NWCRG
Internet-Draft
Intended status: Experimental
Expires: April 30, 2015

J. Detchart
E. Lochin
J. Lacan
ISAE
V. Roca
INRIA

October 27, 2014

Tetrys, an On-the-Fly Network Coding protocol
draft-detchart-nwcrg-tetrys-00

Abstract

This document describes Tetrys, an On-The-Fly Network Coding (NC) protocol which can be used to transport delay and loss sensitive data over a network. This protocol can be used for unicast, multicast or anycast communications. It recovers lost packets by using added coded packets. Tetrys allows to recover all missing data within a time independent of the RTT and does not rely on a reliable feedback path to transmit data.

Table of Contents

1. Introduction 2
 1.1. Requirements Notation 3
2. Terminology 3
3. Architecture Reference 4
4. Tetrys Basic Functions 4
 4.1. Encoding 4
  4.1.1. Encoding Vector Formats 5
  4.1.2. Encoding Windowing 6
 4.2. Decoding 7
  4.2.1. Acknowledgements 7
5. Encapsulation Packet Format 8
 5.1. Source Packet Format 8
 5.2. Coded Packet Format 9
6. Security Considerations 9
7. Privacy Considerations 10
8. IANA Considerations 10
9. Acknowledgments 10
10. References 10
 10.1. Normative References 10
 10.2. Informative References 10
Authors' Addresses 10

1. Introduction

This document presents the implementation details of a novel network coding protocol called Tetrys. Network codes introduced in 2000 [AHL-00] have been developed to address the limitations of transmission over the Internet (delay, capacity and packet loss). While the use of network codes in fairly recent in the Internet community, the use of application layer erasure codes in the IETF has already been standardised in the RMT [RMT] and the FECFRAME [FECFRAME] working groups. The protocol presented here can be seen as a network coding extension to standards solutions. The current proposal can be considered as a combination of network coding and feedback mechanisms [Tetrys].

A coded Tetrys packet is a linear combination over a finite field of the data source packets belonging to the coding window. The choice of the finite field of the coefficients is a trade-off between the best performance (with non-binary coefficients) and the system

constraints (the use of binary codes in an energy constrained environment for example) and is driven by the application. The main novelty of the Tetrys protocol is to build coded packets from an coding window the size of which is periodically updated with the receiver's feedbacks. This update is done in such a way that any source packets coming from an input flow is included in the coding window as long as it is not acknowledged. This mechanism allows for losses on both the forward and return paths and for the acknowledgement messages to be lost.

1.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Terminology

The terminology used in this document is presented below. It is aligned with the FECFRAME terminology as well as with recent activities in the Network Coding Research Group.

Source symbol: a symbol that has to be transmitted between the ingress and egress of the network.

Coded symbol: a linear combination of a set of source symbols.

Input symbol: a symbol at the input of the Tetrys Encoding Building Block. In an end-to-end context, the input symbols are always source symbols.

Output symbol: a symbol generated by the Tetrys Encoding Building Block. For a non systematic mode, all output symbols are coded symbols. For a systematic mode, output symbols can be the input symbols and a number of coded symbols that are linear combinations of the input symbols + the encoding vectors.

Encoding vector: a set of the encoding coefficients and source symbol IDs.

Source packet: a source packet contains a source symbol with its associated ID and size.

Coded packet: a coded packet contains a coded symbol, the coded symbol's ID, size and encoding vector.

Output packet: an output packet is either a coded packet, either a source packet.

Feedback packet: a feedback packet is a packet containing information about the decoded/received source symbols. It can also bring additional information about the Packet Error Rate or the number of various packets in the receiver buffers

Elastic Encoding Window: an encoder-side buffer that stores all non-acknowledged source packets used as an input flow to create coded packets.

3. Architecture Reference

The architecture used in this document is aligned with the Network Coding Architecture draft [NWCRG-ARCH].

Tetrys uses two building blocks to provide a fully reliable protocol. The Tetrys Encoding Building Block creates some linear combinations of all the non-acknowledged input symbols. Each linear combination is called a coded symbol. It is associated to an encoding vector, which defines the input source symbols and the finite field coefficients used. This encoding mechanism is called an elastic encoding window. Each generated output symbols is encapsulated in an output packet format. A Tetrys Decoding Building Block stores all the received output packets. When it is possible, the coded symbols are decoded to generate the lost source symbols. Regularly, this building block sends a feedback packet containing information about the acknowledgement of received and decoded source symbols.

```
            +-----------+                  +-----------+
            |   Tetrys  | Output packets   |   Tetrys  |
input       |  Encoding |----------------->|  Decoding | source
----------->|  Building | Feedback packets |  Building |-------->
symbols     |   Block   |<-----------------|   Block   | symbols
            +-----------+                  +-----------+
```

Figure 1: Tetrys Architecture

4. Tetrys Basic Functions

4.1. Encoding

When a Tetrys Encoding Building Block needs to generate a coded symbol, it considers the set of source symbols stored in the encoding windows. These source symbols are the set of source symbols which are not yet acknowledged by the receiver.

At the generation of a coded symbol, the Tetrys Encoding Building Block builds an encoding vector containing the IDs of the source symbols stored in the encoding window. For each source symbol in the

encoding window, a finite field coefficient is determined from a deterministic function taking as input the source symbol ID and the coded symbol ID. For example, deterministics coefficients can be obtained from traditional codes such as Reed-Solomon, or other methods. Finally, the coded symbol is the sum of the source symbols multiplied by their corresponding generated coefficients.

4.1.1. Encoding Vector Formats

Depending of the signaling type, an encoding packet can contain a list of source symbol IDs.

4.1.1.1. Encoding Vector Format 1

The first encoding vector format uses a deterministic way to generate the coefficients, and a set of blocks for the source symbol IDs. In this format, we need to keep the number of blocks we used, as a 8bits unsigned integer.

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Encoding Vector Size         |  Type |   Nb blocks   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Left Edge of 1st Block                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Right Edge of 1st Block                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                              . . .                            /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Left Edge of nth Block                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Right Edge of nth Block                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Encoding Vector Format 1

Encoding Vector Size: size in bytes for the encoding vector.

Type: the type of the encoding vector. MUST be 1 in this case.

Nb blocks: the number of blocks stored in the encoding vector.

Left Edge of Block: the first source symbol ID of this block.

Right Edge of Block: the last source symbol ID of this block.

4.1.1.2. Encoding Vector Format 2

The second option to send en encoding vectors is to send the symbol source IDs and the coefficients used. The IDs are unsigned 32bits integers, and the coefficients are unsigned 8bits integer corresponding to the elements in the finite field.

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Encoding Vector Size         |  Type |   Nb packets  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      1st Source Symbol ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      2nd Source Symbol ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                             . . .                             /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      nth Source Symbol ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    coef 1     |    coef 2     |      ...      |    coef n     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Encoding Vector Format 2

Encoding Vector Size: size in bytes for the encoding vector.

Type: the type of the encoding vector. MUST be 2 in this case.

Nb packets: the number of packets stored in the encoding vector.

Source Symbol ID: unsigned 32bits integer source symbol ID you want to use.

coef: a coefficient used for the linear combination.

4.1.2. Encoding Windowing

When an input source symbol is passed by the upper layer to the Tetrys Building Block, it is added to the encoding window. In parallel, a copy of this packet can be sent onto the network. As an element of the encoding window, this symbol is included in the linear combinations created to generate coded symbols.

As explained below, the receiver sends periodic feedbacks indicating the received or decoded source symbols. In the case of a unicast

transmission, when the sender receives the information that a source symbol was received or decoded by the receiver, it removes this symbol from the encoding window. In a multicast transmission, a source symbol is removed from the encoding window only when all the receivers have received or decoded it.

4.2. Decoding

The decoder uses the encoding vectors embedded in each coded packet to obtain the linear system used by the Tetrys Encoding Building Block. If the number of innovative coded symbols is greater than the number of lost source symbols, a classical method to solve the linear system, like e.g. matrix inversion, can be then used to recover the source symbols.

4.2.1. Acknowledgements

A Tetrys Decoding Building Block can send back to a Tetrys Encoding Building Block some Acknowledge packets. They contain information about what it is received or decoded, and other information such as a packet loss rate or a size of the decoding buffers.

```
0                   1                   2                   3
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Type |                  Total Length                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Nb missing source symbols                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Nb useless coded symbols                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    First Source Symbol ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sack size   |                                               |
+-+-+-+-+-+-+-+-+                                               +
|                                                               |
/                          Sack Vector                          /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Acknowledgement Packet Format

Version: the version of the packet format. MUST be 1.

Type: the type of acknowledgement packet. MUSE be 1.

Total length: the length of the packet (in bytes).

Nb missing source symbols: the number of missing source symbols in the receiver buffer.

Nb useless coded symnbols: the number of useless coded symbols in the receiver buffer. Meaning the number of linear combinations containing at least 2 unknown source symbols.

First Source Symbol ID: the first source symbol to acknowledge.

Sack size: the size (unsigned integer 8 bits) of the sack vector as a multiple of 32 bits. Ex: if the value is 2, the sack vector is 64 bits.

Sack vector: the vector for the acknowledge symbols following the first source symbol ID.

5. Encapsulation Packet Format

5.1. Source Packet Format

A source packet is the encapsulation of a source symbol and a packet header.

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Type |                  Total Length                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Source Symbol ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                             Payload                           /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Source Packet Format

Version: the version of the packet format. MUST be 1.

Type: the packet type. MUST be 1.

Total length: size in bytes for the packet.

Source Symbol ID: the sequence number to identify a source symbol.

Payload: the payload (source symbol)

5.2. Coded Packet Format

A coded packet is the encapsulation of a coded symbol, the associated encoding vector and a packet header.

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Type |                 Total Length                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Coded Symbol ID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                          Encoding Vector                      /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Encoded Payload Size      |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
/                             Payload                           /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Coded Packet Format

Version: the version of the packet format. MUST be 1.

Type: the packet type. MUST be 2.

Total length: size in byte for the packet.

Coded Symbol ID: the sequence number to identify a coded symbol.

Encoding Vector: an encoding vector to define the linear combination used (coefficients, and source symbols).

Encoded Payload Size: the coded payload size used if source symbols are variable size.

Payload: the payload (coded symbol)

6. Security Considerations

TBD

7. Privacy Considerations

TBD

8. IANA Considerations

N/A

9. Acknowledgments

We would like to thank Dr Marie-Jose Montpetit from MIT for reviewing and commenting on this draft.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

[AHL-00] Ahlswede, R., Ning Cai, , Li, S., and R. Yeung, "Network information flow", IEEE Transactions on Information Theory vol.46, no.4, pp.1204,1216, July 2000.

[FECFRAME] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", Request for Comments 6363, October 2011.

[NWCRG-ARCH] NWCRG, , "Network Coding Architecture", TBD TBD.

[RMT] Vicisano, L., Gemmel, J., Rizzo, L., Handley, M., and J. Crowcroft, "Forward Error Correction (FEC) Building Block", Request for Comments 3452, December 2002.

[Tetrys] Lacan, J. and E. Lochin, "Rethinking reliability for long-delay networks", International Workshop on Satellite and Space Communications 2008 (IWSSC08), October 2008.

Authors' Addresses

Jonathan Detchart
ISAE
10, avenue Edouard-Belin
BP 54032
Toulouse CEDEX 4 31055
France

Email: jonathan.detchart@isae.fr

Emmanuel Lochin
ISAE
10, avenue Edouard-Belin
BP 54032
Toulouse CEDEX 4 31055
France

Email: emmanuel.lochin@isae.fr

Jerome Lacan
ISAE
10, avenue Edouard-Belin
BP 54032
Toulouse CEDEX 4 31055
France

Email: jerome.lacan@isae.fr

Vincent Roca
INRIA
655, av. de l'Europe
Inovallee; Montbonnot
ST ISMIER cedex 38334
France

Email: vincent.roca@inria.fr
URI: http://privatics.inrialpes.fr/people/roca/