Tetrys: a specification of an on-the-fly network coding transport protocol

Jonathan Detchart, Jérôme Lacan, Emmanuel Lochin
ISAE-SUPAERO
Vincent Roca
INRIA
• We, the authors, didn't try to patent any of the material included in this presentation

• We, the authors, are not reasonably aware of patents on the subject that may be applied for by our employers

• If you believe some aspects of this presentation and its corresponding draft may infringe IPR you are aware of, then please fill an IPR disclosure and let us know
Objective

• Propose a transport protocol framework based on elastic encoding window

• This specification must be generic (e.g. independent from the congestion control)

• Propose some building blocks (defined into this ID or not TBD)
To date inside the draft

• Tetrys, a reliable communication protocol + some building blocks:
  • On-the-fly encoding (FEC codes)
  • Signalling (symbol identifiers, coefficients, …)
  • Elastic window management
  • Packet creation and processing
  • Dynamic adjustment of the code rate and flow control

• Optional features:
  • Congestion control management (not CC, only the entry point to manage it)
  • Feedback management
  • Packet loss rate estimation
Layering

Considering as an instance of such protocol, Tetrys is defined as an application layer transport protocol (above UDP)
Use-cases

Defined for end-to-end communications

Unicast

- communication with **feedbacks**: the coded packets are LC of non-acknowledged source symbols

- communication without feedback: the coded packets are LC of a sliding window of source symbols with a fixed size

Multicast

- same as unicast but with the union of feedbacks from all receivers
The protocol part

- Tetrys has 3 types of packet:
  - the source packets
  - the coded packets
  - the acknowledge packets (optional)
The encoding vector

\[
\begin{array}{ccccccccccccccccccccccc}
\text{EV_LEN} & \text{CCGI} & \text{I} & \text{C} & \text{V} & \text{NB_IDS} & \text{NB_COEFS} \\
\text{FIRST_SOURCE_ID} \\
\text{b_id} \\
\text{id_bit_vector} \\
\text{Padding} \\
\text{b_coef} \\
\text{coef_bit_vector} \\
\text{Padding} \\
\end{array}
\]
An example: the sliding window

• Suppose we want to generate an encoding vector by using a sliding window of size 64, by using GF(2^8) from symbol S_0 to S_{63}:

    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-------------------------------------------------------------------+
    | EV_LEN   | CCGI   | I | C|V| NB_IDS   | NB_COEFS   |                     |
    +-------------------------------------------------------------------+
    |                     FIRST_SOURCE_ID                                |
    +-------------------------------------------------------------------+

• We don’t store any coefficient (they are computed in a deterministic way), we only store the first source id included and the number

• We need 8 bytes of data for the full encoding vector
Generating the coefficients

- We propose a deterministic way based on IDs

- In such a way, you can generate each coefficient independently

- About the deterministic functions, see the draft for further details

- In such a way, you can generate each coefficient independently
Discussion with the group

• split the document between protocol part and independent building blocks?

• continue work on the generic protocol document?

• continue independently by defining some building blocks in additional ID?