled RTCP transmission
- $Td$: Deterministic transmission interval

When aggregating: set $tp$ variable to intended transmission time ($tt$) rather than $tc$
- Intended transmission time is derived by calculating $tn$ and doing consideration and updating $tn$ until allowed to send.
- To ensure maintaining bandwidth allocation
Simulations of RTP session where the number of SSRCs per endpoint is reduced uncovered an issue:

- The \( tp \) value can drift into the future
  - Example: \( \sim 2\% \) of the \( tp \) values are more than \( 1.5 \times Td \) from \( tc \)
- Reverse Reconsideration was applied
- RTCP sender may go dormant for many reporting intervals

Previous Algorithm

PDF

\[ \frac{tp-tc}{Td} \]
Issue depends on #SSRCs

› When the number of SSRCs on an endpoint is more than what can be aggregated in one RTCP compound packet:
  - Then an SSRC with $tn$ further into the future is skipped in the current packet
  - But, that SSRC is likely to be sender in the next, thus updating $tp$ to $tc$
  - Thus drift unlikely

› When the number of SSRCs all fit in one aggregate:
  - Algorithm picks the SSRC(s) that gets low random number * $Td$
  - A SSRC not picked for a couple of cycles can get $tp$ further than $1.5/1.21828*Td$ and will never be picked

› Issue arises when the SSRC or SSRCs picked are removed
  - Then $tp$ for the remaining SSRCs is far into the future
Proposed change

› Set \( tp \) to the average of all aggregated SSRCs’ transmission time (\( tt \))
  - SSRC triggering transmission has \( tt = tc \)
  - Other SSRCs calculate \( tt \) = intended transmission time

› Maintains bandwidth consumption

› Ensures that \( tp \) is worst case set to \( tc + 1.5/1.21828*Td \)
  - This prevents drift
Comparison: tp distributions
Transmission interval

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AVPF</td>
<td>0.639</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Previous Algorithm</td>
<td>0.589</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Algorithm</td>
<td>0.591</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PDF

t (s)

Previous Algorithm

New Algorithm

AVPF
Changes

- Scheduling algorithm change
- Limit for transmission of initial RTCP compound packets
- Recommendation for mitigating legacy avg_rtcp_size calculation
- Piggybacking Feedback Packets on other SSRCs’ transmission
- Rules for determining point to point behavior vs. multiparty
- Intend no example configurations

Next Step
Limit initial transmission

› When an endpoint joins an “unicast” session it may use a zero delay before sending the initial compound RTCP packet.

› We propose a limit to this behaviour to a maximum of 4 RTCP compound packets
  - These RTCP compound packets can be aggregates

› Limit chosen based on the TCP Initial Window
Legacy avg_rtcp_size

- Legacy endpoint that doesn’t calculate avg_rtcp_size as in this document:
  - Will arrive on a $Td$ value that is $N$ times longer
  - $N$ is the number of reporting SSRCs in each compound packet
- Results in lower reporting rate
- Timeout modification should prevent timeout as long as non-legacy has $Td$ no larger than 1 second
- For cases where legacy endpoints are likely
  - Limit aggregation to two SSRCs per compound, or
  - Turn off aggregation
Piggyback FB packets

› When an FB packet can’t trigger early transmission of the SSRC(s) that is suitable to report
  – AVPF says schedule regular RTCP, if that is prior to T_max_fb_delay, else
  – Drop FB packet

› We propose that it can be queued to be included (piggybacked) on the first of any other SSRCs’ compound packet which may be sent within T_max_fb_delay.
  – Source of FB packet will still be suitable SSRC (Section 5.4.1)
P2P vs. Multiparty

› Provide clear rule for how to judge Point-to-point vs. Multiparty in scheduling algorithm
  – Not based on number of SSRCs

› If Reporting groups are used:
  – If only one external reporting group then P2P, else multiparty

› Else if number of endpoint external CNAMEs seen on Media sending SSRCs are:
  – Only one then P2P, else multiparty

› Will classify mixer cases as P2P
  – Ok: Mixer will insulate the other legs or multiparty domain from endpoint.
Skipping Examples

› We have for a while considered configuration examples
  – Was a TBD in Section 6.2.2
› In the interest of completing this work we intended to skip this.
Next Step

› Please review!

› Intended to request WG last call soon
  – Giving you some time to review and consider changes

› Related documents are ready for WG last call:
  – draft-ietf-avtcore-multi-media-rtp-session-07
  – draft-ietf-avtcore-rtp-multi-stream-optimisation-05
BACKUP SLIDES
Simulation Setup

› 2 Endpoint, each starts with 16 SSRCs each
› RR: 15000 bps, RS: 10000 bps, T_rr_int = 0, transport delay between endpoints 100 ms (Static: no jitter)
› Simulation loop
  1. Send 300 RTCP packets
  2. Remove one SSRC per endpoint
  3. Perform reverse reconsideration due to the local SSRC
  4. Sample the values of \(tp\)
  5. Goto 1 unless there is only one SSRC per Endpoint
› The shown plots contains all sample values over 50000 repetitions of the above
  – For the old algorithm, the furthest drift that occurred in a specific run was 75,6*Td
### Gain Matrix

<table>
<thead>
<tr>
<th>SSRCs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>24</th>
<th>31</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVPF</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>AVPF-AGG</td>
<td>0.37%</td>
<td>-9.52%</td>
<td>-8.64%</td>
<td>-8.56%</td>
<td>-7.11%</td>
<td>-4.00%</td>
<td>-1.96%</td>
<td>0.43%</td>
<td>-0.17%</td>
<td>-0.02%</td>
<td>0.33%</td>
<td>0.03%</td>
<td>-0.28%</td>
<td>-0.10%</td>
</tr>
<tr>
<td>AVPF-RG</td>
<td>17.16%</td>
<td>-21.48%</td>
<td>-40.46%</td>
<td>-52.22%</td>
<td>-59.99%</td>
<td>-73.04%</td>
<td>-81.07%</td>
<td>-85.37%</td>
<td>-85.26%</td>
<td>-85.31%</td>
<td>-84.53%</td>
<td>-84.04%</td>
<td>-83.32%</td>
<td>-81.44%</td>
</tr>
<tr>
<td>AVPF-RG-AGG</td>
<td>17.38%</td>
<td>-30.72%</td>
<td>-49.37%</td>
<td>-60.18%</td>
<td>-67.11%</td>
<td>-78.66%</td>
<td>-85.42%</td>
<td>-88.76%</td>
<td>-88.27%</td>
<td>-87.30%</td>
<td>-86.49%</td>
<td>-85.54%</td>
<td>-83.98%</td>
<td>-81.91%</td>
</tr>
</tbody>
</table>

- Reduction in average reporting interval compared to AVPF
- Report groups for endpoints with many SSRCs (>16):
  - Under utilize bandwidth
  - Reason is IIR filtering of avg_rtcp_size
  - Example: AVPF-RG with 64 SSRCs per endpoint
    - Packets with Reporting is 2.85% of total number of packets
    - Report packets are ~8 times bigger (big report blocks)