Issues raised and Proposed Solutions for TMMBR

draft-ietf-avt-avpf-ccm-04

Magnus Westerlund
Bo Burman
Stephan Wenger
Umesh Chandra

ERICSSON
TAKING YOU FORWARD

Issues Raised in WG Last Call

- Usage of PT field in VBCM and not in the other (Roni)
  - Resolved by discussion
- How is the IDR behavior specified for scalable codecs? (Franz Kalleitner)
  - Payload specific, needs to be clear in the different formats
- Who is the owner of TMMBR rates initially? (Randal Jesup)
  - There is no owner intially. This should be clearer in the new definition, as it allows to indicate that there are no limitations
- Can SSRC timeout lead to multiple owners?
  - No, as limitations that aren't part of the current set of applicable ones are not remembered. Thus timeout only leads to the limitation disappearing
- A number of editorial improvements

Issues raised in WG Last Call

- It would be an advantage to be able to signal an maximum absolute packet-rate in TSTR (Randal Jesup)
  - This is more a session level constraint
  - May break design guidelines for AVPF messages

Session Maximum Bit-rate

- This issue was raised by Randal Jessup
- In a number of places the TMMBR text talks about setting the bit-rate value to the session maximum bit-rate. Session maximum bit-rate is not well defined!
- It is not necessary to have this value be a session global value. It is sufficient that any TMMBR limitation owner raise the limitation to it's local maximum to make the mechanism work and remove the limitation
- SDP b=AS (for the receiver) or a b=TIAS value from media sender can be used for this
- Needs to be defined in next version of document

TMMBR bit-rate value

- This issue was raised in a discussion between Randal Jesup, Stephan Wenger and Roni Even
- It was unclear what bit-rate value the TMMBR message contained. Several interpretations was floated:
  - Media encoding bit-rate
  - Media payload bit-rate
  - Media stream bit-rate including IP overhead
- Not even the authors were agreeing on interpretation
- There was also a discussion on how it relates to the roles of sender vs. receiver

How to measure bit-rate values

- A single bit-rate value does not consider bursty transmission
- At the same time packet networks are bursty by design at least on some timescales
- A token bucket is needed to charaterize the burstiness
- Proposal is to keep it simpler using a single value to avoid the need to handle the burstiness
- A reception rate should be calculated using averaging/smoothing over at least a timescale of a second
- A sender should consider it's behavior and avoid bursty transmissions that it can't expect the network to smooth

TMMBR Bit-rate value

- Another issue raised is the varying overhead seen by different participants depending on transport stack
- With mixers or translators in the picture it is actually likely that media receivers have different per packet overhead, like IPv4/UDP, IPv6/UDP, header compression in bottleneck links, IPv4/UDP/ESP/UDP or other tunneling mechanisms, compared to what the media sender uses
- This issue is discussed in RFC 3890 that defines b=TIAS for SDP
- A long discussion on the mailing list did produce some ideas and proposals

Using a tuple for TMMBR

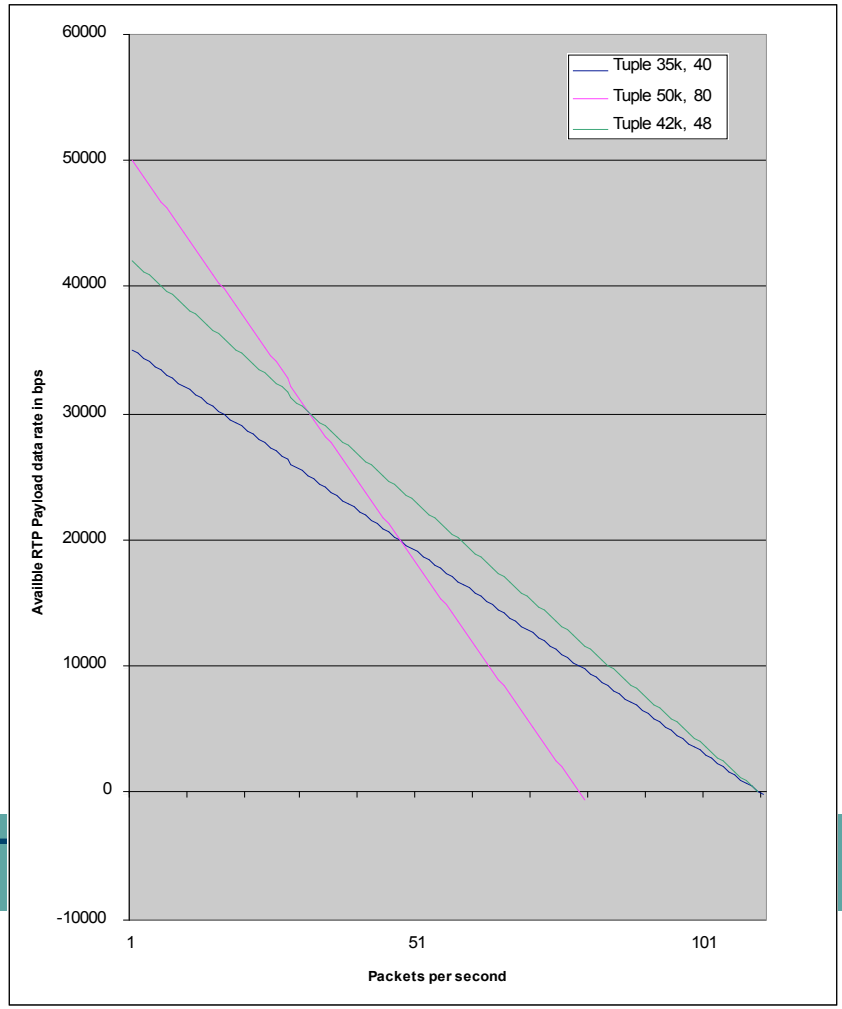
- Guido Franceschini had a good idea that allows us to avoid specifying what level the bit-rate is measured on. Instead we can use the best possible reporting depending on what information is available for the receiver
- Bit-rate combined with the per packet header overhead is used together to define a receiver's limitation
  - Bit-rate is measured in average bits per second
  - The layer the bit-rate is provided for does not matter
  - Header overhead is measured in bytes from the layer the bit-rate value is provided for to the start of the RTP-payload
  - Overhead is measured on actual traffic and initialized to expected values, e.g. IPv4/UDP/RTP = 40 bytes
  - Overhead is IIR filtered using the same filter as for the avg_rtcp_size, i.e. avg_ohd = 15/16*avg_ohd + 1/16*X
- This solution maximizes flexibility and possibility to send accurate data taking all participants into account

Tuple concept Example

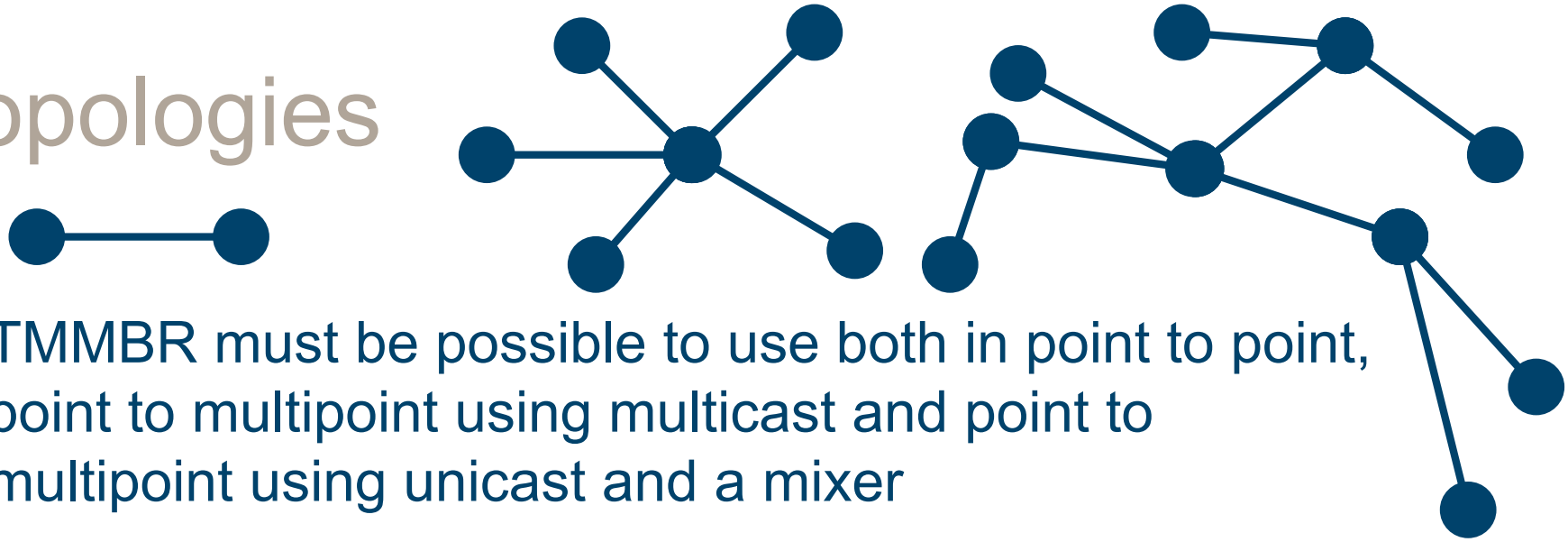
L2	IPv4	UDP	RTP	RTP Payload
8	20	8	12	

- Receiver A have a maximum IP level bit-rate of 35kbps and overhead of 40 bytes
- Receiver B have a maximum link layer bit-rate of 42 kbps and a overhead of 48 bytes
- Receiver C uses IPv6 tunneled in IPv4 and have a maximum IP level bit-rate of 50 kbps and overhead of 80 bytes

IPv4	IPv6	UDP	RTP	RT
20	40	8	12	



Topologies



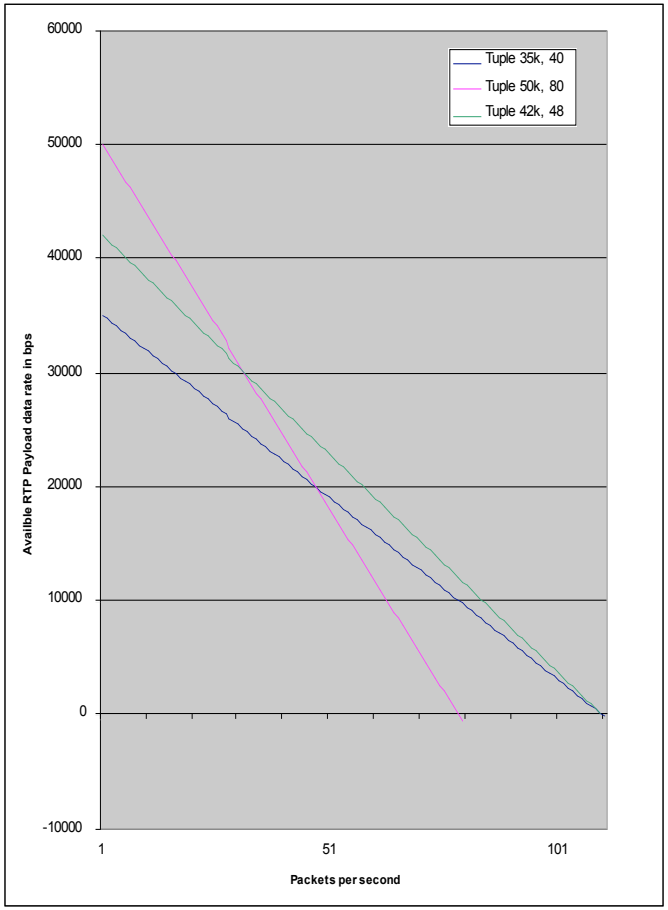
- TMMBR must be possible to use both in point to point, point to multipoint using multicast and point to multipoint using unicast and a mixer
- This requires us to consider:
  - The asynchronous interaction of different media recivers
  - The unreliability of RTCP messages
  - Handling crashed or receivers leaving the session
  - Aggregation and handling of request in mixer and translators
  - Scalability
  - Possibility to suppress requests to avoid request implosions

Impact on owner handling

- As seen by the example on previous slide, using a tuple allows for multiple limitations to have impact on the media sender depending on what packet rate it selects
- The previous mechanism only had a single owner of the limitation put on the sender
- Now we may have multiple oweners of different limiations
- Needs to be able to compare a tuple with a set of existing limitations to
  - Suppress sending of requests with limitation tuples that are not applicable
  - Allow the media sender to select which of a number of requests and an existing set of limitations that are the applicable ones

Selection algorithm

1. Find the tuple with lowest bit-rate. Use the one with highest overhead, if there are more than one. Starting point is 0 pps
2. Calculate the highest packet rate that is possible (Bit-rate/Overhead) and compare with session maximum packet rate
3. Calculate the packet rate with which this tuple intersects with all the others and Find the lowest value that are higher than starting point. (Note: some do not intersect at all)
4. If that value is lower than the session maximum rate, save tuple to the set of limitations, set starting point to found value intersection, select the tuple and return to 3
5. Done

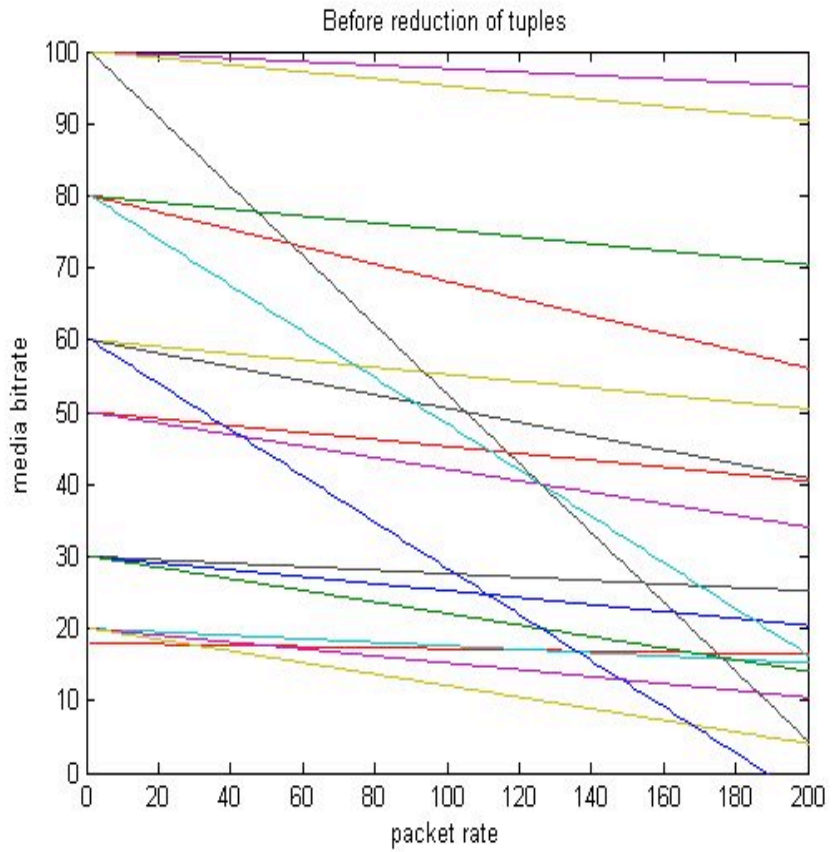


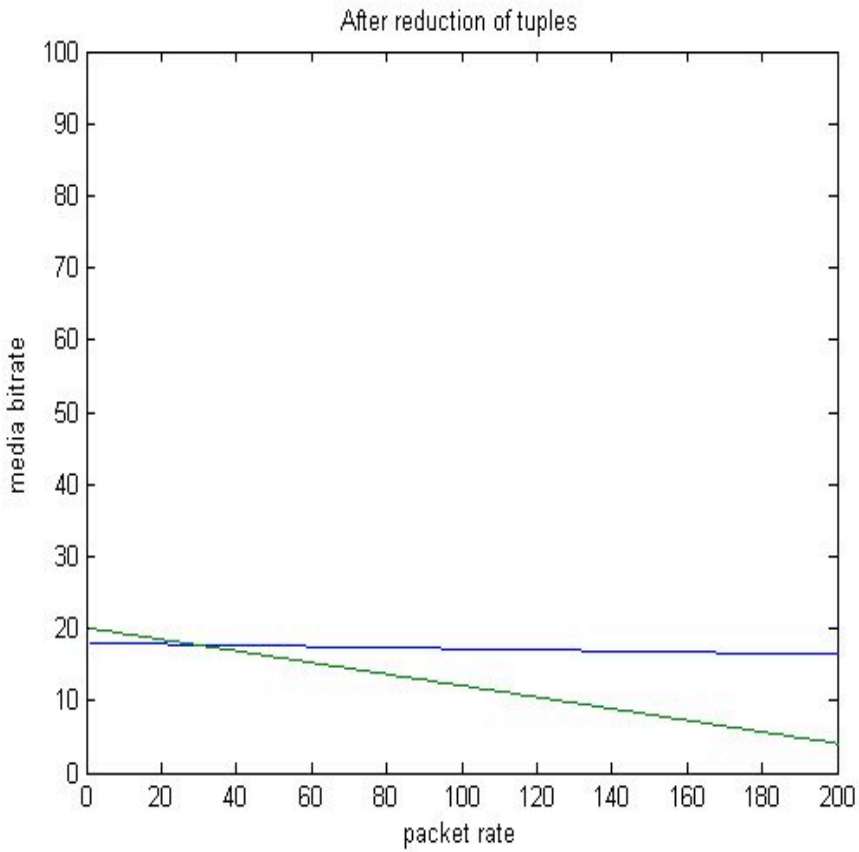
Observations around TMMBN

- TMMBN contains the set of limitation tuples produced by the media sender's usage of the algorithm. The number of tuples will vary:
  - The maximum is equal to the number of different overhead values used in the session
- A maximum session packet rate (based on application behavior) can help reduce the number of limitations that are applicable. We have proposed to add SDP signalling for this value.
- The owners (indicated by their SSRC) of the tuples provided in TMMBN can:
  - Raise or lower the values. Potentially leave the set when raising the value
  - Timeout and be removed by the media sender as it sends a TMMBN without that tuple
- To minimize state keeping, the media sender only remembers the current set of limiations and any received request from the first and until the TMMBN is sent
- When raising a value, another participant may need establish a limit to meet it's need

Larger example

- Maximum session packet rate = 100 pps used





Open issues

- We have a proposed mechanism that we think make limitations based on bit-rate and overhead work.
  - If you think we are wrong, say so **now**!
- We need to update the draft to take care of the session maximum bit-rate definition and usage
- Do another language pass on it
- Hope to be able to do a new WG last call on version 5 when it becomes available.
- If you have comments on the current version, please send them ASAP