Internet Research Task Force (IRTF)
Request for Comments: 9407
Category: Experimental
ISSN: 2070-1721

J. Detchart
ISAE-SUPAERO
E. Lochin
ENAC
J. Lacan
ISAE-SUPAERO
V. Roca
INRIA
June 2023

Tetrys: An On-the-Fly Network Coding Protocol

Tetrys - протокол сетевого кодирования «на лету»

Аннотация

Этот документ описывает Tetrys - протокол сетевого кодирования «на лету» (on-the-fly), который можно применять для чувствительной к задержкам и потерям доставки через сеть с потерями. Tetrys может восстанавливаться после удаления данных с независимой от RTT задержкой, благодаря передаче кодированных пакетов. Этот документ представляет отчёт о результатах, полученных автором в процессе разработки и тестирования протокола Tetrys в реальных условиях.

Документ является результатом исследовательской группы NWCRG[1] и соответствует таксономии NWCRG, описанной в RFC 8406.

Статус документа

Документ не относится к категории Internet Standards Track и публикуется для проверки, экспериментальной реализации и оценки.

Документ является результатом работы IRTF[2]. IRTF публикует результаты относящихся к Internet исследований и разработок. Эти результаты могут оказаться не пригодными для реализации. Данный RFC представляет согласованное мнение исследовательской группы Quantum Internet в рамках IRTF. Документы, одобренные для публикации IRSG, не претендуют на статус Internet Standard (см. раздел 2 в RFC 7841).

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc9407.

Оглавление

[1]Coding for Efficient NetWork Communications Research Group - исследовательская группа по кодированию для эффективных сетевых коммуникаций.
[2]Internet Research Task Force - комиссия по исследовательским задачам Internet.

1. Введение

Этот документ является результатом работы исследовательской группы NWCRG и выражает её согласованное мнение. Документ не является результатом или стандартом IETF.

Документ описывает протокол Tetrys выполняющий сетевое кодирование на лету, который может служить для доставки чувствительных к задержкам и потерям данных через сеть с потерями. Сетевые коды появились в начале 2000-х гг. [AHL-00] и служили для снятия ограничений при передаче через Internet (задержки, пропускная способность, потери пакетов). Хотя сетевые коды получили сравнительно малое распространение в сообществе Internet, использование кодов удаления на уровне приложений уже стандартизовано в рамках IETF рабочими группами RMT [RFC5052] [RFC5445] и FECFRAME [RFC8680]. Представленный здесь протокол можно считать расширением сетевого кодирования на стандартные протоколы индивидуального (unicast) транспорта, а в некоторых случаях также группового и anycast. Текущее предложение можно рассматривать как комбинацию сетевого кодирования стирания и механизмов обратной связи [Tetrys] [Tetrys-RT].

Основным новшеством протокола Tetrys является генерация кодированных пакетов из эластичного окна кодирования. Это окно заполняется любыми пакетами источника, приходящими из входного потока, и периодически обновляется откликами от получателя. Сообщения с откликами предоставляют источнику сведения о наибольшем порядковом номере полученного или заново собранного (rebuilt) пакета, которые позволяют очистить (удалить) соответствующие пакеты источника в окне кодирования. Размер окна может быть фиксированным или динамически изменяемым. Если окно заполнено, входящие пакеты источника заменяют наиболее старые пакеты из окна, которые отбрасываются. На практике нужно выбирать корректный размер окна. Протокол Tetrys позволяет справляться с потерями как на прямом, так и на обратном пути и особенно устойчив к потере подтверждений. Операции протокола более подробно описаны в разделе 4. Базовые функции Tetrys.

Кодированный пакет в Tetrys представляет собой линейную комбинацию над конечным полем пакетов данных источника в окне кодирования. Выбор коэффициентов в качестве элементов конечного поля определяется компромиссом между возможностями удаления при стирании (конечное поле из 256 элементов) и ограничениями системы (предпочтительно конечное поле из 16 элементов), задаваемым приложением.

Благодаря эластичному окну кодирования, кодированные пакеты создаются «на лету» путём использования предопределённого метода выбора коэффициентов. Фактор избыточности может регулироваться динамически, а коэффициенты могут генерироваться разными способами в процессе передачи. По сравнению с блочными кодами упреждающей коррекции ошибок (Forward Error Correction или FEC) сокращается расход полоса и вносимые задержки.

Описание устройства протокола Tetrys в этом документе дополнено опытом, полученным авторами в процессе разработки и тестирования Tetrys в реальных условиях. В частности, в разделе 6 рассмотрено несколько исследовательских тем на основе опыта и наблюдений.

1.1. Уровни требований

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они выделены шрифтом, как показано здесь.

2. Определения, обозначения и сокращения

Используемые в документе обозначения основаны на таксономии NWCRG [RFC8406].

Source Symbol - символ источника
Символ, переносимый между входом и выходом сети.

Coded Symbol - кодированный символ
Линейная комбинация над конечным полем набора символов источника.

Source Symbol ID - идентификатор символа источника
Порядковый номер для идентификации символов источника.

Coded Symbol ID - идентификатор кодированного символа
Порядковый номер для идентификации кодированных символов.

Encoding Coefficients - коэффициенты кодирования
Элементы конечного поля, характеризующие линейную комбинацию для генерации кодированных символов.

Encoding Vector - вектор кодирования
Набор коэффициентов кодирования и идентификаторов входных символов источника.

Source Packet - пакет источника
Пакет источника содержит символы источника с их идентификаторами.

Coded Packet - кодированный пакет
Кодированный пакет содержит кодированные символы, их идентификаторы и вектор кодирования.

Input Symbol - входной символ
Символ на входе кодировщика Tetrys.

Output Symbol - выходной символ
Символ, создаваемый кодером Tetrys. Для несистематического режима все выходные символы являются кодированными, а в систематическом режиме выходные символы **могут** быть входными символами и ряд кодированных символов, которые являются линейной комбинацией входных символов плюс векторы кодирования.

Feedback Packet - пакет обратной связи
Пакет, содержащий сведения о декодированных или принятых символах источника. Он **может** также включать сведения о частоте ошибок в пакетах (Packet Error Rate) или количестве разных пакетов в окне декодирования получателя.

Elastic Encoding Window - эластичное окно кодирования
Буфер на стороне кодера, где хранятся все неподтвержденные пакеты источника из входного потока, вовлечённого в процесс кодировани.

Coding Coefficient Generator Identifier (CCGI) - идентификатор генератора коэффициентов кодирования
Уникальный идентификатор, который указывает функцию или алгоритм, разрешающие генерацию вектора кодирования.

Code Rate - степень кодирования
Отношение между числом входных и выходных символов.

3. Архитектура

3.1. Применение

Tetrys хорошо подходит для отдельного потока из одного источника с внутрипотоковым кодированием на одном узле, но может применяться и в других случаях. Отметим, что входной поток **может** может быть мультиплексом из нескольких потоков вышележащего уровня. Передача **может** происходить по одному или нескольким путям. Это простейший вариант применения, вполне соответствующий предлагаемым сегодня сценариям для сквозных потоков.

3.2. Обзор

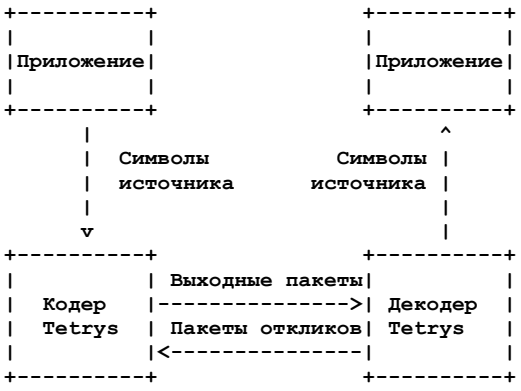

Рисунок 1. Архитектура Tetrys.

Протокол Tetrys обладает несколькими важными функциональными возможностями. Обязательные функции включают:

- кодирование «на лету»;
- декодирование;
- сигнализация для доставки, в частности, идентификаторов символов о окне кодирования и связанных коэффициентов кодирования, когда это значимо;
- управления обратной связью;
- управления эластичным окном;
- создание и обработка заголовков Tetrys.

Необязательные функции включают:

- оценку канала;
- динамическую подстройку степени кодирования (code rate) и управление потоком данных;
- управление контролем перегрузок, когда это приемлемо (6.1. Взаимодействие с контролем перегрузок).

Функциональность обеспечивается несколькими основными блоками.

Tetrys Building Block - блок Tetrys
Этот блок содержит кодер и декодер Tetrys и применяется в процессах кодирования и декодирования. Нужно отметить, что Tetrys не требует конкретного блока и могут применяться любые блоки, совместимые с эластичным окном кодирования Tetrys.

Window Management Building Block - блок управления окном
Управляет окном кодирования у отправителя Tetrys.

Для упрощения добавления новых компонентов и служб в Tetrys включён механизм расширения заголовка, совместимый с транспортом многоуровневого кодирования (Layered Coding Transport или LCT) [RFC5651], ориентированной на негативные подтверждения надёжной групповой передачей (NACK-Oriented Reliable Multicast или NORM) [RFC5740] и моделью упреждающей коррекции ошибок (FEC Framework или FECFRAME) [RFC8680].

4. Базовые функции Tetrys

4.1. Кодирование

В начале передачи кодер Tetrys **должен** выбрать степень кодирования (code rate) для добавления избыточности, поскольку он не знает частоту потери пакетов в канале. В установившемся состоянии кодер Tetrys **может** генерировать кодированные символы при получении символов источника от приложения или какого-либо отклика от блоков декодирования в зависимости от степени кодирования.

Когда кодеру Tetrys нужно создать кодированный символ, он рассматривает набор символов источника, хранящихся в эластичном окне кодирования и генерирует вектор кодирования с кодированным символом. Эти символы источника - набор символов источника, ещё не подтверждённых получателем. Для каждого символа источника определяется коэффициент конечного поля с использованием генератора коэффициентов кодирования. Этот генератор **может** принимать идентификаторы символов источника на входе и **может** определять коэффициент детерминированным способом, представленным в параграфе 5.3. Формат кодированного пакета. Кодированный символ является суммой символов источника, умноженных на соответствующие коэффициенты.

Кодер Tetrys **должен** задавать предел для максимального размера эластичного окна кодирования. Это позволяет контролировать алгоритмическую сложность в кодере и декодере путём ограничения размера линейных комбинаций. Это требуется также в ситуациях, когда все пакеты обновления окна потеряны или отсутствуют.

4.2. Эластичное окно кодирования

Когда входной символ источника передаётся кодеру Tetrys, он добавляется в эластичное окно кодирования. Это окно **должно** иметь предел, заданный блоком кодирования. Если эластичное окно кодирования достигает предела, оно перемещается (скользит) относительно символов с удалением первого (самого старого) и добавлением нового символа. Как элемент окна кодирования этот символ включается в последующие линейные комбинации при генерации кодированных символов.

Как разъяснено ниже, декодер Tetrys периодически передаёт отклики, указывающие принятые или декодированные символы источника. При получении отправителем сведений о приёме или декодировании получателем символа источника он удаляет этот символ из окна кодирования.

4.3. Декодирование

Для восстановления удалённого символа источника достаточно стандартного исключения по Гауссу, когда это позволяет ранг матрицы.

5. Формат пакетов

5.1. Формат базового заголовка

Все пакеты Tetrys используют общий формат базового заголовка (Рисунок 2).

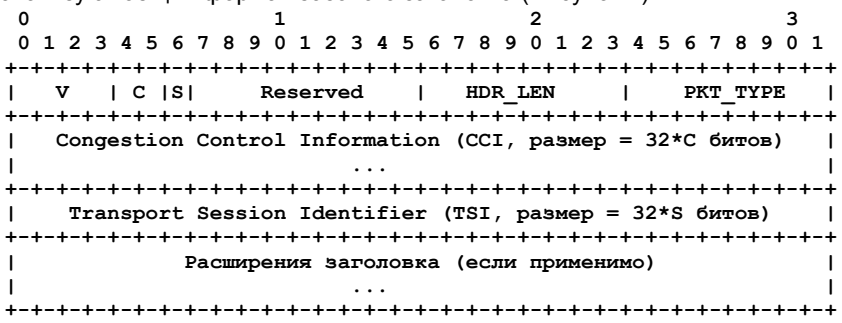

Рисунок 2. Формат базового заголовка.

Как отмечено выше, этот формат наследуется из формата заголовка LCT [RFC5651] с незначительными изменениями.

Номер версии Tetrys (V) - 4 бита
Указывает номер версии протокола Tetrys. Эта спецификация Tetrys задаёт версию 1.

Флаг контроля перегрузок (C) - 2 бита
Значение C = 0b00 указывает, что поле сведений контроля перегрузки (Congestion Control Information или CCI) имеет размер 0. C = 0b01 указывает, что поле CCI имеет размер 32 бита, C = 0b10 указывает CCI размером 64 бита, а C = 0b11 - 96 битов.

Флаг идентификатора транспортной сессии (S) - 1 бит
Число полных 32-битовых слов в поле TSI (0 или 1).

Reserved (Resv) - 9 битов
резервные биты, которые отправитель **должен** сбрасывать (0), а получатель **должен** игнорировать.

Размер заголовка (HDR_LEN) - 8 битов
Общий размер заголовка Tetrys в 32-битовых словах. Заголовок Tetrys **должен** иметь размер, кратный 32 битам. Это поле может применяться для прямого доступа к пакету Tetrys за пределами базового заголовка, т. е. к первому следующему заголовку (при наличии), данным пакета (при наличии) или к концу пакета, если дополнительных заголовков и данных нет.

Тип пакета Tetrys (PKT_TYPE) - 8 битов
Имеется 3 типа пакетов - PKT_TYPE_SOURCE (0b00), параграф 5.2, PKT_TYPE_CODED (0b01), параграф 5.3, и PKT_TYPE_WND_UPT (0b11) для пакетов обновления окна, параграф 5.4.

Congestion Control Information - сведения контроля перегрузки (CCI) - 0, 32, 64 или 96 битов
Служит для передачи сведения о контроле перегрузки и может включать, например, номера уровней, логических каналов и порядковые номера. Данная спецификация не задаёт обработку этого поля (opaque). Поле **должно** отсутствовать (о битов) при C = 0b00 и **должно** занимать 32 бита при C = 0b01, 64 - при C = 0b10, 96 - при C = 0b11.

Transport Session Identifier - идентификатор транспортной сессии (TSI) - 0 или 32 бита

TSI однозначно указывает сессию среди имеющихся с конкретным кодером Tetrys. Область действия TSI определяется IP-адресом отправителя, поэтому адрес отправителя и TSI совместно указывают сессию однозначно. Хотя TSI всегда однозначно указывает сессию совместно с IP-адресом отправителя, включение TSI в заголовок Tetrys зависит от того, что применяется в качестве значения TSI. Если базовым транспортом служит UDP, вместо TSI для сессии **может** служить 16-битовый номер порта UDP у отправителя. Если сетевой транспорт или иной уровень не задаёт базовый идентификатор TSI, значение TSI **должно** включаться в заголовок Tetrys.

5.1.1. Расширения заголовка

Расширения заголовка служат в Tetrys для размещения полей, применяемых не всегда или имеющих переменный размер. Наличие расширений заголовка **можно** определить по значению поля размера заголовка Tetrys (HDR_LEN). Если HDR_LEN превышает размер стандартного заголовка, остальное пространство заголовка занимает расширение.

При наличии расширений заголовка они **должны** обрабатываться, чтобы обеспечить их распознавание до выполнения процедуры контроля перегрузки или иного восприятия пакета. Нераспознанные расширения заголовка по умолчанию игнорируются. Это позволит в будущем вносить в Tetrys совместимые с прежними версиями улучшения без смены номера версии Tetrys. Расширения заголовков, не совместимые с прежними версиями, **недопустимо** вносить без смены номера версии Tetrys.

Имеется два формата расширений заголовка, показанных на рисунке 3.

- Первый формат служит для расширений переменного размера с типами расширения заголовка (header extension type или HET) от 0 до 127.
- Второй тип служит для расширений фиксированного размера (1 слово в 32 бита), использующих HET от 128 до 255.

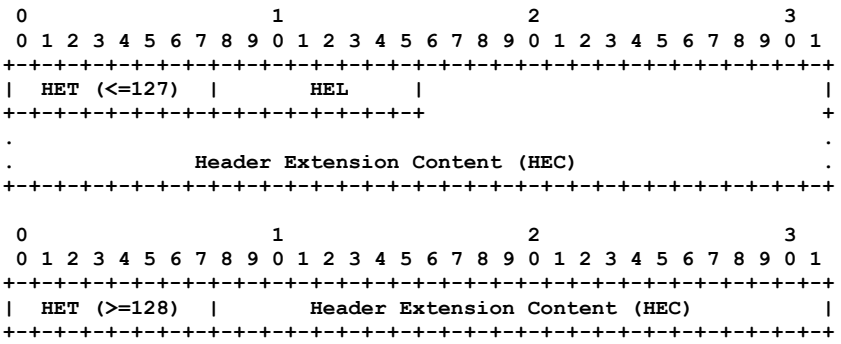

Рисунок 3. Формат расширения заголовка.

Тип расширения заголовка (HET) - 8 битов

Этот документ задаёт несколько возможных типов, а новые типы могут вноситься будущими версиями спецификации. Значения HET от 0 до 127 служат для расширений заголовка с переменным размером, от 128 до 255 - для расширений заголовка размером 32 бита.

Размер расширения заголовка (HEL) - 8 битов

Число 32-битовых слов в расширении заголовка. Это поле **должно** присутствовать в расширениях переменного размера (HET от 0 до 127) и его **недопустимо** включать в расширения фиксированного размера (HET от 128 до 255).

Header Extension Content - содержимое расширения заголовка (HEC) - переменный размер

Содержимое расширения заголовка, формат которого зависит от типа расширения. Для расширений заголовка с фиксированным размером HEC = 24, для расширений переменного размера значение HEC указывает переменный размер, как задано полем HEL. Отметим, что размер расширения заголовка **должен** быть кратным 32 битам. Общий размер заголовка Tetrys, включая заголовок расширения и его необязательные поля, не может превышать 255 (32-битовых слов).

5.2. Формат пакета источника

Пакет источника содержит базовый заголовок, идентификатор символа источника и сам символ (payload), который может иметь переменный размер.

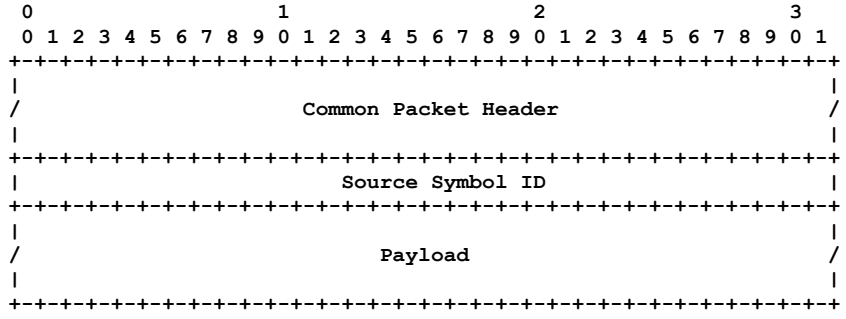

Рисунок 4. Формат пакета источника.

Common Packet Header

Базовый заголовок пакета, где указан тип 0b00.

Source Symbol ID

Порядковый номер символа источника.

Payload

Содержимое пакет (символ источника).

5.3. Формат кодированного пакета

Кодированный пакет включает базовый заголовок, идентификатор кодированного символа, связанный вектор кодирования и закодированный символ (payload). Поскольку символы источника **могут** иметь переменный размер, требуется кодировать размеры всех символов источника. Для генерации этих размеров закодированного содержимого в форме 16-битовых значений без знака используется линейная комбинация с теми же коэффициентами, что и для кодируемого содержимого. Результат **должен** сохраняться в 16-битовом поле кодированного пакета. Поскольку это поле необязательно, вектор кодирования **должен** указывать его наличие в поле V (5.3.1. Вектор кодирования).

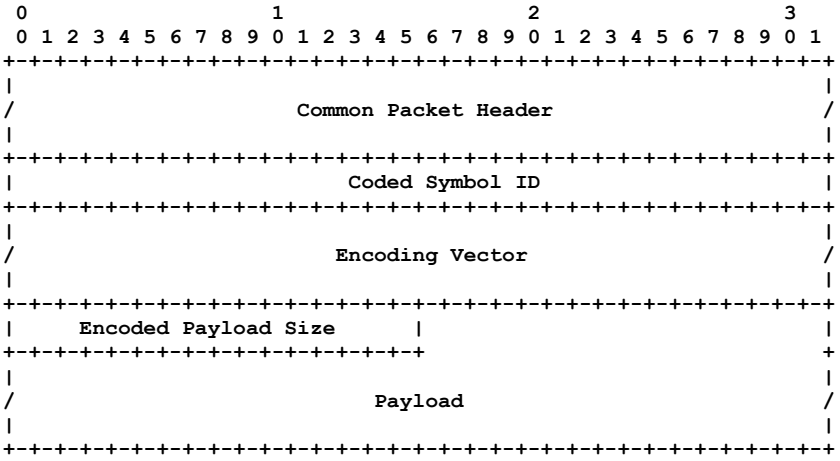

Рисунок 5. Формат кодированного пакета.

Common Packet Header

Базовый заголовок пакета с типом 0b01.

Coded Symbol ID

Порядковый номер для идентификации кодированного символа.

Encoding Vector

Вектор кодирования, задающий использованную линейную комбинацию (коэффициенты и символы источника).

Encoded Payload Size

Размер кодированного содержимого, если символы источника имеют переменный размер (необязательно, см. параграф 5.3.1. Вектор кодирования).

Payload

Кодированный символ.

5.3.1. Вектор кодирования

Вектор кодирования содержит все сведения о линейной комбинации, используемой для генерации кодированного символа. Эта информация включает идентификаторы источника и коэффициенты, используемые для каждого символа источника. Вектор **может** храниться разными способами в зависимости от ситуации.

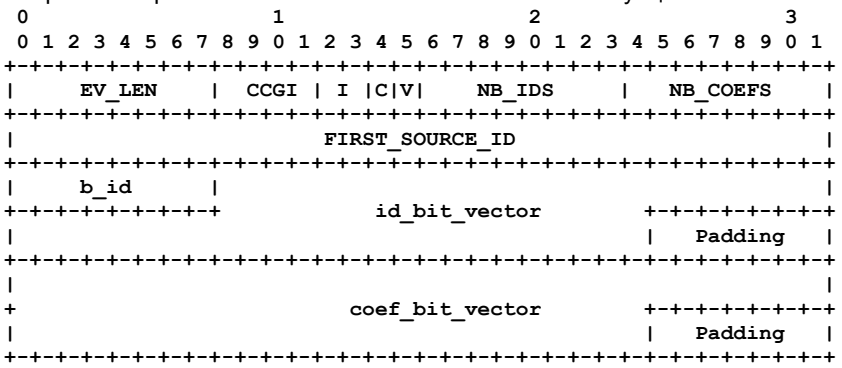

Рисунок 6. Формат вектора кодирования.

Размер вектора кодирования (EV_LEN) - 8 битов

Число 32-битовых слов.

Идентификатор генератора коэффициентов кодирования (CCGI) - 4 бита

Идентификатор для указания алгоритма или функции, применяемой для генерации коэффициентов. Поскольку CCGI включается в каждый кодированный вектор, значение **может** динамически меняться между генерацией двух кодированных символов. CCGI задают создание коэффициентов для генерации кодированных символов и **должны** быть известны всем кодерам и декодерам Tetrys. Заданные в этом документе две схемы RLC FEC используют конечные поля, определённые в параграфе 8.1 [RFC5510]. Элементы поля GF($2^{(m)}$) представлены полиномами с двоичными коэффициентами (т. е. над GF(2)) со степенью не выше m-1. Сложение двух элементов определяется как сложение двоичных полиномов в GF(2), которое эквивалентно побитовой операции XOR над двоичными представлениями этих элементов. При GF($2^{(8)}$) перемножение двух элементов является умножением по модулю данного неприводимого полинома степени 8. Для GF($2^{(8)}$) применяется неприводимый полином

$$x^{(8)} + x^{(4)} + x^{(3)} + x^{(2)} + 1$$

При GF($2^{(4)}$) перемножение двух элементов является умножением по модулю данного неприводимого полинома степени . Для GF($2^{(4)}$) применяется неприводимый полином

$$x^{(4)} + x + 1$$

- 0b00 - коэффициенты Vandermonde над конечным полем GF($2^{(4)}$) в виде alpha$^{((source_symbol_id*coded\text{-}symbol_id) \% 16)}$, где alpha - корень примитивного полинома;
- 0b01 - коэффициенты Vandermonde над конечным полем GF($2^{(8)}$) в виде alpha$^{((source_symbol_id*coded\text{-}symbol_id) \% 256)}$, где alpha - корень примитивного полинома;

- Предположим, что нужно создать кодированный символ 2 как линейную комбинацию символов 1, 2 и 4 с использованием CCGI = 0b01. Коэффициентами будут $alpha^{((1 * 1) \% 256)}$, $alpha^{((1 * 2) \% 256)}$ и $alpha^{((1 * 4) \% 256)}$.

Формат хранения идентификаторов символов источника (I) - 2 бита

- 0b00 - нет сведений об идентификаторе символа источника;
- 0b01 - вектор кодирования содержит граничные блоки идентификаторов символов источника без сжатия;
- 0b10 - вектор кодирования содержит сжатый список идентификаторов символов источника;
- 0b11 - вектор кодирования содержит сжатые граничные блоки идентификаторов символов источника.

Хранение коэффициентов кодирования (C) - 1 бит

Указывает, содержит ли вектор кодирования сведения об используемых коэффициентах.

Наличие символов источника с кодированием переменного размера (V)

V = 0b01, если комбинация, указывающая вектор кодирования, содержит символы источника с переменным размером. В этом случае кодированные пакеты должны включать поле Encoded Payload Size.

NB_IDS

Число идентификаторов источника, сохранённых в векторе кодирования (зависит от I).

Число коэффициентов (NB_COEFS)

Число коэффициентов, используемых для генерации соответствующего кодированного символа.

Первый идентификатор источника (FIRST_SOURCE_ID)

Идентификатор первого символа источника, используемого в комбинации.

Число битов в каждом граничном блоке (b_id)

Число битов, требуемое для сохранения границы (edge).

Сведения об идентификаторах символов источника (id_bit_vector)

При I = 0b01 граничные блоки сохраняются как b_id * (NB_IDS * 2 - 1), при I = 0b10 граничные блоки сжимаются.

Коэффициенты (coef_bit_vector)

Коэффициенты, сохраняемые в зависимости от CCGI (4 или 8 битов на каждый коэффициент).

Padding

Заполнение для выравнивания размера вектора кодирования по 32-битовой границе (для ID и коэффициентов).

Идентификаторы символов источника организуются в упорядоченный список 32-битовых целых чисел без знака. В зависимости от откликов идентификаторы символов источника **могут** идти подряд или содержать пропуски. Если идентификаторы идут подряд, границы сохраняются в векторе кодирования, для чего достаточно 2*32 битов. При наличии пропусков **может** сохраняться полный список или граничные блоки и **могут** применяться различные преобразования для снижения числа используемых битов.

Для последующих параграфов возьмём пример генерации вектора кодирования для кодированного символа, являющегося линейной комбинацией символов источника с идентификаторами 1, 2, 3, 5, 6, 8, 9, 10 (краевые блоки [1..3], [5..6], [8..10]). Есть несколько способов сохранить идентификаторы символов источника в векторе кодирования.

- Если сведения об идентификаторах символов источника не нужны, **должно** устанавливаться I = 0b00, а b_id и id_bit_vector не используются.
- Если граничные блоки сохраняются без сжатия, **должно** устанавливаться I = 0b01. В этом случае устанавливается b_id = 32 (идентификатор символа содержит 32 бита), а список 32-битовых целых чисел без знака (1, 3, 4, 5, 6, 10) помещается в id_bit_vector.
- Если идентификаторы символов источника сохраняются в виде списка со сжатием, **должно** устанавливаться I = 0b10. Этот случай описан в параграфе 5.3.1.1, но вместо сжатия граничных блоков сжимается весь список идентификаторов символов источника.
- Если идентификаторы символов источника сохраняются со сжатием, **должно** устанавливаться I = 0b11 (см. 5.3.1.1. Сжатый список идентификаторов символов источника).

5.3.1.1. Сжатый список идентификаторов символов источника

Продолжим с определенным в предыдущем параграфе кодированным символом с идентификаторами символов источника в линейной комбинации [1..3], [5..6], [8..10]. Для сжатия и сохранения этого списка в векторе кодирования должна применяться приведённая ниже процедура.

1. Первый элемент пакета (1) кодируется как first_source_id.
2. Выполняется дифференциальное преобразование путём вычитания элемента ([3, 5, 6, 8, 10]) из последующего с использованием в качестве «нулевого» элемента значения first_source_id, это даёт список L = [2, 2, 1, 2, 2].
3. Рассчитывается число битов (b), требуемое для сохранения элемента списка, как ceil(log2(max(L))), где max(L) - самый большой элемент в списке L. В нашем случае будет b = 2.
4. Значение b записывается в соответствующее поле (b_id), а из элементов списка создаётся вектор вида b * [(2 * NB) - 1], где NB - число элементов. В результате получается вектор битов 10, 10, 01, 10, 10.

5.3.1.2. Декомпрессия идентификаторов символов источника

Когда блоку декодирования Tetrys нужно выполнить обратные операции, используется приведённый ниже алгоритм.

1. Перестраивается список переданных элементов путём считывания вектора битов и b, а затем преобразования в десятичную форму [10, 10, 01, 10, 10] => [2, 2, 1, 2, 2].
2. Выполняется обратное прежнему преобразование путём сложения элементов, начиная с first_source_id, т. е. [1, 1 + 2, (1 + 2) + 2, (1 + 2 + 2) + 1, ...] => [1, 3, 5, 6, 8, 10].
3. Перестраиваются блоки с использованием списка и first_source_id [1..3], [5..6], [8..10].

5.4. Формат пакета обновления окна

Декодер Tetrys **может** передавать другому блоку пакеты обновления окна со сведениями о полученных, декодированных или отброшенных пакетов и другую информацию, такую как частота потерь или размер буферов

декодирования. Пакеты применяются для оптимизации содержимого окна кодирования, являются **необязательными** и их пропуск или потеря при передаче не влияют на поведение протокола.

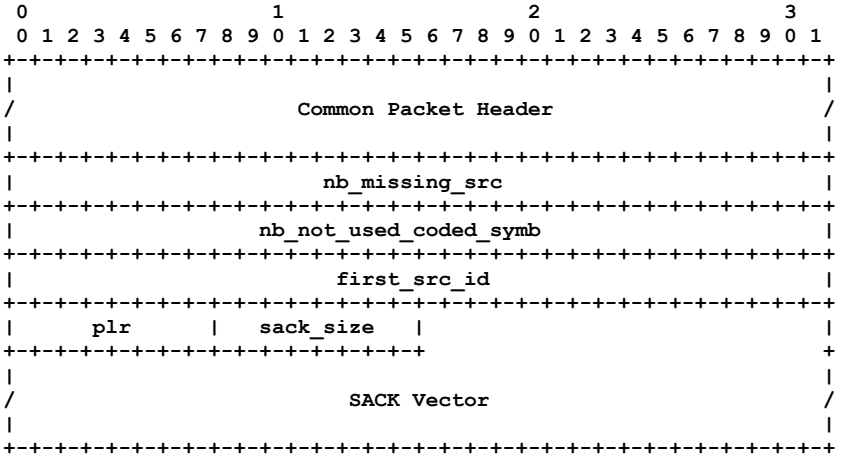

Рисунок 7. Формат пакета обновления окна.

Common Packet Header
Базовый заголовок с полем типа пакета 0b10.

nb_missing_src
Число пропущенных символов источника у получателя с начала сессии.

nb_not_used_coded_symb
Число кодированных символов у получателя, которые ещё не использованы для декодирования (например, в линейной комбинации есть хотя бы 2 неизвестных символа источника).

first_src_id
Идентификатор первого символа источника для указания в векторе частичного подтверждения (SACK).

plr
Доля потерь, выраженная в процентах нормализованным 8-битовым целым числом без знака. Например, 2,5% будут представлены как floor(2,5 * 256/100) = 6. И обратно, если указано значение 6, соответствующая доля теряемых пакетов в процентах будет рассчитываться как 6*100/256 = 2,34%. Значение применяется при динамической степени кодирования (code rate) или для статистики. Выбор расчёта остаётся за декодером Tetrys и зависит от наблюдения за окном, но значение PLR следует просматривать перед декодированием.

sack_size
Размер вектора SACK в 32-битовых словах. Например, значение 2 указывает вектор SACK размером 64 бита.

SACK vector
Битовый вектор, указывающий символы, которые должны быть удалены из окна кодирования, начиная с идентификатора первого символа источника. В большинстве случаев эти символы уже получены приёмником. Иногда это может быть связано с невосстановимыми потерями (например, при всплеске числа потерь), когда лучше отбросить и отказаться от некоторых пакетов и удалить их из окна кодирования, чтобы можно было восстановить последующие пакеты. Первый символ источника (First Source Symbol или first_src_id) включается в этот вектор. Бит со значением 1 в позиции i означает, что этот пакет обновления окна удаляет из окна кодирования символ источника с номером first_src_id + i.

6. Темы исследований

В этом документе описан базовый протокол, позволяющий взаимодействовать кодеру и декодеру Tetrys. На практике Tetrys можно применять как автономный протокол или встраивать его в имеющиеся протоколы на транспортном уровне, а также выше или ниже его. С каждым из этих вариантов связаны вопросы, требующие исследований для улучшения протокола. Сводка таких вопросов приведена в последующих параграфах.

6.1. Взаимодействие с контролем перегрузок

Tetrys и компоненты контроля перегрузок создают 2 отдельных канала (см. параграф 2.1 в [RFC9265]).

- Канал Tetrys передаёт пакеты источника и кодированные пакеты (от отправителя к получателю), а также информацию от получателя к отправителю (например, сигналы о восстановленных символах, частоте потерь до и/или после декодирования и т. п.).
- Канал контроля перегрузки переносит пакеты от отправителя к получателю и сигнальные сведения о сети (например, число принятых пакетов по сравнению с потерянными, маркеры ECN[1] и т. п.) от получателя к отправителю.

Ниже указаны темы, отмеченные и рассмотренные в [RFC9265], которые уже адаптированы в конкретных внедрениях Tetrys (на транспортном уровне, а также выше или ниже его).

- Связанные с перегрузкой потери могут быть скрыты, если протокол Tetrys развернут ниже транспортного уровня без каких-либо мер предосторожности (т. е. Tetrys восстанавливает пакеты, потерянные из-за перегрузки маршрутизатора), что может существенно влиять на эффективность контроля перегрузок. Подход для предотвращения сокрытия таких сигналов представлен в разделе 5 [RFC9265].
- Наличие потоков Tetrys наряду с другими потоками в одном канале сети может препятствовать беспристрастному распределению между потоками. В частности, это зависит от наличия и типа контроля перегрузки на отдельных потоках. Детали этого выходят за рамки документа, но могут оказывать существенное влияние.

[1] Explicit Congestion Notification - явное уведомление о перегрузке.

- Адаптация степени кодирования внутри Tetrys может оказывать сильное влияние на контроль перегрузки при ненадлежащем выполнении (см. 6.2. Адаптивная степень кодирования).
- Tetrys может использовать передачу по нескольким путём для одной пары отправитель-получатель. Поскольку пути могут существенно различаться, может потребоваться адаптация управления потоком данных и контроля перегрузки на каждом пути.
- Защита нескольких потоков приложений внутри одного потока Tetrys вызывает дополнительные вопросы (см. 6.3. Использование Tetrys ниже уровня IP для туннелирования).

6.2. Адаптивная степень кодирования

Когда условия в сети (например, задержки и частота потерь) сильно меняются со временем, можно использовать адаптивную степень кодирования для динамического повышения или сокращения числа кодированных пакетов при передаче (т. е. добавления избыточности) с помощью специального алгоритма, такого как [A-FEC]. Стратегия различается в зависимости от уровня, где развёрнут протокол Tetrys (по отношению к транспортному уровню). Можно разделить стратегии на 2 класса - внедрение Tetrys на транспортном уровне и вне его (выше или ниже). Развёртывание внутри транспортного уровня означает возможность взаимодействия между транспортными механизмами, такими как восстановление при ошибках, контроль перегрузок и/или управление потоком данных. Внедрение Tetrys в транспортных протоколах без контроля перегрузки, таких как UDP, не дало бы никаких преимуществ перед внедрением за пределами транспортного уровня.

Влияние внедрения механизма FEC внутри транспортного уровня рассмотрено в разделе 4 [RFC9265], где исследуются вопросы взаимодействия контроля перегрузок и степени кодирования, а также влияние беспристрастности. Такая адаптация возможна в сочетании с механизмом контроля перегрузок протокола транспортного уровня, как предложено в [CTCP]. Это позволяет использовать отслеживаемые показатели контроля перегрузок (например, RTT, события перегрузки, текущий размер окна перегрузки) для адаптации степени кодирования с учётом рассчитанной скорости транспортной отправки. Смысл заключается в расчёте объёма ремонтного трафика, который не вызывает перегрузки. Такая связанная оптимизация обязательна для предотвращения захвата потоком всей доступной пропускной способности, как описано в [RMCAT-ADAPTIVE-FEC], где авторы отмечают, что коэффициент восстановления (repair ratio) следует повышать при одновременном снижении скорости передачи от источника.

Адаптация степени кодирования может выполняться вне транспортного уровня без учёта показателей этого уровня. В частности, это можно делать вместе с сетью, как предложено в [RED-FEC]. Авторы этой статьи предлагают механизм случайного раннего обнаружения (Random Early Detection) FEC в контексте передачи видео через беспроводную сеть. Вкратце, идея заключается в увеличении числа избыточных пакетов, когда очередь в точке доступа менее загружена, и наоборот. Предложена первая теоретическая попытка доставки видео с применением Tetrys [THAI]. Этот подход интересен тем, что он показывает взаимодействие между требованиями приложения и условиями в сети, а также объединяет сигналы о потребностях приложения и состоянии сети (т. е. ниже и выше транспортного уровня).

В заключения отметим, что имеется много способов включить адаптивную степень кодирования, однако все они зависят от

- сигнальных показателей, которые можно отслеживать и применять для адаптации степени кодирования;
- используемого на транспортном уровне контроля перегрузок;
- преследуемой цели (например, минимизации перегрузок или выполнения требования приложения).

6.3. Использование Tetrys ниже уровня IP для туннелирования

Применение Tetrys для защиты агрегата потоков требует исследования вопроса восстановления дейтаграмм IP, потерянных при туннелировании. Применение избыточности без дифференциации потоков может противоречить требованиям по обслуживанию отдельных потоков, поскольку некоторые из них могут быть чувствительны к задержкам и их вариациям, а другие - к надёжности доставки. На практике блокировка в голове очереди (head-of-line) аналогично влияет на все потоки, несмотря на их разные потребности, что требует более продуманной стратегии в Tetrys.

7. Вопросы безопасности

Для начала следует осознать, что применение защиты FEC для потока данных не обеспечивает безопасности и даже повышает уровень риска. Ситуация с протоколом Tetrys похожа на использование других протоколов доставки содержимого в применением FEC и это хорошо описано в FECFRAME [RFC6363]. Этот раздел основан на данном документе и рассматривает дополнительные соображения, относящиеся к Tetrys, когда это уместно.

7.1. Постановка задачи

Целью атакующего может быть содержимое, протокол или сеть. Последствия будут существенно разными в зависимости от целей, таких как получение доступа к конфиденциальным сведениям, повреждение содержимого, компрометация кодера и/или декодера Tetrys, нарушение поведения сети. В частности, некоторые атаки направлены на отказ в обслуживании (Denial-of-Service или DoS) и их последствия могут ограничиваться одним узлом (например, декодером Tetrys) или влиять на все узлы, подключённые к соответствующей сети (например, делая потоки невосприимчивыми к сигналам перегрузки).

В последующих параграфах атаки рассматриваются по компонентам, на которые нацелены злоумышленники.

7.2. Атаки на поток данных

Целью атакующего может быть получение доступа к конфиденциальным сведениям путём перехвата трафика между кодером и декодером Tetrys. Защитой от этого обычно служит шифрование трафика и его можно применить к потоку от источника к кодеру Tetrys или к пакетам, исходящим от кодера Tetrys. Выбор точки шифрования зависит от разных критериев, а частности, от модели злоумышленника (например, шифрование ниже уровня Tetrys применяется при наличии риска со стороны межсетевого обмена).

Целью атаки может быть повреждение содержимого (например, путём внедрения поддельных или изменённых пакетов источника или кодированных пакетов, чтобы помешать декодеру Tetrys восстановить исходный поток источника). Службы защиты целостности и проверки подлинности источника на уровне пакетов помогут снизить такой риск. Эти услуги нужно предоставлять ниже уровня Tetrys, чтобы получатель мог отбросит нежелательные пакеты и передавать декодеру Tetrys только легитимные пакеты. Следует отметить, что подделка или изменение пакетов обратной связи не повреждает содержимое, хотя может создавать угрозы для работы Tetrys (7.3. Атаки на сигнализацию).

7.3. Атаки на сигнализацию

Атаки на сигнальные данные (например, подделка или изменение пакетов обратной связи для фальсификации приёма или восстановления исходного содержимого) легко могут помешать декодеру Tetrys в восстановлении исходного потока, вызывая тем самым DoS. Для предотвращения таких атак нужны службы защиты целостности и проверки подлинности источников на уровне пакетов для откликов от декодера к кодеру Tetrys. Эти услуги нужно предоставлять ниже уровня Tetrys, чтобы получатель мог отбросит нежелательные пакеты и передавать кодеру Tetrys только легитимные пакеты.

Злоумышленник, способный выборочно отбрасывать пакеты обратной связи (вместо их изменения), не сможет существенно повлиять на работу Tetrys, поскольку протокол по своей природе устойчив к таким потерям. Однако это будет вызывать побочные эффекты, такие как применение более крупных линейных систем (поскольку кодер Tetrys не сможет удалить полученные и декодированные пакеты из своей линейной системы), что ведёт к вычислительным издержкам на обеих сторонах (кодер и декодер).

7.4. Атаки на сеть

Tetrys может реагировать на сигналы перегрузки (6.1. Взаимодействие с контролем перегрузок) для обеспечения некого уровня беспристрастности по отношению к другим потокам в общей сети. Это может использоваться злоумышленником для создания или усиления сигналов перегрузки (например, путём подделки или изменения пакетов обратной связи), которые могут повлиять на множество узлов, подключённых к сети. Для снижения риска нужны службы защиты целостности и проверки подлинности источников на уровне пакетов, позволяющие получателю отбрасывать нежелательные пакеты и передавать кодеру и декодеру Tetrys лишь легитимные пакеты.

7.5. Базовые операции защиты

Tetrys может использовать IPsec/ESP [RFC4303] для защиты конфиденциальности и целостности, аутентификации источника и предотвращения повторного использования пакетов (replay). IPsec/ESP можно применять для защиты потоков данных Tetrys (в обоих направлениях) от злоумышленников, находящихся в промежуточных сетях или способных перехватывать, внедрять и повторно использовать легитимные пакеты.

8. Взаимодействие с IANA

Этот документ не требует действий IANA.

9. Литература

9.1. Нормативные документы

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <https://www.rfc-editor.org/info/rfc5052>.

[RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, DOI 10.17487/RFC5445, March 2009, <https://www.rfc-editor.org/info/rfc5445>.

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, DOI 10.17487/RFC5510, April 2009, <https://www.rfc-editor.org/info/rfc5510>.

[RFC5651] Luby, M., Watson, M., and L. Vicisano, "Layered Coding Transport (LCT) Building Block", RFC 5651, DOI 10.17487/RFC5651, October 2009, <https://www.rfc-editor.org/info/rfc5651>.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <https://www.rfc-editor.org/info/rfc5740>.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <https://www.rfc-editor.org/info/rfc6363>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8406] Adamson, B., Adjih, C., Bilbao, J., Firoiu, V., Fitzek, F., Ghanem, S., Lochin, E., Masucci, A., Montpetit, M., Pedersen, M., Peralta, G., Roca, V., Ed., Saxena, P., and S. Sivakumar, "Taxonomy of Coding Techniques for Efficient Network Communications", RFC 8406, DOI 10.17487/RFC8406, June 2018, <https://www.rfc-editor.org/info/rfc8406>.

[RFC8680] Roca, V. and A. Begen, "Forward Error Correction (FEC) Framework Extension to Sliding Window Codes", RFC 8680, DOI 10.17487/RFC8680, January 2020, <https://www.rfc-editor.org/info/rfc8680>.

[RFC9265] Kuhn, N., Lochin, E., Michel, F., and M. Welzl, "Forward Erasure Correction (FEC) Coding and Congestion Control in Transport", RFC 9265, DOI 10.17487/RFC9265, July 2022, <https://www.rfc-editor.org/info/rfc9265>.

9.2. Дополнительная литература

[A-FEC] Bolot, J., Fosse-Parisis, S., and D. Towsley, "Adaptive FEC-based error control for Internet telephony", IEEE INFOCOM '99, Conference on Computer Communications, New York, NY, USA, Vol. 3, pp. 1453-1460, DOI 10.1109/INFCOM.1999.752166, March 1999, <https://doi.org/10.1109/INFCOM.1999.752166>.

[AHL-00] Ahlswede, R., Cai, N., Li, S., and R. Yeung, "Network information flow", IEEE Transactions on Information Theory, Vol. 46, Issue 4, pp. 1204-1216, DOI 10.1109/18.850663, July 2000, <https://doi.org/10.1109/18.850663>.

[CTCP] Kim, M., Cloud, J., ParandehGheibi, A., Urbina, L., Fouli, K., Leith, D., and M. Medard, "Network Coded TCP (CTCP)", arXiv 1212.2291v3, April 2013, <https://arxiv.org/abs/1212.2291>.

[RED-FEC] Lin, C., Shieh, C., Chilamkurti, N., Ke, C., and W. Hwang, "A RED-FEC Mechanism for Video Transmission Over WLANs", IEEE Transactions on Broadcasting, Vol. 54, Issue 3, pp. 517-524, DOI 10.1109/TBC.2008.2001713, September 2008, <https://doi.org/10.1109/TBC.2008.2001713>.

[RMCAT-ADAPTIVE-FEC] Singh, V., Nagy, M., Ott, J., and L. Eggert, "Congestion Control Using FEC for Conversational Media", Work in Progress, Internet-Draft, draft-singh-rmcat-adaptive-fec-03, 20 March 2016, <https://datatracker.ietf.org/doc/html/draft-singh-rmcat-adaptive-fec-03>.

[Tetrys] Lacan, J. and E. Lochin, "Rethinking reliability for long-delay networks", International Workshop on Satellite and Space Communications, Toulouse, France, pp. 90-94, DOI 10.1109/IWSSC.2008.4656755, October 2008, <https://doi.org/10.1109/IWSSC.2008.4656755>.

[Tetrys-RT] Tournoux, P., Lochin, E., Lacan, J., Bouabdallah, A., and V. Roca, "On-the-Fly Erasure Coding for Real-Time Video Applications", IEEE Transactions on Multimedia, Vol. 13, Issue 4, pp. 797-812, DOI 10.1109/TMM.2011.2126564, August 2011, <http://dx.doi.org/10.1109/TMM.2011.2126564>.

[THAI] Tran Thai, T., Lacan, J., and E. Lochin, "Joint on-the-fly network coding/video quality adaptation for real-time delivery", Signal Processing: Image Communication, Vol. 29 Issue 4, pp. 449-461, DOI 10.1016/j.image.2014.02.003, April 2014, <https://doi.org/10.1016/j.image.2014.02.003>.

Благодарности

Во-первых, авторы хотят сердечно поблагодарить Marie-Jose Montpetit за постоянную помощь и поддержку Tetrys. Спасибо, Marie-Jo!

Авторы также благодарны членам рабочей группы NWCRG за многочисленные обсуждения кодирования «на лету», которые помогли завершить работу над этим документом.

Наконец, авторы хотели бы поблагодарить Colin Perkins за предоставленные комментарии и отзыв о документе.

Адреса авторов

Jonathan Detchart
ISAE-SUPAERO
BP 54032
10, avenue Edouard Belin
31055 Toulouse CEDEX 4
France
Email: jonathan.detchart@isae-supaero.fr

Emmanuel Lochin
ENAC
7, avenue Edouard Belin
31400 Toulouse
France
Email: emmanuel.lochin@enac.fr

Jerome Lacan
ISAE-SUPAERO
BP 54032
10, avenue Edouard Belin
31055 Toulouse CEDEX 4
France
Email: jerome.lacan@isae-supaero.fr

Vincent Roca
INRIA
Inovallee; Montbonnot
655, avenue de l'Europe
38334 St Ismier CEDEX
France
Email: vincent.roca@inria.fr

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru