Design and evaluation of a Low Density Generator Matrix (LDGM) large block FEC codec

Vincent Roca, Zainab Khallouf, and Julien Labouré
{firstname.name}@inrialpes.fr

Projet Planète; INRIA Rhône-Alpes

NGC'03, Munich, Germany, September 2003

- Part 1:

Some background on large block FEC codes

Some background

- FEC is required by all reliable multicast protocols
  - a single FEC packet recovers many different losses at different receivers
  - FEC reduces the number of feedback control information improved scalability
- FEC is fundamental to the ALC (RFC 3450-1) reliable multicast protocol
  - there is no feedback at all
  - reliability is achieved thanks to the heavy use of FEC

Some background… (cont')

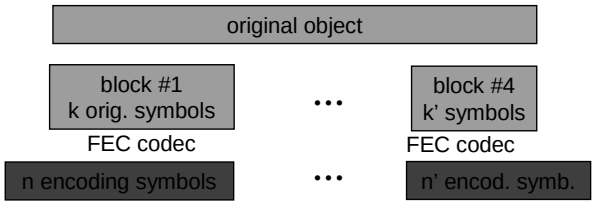
- why « large block » ?
  - **the codec operates on a block (set) of *k* packets**
  - **it produces *n-k* redundant (A.K.A. parity or FEC) packets**

large block == "k amounts to 10,000s or more packets"

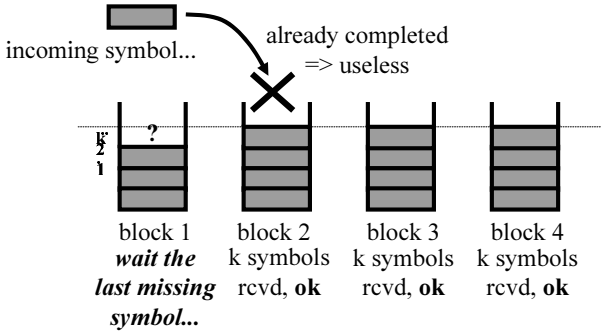
- since a parity packet can recover an erasure only in its block, the optimal solution is to have the file encoded as a ***single block…***
- …which is only possible if large blocks can be used !
- with Reed-Solomon (small block code), n ≤ 255

Some background… (cont')

- waiting for the last packet with a small block FEC code…
  - at a sender



  - at a receiver



Some background… (cont')

- a bit of history…
  - LDPC large block FEC codes discovered in 1960s by Gallager…
  - forgotten during 30 years…
  - re-discovered in 1995 (MacKay & Neal)…
  - and largely improved by Luby, Shokrollahi and al.
    - **they are the Tornado©, LT©… codes**
    - **those codes are patented**
      - see RFC 3453 for full list of patents (!)
    - **Digital Fountain, Inc. has been created to commercialize them**

…but what about LDPC which has surprisingly (!) been omitted in RFC 3453 ?

Goals of this work

- design a patent-free codec
- simplified variant of LDPC: LDGM
  - promising results…
- results can be improved with another variant: LDGM staircase

  (not in the paper)

- Part 2:

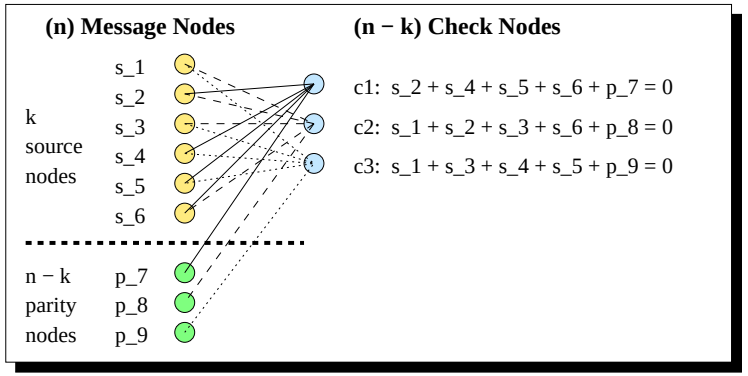
Introduction to LDGM and LDPC

What's this… (cont')

- The Internet is a Packet Erasure Channel (PEC)
  - it works on packets
  - packets can be erased (i.e. lost)
  - but a packet arriving at a receiving applications is error-free
    - **integrity is checked by physical CRC, and TCP/UDP checksum**
  - LDPC was initially designed for a Binary Symmetric Channel (BSC), where bits can be flipped
  - decoding become straightforward with a PEC
    - **normal, there's more information in a PEC**

Low Density Generator Matrix (LDGM)

- Fundamentals
  - based on XOR
  - two representations: ***bipartite graph*** and ***matrix***
  - notations:
    - **s_i are source packets, p_i are FEC packets, c_i are check (A.K.A. constraint) nodes (not sent)**



LDGM… (cont')

- dual (k x n) matrix representation:

$$[H \mid Id_3] = \begin{array}{c} \begin{matrix} s_1 & .. & s_6 & p_7 & .. & p_9 \end{matrix} \\ \left(\begin{array}{cccccc|ccc} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right) \end{array} \begin{matrix} \\ c_1 \\ c_2 \\ c_3 \end{matrix}$$

e.g. it says that for c_1: $s_2 + s_4 + s_5 + s_6 + p_7 = 0$

- Encoding
  - encoding is simple since a + a = 0 (bitwise XOR)
  - each p_i is the sum of the source symbols in the associated constraint equation

e.g. for c_1: $p_7 = s_2 + s_4 + s_5 + s_6$

LDGM… (cont')

- so [H | Id_{n-k}] is also a generator matrix, hence the "GM" in the LDGM name
- this generator matrix has few "1"s, hence the "Low Density" in the LDGM name
  - **NB: this low density is not visible in the example though, but there are a fixed number of "1"s per line and column (often less than 20), no matter how large (k, n) are (e.g. 10,000s).**
- this "low density" makes the encoding time linear *O(n-k)* hence the *very high encoding speed*

LDGM… (cont')

- Iterative decoding algorithm
  - solve a system of linear equations using a trivial algorithm:

    e.g. for c_1: $s_{2\,(missing)} + s_4 + s_5 + s_6 + p_{7\,(known)} = 0$

    then you have: $s_2 = s_4 + s_5 + s_6 + p_7$

  - ***step 1:*** so, you look for equations (set of constraints) where all the variables are known except one, and if one such equation exists you directly find the missing variable.

  - ***step 2:*** each time a packet is received or recovered, you replace its value in the equations, and go to step 1.

LDGM staircase

- Principles
  - replace the identity matrix by a "Staircase" matrix

$$[H \mid \text{Staircase}_5] = \left(\begin{array}{cccccc|ccccc} & s_1 & .. & & s_6 & & p_7 & .. & & p_{11} \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

  - encoding:
    - **calculate the first parity packet: p7 = s2 + S4 + S5**
    - **calculate the remaining parity packets, in the order:**

      p8 = p7 + …

      p9 = p8 + …, etc.
  - this code has a better erasure recovery property, because parity packets are themselves protected

Low Density Parity Check (LDPC)

- Principles
  - that's the general case
  - the random bipartite graph is extended to the whole set of source and parity packets (s_i and p_i)

    e.g. it says that for c_1: $s_2 + s_5 + s_6 + p_7 + p_9 + p_{12} = 0$

    etc.

$$[H] = \left(\begin{array}{cccccc|ccccc} & s_1 & .. & & s_6 & & p_7 & .. & & p_{11} \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{array}\right)$$

LDPC… (cont')

- this code has a better erasure recovery properties than LDGM/LDGM staircase, because parity packets are well protected

- encoding is more complex
  - **the H (sparse) parity check matrix does not say how to encode**
  - **need to solve a system of linear equations first, and produce a G generator matrix**

    time consuming task…
  - **encoding is done by multiplying G by the source packets, but G is a dense matrix**

    time consuming task… O((n-k)*k)

- Part 3:

Some Performance Results

Inefficiency Ratio

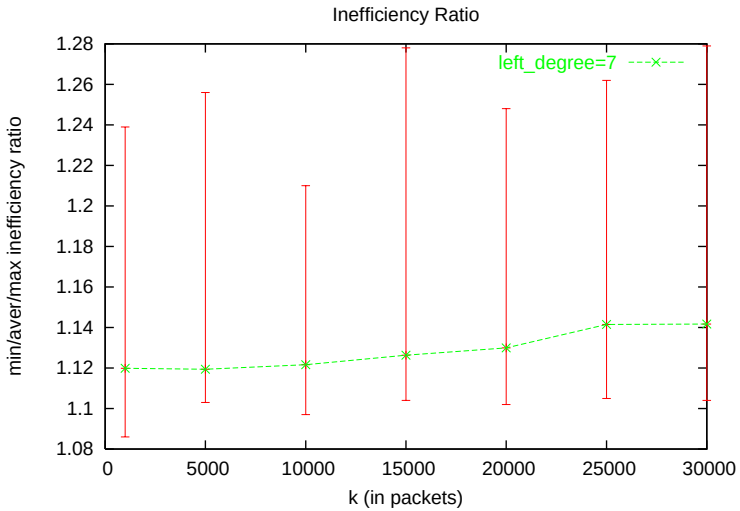
- LDGM/LDPC are not MDS codes !
  - means that more than ***k*** packets out of ***n*** are required to recover the ***k*** source packets
  - introduce an inefficiency ratio:

$$\text{ineff\_ratio} = \frac{\text{total\_nb\_packets\_received\_to\_decode}}{k} \geq 1.0$$

  - this ratio is not constant
    - **consider minimum / average / maximum ineff_ratio during tests**

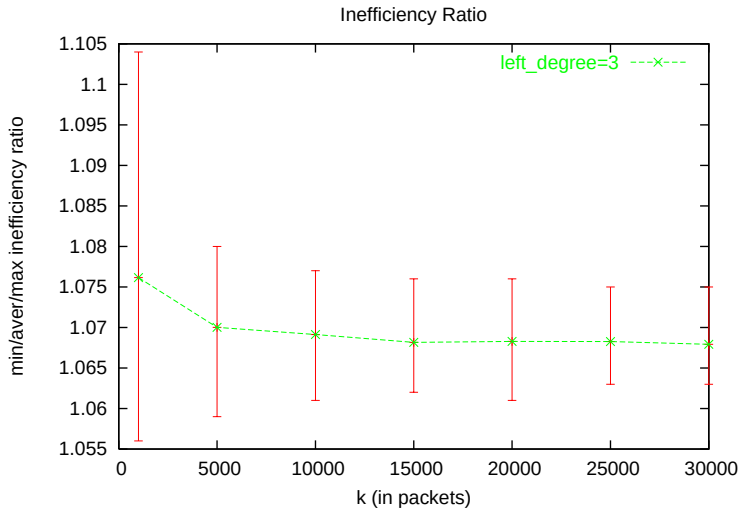
Inefficiency Ratio… (cont')

- LDGM (n-k = k/2)
  - **e.g. k = 10,000; ineff_ratio = {9.7%, 12.2%, 21.0%}**



Inefficiency Ratio… (cont')

- LDGM staircase (n-k = k/2, not in the paper)
  - **e.g. k = 10,000; ineff_ratio = {6.1%; 6.9%; 7.7%}**



Performances… (cont')

- Encoding speed
  - **LDGM: 5MB/0.171s = 239.5 Mbps**
    10MB source block, 5MB parity created, PIII/1GHz
  - **LDGM staircase: similar**
  - **LDPC: not yet available but will be several orders of magnitude slower**
- Decoding speed
  - **similar with LDGM/LDGM staircase/LDPC**
  - **only slightly slower than encoding**
  - **done progressively, each time a new packet arrives**

- Part 4:

Conclusions

Conclusions

- Points not covered here (see paper)
  - LDPC/LDGM are excellent for ***partially reliable*** sessions
  - LDPC/LDGM can be used to protect a stream of video frames
    - **our Scalable Video Streaming over ALC (SVSoA)**
  - LDGM/Reed-Solomon comparison

Conclusions… (cont')

- LDGM/LDGM staircase
  - LDGM staircase replaces favorably LDGM
  - certainly not as efficient as codes from Digital Fountain !
    - **…but nobody can use them except DF**
  - patent-free
  - very high encoding/decoding speed
  - good and stable protection
    - **average inefficiency ratio is good ~7%**
    - **and fairly stable {6%; 8%} range**
  - operates on blocks of several tens of MB

Conclusions… (cont')

- LDPC
  - protection is probably good
    - **evaluation under progress…**
  - for situations where encoding speed is not an issue
    - **appropriate for "on-demand" delivery sessions**
    - **decoding speed is high**
      - burden at the sender, not at the receiver

- you can use it freely…
  - an open-source implementation already available
    - **we used some code from Radford Neal**
  - part or an ambitious project: MCLv3
    - **ALC and NORM implementation**

Thanks for your attention !

and have good fun at

http://www.inrialpes.fr/planete/people/roca/mcl/