Internet Engineering Task Force (IETF)
Request for Comments: 8303
Category: Informational
ISSN: 2070-1721

M. Welzl
University of Oslo
M. Tuexen
Muenster Univ. of Appl. Sciences
N. Khademi
University of Oslo

February 2018

On the Usage of Transport Features Provided by IETF Transport Protocols

Использование транспортных свойств, предоставляемых транспортными протоколами IETF

Аннотация

В этом документе описано, как транспортные протоколы TCP[1], MPTCP[2], SCTP[3], UDP[4] и UDP-Lite[5] раскрывают свои услуги приложениям и как приложение может настроить и использовать функции, составляющие эти услуги. Рассматриваются также услуги, предоставляемые механизмом контроля перегрузок LEDBAT[6]. Описание задает набор транспортных абстракций, которые можно экспортировать в API транспортных служб (TAPS).

Статус документа

Документ не содержит какой-либо спецификации (Internet Standards Track) и публикуется с информационными целями.

Документ является результатом работы IETF[7] и представляет согласованный взгляд сообщества IETF. Документ прошел открытое обсуждение и был одобрен для публикации IESG[8]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 7841.

Информация о текущем статусе документа, найденных ошибках и способах обратной связи доступна по ссылке https://www.rfc-editor.org/info/rfc8303.

Оглавление

[1]Transmission Control Protocol - протокол управления передачей.
[2]MultiPath TCP - TCP с множеством путей.
[3]Stream Control Transmission Protocol - протокол управления передачей потоков.
[4]User Datagram Protocol - протокол пользовательских дейтаграмм.
[5]Lightweight User Datagram Protocol - облегченный протокол пользовательских дейтаграмм.
[6]Low Extra Delay Background Transport - базовый транспорт с очень низкими задержками.
[7]Internet Engineering Task Force.
[8]Internet Engineering Steering Group.

1. Введение

Эта спецификация описывает, как транспортные протоколы предлагают свои услуги приложениям, чтобы те могли использовать их без явной привязки к конкретному протоколу. Разрыв этой жесткой привязки может снизить нагрузку на разработчиков приложений и обеспечить транспортную гибкость за счет переноса сложностей в базовую транспортную систему.

Процесс разработки начался с обзора услуг, предоставляемых транспортными протоколами IETF и механизмами контроля перегрузок [RFC8095]. Данный документ вместе с [RFC8304] дополняют указанный обзор подробным описанием взаимодействия приложений с транспортными протоколами TCP, MPTCP, SCTP, UDP и UDP-Lite. Определены также примитивы для включения, выключения и настройки механизма контроля перегрузок для индивидуального (unicast) трафика LEDBAT. Для UDP и UDP-Lite первым шагом анализа служит обсуждение текста связанных с ними RFC, приведенное в [RFC8304].

Это «временной срез» транспортных протоколов IETF опубликован как RFC для документирования анализа авторами и членами рабочей группы TAPS. Он содержит набор транспортных абстракций, которые можно экспортировать в TAPS API. Документ обеспечивает базу для определения минимального набора транспортных служб, которые следует реализовать конечной системе, поддерживающей TAPS [TAPS-MINSET].

Список примитивов, событий и транспортных функций в этом документе строго основан на тех частях спецификаций протоколов, которые описывают, что протокол предоставляет использующему его приложению и как он взаимодействует с приложением. Транспортные протоколы обеспечивают коммуникации между процессами, работающими на конечных точках сети, что означает возможность мультиплексирования коммуникаций между теми же адресами IP и это мультиплексирование выполняется по номерам портов. Поэтому мультиплексирование по портам предполагается обеспечиваемым всегда и в документе не обсуждается.

Те части протоколов, которые явно означены необязательными, не рассматриваются здесь. Не обсуждаются также взаимодействия между приложением и транспортным протоколом, не связанные напрямую с работой протокола. Например, приложение может разными способами использовать опции сокета для индикации своей заинтересованности в получении тех или иных уведомлений [RFC6458]. Однако для идентификации примитивов, событий и транспортных функций способность включать и отключать получение уведомлений интереса не представляет. Точно так же, сокеты в форме «один со многими» [RFC6458] просто влияют на стиль программирования приложений и не рассматриваются здесь. То же самое относится и к способности получить неизменное значение параметра, ранее установленного приложением (например, с помощью операции get [RFC6458]).

Этот документ представляет трехэтапный процесс получения списка транспортных функций. На первом этапе обсуждается текст соответствующих RFC по протоколам. На втором этапе предыдущее обсуждение служит для выведения списка примитивов и событий, которые единообразно классифицируются по протоколам. Здесь предпринята попытка представить или (при отсутствии теста, представляющего примитивы и события) сконструировать примитивы и события в несколько обобщенной форме для выделений сходства. Это достигается, например, переименованием протокольных примитивов и событий или исключения строго сопоставления (1:1) между примитивами и событиями в спецификации протокола и списке. Финальный этап 3 представляет транспортные функции (свойства) на основе этапа 2, указывая реализующие их протоколы.

В списке, полученном на этапе 2, некоторые транспортные функции отсутствуют, поскольку они являются неявными в некоторых протоколах и становятся явными лишь при рассмотрении множества транспортных функций, предоставляемых всеми протоколами. Например, TCP всегда поддерживает контроль перегрузок, но нужно рассмотреть его вместе с такими протоколами как UDP (нет контроля перегрузок) до того, как включить контроль перегрузок в число транспортных функций. Поэтому полный список транспортных свойств для всех протоколов доступен лишь после этапа 3.

Некоторые протоколы ориентированы на соединения и в них часто применяется начальный вызов определенного примитива для организации соединения до начала обмена данными и требуется явный разрыв соединения путем вызова другого примитива (обычно он называется Close). Соединение - это общее состояние, к которому относятся некоторые транспортные примитивы, например, настройка общих параметров конфигурации. Организация, поддержка и разрыв соединений поэтому служат для классификации транспортных примитивов ориентированных на соединения протоколов на этапах 2 и 3. Поэтому предполагается, что UDP используется с «подключенными» сокетами, т. е. сокетами, которые привязаны к конкретной паре адресов и портов [RFC8304].

2. Терминология

Transport Feature - транспортная функция (свойство)

Сквозная функция обеспечиваемая приложению транспортным уровнем. Примерами являются защита конфиденциальности, надежная или упорядоченная доставка, работа с потоком или сообщениями и т. п.

Transport Service - транспортный сервис (служба)

Набор транспортных функций, обеспечивающих приложению полное обслуживание, без привязки к протоколу кадрирования.

Transport Protocol - транспортный протокол

Реализация, обеспечивающая одну или несколько транспортных служб с использованием конкретного формата кадрирования и заголовков в линии передачи (проводе).

Transport Protocol Component - компонент транспортного протокола

Реализация транспортной функции в протоколе.

Transport Service Instance - экземпляр транспортного сервиса
Набор транспортных протоколов и выбранными свойствами и параметрами конфигурации, реализующий один транспортный сервис, например, стек протоколов RTP на основе UDP.

Application - приложение
Сущность (элемент), использующая интерфейс транспортного уровня для сквозной доставки данных через сеть (возможно, вышележащий протокол или туннель).

End точка - конечная точка
Сущность (элемент), взаимодействующая с одной или несколькими другими конечными точками с помощью протокола транспортного уровня.

Connection - соединение
Общее состояние двух или более конечных точек, сохраняемое в сообщениях между этими точками.

Primitive - примитив
Функция, вызов которой служит для взаимодействия между приложением и транспортной конечной точкой. Примитив связан с одной или несколькими транспортными функциями.

Event - событие
Примитив, вызываемый транспортной конечной точкой.

Parameter - параметр
Значение, передаваемое примитивом между приложением и транспортным протоколом.

Socket - сокет
Комбинация IP-адреса и номера порта у получателя.

Transport Address - транспортный адрес
Комбинация адреса IP, транспортного протокола и номера порта, используемая транспортным протоколом.

3. Этап 1

Эта первая итерация служит обзором текстов RFC, описывающих протоколы, с акцентом на том, что каждый транспортный протокол предоставляет приложениям и как это используется (описания абстрактных API, когда это возможно). Представление примитивов, событий и параметров использует символы верхнего и нижнего регистра для удобочитаемости.

3.1. Примитивы, предоставляемые TCP

В исходной спецификации TCP [RFC0793] сказано:

> Протокол TCP предназначен для надежной и гарантированной доставки данных между хостами в компьютерных сетях с коммутацией пакетов и между такими сетями через промежуточные системы.

В параграфе 3.8 [RFC0793] определено взаимодействие с приложением через несколько транспортных примитивов. Предполагается также, что операционная система обеспечивает для TCP возможности асинхронной передачи сигналов приложению. Примитивы, представляющие такие сигналы, в этом разделе называются событиями (event).

Open
Вызов может быть активным для организации соединения или пассивным для прослушивания входящих вызовов. Все другие примитивы связаны с конкретным соединением, которое предполагается созданным заранее. Активный вызов open содержит сокет, пассивный вызов с сокетом ожидает входящего соединения. Возможно при пассивном вызове не задавать сокет для восприятия любых входящих соединений. Полностью заданный пассивный вызов можно перевести в активное состояние с помощью Send. Можно при желании указать тайм-аут, по истечении которого TCP будет разрывать соединение, если через него получателю не были доставлены данные (если тайм-аут не задан, используется принятый по умолчанию). Процедура разрыва соединения служит для предотвращения избыточных повторов передачи и приложение может контролировать порог, определяющий условия разрыва. Порог может указываться временем ожидания или числом повторов передачи [RFC1122]. Тайм-аут также можно указать числом повторов.
Для многодомных хостов можно указать локальный адрес IP [RFC1122]. Если адрес не задан, при активном вызове будет использован принятый по умолчанию. Пассивный вызов будет ждать входящих соединений по всем локальным адресам и использовать адрес IP, на который поступил вызов. Параметр options позволяет приложению указать опции IP, такие как Source Route, Record Route, Timestamp [RFC1122]. Не указано, к каким сегментам соединения этот параметр следует применять (вероятно, ко всем, поскольку это задано для опции IP Source Route в параграфе 4.2.3.8 в [RFC1122]). Опция Source Route является единственной опцией IP в этом параметре, которая не указана не обязательной (non-optional), и позволяет приложению указать заданный отправителем маршрут при активной организации соединения TCP.
Кортежи первичных ключей (Master Key Tuple или MKT) для аутентификации можно задать при вызове Open (параграф 7.1 в [RFC5925]). При использовании аутентификации защищаются полные сегменты TCP, включая псевдозаголовок IPv4, заголовок и данные TCP.
TCP Fast Open (TFO) [RFC7413] позволяет приложению незамедлительно передать сообщение при активном вызове open на пассивную сторону соединения TCP вместе с первым пакетом организации соединения (SYN). Это может быть полезно для приложений, чувствительных к задержке при организации соединения TCP. В [RFC7413] сказано: «реализациям TCP **недопустимо** использовать TFO по умолчанию и можно применять TFO лишь при явном запросе приложения или настройке на уровне сервисного порта». Размер сообщения, передаваемого с TFO, не может быть больше максимального сегмента TCP (за вычетом опций, применяемых в SYN). Для активной стороны рекомендуется изменить или заменить вызов () для поддержки аргумента, задающего пользовательский буфер [RFC7413]. Для пассивной стороны приложение должно включить прием запросов Fast Open, например, через новую опцию сокета TCP_FASTOPEN setsockopt() перед listen(). Принимающее приложение должно быть готово воспринимать дубликаты сообщения TFO, поскольку первые данные, записанные в сокет, могут доставляться приложению на удаленном хосте в нескольких экземплярах.

Send
Этот примитив приложение использует для передачи локальной транспортной конечной точке TCP числа байтов, которые протоколу TCP нужно гарантированно передать на другую сторону соединения. Флаг важности (urgent) при его установке говорит, что данные этого вызова являются срочными (важными) и эту важность следует указать принимающему процессу, если он еще не воспринял всех полученных данных без такого флага. Необязательный параметр может задавать тайм-аут для соединения (см. Open). Кроме того, необязательные параметры позволяют

указать предпочтительный исходящий MKT (current_key) и/или предпочтительный входящий MKT (rnext_key) для соединения (параграф 7.1 в [RFC5925]).

Receive

Этот примитив выделяет приемный буфер для представленного числа байтов. Примитив возвращает число принятых байтов, сообщенное буфером, когда они были приняты и записаны в буфер протоколом TCP. Приложение информируется о важных данных через флаг urgent, устанавливаемый при получении важных (срочных) данных. Сброшенный флаг говорит об отсутствии важных данных, т. е. они не были получены или данный вызов Receive вернул все срочные данные. Приложению также сообщается current_key и rnext_key из недавно принятого сегмента через необязательный параметр (параграф 7.1 в [RFC5925]).

Close

Этот примитив закрывает соединение с одной стороны. Семантически это говорит об отсутствии данных для передачи, но не закрывает прием поступающих данных, которые могут еще быть у другой стороны. Этот вызов гарантированно доставляет все уже полученные данные TCP (при отказе Close превращается в abort).

Abort

Этот примитив прерывает все ожидающие Send и Receive. Конечной точке TCP на другой стороне соединения передается сообщение TCP RESET [RFC0793].

Событие Close

TCP использует этот примитив для информирования приложения о вызове примитива Close приложением на удаленной стороне, чтобы локальное приложение также могло применить Close и аккуратно завершить соединение (параграф 3.5 в [RFC0793]).

Событие Abort

Когда TCP получает от партнера RESET, протокол: «уведомляет пользователя, и переходит в состояние CLOSED» (параграф 3.4 в [RFC0793]).

Событие User Timeout

Это событие выполняется при достижении заданного пользователем времени ожидания (параграф 3.9 в [RFC0793]), см. Open. Очищаются все очереди и приложение информируется о разрыве соединения по тайм-ауту.

Событие Error_Report

Это событие информирует приложение о «мягкой ошибке» (soft error), которую можно игнорировать [RFC5461], включая прием сообщения ICMP об ошибке или избыточные повторы (порог, который ниже порога прерывания), см. параграф 4.2.4.1 в [RFC1122].

Type-of-Service

В параграфе 4.2.4.2 [RFC1122] сказано: «Прикладному уровню **должна** быть обеспечена возможность задавать тип обслуживания TOS для сегментов, передаваемых через соединение». Приложению следует поддерживать возможность смены TOS в течение срока действия соединения и значение TOS следует без изменений передавать на уровень IP. Позднее поле TOS было переопределено и модель дифференцированного обслуживания (Differentiated Services или Diffserv) [RFC2475] [RFC3260] заменила это поле в заголовке IP, назначив 6 старших битов для передачи кода DSCP (Differentiated Services Code Point) [RFC2474].

Nagle

Алгоритм Nagle на некоторое время задерживает передачу данных для увеличения вероятности отправки полноразмерного сегмента (параграф 4.2.3.4 в [RFC1122]). Приложение может отключить алгоритм Nagle для отдельного соединения.

User Timeout Option

Опция пользовательского тайм-аута (User Timeout Option или UTO) [RFC5482] позволяет одной стороне соединения TCP аносировать свое значение заданного пользователем времени ожидания, чтобы друга сторона могла соответственно скорректировать свое значение. В дополнение к настройке тайм-аута (см. Send) имеется 3 переменных состояния на уровне соединения, которые приложение может настроить для работы UTO - adv_uto задает значение UTO, анонсируемое удаленному партнеру TCP (по умолчанию системное значение пользовательского тайм-аута), флаг enabled (по умолчанию false) для включения и отключения UTO на соединении и флаг changeable (по умолчанию true), определяющий возможность смены тайм-аута на основании опции UTO, полученной от удаленного партнера. Первые 2 переменных работают для приема и передачи, changeable принимает значение false, когда приложение явно задает пользовательский тайм-аут (см. Send).

Set/Get Authentication Parameters

Предпочтительный исходящий кортеж MKT (current_key) и/или предпочтительный входящий MKT (rnext_key) можно настроить для соединения. Можно извлечь информацию о текущих значениях current_key и rnext_key из последнего принятого сегмента (параграф 7.1 в [RFC5925]).

3.1.1. Исключенные примитивы и параметры

Примитиву Open можно передать информацию о предпочтении или защите (разделении - compartment) [RFC0793], но это не рассматривается здесь, поскольку в настоящее время в основно неактуально [RFC7414].

Примитив Status не включен в документ, поскольку в исходной спецификации TCP он указан как «зависящий от реализации» и сказано, что он: «может быть исключен без негативного влияния» [RFC0793]. Более того, хотя описан блок данных, содержащий конкретную информацию, сказано, что вся эта информация может быть доступна не всегда. Хотя в [RFC5925] сказано: «STATUS **следует** дополнить для обеспечения возможности читать MKT текущего или ожидающего соединения (для подтверждения)», та же самая информация доступна через примитив Receive, который в соответствии с [RFC5925] «**должен** быть дополнен» этой функциональностью. Примитив Send включает необязательный флаг push, установка которого требует немедленной передачи данных получателю [RFC0793]. Описанный там же примитив Receive может (в некоторых обстоятельствах) устанавливать флаг push. Поскольку функциональность push необязательная для примитивов Send и Receive [RFC1122], она не рассматривается здесь. В [RFC1122] введены сообщения keep-alive для TCP, но они не обязательны для реализации и не включены в документ. Там же сказано: «Некоторые реализации TCP включают вызовы FLUSH», что говорит о необязательности данного вызова и он не рассматривается здесь.

3.2. Примитивы, предоставляемые MPTCP

MPTCP является расширением TCP, позволяющим применять несколько путей для одного потока данных. Это достигается созданием так называемых субпотоков TCP для каждого из интерфейсов и планированием трафика через эти субпотоки. Сервис, предоставляемый MPTCP описан в [RFC6182]:

> Протокол Multipath TCP **должен** следовать модели сервиса, используемой TCP [1] - упорядоченная, надежная, ориентированная на байты доставка. Кроме того, соединению Multipath TCP **следует** предоставлять приложению пропускную способность не хуже ожидаемой при работе через одно соединение TCP по любому из доступных путей.

Кроме того, имеются некоторые ограничения на API, раскрываемый MPTCP, как отмечено в [RFC6182]:

> Поддерживающий множество путей эквивалент TCP **должен** сохранять некоторый уровень совместимости с имеющимися TCP API, чтобы существующие приложения могли использовать новый транспорт, просто обновив операционные системы конечных хостов.

Таким образом, предоставляемые MPTCP примитивы эквивалентны примитивам TCP. Тем не менее, в MPTCP RFC [RFC6824] и [RFC6897] разъяснены некоторые детали примитивов TCP в отношении MPTCP и добавлены расширения для лучшего управления субпотоками MPTCP. Ниже приведен список уточнений и расширений, которые упомянутые RFC содержат для примитивов TCP.

Open

> «Приложению следует иметь возможность включать и выключать применение MPTCP» [RFC6897]. Эта функциональность может быть обеспечена опцией сокета tcp_multipath_enable. Кроме того протокол MPTCP должен отключаться, если приложение привязано к конкретному адресу [RFC6897].

Send/Receive

> Передача и прием данных не требуют менять приложение для использования MPTCP [RFC6824]. Уровень MPTCP принимает от приложения один поток данных и делит его на субпотоки с достаточным объемом данных управления, позволяющих гарантированно доставить данные с сохранением порядка и собрать их на приемной стороне.
>
> Использование указателя важности (Urgent Pointer) отличается MPTCP и в [RFC6824] сказано: «Субпотоку TCP **недопустимо** использовать Urgent Pointer для прерывания имеющегося сопоставления».

Управление адресами и субпотоками

> MPTCP использует разные адреса и позволяет хосту анонсировать их как часть протокола. В [RFC6897] сказано: «Приложению следует обеспечивать возможность ограничить MPTCP привязкой к данному набору адресов», что позволяет приложению указать ограниченный набор адресов для использования протоколом MPTCP. Там же сказано: «Приложению следует иметь возможность получить список пар адресов, используемых субпотоками MPTCP».

3.3. Примитивы, предоставляемые SCTP

TCP имеет множество ограничений, которые устранены в SCTP (параграф 1.1 в [RFC4960]). Три снятых ограничения напрямую отражаются в транспортных возможностях, которые видит использующее SCTP приложение - 1) возможность сохранить разграничители сообщений, 2) отсутствие гарантий доставки и сохранения порядка без запроса приложения, 3) поддержка многодомности. В SCTP соединения называются ассоциациями (association) и могут создаваться не только между парой (как в TCP), но и между множеством адресов на каждой конечной точке.

Раздел 10 базовой спецификации SCTP [RFC4960] задает взаимодействие с приложением, которое в SCTP называется протоколом вышележащего уровня (Upper-Layer Protocol или ULP). Предполагается, что операционная система обеспечивает SCTP возможность асинхронной передачи сигналов приложению, примитивы такой сигнализации называются здесь событиями (event) и описаны ниже. В дополнение к абстрактному API, заданному в разделе 10 [RFC4960], были описаны расширения для API сокетов [RFC6458], включающие функциональность базового протокола [RFC4960] и некоторые расширения [RFC3758] [RFC4895] [RFC5061]. Для других расширений протокола ([RFC6525] [RFC6951] [RFC7053] [RFC7496] [RFC7829] [RFC8260]) соответствующие расширения API сокетов описаны в спецификациях. Функциональность, раскрываемая ULP через все эти API описана здесь.

Абстрактный API содержит примитив SetProtocolParameters, позволяющий настраивать элементы списка параметров [RFC4960]. В спецификации протокола сказано: «Реализация SCTP может разрешать ULP изменение некоторых параметров протокола». Это указывает, что ни один из элементов этого списка параметров не является обязательным для настройки ULP. Поэтому здесь рассматриваются лишь параметры в абстрактном API, которые также указаны в одном из других RFC, упомянутых выше, что привело к исключению параметров RTO.Alpha, RTO.Beta, HB.Max.Burst. Для четкости сам примитив SetProtocolParameters здесь заменен примитивами настройки параметров или групп параметров.

Initialize

> Примитив Initialize создает локальный экземпляр SCTP, связывая его с набором локальных адресов и номером порта (если он задан) [RFC4960]. Примитив нужно вызывать лишь один раз для набора локальных адресов. При создании ассоциации может использоваться множество параметров для нее, это нужно сделать до соединения (с помощью описанного ниже примитива Associate). Можно указать максимальное число входных потоков, которые приложение готово поддерживать, максимальное число попыток передачи INIT (первое сообщение при создании ассоциации) и максимальный тайм-аут повторной передачи (maximum retransmission timeout - RTO) для попыток INIT [RFC6458]. С этого момента (до соединения) приложение может также включить инкапсуляцию UDP путем настройки номера удаленного порта для нее [RFC6951].

Associate

> Создает ассоциацию (эквивалент SCTP для соединения), связывающую локальный и удаленный экземпляр SCTP. Для идентификации удаленной конечной точки можно ей может быть назначен один или несколько (с помощью connectx) сокетов (параграф 9.9 в [RFC6458]). Большинство примитивов связано с конкретной ассоциацией, которая предполагается созданной первой. Associate может возвращать список транспортных адресов получателя, что позволяет использовать затем множество путей. Один из возвращенных сокетов выбирается локальной конечной точкой как используемый по умолчанию основной путь для передачи партнеру пакетов SCTP, но этот выбор приложение может изменить, используя список адресов получателя. Associate также получает число исходящих потоков для запроса и может возвращать число согласованных исходящих потоков. Может быть представлен необязательный 32-битовый параметр индикации уровня адаптации [RFC5061]. При использовании аутентифицированных блоков (chunk) могут быть представлены типы блоков, которые требуется передавать с проверкой подлинности [RFC4895]. Уведомление SCTP_Cant_Str_Assoc служит для информирования приложения об отказе при создании ассоциации [RFC6458]. Приложение может использовать sendto() или sendmsg() для

неявного создания ассоциации, передавая тем самым сообщение, что SCTP может передавать на этапе создания ассоциации [RFC6458]. Отметим, что этот механизм отличается от TCP TFO и сообщение будет приходить лишь один раз по истечении по меньшей мере одного интервала RTT, поскольку передается вместе с третьим сообщением организации ассоциации (блок COOKIE-ECHO).

Send

Примитив передает через ассоциацию сообщение с неким числом байтов. Сообщению может быть назначен номер для последующего обращения к корректному сообщению при возникновении ошибки, а также предоставляется идентификатор для указания потока, используемого внутри ассоциации (этот параметр для простоты считается обязательным, а при его отсутствии просто используется 0). Могут быть заданы условия отказа от сообщения (например, ограничение числа повторных попыток передачи или срока действия пользовательского сообщения). Это позволяет управлять расширением частичных гарантий [RFC3758] [RFC7496]. Необязательный срок действия сообщения позволяет отбросить устаревшее сообщение без его отправки. Можно указать предпочтительный путь (выбор его не гарантирован) путем задания сокета, а также возможность неупорядоченной доставки с помощью флага unordered. Рекомендательный флаг указывает, что партнеру не следует задерживать подтверждение пользовательского сообщения [RFC7053], а другой рекомендательный флаг управляет предпочтениями приложения в части группировки данных пользователя с другими исходящими блоками DATA в один пакет. Идентификатор протокола для данных (payload) может предоставляться для партнеру передачи значения, указывающего тип данных. При использовании аутентификации блоков может предоставляться идентификатор ключа для проверки подлинности блоков DATA [RFC4895].

Receive

Принимает из ассоциации сообщения и, возможно, поток внутри ассоциации с возвратом его размера. Приложение узнает о доступности данный с помощью уведомления Data Arrive. Если отправитель включил идентификатор протокола для данных, примитив возвращает его. Если принятое сообщение является лишь частью полного, это будет указано флагом partial и в этом случае приложению предоставляются идентификатор и порядковый номер потока.

Shutdown

Примитив аккуратно закрывает ассоциацию, гарантированно доставляя данные, которые уже переданы SCTP. Параметр предоставляет приложению контроль над операциями дальнейшего приема или отправки. Код возврата информирует приложение об выполнении или отказе процедуры.

Abort

Примитив закрывает ассоциацию без соблюдения аккуратности, отбрасывая данные из локальных очередей и информируя партнера о разрыве ассоциации. Приложение может указать причину разрыва для передачи партнеру. Код возврата указывает успех или неудачу процедуры.

Change Heartbeat/Request Heartbeat

Эти примитивы позволяют приложению включить или выключить сообщения heartbeat, а также задать частоту их передачи или запросить передачу одного сообщения heartbeat при вызове функции с уведомлением получателя о результате (успех или неудача) передачи блока HEARTBEAT.

Configure Max. Retransmissions of an Association

Параметр Association.Max.Retrans [RFC4960] (sasoc_maxrxt расширении API сокетов [RFC6458]) позволяет задать число неудачных повторов, после которого принимается решение об отказе ассоциации в целом, кода следует вызывать уведомление Communication Lost.

Set Primary

Позволяет изменить принятый по умолчанию путь для ассоциации предоставлением сокета. Можно также указать используемый по умолчанию адрес для дейтаграмм IP.

Change Local Address/Set Peer Primary

Позволяет конечной точке добавлять и удалять локальные адреса для ассоциации. Кроме того, партнеру можно посоветовать адрес для использования в качестве первичного [RFC5061].

Configure Path Switchover

Этот абстрактный API содержит примитив Set Failure Threshold [RFC4960], задающий параметр Path.Max.Retrans, который определяет число повторов, после которого определенный транспортный адрес считается недоступным. Если в ассоциации еще остаются транспортные адреса, достижение этого предела ведет к переключению пути. Расширение SCTP-PF добавляет в этот метод концепцию «ненадежных путей» (Potentially Failed или PF) [RFC7829]. SCTP не отказывается полностью от передачи по путям со статусом PF, но будет предпочитать другие активные пути, если они доступны. Переход в состояние PF происходит при превышении заданного максимума повторов передачи. Таким образом, для всех путей, гже применяется этот механизм, имеется два настраиваемых порога ошибок, один из которых определяет переход в состояние PF, а другой - принятие решения о недоступности адреса.

Set/Get Authentication Parameters

Позволяет приложению добавлять и удалять ключевой материал для ассоциации. Кроме того, можно запросить типы блоков, которые нужно аутентифицировать [RFC4895].

Add/Reset Streams, Reset Association

Позволяет конечной точке добавлять потоки в имеющуюся ассоциацию или сбрасывать потоки индивидуально. Кроме того, можно сбросить ассоциацию [RFC6525].

Status

Примитив возвращает блок данных с информацией об указанной ассоциации, списком транспортных адресов получателей, состояниях доступности этих транспортных адресов, текущих размерах локального и удаленного окна, текущем размере локального окна перегрузок, числе неподтвержденных блоков DATA; основном пути, последнем сглаженном времени кругового обхода (Smoothed Round-Trip Time или SRTT) на основном пути, RTO на основном пути, SRTT и RTO для других адресов получателя [RFC4960] и MTU на путях [RFC6458].

Enable/Disable Interleaving

Позволяет включить или выключить согласование поддержки чередования пользовательских сообщений для будущих ассоциаций. Для имеющихся ассоциаций можно запросить, было ли согласовано чередование пользовательских сообщений в конкретной ассоциации [RFC8260].

Set Stream Scheduler

Позволяет выбрать планировщик потоков для ассоциации из числа First-Come-First-Served (обслуживание в порядке очередности поступления), Round-Robin (циклический перебор), Round-Robin per Packet (циклический

перебор на уровне пакетов), Priority-Based (по приоритету), Fair Bandwidth (беспристрастное распределение пропускной способности, Weighted Fair Queuing (взвешенная беспристрастная очередь) [RFC8260].

Configure Stream Scheduler

Позволяет менять параметры планировщика на уровне потока - приоритет для планировщика Priority-Based и вес для Weighted Fair Queuing.

Enable/Disable NoDelay

Включает и отключает использование любого алгоритма в стиле Nagle для ассоциации [RFC6458].

Configure Send Buffer Size

Управляет объемом данных, которые могут ожидать передачи (включая повтор) во внутренних буферах SCTP [RFC6458].

Configure Receive Buffer Size

Задает размер приемного буфера в октетах, управляя тем самым окном приема для ассоциации [RFC6458].

Configure Message Fragmentation

Если пользовательское сообщение создает пакет SCTP, размер которого превышает допустимый для передачи (задается приложением или определяется Path MTU), протокол SCTP фрагментирует его. Запрет фрагментации в таких случаях будет приводить к ошибке вместо фрагментирования сообщения [RFC6458].

Configure Path MTU Discovery

Механизм Path MTU Discovery (PMTUD) можно включить и выключить для каждого адреса партнера в ассоциации (параграф 8.1.12 в [RFC6458]). При включенном механизме определяется текущее значение Path MTU, при выключенном значение Path MTU контролируется приложением.

Configure Delayed SACK Timer

Время задержки отправки SACK может быть настроено, вплоть до запрета задержки. Можно задать также число пакетов, которые должны быть получены перед отправкой SACK без ожидания таймера задержки [RFC6458].

Set Cookie Life Value

Значение Cookie life может быть настроено (параграф 8.1.2 в [RFC6458]). Valid.Cookie.Life является одним из параметров, которые могут настраиваться с помощью SetProtocolParameters [RFC4960].

Set Maximum Burst

Максимальный пик пакетов, которые могут быть отправлены конкретной ассоциацией (по умолчанию 4 и большие значения поддерживать не требуется), см. параграф 8.1.2 в [RFC6458]). Max.Burst является одним из параметров, которые могут настраиваться с помощью SetProtocolParameters [RFC4960].

Configure RTO Calculation

Абстрактный API содержащий настраиваемые параметры RTO.Initial, RTO.Min, RTO.Max, RTO.Alpha. RTO.Beta. Лишь начальное, минимальное и максимальное значение RTO указаны как настраиваемые в расширении API сокетов SCTP [RFC6458].

Set DSCP Value

Значение DSCP можно установить для адреса партнера в ассоциации (параграф 8.1.12 в [RFC6458]).

Set IPv6 Flow Label

Метку потока можно установить для адреса партнера в ассоциации (параграф 8.1.12 в [RFC6458]).

Set Partial Delivery Point

Позволяет задать размер сообщения, при котором будет использоваться частичная доставка. Снижение порога приводит к более частому применению частичной доставки [RFC6458].

Уведомление Communication Up

При восстановлении потерянной связи и обретении SCTP готовности к приему или передаче пользовательских сообщений это уведомление информирует прикладной процесс о затронутой ассоциации, типе события, полном наборе сокетов партнера и числе входных потоков (число потоков, запрошенных партнером). При поддержке чередования на обеих сторонах информация об этом также включается в уведомление.

Уведомление Restart

Передается на вышележащий уровень, когда конечная точка SCTP детектирует перезапуск партнера [RFC6458].

Data Arrive

Информирует приложение о готовности к приему сообщения с помощью примитива Receive.

Уведомление Send Failure Notification/Receive Unsent Message/Receive Unacknowledged Message

При невозможности доставить сообщение через ассоциацию отправителя можно информировать об этом и он узнает, было ли сообщение просто не подтверждено или не было доставлено (например, по истечении срока жизни). Это может также информировать отправителя о доставке части сообщения.

Уведомление Network Status Change

Информирует приложение о смене состояния активности сокета [RFC4960] или состоянии PF [RFC7829].

Уведомление Communication Lost

Когда SCTP теряет связь с конечной точкой (определяется по heartbeat или числу повторов передачи) или узнает о прерывании, это уведомление информирует прикладной процесс о затронутой ассоциации, типе события (отказ или прерывание в ответ на запрос shutdown или abort).

Уведомление Shutdown Complete

Когда SCTP завершает процедуры shutdown, это уведомление передается на вышележащий уровень для информирования о затронутой ассоциации.

Уведомление Authentication

Передается на вышележащий уровень, когда SCTP хочет уведомить его об управлении ключами, связанном с аутентифицируемыми блоками [RFC4895].

Уведомление Adaptation Layer Indication

Передается на вышележащий уровень, когда SCTP завершает создание ассоциации и партнер узнает уровень адаптации [RFC5061] [RFC6458].

Уведомление Stream Reset

Информирует вышележащий уровень о результате, когда SCTP завершает процедуры сброса потоков [RFC6525].

Уведомление Association Reset

Информирует вышележащий уровень о результате, когда SCTP завершает процедуру сброса [RFC6525],.

Уведомление Stream Change

Информирует вышележащий уровень о результате, когда SCTP завершает процедуру, используемую для увеличения числа потоков [RFC6525].

Уведомление Sender Dry

Передается на вышележащий уровень, когда SCTP больше нет пользовательских данных для передачи или повтора [RFC6458].

Уведомление Partial Delivery Aborted

Передается на вышележащий уровень, когда получатель начал принимать части пользовательского сообщения, но доставка сообщения затем была прервана (параграф 6.1.7 of [RFC6458]).

3.3.1. Исключенные примитивы и параметры

Примитив Receive может возвращать дополнительную информацию, но она не обязательна для реализации и не рассматривается здесь. С уведомлением Communication Lost также может предоставляться дополнительная информация (например, идентификация для извлечения не отправленных или не подтвержденных данных). SCTP «может вызывать» уведомление Communication Error и «передавать» уведомление Restart, которые не обязательны для реализации. Список, для примитива Status включает «и т. п.», что говорит о дополнительной информации. Примитив Get SRTT Report возвращает информацию, предоставляемую примитивом Status, поэтому здесь не рассматривается. Функция API Destroy SCTP Instance была исключена - она удаляет экземпляр SCTP, созданный Initialize, но не является примитивом в смысле данного документа, поскольку не относится к транспортным свойствам. Событие Shutdown информирует приложение о передаче партнером SHUTDOWN, поэтому данные больше не следует передавать в сокет (параграф 6.1 в [RFC6458]). Однако при попытке приложения передать данные в сокет оно будет получать сообщение об ошибке, поэтому данное событие классифицировано лишь как воздействие на стиль программирования приложений и не включено в этот документ.

3.4. Примитивы, предоставляемые UDP и UDP-Lite

Набор примитивов этапа 1 для UDP и UDP-Lite приведен в [RFC8304].

3.5. Служба LEDBAT

Описание сервиса механизма контроля перегрузок LEDBAT приведено ниже.

> LEDBAT предназначен для использования фоновыми процессами с передачей больших объемов данных и быть не более активным, нежели контроль перегрузок в стандартном TCP (RFC 5681), уступая в присутствии конкурирующих потоков, что ограничивает негативное влияние одновременных потоков на производительность сети [RFC6817].

LEDBAT не имеет своих примитивов и не является транспортным протоколом. В соответствии с [RFC6817]:

> LEDBAT может применяться как часть транспортного протокола или приложения, пока механизмы передачи данных способны доставлять временные метки и подтверждения достаточно часто. LEDBAT можно применять с TCP, SCTP и DCCP[1]) с соответствующими расширениями, а также с фирменными протоколами, такими как работающие на основе одноранговых (P2P[2]) приложений UDP.

На момент написания этого документа расширений для TCP, SCTP или DCCP еще не было.

В спецификации LEDBAT имеется много настраиваемых параметров. Параметр TARGET, задающий целевое значение задержки, при которой LEDBAT пытается работать, должен иметь значение не более 100 мсек. Параметр allowed_increase (следует устанавливать 1 и должно быть больше 0) ограничивает скорость, с которой LEDBAT повышает свою скорость работы. Параметр gain, который в соответствии с [RFC6817] «**должен** быть не больше 1» для предотвращения более быстрого роста скорости, чем TCP Reno, определяет, как быстро отправитель реагирует на изменение задержки в очередях. Реализации могут делить gain на два параметра, один для роста, другой (возможно, больший) для снижения. Здесь эти параметры названы Gain_Inc и Gain_Dec. Base_History указывает размер списка измеренных базовых задержек и в соответствии с [RFC6817] для него «**следует** устанавливать значение 10». Этот список может фильтроваться с помощью функции Filter, которая не задана в [RFC6817], но дает список размером Current_Filter. Для начального и минимального размера окна (Init_CWND и Min_CWND) следует задавать значение 2.

Что касается контролируемых приложением параметров, как крайний вариант они могут не раскрываться совсем, а в качестве другой крайности, все, что не указано в спецификации как **требуемое**, может считаться параметром. Реализации функций не предоставляются в качестве параметров для какого-либо из рассматриваемых здесь транспортных протоколов, поэтому функция Filter не считается параметром. Однако для предотвращения ненужных ограничений в будущих реализациях все остальные параметры, упомянутые выше, считаются настраиваемыми параметрами, которые следует раскрывать.

4. Этап 2

Здесь классифицируются примитивы, выбранные на этапе 1 в зависимости от того, относятся они к организации соединения или передаче данных. Примитивы представлены в соответствии с номенклатурой «Категория.[Субкатегория].Свойство.Протокол» (CATEGORY.[SUBCATEGORY].FEATURENAME.PROTOCOL). Категориями могут быть соединения (CONNECTION) и данные (DATA). В категории соединений могут рассматриваться подкатегории организации (ESTABLISHMENT), доступности (AVAILABILITY), поддержки (MAINTENANCE) и прерывания (TERMINATION). В категории данных подкатегорий не задано. Имя протокола UDP(-Lite) используется в примитивах, которые эквивалентны для UDP и UDP-Lite, TCP - в примитивах, эквивалентных для TCP и MPTCP. Соединения представлены как базовая концепция, независимая от протокола и служащая для обозначения, например, соединений TCP (указываются уникальными парами адресов IP и портов TCP), ассоциаций SCTP (указываются множеством пар адресов IP и номеров портов), а также соединения UDP и UDP-Lite (указываются уникальной парой сокетов).

Для общности некоторые мелкие детали не рассматриваются, например, Close в SCTP [RFC4960] возвращает успех или отказ и позволяет приложению контролировать разрешение или запрет последующих операций приема и передачи [RFC6458]. Это не описывается так же для TCP [RFC0793], но эти детали не играют важной роли для примитивов, предоставляемых TCP или SCTP (для общности можно предположить запрет операций приема и передачи в обоих случаях).

[1]Datagram Congestion Control Protocol - протокол контроля перегрузок для дейтаграмм.
[2]Peer-to-peer - «равный с равным».

Примитивы TCP Send и Receive включают использование параметра urgent, который контролирует механизм, требуемый для реализации «сигналов синхронизации», применяемых в telnet [RFC0854], но в [RFC6093] сказано: «новым приложениям **не следует** поддерживать механизм TCP». Поскольку этап 2 создает основу для будущих систем, механизм urgent здесь исключен. Это отноится и к уведомлению Urgent Pointer Advance в Error_Report (параграф 4.2.4.1 в [RFC1122]).

Поскольку LEDBAT является механизмом контроля насыщения, а не протоколом, в настоящее время не задано, когда следует включать и отключать или настраивать этот механизм. Например, это может быть однократным выбором при организации соединения или прослушивании входящих вызовов и в этих случаях его следует отнести к категории CONNECTION.ESTABLISHMENT или CONNECTION.AVAILABILITY, соответственно. Для предотвращения ненужных ограничений в будущих реализациях принято решение поместить его в категорию CONNECTION.MAINTENANCE с параметрами, описанными в [RFC6817], сделав все параметры настраиваемыми.

4.1. Примитивы, связанные с соединением

Организация

Активная организация соединения одной транспортной конечной точки с одной или множеством транспортных конечных точек. Интерфейсы UDP и UDP-Lite позволяют использовать API на основе соединений и без них [RFC8085].

CONNECT.TCP

Событие или примитив этапа 1: Open (активно) или Open (пассивно) с сокетом, затем Send.
Параметры: 1 локальный адрес IP (необязательно), 1 транспортный адрес получателя (для активного open, иначе сокет и локальный адрес IP последующего входящего запроса на соединение), тайм-аут (необязательно), опции (необязательно), конфигурация MKT (необязательно), пользовательское сообщение (необязательно).
Комментарии: Если локальный адрес IP не указан, автоматически выбирается принятый по умолчанию. Тайм-аут может быть задан числом попыток повтора. Опциями являются опции IP для использования во всех сегментах соединения. Для соединения TCP требуется представлять хотя бы одну опцию Source Route. Конфигурация MKT относится к возможности настроить кортежи MKT для проверки подлинности. Пользовательское сообщение может передаваться партнерскому приложению сразу после приема пакета TCP SYN. Для снижения задержки с помощью экспериментального механизма TFO размер первого сообщения должен быть не больше максимального сегмента TCP (за вычетом опций TCP, использованных в SYN). Сообщение (первое) может быть доставлено приложению на удаленном хосте в нескольких экземплярах.

CONNECT.SCTP

Событие или примитив этапа 1: Initialize, затем Enable/Disable Interleaving (необязательно) и Associate.
Параметры: список локальных пар «порт SCTP - адрес IP» (Initialize), один или несколько сокетов (идентификация партнера), число выходных потоков, максимальное число входных потоков, уровень адаптации (необязательно), аутентифицируемые типы блоков (необязательно), запрос чередования (on/off)Ю максимальное число попыток INIT (необязательно), максимальное начальное значение RTO для INIT (необязательно), пользовательское сообщение (необязательно), номер удаленного порта UDP (необязательно).
Возврат: список сокетов или отказ.
Комментарии: Initialize нужно вызывать лишь один раз для списка локальных пар «порт SCTP - адрес IP». Автоматически выбирается один сокет, который можно потом изменить с помощью MAINTENANCE. Пользовательское сообщение может быть передано партнерскому приложению сразу после приема пакета с блоком COOKIE-ECHO. Для сокращения задержки размер первого сообщения должен ограничиваться так, чтобы оно помещалось в блок COOKIE-ECHO. Если указан удаленный порт UDP, пакеты SCTP инкапсулируются в UDP.

CONNECT.MPTCP

Похож на CONNECT.TCP, но включает 1 дополнительных логический параметр, позволяющий включать или отключать MPTCP для отдельного соединения или сокета (по умолчанию включено).

CONNECT.UDP(-Lite)

Событие или примитив этапа 1: Connect, затем Send.
Параметры: 1 локальный адрес IP (по умолчанию ANY или указанный адрес), 1 транспортный адрес получателя, 1 локальный порт (по умолчанию выбранный ОС или указанный), 1 порт получателя (по умолчанию выбранный ОС или указанный).
Комментарии: Связывает транспортный адрес, создавая соединение сокета UDP(-Lite). Может вызываться неоднократно с разными транспортными адресами для организации новых соединений. Функция CONNECT позволяет приложению получать сведения об ошибках при передаче сообщений по транспортному адресу.

Доступность

Подготовка к приему входящих запросов на соединения.

LISTEN.TCP

Событие или примитив этапа 1: Open (пассивно).
Параметры: 1 локальный адрес IP (необязательно), 1 сокет (необязательно), тайм-аут (необязательно), буфер для приема пользовательского сообщения (необязательно), конфигурация MKT (необязательно).
Комментарии: Если представлен сокет и/или локальный адрес IP, входящие соединения ожидаются лишь из указанного адреса и/или сокета, в ином случае принимаются все вызовы. Позднее можно выполнить организацию соединения (ESTABLISHMENT) с примитивом Send. Если представлен приемный буфер для пользовательского сообщения, это сообщение может быть принято от отправителя с поддержкой TFO до завершения согласования TCP. Такое сообщение может быть получено несколько раз. Конфигурация MKT указывает возможность настройки MKT для аутентификации.

LISTEN.SCTP

Событие или примитив этапа 1: Initialize, затем уведомление Communication Up или Restart и, возможно, уведомление Adaptation Layer.
Параметры: Список локальных пар «порт SCTP - адрес IP» (инициализация).
Возврат: Список сокетов, число выходных потоков, число входных потоков, уровень адаптации, типы аутентифицируемых блоков, поддержка чередования на обеих сторонах (yes/no).
Комментарии: Initialize нужно вызывать лишь один раз для списка локальных пар «порт SCTP - адрес IP». За уведомлением Communication Lost может следовать Communication Up, указывая восстановление потерянной связи. Если партнер указал уровень адаптации, выдается уведомление Adaptation Layer.

LISTEN.MPTCP

Похож на LISTEN.TCP, но включает дополнительный логический параметр, включающий или отключающий MPTCP для определенного соединения или сокета (по умолчанию включено).

LISTEN.UDP(-Lite)

Событие или примитив этапа 1: Receive.

Параметры: 1 локальный адрес IP (по умолчанию ANY или заданный), 1 транспортный адрес получателя, локальный порт (по умолчанию выбранный ОС или заданный), порт назначения (по умолчанию выбранный ОС или заданный).

Комментарии: Функция Receive регистрирует приложение, слушающее входящие дейтаграммы UDP(-Lite) на конечной точке.

Поддержка

Выполняет настройку имеющегося соединения или уведомлений о нем. Существуют передаваемые по отдельному каналу (out-of-band) сообщения для протокола, которые могут передаваться в любой момент, по крайней мере после организации соединения и до его разрыва (за исключением CHANGE_TIMEOUT.TCP, которое может передаваться лишь для открытых соединений, когда вызывается DATA.SEND.TCP). В некоторых случаях эти примитивы могут напрямую вызываться при организации (ESTABLISHMENT) или контроле доступности (AVAILABILITY) соединения без ожидания его организации (например, CHANGE_TIMEOUT.TCP можно выполнить с использованием примитива TCP Open). Для UDP и UDP-Lite эти функции могут задавать настройки на уровне соединения или отдельного сообщения.

CHANGE_TIMEOUT.TCP

Событие или примитив этапа 1: Open или Send с незаданными переменными контроля состояния соединения.

Параметры: Тайм-аут, adv_uto (необязательно); uto_enabled (необязательно, по умолчанию false), изменяемость (необязательно, по умолчанию true).

Комментарии: При передаче данных приложение может настроить тайм-аут для соединения (время, по истечении которого соединение разрывается, если данные не удается доставить). Если uto_enabled = true, значение тайм-аута (или adv_uto при наличии) будет анонсироваться TCP на другой стороне соединения для адаптации там значения пользовательского тайм-аута. Значение uto_enabled контролирует опцию UTO для соединения в обоих направлениях. Параметр changeable управляет возможностью изменять тайм-аут на основе опции UTO, принятой с другой стороны соединения, он получает значение becomes при использовании значения timeout.

CHANGE_TIMEOUT.SCTP

Событие или примитив этапа 1: Change Heartbeat вместе с Configure Max. Retransmissions of an Association.

Параметры: Change Heartbeat для частоты heartbeat и Configure Max. Retransmissions of an Association для Association.Max.Retrans.

Комментарии: Change Heartbeat может включать или выключать сообщения heartbeat в SCTP, а также менять их частоту. Параметр Association.Max.Retrans определяет число неудачных передач любых пакетов (включая heartbeat), после которого ассоциация разрывается. Таким образом, эти примитивы и параметры вместе могут задавать поведение ассоциации SCTP как CHANGE_TIMEOUT.TCP для соединений TCP.

DISABLE_NAGLE.TCP

Событие или примитив этапа 1: Не задано.

Параметры: 1 логическое значение.

Комментарии: Алгоритм Nagle задерживает передачу данных для повышения вероятности отправки полноразмерного сегмента. Приложение должно иметь возможность отключать этот алгоритм для соединения.

DISABLE_NAGLE.SCTP

Событие или примитив этапа 1: Enable/Disable NoDelay.

Параметры: 1 логическое значение.

Комментарии: Алгоритмы класса Nagle задерживают передачу данных для повышения вероятности отправки полноразмерного сегмента. Приложение должно иметь возможность отключать этот алгоритм для соединения.

REQUEST_HEARTBEAT.SCTP

Событие или примитив этапа 1: Request Heartbeat.

Параметры: Сокет.

Возврат: успех или отказ.

Комментарии: Запрашивает незамедлительное сообщение heartbeat на пути, возвращая отказ или успех.

ADD_PATH.MPTCP

Событие или примитив этапа 1: Не задано.

Параметры: Локальный адрес IP, локальный номер порта (необязательно).

Комментарии: Приложение указывает локальный адрес IP и номер порта, которые должны использоваться для субпотока.

ADD_PATH.SCTP

Событие или примитив этапа 1: Change Local Address/Set Peer Primary.

Параметры: Локальный адрес IP.

REM_PATH.MPTCP

Событие или примитив этапа 1: Не задано.

Параметры: Локальный адрес IP, локальный номер порта, удаленный адрес IP, удаленный номер порта.

Комментарии: Приложение удаляет субпоток, указанный парой «адрес IP - порт». Реализация MPTCP должна инициировать удаление субпотока, относящегося к этой паре.

REM_PATH.SCTP

Событие или примитив этапа 1: Change Local Address/Set Peer Primary.

Параметры: Локальный адрес IP.

SET_PRIMARY.SCTP

Событие или примитив этапа 1: Set Primary.

Параметры: Сокет.

Возврат: Результат предпринятой попытки.

Комментарии: Обновляет текущий первичный адрес на основе доступных в ассоциации сокетов.

SET_PEER_PRIMARY.SCTP

Событие или примитив этапа 1: Change Local Address/Set Peer Primary.

Параметры: Локальный адрес IP.

Комментарии: Это лишь рекомендация для партнера.

CONFIG_SWITCHOVER.SCTP

Событие или примитив этапа 1: Configure Path Switchover.

Параметры: Основное максимальное число повторов (после которого путь считается неактивным) и максимальное число повторов для PF (после которого путь считается ненадежным и применяются другие пути) (необязательно).

STATUS.SCTP

Событие или примитив этапа 1: Уведомления Status, Enable/Disable Interleaving и Network Status Change.

Возврат: Блок данных о конкретной ассоциации, содержащий состояние соединения, список транспортных адресов получателей и состояния их доступности, текущие размеры окна (локального и принимающего партнера), текущие размеры локального окна насыщения, число неподтвержденных блоков DATA, число блоков DATA, ожидающих приема, первичный путь, последнее значение SRTT на первичном пути, RTO на первичном пути, SRTT и RTO на других адресах получателей, MTU на каждом пути и поддержка чередования (yes/no).

Комментарии: Уведомление Network Status Change информирует приложение а смене состояния активности сокета. Оно влияет лишь на стиль программирования, поскольку эти же данные доступны через примитив Status.

STATUS.MPTCP

Событие или примитив этапа 1: Не задано.

Возврат: Список пар «адрес IP - порт TCP» каждого субпотока. Первая пара содержит локальный адрес IP и порт, вторая — удаленные.

SET_DSCP.TCP

Событие или примитив этапа 1: Не задано.

Параметры: Значение DSCP.

Комментарии: Примитив позволяет приложению изменить значение DSCP для исходящих сегментов.

SET_DSCP.SCTP

Событие или примитив этапа 1: Установка DSCP.

Параметры: Значение DSCP.

Комментарии: Примитив позволяет приложению изменить значение DSCP для исходящих пакетов на пути.

SET_DSCP.UDP(-Lite)

Событие или примитив этапа 1: Set_DSCP.

Параметры: Значение DSCP.

Комментарии: Примитив позволяет приложению изменить значение DSCP для исходящих дейтаграмм UDP(-Lite). Рекомендации по использованию поля приведены в [RFC7657] и [RFC8085].

ERROR.TCP

Событие или примитив этапа 1: Error_Report.

Возврат: Причина (кодирование не занято) и субпричина (кодирование не занято).

Комментарии: Мягкие ошибки многие приложения могут игнорировать без ущерба и следует обеспечивать возможность такого игнорирования. Ошибки включают сообщения ICMP об ошибках и избыточные повторы.

ERROR.UDP(-Lite)

Событие или примитив этапа 1: Error_Report.

Возврат: Сообщение об ошибке.

Комментарии: Примитив возвращает мягкие ошибки, которые приложения могут игнорировать без ущерба и следует обеспечивать возможность такого игнорирования.

SET_AUTH.TCP

Событие или примитив этапа 1: Не задано.

Параметры: current_key и rnext_key.

Комментарии: Ключи current_key и rnext_key указывают предпочтительный исходящий и входящий MKT, соответственно, для сегмента, передаваемого в соединение.

SET_AUTH.SCTP

Событие или примитив этапа 1: Set/Get Authentication Parameters.

Параметры: key_id, key, hmac_id.

GET_AUTH.TCP

Событие или примитив этапа 1: Не задано.

Параметры: current_key и rnext_key

Комментарии: current_key и rnext_key указывают предпочтительный исходящий и входящий MKT, соответственно, переданные в недавно принятом сегменте.

GET_AUTH.SCTP

Событие или примитив этапа 1: Set/Get Authentication Parameters.

Параметры: key_id и chunk_list.

RESET_STREAM.SCTP

Событие или примитив этапа 1: Add/Reset Streams, Reset Association.

Параметры: sid и направление.

RESET_STREAM-EVENT.SCTP

Событие или примитив этапа 1: Уведомление Stream Reset.

Параметры: Информация о результате RESET_STREAM.SCTP.

Комментарии: Вызывается при завершении процедуры сброса потоков.

RESET_ASSOC.SCTP

Событие или примитив этапа 1: Add/Reset Streams, Reset Association.

Параметры: Информация, относящаяся к расширению, как определено в [RFC3260].

RESET_ASSOC-EVENT.SCTP

Событие или примитив этапа 1: Уведомление Association Reset.

Параметры: Информация о результате RESET_ASSOC.SCTP.

Комментарии: Вызывается при завершении процедуры сброса ассоциации.

ADD_STREAM.SCTP

Событие или примитив этапа 1: Add/Reset Streams, Reset Association.

Параметры: Число добавленных исходящих и входящих потоков.

ADD_STREAM-EVENT.SCTP

Событие или примитив этапа 1: Уведомление Stream Change.

Параметры: Информация о результате ADD_STREAM.SCTP.

Комментарии: Вызывается при завершении процедуры добавления потока.

SET_STREAM_SCHEDULER.SCTP

Событие или примитив этапа 1: Set Stream Scheduler.

Параметры: Идентификатор планировщика.

Комментарии: Выбор из планировщиков First-Come-First-Served, Round-Robin, Round-Robin по пакетам, Priority-Based, Fair Bandwidth, Weighted Fair Queuing.

CONFIGURE_STREAM_SCHEDULER.SCTP

Событие или примитив этапа 1: Configure Stream Scheduler.

Параметры: Приоритет.

Комментарии: Значение приоритета применимо лишь при выборе планировщика Priority-Based или Weighted Fair Queuing с помощью SET_STREAM_SCHEDULER.SCTP. Смысл параметра разный для этих планировщиков, но в обоих случаях он реализует ту или иную форму приоритизации в части распределения пропускной способности между потоками.

SET_FLOWLABEL.SCTP

Событие или примитив этапа 1: Set IPv6 Flow Label

Параметры: Метка потока.

Комментарии: Примитив позволяет приложению изменять метку потока в заголовке IPv6 для исходящих пакетов на пути.

AUTHENTICATION_NOTIFICATION-EVENT.SCTP

Событие или примитив этапа 1: Уведомление Authentication.

Возврат: Информация, относящаяся к управлению ключами.

CONFIG_SEND_BUFFER.SCTP

Событие или примитив этапа 1: Configure Send Buffer Size.

Параметры: Размер в октетах.

CONFIG_RECEIVE_BUFFER.SCTP

Событие или примитив этапа 1: Configure Receive Buffer Size.

Параметры: Размер в октетах.

Комментарии: Примитив управляет размером окна у получателя.

CONFIG_FRAGMENTATION.SCTP

Событие или примитив этапа 1: Configure Message Fragmentation.

Параметры: Логическое значение (enable/disable) и максимальный размер (необязательно, по умолчанию PMTU).

Комментарии: Если фрагментация разрешена, сообщения с размером больше максимально разрешенного будут фрагментироваться, а при отключенной фрагментации такие сообщения будут вызывать ошибку.

CONFIG_PMTUD.SCTP

Событие или примитив этапа 1: Configure Path MTU Discovery.

Параметры: Одно логическое значение (PMTUD вкл./выкл.) и размер PMTU (необязательно).

Возврат: Значение PMTU.

Комментарии: Примитив возвращает осмысленное значение PMTU, когда PMTUD включено (логический параметр true) и позволяет установить PMTU, если PMTUD отключено (логический параметр false).

CONFIG_DELAYED_SACK.SCTP

Событие или примитив этапа 1: Configure Delayed SACK Timer.

Параметры: Одно логическое значение (задержка SACK вкл./выкл.), значение таймера (необязательно) и число пакетов для ожидания (по умолчанию 2).

Комментарии: Если задержка SACK разрешена, SCTP будет слоть SACK при получении заданного числа пакетов или по таймеру (что наступит раньше).

CONFIG_RTO.SCTP

Событие или примитив этапа 1: Configure RTO Calculation.

Параметры: Начальное (необязательно), минимальное (необязательно), максимальное (необязательно) значение RTO.

Комментарии: Примитив настраивает начальное, минимальное и максимальное значение RTO.

SET_COOKIE_LIFE.SCTP

Событие или примитив этапа 1: Set Cookie Life Value

Параметры: Значение cookie life.

SET_MAX_BURST.SCTP

Событие или примитив этапа 1: Set Maximum Burst.

Параметры: Максимальный размер пика.

Комментарии: Не все реализации поддерживают значения больше 4.

SET_PARTIAL_DELIVERY_POINT.SCTP

Событие или примитив этапа 1: Set Partial Delivery Point

Параметры: Граница частичной доставки (integer).

Комментарии: Этот параметр должен быть не больше размера приемного буфера сокета.

SET_CHECKSUM_ENABLED.UDP

Событие или примитив этапа 1: Checksum_Enabled.

Параметры: 0 когда отправитель задает нулевую контрольную сумму, 1 при расчете контрольной суммы отправителем (принято по умолчанию).

SET_CHECKSUM_REQUIRED.UDP

Событие или примитив этапа 1: Require_Checksum.

Параметры 0 разрешает нулевую контрольную сумму, 1 требует на приемной стороне контрольную сумму, отличную от 0 (принято по умолчанию).

SET_CHECKSUM_COVERAGE.UDP-Lite

Событие или примитив этапа 1: Set_Checksum_Coverage.

Параметры: Размер покрытия контрольной суммой у отправителя (по умолчанию максимальное покрытие).

SET_MIN_CHECKSUM_COVERAGE.UDP-Lite

Событие или примитив этапа 1: Set_Min_Coverage.

Параметры: Размер покрытия контрольной суммой у получателя (по умолчанию минимальное покрытие).

SET_DF.UDP(-Lite)

Событие или примитив этапа 1: Set_DF.

Параметры: 0, если флаг DF не установлен (принято по умолчанию) в заголовке IPv4, 1 при установленном флаге.

GET_MMS_S.UDP(-Lite)

Событие или примитив этапа 1: Get_MM_S.

Комментарии: Примитив позволяет определить максимальный размер транспортного сообщения, которое может быть передано без фрагментации IP с настроенного интерфейса.

GET_MMS_R.UDP(-Lite)

Событие или примитив этапа 1: Get_MMS_R.

Комментарии: Примитив позволяет определить максимальный размер транспортного сообщения, которое можно принять через настроенный интерфейс.

SET_TTL.UDP(-Lite) (IPV6_UNICAST_HOPS)

Событие или примитив этапа 1: Set_TTL и Set_IPV6_Unicast_Hops.

Параметры: Значение IPv4 TTL или IPv6 Hop Count.

Комментарии: Примитив позволяет приложению изменить значение IPv4 TTL или IPv6 Hop Count для исходящих дейтаграмм UDP(-Lite).

GET_TTL.UDP(-Lite) (IPV6_UNICAST_HOPS)

Событие или примитив этапа 1: Get_TTL и Get_IPV6_Unicast_Hops.

Возврат: Значение IPv4 TTL или IPv6 Hop Count.

Комментарии: Примитив позволяет приложению узнать значение IPv4 TTL или IPv6 Hop Count из входящей дейтаграммы UDP(-Lite).

SET_ECN.UDP(-Lite)

Событие или примитив этапа 1: Set_ECN.

Параметры: Значение ECN.

Комментарии: Примитив позволяет приложению UDP(-Lite) установить код явного контроля перегрузок (ECN) в исходящих дейтаграммах UDP(-Lite). По умолчанию передается 00.

GET_ECN.UDP(-Lite)

Событие или примитив этапа 1: Get_ECN.

Параметры: Значение ECN.

Комментарии: Примитив позволяет приложению UDP(-Lite) прочитать код явного контроля перегрузок (ECN) из входящей дейтаграммы UDP(-Lite).

SET_IP_OPTIONS.UDP(-Lite)

Событие или примитив этапа 1: Set_IP_Options.

Параметры: Опции.

Комментарии: Примитив позволяет приложению UDP(-Lite) установить опции IP для исходящих дейтаграмм UDP(-Lite). Опции могут включать по меньшей мере Source Route, Record Route, Timestamp.

GET_IP_OPTIONS.UDP(-Lite)

Событие или примитив этапа 1: Get_IP_Options.

Возврат: Опции.

Комментарии: Примитив позволяет приложению UDP(-Lite) прочитать любые опции IP из входящей дейтаграммы UDP(-Lite).

CONFIGURE.LEDBAT

Событие или примитив этапа 1: Не задано.

Параметры: enable (логическое значение), target, allowed_increase, gain_inc, gain_dec, base_history, current_filter, init_cwnd, min_cwnd.

Комментарии: Новый параметр enable включает или отключает сервис LEDBAT.

Прерывание

Gracefully or forcefully closing a connection or being informed about this event happening.

CLOSE.TCP

Событие или примитив этапа 1: Close.

Комментарии: Примитив закрывает соединение на передающей стороне после гарантированной доставки оставшихся данных.

CLOSE.SCTP

Событие или примитив этапа 1: Shutdown.

Комментарии: Прерывает соединение после гарантированной доставки оставшихся данных.

ABORT.TCP

Событие или примитив этапа 1: Abort.

Комментарии: Прерывает соединение без доставки оставшихся данных и передает удаленной стороне сообщение об ошибке.

ABORT.SCTP

Событие или примитив этапа 1: Abort.

Параметры: Причина разрыва для партнера (необязательно).

Комментарии: Прерывает соединение без доставки оставшихся данных и передает другой стороне сообщение об ошибке.

ABORT.UDP(-Lite)

Событие или примитив этапа 1: Close.

Комментарии: Прерывает соединение без доставки оставшихся данных. Через этот экземпляр соединения дейтаграммы UDP(-Lite) больше не передаются и не принимаются.

TIMEOUT.TCP

Событие или примитив этапа 1: Событие User Timeout.

Комментарии: Приложение уведомляется о разрыве соединения. Событие вызывается по таймеру, установленному CONNECTION.ESTABLISHMENT.CONNECT.TCP (возможно изменено с помощью CONNECTION.MAINTENANCE.CHANGE_TIMEOUT.TCP).

TIMEOUT.SCTP

Событие или примитив этапа 1: Событие Communication Lost.

Комментарии: Приложение уведомляется о разрыве соединения. Событие вызывается по тайм-ауту, который следует включать по умолчанию (см. начало параграфа 8.3 в [RFC4960]). Тайм-аут может быть изменен с помощью CONNECTION.MAINTENANCE.CHANGE_TIMEOOUT.SCTP.

ABORT-EVENT.TCP

Событие или примитив этапа 1: Не задано.

ABORT-EVENT.SCTP

Событие или примитив этапа 1: Событие Communication Lost.

Возврат: Причина разрыва от партнера (при ее доступности).

Комментарии: Сообщает приложению о разрыве соединения партнером с помощью CONNECTION.TERMINATION.ABORT.SCTP.

CLOSE-EVENT.TCP

Событие или примитив этапа 1: Не задано.

CLOSE-EVENT.SCTP

Событие или примитив этапа 1: Событие Shutdown Complete.

Комментарии: Приложение информируется об успешном вызове CONNECTION.TERMINATION.CLOSE.SCTP.

4.2. Примитивы, связанные с передачей данных

Все примитивы в этом разделе относятся к существующему соединению, т. е. соединению, которое было организовано или стало доступным для получения данных (хотя это не обязательно для примитивов UDP(-Lite)). В дополнение к указанным параметра все примитивы передачи включают ссылку на блок данных, а примитивы приема - ссылку на буфер для размещения принимаемых данных. Отметим, что примитивы CONNECT.TCP и LISTEN.TCP в категории организации (ESTABLISHMENT) и доступности (AVAILABILITY) соединения также позволяют передавать данные (необязательное пользовательское сообщение) до завершения процесса организации соединения.

SEND.TCP

Событие или примитив этапа 1: Send.

Параметры: timeout (необязательно), current_key (необязательно), rnext_key (необязательно).

Комментарии: Примитив отдает TCP блок данных для гарантированной передачи TCP на другой стороне соединения. Этот вызов позволяет указать тайм-аут(см. CONNECTION.MAINTENANCE.CHANGE_TIMEOUT.TCP). Параметры current_key и rnext_key связаны с проверкой подлинности и могут мыть настроены этим вызовом (см. CONNECTION.MAINTENANCE.SET_AUTH.TCP).

SEND.SCTP

Событие или примитив этапа 1: Send.

Параметры: Номер потока, контекст (необязательно), сокет (необязательно), флаг неупорядоченности (необязательно), флаг управления группировкой (необязательно), идентификатор протокола в данных (необязательно), pr-policy (необязательно), pr-value (необязательно), флаг незамедлительного подтверждения (необязательно), key-id (необязательно).

Комментарии: Примитив отдает SCTP блок данных для передачи SCTP на другой стороне соединения (ассоциации SCTP). Параметр stream number указывает используемый поток, context может позднее применяться для ссылки на корректное сообщение при уведомлении об ошибке, socket может служить для указания предпочтительного состояния пути при наличии нескольких путей (см. CONNECTION.MAINTENANCE.SETPRIMARY.SCTP). Блок данных может доставляться без сохранения порядка, если установлен флаг unordered. Флаг no-bundle устанавливается для указания предпочтительности отказа от группировки сообщений. Идентификатор протокола в поле данных указывает принимающему приложению способ обработки данных. С помощью параметров pr-policy и pr-value можно управлять уровнем гарантий, флаг sack-immediately указывает партнеру, что следует передать соответствующее подтверждение SACK без задержки. Параметр key-id применяется для аутентификации пользовательского сообщения.

SEND.UDP(-Lite)

Событие или примитив этапа 1: Send.

Параметры: IP-адрес и номер порта у получателя (необязательно при соединении).

Комментарии: Примитив предоставляет сообщение для гарантированной доставки с использованием UDP(-Lite) по указанному транспортному адресу. Адрес IP и номер порта можно не указывать для соединенных сокетов UDP(-Lite). При отправке сообщения применяются все примитивы CONNECTION.MAINTENANCE.SET_*.UDP(-Lite).

RECEIVE.TCP

Событие или примитив этапа 1: Receive.

Параметры: current_key (необязательно) и rnext_key (необязательно).

Комментарии: Параметры аутентификации current_key и rnext_key могут быть прочитаны этим вызовом (см. CONNECTION.MAINTENANCE.GET_AUTH.TCP).

RECEIVE.SCTP

Событие или примитив этапа 1: Уведомление Data Arrive, затем Receive.

Параметры: Номер потока (необязательно).

Возврат: Порядковый номер потока (необязательно) и флаг частичной передачи (необязательно).

Комментарии: При указании номера потока вызов принимает данные лишь из этого потока. Если принимается неполное сообщения, это указывается флагом partial и должен быть представлен порядковый номер потока, чтобы приложение могло восстановить корректный порядок блоков данных всего сообщения.

RECEIVE.UDP(-Lite)

Событие или примитив этапа 1: Receive.

Параметры: Буфер для принимаемой дейтаграммы.

Комментарии: К сообщению применяются все примитивы CONNECTION.MAINTENANCE.GET_*.UDP(-Lite).

SENDFAILURE-EVENT.SCTP

Событие или примитив этапа 1: Уведомление Send Failure, за которым может слодовать сообщение Receive Unsent или Receive Unacknowledged.

Возврат: Код причины и не переданное (не подтвержденное) сообщение (необязательно).

Комментарии: Код причины указывает причину отказа, а context - номер контекста, если он предоставлен в DATA.SEND.SCTP, для последующего использования с сообщением Receive Unsent или Receive Unacknowledged. Эти примитивы можно применять для извлечения сообщений (или частей в случае частичной передачи), которые не были переданы или подтверждены.

SEND_FAILURE.UDP(-Lite)

Событие или примитив этапа 1: Send.

Комментарии: Этот примитив можно применять для определения эффективного PMTU при использовании с примитивом MAINTENANCE.SET_DF.

SENDER_DRY-EVENT.SCTP
Событие или примитив этапа 1: Уведомление Sender Dry.
Комментарии: Информирует приложение об отсутствии в стеке пользовательских данных для передачи.

PARTIAL_DELIVERY_ABORTED-EVENT.SCTP
Событие или примитив этапа 1: Уведомление Partial Delivery Aborted.
Комментарии: Информирует получателя частичного сообщения о том, что последующая доставка была прервана.

5. Этап 3

Здесь представлен расширенный набор всех транспортных свойств всех протоколов, описанных в предшествующих разделах, на основе списка примитивов этапа 2, а также текста этапа 1 для включения транспортных свойств, которые могут быть настроены в одном протоколе и являются статическими в другом (например, контроль перегрузок). Некоторые мелкие детали опущены в целях общности, например, TCP может предоставлять разные опции IP, но лишь опция source route обязательна для реализации и эта деталь не видна на этапе 3 в транспортном свойстве «задание опций IP». Как и раньше, UDP(-Lite) представляет протоколы UDP и UDP-Lite, а TCP - TCP и MPTCP.

5.1. Свойства, связанные с соединением

Организация

Активное создание соединения одной транспортной конечной точки с одной или несколькими другими конечными транспортными точками.

Connect
Протоколы: TCP, SCTP, UDP(-Lite).

Задание опций IP, которые должны применяться всегда
Протоколы: TCP и UDP(-Lite).

Запрос множества потоков
Протоколы: SCTP.

Ограничение числа входящих потоков
Протоколы: SCTP.

Задание числа попыток и/или тайм-аутов для первого сообщения при организации
Протоколы: TCP и SCTP.

Получение множества сокетов
Протоколы: SCTP.

Отключение MPTCP
Протоколы: MPTCP.

Настройка аутентификации
Протоколы: TCP и SCTP.
Комментарии: В TCP это позволяет настраивать кортежи MKT, в SCTP - указание блоков, для которых нужна аутентификация. DATA, ACK и т. п. Являются в SCTP разными блоками и могут включаться в один пакет.

Указание кода уровня адаптации
Протоколы: SCTP.

Запрос согласования для чередования пользовательских сообщений
Протоколы: SCTP.

Отправка сообщения с гарантией доставки (возможно несколько раз) до организации соединения
Протоколы: TCP.

Отправка сообщения с гарантией доставки в процессе организации соединения
Протоколы: SCTP.

Включение UDP-инкапсуляции с заданным удаленным портом UDP
Протоколы: SCTP.

Доступность

Подготовка к приему входящих запросов на соединение.

Listen, 1 локальный интерфейс
Протоколы: TCP, SCTP, UDP(-Lite).

Listen, N локальных интерфейсов
Протоколы: SCTP.

Listen, все локальные интерфейсы
Протоколы: TCP, SCTP, UDP(-Lite).

Получение запрошенного числа потоков
Протоколы: SCTP.

Ограничение числа входящих потоков
Протоколы: SCTP.

Задание опций IP, которые должны применяться всегда
Протоколы: TCP и UDP(-Lite).

Отключение MPTCP
Протоколы: MPTCP.

Настройка аутентификации
Протоколы: TCP и SCTP.
Комментарии: В TCP это позволяет настраивать кортежи MKT, в SCTP - указание блоков, для которых нужна аутентификация. DATA, ACK и т. п. Являются в SCTP разными блоками и могут включаться в один пакет.

Указание кода уровня адаптации
Протоколы: SCTP.

Поддержка

Настройки для открытых соединений или уведомления о них.

Смена тайм-аута для разрыва соединения (время или число повторов)
Протоколы: TCP и SCTP.

Предложенный партнеру тайм-аут
Протоколы: TCP.
Отключение алгоритма Nagle
Протоколы: TCP и SCTP.
Запрос немедленной передачи heartbeat с возвратом результата
Протоколы: SCTP.
Уведомление об избыточных повторах (ранее до достижения порога разрыва)
Протоколы: TCP.
Добавление пути
Протоколы: MPTCP и SCTP.
Параметры MPTCP: source-IP, source-Port, destination-IP, destination-Port.
Параметры SCTP: локальный IP-адрес.
Удаление пути
Протоколы: MPTCP и SCTP.
Параметры MPTCP: source-IP, source-Port, destination-IP, destination-Port.
Параметры SCTP: локальный IP-адрес.
Задание основного пути
Протоколы: SCTP.
Предложение основного пути партнеру
Протоколы: SCTP.
Настройка переключения пути
Протоколы: SCTP.
Определение статуса (запрос или уведомление)
Протоколы: SCTP и MPTCP.
Параметры SCTP: Состояние соединения для ассоциации, список транспортных адресов получателей, состояния доступности транспортных адресов получателей, текущие размеры окна приема (локальный и у партнера), текущий размер локального окна перегрузки, число неподтвержденных блоков DATA, число блоков DATA, ожидающих приема, последнее значение SRTT на основном пути, RTO на основном пути, SRTT и RTO для других адресов получателей, MTU для путей, поддержка чередования (yes/no).
Параметры MPTCP: Список субпотоков (идентификация по source-IP, source-Port, destination-IP, destination-Port).
Задание поля DSCP
Протоколы: TCP, SCTP, UDP(-Lite).
Уведомление о приеме сообщения ICMP об ошибке
Протоколы: TCP и UDP(-Lite).
Смена параметров аутентификации
Протоколы: TCP и SCTP.
Получение данных аутентификации
Протоколы: TCP и SCTP.
Сброс потока
Протоколы: SCTP.
Уведомление о сбросе потока
Протоколы: STCP.
Сброс ассоциации
Протоколы: SCTP.
Уведомление о сбросе ассоциации
Протоколы: STCP.
Добавление потоков
Протоколы: SCTP.
Уведомление о добавлении потоков
Протоколы: STCP.
Выбор планировщика для потоков в ассоциации
Протоколы: SCTP.
Настройка приоритета или веса для планировщика
Протоколы: SCTP.
Установка метки потока IPv6
Протоколы: SCTP.
Настройка размера буфера передачи
Протоколы: SCTP.
Настройка размера приемного буфера (и rwnd)
Протоколы: SCTP.
Настройка фрагментации сообщений
Протоколы: SCTP.
Настройка PMTUD
Протоколы: SCTP.
Настройка таймера задержки SACK
Протоколы: SCTP.
Установка значения Cookie life
Протоколы: SCTP.
Установка максимального пика
Протоколы: SCTP.
Настройка размера сообщений для частичной доставки
Протоколы: SCTP.
Запрет контрольной суммы при отправке
Протоколы: UDP.
Запрет требования контрольной суммы при получении
Протоколы: UDP.

Задание покрытия для контрольной суммы у отправителя
Протоколы: UDP-Lite.
Минимальное покрытие для контрольной суммы, требуемое получателем
Протоколы: UDP-Lite.
Задание поля DF
Протоколы: UDP(-Lite).
Определение максимального размера транспортного сообщения для передачи без фрагментации IP через настроенный интерфейс
Протоколы: UDP(-Lite).
Определение максимального размера транспортного сообщения от настроенного интерфейса
Протоколы: UDP(-Lite).
Задание поля TTL/Hop Count
Протоколы: UDP(-Lite).
Получение поля TTL/Hop Count
Протоколы: UDP(-Lite).
Задание поля ECN
Протоколы: UDP(-Lite).
Получение поля ECN
Протоколы: UDP(-Lite).
Задание опций IP
Протоколы: UDP(-Lite).
Получение опций IP
Протоколы: UDP(-Lite).
Включение и настройка LEDBAT
Протоколы: Протоколы, реализующие механизм контроля перегрузок LEDBAT.
Прерывание

Аккуратное или жесткое завершение соединения или информирование о таком событии.

Закрытие после гарантированной доставки оставшихся данных с уведомлением приложения на другой стороне
Протоколы: TCP и SCTP.
Комментарии: Конечная точка TCP локально закрывает соединение лишь для передачи и может ждать приема данных.
Разрыв без доставки оставшихся данных с уведомлением приложения на другой стороне
Протоколы: TCP и SCTP.
Комментарии: В SCTP приложение может указывать причину разрыва, которая может передаваться приложению на другой стороне.
Разрыв без доставки оставшихся данных и уведомления приложения на другой стороне
Протоколы: UDP(-Lite).
Тайм-аут, когда данные не удается доставить слишком долго
Протоколы: TCP и SCTP.
Комментарии: Тайм-аут задается CONNECTION.MAINTENANCE.

5.2. Свойства, связанные с передачей данных

Все транспортные свойства в этом разделе относятся к существующему соединению, т. е. соединению, которое организовано или стало доступным для получения данных. Отметим, что TCP позволяет передать данные (одно необязательное пользовательское сообщение, которое может быть отправлено несколько раз) еще до завершения организации соединения. Гарантии передачи данных влекут за собой задержку, например, отправитель ждет возможности передачи или передача повторяется в случае потери пакетов.

5.2.1. Передача

Все транспортные свойства в этом разделе предоставляются примитивом DATA.SEND этапа 2. DATA.SEND получает блок данных от приложения, который называется сообщением, если можно указать его начало и конец, и данными - в противном случае.

Гарантированная доставка данных с контролем перегрузок
Протоколы: TCP.
Гарантированная доставка сообщения с контролем перегрузок
Протоколы: SCTP.
Негарантированная доставка сообщения с контролем перегрузок
Протоколы: SCTP.
Негарантированная доставка сообщения без контроля перегрузок
Протоколы: UDP(-Lite).
Настраиваемые гарантии для сообщений
Протоколы: SCTP.
Выбор потока
Протоколы: SCTP.
Выбор пути (адреса получателя)
Протоколы: SCTP.
Упорядоченная доставка сообщений (возможно медленней неупорядоченной)
Протоколы: SCTP.
Неупорядоченная доставка сообщений (возможно быстрей упорядоченной)
Протоколы: SCTP и UDP(-Lite).
Запрос отмены группировки сообщений
Протоколы: SCTP.
Задание идентификатора протокола в области данных (для получателя)
Протоколы: SCTP.

Задание идентификатора ключа для аутентификации сообщения
Протоколы: SCTP.
Запрос передачи без задержки подтверждения SACK для сообщения
Протоколы: SCTP.

5.2.2. Прием

Все транспортные свойства в этом разделе предоставляются примитивом DATA.RECEIVE этапа 2. DATA.RECEIVE получает блок данных от приложения, который называется сообщением, если можно указать его начало и конец, и данными - в противном случае.

Прием данных (без границ сообщений)
Протоколы: TCP.
Прием сообщения
Протоколы: SCTP и UDP(-Lite).
Выбор потока для приема
Протоколы: SCTP.
Информация о частичной доставке сообщения
Протоколы: SCTP.
Комментарии: В SCTP неполные сообщения комбинируются с порядковым номером потока, чтобы принимающее приложение могло восстановить корректный порядок блоков данных, содержащих сообщение.

5.2.3. Ошибки

Здесь описаны отказы при передаче, связанные с конкретными вызовами примитива DATA.SEND этапа 2.

Уведомление о неотправленном (частино) сообщении
Протоколы: SCTP и UDP(-Lite).
Уведомление о неподтвержденном (частино) сообщении
Протоколы: SCTP.
Уведомление об отсутствии в стеке данных для передачи
Протоколы: SCTP.
Уведомление получателя об отказе при частичной доставке сообщения
Протоколы: SCTP.

6. Взаимодействие с IANA

Этот документ не требует действий со стороны IANA.

7. Вопросы безопасности

Проверка подлинности, защита целостности и конфиденциальности указаны как транспортные свойства в [RFC8095]. Эти свойства обычно присущи транспортному протоколу или расположенному над ним уровню. Ни один из рассмотренных здесь транспортных протоколов сам по себе не обеспечивает этих свойств, поэтому они не рассматриваются в документе за исключением естественных функций аутентификации в TCP и SCTP, для которых применимы вопросы безопасности, отмеченные в [RFC5925] и [RFC4895].

Вопросы безопасности для UDP и UDP-Lite рассмотрены в упомянутых RFC. Рекомендации по защите приложений, использующих UDP, даны в [RFC8085].

8. Литература

8.1. Нормативные документы

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <https://www.rfc-editor.org/info/rfc3758>.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <https://www.rfc-editor.org/info/rfc4895>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <https://www.rfc-editor.org/info/rfc5061>.

[RFC5482] Eggert, L. and F. Gont, "TCP User Timeout Option", RFC 5482, DOI 10.17487/RFC5482, March 2009, <https://www.rfc-editor.org/info/rfc5482>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

[RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, DOI 10.17487/RFC6182, March 2011, <https://www.rfc-editor.org/info/rfc6182>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <https://www.rfc-editor.org/info/rfc6458>.

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, DOI 10.17487/RFC6525, February 2012, <https://www.rfc-editor.org/info/rfc6525>.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <https://www.rfc-editor.org/info/rfc6817>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <https://www.rfc-editor.org/info/rfc6824>.

[RFC6897] Scharf, M. and A. Ford, "Multipath TCP (MPTCP) Application Interface Considerations", RFC 6897, DOI 10.17487/RFC6897, March 2013, <https://www.rfc-editor.org/info/rfc6897>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

[RFC7053] Tuexen, M., Ruengeler, I., and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol", RFC 7053, DOI 10.17487/RFC7053, November 2013, <https://www.rfc-editor.org/info/rfc7053>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <https://www.rfc-editor.org/info/rfc7413>.

[RFC7496] Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partially Reliable Stream Control Transmission Protocol Extension", RFC 7496, DOI 10.17487/RFC7496, April 2015, <https://www.rfc-editor.org/info/rfc7496>.

[RFC7829] Nishida, Y., Natarajan, P., Caro, A., Amer, P., and K. Nielsen, "SCTP-PF: A Quick Failover Algorithm for the Stream Control Transmission Protocol", RFC 7829, DOI 10.17487/RFC7829, April 2016, <https://www.rfc-editor.org/info/rfc7829>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8260] Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol", RFC 8260, DOI 10.17487/RFC8260, November 2017, <https://www.rfc-editor.org/info/rfc8260>.

[RFC8304] Fairhurst, G. and T. Jones, "Transport Features of the User Datagram Protocol (UDP) and Lightweight UDP (UDP-Lite)", RFC 8304, DOI 10.17487/RFC8304, February 2018, <https://www.rfc-editor.org/info/rfc8304>.

8.2. Дополнительная литература

[RFC0854] Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, DOI 10.17487/RFC0854, May 1983, <https://www.rfc-editor.org/info/rfc854>.

[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>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/info/rfc2475>.

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, DOI 10.17487/RFC3260, April 2002, <https://www.rfc-editor.org/info/rfc3260>.

[RFC5461] Gont, F., "TCP's Reaction to Soft Errors", RFC 5461, DOI 10.17487/RFC5461, February 2009, <https://www.rfc-editor.org/info/rfc5461>.

[RFC6093] Gont, F. and A. Yourtchenko, "On the Implementation of the TCP Urgent Mechanism", RFC 6093, DOI 10.17487/RFC6093, January 2011, <https://www.rfc-editor.org/info/rfc6093>.

[RFC7414] Duke, M., Braden, R., Eddy, W., Blanton, E., and A. Zimmermann, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 7414, DOI 10.17487/RFC7414, February 2015, <https://www.rfc-editor.org/info/rfc7414>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <https://www.rfc-editor.org/info/rfc7657>.

[RFC8095] Fairhurst, G., Ed., Trammell, B., Ed., and M. Kuehlewind, Ed., "Services Provided by IETF Transport Protocols and Congestion Control Mechanisms", RFC 8095, DOI 10.17487/RFC8095, March 2017, <https://www.rfc-editor.org/info/rfc8095>.

[TAPS-MINSET] Welzl, M. and S. Gjessing, "A Minimal Set of Transport Services for TAPS Systems", Work in Progress[1], draft-ietf-taps-minset-01, February 2018.

Приложение A. Список RFC, использованных для этапа 1

TCP

[RFC0793], [RFC1122], [RFC5482], [RFC5925], [RFC7413].

MPTCP

[RFC6182], [RFC6824], [RFC6897].

[1]Опубликовано в RFC 8923. *Прим. перев.*

SCTP

RFC без спецификаций API сокетов: [RFC3758], [RFC4895], [RFC4960], [RFC5061].

RFC со спецификациями API сокетов: [RFC6458], [RFC6525], [RFC6951], [RFC7053], [RFC7496], [RFC7829].

UDP(-Lite)

См. [RFC8304].

LEDBAT

[RFC6817].

Приложение B. Как разрабатывался этот документ

В этом приложении представлен обзор метода, использованного при разработке документа. Метод представлен для разработчиков и может быть полезен при будущих изменениях или обновлениях.

Документ относится лишь к транспортным функциям, раскрываемым приложениям через примитивы. Документ строго следует текстам RFC. Если транспортная функция действительно относится к приложению, в RFC должно быть указано это с описанием способов использования и настройки. Таким образом, использованный при разработке подход заключался в определении нужных RFC с последующим анализом и обработкой их текстов.

Примитивы, которые **могут** быть реализованы транспортным протоколом, были исключены из рассмотрения. Для включения в документ минимальным требованием служило указание, что примитив **следует** реализовать. Там, где не применялся стиль обозначения требований из [RFC2119], примитивы исключались, если они были описаны вместе с такими утверждениями, как «некоторые реализации также представляют» или «реализация может также». Исключенные примитивы и параметры кратко упомянуты в специальных параграфах документа.

Этап 1 начинался с определения текста, описывающего примитив. Обычно примитивы описываются в спецификации API (возможно, абстрактной), но здесь представляли интерес **не только** спецификации API. Текст, описывающий примитив Send в API [RFC0793], например, не говорит о гарантированной передаче данных. Однако такие гарантии становятся очевидными из текста раздела 1 в [RFC0793].

> Протокол TCP предназначен для надежной и гарантированной доставки данных между хостами в компьютерных сетях с коммутацией пакетов и между такими сетями через промежуточные системы.

Часть текста на этапе 1 была заимствована из соответствующих RFC с корректировкой терминов в соответствии с разделом 2 и сокращением фраз для сохранения стиля документа. Была попытка представить все как описания примитивов, сделав их максимально полными (например, примитив SEND.TCP на этапе 2 явно описан как гарантированная доставка данных). Текст, относящийся к представленным на этом этапе примитивам, но не входящий непосредственно в описание какого-либо примитива, использовался во вводной части параграфов.

Этап 2 служил для унификации примитивов. Входными данными служил лишь текст этапа 1 (без внешних источников). Список на этапе 2 упорядочен не по протоколам (в стиле «протокол X и его примитивы, протокол Y и его примитивы»), а по примитивам («примитив A реализован определенным способомв протоколе X, иным способом в протоколе Y ...»). Цель состояла в получении на этапе 2 максимального числа похожих примитивов. Например, иной раз это достигалось за счет того, что не всегда поддерживалось отображение 1:1 между этапами 1 и 2, примитивы переименовывались и пр. Для каждого нового примитива рассматривались уже имеющиеся для обеспечения максимальной согласованности.

Для каждого примитива описание было представлено в стиле

ИмяПримитива.Протокол

Событие или примитив этапа 1:
Параметры:
Возврат:
Комментарии:

Записи «Параметры», «Возврат», «Комментарии» пропускались, если с примитивом не было связано соответствующей информации. Необязательные параметры были помечены текстом «(необязательно)», при наличии принятых по умолчанию значений они указывались.

Этап 3 определял транспортные свойства (функции), являющиеся статическими свойствами протокола, и для них указывались соответствующие протоколы. Это создавало финальный список всех доступных транспортных функций. Список основан прежде всего на тексте этапа 2 с дополнительными данными этапа 1 (но без внешних источников).

Благодарности

Авторы благодарят (в алфавитном порядке) Bob Briscoe, Spencer Dawkins, Aaron Falk, David Hayes, Karen Nielsen, Tommy Pauly, Joe Touch, Brian Trammell за ценные отклики на этот документ. Большое спавибо Christoph Paasch за информацию о Multipath TCP, а Gorry Fairhurst и Tom Jones - за UDP(-Lite). Эта работа финансировалась исследовательской и инновационной программой Европейского Союза Horizon 2020 в рамках гранта № 644334 (NEAT).

Адреса авторов

Michael Welzl

University of Oslo

PO Box 1080 Blindern

Oslo N-0316

Norway

Email: michawe@ifi.uio.no

Michael Tuexen

Muenster University of Applied Sciences

Stegerwaldstrasse 39

Steinfurt 48565

Germany

Email: tuexen@fh-muenster.de

Naeem Khademi

University of Oslo

PO Box 1080 Blindern

Oslo N-0316

Norway

Email: naeemk@ifi.uio.no

Перевод на русский язык
Николай Малых

nmalykh@protocols.ru