Internet Engineering Task Force (IETF)
Request for Comments: 8304
Category: Informational
ISSN: 2070-1721

G. Fairhurst
T. Jones
University of Aberdeen

February 2018

Transport Features of the User Datagram Protocol (UDP) and Lightweight UDP (UDP-Lite)

Транспортные свойства UDP и UDP-Lite

Аннотация

Этот информационный документ описывает примитивы интерфейса транспортного протокола, предоставляемые протоколами UDP[1] и UDP-Lite[2]. Он указывает службы дейтаграмм, раскрываемые приложениям, способы настройки приложений и использование ими функций, предоставляемых службой транспортировки дейтаграмм в Internet. В RFC 8303 описано применение транспортных функций, обеспечиваемых транспортными протоколами IETF[3], с указанием как UDP, UDP-Lite и другие транспортные протоколы раскрывают свои услуги приложениям и как приложения могут настроить и использовать эти услуги. Этот документ предоставляет исходные данные и контекст для упомянутого документа, а также предлагает план документации, которая может помочь пользователям протоколов UDP и UDP-Lite.

Статус документа

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

Документ является результатом работы IETF и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[4]. 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/rfc8304.

Оглавление

1. Введение

В этом документе представлен некоторые взаимодействия между транспортными протоколами и приложениями в форме примитивов (вызовы функций) для протоколов UDP [RFC0768] и UDP-Lite [RFC3828]. В данном случае приложением считается юбая программа, работающая на основе интерфейса дейтаграмм, включая туннели и другие протоколы вышележащих уровней, применяющий UDP и UDP-Lite.

Протокол UDP имеет множество реализаций и применяется для широкого спектра приложений. Особый класс приложений может получать преимущества при доставке поврежденных данных вместо их отбрасывания, когда нужно работать по путям с каналами, подверженными ошибкам. Приложения, устойчивые к повреждению данных, могут выбрать UDP-Lite вместо UDP и использовать API[5] приложения для работы с контрольными суммами. Приложения UDP также могут выбрать UDP-Lite, но пока это менее распространено и пользователи могут столкнуться с путями, где UDP-Lite не поддерживается. Эти вопросы более подробно рассмотрены в параграфе 3.4 [RFC8085].

[1]User Datagram Protocol - протокол пользовательских дейтаграмм.
[2]Lightweight User Datagram Protocol - облегченный протокол UDP.
[3]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[4]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.
[5]Application programming interface - интерфейс с прикладными программами.

Стандартный IEEE API для приложений TCP/IP использует интерфейс сокетов [POSIX]. Приложение может использовать функции POSIX recv() и send() POSIX, а также recvfrom(), sendto(), recvmsg(), sendmsg(). API сокетов UDP и UDP-Lite отличается от интерфейса для TCP в нескольких важных аспектах (примеры использования API даны в [STEVENS]). В UDP и UDP-Lite каждая дейтаграмма является самодостаточным сообщением заданного размера и на транспортном уровне могут применяться опции, устанавливающие свойства для всех последующих дейтаграмм, передаваемых через этот сокет, или изменяемые для каждой дейтаграммы. Для дейтаграмм от приложения может потребоваться использование API при установке данных уровня IP (IP TTL[1], кодов дифференцированного обслуживания DSCP[2], фрагментирования IP и т. п.) для передаваемых и принимаемых дейтаграмм. При использовании TCP и другого транспорта, ориентированного на соединения) данные уровня IP обычно остаются неизменными в течение срока соединения или контролируются транспортным протоколом, а не приложением.

Опции сокета применяются в API для обеспечения дополнительных функций. Например, опция IP_RECVTTL применяется некоторыми групповыми приложениями UDP для возврата значения поля IP TTL из заголовка принятой дейтаграммы.

Некоторые платформы предлагают приложениям возможность напрямую собирать и передавать пакеты IP через «необрабатываемые» (raw) сокеты или аналогичные средства. API для raw-сокетов является вторым, более громоздким методом передачи дейтаграмм UDP. Использование этого API рассматривается в [RFC8085].

Список свойств и примитивов транспортного сервиса в этом документе строго базируется на тех частях спецификаций протоколов в RFC, которые относятся к тому, что транспортный протокол предоставляет использующему его приложению и как приложение взаимодействует с транспортным протоколом. Примитивы может вызывать приложение или транспортный протокол (в этом случае их называют событиями).

В описании раздела 3 используется терминология, определенная рабочей группой IETF TAPS в [RFC8303]. В частности, документ обеспечивает этап 1 описанного там процесса, который рассматривает RFC, описывающие примитивы каждого протокола. В [RFC8303] эти данные служат входной информацией для описания использования транспортных функций, предоставляемых транспортными протоколами IETF, показывающего как UDP, UDP-Lite и другие транспортные протоколы раскрывают свои услуги приложениям и как приложения могут настроить и применять эти функции для использования транспортных услуг.

Представленный план документации транспортного интерфейса может также помочь разработчикам при работе с UDP и UDP-Lite.

2. Терминология

В этом документе представлены детали для этапа в анализе UDP и UDP-Lite, который используется в [RFC8303]. Документ использует определенную там терминологию , а также уровни требований из RFC 2119 [RFC2119].

3. Примитивы UDP и UDP-Lite

UDP [RFC0768] [RFC8200] и UDP-Lite [RFC3828] - это стандартизованные IETF транспортные протоколы, обеспечивающие односторонние услуги на основе дейтаграмм, поддерживающие операции приема и передачи с сохранением границ сообщений.

В этом разделе приведены относящиеся к теме фрагменты текста RFC, описывающих протоколы UDP и UDP-Lite, с акцентом на предоставление транспортными протоколами своих услуг приложениях и способах использования приложениями этих услуг (на основе описаний абстрактных API, когда они доступны). Описано, как UDP применяется с протоколами IPv4 и IPv6 для отправки индивидуальных и anycast-дейтаграмм, а также для отправки широковещательных дейтаграмм для IPv4. Набор примитивов сетевого уровня для использования UDP или UDP-Lite при групповой адресации IP (IPv4 и IPv6) был задан в серии RFC. Приложение A описывает источники информации о примитивах сетевого уровня, нужных для использования с UDP или UDP-Lite при групповой адресации IP (IPv4 и IPv6).

3.1. Примитивы, предоставляемые UDP

В [RFC0768] сказано:

> Протокол передачи пользовательских дейтаграмм (User Datagram Protocol или UDP) предназначен для поддержки режима обмена дейтаграммами на основе коммутации пакетов в среде связанных между собой компьютерных сетей. … Протокол UDP обеспечивает прикладным программам процедуры для передачи сообщений другим приложениям с минимальным сервисом.

В разделе «Пользовательский интерфейс» RFC 768 сказано, что интерфейс с приложением должен предоставлять:

> возможность создания новых приемных портов, приема данных через приемные порты, которые возвращают октеты данных и индикацию адреса и порта отправителя, операции передачи дейтаграмм с указанием адреса и порта отправителя.

Протокол UDP был определен для IPv6 [RFC8200] вместе с расширениями API "Расширения базового интерфейса сокетов для IPv6" [RFC3493].

В [RFC6935] и [RFC6936] обновлено определение транспорта UDP, заданное исходно в [RFC2460] (заменен RFC 8200). Это позволяет использовать режим с нулевой контрольной суммой UDP для туннельных протоколов при условии, что метод удовлетворяет требованиям в соответствующем заявлении о применимости [RFC6936].

UDP предлагает лишь базовый транспортный интерфейс. Дейтаграммы UDP можно напрямую передавать и принимать без обмена между конечными точками сообщениями для организации соединения (т. е. транспортный протокол не выполняет согласования перед обменом данными). Используя API сокетов, приложения могут принимать пакеты от нескольких адресов IP на один сокет UDP. Общая поддержка позволяет указать один локальный IP-адрес, IP-адрес получателя, локальный и удаленный порт. Можно использовать для любого или всех этих полей принятые по умолчанию значения, предоставляемые локальной системой. Локальный адрес конечной точки устанавливается с

[1]Time To Live - время жизни.
[2]Differentiated Services Code Point.

помощью вызова BIND, адрес удаленной точки - с помощью CONNECT. Функция CLOSE имеет лишь локальную значимость и не влияет на состояние удаленной точки.

Ни UDP, ни UDP-Lite не поддерживают контроля перегрузок, повтора передачи, механизмов пакетизации на прикладном уровне, позволяющих избежать фрагментации IP, и других транспортных функций. Это означает, что использующие UDP приложения должны обеспечивать дополнительные функции на основе транспортного API UDP [RFC8085]. Некоторые транспортные функции требует передачи через API параметров для управления сетевым уровнем (IPv4 или IPv6). Эти дополнительные примитивы можно считать частью сетевого уровня (например, контроль установки флага запрета фрагментирования DF в передаваемых дейтаграммах IPv4), но они необходимы для того, чтобы позволить пользователю UDP API реализовать функции, которые обычно связаны с транспортным уровнем (такие как зондирование максимального размера пакетов на пути). Этот документ включает такие примитивы.

Руководство по использованию услуг UDP представлено в [RFC8085], где также сказано:

> Многие операционные системы позволяют подключить сокет UDP, т. е. связать его с конкретной парой адресов и портов. Это похоже на соответствующую функциональность API сокетов TCP, однако в UDP эта операция локальна и служит лишь для упрощения локальных функций приема и передачи, а также фильтрации трафика для конкретных адресов и портов. Привязка сокета UDP не организует соединения и UDP не уведомляет удаленную сторону о привязке локального сокета UDP. Привязка сокета также помогает в настройке опций, влияющих на уровень UDP или IP, например, использование контрольной суммы UDP или опции IP Timestamp. В некоторых стеках привязка сокета также позволяет приложению получать уведомления при получении сообщения ICMP об ошибках для переданных через сокет пакетов [RFC1122].

Базовая спецификация POSIX [POSIX] определяет API, который предлагает приложениям механизмы получения асинхронных событий, связанных с данными, на уровне сокета. Такие вызовы, как poll, select, queue позволяют приложению получать уведомления о прибытии данных в сокет или очистке сокетом своих буферов.

Управляемый обратными вызовами (callback) API сетевой интерфейс можно структурировать поверх этих вызовов. Неявная организация соединений позволяет приложениям передать управление «жизнью» соединения транспортному API, который использует примитивы протокола, чтобы предлагать приложению примитивы через API сокета. Комбинация примитивов UDP (CONNECT.UDP и SEND.UDP) с API вышележащего уровня может предоставлять похожие услуги.

Ниже перечислены примитивы дейтаграмм.

CONNECT

Примитив CONNECT позволяет связать наборы портов отправителя и получателя с сокетом, чтобы организовать «соединение» для трафика UDP. Такое соединение позволяет приложению получать сведения об ошибках от сетевого стека и обеспечивает сокращенный доступ к примитивам SEND и RECEIVE. Поскольку сам протокол UDP не использует соединений, выполнение этого примитива не вызывает передачи дейтаграммы. Для изменения ассоциации можно снова вызвать connect.

Роли клиента и сервера часто не подходят для UDP, где соединения могут быть peer-to-peer (равный с равным или одноранговые). Функция прослушивания выполняется с использованием одной из форм примитива CONNECT.

1. bind() - операция привязки устанавливает локальный порт неявно, запуская операцию sendto на непривязанном и несоединенном сокете и использованием эфемерного порта, или с явной привязкой для использования настроенного или общепринятого порта.
2. bind(); connect() - операция привязки с последующим примитивом CONNECT. Привязка организует использование известного локального порта для дейтаграмм вместо эфемерного порта. Операция connect задает известную комбинацию адрес-порт для использования по умолчанию в будущих дейтаграммах. Эта форма применяется после приема дейтаграммы от конечной точки, которая вызвала создание соединения или может быть инициирована сторонней конфигурацией или протокольным триггером (например, прием записи протокола описания сервиса UDP SDP [RFC4566]).

SEND

Примитив SEND передает предоставленное число байтов, которые UDP следует отправить на другую сторону соединения UDP в дейтаграмме UDP. Примитив может применяться приложением для отправки дейтаграмм напрямую конечной точке, указанной адресом и портом. Если соединение создано, пара адрес-порт выводится из текущего соединения для сокета. Подключение сокета позволяет возвращать приложению сетевые ошибки в качестве уведомлений от примитива SEND. Переданные SEND сообщения, которые нельзя передать неделимо в пакете IP, не будут передаваться на сетевой уровень и вызовут ошибку.

RECEIVE

Примитив RECEIVE выделяет приемный буфер для размещения полученной дейтаграммы. Примитив возвращает число байтов из полученной дейтаграммы UDP. В параграфе 4.1.3.5 [RFC1122] сказано: «При получении дейтаграммы UDP указанный адрес получателя **должен** передаваться на уровень приложений».

CHECKSUM_ENABLED

Необязательный примитив CHECKSUM_ENABLED определяет, включает ли отправитель контрольную сумму UDP при передаче дейтаграмм [RFC0768] [RFC6935] [RFC6936] [RFC8085]. Если примитив сброшен (unset), это отменяет принятое по умолчанию поведение UDP, отключая контрольную сумму при отправке. В параграфе 4.1.3.4 [RFC1122] сказано: «Приложения **могут** управлять процессом генерации контрольных сумм UDP, но по умолчанию протокол **должен** включать контрольную сумму».

REQUIRE_CHECKSUM

Необязательный примитив REQUIRE_CHECKSUM определяет поведение при получении дейтаграмм с нулевой контрольной суммой UDP. По умолчанию UDP требует наличия контрольной суммы. В параграфе 4.1.3.4 [RFC1122] сказано: «Приложения **могут** управлять решением вопроса об отбрасывании дейтаграмм без контрольной суммы». Параграф 3.1 спецификации UDP-Lite [RFC3828] требует отличной от 0 контрольной суммы, поэтому интерфейс UDP-Lite API должен отбрасывать дейтаграммы с нулевой контрольной суммой.

SET_IP_OPTIONS

Примитив SET_IP_OPTIONS запрашивает у сетевого уровня передачу дейтаграммы с заданными опциями IP. В параграфе 4.1.3.2 [RFC1122] сказано: «Приложениям **должна** предоставляться возможность установки опций IP для передаваемых дейтаграмм UDP и протокол UDP **должен** передавать эти опции уровню IP».

GET_IP_OPTIONS

Примитив GET_IP_OPTIONS отыскивает опции IP для принятой сетевым уроынем дейтаграммы. В параграфе 4.1.3.2 [RFC1122] указано, что на стороне получателя UDP: «Протокол UDP **должен** передавать прикладным программам все опции IP, полученные от уровня IP».

SET_DF

Примитив SET_DF позволяет сетевому уровню фрагментировать пакеты с использованием Fragment Offset в IPv4 [RFC6864], а хосту - использовать Fragment Headers в IPv6 [RFC8200]. SET_DF устанавливает флаг DF в заголовке пакета IPv4 с дейтаграммой UDP, управляющий возможностью фрагментирования пакетов IPv4 маршрутизаторами. Хотя некоторые приложения опираются на поддержку фрагментирования, в общем случае приложениям UDP следует реализовать метод предотвращения фрагментации IP (раздел 4 в [RFC8085]). Отметим, что во многих других транспортных протоколах IETF (например, TCP и SCTP) транспорт обеспечивает поддержку, требуемую для применения DF. Однако при использовании UDP приложение отвечает за методы, требуемые для определения эффективного Path MTU (PMTU) на пути через сеть в координации с сетевым уровнем. Классический механизм Path MTU Discovery (PMTUD) [RFC1191] основан на возврате с пути сообщений ICMP Fragmentation Needed или ICMPv6 Packet Too Big. Когда такие сообщения ICMP не доставляются (или фильтруются), отправитель не может узнать реальное значение PMTU и дейтаграммы UDP, размер которых превышает PMTU, уйдут в «черную дыру». Для предотвращения этого приложение может реализовать механизм PLPMTUD (Packetization Layer Path MTU Discovery) [RFC4821], который не опирается на поддержку в сети сообщений ICMPv6 и поэтому считается более отказоустойчивым, нежели стандартный механизм PMTUD, как отмечено в [RFC8085] и [RFC8201].

GET_MMS_S

Примитив GET_MMS_S извлекает значение сетевого уровня, указывающее максимальный размер сообщения (MMS), которое можно передать на транспортном уровне в нефрагментируемом пакете IP через настроенный интерфейс. Это значение задано в параграфе 6.1 [RFC1191] и параграфе 5.1 [RFC8201]. Значение рассчитывается по эффективному MTU для передачи (Effective MTU for Sending или EMTU_S) и MTU на канале для данного IP-адреса отправителя. Учитывается размер заголовка IP и пространство, резервируемое уровнем IP для дополнительных заголовков (при наличии). Приложениям UDP следует использовать это значение как часть метода предотвращения отправки дейтаграмм UDP, порождающих пакеты IP, размер которых превышает эффективное значение PMTU на пути через сеть. Эффективное значение PMTU (см. раздел 1 в [RFC1191]) эквивалентно EMTU_S (задано в [RFC1122]). В спецификации PLPMTUD [RFC4821] сказано:

> Если PLPMTUD обновляет MTU для определенного пути, все сессии уровня пакетизации, использующие этот путь (см. параграф 5.2), **следует** уведомить об использовании нового MTU и выполнении требуемой корректировки контроля перегрузок.

GET_MMS_R

Примитив GET_MMS_R извлекает значение сетевого уровня, указывающее MMS для пакетов, принимаемых транспортным уровнем через настроенный интерфейс. Значение задано в параграфе 3.1 [RFC1191] и рассчитывается из эффективного MTU для приема (Effective MTU for Receiving или EMTU_R) и MTU канала для данного IP-адреса отправителя с учетом размера заголовка IP и пространства, резервируемого уровнем IP для дополнительных заголовков (при наличии).

SET_TTL

Примитив SET_TTL устанавливает значение Hop Limit (поле TTL) на сетевом уровне, которое используется в заголовке IPv4 пакета с дейтаграммой UDP. Поле служит для ограничения области действия индивидуальных дейтаграмм. В параграфе 3.2.2.4 [RFC1122] сказано: «Принимаемые сообщения Time Exceeded **должны** передаваться на транспортный уровень».

GET_TTL

Примитив GET_TTL извлекает значение поля TTL в пакете IP, принятом сетевым уровнем. Приложения с поддержкой обобщенного механизма защиты GTSM [RFC5082] могут использовать эту информацию, чтобы доверять дейтаграммам со значением TTL в ожидаемом диапазоне, как указано в разделе 3 RFC 5082.

SET_MIN_TTL

Примитив SET_MIN_TTL ограничивает доставку дейтаграмм приложению на основании поля IP TTL, значение которого должно быть не меньше указанного параметром. Примитив можно использовать для реализации таких приложений, как GTSM (раздел 3 в RFC 5082) [RFC5082], но RFC не задает этого метода.

SET_IPV6_UNICAST_HOPS

Примитив SET_IPV6_UNICAST_HOPS задает поле сетевого уровня Hop Limit в заголовке пакета IPv6 [RFC8200] с дейтаграммой UDP. Для индивидуальных дейтаграмм IPv6 это функционально эквивалентно SET_TTL в IPv4.

GET_IPV6_UNICAST_HOPS

Примитив GET_IPV6_UNICAST_HOPS является функцией сетевого уровня, считывающей значение счетчика интервалов в заголовкеIPv6 header [RFC8200] полученной дейтаграммы UDP. Это задано в параграфе 6.3 RFC 3542. Для индивидуальных дейтаграмм IPv6 это функционально эквивалентно GET_TTL в IPv4.

SET_DSCP

Примитив SET_DSCP является функцией сетевого уровня, устанавливающей значение DSCP (или Type of Service - ToS) [RFC2474] в поле заголовка IP пакета с дейтаграммой UDP. В параграфе 2.4 [RFC1123] сказано: «Приложение **должно** выбирать приемлемые значения TOS при обращении к службам транспортного уровня и эти значения **должны** быть настраиваемыми». Приложению следует поддерживать возможность изменения ToS в процессе работы соединения и значение ToS следует передавать на уровень IP без изменения. В параграфе 4.1.4 [RFC1122] сказано: «UDP **может** передавать полученные значения TOS на уровень приложений». Модель Diffserv [RFC2475] [RFC3260] заменила это поле заголовка IP, выделив 6 старших битов для поля DSCP [RFC2474]. Сохранение требования [RFC1122] разрешать приложению задавать тип обслуживания следует понимать как сохранение возможности API разрешать приложению установку DSCP. В параграфе 3.1.8 [RFC8085] описаны способы использования этого поля приложениями UDP. Обычно сокет UDP будет назначать одно значение DSCP для всех дейтаграмм потока, но отправителю разрешено в некоторых случаях применять разные значения DSCP для дейтаграмм одного потока [RFC8085]. Рекомендации для WebRTC иллюстрируют такой случай [RFC7657].

SET_ECN

Примитив SET_ECN является функцией сетевого уровня, устанавливающей поле явного контроля перегрузок (Explicit Congestion Notification или ECN) в заголовке IP дейтаграммы UDP. Поле ECN по умолчанию имеет значение 00. Когда использование поля ToS было переопределено в Diffserv [RFC3260], 2 бита поля были

выделены для ECN [RFC3168]. В параграфе 3.1.7[1] [RFC8085] описано, как приложению UDP следует использовать это поле. Отметим, что во многих других транспортных протоколах IETF (например, TCP) транспорт обеспечивает поддержку ECN. При использовании UDP за методы, требуемые для применения ECN отвечает приложение или протокол вышележащего уровня.

GET_ECN

Примитив GET_ECN является функцией сетевого уровня, возвращающей значение поля ECN в заголовке IP принятой дейтаграммы UDP. В параграфе 3.1.5 [RFC8085] сказано: «UDP, **должен** проверять поле ECN для каждой принятой на этом порту дейтаграммы UDP». Это требует от UDP API на стороне получателя передавать полученное поле ECN прикладному уровню для подобающей реакции на перегрузку.

ERROR_REPORT

Событие ERROR_REPORT информирует приложение о некритичной ошибке (soft errors), включая прием сообщения ICMP или ICMPv6 об ошибке. В параграфе 4.1.3.3[2] [RFC1122] сказано: «Протокол UDP **должен** передавать на уровень приложений все сообщения ICMP об ошибках, полученные от уровня IP». Это событие требуется, например, для реализации на основе ICMP механизма Path MTU Discovery [RFC1191] [RFC8201]. Приложения UDP должны выполнять примитив CONNECT для получения сообщений ICMP об ошибках.

CLOSE

Примитив CLOSE закрывает соединение. После этого дейтаграммы не передаются и не принимаются. Поскольку сам протокол UDP не использует соединения, после выполнения этого примитива дейтаграммы не передаются.

3.1.1. Исключенные примитивы

В параграфе 3.4 [RFC1122] описаны примитивы GET_MAXSIZES и ADVISE_DELIVPROB, а в параграфе 3.3.4.4 - GET_SRCADDR. Эти механизмы больше не применяются. Документ также задает использование сообщений ICMP Source Quench, которые были отменены [RFC6633].

Функция IPV6_V6ONLY является примитивом сетевого уровня, который применяется ко всем транспортным службам, как указано в параграфе 5.3 для базового интерфейса сокетов IPv6 [RFC3493]. Это ограничивает использование информации от распознавателя имен, разрешая лишь коммуникации сокетов AF_INET6 с использованием IPv6 и не считается частью транспортного сервиса.

3.2. Примитивы, предоставляемые UDP-Lite

UDP-Lite [RFC3828] предоставляет услуги, похожие на UDP. В протоколе изменена семантика поля размера данных UDP (payload length) на семантику поля покрытия контрольной суммой. UDP-Lite требует расчета контрольной суммы псевдозаголовка отправителем и ее проверки получателем. В остальном UDP-Lite семантически эквивалентен UDP.

Передающий интерфейс UDP-Lite отличается от UDP добавление одной опции (сокета), указывающей область покрытия контрольной суммы, которая задает учитываемую в расчете часть данных, а остальные называют нечувствительными к ошибкам (error-insensitive part).

Приемный интерфейс UDP-Lite отличается от UDP добавлением одной опции (сокета), задающей минимальное покрытие контрольной суммой. База UDP-Lite MIB (Management Information Base) [RFC5097] дополняет метод покрытия контрольной суммой. Рекомендации по использованию услуг UDP-Lite приведены в [RFC8085].

UDP-Lite требует наличия контрольной суммы UDP или UDP-Lite, поэтому не разрешает использовать функцию DISABLE_CHECKSUM для запрета расчета контрольной суммы и не позволяет отключить проверку контрольной суммы получателем с помощью REQUIRE_CHECKSUM. Остальные примитивы и функции UDP разрешены.

Кроме того, добавлено 2 примитива, описанных ниже.

SET_CHECKSUM_COVERAGE

Примитив SET_CHECKSUM_COVERAGE задает облать дейтаграммы, учитываемую в контрольной сумме. Трафик UDP-Lite использует этот примитив для установки размера области покрытия контрольной суммой UDP. В параграфе 3.3 спецификации UDP-Lite [RFC3828] сказано: «Приложениям, желающим определить часть своих данных, как нечувствительные к битовым ошибкам ..., следует делать это путем явного системного вызова на стороне отправителя». По умолчанию применяется такое же покрытие как в UDP.

SET_MIN_COVERAGE

Примитив SET_MIN_COVERAGE задает минимальную приемлемую область покрытия контрольной суммой в принимаемой дейтаграмме. Трафик UDP-Lite использует этот примитив для установки области покрытия, проверяемой на приеме (в параграфе 1.1 [RFC5097] указана запись MIB udpliteEndpointMinCoverage). В параграфе 3.3 спецификации UDP-Lite [RFC3828] сказано: «Приложениям, желающим получать данные с неполным покрытием для контрольной суммы, следует информировать принимающую систему с помощью явного системного вызов». По умолчанию для принимаемых дейтаграмм требуется лишь минимальное покрытие.

4. Взаимодействие с IANA

Этот документ не требует действий со стороны IANA.

5. Вопросы безопасности

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

6. Литература

6.1. Нормативные документы

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[1] В оригинале ошибочно указан параграф 3.1.5. См. https://www.rfc-editor.org/errata/eid6370. *Прим. перев.*

[2] В оригинале ошибочно указан параграф 4.1.4. https://www.rfc-editor.org/errata/eid6371. *Прим. перев.*

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[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>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[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>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <https://www.rfc-editor.org/info/rfc3493>.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed., and G. Fairhurst, Ed., "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, DOI 10.17487/RFC3828, July 2004, <https://www.rfc-editor.org/info/rfc3828>.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, DOI 10.17487/RFC6864, February 2013, <https://www.rfc-editor.org/info/rfc6864>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <https://www.rfc-editor.org/info/rfc6935>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <https://www.rfc-editor.org/info/rfc6936>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

[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>.

6.2. Дополнительная литература

[POSIX] IEEE, "Standard for Information Technology — Portable Operating System Interface (POSIX(R)) Base Specifications", Issue 7, IEEE 1003.1, <http://ieeexplore.ieee.org/document/7582338/>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

[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>.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <https://www.rfc-editor.org/info/rfc3376>.

[RFC3678] Thaler, D., Fenner, B., and B. Quinn, "Socket Interface Extensions for Multicast Source Filters", RFC 3678, DOI 10.17487/RFC3678, January 2004, <https://www.rfc-editor.org/info/rfc3678>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <https://www.rfc-editor.org/info/rfc4604>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <https://www.rfc-editor.org/info/rfc5082>.

[RFC5097] Renker, G. and G. Fairhurst, "MIB for the UDP-Lite protocol", RFC 5097, DOI 10.17487/RFC5097, January 2008, <https://www.rfc-editor.org/info/rfc5097>.

[RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, DOI 10.17487/RFC5790, February 2010, <https://www.rfc-editor.org/info/rfc5790>.

[RFC6633] Gont, F., "Deprecation of ICMP Source Quench Messages", RFC 6633, DOI 10.17487/RFC6633, May 2012, <https://www.rfc-editor.org/info/rfc6633>.

[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>.

[STEVENS] Stevens, W., Fenner, B., and A. Rudoff, "UNIX Network Programming, The sockets Networking API", Volume 1, ISBN-13: 9780131411555, October 2003.

Приложение A. Примитивы групповой адресации

В этом приложении описаны примитивы, служащие для поддержки в UDP и UDP-Lite групповой адресации IPv4 и IPv6. Групповые услуги не рассматриваются рабочей группой IETF TAPS, но имеющиеся примитивы для полноты указаны в этом приложении. Рекомендации по использованию групповых услуг UDP и UDP-Lite даны в [RFC8085].

Групповая адресация IP может поддерживаться с помощью модели ASM[1] или SSM[2]. Во втором случае требуется применять фильтр отправителей (Multicast Source Filter или MSF) при указании группового IP-адреса получателей.

Для использования групповой адресации нужны дополнительные примитивы в транспортном API, которые вызываются для координации действий протоколов сетевого уровня IPv4 и IPv6. Например, для получения отправленных в группу дейтаграмм конечная точка должна сначала стать членом группы на сетевом уровне. Локальный прием групповых пакетов IPv4 использует сигнализацию протокола Internet Group Management Protocol (IGMP) [RFC3376] [RFC4604]. В IPv6 используется эквивалентный протокол Multicast Listener Discovery (MLD) [RFC3810] [RFC5790], работающий на основе ICMPv6. Облегченные версии этих протоколов заданы в [RFC5790].

Ниже перечислены примитивы, применяемые для групповой адресации.

JoinHostGroup
: В параграфе 7.1 [RFC1112] представлены функции, позволяющие принимать трафик из multicast-групп IP.

JoinLocalGroup
: В параграфе 7.3 [RFC1112] представлены функции, позволяющие принимать трафик из локальных групп IP.

LeaveHostGroup
: В параграфе 7.1 [RFC1112] представлены функции, позволяющие выходить из multicast-групп IP.

LeaveLocalGroup
: В параграфе 7.3 представлены функции, позволяющие выходить из локальных multicast-групп IP.

IPV6_MULTICAST_IF
: В параграфе 5.2 IPv6 [RFC3493] сказано, что этот примитив задает интерфейс, используемый для исходящих групповых пакетов.

IP_MULTICAST_TTL
: Устанавливает поле времени жизни t в исходящих групповых пакетах IPv4 для ограничения области распространения дейтаграмм. Такие методы, как GTSM [RFC5082], устанавливают это значение для обеспечения передачи в локальный канал. GTSM также требует у получателя UDP API для передачи этого значения приложению.

IPV6_MULTICAST_HOPS
: В параграфе 5.2 [RFC3493] сказано, что этот примитив задает число интервалов (hop count) для исходящих групповых пакетов IPv6 (эквивалент IP_MULTICAST_TTL в IPv4).

IPV6_MULTICAST_LOOP
: В параграфе 5.2 [RFC3493] сказано, что этот примитив задает, нужно ли возвращать копию дейтаграммы на уровень IP для локальной доставки при передаче дейтаграммы в группу, к которой относится передающий хост.

IPV6_JOIN_GROUP
: В параграфе 5.2 [RFC3493] представлена функция для включения конечной точки в multicast-группу IPv6.

SIOCGIPMSFILTER
: В параграфе 8.1 [RFC3678] представлена функция, позволяющая читать фильтры multicast-отправителей.

SIOCSIPMSFILTER
: В параграфе 8.1 [RFC3678] представлена функция для установки и изменения фильтров multicast-отправителей.

IPV6_LEAVE_GROUP
: В параграфе 5.2 [RFC3493] представлена функция для выхода конечной точки из multicast-группы IPv6.

В [RFC3678] обновлен групповой интерфейс с добавлением поддержки MSF для IPv4 и IPv6, требуемой IGMPv3. В разделе 3 определены базовый и расширенный API, а раздел 5 описывает независимые от протокола версии этих API. Определены 4 набора функциональности API:

1. Базовый IPv4 API. Каждый вызов функции задает один адрес отправителя, который следует добавить или удалить для фильтра данной прослушиваемой группы адресов.
2. Расширенный IPv4 API. Позволяет приложению задать полный фильтр отправителей на замену имеющемуся.
3. Независимый от протокола базовый MSF API.
4. Независимый от протокола расширенный MSF API.

Определенные вновь примитивы перечислены ниже.

IP_ADD_MEMBERSHIP
: Служит для присоединения к группе ASM.

IP_BLOCK_SOURCE
: Этот фильтр MSF может блокировать данные от указанного отправителя в данную группу ASM или SSM.

[1] Any Source Multicast - групповая адресация для любого источника.
[2] Source-Specific Multicast - зависимая от источника групповая адресация.

IP_UNBLOCK_SOURCE
Обновляет MSF для отмены предыдущего вызова IP_UNBLOCK_SOURCE для группы ASM или SSM.
IP_DROP_MEMBERSHIP
Служит для выхода из группы ASM или SSM (в SSM ведет к отбрасыванию всех отправителей, присоединенных к конкретной группе и интерфейсу, что эквивалентно закрытию сокета).

В параграфе 4.1.2 MSF [RFC3678] обновлен интерфейс путем добавления поддержки IPv4 MSF в IGMPv3 с ASM.

IP_ADD_SOURCE_MEMBERSHIP
Служит для присоединения к группе SSM.
IP_DROP_SOURCE_MEMBERSHIP
Служит для выхода из группы SSM.

Параграф 4.2 [RFC3678] определяет расширенный API, примитивы которого указаны ниже.

setipv4sourcefilter
Служит для присоединения к группе IPv4 или разрешения multicast-трафика из указанного источника.
getipv4sourcefilter
Служит для выхода из группы IPv4 или фильтрации multicast-трафика из указанного источника.

В параграфе 5.1 [RFC3678] заданы функции независимого от протокола базового API, перечисленные ниже.

MCAST_JOIN_GROUP
Служит для присоединения к группе ASM.
MCAST_JOIN_SOURCE_GROUP
Служит для присоединения к группе SSM.
MCAST_BLOCK_SOURCE
Служит для блокировки отправителя в группе ASM.
MCAST_UNBLOCK_SOURCE
Удаляет прежний фильтр MSF, установленный MCAST_BLOCK_SOURCE.
MCAST_LEAVE_GROUP
Служит для выхода из группы ASM или SSM.
MCAST_LEAVE_SOURCE_GROUP
Служит для выхода из группы SSM.

В параграфе 5.2 [RFC3678] заданы функции независимого от протокола расширенного API для IPv4 и IPv6.

setsourcefilter
Служит для присоединения к группе IPv4 или IPv6 и разрешения multicast-трафика из указанного источника.
getsourcefilter
Служит для выхода из группы IPv4 или IPv6 и фильтрации multicast-трафика из указанного источника.

Протоколы Lightweight IGMPv3 (LW_IGMPv3) и MLDv2 [RFC5790] обновляют этот интерфейс (параграф 7.2 в RFC 5790).

Благодарности

Эта работа частично финансировалась исследовательской и инновационной программой Европейского Союза Horizon 2020 в рамках гранта № 644334 (NEAT). Спасибо всем, кто внес свои комментарии или вклад в работу, включая Joe Touch, Ted Hardie, Aaron Falk, Tommy Pauly и Francis Dupont.

Адреса авторов

Godred Fairhurst

University of Aberdeen

School of Engineering

Fraser Noble Building

Fraser Noble Building Aberdeen AB24 3UE

United Kingdom

Email: gorry@erg.abdn.ac.uk

Tom Jones

University of Aberdeen

School of Engineering

Fraser Noble Building

Aberdeen AB24 3UE

United Kingdom

Email: tom@erg.abdn.ac.uk

Перевод на русский язык
Николай Малых

nmalykh@protokols.ru