Internet Engineering Task Force (IETF)
Request for Comments: 8923
Category: Informational
ISSN: 2070-1721

M. Welzl
S. Gjessing
University of Oslo

October 2020

A Minimal Set of Transport Services for End Systems

Минимальный набор транспортных служб для конечной системы

Аннотация

Этот документ рекомендует минимальный набор транспортных служб (Transport Service), поддерживаемых конечной системой, и дает рекомендации по выбору доступных механизмов и протоколов. Документ основан на наборе транспортных функций из RFC 8303.

Статус документа

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

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. 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/rfc8923.

Оглавление

[1]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

В настоящее время набор транспортных служб, используемых большинством приложений, основан на TCP и UDP (и вышележащих протоколах на исх основе), что ограничивает для сетевого стека возможности использовать свойства других транспортных протоколов. Например, если протокол поддерживает доставку пакетов с нарушением порядка, а приложение предполагает упорядоченность потока байтов в сети, сетевой стек не сможет сразу же доставить сообщение, полученное с нарушением порядка, и это вступит в конфликт с базовым допущением приложения. Результатом этого станет неоправданная задержка из-за блокировки HOL[1].

Раскрывая транспортные службы нескольких протоколов, транспортная система позволяет приложениям использовать эти службы без статической привязки к определенному транспортному протоколу. Первый шаг к разработке таких систем был предложен в [RFC8095], где было исследовано множество транспортных протоколов, а также в [RFC8303] и [RFC8304], указавших конкретные свойства транспорта, которые раскрываются приложениям протоколами TCP, Multipath TCP (MPTCP), UDP(-Lite), SCTP[2], а также механизмом контроля перегрузок LEDBAT[3]. Механизм LEDBAT был включен в этот список единственным среди механизмов контроля перегрузок, поскольку предоставляемые им услуги существенно отличаются от других механизмов этого типа. Этот документ основан на упомянутых работах и использует принятую в них терминологию (приведена ниже). Поскольку рассматриваемые транспортные протоколы совместно охватывают широкий спектр транспортных функций, есть основания надеяться, что предлагаемый набор (о приведшие к его выбору рассуждения) будут применимы ко многим аспектам других транспортных протоколов, которые смогут использоваться в современных и будущих решениях.

Отделив приложения от транспортных протоколов, транспортная система обеспечивает иной уровень абстракции, нежели интерфейс сокетов Berkeley [POSIX]. Как и в языках программирования, повышение уровня абстракции обеспечивает большую свободу автоматизации под интерфейсом, но отнимает часть контроля у программиста приложений. Это компромисс, с которым сталкивается разработчик транспортной системы, и в данном документе приведены рекомендации для этого уровня абстракции. Некоторые транспортные свойства в настоящее время редко включаются в API, однако их нужно предлагать, иначе они никогда не будут использоваться. Другие транспортные функции предлагаются API описываемых здесь протоколов, но если их не раскрывать в API, это даст больше свободы при автоматизации использования протокола в транспортной системе. Представленный здесь минимальный набор является попыткой найти компромисс, который можно рекомендовать для реализации транспортных систем на основе функций, рассматриваемых в [RFC8303].

Современные приложения используют множество API. Хотя этот документ предназначен для включения в API, разработанный группой Transport Services (TAPS) [TAPS-INTERFACE], большинства наиболее важных транспортных функций, представленный здесь «минимальный набор» должен включаться во **все** сетевые API, для обеспечения возможности использовать базовую функциональность. Например, это может не помочь приложению, взаимодействующему с библиотекой, которая обеспечивает свой коммуникационный интерфейс, если базовый Berkeley Sockets API расширен для поддержки неупорядоченной доставки, но библиотека раскрывает лишь упорядоченный поток байтов. Для работы Berkeley Sockets API и библиотека должны раскрыть транспортное свойство неупорядоченной доставки (или использовать это свойство без его раскрытия на основе информации о приложении, но это не подходит в общем случае). Точно так же транспортные протоколы, такие как SCTP, предлагают многопотоковую передачу, которая не может использоваться, например, для приоритизации сообщений в разных потоках, пока приложения не обмениваются значениями приоритета, которые следует применять, и группой соединений для этого. В большинстве случаев для максимальной гибкости и эффективности лучшим решением для библиотеки будет раскрытие по меньшей мере тех возможностей, которые включены здесь в «минимальный набор».

«Минимальный набор» можно реализовать «односторонне» на основе TCP. Это означает, что транспортная система на стороне отправителя может взаимодействовать со стандартным получателем TCP, а на стороне получателя - со стандартным отправителем TCP. С некоторыми ограничениями возможна реализация минимального набора на одной стороне по протоколу UDP. Хотя возможность реализации на одной стороне может быть полезна при развертывании, за это приходится платить ограничением набора услуг, которые могут быть предоставлены TCP (с большими ограничениями и UDP). Таким образом, минимальный набор применим для многих, но не для всех приложений, поскольку некоторые прикладные протоколы предъявляют требования, не выполнимые минимальным набором.

Отметим, что упоминаемые в документе протоколы используются естественным способом. Например, при обсуждении транспортных свойств TCP или реализованных на основе TCP имеется в виду естественное использование TCP, а не инкапсуляция этого протокола в иной транспортный протокол, такой как UDP.

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).

Application - приложение

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

[1]Head-of-line - блокировка в начале (голове) линии.
[2]Stream Control Transmission Protocol - потоковый протокол управления передачей.
[3]Low Extra Delay Background Transport - транспорт с очень малыми задержками.

Application-specific knowledge
Информация, которую имеет лишь приложение.

End system - конечная система
Сущность (элемент), взаимодействующая с одной или несколькими другими конечными системами с помощью протокола транспортного уровня. Конечная система предоставляет приложению интерфейс транспортного уровня.

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

Connection Group - группа соединений
Набор соединений с одной конфигурацией (настройка одного из соединений передается другим членам группы). Такие соединения будут называться групповыми или сгруппированными в отличие от одиночных.

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

Кроме того, в этом документе применяется термин UDP(-Lite) при обсуждении транспортных свойств, присущих UDP и UDP-Lite. Точно так же TCP обозначает TCP и MPTCP.

3. Определение минимального набора

Предполагается, что у приложений нет специальных требований к наличию сведений о сети, например, в части выбора сетевого интерфейса или сквозного пути. Но даже при таком допущении имеется некоторые требования, которые строго соблюдаются современными транспортными протоколами и должны соблюдаться транспортной системой. Некоторые их таких требований к транспортным свойствам здесь называются функциональными.

Функциональные свойства транспорта предоставляют возможности, которые не могут использоваться без ведома приложения, поскольку иначе приложение может отказать. Например, функциональным свойством является упорядоченная доставка сообщений и она не может быть задана в конфигурации без ведома приложения, поскольку приложение может полагаться на упорядоченную доставку сообщений. Отказами будут любые изменения работы приложения, которые не связаны с производительностью, например, изменение уровня защищенности.

Изменение DSCP и запрет алгоритма Nagle являются примерами транспортных свойств, которые здесь называются оптимизацией. Если транспортная система сама принимает решение об их включении или отключении у приложений не будет возникать отказов, но их работа может стать менее эффективной, если приложению будет недоступен контроль функций оптимизации. Для этих функций требуется информация от конкретного приложения (например, требования к задержке или пропускной способности, размер планируемых к передаче блоков данных).

Транспортные функции транспортных протоколов IETF, для которых не требуются сведения от конкретного приложения и которые могут контролироваться без участия приложений, называются автоматизируемыми.

Подход к определению минимального набора транспортных свойств описана ниже.

1. *Классификация* (Приложение A). Представлено надмножество транспортных свойств из [RFC8303] и эти функции разделены на функциональные, оптимизационные и автоматизируемые.
2. *Сокращение* (раздел 4). Сокращенный список транспортных свойств выводится из категорий п. 1. при этом исключаются все транспортные функции, которые не требуют сведений от конкретных приложений или могут приводить к семантически некорректному поведению при реализации на основе TCP или UDP.
3. *Обсуждение* (раздел 5). Полученный в результате список показывает набор обсуждаемых свойств для определения основы при выводе минимального набора свойств.
4. *Построение* (раздел 6). На основе сокращения и обсуждения транспортных свойств определяется их минимальный набор.

Следуя [RFC8303] и сохраняя принятую там терминологию транспортные функции поделены на две основных группы.

1. Связанные с соединением транспортные функции:
   - организация;
   - доступность;
   - поддержка;
   - завершение.
2. Связанные с передачей данных транспортные функции:
   - передача данных;
   - прием данных;
   - ошибки.

4. Сокращенный набор транспортных функций

Скрывая автоматизируемые транспортные функции от приложения, транспортная система может использовать автоматизацию связанных с сетью функций. Это может облегчить использование транспортной системы для программистов приложений и позволяет выполнить автоматизацию, которая недоступна для приложения. Например, настройки в масштабе системы, связанные с использованием нескольких интерфейсов, проще использовать, если выбор интерфейса не определяется приложением полностью. Поскольку автоматизируемые функции не требуется обязательно раскрывать в транспортной системе, они не включены в минимальный набор функций. Это оставляет лишь те транспортные свойства, которые являются оптимизируемыми или функциональными.

Транспортной системе следует поддерживать возможность взаимодействия по протоколу TCP или UDP, если не найдено других протоколов для работы. Для многих транспортных функций это зачастую возможно лишь потому, что при конкретном запросе не делается ничего. Однако для некоторых транспортных функций установлено, что прямое

использование TCP или UDP невозможно, в таких случаях даже бездействие ведет к семантически некорректному поведению. Всякий раз при использовании приложением одной из таких транспортных функций возможность применения TCP или UDP исключается. Таким образом, в сокращенном наборе остаются лишь функциональные и оптимизируемые функции, для которых возможна реализация на основе TCP или UDP.

Приведенный ниже список содержит транспортные функции из Приложения A, сокращенные с использованием приведенных правил. Выведенный в документе «минимальный набор» может быть реализован на одной стороне по протоколу TCP и с некоторыми ограничениями - UDP. В списке транспортные функции помечены T:, если возможна реализация на основе TCP, U: - для UDP и T,U: - когда возможна реализация на основе обоих протоколов.

4.1. Связанные с соединением транспортные функции

Организация

- T,U: соединение;
- T,U: задание числа попыток и/или тайм-аута для первого сообщения организации соединения;
- T,U: отключение MPTCP;
- T: настройка аутентификации;
- T: отправка сообщения для гарантированной передачи (возможно неоднократная) до соединения;
- T: отправка сообщения для гарантированной передачи при организации соединения.

Доступность

- T,U: прослушивание;
- T,U: отключение MPTCP;
- T: настройка аутентификации.

Поддержка

- T: смена тайм-аута разрыва соединения (ограничение числа повторов или время);
- T: предложенный партнеру тайм-аут;
- T,U: отключение алгоритма Nagle;
- T,U: уведомление об избыточных повторах (упреждение перед достижением порога разрыва);
- T,U: указание поля DSCP;
- T,U: уведомление о получении сообщения ICMP об ошибке;
- T: смена параметров аутентификации;
- T: получение данных аутентификации;
- T,U: установка значение Cookie;
- T,U: выбор планировщика для управления потоками в ассоциации;
- T,U: настройка приоритета или веса для палнировщика;
- T,U: отключение контрольных сумм при передаче;
- T,U: запрет требования контрольной суммы на приеме;
- T,U: задание покрытия контрольной суммы, используемой отправителем;
- T,U: задание минимального покрытия контрольной суммы, требуемого получателем;
- T,U: задание поля DF;
- T,U: получение максимального размера транспортного сообщения, которое может быть передано без фрагментации IP, от настроенного интерфейса;
- T,U: получение максимального размера транспортного сообщения, которое может быть принято, от настроенного интерфейса;
- T,U: получение поля ECN;
- T,U: включение и настройка LEDBAT[1].

Завершение

- T: закрытие после гарантированной доставки всех оставшихся данных, вызывающее событие для информировании приложения на другой стороне;
- T: разрыв без доставки оставшихся данных, вызывающий информирование приложения на другой стороне;
- T,U: разрыв без доставки оставшихся данных и без информирования приложения на другой стороне;
- T,U: тайм-аут, когда данные не могут быть доставлены слишком долго.

[1]Low Extra Delay Background Transfer - фоновая передача с очень малой задержкой.

4.2. Связанные с передачей данных транспортные функции

4.2.1. Передача данных

- T: гарантированная доставка данных с контролем перегрузок;
- T: гарантированная доставка сообщения с контролем перегрузок;
- T,U: доставка данных без гарантии;
- T: настраиваемые гарантии доставки сообщений;
- T: упорядоченная доставка сообщений (может быть медленней неупорядоченной);
- T,U: неупорядоченная доставка сообщений (потенциально быстрее упорядоченной);
- T,U: запрос передачи сообщений без группировки (bundle);
- T: задание идентификатора ключа для проверки подлинности сообщений;
- T,U: запрос передачи подтверждений без задержки (SACK).

4.2.2. Прием данных

- T,U: получение данных (без разграничения сообщений);
- U: получение сообщений;
- T,U: информирование о частичной доставке сообщения.

4.2.3. Ошибки

В этом параграфе указаны отказы при передаче, связанные с конкретными вызовами категории «передача данных» (A.2.1. Передача).

- T,U: уведомление об отказе при передаче;
- T,U: уведомление о том, что стек больше не имеет пользовательских данных для передачи;
- T,U: уведомление получателя о прерывании частичной доставки сообщения.

5. Обсуждение

Сокращенный набор из предыдущего параграфа имеет ряд особенностей, рассматриваемых ниже. Этот раздел посвящен TCP, поскольку за исключением одной транспортной функции (5.1. Передача сообщений, прием байтов), UDP предоставляет подмножество функций TCP. Сначала рассмотрим создание транспортной системы, которая может работать на основе TCP, а затем сузим результат, чтобы система всегда могла работать по TCP или UDP (это по сути означает исключение всего, связанного с гарантиями, упорядочением, аутентификацией и разрывом соединений с уведомлением партнера).

Отметим, что по причине того, что транспортные функции UDP (кроме получения сообщений) являются подмножеством функций TCP, протокол TCP можно использовать вместо UDP, когда приложению не требуется разграничивать сообщения (например, это делает протокол уровня приложений). Это уже используется многими приложениями на практике - сначала попытаться работать по протоколу UDP, а при отказе переходить на TCP.

5.1. Передача сообщений, прием байтов

Для реализации транспортной системы на основе TCP имеется несколько транспортных функций, относящихся к передаче, и лишь одна функция, связанная с приемом - получение сообщений без их разграничения (а также, как ни странно, информирование о частичной доставке сообщения). Примечательно, что транспортная функция приема сообщения является единственной не автоматизируемой функцией UDP(-Lite), для которой невозможна реализация на основе TCP.

Для поддержки семантики получателя TCP определен «кадрированный приложением поток байтов (Application-Framed Byte Stream или байтовый поток AFra). Байтовые потоки AFra позволяют отправителям работать с сообщениями при минимальном изменении API сокета TCP. В частности, на приемной стороне изменений просто не требуется и данные можно принимать через обычный соект TCP.

В байтовом потоке AFra передающее приложение может информировать транспорт о границах сообщений и требуемых на уровне сообщения свойствах (настройка порядка и гарантий или встраивание запроса отправки подтверждений без задержки). Когда приложение задает на уровне сообщения свойства, смягчающие требования упорядоченной доставки байтов, можно предположить, что принимающее приложение 1) способно определить границы сообщений при условии неизменности сообщений и 2) способно принять эти смягченные на уровне сообщения требования. Любая сигнализация таких данных партнеру отдается прикладным протоколам и считается выходящей за рамки этого документа.

Например, если приложение запрашивает передачу сообщений постоянного размера 100 байтов с частичными гарантиями, принимающему приложению нужно быть готовым к приему 100-байтовых блоков. Приложению также нужно быть готовым к отсутствию некоторых блоков (например, в результате применения SCTP с настраиваемыми гарантиями). При использовании TCP пропусков блоков не будет, но это верно и для приложения, а возможная задержка из-за повтора приемлема в модели обслуживания по возможности (best-effort, *См. параграф* 3.5 в [RFC7305]). Тем не менее, принимающее приложение будет делить поток байтов на 100-байтовые блоки.

Отметим, что такое использование сообщений не требует одинакового их размера. Многие прикладные протоколы используют формат TLV (Type-Length-Value - тип, размер, значение), например, определяя заголовки с полем размера или используя методы заполнения байтов, такие как COBS (Consistent Overhead Byte Stuffing) [COBS]. Если приложению нужны номера сообщений, например для восстановления порядка, это тоже должно указывать само

приложение, поскольку транспортные свойства SCTP, связанные с порядковыми номерами, не включаются в минимальный набор (в интересах применения TCP).

5.2. Планировщики потоков без потока

Выше уже отмечено, что многопоточность не требует относящихся к приложению сведений. Возможные преимущества или недостатки в результате использования, например, двух потоков ассоциации SCTP в сравнении с применением двух отдельных ассоциаций SCTP или соединений TCP связаны с информацией о сети и конкретном транспортном протоколе, а не о приложении. Однако транспортные функции «выбора планировщика для потоков в ассоциации» и «настройки приоритета или веса в планировщике» работают с потоками, которыми здесь называются каналы связи, с которыми работает планировщик и которым назначается приоритет. Кроме того, транспортные функции категории «Поддержка» работают с ассоциациями в случае SCTP, т. е. применяются ко всем потокам в рамках ассоциации.

Имея лишь такую семантику представления, интерфейс с транспортной системой можно сделать проще, если предположить, что соединения могут быть не только ассоциациями или соединениями транспортного протокола, но и потоками имеющихся ассоциаций SCTP, например. Нужно лишь разрешить способ определения возможной группировки соединений. Затем все транспортные функции поддержки можно считать работающими с группой, а не отдельными соединениями, а планировщик будет работать с соединениями внутри группы.

Для совместимости с разными транспортными протоколами и однородного предоставления доступа к транспортным соединениям и потокам протокола с множеством потоков, семантику создания и закрытия групп нужно сделать более ограничительной относительно базовых опций. Например, поддержку полузакрытых соединений TCP можно рассматривать как свойство, основанное на более строгой функции ABORT. Эта функция не может поддерживаться всеми протоколами транспортной системы (включая потоки и ассоциации), поскольку некоторые протоколы не поддерживают полузакрытых соединений.

5.3. Упреждающая передача данных

Есть две транспортных функции, связанные с упреждающей передачей сообщений (возможно многократной) - TCP Fast Open [RFC7413] и «Передача сообщения для гарантированной доставки в процессе организации соединения», которая относится к способности SCTP передавать данные вместе с блоком COOKIE-Echo. Даже без TCP Fast Open протокол TCP может передавать данные в процессе согласования вместе с пакетом SYN, однако получатель таких данных может не передать их приложению, пока согласование не завершится. В разных вариантах TCP Fast Open эти данные не разграничены как сообщение протоколом TCP (не представляются сообщением). Эта функциональность обычно доступна в TCP и поддерживается несколькими реализациями, хотя спецификация TCP не указывает способ передачи таких данных приложениям.

Транспортная система может различать передачу данных до (возможно многократную) и в процессе организации соединения. Кроме того, можно предположить, что заранее переданные данные (до завершения согласования) будут использоваться только для сообщений, явно помечаемых как «идемпотентные», т. е. воспринимаемые для многократной передачи.

Объем данных, которые можно успешно передать до или в процессе организации соединения, зависит от разных факторов, включая транспортный протокол, использование опций заголовка, выбор IPv4 или IPv6, а также Path MTU. Поэтому транспортной системе следует разрешать отправку приложению максимального объема данных, которые могли быть переданы до или в процессе организации соединения.

5.4. Работа отправителя «всухую»

Транспортная функция «уведомления отсутствия в стеке данных для передачи» относится к уведомлению SCTP SENDER DRY. Такие уведомления могут в принципе использоваться для предотвращения избыточно больших буферов передачи, сохраняя гарантию того, что транспортный отправитель всегда имеет данные для передачи. Это может обеспечивать преимущества некоторым приложениям [WWDC2015]. Однако SENDER DRY на деле означает, что весь буфер передачи (включая неотправленные и неподтвержденные данные) пуст, т. е. это уведомляет отправителя, но уже слишком поздно, когда у транспортного протокола уже нет данных для передачи. Некоторые современные реализации TCP включают отсутствующую в спецификации опцию сокета TCP_NOTSENT_LOWAT, которая предложена в [WWDC2015] и ограничивает объем неотправленных данных, которые TCP может сохранять в буфере сокета. Это позволяет задать уровень заполнения буфера, при котором сокет открывается для записи, без ожидания опустошения буфера.

SCTP позволяет также настроить размер буфера на стороне отправителя - автоматизируемая транспортная функция «настроить размер буфера передачи» обеспечивает это, но только для всего буфера, который включает неотправленные и неподтвержденные данные. SCTP не позволяет настраивать эти части раздельно. Поэтому для транспортной системы имеет смысл разрешать однотипный доступ в TCP_NOTSENT_LOWAT и уведомлениями SENDER DRY.

5.5. Профиль производительности

Транспортные функции включают:

- запрет алгоритма Nagle;
- включение и настройку LEDBAT;
- указание поля DSCP.

Все они связаны с приложениями класса QoS, такими как низкая задержка или сборка мусора (scavenger). В интересах гибкости транспортной системы они могут предлагаться в однородной, более абстрактной форме, когда транспортная система может, например, решить самостоятельно как использовать контроль перегрузок в стиле LEDBAT и выбрать определенные значения DSCP, а приложение будет лишь задавать общий профиль производительности (описание способа ее получения). Потребность в «минимально возможной задержке за счет издержек» можно тогда транслировать в автоматическое отключение алгоритма Nagle.

В некоторых случаях алгоритмом Nagle лучше управлять напрямую из приложения, поскольку он связан не только с общим профилем, но и со сведениями о размере будущих сообщений. Для тонкой настройки функциональности в стиле Nagle поддерживается функция «запроса отмены группировки сообщений».

5.6. Защита

Протоколы TCP и SCTP предлагают аутентификацию. TCP аутентифицирует полные сегменты. SCTP позволяет настраивать типы блоков SCTP, которые всегда должны аутентифицироваться и если такая возможность раскрывается, она создает нежелательную зависимость от транспортного протокола. Для совместимости с TCP транспортной системе следует позволять лишь настройку для полный пакетов транспортного уровня, включая заголовки, псевдозаголовок IP (при наличии) и данные (payload).

Защита рассматривается в отдельном документе [RFC8922]. Представленный здесь минимальный набор включает все связанные с защитой транспортные функции из Приложения A - настраиваемую аутентификацию, смену параметров аутентификации, получение данных аутентификации, установку значения Cookie life, а также задание ключа для аутентификации сообщения. Включены также транспортные функции, не указанные в приложении A, такие как приватность содержимого на промежуточных устройствах.

5.7. Размер пакета

UDP(-Lite) включает транспортную функцию задания поля DF. Это вызывает сообщения об ошибке в случае отправки сообщений размером больше Path MTU, которые нужны приложению на основе UDP для реализации механизма Path MTU Discovery (функция, которую приложения на базе UDP должны выполнять самостоятельно). Транспортная функция определения максимального размера транспортного сообщения, которое можно передать в пакете IP без фрагментации с заданного интерфейса, дает верхний предел для Path MTU (без заголовков) и может поэтому реализовать Path MTU Discovery более эффективно.

6. Минимальный набор транспортных свойств

На основе классификации, сокращения и обсуждения в разделе 3 здесь предлагается минимальный набор транспортных функций, которые конечным системам следует предлагать. Любая конфигурация на основе предлагаемого здесь минимального набора функций может быть реализована по протоколу TCP, а также обеспечивает гибкость транспортной системы в части выбора иного транспорта, если он доступен. В этом разделе термин не-UDP указывает элементы системы, которые не могут быть реализованы по протоколу UDP, тогда как элементы без такой пометки могут работать на основе транспорта UDP.

Использованы аргументы раздела 5. Обсуждение, чтобы сделать финальное представление минимального набора коротким, простым и общеупотребимым. Возможны ситуации, когда эти аргументы не применимы, например, у разработчиков могут быть свои причины раскрывать приложениям многопоточность или ограничение семантики создания и завершения соединений может оказаться проблематичным. В таких случаях следует рассмотреть представления раздела 4. Сокращенный набор транспортных функций.

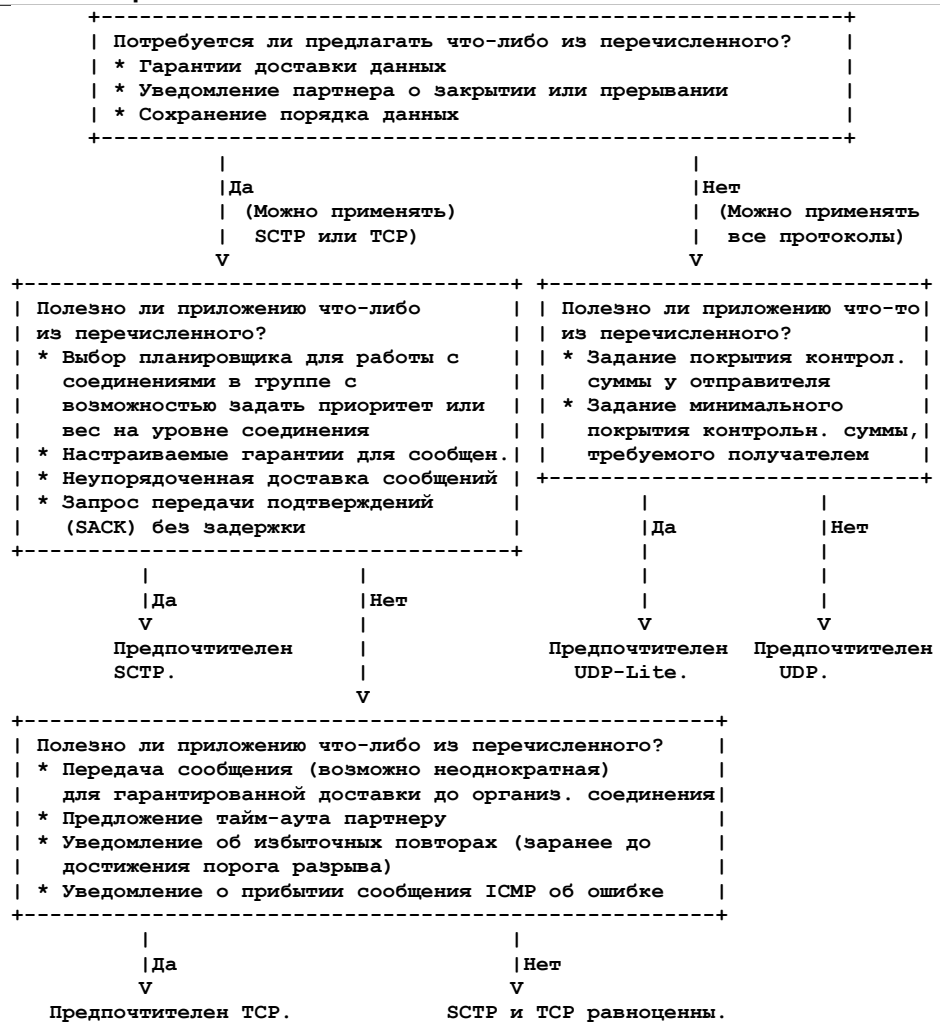
Как в разделах 3, 4 и [RFC8303] минимальный набор транспортных функций разделен на категории, связанные с 1) соединениями (организация, доступность, поддержка, завершение) и 2) передачей данных (отправка данных, прием данных, ошибки). Здесь основное внимание уделено соединениям, которые транспортные системы в абстрактной форме предлагают приложениям, в отличие от соединений транспортных протоколов, используемых транспортной системой.

6.1. Организация, доступность, завершение

Сначала должно быть «организовано» соединение, чтобы можно было выполнить ту или иную начальную настройку, прежде чем транспортная система сможет организовать пассивное или активное взаимодействие с удаленной конечной системой. При настройке вновь созданного соединения приложение может отказаться от использования MPTCP. Кроме того, все параметры конфигурации из параграфа 6.2 могут применяться изначально, хотя некоторые из них начнут работать лишь после организации соединения с выбранным транспортным протоколом. Ранняя настройка соединения помогает транспортной системе принять верное решение. Например, данные о группировке могут повлиять на решение вопроса о реализации соединения в форме потока многопоточного протокола в имеющейся ассоциации.

Для разгруппированных соединений требуется настройка заранее, поскольку это позволяет транспортной системе узнать, какие протоколы следует пытаться использовать. В частности, транспортная система, которая выбирает протокол лишь однократно, должна знать заранее строгие требования, которые нужно выполнить, иначе может возникнуть тупиковая ситуация (например, выбор протокола UDP и последующий запрос гарантированной доставки). В качестве примера корректной обработки ситуаций ниже представлено дерево решений, выведенное из параграфа 4.1, но без аутентификации, как указано в разделе 8.

Отметим, что это дерево решений не является оптимальным для всех случаев. Например, если приложение хочет использовать задание области покрытия контрольной суммы отправителем, которое предлагает лишь UDP-Lite, и настройку приоритета или веса для планировщика, которая доступна только в SCTP, выбор в соответствии с представленным деревом всегда приводит к UDP-Lite, делая невозможным использование планировщиков SCTP с планировщиками по приоритету внутри группы соединений. Есть и другие факторы, способные влиять на выбор протокола (за или против), например уровень распространенности, возможность работы через NAT и т. п. Разработчикам следует принимать во внимание все компромиссные требования, указанные в параграфе 4.1, при выборе способа инициализации соединения.



В заключение приводится список параметров, служащих входными для транспортной системы при выборе и настройке подходящего протокола.

Reliability (надежность)
Логическая переменная, в которой следует устанавливать значение true, когда приложению полезно использовать что-либо из числа перечисленного: надежная доставка данных, уведомление партнера о закрытии или разрыве соединения, упорядоченность данных.

Checksum coverage (покрытие контрольной суммы)
Логическая переменная, указывающая пользу задания области покрытия контрольной суммы при отправке или получении.

Configure message priority (настраиваемый приоритет сообщений)
Логическая переменная, в которой следует устанавливать значение true, когда приложению полезно использовать что-либо из числа перечисленных механизмов настройки или приоритизации на уровне сообщения: выбор планировщика для работы в рамках группового соединения с возможностью настрйоки для соединения приоритета или веса, настраиваемая надежность доставки сообщения, неупорядоченная доставка сообщений, запрос отправки подтверждений (SACK) без задержки.

Early message timeout notifications (упреждающее информирование о тайм-ауте)
Логическая переменная, в которой следует устанавливать значение true, когда приложению полезно использовать что-либо из числа перечисленного: передача сообщения для гарантированной доставки (возможно многократная) до организации соединения, указание (предложение) тайм-аута партнеру, уведомление об избыточных повторах (заранее до достижения порога разрыва соединения), уведомление о приеме сообщения ICMP об ошибке.

После организации соединения для него можно запросить максимальный объем данных, для которых приложение может ожидать гарантированную передачу до или в процессе организации транспортного соединения (с возможностью указать 0), как указано в параграфе 6.2.1. Приложение также может передавать в соединение сообщения для гарантированной доставки до или в процессе организации соединения (не-UDP), после чего транспортная система будет передавать их как можно раньше. Приложение может облегчить отправку сообщения как можно раньше, пометив его как «идемпотентное» (6.3.1. Передача) и в этом случае принимающая сторона должна быть гтова к возможности получения нескольких копий сообщения (поскольку идемпотентные сообщения передаются с гарантиев, запрос идемпотентности не требуется для систем, поддерживающих UDP).

После создания транспортная система может активно организовать соединение с партнером или пассивно прослушивать входящие вызовы. Отметим, что активная организация соединения может (но не обязательно) вызывать уведомление слушающей стороны. Возможно получение слушающей стороной уведомления вместе с первыми данными от активной стороны (транспортная система на приемной стороне может обрабатывать это, продолжая блокировать вызов Listen, за которым следует, например, вызов Receive; реализации callback могут просто пропустить эквивалент Listen). Это также означает, что активная сторона считается первой в передаче данных.

Транспортная система может активно закрыть соединение, т. е. прервать его после гарантированной доставки партнеру всех оставшихся данных (если раньше была запрошена гарантированная доставка, не UDP) и в этом случае партнер уведомляется о закрытии соединения. Кроме того, соединение может быть разорвано без доставки партнеру оставшихся данных. В случае запрошенной ранее гарантированной или почти гарантированной доставки (не-UDP)

партнер уведомляется о разрыве соединения. Может быть задан тайм-аут для разрыва соединения с случае, когда данные не доставляются слишком долго (не-UDP), однако при разрыве по тайм-ауту партнер не уведомляется о разрыве соединения. Поскольку полуоткрытые соединения не поддерживаются, при получении реализующим транспортную систему хостом уведомления от партнера о закрытии или разрыве соединения (не-UDP) может оказаться невозможным прочтение остающихся данных. Это значит, что неподтвержденные данные в буфере передачи транспортной системы могут быть отброшены из буфера при получении от партнера уведомления close или abort.

6.2. Поддержка

Транспортная система должна предлагать способы группировки соединений, но это не может гарантировать их правильную группировку с помощью используемых транспортных протоколов (например, невозможно гарантировать мультиплексирование соединений как потоков одной ассоциации SCTP, когда протокол SCTP может быть недоступен). Поэтому транспортная система должна тем или иным способом обеспечить корректную обработку конфигураций с группами и без таковых (например, применяя конфигурацию ко всем сгруппированным соединениям, даже если они не мультиплексируются, или информируя приложение об успехе или неудаче группировки).

Как общее правило, любую из описанных ниже конфигураций следует применять как можно раньше, чтобы помочь транспортной системе принять решение.

6.2.1. Группы соединений

Перечисленные ниже транспортные свойства и уведомления (некоторые напрямую из раздела 4, иные новые или измененные на основе раздела 5) автоматически применяются ко всем сгруппированным соединениям.

Настройка тайм-аута (не-UDP)

Это может выполняться со следующими параметрами:

- значение тайм-аута для разрыва соединения (в секундах);
- предлагаемое партнеру (если возможно) значение тайм-аута (в секундах);
- число повторов, по достижении которого приложение следует информировать об избыточных повторах (Excessive Retransmissions).

Настройка важности (срочности)

Это может выполняться с указанными ниже параметрами.

- Число для идентификации планировщика, который следует использовать для работы с соединениями внутри группы (без гарантий). Планировщики определены в [RFC8260].
- Номер «профиля производительности» для указания способа использования приложением доступной производительности. Вариантами могут быть «наименьшая возможная задержка за счет накладных расходов» (отключает любой алгоритм в стиле Nagle), «сборка мусора» (scavenger) или иные значения, помогающие определить DSCP для соединения.
- Предельный размер буфера (в байтах). Приложение может уведомляться, когда у отправителя в буфере объем данных меньше установленного предела. Уведомления не гарантируются и транспортная система не обязана поддерживать предельный размер буфера больше 0. Отметим, что этот предел и уведомления следует применять для буферов всей транспортной системы, т. е. для любых возможных буферов, которые сама транспортная система может применять «поверх» транспортного буфера передачи.

В соответствии с параграфом 5.7 можно запросить перечисленные ниже свойства.

- *Максимальный размер сообщений*, которые можно передать без фрагменттации через настроенный интерфейс. Транспортная система не обязана предоставлять это свойство и может возвращать ошибку (not available). Это свойство может помочь приложениям, реализующим Path MTU Discovery.

- *Максимальный размер транспортного сообщения*, которое можно передать (в байтах). Независимо от фрагментации имеется предельный размер для сообщений, обслуживаемых SCTP или UDP(-Lite). Поскольку предоставляемые транспортной системой услуги не зависят от транспортного протокола, система должна позволять приложениям запрашивать это значение - максимальный размер сообщения в кадрированном приложением потоке байтов (AFra, параграф 5.1). Система может возвращать ошибку (not available), если данные не разграничиваются.

- *Максимальный размер транспортного сообщения*, которое может быть принято от настроенного интерфейса (в байтах) или not available.

- *Максимальный объем данных*, которые можно передать до организации соединения (в байтах).

В дополнение к упомянутым уведомлениям о закрытии/прерывании соединения и возможных ошибках передачи могут передаваться перечисленные ниже уведомления.

Excessive Retransmissions - избыточные повторы передачи

Достигнуто настроенное (или принятое по умолчанию) число повторов передачи, что является упреждающим разрыв соединения уведомлением.

ICMP Arrival - прибытие ICMP (параметром является сообщение ICMP)

Прибыл пакет ICMP, содержащий сообщение ICMP.

ECN Arrival - прибытие ECN (параметром является значение ECN)

Прибыл пакет с явным уведомлением о перегрузке (Explicit Congestion Notification или ECN). Это может быть полезно для приложений с контролем перегрузки.

Timeout - тайм-аут (параметром служит число секунд)

Данные не могут быть доставлены в течение s секунд.

Drain:

Буфер передачи был опустошен ниже заданного предела или полностью. Это базовое уведомление, которое пытается включить унифицированный доступ к TCP_NOTSENT_LOWAT, а также уведомлению SENDER DRY (как описано в параграфе 5.4, SCTP SENDER DRY является особым случаем, где порог для неотправленных данных равен 0 и в буфере передачи не осталось неподтвержденных данных).

6.2.2. Отдельные соединения

Настройка приоритета или веса для планировщика в соответствии с [RFC8260].

Настройка использования контрольных сумм, которая может быть выполнена с указанными ниже параметрами, но без гарантии выполнения каких-либо ограничений для контрольных сумм (по умолчанию контрольные суммы включены и обеспечивают полное покртытие).

- Логическое значение для выключение или выключения контрольной суммы при передаче.
- Желаемая область покрытия контрольной суммой (в байтах) при передаче.
- Логическое значение для выключение или выключения контрольной суммы при получении.
- Требуемая минимальная область покрытия контрольной суммой (в байтах) при получении.

6.3. Обмен данными

6.3.1. Передача

При передаче сообщения партнеру не дается гарантий сохранения границ сообщения. Если границы нужны, принимающее приложение у партнера должно знать о них заранее (иначе транспортная система не сможет использовать TCP). Отметим, что приложению следует уже быть готовым к передаче данных до того, как транспортная система организует соединение с выбранным транспортным протоколом. Применительно к передаваемого сообщения могут использоваться приведенные ниже параметры.

Reliability - гарантия доставки
: Этот параметр служит для указания выбора - полностью надежная доставка с контролем перегрузки (не-UDP), доставка без гарантий и контроля перегрузки, доставка без гарантий с контролем перегрузки (не-UDP), частичные гарантии с контролем перегрузки (см. [RFC3758] и [RFC7496], где приведено описание частичных гарантий) (не-UDP). Два последних варианта транспортная система не обязана предлагать и это может приводить к полной гарантии. Отметим, что приложению, передающему данные без гарантий и контроля перегрузок, следует самому контролировать перегрузки в соответствии с [RFC8085].

Ordered - упорядоченность (не-UDP)
: Логическое значение, позволяющее приложению выбрать упорядоченную (true) или неупорядоченную (false), но возможно более быструю доставку сообщений.

Bundle - связка (группировка)
: Логическое значение, разрешающее (true) или запрещающее (false) группировать сообщения. Без гарантий.

DelAck - задержка подтверждения
: Логическое значение, позволяющее приложению запросить (false) у партнера отправку подтверждения доставки этого сообщения без задержки.

Fragment - фрагментирование
: Логическое значение, разрешающее (true) или запрещающее (false) фрагментировать сообщения на уровне IP. Без гарантий.

Idempotent - идемпотентность (не-UDP)
: Логическое значение, указывающее является (true) или нет (false) данное сообщение идемпотентным. Идемпотентные сообщения могут приходить к получателю в нескольких экземплярах (не менее 1). Идемпотентные данные могут использоваться получателем непосредственно при попытке организации соединения. Таким образом, при передачи данных до организации транспортной системой соединения с выбранным транспортным протоколом указание идемпотентности данных передающей стороной облегчает их передачу партнеру на приемной стороне как можно раньше.

Приложение может уведомляться об отказе при передаче конкретного сообщения. Такие уведомления не гарантируются, т. е. отказ может происходить «тихо».

6.3.2. Прием

Принимающее приложение получает поток байтов AFra (*См. параграф* 5.1). В соответствии с семантикой получателя TCP поток AFra для получателя является лишь потоком байтов. Если отправитель задал границы сообщений, транспортная система на приемной стороне, реализующая лишь минимальный набор определенных здесь транспортных услуг, не будет информировать приложение о границах это ограничение требуется лишь для транспортных систем, реализованных для прямого использования TCP).

В отличие от семантики TCP, если передающее приложение разрешило передачу сообщений без полной гарантии или с нарушением порядка, это может влиять на каждое прибывающее сообщение. Сообщения не будут затронуты, т. е. при наличии в конце принимаемого блока неполного сообщения, оно гарантированно будет продолжено в следующем полученном блоке данных.

7. Взаимодействие с IANA

Этот документ не требует действий со стороны IANA.

8. Вопросы безопасности

Аутентификация, защита конфиденциальности и целостности указаны как транспортные свойства в [RFC8095]. Зачастую эти свойства обеспечиваются протоколом или уровнем над транспортным протоколом. Ни один из полнофункциональных стандартных протоколов из [RFC8303], на которых базируется этот документ, не поддерживает все эти транспортные функции, поэтому они не рассматриваются в данном документе, за исключением естественных функций аутентификации в TCP и SCTP, к которым применимо рассмотрение вопросов безопасности в [RFC5925] и [RFC4895]. Минимальные требования к защищенной транспортной системе рассмотрены в отдельном документе [RFC8922].

9. Литература

9.1. Нормативные документы

[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>.

[RFC8303] Welzl, M., Tuexen, M., and N. Khademi, "On the Usage of Transport Features Provided by IETF Transport Protocols", RFC 8303, DOI 10.17487/RFC8303, February 2018, <https://www.rfc-editor.org/info/rfc8303>.

[RFC8922] Enghardt, T., Pauly, T., Perkins, C., Rose, K., and C. Wood, "A Survey of the Interaction between Security Protocols and Transport Services", RFC 8922, DOI 10.17487/RFC8922, October 2020, <https://www.rfc-editor.org/info/rfc8922>.

9.2. Дополнительная литература

[COBS] Cheshire, S. and M. Baker, "Consistent overhead byte stuffing", IEEE/ACM Transactions on Networking, Volume 7, Issue 2, DOI 10.1109/90.769765, April 1999, <https://doi.org/10.1109/90.769765>.

[POSIX] The Open Group, "IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7", (Revision of IEEE Std 1003.1-2008), IEEE Std 1003.1-2017, January 2018, <https://www.opengroup.org/onlinepubs/9699919799/functions/contents.html>.

[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>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <https://www.rfc-editor.org/info/rfc4987>.

[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>.

[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>.

[RFC7305] Lear, E., Ed., "Report from the IAB Workshop on Internet Technology Adoption and Transition (ITAT)", RFC 7305, DOI 10.17487/RFC7305, July 2014, <https://www.rfc-editor.org/info/rfc7305>.

[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>.

[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>.

[RFC8622] Bless, R., "A Lower-Effort Per-Hop Behavior (LE PHB) for Differentiated Services", RFC 8622, DOI 10.17487/RFC8622, June 2019, <https://www.rfc-editor.org/info/rfc8622>.

[SCTP-STREAM-1] Weinrank, F. and M. Tuexen, "Transparent Flow Mapping for NEAT", IFIP Networking 2017, Workshop on Future of Internet Transport (FIT 2017), June 2017.

[SCTP-STREAM-2] Welzl, M., Niederbacher, F., and S. Gjessing, "Beneficial Transparent Deployment of SCTP: The Missing Pieces", IEEE GlobeCom 2011, DOI 10.1109/GLOCOM.2011.6133554, December 2011, <https://doi.org/10.1109/GLOCOM.2011.6133554>.

[TAPS-INTERFACE] Trammell, B., Welzl, M., Enghardt, T., Fairhurst, G., Kuehlewind, M., Perkins, C., Tiesel, P. S., Wood, C. A., and T. Pauly, "An Abstract Application Layer Interface to Transport Services", Work in Progress, Internet-Draft, draft-ietf-taps-interface-09, 27 July 2020, <https://tools.ietf.org/html/draft-ietf-taps-interface-09>.

[WWDC2015] Lakhera, P. and S. Cheshire, "Your App and Next Generation Networks", Apple Worldwide Developers Conference 2015, San Francisco, USA, June 2015, <https://developer.apple.com/videos/wwdc/2015/?id=719>.

Приложение A. Расширенное множество транспортных функций

В этом приложении представлены транспортные свойства в соответствии с номенклатурой «Категория.[Субкатегория].Свойство.Протокол» (CATEGORY.[SUBCATEGORY].FEATURENAME.PROTOCOL), эквивалентной этапу 2 в [RFC8303]. Очерчена также реализация транспортной системой транспортных свойств (функций). «Минимальный

набор», заданный в этом документе, предназначен для «односторонней» реализации по протоколу TCP и (с некоторыми ограничениями) UDP. Поэтому для всех транспортных функций, отнесенных к категории «функциональные» или «оптимизирующие», для которых нет соответствующих примитивов TCP и/или UDP на этапе 2 в [RFC8303], приведено краткое описание реализации на основе TCP и/или UDP.

Некоторые транспортные функции отмечены как «автоматизируемые» на основе широкого решения, влияющего на несколько транспортных функций, как указано ниже.

- Большинство транспортных функций, связанных с многопоточностью, помечены как автоматизируемые, поскольку принятие решения об использовании множества потоков не требует знаний о конкретном приложении. Это значит, что соединение, предоставляемое приложению, может быть реализовано с использованием одного потока ассоциации SCTP вместо использования всей ассоциации или соединения TCP. Это можно сделать путем использования более одного потока в момент создания ассоциации SCTP (параметр CONNECT.SCTP - число исходящих потоков), поддержки внутренних номеров потоков и их применения при передаче данных (параметр SEND.SCTP - номер потока). Закрытие или разрыв соединения может просто освобождать номер потока для последующего использования (*См. параграф* 5.2).

- Все транспортные функции, кроме отключения MPTCP, связанные с использованием нескольких путей или выбором сетевого интерфейса, считаются автоматизируемыми. Например, Listen всегда может прослушивать все доступные интерфейсы, а Connect - использовать принятый по умолчанию интерфейс для целевого адреса IP.

В трех случаях транспортные функции в приведенном ниже описании были собраны вместе или слегка изменены по сравнению с [RFC8303] и эти отличия специально помечены. Они не добавляют новых функций, а просто представляют некий пересмотр, который поможет упростить процесс вывода (например, за счет удаления параметров для приложений, которые могут не заботиться о своем выборе. Соответствующие транспортные функции являются автоматизируемыми и они помечены ниже как отличающиеся от RFC 8303.

A.1. Связанные с соединением транспортные свойства

Организация

Connect

Протоколы: TCP, SCTP, UDP(-Lite).

Функционально, поскольку понятие соединения часто отражается в приложениях как ожидание возможности обмениваться данными после успешного вызова Connect с коммуникационной последовательностью, относящейся к этому транспортному свойству, которое определено протоколом приложения.

Реализация: CONNECT.TCP, CONNECT.SCTP, CONNECT.UDP(-Lite).

Задание опций IP, которые должны применяться всегда

Протоколы: TCP, UDP(-Lite).

Автоматизируемо, поскольку опции IP относятся к информации о сети, а не о приложении.

Запрос множества потоков

Протоколы: SCTP.

Автоматизируемо, поскольку применение множества потоков не требуется связанного с приложением знания (примеры реализации множества потоков без участия приложения даны в [SCTP-STREAM-1] и [SCTP-STREAM-2]).

Реализация: См. параграф 5.2.

Ограничение числа входящих потоков

Протоколы: SCTP.

Автоматизируемо, поскольку применение множества потоков не требуется связанного с приложением знания.

Реализация: См. параграф 5.2.

Задание числа попыток и/или тайм-аутов для первого сообщения при организации

Протоколы: TCP, SCTP.

Функционально, поскольку тесно связано с предполагаемыми сведениями о гарантиях доставки для данных, отправленных до или в процессе организации соединения.

Реализация: Используются параметры CONNECT.TCP и CONNECT.SCTP.

Реализация на основе UDP: Ничего (безотносительно к варианту UDP, поскольку гарантии не предполагаются).

Получение множества сокетов

Протоколы: SCTP.

Автоматизируемо, поскольку непараллельное применение множества путей для связи между теми же конечными хостами связано со знаниями о сети, а не о приложении.

Отключение MPTCP

Протоколы: MPTCP.

Оптимизируемо, поскольку непараллельное применение множества путей для связи между теми же конечными хостами может повышать производительность. Применение этого свойства зависит от информации о сети и приложении (См. параграф 3.1 of [RFC6897]).

Реализация: С помощью логического параметра в CONNECT.MPTCP.

Реализация с TCP: Ничего.

Реализация с UDP: Ничего.

Настройка аутентификации

Протоколы: TCP, SCTP.

Функционально, поскольку напрямую влияет на безопасность.

Реализация: Через параметры в CONNECT.TCP и CONNECT.SCTP. С TCP это позволяет настроить кортежи первичных ключей (Master Key Tuple или MKT) для аутентификации полных сегментов (включая псевдозаголовок TCP IPv4, заголовок и данные TCP). В SCTP это позволяет задать типы аутентифицируемых блоков. Аутентификация лишь определенных типов блоков снижает уровень защиты, что не поддерживается в TCP. Для совместимости следует разрешать лишь аутентификацию всех блоков. Способ предоставления ключевого материала должен соответствовать [RFC4895] и [RFC5925].

Реализация с UDP: Невозможна (UDP не предлагает такой функциональности).

Указание (и/или получение по завершении) уровня адаптации с помощью его кода

Протоколы: SCTP.

Функционально, поскольку позволяет передать дополнительные данные для определения уровня адаптации, который сам по себе зависит от приложения.

Реализация: С помощью параметра в CONNECT.SCTP.

Реализация с TCP: Невозможна (TCP не предлагает такой функциональности).

Реализация с UDP: Невозможна (UDP не предлагает такой функциональности).

Запрос согласования для чередования пользовательских сообщений

Протоколы: SCTP.

Автоматизируемо, поскольку требует использования множества потоков, а запрос множества потоков в категории CONNECTION.ESTABLISHMENT является автоматизируемым.

Реализация: Контролируется через параметр в CONNECT.SCTP. Одной из возможных реализаций является попытка использовать чередования всегда.

Отправка сообщения с гарантией доставки (возможно несколько раз) до организации соединения

Протоколы: TCP.

Функционально по причине тесной связи со свойствами данных, которые приложение передает или хочет принять.

Реализация: Через параметр в CONNECT.TCP.

Реализация с UDP: Невозможна (UDP не предлагает такой функциональности).

Отправка сообщения с гарантией доставки в процессе организации соединения

Протоколы: SCTP.

Функционально, поскольку может работать лишь с сообщениями ограниченного размера, что задает тесную связь со свойствами данных, которые приложение передает или хочет принять.

Реализация: Через параметр в CONNECT.SCTP.

Реализация с TCP: Передача сообщения в пакете SYN без возможности указать границы сообщения.

Реализация с UDP: Невозможна (UDP не предлагает такой функциональности).

Включение UDP-инкапсуляции с заданным удаленным портом UDP

Протоколы: SCTP.

Автоматизируема, поскольку инкапсуляция UDP относится к знанию о сети, а не о приложении.

Доступность

Listen

Протоколы: TCP, SCTP, UDP(-Lite).

Функционально, поскольку восприятие запросов соединения часто отражается в приложениях как ожидание возможности обмениваться данными после успешного вызова Listen с коммуникационной последовательностью, относящейся к этому транспортному свойству, которое определено протоколом приложения.

RFC 8303 отличается от 3 автоматизируемых транспортных свойств, приведенных ниже, в том, что выбор интерфейса для прослушивания остается открытым.

Реализация: Прослушивание на всех интерфейсах с помощью LISTEN.TCP (не указан локальный IP-адрес) или LISTEN.SCTP (указаны пары «порт SCTP - адрес» для всех локальных адресов IP). LISTEN.UDP(-Lite) поддерживает оба метода.

Listen, 1 локальный интерфейс

Протоколы: TCP, SCTP, UDP(-Lite).

Автоматизируемо, поскольку решения о локальных интерфейсах связаны со знанием о сети и операционной системе, а не о приложении.

Listen, N локальных интерфейсов

Протоколы: SCTP.

Автоматизируемо, поскольку решения о локальных интерфейсах связаны со знанием о сети и операционной системе, а не о приложении.

Listen, все локальные интерфейсы

Протоколы: TCP, SCTP, UDP(-Lite).

Автоматизируемо, поскольку решения о локальных интерфейсах связаны со знанием о сети и операционной системе, а не о приложении.

Задание опций IP, которые должны применяться всегда

Протоколы: TCP, UDP(-Lite).

Автоматизируемо, поскольку опции IP относятся к знанию о сети, а не о приложении.

Отключение MPTCP

Протоколы: MPTCP.

Оптимизируемо, поскольку непараллельное применение множества путей для связи между теми же конечными хостами может повышать производительность. Применение этого свойства зависит от информации о сети и приложении (См. параграф 3.1 of [RFC6897]).

Реализация: Через логический параметр в LISTEN.MPTCP.

Реализация с TCP: Ничего.

Реализация с UDP: Ничего.

Настройка аутентификации

Протоколы: TCP, SCTP.

Функционально по причине прямого влияния на безопасность.

Реализация: Через параметры в LISTEN.TCP и LISTEN.SCTP.

Реализация с UDP: Невозможна (UDP не предлагает такой функциональности).

Реализация с TCP: С TCP это позволяет настроить кортежи первичных ключей (Master Key Tuple или MKT) для аутентификации полных сегментов (включая псевдозаголовок TCP IPv4, заголовок и данные TCP). В SCTP это позволяет задать типы аутентифицируемых блоков. Аутентификация лишь определенных типов блоков снижает уровень защиты, что не поддерживается в TCP. Для совместимости следует разрешать лишь аутентификацию всех блоков. Способ предоставления ключевого материала должен соответствовать [RFC4895] и [RFC5925].

Реализация с UDP: Невозможна (UDP не предлагает аутентификации).

Получение запрошенного числа потоков

Протоколы: SCTP.

Автоматизируемо, поскольку использование множества потоков не требует знания о приложении.

Реализация: См. параграф 5.2.

Ограничение числа входящих потоков

Протоколы: SCTP.

Автоматизируемо, поскольку использование множества потоков не требует знания о приложении.
Реализация: См. параграф 5.2.

Указание (и/или получение по завершении) кода уровня адаптации

Протоколы: SCTP.
Функционально, поскольку позволяет передать дополнительные данные для определения уровня адаптации, который сам по себе зависит от приложения.
Реализация: С помощью параметра в LISTEN.SCTP.
Реализация с TCP: Невозможна (TCP не предлагает такой функциональности).
Реализация с UDP: Невозможна (UDP не предлагает такой функциональности).

Запрос согласования для чередования пользовательских сообщений

Протоколы: SCTP.
Автоматизируемо, поскольку требует множество потоков, а запрос множества потоков в категории CONNECTION.ESTABLISHMENT является автоматизируемым.
Реализация: Через параметр в LISTEN.SCTP.

Поддержка

Смена тайм-аута для разрыва соединения (время или число повторов)

Протоколы: TCP, SCTP.
Функционально по причине тесной связи с предполагаемой гарантией доставки данных.
Реализация: Через CHANGE_TIMEOUT.TCP или CHANGE_TIMEOUT.SCTP.
Реализация с UDP: Невозможна (UDP не дает гарантий и не поддерживает тайм-аут для соединения).

Предложенный партнеру тайм-аут

Протоколы: TCP.
Функционально по причине тесной связи с предполагаемой гарантией доставки данных.
Реализация: Через CHANGE_TIMEOUT.TCP.
Реализация с UDP: Невозможна (UDP не дает гарантий и не поддерживает тайм-аут для соединения).

Отключение алгоритма Nagle

Протоколы: TCP, SCTP.
Оптимизируемо, поскольку это решение зависит от размера будущих блоков и задержки между ними.
Реализация: Через DISABLE_NAGLE.TCP и DISABLE_NAGLE.SCTP.
Реализация с UDP: Ничего (UDP не реализует алгоритм Nagle).

Запрос немедленной передачи heartbeat с возвратом результата

Протоколы: SCTP.
Автоматизируемо, поскольку информирует о знании, относящемся к сети.

Уведомление об избыточных повторах (ранее до достижения порога разрыва)

Протоколы: TCP.
Оптимизируемо, поскольку это раннее предупреждение приложения, информирующее о приближающемся функциональном событии.
Реализация: Через ERROR.TCP.
Реализация с UDP: Ничего (нет порога разрыва соединения).

Добавление пути

Протоколы: MPTCP, SCTP.
Параметры MPTCP: source-IP, source-Port, destination-IP, destination-Port.
Параметры SCTP: Локальный адрес IP.
Автоматизируемо, поскольку выбор пути для коммуникаций между одной парой конечных хостов относится к знанию о сети, а не о приложении.

Удаление пути

Протоколы: MPTCP, SCTP.
Параметры MPTCP: source-IP, source-Port, destination-IP, destination-Port.
Параметры SCTP: Локальный адрес IP.
Автоматизируемо, поскольку выбор пути для коммуникаций между одной парой конечных хостов относится к знанию о сети, а не о приложении.

Задание основного пути

Протоколы: SCTP.
Автоматизируемо, поскольку выбор пути для коммуникаций между одной парой конечных хостов относится к знанию о сети, а не о приложении.

Предложение основного пути партнеру

Протоколы: SCTP.
Автоматизируемо, поскольку выбор пути для коммуникаций между одной парой конечных хостов относится к знанию о сети, а не о приложении.

Настройка переключения пути

Протоколы: SCTP.
Автоматизируемо, поскольку выбор пути для коммуникаций между одной парой конечных хостов относится к знанию о сети, а не о приложении.

Определение статуса (запрос или уведомление)

Протоколы: SCTP, 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).
Автоматизируемо, поскольку эти параметры относятся к знанию о сети, а не о приложении.

Задание поля DSCP

Протоколы: TCP, SCTP, UDP(-Lite).
Оптимизируемо, поскольку выбор подходящего значения DSCP требует сведений о приложении.
Реализация: Через SET_DSCP.TCP, SET_DSCP.SCTP, SET_DSCP.UDP(-Lite).

Уведомление о приеме сообщения ICMP об ошибке

Протоколы: TCP, UDP(-Lite).

Оптимизируемо, поскольку эти сообщения могут информировать об успехе или отказе функционального транспортного свойства (например, host unreachable относится к Connect).

Реализация: Через ERROR.TCP или ERROR.UDP(-Lite.).

Obtain information about interleaving support

Протоколы: SCTP.

Автоматизируемо, поскольку требует множество потоков, а запрос множества потоков в категории CONNECTION.ESTABLISHMENT является автоматизируемым.

Реализация: Через STATUS.SCTP.

Смена параметров аутентификации

Протоколы: TCP, SCTP.

Функционально по причине прямого влияния на безопасность.

Реализация: Через SET_AUTH.TCP и SET_AUTH.SCTP.

Реализация с TCP: В SCTP это позволяет настраивать key_id, key и hmac_id. В TCP это позволяет менять предпочтительный исходящий (current_key) и предпочтительный входящий (rnext_key) кортеж MKT для сегментов в соединении. Ключевой материал должен предоставляться способом, совместимым с [RFC4895] и [RFC5925].

Реализация с UDP: Невозможна (UDP не поддерживает аутентификацию).

Получение данных аутентификации

Протоколы: SCTP.

Функционально, поскольку решения при проверке подлинности может принимать партнер, а это влияет на зависящие от приложения меры по обеспечению уровня защиты.

Реализация: Через GET_AUTH.SCTP.

Реализация с TCP: В SCTP это позволяет получить key_id и список блоков. В TCP это позволяет получить current_key и rnext_key из принятого ранее сегмента. Ключевой материал должен предоставляться способом, совместимым с [RFC4895] и [RFC5925].

Реализация с UDP: Невозможна (UDP не поддерживает аутентификацию).

Сброс потока

Протоколы: SCTP.

Автоматизируемо, поскольку применение множества потоков не требует знания о приложении.

Реализация: См. параграф 5.2.

Уведомление о сбросе потока

Протоколы: SCTP.

Автоматизируемо, поскольку применение множества потоков не требует знания о приложении.

Реализация: См. параграф 5.2.

Сброс ассоциации

Протоколы: SCTP.

Автоматизируемо, поскольку для решения о сбросе ассоциации не требует знания о приложении.

Реализация: Через RESET_ASSOC.SCTP.

Уведомление о сбросе ассоциации

Протоколы: SCTP.

Автоматизируемо, поскольку это уведомление не требует знания о приложении.

Добавление потоков

Протоколы: SCTP.

Автоматизируемо, поскольку применение множества потоков не требует знания о приложении.

Реализация: См. параграф 5.2.

Уведомление о добавлении потоков

Протоколы: SCTP.

Автоматизируемо, поскольку применение множества потоков не требует знания о приложении.

Реализация: См. параграф 5.2.

Выбор планировщика для потоков в ассоциации

Протоколы: SCTP.

Оптимизируемо, поскольку решение о выборе планировщика требует знаний о приложении. Однако, если транспортная система не будет выбирать планировщик или сама настроит выбор некорректно, это повлияет лишь на производительность доставки данных и результат останется корректным в рамках модели best effort.

Реализация: С использованием SET_STREAM_SCHEDULER.SCTP.

Реализация с TCP: Ничего (потоки недоступны в TCP и нет гарантий влияния этого транспортного свойства).

Реализация с UDP: Ничего (потоки недоступны в UDP и нет гарантий влияния этого транспортного свойства).

Настройка приоритета или веса для планировщика

Протоколы: SCTP.

Оптимизируемо, поскольку выбор приоритета или веса требует знания о приложении. Однако, если транспортная система не будет выбирать планировщик или сама настроит выбор некорректно, это повлияет лишь на производительность доставки данных и результат останется корректным в рамках модели best effort.

Реализация: С использованием CONFIGURE_STREAM_SCHEDULER.SCTP.

Реализация с TCP: Ничего (потоки недоступны в TCP и нет гарантий влияния этого транспортного свойства).

Реализация с UDP: Ничего (потоки недоступны в UDP и нет гарантий влияния этого транспортного свойства).

Настройка размера буфера передачи

Протоколы: SCTP.

Автоматизируемо, поскольку это решение связано со знанием о сети и операционной системе, а не о приложении (см. параграф 5.4).

Настройка размера приемного буфера (и rwnd)

Протоколы: SCTP

Автоматизируемо, поскольку это решение связано со знанием о сети и операционной системе, а не о приложении.

Настройка фрагментации сообщений

Протоколы: SCTP.

Автоматизируемо, поскольку это решение связано со знанием о сети и операционной системе, а не о приложении. Отметим, что это свойство SCTP не контролирует фрагментацию на уровне IP и относится лишь к фрагментации сообщений протоколом SCTP в конечной системе.

Реализация: Всегда выполняется путем включения через CONFIG_FRAGMENTATION.SCTP и автоматической настройки размера фрагментов в соответствии с условиями в сети и операционной системе.

Настройка PMTUD

Протоколы: SCTP.

Автоматизируемо, поскольку механизм Path MTU Discovery связан со знанием о сети, а не о приложении.

Настройка таймера задержки SACK

Протоколы: SCTP.

Автоматизируемо, поскольку решение принимающей стороны о задержке отправки SACK связано со знанием о сети, а не о приложении (для передающего приложения может относиться ка запросу не задерживать SACK, но это другое транспортное свойство).

Установка значения Cookie life

Протоколы: SCTP.

Функционально, поскольку относится к безопасности (возможно снижение защиты при продолжительном использовании cookie), а не ко времени между попытками организации соединения. Эта информация может зависеть от приложения.

Реализация с TCP: Ближайшей похожей функциональностью TCP является cookie в TCP Fast Open. В [RFC7413] сказано, что сервер может в любой момент счесть cookie устаревшим в целях повышения уровня защиты, а в параграфе 4.1.2 [RFC7413] приведен пример реализации, где обновление ключа на стороне сервера ведет к завершению строка действия cookie. Для реализаций, не поддерживающих TCP Fast Open, это транспортное свойство может влияет на действительность SYN cookie (см. параграф 3.6 в [RFC4987]).

Реализация с UDP: Невозможна (UDP не поддерживает такой функциональности).

Установка максимального пика

Протоколы: SCTP.

Автоматизируемо, поскольку связано со знанием о сети, а не о приложении.

Настройка размера сообщений для частичной доставки

Протоколы: SCTP.

Функционально по причине тесной связи со свойствами данных, которые приложение передает или ожидает.

Реализация с TCP: Невозможна (TCP не предлагает идентификацию границ сообщения).

Реализация с UDP: Невозможна (UDP не фрагментируетсообщения).

Запрет контрольной суммы при отправке

Протоколы: UDP.

Функционально, поскольку требуются сведения о приложении для принятия решения о приемлемости утраты контроля целостности данных при случайном повреждении.

Реализация: Через SET_CHECKSUM_ENABLED.UDP.

Реализация с TCP: Ничего (TCP не предлагает отключать контрольные суммы, а передача данных с неповрежденной контрольной суммой не может давать семантически неверного результата).

Запрет требования контрольной суммы при получении

Протоколы: UDP.

Функционально, поскольку требуются сведения о приложении для принятия решения о приемлемости утраты контроля целостности данных при случайном повреждении.

Реализация: Через SET_CHECKSUM_REQUIRED.UDP.

Реализация с TCP: Ничего (TCP не предлагает отключать контрольные суммы, а передача данных с неповрежденной контрольной суммой не может давать семантически неверного результата).

Задание покрытия для контрольной суммы у отправителя

Протоколы: UDP-Lite.

Функционально, поскольку требуются сведения о приложении для выбора покрываемых контрольной суммой частей, приемлемого с точки зрения защиты целостности с учетом случайных повреждений.

Реализация: Через SET_CHECKSUM_COVERAGE.UDP-Lite.

Реализация с TCP: Ничего (TCP не предлагает ограничивать область охвата контрольной суммой, а передача данных с неповрежденной контрольной суммой не может давать семантически неверного результата).

Реализация с UDP: Если для контрольной суммы задано покрытие данных, не делает ничего. В иных случаях не делает ничего (передача данных с неповрежденной контрольной суммой не может давать семантически неверного результата) или используется транспортное свойство запрета контрольной суммы при передаче.

Минимальное покрытие для контрольной суммы, требуемое получателем

Протоколы: UDP-Lite.

Функционально, поскольку требуются сведения о приложении для выбора покрываемых контрольной суммой частей, приемлемого с точки зрения защиты целостности с учетом случайных повреждений.

Реализация: Через SET_MIN_CHECKSUM_COVERAGE.UDP-Lite.

Реализация с TCP: Ничего (TCP не предлагает ограничивать область охвата контрольной суммой, а передача данных с неповрежденной контрольной суммой не может давать семантически неверного результата).

Реализация с UDP: Если для контрольной суммы задано покрытие данных, не делает ничего. В иных случаях не делает ничего (передача данных с неповрежденной контрольной суммой не может давать семантически неверного результата) или используется транспортное свойство запрета контрольной суммы при передаче.

Задание поля DF

Протоколы: UDP(-Lite).

Оптимизируемо, поскольку поле DF может служить для передачи Path MTU Discovery и это может приводить к выбору приложением размера, обеспечивающего более эффективную доставку.

Реализация: Через MAINTENANCE.SET_DF.UDP(-Lite) и SEND_FAILURE.UDP(-Lite).

Реализация с TCP: Ничего (в TCP передающее приложение не контролирует размер транспортных сообщений).

Определение максимального размера транспортного сообщения для передачи без фрагментации IP через настроенный интерфейс

Протоколы: UDP(-Lite).

Оптимизируемо, поскольку может приводить к выбору приложением размера, обеспечивающего более эффективную доставку.

Реализация с TCP: Ничего (в TCP эта информация недоступна).

Определение максимального размера транспортного сообщения от настроенного интерфейса

Протоколы: UDP(-Lite).

Оптимизируемо, поскольку может влиять, например, на управление памятью в приложении.

Реализация с TCP: Ничего (в TCP эта информация недоступна).

Задание поля TTL/Hop Count

Протоколы: UDP(-Lite).

Автоматизируемо, поскольку транспортная система может использовать достаточно большое значение, принятое в системе по умолчанию, для предотвращения коммуникационных отказов. Возможность приложения настраивать значение иначе может приводить к получению сообщений ICMP об ошибках, содержащих информацию, связанную с сетью, а не с приложением.

Получение поля TTL/Hop Count

Протоколы: UDP(-Lite).

Автоматизируемо, поскольку поле TTL/Hop Count связано со знанием о сети, а не о приложении.

Задание поля ECN

Протоколы: UDP(-Lite).

Автоматизируемо, поскольку поле ECN связано со знанием о сети, а не о приложении.

Получение поля ECN

Протоколы: UDP(-Lite).

Оптимизируемо, поскольку эта информация может использоваться приложением для более эффективного контроля перегрузок (это относится к выбору транспортного сервиса, который еще не имеет контроля перегрузок).

Реализация с TCP: Ничего (в TCP эта информация недоступна).

Задание опций IP

Протоколы: UDP(-Lite).

Автоматизируемо, поскольку опции IP связаны со знанием о сети, а не о приложении.

Получение опций IP

Протоколы: UDP(-Lite).

Автоматизируемо, поскольку опции IP связаны со знанием о сети, а не о приложении.

Включение и настройка LEDBAT

Протоколы: Протокол, поддерживающий механизм контроля перегрузки LEDBAT.

Оптимизируемо, поскольку решение о включении или отключении этого механизма зависит от сведений о приложении. Однако ошибочное применение этого механизма влияет лишь на скорость доставки данных (и другие передачи, которые могут пользоваться этим транспортным сервисом в сети), поэтому оно остается корректным в рамках модели best effort.

Реализация: Через CONFIGURE.LEDBAT и/или SET_DSCP.TCP, SET_DSCP.SCTP, SET_DSCP.UDP(-Lite) [RFC8622].

Реализация с TCP: Ничего (TCP не поддерживает механизм LEDBAT, но реализация этой функциональности не вызывает семантически некорректного поведения).

Реализация с UDP: Ничего (UDP не предлагает контроля перегрузок).

Завершение

Закрытие после гарантированной доставки оставшихся данных с уведомлением приложения на другой стороне

Протоколы: TCP, SCTP

Функционально, поскольку трактовка соединения часто отражается в приложениях как ожидание доставки остающихся данных и невозможности обмена после выполнения Close в цепочке коммуникаций, относящейся к этому транспортному свойству, которая определяется протоколом приложения.

Реализация: Через CLOSE.TCP и CLOSE.SCTP.

Реализация с UDP: Невозможна (UDP не обеспечивает гарантий доставки и не знает о доставке остающихся данных, а также не указывает причину закрытия соединения партнером).

Разрыв без доставки оставшихся данных с уведомлением приложения на другой стороне

Протоколы: TCP, SCTP.

Функционально, поскольку трактовка соединения часто отражается в приложениях как ожидание доставки остающихся данных и невозможности обмена после выполнения Abort в цепочке коммуникаций, относящейся к этому транспортному свойству, которая определяется протоколом приложения.

Реализация: Через ABORT.TCP and ABORT.SCTP.

Реализация с UDP: Невозможна (UDP не указывает причину закрытия соединения партнером).

Разрыв без доставки оставшихся данных и уведомления приложения на другой стороне

Протоколы: UDP(-Lite).

Функционально, поскольку трактовка соединения часто отражается в приложениях как ожидание доставки остающихся данных и невозможности обмена после выполнения Abort в цепочке коммуникаций, относящейся к этому транспортному свойству, которая определяется протоколом приложения.

Реализация: Через ABORT.UDP(-Lite).

Реализация с TCP: Прекращение использования соединения, ожидание тайм-аута.

Тайм-аут, когда данные не удается доставить слишком долго

Протоколы: TCP, SCTP.

Функционально, так как уведомляет о том, что предполагаемая доставка с гарантией больше не обеспечивается.

Реализация: Через TIMEOUT.TCP и TIMEOUT.SCTP.

Реализация с UDP: Ничего (такое событие невозможно в UDP).

A.2. Связанные с обменом данными транспортные свойства

A.2.1. Передача

Гарантированная доставка данных с контролем перегрузок

Протоколы: TCP, SCTP.

Функционально по причине тесной связи со свойствами данных, которые приложение передает или ожидает.

Реализация: Через SEND.TCP и SEND.SCTP.

Реализация с UDP: Невозможна (UDP не предоставляет гарантий).

Гарантированная доставка сообщения с контролем перегрузок

Протоколы: SCTP.

Функционально по причине тесной связи со свойствами данных, которые приложение передает или ожидает.

Реализация: Через SEND.SCTP.

Реализация с TCP: Через SEND.TCP. Границы сообщения не видны получателю, поскольку TCP предоставляет услуги потока байтов.

Реализация с UDP: Невозможна (UDP не предоставляет гарантий).

Негарантированная доставка сообщения

Протоколы: SCTP, UDP(-Lite).

Оптимизируемо, поскольку лишь приложение знает о временной критичности коммуникаций и гарантированная доставка не будет некорректной для получателя сервиса без гарантий, даже если она медленней.

RFC 8303 отличается от двух приведенных ниже автоматизируемых транспортных свойств в том, что выбор контроля перегрузок остается открытым.

Реализация: Через SEND.SCTP или SEND.UDP(-Lite).

Реализация с TCP: Через SEND.TCP. Сообщения доставляются с гарантией, но получатель не видит границ.

Негарантированная доставка сообщения с контролем перегрузок

Протоколы: SCTP.

Автоматизируемо, поскольку контроль перегрузок связан со знанием о сети, а не о приложении.

Негарантированная доставка сообщения без контроля перегрузок

Протоколы: UDP(-Lite).

Автоматизируемо, поскольку контроль перегрузок связан со знанием о сети, а не о приложении.

Настраиваемые гарантии для сообщений

Протоколы: SCTP.

Оптимизируемо, поскольку лишь приложение знает о временной критичности коммуникаций и гарантированная доставка не будет некорректной для получателя сервиса без гарантий, даже если она медленней.

Реализация: Через SEND.SCTP.

Реализация с TCP: Выполняется через SEND.TCP и игнорирует эту настройку. В предположении модели сервиса best-effort доставка излишних данных не нарушает ожиданий приложения. Кроме того, в TCP невозможно связать запрашиваемые гарантии с сообщением.

Реализация с UDP: Невозможна (UDP не предоставляет гарантий).

Выбор потока

Протоколы: SCTP.

Автоматизируемо, поскольку требует множество потоков, а запрос множества потоков в категории CONNECTION.ESTABLISHMENT является автоматизируемым.

Реализация: См. параграф 5.2.

Выбор пути (адреса получателя)

Протоколы: SCTP.

Реализация с TCP: Н

Упорядоченная доставка сообщений (возможно медленней неупорядоченной)

Протоколы: SCTP.

Функционально по причине тесной связи со свойствами данных, которые приложение передает или ожидает.

Реализация: Через SEND.SCTP.

Реализация с TCP: Через SEND.TCP. Получатель не видит границ сообщения.

Реализация с UDP: Невозможна (UDP не предоставляет гарантий упорядоченности).

Неупорядоченная доставка сообщений (возможно быстрей упорядоченной)

Протоколы: SCTP, UDP(-Lite).

Функционально по причине тесной связи со свойствами данных, которые приложение передает или ожидает.

Реализация: Через SEND.SCTP.

Реализация с TCP: Выполняется через SEND.TCP с сохранением порядка. В предположении модели сервиса best-effort упорядоченная доставка не нарушает ожиданий приложения. Кроме того, в TCP невозможно связать запрос упорядоченной доставки с сообщением.

Запрос отмены группировки сообщений

Протоколы: SCTP.

Оптимизируемо, поскольку решение зависит от сведений о будущих блоках данных и задержке между ними.

Реализация: Через SEND.SCTP.

Реализация с TCP: Выполняется через SEND.TCP и DISABLE_NAGLE.TCP для запрета алгоритма Nagle по запросы и включении снова, когда запроса больше нет. Отметим неполную эквивалентность, связанную со временем подачи запроса, а не с конкретным сообщением.

Реализация с UDP: Ничего (UDP не группирует сообщений).

Задание идентификатора протокола в области данных (для получателя)

Протоколы: SCTP.

Функционально, поскольку позволяет передать дополнительные данные с каждым сообщением для идентификации данных, которая зависит от приложения.

Реализация: SEND.SCTP.

Реализация с TCP: Невозможна (эта функциональность недоступна в TCP).

Реализация с UDP: Невозможна (эта функциональность недоступна в UDP).

Задание идентификатора ключа для аутентификации сообщения

Протоколы: SCTP.

Функционально по причине прямого влияния на безопасность.

Реализация: Через параметр SEND.SCTP.

Реализация с TCP: Возможна эмуляция с использованием SET_AUTH.TCP до и после передачи сообщения. Отметим неполную эквивалентность, связанную со временем подачи запроса, а не с конкретным сообщением.

Реализация с UDP: Невозможна (UDP не поддерживает аутентификации).

Запрос передачи без задержки подтверждения SACK для сообщения

Протоколы: SCTP.

Оптимизируемо, поскольку лишь приложение знает, для каких сообщений ему нужно быстро получить информацию о результате доставки.

Реализация с TCP: Ничего (TCP не предлагает такой функциональности и игнорирует такие запросы приложений, поскольку они задают сементически некорректное поведение).
Реализация с UDP: Ничего (UDP не предлагает такой функциональности и игнорирует такие запросы приложений, поскольку они задают сементически некорректное поведение).

A.2.2. Прием

Прием данных (без границ сообщений)
Протоколы: TCP.
Функционально, поскольку транспортная система должна быть способна передавать и принимать данные.
Реализация: Через RECEIVE.TCP.
Реализация с UDP: Ничего (UDP работает лишь с сообщениями и приложение может игнорировать границы).

Прием сообщения
Протоколы: SCTP, UDP(-Lite).
Функционально по причине тесной связи со свойствами данных, которые приложение передает или ожидает.
Реализация: Через RECEIVE.SCTP и RECEIVE.UDP(-Lite).
Реализация с TCP: Невозможна (TCP не указывает границы сообщений).

Выбор потока для приема
Протоколы: SCTP.
Автоматизируемо, поскольку требует множество потоков, а запрос множества потоков в категории CONNECTION.ESTABLISHMENT является автоматизируемым.
Реализация: См. параграф 5.2.

Информация о частичной доставке сообщения
Протоколы: SCTP.
Функционально по причине тесной связи со свойствами данных, которые приложение передает или ожидает.
Реализация: Через RECEIVE.SCTP.
Реализация с TCP: Ничего (эта информация недоступна в TCP).
Реализация с UDP: Ничего (эта информация недоступна в UDP).

A.2.3. Ошибки

В этом параграфе описаны отказы при передаче, связанные с конкретным вызовом категории «Передача данных» (Приложение A.2.1).

Уведомление об отказе при передаче
Протоколы: SCTP, UDP(-Lite).
Функционально, так как уведомляет о том, что предполагаемая доставка с гарантией больше не обеспечивается.
RFC 8303 отличается от 2 указанных ниже транспортных свойств в том, что не различаются непереданные и неподтвержденные данные.
Реализация: Через SENDFAILURE-EVENT.SCTP и SEND_FAILURE.UDP(-Lite).
Реализация с TCP: Ничего (это уведомление недоступно в TCP).

Уведомление о неотправленном (частино) сообщении
Протоколы: SCTP, UDP(-Lite).
Автоматизируемо, поскольку различие между непереданными и неподтвержденными данными не связано со знанием о приложении.

Уведомление о неподтвержденном (частино) сообщении
Протоколы: SCTP.
Автоматизируемо, поскольку различие между непереданными и неподтвержденными данными не связано со знанием о приложении.

Уведомление об отсутствии в стеке данных для передачи
Протоколы: SCTP.
Оптимизируемо, поскольку реакция на это уведомление требует участия приложения и предотвращение (долгой) работы стека «всухую» может повысить производительность.
Реализация с TCP: Ничего (см. параграф 5.4).
Реализация с UDP: Ничего (это уведомление недоступно в UDP).

Уведомление получателя об отказе при частичной доставке сообщения
Протоколы: SCTP.
Функционально по причине тесной связи со свойствами данных, которые приложение передает или ожидает.
Реализация с TCP: Ничего (это уведомление недоступно в TCP).
Реализация с UDP: Ничего (это уведомление недоступно в UDP).

Благодарности

Авторы благодарят участников рабочей группы TAPS, а также исследовательских проектов NEAT и MAMI за ценный вклад в документ. Особая признательность Michael Tüxen за помощь по вопросам организации и завершения соединений, Gorry Fairhurst за предложения по части фрагментации и размера пакетов, Spencer Dawkins за очень подробную и конструктивную рецензию. 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

N-0316 Oslo

Norway

Phone: +47 22 85 24 20

Email: michawe@ifi.uio.no

Stein Gjessing

University of Oslo

PO Box 1080 Blindern

N-0316 Oslo

Norway

Phone: +47 22 85 24 44

Email: steing@ifi.uio.no

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru