Multiple Packetization Time in SDP
Problem statement & Requirements
New solution proposal

draft-garcia-mmusic-multiple-ptimes-problem-02.txt

IETF 71, MMUSIC WG
Philadelphia, 3/2008
Marc Willekens
Miguel Garcia
Peili Xu
Problem

- SDP defines the ptime/maxptime
  - common parameter for all media formats in m-line
  - not possible to specify this in f(codec)

  m=audio 49170 RTP/AVP 0 4 8
  a=ptime:30
  a=maxptime:60
Changes in version 02

- New clarification section
- New solution section
New proposal

- don’t introduce extra complexity
  - no signaling overkill
  - don’t punish implementations that conform to the RFCs
  - IETF architecture principle:
    - “be strict when sending” & “be tolerant when receiving”
- keep ptime/maxptime on media level
- Decouple the ptime/maxptime value that the endpoint wants to use for reception (i.e. what is expressed in SDP) from the value that the endpoint uses for sending)
- New algorithm helps to determine the actual value of ptime and maxptime that the sender should use.
New sources for ptime/maxptime

- Static
  - Default values or manually defines values in the end-device.

- Dynamic
  - Defined by the network architecture.

- Indicated
  - Proposed value from the receiving side.
Algorithm - parameters

- **Codec independent** parameters
  - $p$ vector with all provided ptime values
    static, dynamic, indicated
  - $mp$ vector with all provided maxptime values

- **Codec dependent** parameters
  - $fc$ frame size codec related
  - $mc$ maxptime codec related
    $f$(codec, frame size, frame datarate, MTU)
Algorithm - method

- packetization time for media transmission
  \[ pt = f(p, mp, fc, mc) \]
  
  - Take min. value of “mp” and “mc”
  - Take max. value of “p”
  - Normalize in function of the codec frame size.
## Examples

<table>
<thead>
<tr>
<th>p</th>
<th>mp</th>
<th>fc</th>
<th>mc</th>
<th>pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>60</td>
<td>30</td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>30</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>30</td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td>60</td>
<td>80</td>
<td>30</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>120</td>
<td>200</td>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>[40,50,20]</td>
<td>200</td>
<td>10</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>[40,50,20]</td>
<td>[40,50,20]</td>
<td>10</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>[120,40]</td>
<td>[150,200,100]</td>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Advantages

- Basic idea of ptime is kept.
  - No new SDP parameters
  - No new interpretations
- Strict in sending, tolerant in receiving
  - Sends with maximum allowed ptime <= minimal maxptime
- Different sources for ptime/maxptime
  - static, dynamic, indicated
- Local policy in end-device
- Same algorithm for sending/receiving
- Small and straight forward algorithm
  - distinction between coded dependent/independent parameters
Next steps

- Document his methodology in a separate draft/RFC?
- Include this methodology in a newer version of the SDP when/if that happens?
- Throw this draft on the garbage bin and forget about it?