Pseudo-CR Convolutional FEC for MCVideo

Cédric Thienot, Christophe Burdinat, Tuan Tran, Vincent Roca, Belkacem Teibi

Source:	**Expway**
Title:	**Pseudo-CR Convolutional FEC for MCVideo**
Spec:	**3GPP TR 26.881**
Agenda item:	**9.10**
Document for:	**Approval**
Contact:	**Christophe Burdinat (christophe.burdinat@expway.com)**

1. Introduction

In SA4#93, SA4 has initiated the FS_FEC_MCS study item about the applicability of FEC schemes to mission critical services, in particular for MCVideo. MBMS bearer modeling has been accepted in SA#94, and evaluation procedure discussed and pre-agreed in AHI #86 and #88.

In SA4#95, we presented the pCR S4-170912, which introduced a solution for MCVideo, based on a convolutional FEC. This solution was just noted, to give more time to delegates to consider this solution. Additionnally, to help the decision was required to include raptor10 within the performance evaluation, which is done in this revision.

This solution has been designed and evaluated in close collaboration with INRIA, in the LEELCO laboratory.

2. Reason for Change

This pCR propose a solution to the key issue 8.1 (Forward Error Correction for MCVideo). The solution relies on the usage of a convolutional AL-FEC, designed for low-latency conditions.

3. Conclusions

<Conclusion part (optional)>

4. Proposal

It is proposed to agree the following changes.

* * * First Change * * * *

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.146: "Multimedia Broadcast/Multicast Service (MBMS); Stage 1".

[3] 3GPP TS 22.179: "Mission Critical Push to Talk (MCPTT) over LTE; Stage 1".

[4] 3GPP TS 22.281: "Mission Critical Video services over LTE".

[5] 3GPP TS 22.282: "Mission Critical Data over LTE".

[6] 3GPP TS 23.203: "Policy and charging control architecture".

[7] 3GPP TS 23.280: "Common functional architecture to support mission critical services; Stage 2".

[8] 3GPP TS 23.281: "Functional architecture and information flows to support Mission Critical Video (MCVideo); Stage 2".

[9] 3GPP TS 23.282: "Functional architecture and information flows to support Mission Critical Data (MCData); Stage 2".

[10] 3GPP TS 23.379: "Functional architecture and information flows to support Mission Critical Push To Talk (MCPTT); Stage 2".

[11] 3GPP TS 23.468: "Group Communication System Enablers for LTE (GCSE_LTE); Stage 2".

[12] 3GPP TR 23.780: "Study on Multimedia Broadcast and Multicast Service (MBMS) usage for mission critical communication services".

[13] 3GPP TS 24.380: "Mission Critical Push To Talk (MCPTT) media plane control; Protocol specification".

[14] 3GPP TS 26.179: "Mission Critical Push To Talk (MCPTT); Codecs and media handling".

[15] 3GPP TS 26.281: "Mission Critical Video (MCVideo); Codecs and media handling".

[16] 3GPP TS 26.346: "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs".

[17] 3GPP TR 26.947: "Multimedia Broadcast/Multicast Service (MBMS); Selection and characterisation of application layer Forward Error Correction (FEC)".

[18] 3GPP TR 26.989: "Mission Critical Push To Talk (MCPTT); Media, codecs and Multimedia Broadcast/Multicast Service (MBMS) enhancements for MCPTT over LTE".

[19] 3GPP TR 36.868: "Evolved Universal Terrestrial Radio Access (E-UTRA); Study on group communication for E-UTRA".

[20] 3GPP TR 36.890: "Study on Support of single-cell point-to-multipoint transmission in LTE".

[21] IETF RFC6363, "Forward Error Correction (FEC) Framework" M. Watson, A. Begen and V. Roca, October 2011.

[22] ITU-RM.2377-0 Radiocommunication objectives and requirements for Public Protection and Disaster Relief (PPDR)

[23] R1-120831: "Reply LS on LS on MBMS FEC Evaluation Framework ".

[x1] IETF Internet Draft "Forward Error Correction (FEC) Framework Extension to Sliding Window Codes" V. Roca and A. Begen.

[x2] IETF Internet Draft "The Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Scheme for FECFRAME".

[x3] IETF RFC6681, "Raptor FEC Schemes for FECFRAME" M. Watson, T. Stockhammer and M. Luby, August 2012.

[x4] "Mobile Data Broadcasting over MBMS Tradeoffs in Forward Error Correction" M. Luby, M. Watson, T. Gasiba, and T. Stockhammer, Proceedings of the 4th international conference on Mobile and ubiquitous multimedia, 2006.* * * Next Change * * * *

9.x Solution #x: Convolutional FEC for MCVideo

9.x.1 Solution description

9.x.1.1 Introduction

This solution proposes to use an extension to FEC Frame (IETF RFC 6363 [21]), namely FEC Frame Ext [x1] associated to the RLC FEC scheme [x2]. IETF Internet Drafts [x1] and [x2] are Working Group Item documents of the IETF TSVWG group.

FECFrame (IETF RFC 6363 [21]) allows applying FEC to arbitrary packet flows over unreliable transport and is primarily intended for real-time, or streaming, media. FECFrame as per RFC 6363 is restricted to block FEC codes. The backward compatible extension FEC Frame Ext [x1] allows the use of sliding window codes (i.e. convolutional codes).

When applying a block FEC code, the block size is a balance between robustness (in particular in front of long loss bursts for which there is an incentive to increase the block size), and the FEC latency budget. The FEC repair symbols are generated by the encoder only when the source block is complete (enough packets have been reserved from the source) and the decoder have to wait the end reception of the source and repair symbols of a given block before decoding and forwarding them to the packet flow consumer (typically the media player). Oppositely, a sliding window FEC code allows generating on the fly repair packets, and the decoder continuously decodes. Sliding window FEC code are specifically specified to respond to the low-latency constraint.

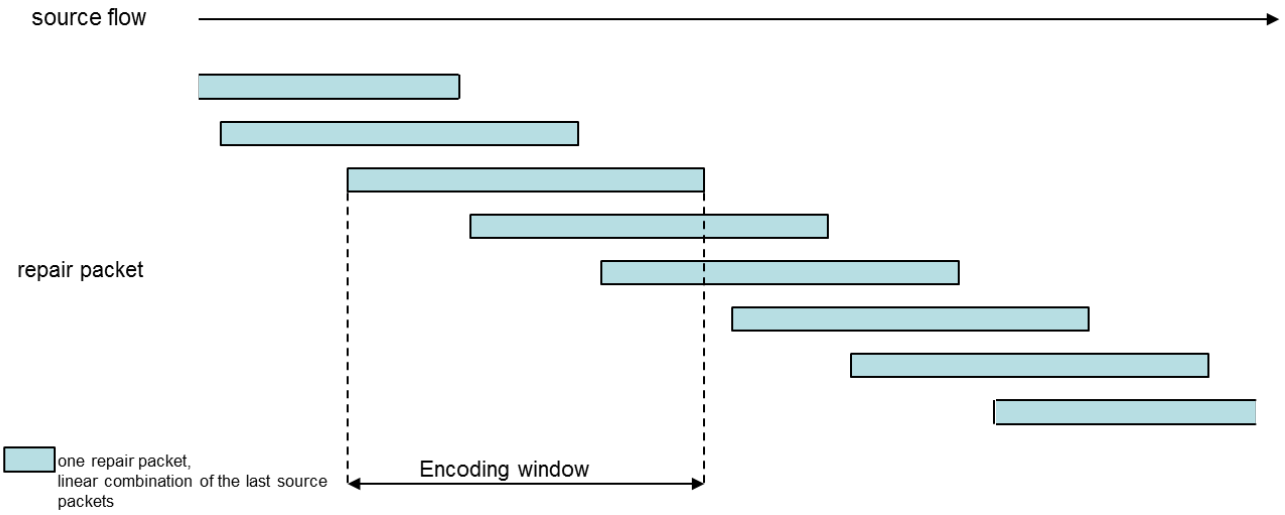


Figure 9.x.1.1-1 Sliding window FEC principle

In the following clauses are detailed FECFrame Ext and the RLC FEC Scheme. Then this solution is evaluated and its performance is compared to block FEC schemes.

9.x.1.2 FECFrame extension for convolutional FEC

As an extension of FECFrame, the following statements are still valid for its extension (FECFrame Ext):

- FECFrame Ext is described in terms of an additional layer between the transport layer (e.g., UDP) and protocols running over this transport layer.
- FECFrame Ext is applicable to protect a set of UDP source streams, identified by their destination IP and port (and also possibly source IP and port).

- With FECFrame Ext, the source packets are transport unaltered, with the exception of a possible additional trailer or footer (containing the Explicit Source FEC Payload ID).

- FECFrame Ext makes use of a FEC scheme, which defines the FEC encoding and decoding, the protocol fields and procedures used to identify packet payload data in the context of the FEC scheme.

The fundamental difference between FECFrame and its extension consists in the extension ability to transmit immediately any new ADU (Application Data Unit, e.g. source packet) to a convolutional FEC Scheme and to ask for on-the-fly generation of new repair symbols, as shown below.

The figure 9.x.1.2 illustrates the FECFrame Ext encoder operation with a convolutional FEC Code and reproduces description of the steps from clause 4.2. in [x1]:

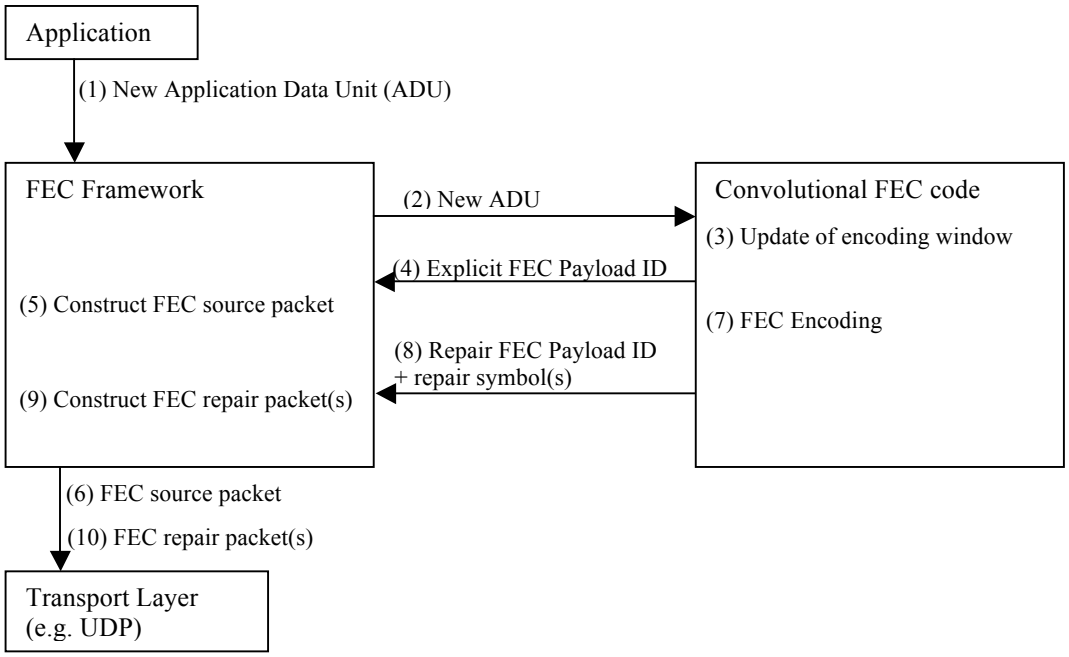


Figure 9.x.1.2 FECFrame Ext encoder operation with a convolutional code, from the figure 2 in [x1]

1. A new Application Data Unit (ADU) is provided by the application. In this study context, the ADU is a UDP packet.

2. The FEC Framework communicates immediately this ADU to the FEC scheme (With a block FEC scheme, the FEC Framework would have to wait for the building of a FEC Source Block).

3. The sliding encoding window is updated by the FEC scheme. The ADU to source symbols mapping as well as the encoding window management details are both the responsibility of the FEC scheme.

4. The Source FEC Payload ID information of the source packet is determined by the FEC scheme. If required by the FEC scheme, the Source FEC Payload ID is encoded into the Explicit Source FEC Payload ID field and returned to the FEC Framework.

5. The FEC Framework constructs the FEC source packet according to [RFC6363] Figure 6, using the Explicit Source FEC Payload ID provided by the FEC scheme if applicable.

6. The FEC source packet is sent using normal transport-layer procedures. This packet is sent using the same ADU flow identification information as would have been used for the original source packet if the FEC Framework were not present (for example, in the UDP case, the UDP source and destination addresses and ports on the IP datagram carrying the source packet will be the same whether or not the FEC Framework is applied).

7. When the FEC Framework needs to send one or several FEC repair packets (e.g., according to the target Code Rate), it asks the FEC scheme to create one or several repair packet payloads from the current sliding encoding window along with their Repair FEC Payload ID.

8. The Repair FEC Payload IDs and repair packet payloads are provided back by the FEC scheme to the FEC Framework.

9. The FEC Framework constructs FEC repair packets according to [RFC6363] Figure 7, using the FEC Payload IDs and repair packet payloads provided by the FEC scheme.

10. The FEC repair packets are sent using normal transport-layer procedures. The port(s) and multicast group(s) to be used for FEC repair packets are defined in the FEC Framework Configuration Information.

9.x.1.3 The Sliding Window RLC FEC Scheme

[x2] introduces a fully-specified FEC scheme for FECFrame Ext: the Sliding Window RLC (Random Linear Code) FEC Scheme.

[x2] provides, particular:

- specific related procedures: RLC parameters derivation, source symbols mapping, pseudo-random number generator, and coding coefficients generation function;
- the format Source FEC Payload ID and Repair FEC Payload ID formats.
- the FEC Framework Configuration Information (FFCI) carrying signalling information for the session;
- the code specification.

This code is defined over GF(2^m) (Gallois Field), where m equals 1, 4 or 8. Repair packets are generated and send on-the-fly after computing a linear combination of the source symbols present is the current encoding window.

Main principles of the Sliding Window RLC FEC Scheme are described in [x2, subclause 1.2]:

At the receiver, a linear system is managed from the set of received source and repair packets. new variables (representing source symbols) and equations (representing the linear combination of each repair symbol received) are added upon receiving new packets. Variables are removed when they are too old with respect to their validity period (real-time constraints), as well as the associated equations they are involved in. Erased source symbols are then recovered thanks this linear system whenever its rank permits it.

The Sliding Window RLC FEC scheme is designed so as to reduce the transmission overhead. The main requirement is that each repair packet header must enable a receiver to reconstruct the list of source symbols and the associated random coefficients used during the encoding process. In order to minimize packet overhead, the set of symbols in the encoding window as well as the set of coefficients over GF(2^m) used in the linear combination are not individually listed in the repair packet header. Instead, each FEC repair packet header contains:

- *the Encoding Symbol Identifier (ESI) of the first source symbol in the encoding window as well as the number of symbols. These two pieces of information enable each receiver to easily reconstruct the set of source symbols considered during encoding*

- *the seed used by a coding coefficients generation function. This information enables each receiver to generate the same set of coding coefficients over GF(2^m) as the sender*

Each FEC repair packet features a header, called Repair FEC Payload ID. Similarly, each FEC source packet features a trailer, called Explicit Source FEC Payload ID, that contains the ESI of the first source.

9.x.1.4 Parameters

The RLC FEC scheme relies on several internal parameters:

- Maximum FEC-related latency budget (**max_lat**): This can be regarded as the latency budget permitted for all FEC-related operations.

- Encoding window size (**ew_size** in symbols): used by a sender during FEC encoding. More precisely, each repair symbol is a linear combination of the ew_size source symbols present in the encoding window when RLC encoding took place.

- Linear system size (**ls_size** , in symbols): used by a receiver when managing the linear system used for decoding. ls_size is the size of the linear system, i.e., the set of received or erased source symbols that are part of the linear system.

- Decoding window size (**dw_size**, in symbols): used by a receiver when managing the linear system used for decoding. dw_size is the size of the decoding window, i.e., a subset of the linear system, corresponding to the last received or erased source symbols that are part of the linear system. Only decoded symbols from this window should be delivered to the application. Symbols which are decoded too late, out of the decoding window, are not delivered but help solving the linear system and decoding the newest.

In addition, a target **code rate** is configured in the FEC Frame Ext encoder to manage the frequency of repair symbol generation.

2 parameters are transmitted within the FEC Scheme-Specific Information:

- Encoding symbol size (**T**): a non-negative integer that indicates the size of each encoding symbol in bytes;

- m parameter (**m**): the length of the elements in the finite field, in bits, where m is equal to 1, 4 or 8

9.x.1.4 Performance evaluation

To evaluate the performance of this solution, its performance is compared to the performance of an ideal MDS (Maximum Distance Separable) FEC block code. Evaluating the performance of an ideal MDS FEC code provides an upper bound for the performance of any block FEC scheme in term of maximum supported media rate.

Its performance is also compared to raptor10 [x3], which protects non-mission critical RTP streams within the MBMS Streaming Delivery Method as specified in 3GPP TS 26.346 [16].

Raptor10 is not MDS. With a MDS FEC code, a block can always be decoded on an erasure channel if $m \geq k$ where m is the number of received symbols and k the number of source symbols per block. With raptor10, the decoding failure probability for a given block is estimated in [x4] as $0.85 * 0.567^{m-k}$. By using a high number of symbols per packet (G parameter), we can reduce arbitrarily the decoding failure probability when one more packet has been received. Consequently, the evaluation for raptor10 is done with a high G, set to 20. As the considered payload sizes in the evaluation procedure (454 and 952 bytes) are, not divisible by 20, the evaluation is done for raptor10 with 440 and 940 as payload sizes.NOTE : Raptor10 is only evaluated here in the "block – DURING" mode, as the "block BEGINNING" mode for a MDS code at 3km/h already shows bad performances as described in the next subclauses.

The simulation procedures are described in XX.1 Simulation procedure for MCVideo. In the simulation, the *ew_size* parameter is set to floor (0.75 * *dw_size*)). *ls_size* is set to 60 when the FEC latency budget is 240 or 480 ms, and 120 when the latency budget is set to 960 ms.

The list of results for RLC and ideal MDS FEC Block is provided in the submission_RLC_MDS-Block-Beginning_MDS-Block-During_Raptor10-Block-During.xls attached file.

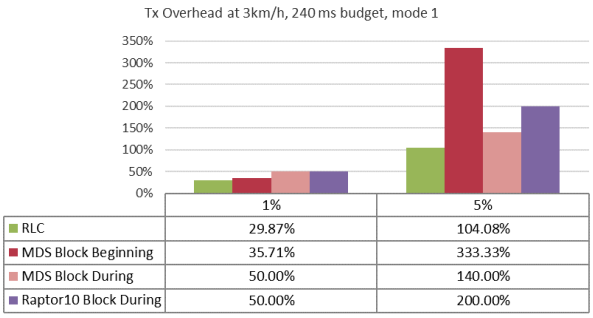
These results are presented below by the repair traffic overhead, which is linked to the code rate:

repair traffic overhead = 1/code_rate - 1

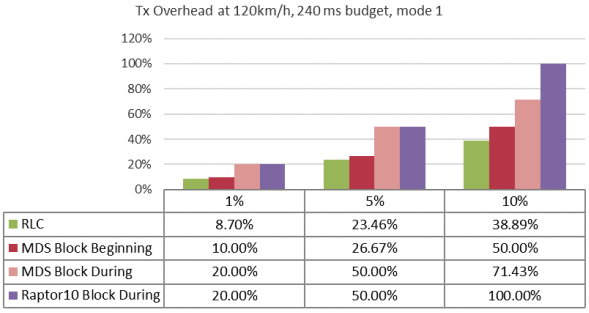
For instance, code_rate = 2/3 corresponds to a 50% repair traffic overhead (50% traffic in addition to source traffic), and code_rate = 0:5 to a 100% repair overhead (traffic is doubled).

9.x.1.4.1 Repair traffic overhead for mode 1 (bitrate 398.4 kbps, packet size: 498 bytes)

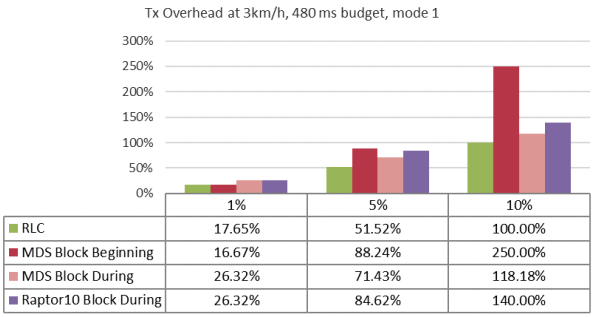
Figure 9.x.1.4.1-1 shows the required traffic overhead to obtain a residual packet loss rate below 10^{-3}, across the 3km/h and 120 km/h channels for with the 240ms, 480 ms and 960 ms latency budgets, for the mode 1.



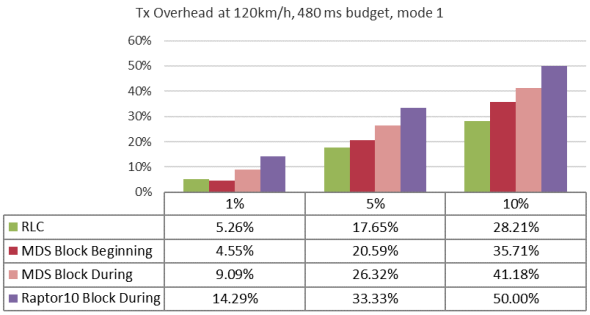
	1%	5%
RLC	29.87%	104.08%
MDS Block Beginning	35.71%	333.33%
MDS Block During	50.00%	140.00%
Raptor10 Block During	50.00%	200.00%



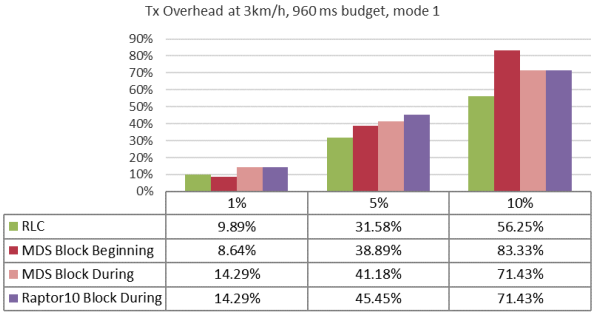
	1%	5%	10%
RLC	8.70%	23.46%	38.89%
MDS Block Beginning	10.00%	26.67%	50.00%
MDS Block During	20.00%	50.00%	71.43%
Raptor10 Block During	20.00%	50.00%	100.00%



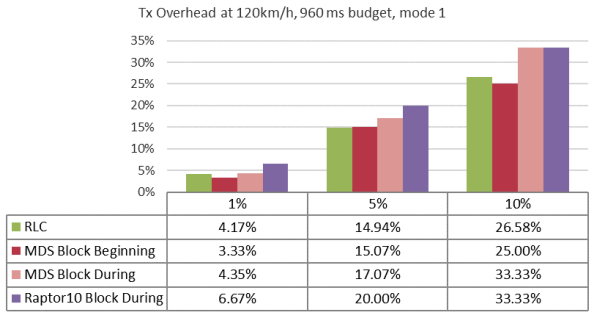
	1%	5%	10%
RLC	17.65%	51.52%	100.00%
MDS Block Beginning	16.67%	88.24%	250.00%
MDS Block During	26.32%	71.43%	118.18%
Raptor10 Block During	26.32%	84.62%	140.00%



	1%	5%	10%
RLC	5.26%	17.65%	28.21%
MDS Block Beginning	4.55%	20.59%	35.71%
MDS Block During	9.09%	26.32%	41.18%
Raptor10 Block During	14.29%	33.33%	50.00%



	1%	5%	10%
RLC	9.89%	31.58%	56.25%
MDS Block Beginning	8.64%	38.89%	83.33%
MDS Block During	14.29%	41.18%	71.43%
Raptor10 Block During	14.29%	45.45%	71.43%



	1%	5%	10%
RLC	4.17%	14.94%	26.58%
MDS Block Beginning	3.33%	15.07%	25.00%
MDS Block During	4.35%	17.07%	33.33%
Raptor10 Block During	6.67%	20.00%	33.33%

Figure 9.x.1.4.1-1 Repair traffic overhead for mode 1

Figure 9.x.1.4.1 provides the results from the less favourable conditions (FEC latency budget down to 240ms, and bursty losses at 3km/h), to the best conditions (latency budget up to 960 ms, and random losses at 120 km/h).

In the less favourable conditions (240 ms latency budget at 3 km/h), the 10^{-3} target loss rate can not be achieved for a 10% BLER. With a 5% BLER, a 104% repair traffic overhead is enough for RLC to reach the target.

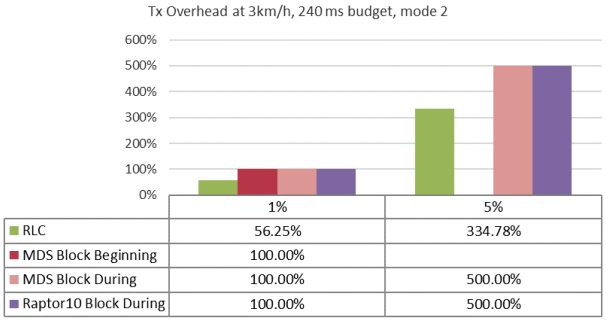
Not surprisingly, an ideal MDS FEC scheme, using the Block-Beginning transmission mode, suffers far more from the loss bursts at 3km/h, as full blocks are sent in a row, and can be more severely damaged by loss bursts than the Block-During mode, where the blocks are spread across the full latency budget. Oppositely, when the losses are random, as at 120 km/h, the Block-Beginning mode is more efficient than the Block-During mode, as the Block-Beginning mode allows bigger blocks. However, whatever the loss distribution, the RLC provides the best results.

Only at 120 km/h, when the losses are random, with a big FEC latency budget at 960 ms, the Block-Beginning mode provides equivalent perfomances to RLC.

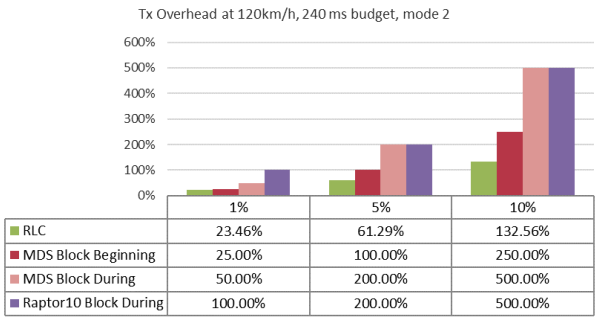
Raptor10 performances can be compared to the MDS Fec scheme in the Block-During mode. The need for raptor10 to sent more repair packets is unfavourable. Only in few cases (e.g. [120kmh, 240 ms budget], [3kmh, 960 ms budget]), performances are equivalent.

9.x.1.4.2 Repair traffic overhead for mode 2 (bitrate 398.4 kbps, packet size: 996 bytes)

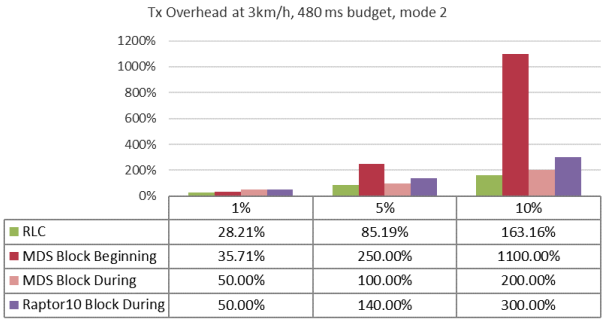
Figure 9.x.1.4.2-1 shows the required traffic overhead to obtain a residual packet loss rate below 10^{-3}, across the 3km/h and 120 km/h channels for with the 240ms, 480 ms and 960 ms latency budgets, for the mode 2.



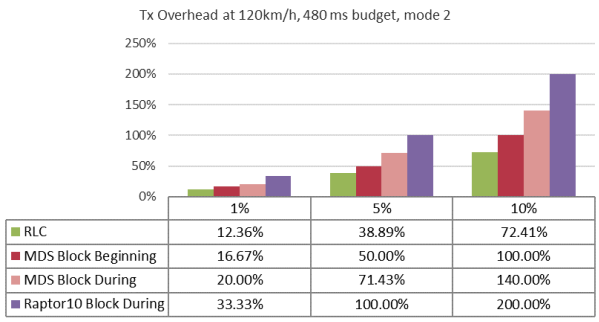
	1%	5%
RLC	56.25%	334.78%
MDS Block Beginning	100.00%	
MDS Block During	100.00%	500.00%
Raptor10 Block During	100.00%	500.00%



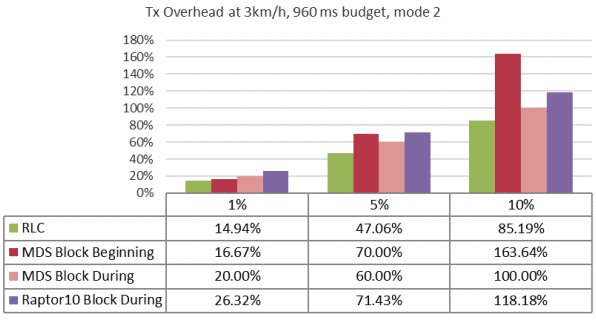
	1%	5%	10%
RLC	23.46%	61.29%	132.56%
MDS Block Beginning	25.00%	100.00%	250.00%
MDS Block During	50.00%	200.00%	500.00%
Raptor10 Block During	100.00%	200.00%	500.00%



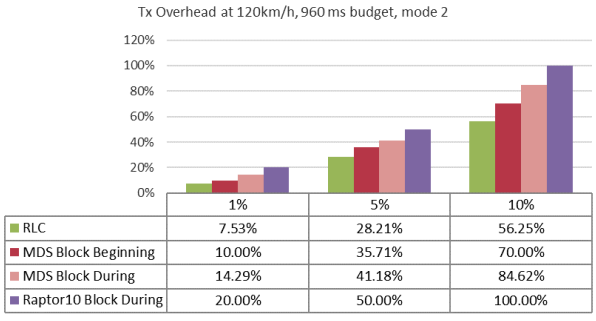
	1%	5%	10%
RLC	28.21%	85.19%	163.16%
MDS Block Beginning	35.71%	250.00%	1100.00%
MDS Block During	50.00%	100.00%	200.00%
Raptor10 Block During	50.00%	140.00%	300.00%



	1%	5%	10%
RLC	12.36%	38.89%	72.41%
MDS Block Beginning	16.67%	50.00%	100.00%
MDS Block During	20.00%	71.43%	140.00%
Raptor10 Block During	33.33%	100.00%	200.00%



	1%	5%	10%
RLC	14.94%	47.06%	85.19%
MDS Block Beginning	16.67%	70.00%	163.64%
MDS Block During	20.00%	60.00%	100.00%
Raptor10 Block During	26.32%	71.43%	118.18%



	1%	5%	10%
RLC	7.53%	28.21%	56.25%
MDS Block Beginning	10.00%	35.71%	70.00%
MDS Block During	14.29%	41.18%	84.62%
Raptor10 Block During	20.00%	50.00%	100.00%

Figure 9.x.1.4.2-1 Repair traffic overhead for mode 2

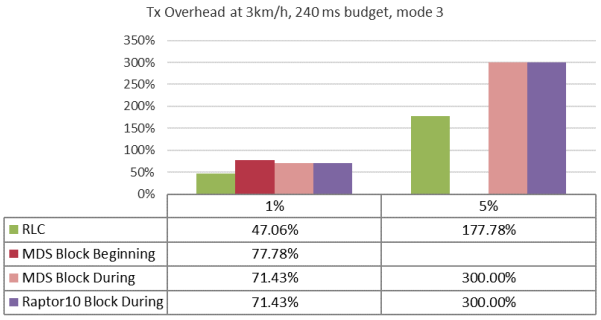
In the mode 2 (bitrate 398.4 kbps, packet size: 996 bytes), IP packets are sent over 2 MAC-PDUs, into different subframes, one packet every 20 ms. These conditions are less favourable than mode 1: packets have more chances to be lost, as each MAC-PDU can be lost, and the number of packet per FEC latency budget is decreased by half : a 240 ms duration contains only 12 packets which strongly reduces the block size for Block FEC schemes, and the decoding window size for convolutional FEC schemes. This is why, in the worst conditions (3km/h, 240ms budget), the target can not be reached at a 5% BLER (334% of overhead would be required for RLC, and such a code rate can be considered unreasonable).

With a 480 ms FEC latency budget and a BLER at 5%, RLC requires overheads of 85.19% at 3km/h and 38.89% at 120 km/h. Block-Beginning transmission mode behaves badly against the loss bursts at 3km/h (250% overhead) while 50% overhead is required at 120 km/h. Again, Block-During transmission mode is more performant at 3km/h than Block-Beginning mode, less performant at 120 km/h and both are behind RLC.

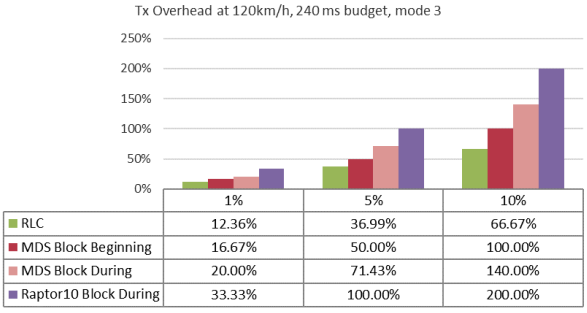
As illustrated in figure 9.x.1.4.1-2, in mode 2, RLC yields the best results in every considered cases.

9.x.1.4.3 Repair traffic overhead for mode 3 (bitrate 796.8.4 kbps, packet size: 996 bytes)

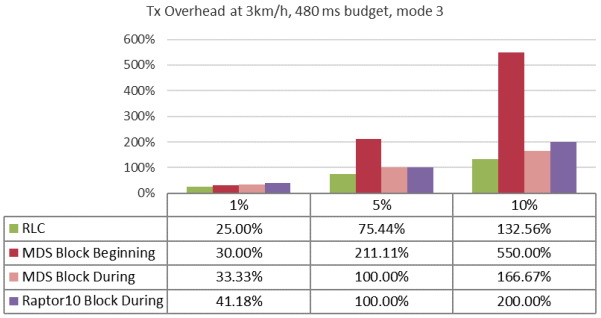
Figure 9.x.1.4.3-1 shows the required traffic overhead to obtain a residual packet loss rate below 10^{-3}, across the 3km/h and 120 km/h channels for with the 240ms, 480 ms and 960 ms latency budgets, for the mode 3.



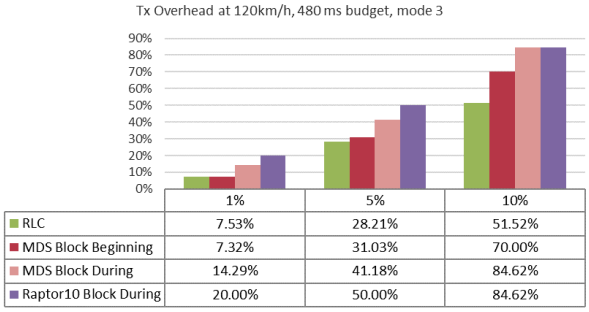
	1%	5%
RLC	47.06%	177.78%
MDS Block Beginning	77.78%	
MDS Block During	71.43%	300.00%
Raptor10 Block During	71.43%	300.00%



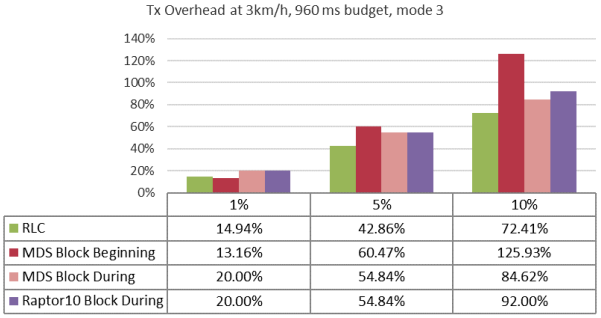
	1%	5%	10%
RLC	12.36%	36.99%	66.67%
MDS Block Beginning	16.67%	50.00%	100.00%
MDS Block During	20.00%	71.43%	140.00%
Raptor10 Block During	33.33%	100.00%	200.00%



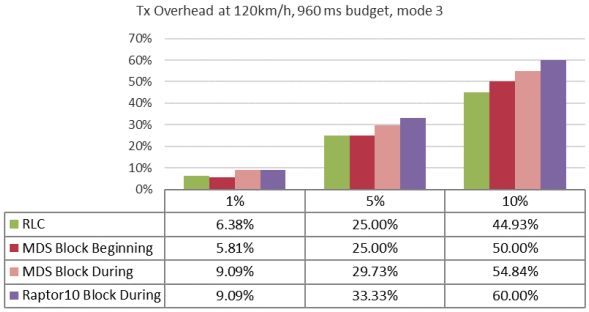
	1%	5%	10%
RLC	25.00%	75.44%	132.56%
MDS Block Beginning	30.00%	211.11%	550.00%
MDS Block During	33.33%	100.00%	166.67%
Raptor10 Block During	41.18%	100.00%	200.00%



	1%	5%	10%
RLC	7.53%	28.21%	51.52%
MDS Block Beginning	7.32%	31.03%	70.00%
MDS Block During	14.29%	41.18%	84.62%
Raptor10 Block During	20.00%	50.00%	84.62%



	1%	5%	10%
RLC	14.94%	42.86%	72.41%
MDS Block Beginning	13.16%	60.47%	125.93%
MDS Block During	20.00%	54.84%	84.62%
Raptor10 Block During	20.00%	54.84%	92.00%



	1%	5%	10%
RLC	6.38%	25.00%	44.93%
MDS Block Beginning	5.81%	25.00%	50.00%
MDS Block During	9.09%	29.73%	54.84%
Raptor10 Block During	9.09%	33.33%	60.00%

Figure 9.x.1.4.3-1 Repair traffic overhead for mode 3

In mode 3, there is one packet per frame, sent over 2 PDUs. There are any many packets per FEC Latency budget, as in mode 1, but packets are sent within 2 PDUs, as in mode 2. Overheads with mode 3 are higher than mode 1, and lower than mode 2.

Again, in mode 3, RLC yields the best results in every conditions.

9.x.2 Solution evaluation

RLC corresponds to a current effort at IETF, which identified the interest of convolution FEC schemes for protection against losses under low latency constraints.

Evaluation of the RLC scheme within a MBMS channel shows significant gains against MDS and raptor10 FEC block schemes, and provides better protection in all the conditions identified by the modelisation.

* * * Next Change * * * *

Annex B:
Simulation Conditions

B.1 Simulation Procedure for MCVideo

The MCVideo simulation procedure for block schemes:

- Select a test case from Table X3.2-1.
- Generate PDU loss transcripts (the tool described in XY.1 can be used). The transcript length has to be long enough to cover transmission of the whole stream duration.
- Compute the budget latency in packets (*lat_budget_in_pkts*) for the given FEC latency budget given in milliseconds:
    - *lat_budget_in_pkts* = Bitrate * Latency Budget / packet size
- *R* = 0, the number of repair packets per block
- Loop 1: while number of packets in error E is more than target error maxE do:
    - Compute the number of packets per block (*Np*) for the given latency budget and the given number of repair packets:
        - For Block-BEGINNING mode: *Np = lat_budget_in_pkts – R*
        - For Block-DURING mode: *Np = lat_budget_in_pkts* /2
    - *N = Np * G* where *N* is the number of symbols per block and *G* the number of symbols per packet
    - *Kp = Np – R*, where *Kp* = number of source packets per block
    - *K = Kp * G* where K is the number of source symbols per block
    - For all packets in stream do:
        - Send next SDU (NOTE 1)
            - If SDU is received according to loss transcript A, record corresponding ESI as received (NOTE 2)
        - When all symbols of one block have been sent, try decoding the block with the set of received ESI for this block
        - If not successful, *E = E* + num source symbols not received
    - If *E > maxE*, *R = R* + 1, restart Loop 1

- Record last value of *K* as *maxK*
- Report maximum streaming rate as *G*K*T**8*2 / latency budget for Block-DURING mode and (*G*K*T* *8) / (latency budget * (1 – *R*/ *lat_budget_in_pkts*)) for Block-BEGINNING mode, where *T* is the symbol size.

NOTE 1: according the mode (Block-BEGINNING, Block-DURING) one or two FIFO queues must be managed.

NOTE 2: When SDUs are sent over two PDUs, a SDU is lost if one of the 2 following PDUs in the loss transcript is lost.

The MCVideo simulation procedure for convolutional schemes:

This procedure requires a FEC decoder, managing its linear system, and able to continuously decode the received streams and count the number of successfully decoded source packets.

- Select a test case from Table X3.2-1.
- Generate PDU loss transcripts (the tool described in XY.1 can be used). The transcript length has to be long enough to cover transmission of the whole stream duration.
- Compute the budget latency in packets (*lat_budget_in_pkts*) for the given FEC latency budget given in milliseconds:
    - *lat_budget_in_pkts* = Bitrate * Latency Budget / packet size
- *code_rate* = 1, the FEC code rate
- Loop 1: While number of packets in error *E* is more than target error *maxE* do:
    - *dw_size* = floor (*lat_budget_in_pkts * code_rate)*
    - derive *ew_size* from *dw_size* (the computation depends on the FEC internal configuration and usage, for instance *ew_size* = floor (0.75 * *dw_size*))
    - derive *ls_size* (the computation depends on the FEC internal configuration and usage, for instance, for instance *ls_size* = 2 * *dw_size*)

- *Ns* = 0, where *Ns* is the number of sent source packets
- *Nr* = 0, where *Nr* is the number of sent repair packets
- For all packets in stream do:
  - while *Ns* /(*Ns* + *Nr* + 1) < *code_rate*
    - generate and send new repair packet (NOTE 3); *Nr* ++;
  - Send next source packet (NOTE 3) ; *Ns* ++;
- Get *E* from the decoder: number of missing/undecoded source packets

- If *E* > *maxE*,
  code_rate = *code_rate* – 1%, restart Loop 1

- Record last value of *code_rate*
- Report maximum streaming rate as *code_rate* * *lat_budget_in_pkts* * *T* * 8, where *T* is the symbol size.

NOTE 3: sent packets are communicated to the decoder if the following PDU(s) in the loss transcript are marked as lost.

* * * End of Changes * * * *