Internet Engineering Task Force (IETF)
Request for Comments: 8684
Obsoletes: 6824
Category: Standards Track
ISSN: 2070-1721

A. Ford
Pexip
C. Raiciu
U. Politehnica of Bucharest
M. Handley
U. College London
O. Bonaventure
U. catholique de Louvain
C. Paasch
Apple, Inc.

March 2020

TCP Extensions for Multipath Operation with Multiple Addresses

Расширения TCP для работы по нескольким путям с множеством адресов

Аннотация

Коммуникации TCP/IP в настоящее время ограничены одним путем на соединение даже при наличии между партнерами множества путей. Одновременное использование таких путей для сессии TCP/IP будет повышать эффективность использования ресурсов в сети, а также улучшать взаимодействие с пользователем за счет роста пропускной способности и повышения устойчивости к отказам в сети.

Multipath TCP обеспечивает возможность одновременного использования множества путей между партнерами. В этом документе представлен набор расширений для традиционного протокола TCP с целью поддержки работы по нескольким путям. Протокол предоставляет приложениям такой же набор услуг как TCP (надежная доставка потока байтов) и обеспечивает компоненты, требуемые для организации использования множества потоков TCP через потенциально не связанные пути.

Документ задает протокол Multipath TCP v1, отменяя предыдущую версию (v0), определенную в RFC 6824, разъясняя и меняя некоторые свойства на основе опыта развертывания.

Статус документа

Документ относится к категории Internet Standards Track.

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/rfc8684.

Оглавление

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

1. Введение

Multipath TCP (MPTCP) представляет собой набор расширения протокола TCP [RFC0793], обеспечивающих услуги Multipath TCP [RFC6182], когда транспортное соединение может использовать одновременно множество путей. В документе представлены изменения, которые нужны для поддержки множества путей в TCP, в частности, для сигнализации и организации множества путей (субпотоков), управления этими субпотоками, сборки данных и завершения сессий. Однако документ включает не только информацию, требуемую для реализации Multipath TCP, но и дополняет три других документа, указанных ниже.

- [RFC6182] (архитектура MPTCP), разъясняющий мотивы разработки Multipath TCP и обсуждающий устройство протокола на высоком уровне, а также разъясняющий функциональное деление, которое может быть реализовано в расширяемом MPTCP.
- [RFC6356] (контроль перегрузок), представляющий механизмы безопасного контроля перегрузок для связанного набора путей, не препятствующие работе других пользователей сети.
- [RFC6897] (взаимодействие с приложением), где обсуждается влияние MPTCP на приложения, ожидания приложений от MPTCP и последствия этих факторов, а также расширения API для MPTCP.

Этот документ отменяет спецификацию Multipath TCP v0 [RFC6824] и задает спецификацию MPTCP v1, не совместимую с MPTCP v0. Кроме того, документ определяет процедуру согласования версии для реализаций, поддерживающих оба варианта.

1.1. Допущения

Для ограничения пространства проектирования рабочая группа MPTCP внесла два важных ограничения в протокол Multipath TCP, представленный в этом документе:

- требуется совместимость с обычным TCP для упрощения развертывания в сети;
- можно предполагать, что один или оба хоста являются многодомными и многоадресными.

Для упрощения предполагается, что наличия на хосте множества адресов достаточно для индикации существования множества путей. Эти пути не обязаны быть полностью разными и допускается совместное использование одного или нескольких маршрутизаторов разными путями. Даже в такой ситуации использование множества путей обеспечивает преимущества, улучшая использование ресурсов и повышение устойчивости к некоторым отказам узлов. Алгоритмы контроля перегрузок, определенные в [RFC6356], обеспечивают отсутствие вредных воздействий при использовании множества путей. Кроме того, могут возникать ситуации, когда несколько портов TCP на одном хосте могут предоставлять разные пути (например, при использовании некоторых реализаций ECMP[1] [RFC2992]), поэтому MPTCP поддерживает идентификацию путей по портам.

Для обеспечения упомянутой выше совместимости важны три аспекта, подробно рассмотренных в [RFC6182].

Внешние ограничения
Протокол должен работать через большинство имеющихся промежуточных устройств, таких как NAT, межсетевые экраны (МСЭ) и прокси, напоминая им обычный протокол TCP, насколько это возможно в линии. Кроме того, протокол не должен предполагать неизменность отправленных в сеть сегментов, поскольку они могут быть разделены или объединены, а опции TCP могут удаляться или дублироваться.

Ограничения приложение
Протокол должен обеспечивать возможность работы имеющихся приложений на основе TCP API без из изменения (хотя часть функций может быть недоступна для устаревших приложений). Кроме того, протокол должен предоставлять приложениям такую же модель обслуживания, как обычный протокол TCP.

Возврат к TCP
Протоколу следует поддерживать возможность возврата к стандартному TCP без влияния на пользователя, чтобы обеспечить взаимодействие с устаревшими хостами.

Дополнительное рассмотрение взаимодействия с приложениями в [RFC6897] указывает требуемые функции API для обеспечения совместимости со старыми версиями, а также расширения API, обеспечивающие работу MPTCP на уровнях управления и данных, эквивалентную обычному TCP с одним путем.

Дополнительное рассмотрение ограничений и связанных с ними решений в части архитектуры MPTCP приведено в [RFC6182] и [howhard].

1.2. Multipath TCP в сетевом стеке

MPTCP работает на транспортном уровне и стремиться быть прозрачным для вышележащего и нижележащего уровня. Протокол добавляет ряд свойств к стандартному TCP. Уровни протоколов показаны на рисунке 1. MPTCP разработан для использования унаследованными приложениями без их изменения. Этот вопрос подробно рассматривается в [RFC6897].

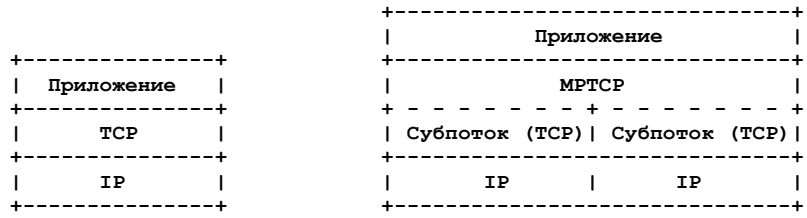


Рисунок 1. Сравнение стеков TCP и MPTCP.

1.3. Терминология

В этом документе используется множество терминов, относящихся к MPTCP или определенных в контексте MPTCP.

Path - путь
Последовательность каналов между отправителем и получателем, определяемая в контексте кортежем с адресом и номером порта отправителя и получателя.

Subflow - субпоток
Поток сегментов TCP, передаваемых по отдельному пути, являющийся частью соединения Multipath TCP. Субпоток начинается и завершается подобно обычному соединению TCP.

(Multipath TCP) Connection - соединение (Multipath TCP)
Множество из одного или нескольких субпотоков, через которые могут взаимодействовать приложения на двух хостах. Между соединением и сокетом приложения имеется взаимно-однозначное соответствие.

Data-level - уровень данных
Данные (payload), номинально передаваемые через соединение (в субпотоках). Термины «уровень данных» и «уровень соединения» относятся к соединению в целом, а «уровень субпотока» - к отдельному субпотоку.

Token - маркер
Логически уникальный идентификатор, предоставленный соединению хостом. Может называться также идентификатором соединения (Connection ID).

Host - хост
Конечный хост с реализацией MPTCP, который инициирует или воспринимает соединение MPTCP.

В дополнение к этим терминам отметим обычную семантику TCP, обсуждаемую в разделе 4.

1.4. Концепции MPTCP

В этом параграфе представлен высокоуровневый обзор работы MPTCP. Типичный вариант применения протокола показан на рисунке 2. Подробное рассмотрение работы MPTCP приведено в разделе 3.

- Для приложений, не понимающих MPTCP протокол будет вести себя как обычный TCP. Расширенные API могут обеспечивать дополнительный контроль для приложений MPTCP [RFC6897]. Приложение начинает работу с обычного создания сокета TCP, а сигнализация и работа MPTCP обеспечиваются реализацией.

[1]Equal-Cost Multipath - множество равноценных путей.

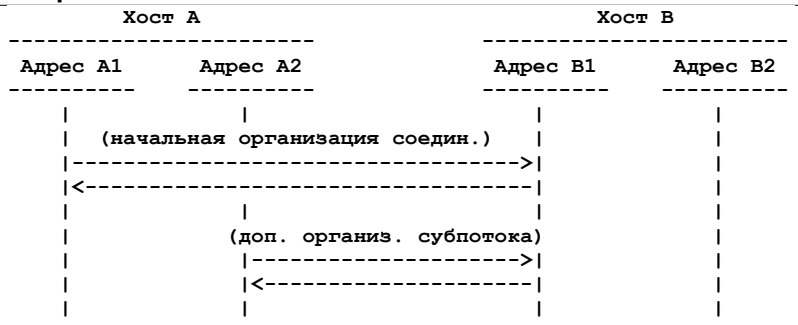


Рисунок 2. Пример использования MPTCP.

- Соединение MPTCP начинается подобно обычному соединению TCP. Это показано на рисунке 2, где соединение MPTCP организуется между адресами A1 и B1 на хостах A и B.
- При наличии дополнительных путей открываются новые сессии TCP (субпотоки MPTCP) и объединяются с имеющейся сессией для организации одного соединения между приложениями на двух сторонах. Создание дополнительной сессии между адресами A2 хоста Host A и B1 хоста B показано на рисунке 2.
- MPTCP идентифицирует наличие множества путей по присутствию на хосте множества адресов. Комбинации этих адресов приравниваются к дополнительным путям. В приведенном выше примере дополнительными путями могут быть A1<->B2 и A2<->B2. Хотя эти сессии показаны как инициированные с адреса A2, их можно инициировать также с адреса B1 или B2.
- Обнаружение и организация дополнительных субпотоков выполняются с помощью методов управления путями. В этом документе описан механизм, с помощью которого хост может инициировать новый субпоток с использованием своего дополнительного адреса или указать другому хосту доступные адреса.
- MPTCP добавляет порядковые номера на уровне соединения для обеспечения сборки сегментов, приходящих по множеству путей с разной задержкой в сети.
- Субпотоки завершаются как обычные соединения TCP с помощью 4-этапного согласования FIN. Соединение MPTCP прерывается FIN на уровне соединения.

1.5. Уровни требований

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

2. Обзор работы протокола

В этом разделе представлено описание работы MPTCP со ссылками на протокольные операции. Это высокоуровневый обзор основных функций, а полная спецификация приведена в разделе 3. Расширяемый и согласуемый набор функций не рассматривается здесь. В этом разделе часто упоминаются символьные имена опций MPTCP, которые относятся к субтипам выделенной IANA опции MPTCP (раздел 7), а их форматы определены в спецификации протокола (раздел 3).

Соединение Multipath TCP поддерживает двухсторонний поток байтов между двумя хостами, взаимодействующими как при обычном TCP, и в приложения не требуется вносить какие-либо изменения. Однако Multipath TCP позволяет хостам использовать разные адреса IP для обмена пакетами, относящимися к одному соединению MPTCP. Соединение Multipath TCP выглядит для приложения подобно обычному соединению TCP, однако для сетевого уровня каждый субпоток MPTCP похож на обычный поток TCP, сегменты которого имеют новый тип опций TCP. Multipath TCP управляет созданием, удалением и использованием этих субпотоков для передачи данных. Число субпотоков, поддерживаемых в соединении Multipath TCP, не фиксировано и может меняться в процессе работы соединения.

Все операции MPTCP используют сигнализацию через опцию TCP - один численный тип для MPTCP с субтипами для каждого сообщения MPTCP. Ниже кратко описано назначение и приведено обоснование для каждого из сообщений.

2.1. Инициирование соединения MPTCP

Здесь применяется такая же сигнализация как в соединениях TCP но пакеты SYN, SYN/ACK и начальный ACK (и данные) включают также опцию MP_CAPABLE, которая имеет переменный размер и служит нескольким целям. Во-первых, проверяется поддержка Multipath TCP удаленным хостом, во-вторых, опция позволяет хостам обменяться данными для аутентификации при создании дополнительных потоков. Полное описание приведено в параграфе 3.1.

```
Хост A                                  Хост B
------                                  ------
MP_CAPABLE                  ->
[флаги]
                            <-          MP_CAPABLE
                                        [ключ B, флаги]
ACK + MP_CAPABLE (+ данные)->
[ключ A, ключ B, флаги, (детали уровня данных)]
```

Повтор ACK + MP_CAPABLE может возникать, если неизвестно о получении пакета. Ниже показаны все возможные обмены при начальной организации субпотока для обеспечения надежности.

```
Хост A (с данными для передачи сразу)   Хост B
------                                  ------
MP_CAPABLE                  ->
[флаги]
```

```
                              <-            MP_CAPABLE
                                            [ключ B, флаги]
ACK + MP_CAPABLE + данные ->
[ключ A, ключ B, флаги, детали уровня данных]

Хост A (с данными для передачи позднее) Хост B
------                                  ------
MP_CAPABLE                    ->
[флаги]
                              <-            MP_CAPABLE
                                            [ключ B, флаги]
ACK + MP_CAPABLE              ->
[ключ A, ключ B, флаги]

ACK + MP_CAPABLE + данные ->
[ключ A, ключ B, флаги, детали уровня данных]

Хост A                                      Хост B (передает первым)
------                                      ------
MP_CAPABLE                    ->
[флаги]
                              <-            MP_CAPABLE
                                            [ключ B, флаги]
ACK + MP_CAPABLE              ->
[ключ A, ключ B, флаги]
                              <-            ACK + DSS + data
                                            [детали уровня данных]
```

2.2. Связывание нового субпотока с соединением MPTCP

Обмен ключами в согласовании MP_CAPABLE обеспечивает материал, который может служить для проверки подлинности конечных точек при организации новых субпотоков. Дополнительные субпотоки начинаются как обычные соединения TCP, но пакеты SYN, SYN/ACK, ACK включают также опцию MP_JOIN.

Хост A инициирует новый субпоток между одним из своих адресов и адресом хоста B. Маркер, созданный из ключа, служит для идентификации соединения MPTCP, к которому добавляется субпоток, а код HMAC[1] - для аутентификации. HMAC использует ключи, переданные при согласовании MP_CAPABLE и случайные значения (nonce) переданные в опциях MP_JOIN. Опция MP_JOIN включает также флаги и Address ID, который можно использовать для указания адреса источника без необходимости отправителю знать о его возможном изменении транслятором NAT. Более подробное описание дано в параграфе 3.2.

```
Хост A                                  Хост B
------                                  ------
MP_JOIN                 ->
[маркер B, nonce A,
 Address ID A, флаги]
                        <-              MP_JOIN
                                        [HMAC B, nonce B,
                                        Address ID B, флаги]
ACK + MP_JOIN           ->
[HMAC A]
                        <-              ACK
```

2.3. Информирование другого хоста о возможных адресах

Набор адресов IP, связанных с многодомным хостом, может меняться в течение срока действия соединения MPTCP. Протокол MPTCP поддерживает добавление или удаление адресов на хосте в явной и неявной форме. Если хост A организовал субпоток со стартовой парой «адрес-порт» IP#-A1 и хочет создать второй субпоток с парой IP#-A2, он просто инициирует организацию субпотока, как описано выше. Удаленный хост получит информацию о новом адресе в неявной форме.

В некоторых обстоятельствах хост может захотеть анонсировать удаленному хосту доступность своих адресов без организации субпотока, например, в случаях, когда NAT препятствует организации соединений в одном из направлений. В приведенном ниже примере хост A информирует хост B о дополнительной паре «адрес-порт» (IP#-A2). Хост B позднее передает опцию MP_JOIN для этого адреса. Опция ADD_ADDR содержит HMAC для аутентификации адреса как отправленного инициатором соединения. Получатель этой опции возвращает ее клиенту для индикации успешного приема. Более подробное описание дано в параграфе 3.4.1.

```
Хост A                                  Хост B
------                                  ------
ADD_ADDR                ->
[Echo-flag=0,
 IP#-A2,
 Address ID IP#-A2,
 HMAC IP#-A2]
                        <-            ADD_ADDR
                                      [Echo-flag=1,
                                       IP#-A2,
                                       Address ID IP#-A2,
                                       HMAC IP#-A2]
```

Имеется соответствующий сигнал для удаления адреса с использованием Address ID из согласования ADD_ADDR. Более подробное описание дано в параграфе 3.4.2.

```
Хост A                                  Хост B
```

[1]Hash-based Message Authentication Code - основанный на хэшировании код аутентификации сообщения.

```
------                                  ------
REMOVE_ADDR                 ->
[Address ID IP#-A2]
```

2.4. Передача данных с использованием MPTCP

Для обеспечения надежной и упорядоченной доставки потока данных через субпотоки, которые могут появляться и исчезать в любой момент, MPTCP использует 64-битовый порядковый номер данных (Data Sequence Number или DSN) для нумерации всех данных, передаваемых через соединение MPTCP. Каждый субпоток имеет свое 32-битовое пространство номеров, используя для них стандартное поле заголовка TCP, а опция MPTCP отображает пространство номеров субпотока на пространство порядковых номеров данных. Таким образом, данные можно передавать повторно в другом субпотоке (с отображением на тот же DSN) при возникновении отказа.

Сигнал последовательности данных (Data Sequence Signal или DSS) переносит отображение последовательности данных (Data Sequence Mapping), состоящее из порядкового номера в субпотоке, номера данных и размера, для которого действительно отображение. Эта опция может также включать подтверждение на уровне соединения (Data ACK) для полученного DSN.

В MPTCP все субпотоки используют общий буфер приема и анонсируют одинаковое приемное окно. MPTCP использует подтверждения двух типов - обычные подтверждения TCP используются в каждом субпотоке при получении отправленных в этот субпоток сегментов независимо от DSN и дополнительные подтверждения для пространства номеров данных. Эти подтверждения отслеживают перемещение потока байтов и сдвигают окно приема.

Более подробное описание дано в параграфе 3.3.

```
Хост А                                  Хост В
------                                  ------
DSS                         ->
[Data Sequence Mapping]
[Data ACK]
[Checksum]
```

2.5. Запрос смены приоритета путей

При организации субпотока хосты могут указать его назначение - основной или резервный путь (включается при отсутствии основных). Во время соединения хост А может запросить изменение приоритета субпотока, передав хосту В сигнал MP_PRIO. Более подробное описание дано в параграфе 3.3.8.

```
Хост А                                  Хост В
------                                  ------
MP_PRIO                     ->
```

2.6. Закрытие соединения MPTCP

Если хосту нужно закрыть имеющийся субпоток, сохранив соединение в целом, он может инициировать обычный обмен TCP FIN/ACK.

Когда хост А хочет уведомить хост В об отсутствии данных для передачи, он передает Data FIN как часть DSS (см. выше). Семантика и поведение сигнала не отличаются от обычного TCP FIN, но применяются на уровне соединения. После успешного приема всех данных в соединении MPTCP это сообщение подтверждается на уровне соединения с помощью Data ACK. Более подробное описание дано в параграфе 3.3.3.

```
Хост А                                  Хост В
------                                  ------
DSS                         ->
[Data FIN]
                            <-          DSS
                                        [Data ACK]
```

Имеется дополнительный метод закрытия соединений, названный Fast Close, который похож на закрытие обычного соединения TCP сигналом RST. Сигнал MP_FASTCLOSE служит для индикации партнеру внезапного завершения соединения и прекращения восприятия данных. Это можно использовать в ACK (надежная сигнализация) или RST (без гарантии), как показано ниже. Более подробное описание дано в параграфе 3.5.

```
Хост А                                  Хост В
------                                  ------
ACK + MP_FASTCLOSE            ->
[ключ B]
[RST во всех других субпотоках] ->
                              <-        [RST во всех субпотоках]

Хост А                                  Хост В
------                                  ------
RST + MP_FASTCLOSE            ->
[ключ B] [во всех субпотоках]
                              <-        [RST во всех субпотоках]
```

2.7. Примечательные особенности

Важно отметить, что сигнализация MPTCP разработана с учетом перечисленных ниже важных требований.

- Для работы через трансляторы NAT в пути адреса указываются идентификаторами Address ID в случаях, где IP-адрес отправителя пакета меняется в NAT. Организация нового потока TCP невозможна, если получатель SYN находится за NAT и для организации субпотоков, когда одна сторона расположена за NAT, в MPTCP применяется сообщение ADD_ADDR.
- MPTCP возвращается к обычному TCP при невозможности работы, например, когда один из хостов не поддерживает MPTCP или промежуточное устройство меняет данные в пакете (см. параграф 3.7).

- Для устранения угроз, указанных в [RFC6181], ключи передаются в сообщениях MP_CAPABLE в открытом виде, сообщения MP_JOIN защищаются с помощью HMAC-SHA256 ([RFC2104] с использованием алгоритма [RFC6234]) и переданных ключей, а также применяются стандартные проверки TCP для других сообщений (гарантия попадания порядковых номеров в окне [RFC5961]). Оставшиеся угрозы для MPTCP v0 отмечены в [RFC7430], а сохранившие влияние на протокол угрозы (т. е. изменение ADD_ADDR) включены в этот документ. Дополнительное рассмотрение вопросов безопасности приведено в разделе 5.

3. Обзор работы MPTCP

В этом разделе описывается работа MPTCP с подробным рассмотрением каждого аспекта работы протокола.

Все операции MPTCP указываются необязательными полями заголовка TCP. Для этого агентство IANA выделило протоколу MPTCP один номер опции TCP (Kind), как указано в разделе 7, а отдельные сигнальные сообщения указываются субтипами (Subtype), значения которых тоже хранятся в реестре IANA (раздел 7). Как во всех опциях TCP поле Length указывает размер в байтах с учетом 2 байтов полей Kind и Length.

В этом документе опции MPTCP всегда указываются символьными именами, такими как MP_CAPABLE, задающими опцию TCP с одним типом MPTCP и значением субтипа, определенным в разделе 7. Поле субтипа занимает 4 бита в начале поля данных (payload) опции, как показано на рисунке 3. Сообщения MPTCP описаны в последующих параграфах.

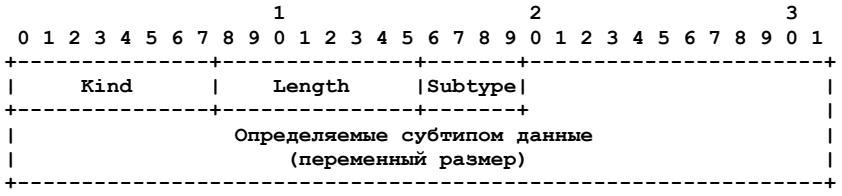


Рисунок 3. Формат опции MPTCP.

Опции MPTCP, связанные с инициирование субпотоков, применяются в пакетах с флагом SYN. Кроме того, имеется опция MPTCP для передачи метаданных, позволяющая объединить сегментированные данные для доставки приложению.

Однако остальные опции являются сигналами, которые не обязательно передавать в определенных пакетах, например, указание дополнительных адресов. Хотя у реализации MPTCP может быть желание передать опции как можно быстрее, объединение всех желаемых опций (MPTCP и обычного TCP, такие как селективные подтверждения SACK [RFC2018]) в одном пакете может оказаться невозможным. Поэтому реализация может выбрать отправку дубликатов ACK с дополнительными сигналами. Это меняет семантику дубликатов ACK, которые в обычном TCP как правило передаются лишь для сигнализации о потере сегмента [RFC5681]. Поэтому реализации MPTCP, получившей дубликат ACK с опцией MPTCP, **недопустимо** считать его сигналом перегрузки. Кроме того, реализации MPTCP **не следует** передавать более 2 дубликатов ACK подряд для отправки лишь опций MPTCP, чтобы предотвратить их ложную интерпретацию промежуточными устройствами как сигнала перегрузки.

Кроме того, стандартные проверки пригодности TCP (такие как контроль попадания в окно порядкового номера и номера подтверждения) **должны** выполняться до обработки сигналов MPTCP, как указано в [RFC5961], а начальные порядковые номера **следует** создавать в соответствии с рекомендациями [RFC6528].

3.1. Инициирование соединений

Инициирование соединения начинается с обмена SYN, SYN/ACK, ACK на одном пути. Каждый пакет включает опцию поддержки множества путей MP_CAPABLE (рисунок 4), которая говорит о поддержке отправителем Multipath TCP и желании использовать протокол в данном соединении.

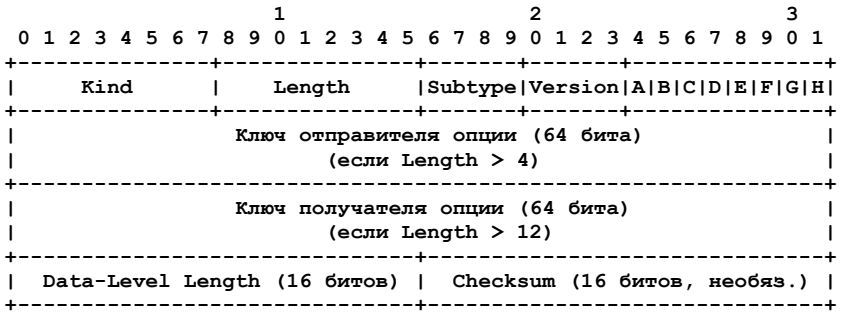


Рисунок 4. Опция MP_CAPABLE.

Обмен MP_CAPABLE в этой спецификации (v1) отличается от заданного в версии v0. Если хост поддерживает несколько версий MPTCP, отправителю MP_CAPABLE **следует** указать наибольший поддерживаемый номер. В ответной опцииMP_CAPABLE получатель указывает желаемый для него номер версии, который **должен** быть не больше номера в исходной опции MP_CAPABLE. Однако имеется предостережение в части согласования версии со старыми реализациями, поддерживающими лишь v0. Узел, поддерживающий v0, ожидает увидеть в опции MP_CAPABLE сегмента SYN ключ инициатора. Однако, если инициатор уже перешел на v1, он не будет помещать ключ в сегмент SYN. В результате принимающая сторона будет игнорировать MP_CAPABLE в этом сегменте SYN и возвращать SYN/ACK без включения опции MP_CAPABLE. Инициатор **может** сразу вернуться к использованию TCP, а **может** попытаться организовать соединение MPTCP v0 (при поддержке этой версии), чтобы убедиться в поддержке получателем старой версии MPTCP. В общем случае соединение MPTCP v0 предпочтительней соединения TCP, однако в конкретном варианте развертывания может быть известно о малой вероятности поддержки другой стороной протокола MPTCP v0 и инициатор может отказаться от попытки создать соединение v0. Инициатор **может** кэшировать информацию о поддерживаемой партнером версии для ее использования в будущих соединениях.

Опция MP_CAPABLE имеет переменный размер и включает разные поля в зависимости от пакета, в который она помещается. Полностью опция MP_CAPABLE показана на рисунке 4.

Опция MP_CAPABLE передается в пакетах SYN, SYN/ACK и ACK, которые создают первый субпоток соединения MPTCP, а также в первом пакете с данными, если инициатор желает отправить их. Данные каждой опции указаны ниже (A - инициатора, B - принимающая сторона).

- SYN (A->B): только первые 4 октета (Length = 4).
- SYN/ACK (B->A): ключ хоста B для этого соединения (Length = 12).
- ACK (без данных) (A->B): ключ A, затем ключ B (Length = 20).
- ACK (с данными) (A->B): ключ A, затем ключ B, Data-Level Length и необязательное поле Checksum (Length = 22 или 24).

Содержимое опции определяется флагами SYN и ACK в пакете, а также полем Length. На рисунке 4 отправителем и получателем названы отправитель и получатель пакета TCP (которые могут быть любым из хостов).

Начальный пакет SYN, содержащий лишь заголовок MP_CAPABLE, служит для указания запрашиваемой версии MPTCP и обмена флагами согласования свойств соединения, как описано ниже.

Эта опция служит для объявления 64-битовых ключей, которые конечные хосты создали для этого соединения MPTCP. Ключи служат для проверки подлинности добавляемых субпотоков данного соединения. Это единственный случай передачи ключа в линию в открытом виде (если не применяется Fast Close, параграф 3.5), все будущие субпотоки идентифицируются 32-битовым маркером (token), который является криптографическим хэш-значеним для ключа. Алгоритм хеширования зависит от выбранного способа аутентификации, а метод выбора описан ниже.

При получении начального сегмента SYN сервер с поддержкой стстояния генерирует случайный ключ и отвечает сегментом SYN/ACK. Метод создания ключа определяется реализацией. Ключ **должен** быть сложнопредсказуемым и **должен** быть уникальным для передающего хоста среди всех его действующих соединений MPTCP. Рекомендации по созданию случайных значений, применяемых в ключах, даны в [RFC4086]. Соединения индексируются на каждом хосте маркером (необратимый хэш ключа). Поэтому для реализации требуется сопоставление каждого маркера с соединением и, следовательно, с ключами для соединения.

Существует вероятность того, что хэш-значения двух ключей будут давать один маркер. Риск этот обычно невелик, если на хосте не используются десятки тысяч одновременных соединений. Поэтому реализации **следует** проверять список маркеров соединений, чтобы убедиться в отсутствии конфликтов, до отправки своего ключа, а при обнаружении конфликта следует генерировать новый ключ. Однако такая проверка будет дорогостоящей для серверов с тысячами соединений. Механизм согласования субпотоков (параграф 3.2) обеспечивает добавление субпотока лишь к нужному соединению (за счет криптографического согласования и проверки маркеров соединения в каждом направлении), а также попадание порядковых номеров в окно. Поэтому в худшем случае при возникновении конфликта маркеров новый субпоток не удастся добавить, но соединение MPTCP будет обеспечивать обычные услуги .

Поскольку генерация ключей зависит от реализации, не требуется, чтобы ключи были просто случайными числами. Реализация может выполнять обмен криптографическим материалом по отдельному каналу (out of band) и создавать из такого материала ключи для идентификации взаимодействующих элементов. Например, реализация может связать свои ключи MPTCP с ключами соединений TLS или SSH вышележащего уровня.

Если сервер не поддерживает состояний, он генерирует свои ключи проверяемым способом, что можно сделать с помощью хэширования квартета адресов и портов, порядкового номера и локального секрета (как это происходит для порядковых номеров TCP [RFC4987]). Таким образом, сервер сможет проверить, является ли он создателем возвращенного в следующей опции MP_CAPABLE ключа. Как и для сервера с состояними, **следует** проверять уникальность маркеров. Если маркеры не уникальны и нет возможности создать другой проверяемый ключ, соединение **должно** вернуться к использованию обычного TCP без отправки опции MP_CAPABLE в SYN/ACK.

Сегмент ACK передает ключи хостов A и B. Это первый случай появления ключа в линии, хотя предполагается, что хост A будет иметь у себя созданный ключ до отправки начального SYN. Возврат ключа B позволяет хосту B работать без поддержки состояния, как описано выше. Поэтому ключ A должен гарантированно доставляться хосту B, а для этого данный пакет должен передаваться с гарантией доставки.

Если у хоста B есть данные для первой передачи, гарантированная доставка ACK с MP_CAPABLE обеспечивается получением данных с опцией MPTCP DSS (параграф 3.3), содержащей DATA_ACK для MP_CAPABLE (первый октет в пространстве номеров данных). Однако, если хост A хочет отправить данные в начале, у него есть два варианта гарантированной доставки ACK с MP_CAPABLE. Если у хоста имеются данные для передачи сразу, первый сегмент ACK (с данными) будет включать также опцию MP_CAPABLE с параметрами данных (Data-Level Length и необязательное поле Checksum, как показано на рисунке 4). Если у хоста A нет данных для передачи сразу, он **должен** включить MP_CAPABLE в первый сегмент ACK, но без параметров данных. Если у A есть данные для передачи, он должен повторить опцию MP_CAPABLE из первого ACK с дополнительными параметрами данных. Эта опция MP_CAPABLE используется вместе с DSS и просто указывает (1) Data-Level Length для данных (payload) и (2) контрольную сумму (если ее использование согласовано). Эти минимальные сведения нужны для организации соединения MPTCP - они позволяют проверить данные (payload) и сучетом того, что это первые данные, становится известным исходный порядковый номер данных (Initial Data Sequence Number или IDSN), поскольку он создается на основе ключа, как указано выше. Перенос ключей в первом пакете данных позволяет механизмам гарантированной доставки TCP обеспечить надежную достаку пакета. Получатель будет подтверждать эти данные на уровне соединения с помощью Data ACK как при получении опции DSS.

Возможны ситуации, когда A и B одновременно попытаются передать начальные данные. Например, если у A в начале не будет данных для передачи, но ему нужно отправить данные до приема чего-либо от B, он будет использовать опцию MP_CAPABLE с параметрами данных (поскольку он не знает о наличии MP_CAPABLE в подтверждении ACK, которое еще не получено). В таком случае B тоже мог передать данные с опцией DSS, но они еще не приняты хостом A. В результате B имеет принятые данные с отображением MP_CAPABLE после того, как он передал данные с опцией DSS. Для корректной обработки таких ситуаций следует принять, что параметры данных в MP_CAPABLE семантически

эквивалентны параметрам в опции DSS и могут использоваться взаимозаменяемо. Похожая ситуация может возникнуть при потере и повторной передаче MP_CAPABLE с данными. Кроме того, в случае выгрузки сегментации TCP опция MP_CAPABLE с параметрами данных может дублироваться в нескольких пакетах и реализация должна быть способна справиться с дубликатами отображений MP_CAPABLE, а также с дубликатами отображений DSS.

Дополнительно обмен MP_CAPABLE позволяет определить безопасное прохождение опций MPTCP в пакетах SYN. Если какая-либо из таких опций отбрасывается, MPTCP будет аккуратно возвращаться к обычному TCP, как указано в параграфе 3.7. Если любая из сторон согласования в любой момент сочтет согласование MPTCP скомпрометированным, например, промежуточное устройство повредит опции TCP или будут получены неожиданные номера ACK, хост **должен** остановить использование MPTCP и больше не включать опции MPTCP в последующие пакеты TCP. Другой хост тогда тоже вернется к обычному TCP с использованием механизма отката (fallback). Отметим, что новые субпотоки **недопустимо** создавать (с использованием процесса из параграфа 3.2), пока не была принята опция DSS, доставленная через путь (см. параграф 3.3).

Как и другие опции MPTCP, MP_CAPABLE начинается с полей Kind и Length, задающих тип и размер опции TCP. Первые 4 бита первого октета тела опции MP_CAPABLE (рисунок 4) определяют субтип опции MPTCP (раздел 7, для MP_CAPABLE эти биты имеют значение 0x0), а 4 оставшихся бита этого октета указывают используемую версию MPTCP (данной спецификации соответствует значение 1).

Второй октет зарезервирован для флагов, перечисленных ниже.

A

В левом бите, обозначенном A, **следует** устанавливать 1 для индикации требования контрольной суммы, если администратор не принял решения об отказе от контрольных сумм (например, в контролируемой среде без промежуточных устройств, способных изменять данные).

B

Второй бит (B) служит флагом расширяемости и в текщих реализациях **должен** иметь значение 0. Флаг будет применяться механизмами расширения в будущих спецификациях и его влияние будут определено позднее. Предполагается (но не обязательно), что этот флаг станет частью дополнительного механизма защиты, который не потребует смены номера версии протокола, но потребует заново определить некоторые элементы согласования. При получении сообщения с установленным флагом B, что непонятно, опция MP_CAPABLE в этом пакете SYN **должна** игнорироваться с возвратом соединения к обычному TCP. Предполагается, что отправитель повторит пакет, используя формат, совместимый с устаревшей спецификацией. Отметим, что размер опции MP_CAPABLE и назначение флагов D - H может меняться при установке B=1.

C

В третьем бите (C) устанавливается 1 для индикации того, что отправитель этой опции не будет воспринимать дополнительные субпотоки MPTCP по адресу и порту отправителя, поэтому получателю **недопустимо** пытаться создавать дополнительные собпотоки с этим адресом и портом. Это повышает эффективность в ситуациях, когда отправитель знает об имеющихся ограничениях (например, он размещен за строгим транслятором NAT или работает через традиционных балансировщик нагрузки L4).

D - H

Оставшиеся биты D - H служат для согласования криптоалгоритма. В данной спецификации назначен лишь правый бит (H), указывающий применение HMAC-SHA256 (см. параграф 3.2). Реализация, поддерживающая лишь этот алгоритм **должна** устанавливать для бита H значение 1, а для битов D - G - значение 0.

Криптоалгоритм **должен** быть указан. Если флаги D - H установлены в 0, опция MP_CAPABLE **должжна** считаться некорректной и игнорироваться (т. е. нужно считать это переходом к обычному согрласованию TCP).

Выбор алгоритма аутентификации влияет также на алгоритм, применяемый для генерации маркера и IDSN. Данная спецификация задает лишь алгоритм SHA-256 (бит H), поэтому маркер **должен** принимать 32 старших бита значения SHA-256 [RFC6234] для ключа. Для IDSN **должны** выбираться 64 младших бита хэш-значения SHA-256 от ключа. Отметим, что ключ **должен** хэшироваться в сетевом порядке байтов. Младшими считаются правые биты результата SHA-256 в соответствии с [RFC6234]. В будущих спецификациях флаги могут позволить выбирать другие алгоритмы для генерации маркеров и IDSN.

Биты криптоалгоритма и контрольной суммы согласуют возможности похожим способом. Контрольная сумма (бит A) **должна** использоваться, если она нужна любому из хостов. Иными словами, для отказа от контрольной суммы оба хоста должны установить в своеих SYN бит A=0. решение подтверждается значением бита A в третьем пакете согласования (ACK). Например, если инициатор указал A=0 в пакете SYN, а отвечающий установил A=1 в SYN/ACK, контрольная сумма **должна** использоваться и инициатор установит A=1 в пакете ACK. Решение о применении контрольной суммы сохраняется реализацией в переменной состояния на уровне соединения. Если A=1 получено хостом, который не хочет использовать контрольную сумму, он **должен** вернуться к обычному TCP, игнорируя опцию MP_CAPABLE, как будто она недействительна.

При согласовании криптоалгоритма у отвечающей стороны есть выбор. Инициатор устанавливает 1 для каждого предлагаемого алгоритма (в данной версии это лишь бит H, который всегда имеет значение 1). Принимающая сторона устанавливает 1 лишь в одном бите (ее выбор). Основанием для такого поведения служит то, что отвечающей стороной обычно является сервер, который может иметь тысячи соединений и поэтому выберет простейший с точки зрения вычислительной сложности алгоритм (в зависимости от своей загрузки). Если отвечающая сторона не поддерживает (или не хочет применять) ни одного из предложенных алгоритмов, она **должна** вернуть отклик без опции MP_CAPABLE, переводя соединение на обычный протокол TCP.

Опция MP_CAPABLE применяется лишь в первом субпотоке соединения для его идентификации. Все остальные субпотоки применяют опцию MP_JOIN (параграф 3.2) для подключения к имеющемуся соединению.

Если SYN включает опцию MP_CAPABLE, но ее нет в SYN/ACK, предполагается, что отправитель SYN/ACK не поддерживает множество путей и сессия MPTCP **должна** перейти в обычную сессию TCP. Если в SYN нет опции MP_CAPABLE, ее включение в SYN/ACK **недопустимо**. Если третий пакет (ACK) не включает опцию MP_CAPABLE, сессия **должна** вернуться в обычный режим TCP. Это делается для совместимости с промежуточными устройствами на пути, которые отбрасывают часть или все опции TCP. Отметим, что реализация **может** пытаться повторить опции MPTCP несколько раз до принятия решения о возврате к обычному TCP (параграф 3.9).

Реакцию на отсутствие подтверждений для пакетов SYN определяет локальная политика. Предполагается, что отправитель в конечном итоге вернется к обычному TCP (без опции MP_CAPABLE) для работы через промежуточные устройства, которые могут отбрасывать пакеты с неизвестными опциями, однако число попыток использовать несколько путей задает локальная политика. В сети возможно изменение порядка пакетов SYN с MPTCP и без него, поэтому финальное решение определяется наличием или отсутствием опции MP_CAPABLE в третьем пакете согласования TCP. Если опции в нем нет, соединение **следует** вернуть к обычному TCP, как указано в параграфе 3.7.

Номер IDSN в соединении MPTCP генерируется из ключа, а алгоритм создания IDSN определяется согласованным алгоритмом аутентификации. Эта спецификация задает лишь алгоритм SHA-256, поэтому для IDSN хост **должен** брать 64 младших бита хэш-значения SHA-256 от ключа хоста, т. е. IDSN-A = Hash(Key-A), а IDSN-B = Hash(Key-B). Такая детерминированная генерация IDSN позволяет получателю обеспечить отсутствие пропусков порядковых номеров при старте соединения. SYN с опцией MP_CAPABLE занимает первый октет пространства номеров данных, хотя его не требуется подтверждать на уровне соединения до отправки первых данных (параграф 3.3).

3.2. Запуск нового субпотока

После запуска соединения MPTCP с помощью обмена MP_CAPABLE можно добавлять в него субпотоки. Хост знает свои адреса и может узнать адреса другого хоста в результате сигнального обмена, описанного в параграфе 3.4. Используя эту информацию, хост может инициировать новый поток черех адреса, которые еще не используются. Новые потоки может создавать любой из участвующих в соединении хостов, но предполагается, что обычно это делает инициатор исходного соедиения (эвристика представлена в параграфе 3.9).

Новый субпоток начинается с обычного обмена TCP SYN/ACK. Используется опция MPTCP MP_JOIN для указания соединения, к которому добавляется новый субпоток. В опции применяется ключевой материал, полученный при начальном согласовании MP_CAPABLE (параграф 3.1), которое задает также криптографический алгоритм для согласования MP_JOIN.

В этом разделе задано поведение MP_JOIN при использовании алгоритма HMAC-SHA256. Опция MP_JOIN включается в пакеты SYN, SYN/ACK и ACK трехэтапного согласования, но формат ее в каждом случае различается.

В первом пакете MP_JOIN (SYN), показанном на рисунке 5, инициатор передает маркер, случайное число и Address ID.

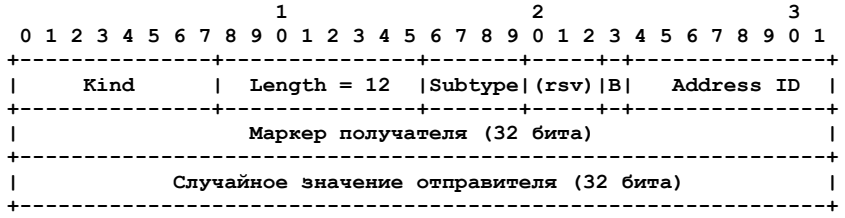


Рисунок 5. Опция MP_JOIN (для начального SYN).

Маркер служит для указания соединения MPTCP и является криптографическим хэш значением от ключа получателя из начального согласования MP_CAPABLE (параграф 3.1). В этой спецификации маркеры для опции создаются алгоритмом SHA-256 [RFC6234], из результата которого берется 32 старших бита. Маркер, включенный в опцию MP_JOIN, является маркером, который получатель применяет для идентификации сообщения, т. е. хост A передает Token-B (создается из Key-B). отметим, что алгоритм генерации хэш-значения может быть переопределен выбором алгоритма криптографического согласования, как определено в параграфе 3.1.

MP_JOIN в SYN содержит не только маркер (не меняется в соединении), но и случайное значение (nonce), служащее для предотвращения replay-атак на метод аутентификации. Рекомендации по созданию таких случайных значений даны в [RFC4086].

Опция MP_JOIN включает поле Address ID, содержащее созданный отправителем опции идентификатор, указывающий адрес отправителя на случай замены адреса в заголовке IP промежуточным устройством. Численное значение этого поля создается отправителем и долно однозначно соответствовать IP-адресу передающего хоста. Поле Address ID позволяет удалять адреса (параграф 3.4.2) без необходимости знать, какой адрес отправителя используется на приемной стороне, что дает возможность работы через NAT с удалением адресов. Address ID также позволяет сопоставлять попытки организации новых субпотоков с сигнализацией адресов (параграф 3.4.1) для предотвращения дублирования субпотоков по одному пути, если MP_JOIN и ADD_ADDR переданы одновременно.

Address ID субпотока, используемые в начальном обмене SYN первого субпотока в соединении, являются неявными и имеют значение 0. Хост **должен** хранить отображение между Address ID и адресами для себя и удаленного партнера. Реализации также нужно знать, какие локальные и удаленные Address ID связаны с каждым из созданных субпотоков при удалении адресов на локальном или удаленном хосте.

Опция MP_JOIN в пакетах с флагом SYN включает также 4 бита флагов, 3 из которых являются резервными и должны быть сброшены (0) при передаче. Последний бит (B) указывает желание отправителя опции использовать этот субпоток (1) в качестве резервного path (B=1) или (2) сразу включить в соединение. Установкой B=1 отправитель опции запрашивает у удаленного хоста отправку данных через этот субпоток лишь в случае отсутствия доступных субпотоков с B=0. Политика для субпотоков подробно рассматривается в параграфе 3.3.8.

При получении SYN с опцией MP_JOIN, содержащей действительный маркер существующего соединения MPTCP приемной стороне **следует** ответить пакетом SYN/ACK с опцией MP_JOIN, содержащей случайное значение и усеченный (64 бита слева) код HMAC. Этот вариант опции показан на рисунке 6. Если маркер неизвестен или хост желает отвергнуть создание субпотока (например, в результате ограничения их числа), он будет возвращать сигнал сброса (RST), аналогичный применяемому для неизвестного порта в TCP, с опцией MP_TCPRST (параграф 3.6), содержащей код причины ошибки MPTCP. Хотя расчет HMAC требует криптографических операций, предполагается, что 32-битовый маркер в MP_JOIN SYN обеспечивает достаточную защиту от атак вслепую на истощение и не требуется механизма, позволяющего отвечающей стороне работать без поддержки состояния на этапе MP_JOIN.

HMAC передают оба хоста - инициатор (A) в третьем пакете (the ACK), отвечающий (B) - во втором (SYN/ACK). Обмен HMAC на этом этапе позволяет хостам сначала обменяться случайными данными (в 2 первых пакетах SYN), которые

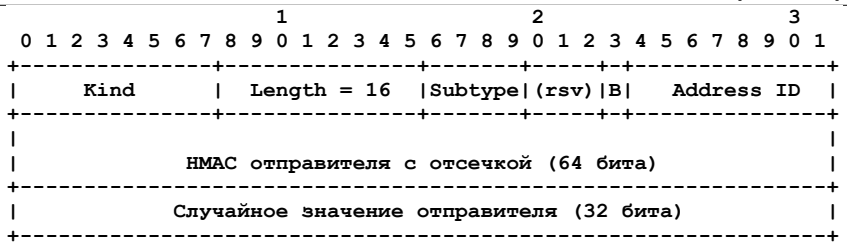


Рисунок 6. Опция MP_JOIN (для откликов SYN/ACK).

служат «сообщением». Эта спецификация задает использование HMAC в соответствии с [RFC2104] вместе с алгоритмом хэширования SHA-256 [RFC6234] с выбором 160 битов слева (20 октетов). Ограниченное пространство потребовало отсечь HMAC в SYN/ACK до 64 битов (слева), но это приемлемо, поскольку применяются случайные значения. Таким образом, у атакующего есть единственный шанс угадать код HMAC, соответствующий переданному ранее партнером случайному значению (если код HMAC не корректен, соединение TCP закрывается, поэтому новое согласование MP_JOIN с новым случайным значением не требуется).

Аутентификационные данные инициатора передаются в его первом пакете ACK (третий пакет согласования), как показано на рисунке 7. Эти данные нужно передать с гарантией доставки, поскольку этот код HMAC передается лишь один раз. Поэтому прием такого **должен** вызывать отпраку обычного отклика TCP ACK и перадача пакета **должна** повторяться при отсутствии этого подтверждения ACK. Иными словами, отправка пакета ACK с MP_JOIN выполняется в субпотоке в состоянии PRE_ESTABLISHED и он переходит в состояние ESTABLISHED лиши в случае приема ACK от получателя. Передача данных в состоянии PRE_ESTABLISHED не допускается. Для резервных битов этой опции отправитель **должен** устанавливать значение 0.

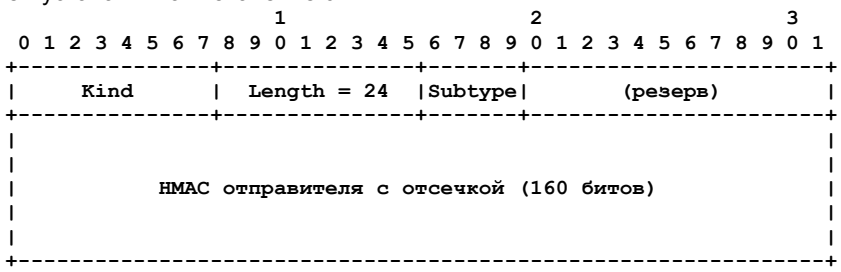
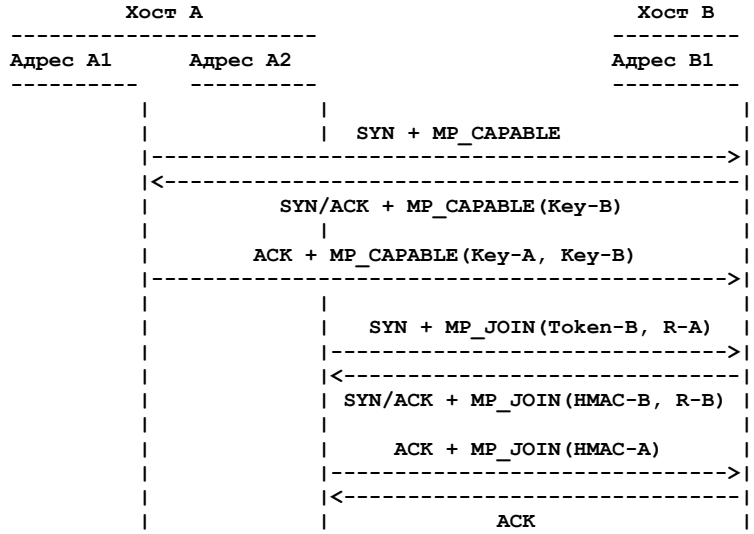


Рисунок 7. Опция MP_JOIN (для первого ACK от инициатора)

Ключом для алгоритма HMAC в случае передачи первого сообщения хостом A будет Key-A, за которым следует Key-B, а в случае первого сообщения от хоста B - Key-B, за которым следует Key-A. Эти ключи были переданы с исходном согласовании MP_CAPABLE. «Сообщением» для алгоритма HMAC в каждом случае является конкатенация случайных значений каждого хоста (R) - для хоста A это будет R-A, за которым следует R-B, а для хоста B - R-B, за которым следует R-A.

Вместе эти опции MPTCP обеспечивают проверку подлинности при организации субпотока, как показано на рисунке 8.




```
HMAC-A = HMAC(Key=(Key-A + Key-B), Msg=(R-A + R-B))
HMAC-B = HMAC(Key=(Key-B + Key-A), Msg=(R-B + R-A))
```

Рисунок 8. Пример использования аутентификации MPTCP.

Если маркер, полученный хостом B, неизвестен или локальная политика не позволяет принять новый субпоток, получатель **должен** передать для этого субпотока TCP RST. Если это приемлемо, следует включить опцию MP_TCPRST с кодом причины Administratively prohibited (параграф 3.6).

Если маркер воспринят хостом B, но код HMAC, возвращенный хосту A, не соответствует ожидаемому, хост A **должен** закрыть субпоток пакетов TCP RST. В этом и последующих случаях при передаче RST, указанной в этом параграфе, отправителю **следует** передавать в RST опцию MP_TCPRST (параграф 3.6) с кодом причины MPTCP-specific error.

Если хост B не получил ожидаемого кода HMAC или опция MP_JOIN отсутствует в ACK, хост **должен** закрыть субпоток отправкой TCP RST.

Если коды HMAC проверены и корректны, оба хоста могут считать партнера проверенным и соответствующим хостам при организации соединения, а также принимают добавление субпотока с соединению.

Если принятый хостом A пакет SYN/ACK не включает опции MP_JOIN, хост A **должен** закрыть субпоток отправкой RST.

Выше приведены все случаи потери опции MP_JOIN. Если опция MP_JOIN вырезается из SYN на пути от A к B и хост B не прослушивает соответствующий порт, он будет отвечать обычным пакетом RST. Если в ответ на SYN с MP_JOIN получен пакет SYN/ACK без MP_JOIN (в результате вырезания на обратном пути или на выходном пути к хосту B, отвечающему как в обычной новой сессии TCP), субпоток не пригоден для использования и хост A **должен** закрыть субпоток отправкой RST.

Отметим, что дополнительные субпотоки могут создаваться между любой парой портов (см. параграф 3.9) и для их организации не требуется явных вызовов на уровне приложения accept или bind. Для привязки нового субпотока к имеющемуся соединению служит маркер, переданный в обмене SYN для субпотока. Затем квинтет субпотока TCP привязывается к локальному маркеру соединения. Одним из следствий этого является возможность использовать для соединения любую пару портов.

Демультиплексирование SYN субпотока **должно** выполняться на основе маркера, что отличается от стандартного TCP, где для демультиплексирования SYN служит порт получателя. После организации субпотока демультиплексирование выполняется с использованием обычного квинтета (адреса, порты, протокол) как в традиционном TCP. Квинтеты отображаются на локальный идентификатор соединения (маркер). Отметим, что хост A будет знать свой локальный маркер для субпотока, даже если он не передавался в линию (передан лишь маркер отвечающей стороны).

3.3. Работа MPTCP и перенос данных

В этом параграфе описана передача данных в MPTCP. На верхнем уровне реализация MPTCP будет принимать один поток данных от приложения и расщеплять его на субпотоки с обеспечением управляющей информации, достаточной для гарантированной доставки и сборки на приемной стороне. В последующих параграфах поведение протокола описано более подробно.

Data Sequence Mapping и Data ACK передаются в опции DSS (рисунок 9). Любое или оба значения могут передаваться в одном DSS в зависимости от флагов. Data Sequence Mapping задает отображение номеров в субпотоке на номера в соединении, а Data ACK подтверждает прием данных на уровне соединения. Эти функции подробно описаны в двух следующих параграфах.

Содержимое опции задает установка фланов, как показано ниже:

- A - Data ACK присутствует;
- a - Data ACK занимает 8 октетов (в противном случае 4);
- M - DSN, SSN (Subflow Sequence Number - порядковый номер в субпотоке), Data-Level Length и Checksum (при согласовании) присутствуют;
- m - DSN занимает 8 октетов (в противном случае 4).

Флаги a и m имеют значение лишь при установке соответствующих флагов A и M, а в ином случае они игнорируются. Максимальный размер опции при установке всех флагов составляет 28 октетов.

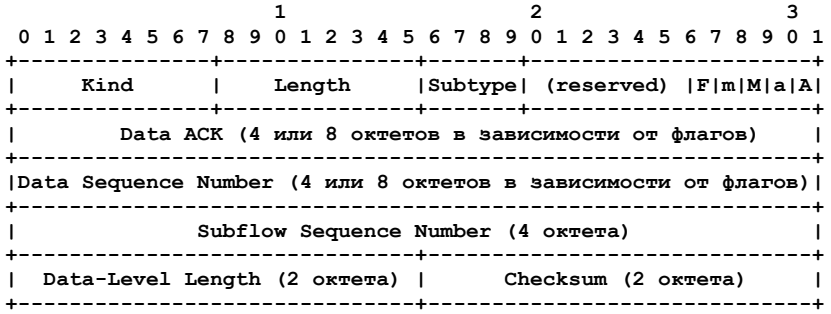


Рисунок 9. Опция DSS (Data Sequence Signal).

Флаг F указывает завершение данных (Data FIN) и его установка означает, что отображение покрывает финальную часть данных от отправителя. Это эквивалент флага FIN в обычном TCP на уровне соединения. Соединение не закрывается, если не было обмена Data FIN, сообщения MP_FASTCLOSE (параграф 3.5) или определяемого реализацией тайм-аута передачи на уровне соединения. Назначение Data FIN и взаимодействие этого флага с флагами FIN на уровне субпотока, а также Data Sequence Mapping описаны в параграфе 3.3.3. Для остальных резервных битов реализации данной спецификации **должны** устанавливать значение 0.

Отметим, что контрольная сумма включается в эту опции лишь при условии ее согласования в обмене MP_CAPABLE (параграф 3.1). О наличии контрольной суммы можно судить по размеру опции. Если контрольная сумма указана, но ее использование не согласовано в MP_CAPABLE, получатель **должен** закрыть субпоток пакетом RST за несограсованное поведение. Если использование контрольной суммы согласовано, но ее нет, олучатель **должен** закрыть субпоток пакетом RST как поврежденный. В обоих случаях в RST **следует** включать опцию MP_TCPRST (параграф 3.6) с кодом причины MPTCP-specific error.

3.3.1. Отображение последовательности данных

Поток данных как целое можно собрать с помощью полей Data Sequence Mapping в опции DSS (рисунок 9), которые задают отображение порядковых номеров в субпотоке на номера в соединении. Это используется получателем для обеспечения упорядоченной доставки на прикладной уровень. Тогда как порядковые номера в субпотоке (обычные номера в заголовках TCP) имеют смысл лишь в данном субпотоке. Предполагается (но не требуется) использование SACK [RFC2018] в субпотоках для повышения эффективности.

Data Sequence Mapping определяет сопоставление пространства номеров субпотока с пространством соединения и задается начальными номерами в субпотоке и на уровне данных, а также размером области отображения. Явное отображение диапазонов данных, а не сигнализация на уровне пакетов выбрано для совместмости с ситуациями, когда сегментирование или объединение TCP/IP происходят вне стека, создающего поток данных (например, при выгрузке сегментрования TCP в сетевой адаптер или при работе через промежуточные прокси-ускорители PEP [RFC3135]). Это позволяет охватить одним отображением весь пакет и может быть полезно при передаче больших объемов данных.

Отображение фиксировано в том смысле, что номере в субпотоке привязывается к порядковому номеру в соединении после обработки отображения. Отправителю **недопустимо** менять объявленное отображение, однако одни и те же номера данных могут сопоставляться с разными субпотоками для повторных передач (параграф 3.3.6). Это также позволяет передать одни и те же данные одновременно в несколько субпотоков для повышения отказоустойчивости или эффективности, особенно в случаях каналов с потерями. Хотя подробная спецификация таких операций выходит за рамки документа, реализации **следует** обрабатывать данные, полученные первыми в субпотоке, как данные, которые следует доставить приложению, а последующие данные с теми же номерами **следует** игнорировать.

Пространство номеров данных задано как абсолютное значение, а нумерация в субпотоках относительна (SYN при запуске субпотока имеет номер 0). Это сделано для устойчивости к промежуточным устройствам (таким как МСЭ), меняющим начальный порядковый номер (Initial Sequence Number или ISN) в субпотоках на случайное значение ISN.

Data Sequence Mapping включает также контрольную сумму охваченных отображением данных, если использование контрольных сумм согласовано в обмене MP_CAPABLE. Контрольные суммы позволяют обнаружить изменение данных (payload) в пути промежуточными устройствами, не знающими о MPTCP. Несовпадение контрольной суммы вызывает отказ субпотока или возврат к обычному TCP, как указано в параграфе 3.7, поскольку MPTCP уже не может точно знать пространство номеров субпотока у получателя для постороения Data Sequence Mapping. Без контрльной суммы данные могут быть доставлены приложению даже когда промежуточное устройство поменяло границы сегментов или собержимое и даже доставило не все сегменты, учтенные в Data Sequence Mapping. Поэтому **рекомендуется** использовать контрольную сумму, если достоверно не известно об отсутствии в пути таких устройств. Применяется стандартная контрольная сумма TCP [RFC0793] для данных, учтенных в отображении, и псевдозаголовка (рисунок 10).

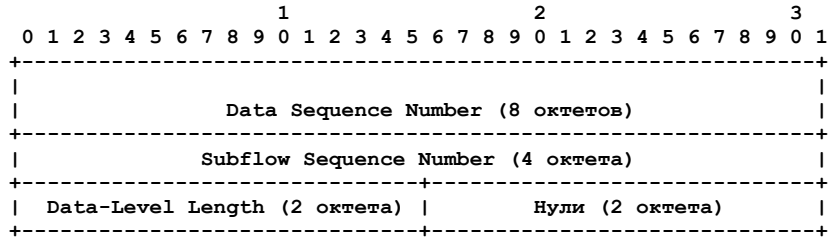


Рисунок 10. Псевдозаголовок для контрольной суммы DSS.

Отметим, что порядковый номер данных в псевдозаголовке всегда имеет размер 64 бита, независимо от размера, используемого в самой опции DSS. Стандартный алгоритм контрольных сумм TCP был выбран потому, что он в любом случае применяется для субпотока TCP и если это происходит для данных до добавления псевдозаголовков, достаточно однократного расчет. Кроме того, аддитивность контрольной суммы TCP позволяет получить контрольную сумму для DSN_MAP путем простого сложения контрольных сумм для данных каждого включенного сегмента TCP и контрольной суммы псевдозаголовка DSS.

Контрольная сумма вычисляется TCP для непрерывной последовательности данных субпотока, поэтому **недопустимо** использовать в субпотоке TCP Urgent Pointer для прерывания имеющегося отображения. Однако при получении срочных данных в субпотоке их **следует** сопоставить с пространством номеров данных и доставить приложению, как это выполняется для Urgent-данных в обычном TCP.

Для предотвращения возможной взаимоблокироваки (deadlock) обработку на уровне субпотока следует выполнять независио от обработки на уровне соединения. Поэтому даже при отсутствии отображения номеров собпотока на номера данных **следует** подтверждать (ACK) доставку данных в субпотоке (если они попадают в окно). Однако эти данные не могут быть подтверждены на уровне соединения (параграф 3.3.2), поскольку понядковые номера данных неизвестны. Реализация **может** удерживать неотображенные данные в течение короткого времени в ожидании приема отображения для них. Неотображенные данные не могут быть считаться попавшими в окно приема на уровне соединения, поскольку это окно связано с порядковыми номерами данных (соединения) и при нехватке памяти у получателя данные будут отбрасываться. Если отображение для номеров из субпотока не поступило в окне приема соединения, субпоток **следует** считать поврежденным и закрыть пакетов RST, а неотображенные данные отбросить.

Порядковые номера данных всегда задаются 64-битовыми значениями и реализация **должна** поддерживать их в таком формате. Если соединение работает с малой скоростью и защита от переполнения (wrap) порядковых номеров не требуется, реализация **может** для оптимизации включать в Data Sequence Mapping и/или Data ACK лишь 32 младших бита порядкового номера, делая это независимо для каждого пакета. Реализация **должна** поддерживать прием и обработку 64- и 32-битовых значений порядковых номеров, но не требуется способность передавать оба.

Реализация **должна** передавать полный 64-битовый номер при работе с достаточно высокой скоростью, когда 32-битовые значения могут переполняться (wrap) в течение максимального срока жизни сегмента (Maximum Segment Lifetime или MSL) [RFC7323]. Размеры DSN, используемые в этих значениях (могут быть разными) объявляются с помощью флагов опции DSS. Реализации **должны** воспринимать 32-битовые DSN и неявно преобразовывать их в 64-битовые, инкрементируя старшие 32 бита при каждом достижении максимума в 32 младших (wrap). **Должна** быть реализована проверка того, что перенос (wrap, например, смена очень большого номера очень малым) произошел в ожидаемое время и не вызван пакетами с нарушенным порядком доставки.

Как и порядковые номера в обычном TCP, номера данных в соединении следует начинать не с 0, а с некого случайного значения, чтобы затруднить захват сессии вслепую. Эта спецификация требует у станавливать в IDSN каждого из хостов младшие 64 бита хэш-значения SHA-256 от ключа хоста, как описано в параграфе 3.1. Это нужно еще для того, чтобы получатель знал ожидаемый номер IDSN и мог проверить отсутствие начальных пакетов в соединении (это особо важно при одновременном начале передачи в двух субпотоках).

Отображение в Data Sequence Mapping **должно** применяться ко всему или части пространства порядковых номеров субпотока в содержащем опцию сегменте TCP. Его не требуется включать в каждый пакет MPTCP, поскольку пространство субпотока в пакете охватывается отображением, известным получателю. Это можно использовать для сокращения издержек в случаях, когда отображение известно заранее. Один из таких случаев возникает при наличии между хостами одного субпотока, а другой - при сборке сегментов данных в большие блоки (несколько пакетов).

Может применяться «бесконечное» отображение при возврате к обычному TCP пуетм сопоставления данных субпотока с данными соединения для продолжающегося соединения (параграф 3.7). Это достигается установкой в поле Data-Level Length опции DSS резервного значения 0. Для контрольной суммы в таком случае также устанавливается 0.

3.3.2. Подтверждение данных

Для обеспечения полной сквозной устойчивости к отказам MPTCP предоставляет подтверждения на уровне соединения, работающие как кумулятивные ACK для соединения в целом. Это делается с помощью поля Data ACK в опции DSS (рисунок 9). Поведение Data ACK аналогично кумулятивным ACK в стандартном TCP - они показывают количество успешно принятых данных (без пропусков). Для сравнения, ACK на уровне субпотока похожи на TCP SACK, с учетом возможно наличия пропусков в потоке данных на уровне соединения. Data ACK указывает следующий порядковый номер данных, которые приемная сторона ожидает получить. Data ACK, как и DSN, могут передаваться в 64-битовых значениях или в виде 32 младших битов. Если данные получены с 64-битовым DSN, они **должны** подтверждаться 64-битовым Data ACK. При получении DSN в 32 битах реализация может передавать 32- или 64-битовый Data ACK и приниматься **должны** оба варианта.

Data ACK подтвеждает, что данные и все требуемые сигналы MPTCP получены и восприняты удаленной стороной. Одной из важнейших функций Data ACK является указание левого края анонсированного окна приема. Как указано в параграфе 3.3.4, окно приема используют все субпотоки и оно привязано к Data ACK. В результате реализациям **недопустимо** использовать поле RCV.WND в сегменте TCP на уровне соединения, если это же значение не указано в опции DSS с полем Data ACK. Кроме того, раздельные подтверждения для соединения и субпотоков позволяют раздельную обработку и получатель может отбрасывать сегменты после подтверждения на уровне субпотока, например, в случае нехватки памяти при нарушении порядка доставкисегментов.

Отправителю MPTCP **недопустимо** освобождать данные из буфера передачи до их подтверждения в Data ACK, принятом через любой из субпоток, и на уровне передавшего эти данные субпотока. Первое условие обеспечивает живучесть соединений, а второе - живучесть и самосогласованность субпотока при необходимости повторной передачи данных. Отметим, однако, что при необходимости несколько раз повторить передачу данных через субпоток возникает риск блокировки окна передачи. В таком случае отправитель MPTCP может прервать субпоток с недопустимым поведением, передав в него RST с опцией MP_TCPRST (параграф 3.6), указывающей код ошибки.

Data ACK **можно** включать во все сегменты, однако **следует** рассмотреть оптимизацию за счет включения Data ACK в сегменты лишь при увеличении Data ACK и это **должно** считаться корректным поведением, обеспечивающим освобождение буфера передачи и снижение издержек при односторонней передаче данных.

3.3.3. Закрытие соединения

В обычном TCP пакет FIN сообщает получателю о завершении передачи данных отправителем. Что субпотоки могли работать назависимо и выглядеть в линии как TCP, FIN в MPTCP влияет лишь на передавший его субпоток. Это дает узлам свободу выбора используемых в каждый момент путей. Для FIN сохраняется семантика обычного TCP, т. е. субпоток закрывается после подтверждения (ACK) обеими сторона сегмента FIN от партнера.

Вызов приложением close() для сокета указывает отсутствие у него данных для передачи и в обычном TCP ведет к отправке FIN для соединения. В MPTCP имеется эквивалентный механизм DATA_FIN. Пакет DATA_FIN служит индикацией отсутствия у отправителя данных для дальнейшей передачи и поэтому может использоваться для проверки доставки всех отправленных данных. DATA_FIN, как FIN в обычном TCP является односторонним сигналом.

DATA_FIN указывается установкой (1) флага F в опции DSS (рисунок 9) DATA_FIN занимает 1 октет (последний) в пространстве номеров соединения. Отметим, что DATA_FIN учитывается в Data-Level Length, но не на уровне субпотока. Например, сегмент с DSN = 80, Data-Level Length = 11 и установленным флагом DATA_FIN будет отображать 10 октетов субпотока с номерами 80-89, а DATA_FIN будет иметь значение DSN = 90, поэтому данный сегмент, включающий DATA_FIN, будет подтвежден DATA_ACK = 91.

Отметим, что при передаче DATA_FIN в сегменте TCP без данных поле DSS **должно** содержать порядковый номер субпотока 0, Data-Level Length = 1 и порядковый номер данных, соответствующий самому. Поле контрольной суммы в таком случае учитывает лишь псевдозаголовок.

Семантика и поведение DATA_FIN такие же как у TCP FIN, но на уровне соединения. Отметим, что DATA_ACK передается лишь после успешной доставки всех данных на уровне соединения, поэтому сигнал DATA_FIN не связан с FIN в субпотоках. Допускается объединение этих сигналов в одном субпотоке, если в дугих субпотоках уже не остается данных. В противном случае может потребоваться повтрная передача данных в других субпотоках. По сути, хосту **недопустимо** закрывать действующик субпотоки, пока он не может сделать это безопасно, т. е. пока для всех остающихся данных не переданы DATA_ACK или сегмент с флагом DATA_FIN не будет единственным из оставшихся.

После подтверждения DATA_FIN все оставшиеся субпотоки **должны** быть закрыты стандартным обменом FIN. Обоим хостам **следует** передать FIN во все субпотоки, чтобы промежуточные устройства могли очистить свои состояния даже для субпотоков с отказами. Поощряется также снижение тайм-аутов (MSL) для субпотоков на конечных хостах после получения DATA_FIN. В частности, любые субпотоки, в очередях которых еще остаются данные (повторенные в другом субпотоке для подтверждения DATA_FIN) **можно** закрыть с помощью RST с опцией MP_TCPRST (параграф 3.6), указывающей причину too much outstanding data (слишком много ожидающих данных).

Соединение считается закрытым, когда DATA_FIN обоих хостов подтверждены DATA_ACK.

Как отмечено выше, стандартный пакет TCP FIN в отдельном субпотоке закрывает лишь данный субпоток. Если все субпотоки были закрыты с помощью FIN, но подтверждения DATA_FIN не получено, соединение MPTCP будет считаться закрытым лишь по тайм-ауту. Это предполагает, что реализации будут находиться в состоянии TIME_WAIT

как на уровне субпотоков, так и для соединения (Приложение D). Это разрешает сценарии break-before-make, когда соединение теряется во всех субпотоках до того, как может быть организовано заново.

3.3.4. Получатель
В обычном TCP окно приема анонсируется в каждом пакете, говоря отправителю, сколько данных получатель готов принять после кумулятивного ACK. Окно служит для управления потоком данных с «торможением» слишком быстрых отправителей, за которыми получатель может не успеть.

MPTCP применяет уникальное окно приема, общее для всех субпотоков. Идея состоит в том, чтобы позволить любому субпотоку передавать данные, пока получатель способен их принять. Вариант с поддержкой окна приема на уровне субпотоков может приводить к остановке некоторых субпотоков, тогда как другие не смогут использовать свое окно.

Окно приема задается относительно DATA_ACK. Как и в TCP, получателю **недопустимо** сжимать окно справа (уменьшать DATA_ACK + окно приема). Получатель будет использовать порядковый номер для решения о восприятии пакета на уровне соединения.

При решении вопросов восприятия пакетов на уровне субпотока выполняется обычная проверка TCP на предмет попадания порядкового номера пакета в разрешенное окно приема. В MPTCP такая проверка выполняется лишь для окна на уровне соединения. Проверку **следует** выполнять на уровне субпотока, чтобы убедиться, что порядковые номера субпотока и отображения соответствуют условию SSN - SUBFLOW_ACK <= DSN - DATA_ACK, где SSN - номер принятого пакета в субпотоке, а SUBFLOW_ACK - RCV.NXT (следующий ожидаемый номер) для субпотока (эквивалентами на уровне соединения являются DSN и DATA_ACK). В обычном TCP попадающий в окно сегмент помещается в очередь упорядоченных или разупорядоченных пакетов. В Multipath TCP происходит то же самое, но на уровне соединения - сегмент помещается в очередь упорядоченных или разупорядоченных пакетов, если он попадает в окно на уровне субпотока и соединения. Стек все еще должен запоминать для каждого субпотока полученные сегменты, чтобы передавать ACK для них в субпотоке. Обычно это реализуется путем хранения разупорядоченных очередей (содержат лишь заголовки без данных) и запоминания кумулятивного ACK.

Для реализаций важно понимать приемлемый размер буфера приема. Нижней границей для полного использования сети будет максимальное произведение пропускной способности и задержки среди путей. Однако этого может оказаться недостаточно при потере пакетов в медленном субпотоке с их потором (параграф 3.3.6). Точной верхней границей будет максимальное время кругового обхода (RTT) среди путей, умноженное на общую пропускную способность путей. Это позволит все потокам работать с полной скоростью при ускоренном повторе передачи по пути с наибольшим RTT. Но даже этого может быть недостаточно для поддержки полной производительности при тайм-ауте повтора на пути с наибольшим RTT. Изучение связи стратегии повтора с размером буфера оставлено на будущее.

3.3.5. Отправитель
Отправитель запоминает анонсы окна приема от полуяателя. Ему следует обновлять свое окно приема лишь при увеличении максимально разрешенного порядкового номера (DATA_ACK + окно приема) в ответ на DATA_ACK. Это важно для возможности использования путей с разными RTT и, следовательно, разными контурами обратной связи.

MPTCP применяет одно окно приема для всех субпотоков и при гарантированной сквозной неизменности окна приема хост всегда может прочитать последнее значение окна приема. Однако некоторые классы промежуточных устройств могут менять окно приема на уровне TCP. Обычно они сокращают предложенное окно, хотя на короткое время могут его расширять (это не будут долгим, поскольку в конечном итоге промежуточное устройство должно успевать доставить данные получателю). Поэтому при разных размерах окон приема в субпотоках для передачи данных MPTCP **следует** при расчетах выбирать наибольшее. Это правило подразумевается в запрете сокращать окно справа.

Отправитель **должен** также помнить окна приема, анонсированные каждым субпотоком. Разрешенным окном для субпотока i будет (ack_i, ack_i + rcv_wnd_i), где ack_i - кумулятивный ACK в субпотоке i. Это предотвратит отправку данных промежуточному устройству, не готовому их принять.

Объединнение этих правил разрешает отправителю передавать сегменты данных с номерами на уровне соединения из диапазона (DATA_ACK, DATA_ACK + receive_window). Каждый из этих сегментов будет отображаться в субпотоки, пока порядковые номера субпотоков попадает в разрешенные для них окна. Отметим, что номера в субпотоках обычно не влияют на управление потоком данных, если для всех субпотоков анонсировано одинаковое окно приема. Эти номера будут влиять лишь на управление потокм данных в субпотоках с меньшим анонсированным окном приема.

Для обеспечения максимальной производительности буфер передачи **должен** быть не меньше приемного буфера.

3.3.6. Надежность и повтор передачи
Data Sequence Mapping позволяет отправителю повторить передачу данных с тем же номером в другом субпотоке. В таких случаях хост **должен** продолжать их передачу в исходном субпотоке для сохранения его целостности (промежуточные устройства могут повторять старые данные и/или отвергать пропуски в субпотоке), а получатель будет игнорировать дубликаты). Хотя это явно не оптимально, с учетом совместимости такое поведение приемлемо, а оптимизация может быть согласована в будущих версиях протокола. Однако следует отметить, что это свойство также позволяет отправителю всегда передавать одни данные с одним номером в разных субпотоках для надежности.

Эта спецификация протокола не задает каких-либо механизмов обработки повторов передачи и многое будет зависеть от локальной политики (параграф 3.3.8). Моэно представить энергичный постор передачи на уровне соединения, когда каждый потерянный пакет повторяется в другом субпотоке (это ведет к расходу полосы, но может снизить задержки на уровне приложения) или осторожные повторы где на уровне соединения передача пакета овторяется лишь по истечении нескольких тайм-аутов на уровне субпотока.

Предполагается, что стандартный механизм постора на уровне соединения будет реализован «вокруг» очереди данных соединения, где сохраняются все сегменты, для которых не получено DATA_ACK. Устанавливается таймер в момент подтвержнения «головы» очереди на уровне субпотока при отсутствии DATA_ACK в соединении. Этот таймер будет защищать от отказов при повторах через промежуточные устройства, заранее передающие ACK.

Отправитель **должен** хранить данные в своем буфере передачи, пока они не будут подтверждены (1) на уровне соединения и (2) на уровне передавшего субпотока. Благодаря этому, отправитель всегда может при возникновении необходимости повторить передачу в том же или ином субпотоке. Особым случаем является отказ субпотока - отправитель обычно повторяет отправку данных в другой доступный субпоток после тайм-аута, продолжая повторные попытки и в отказавшем субпотоке. Отправитель будет принимать отказ потока по достижении предопределенного числа повторов (это **может** быть меньше обычно для TCP предела MSL) или получении сообщения ICMP об ошибке и лишь тогда удалит ожидающие сегменты.

Если использовано множество попыток повтора, показавших плохую работу субпотока, это **может** привести к сбросу хостом этого субпотока с помощью RST. Однако нужны дополнительные исследования для понимания, когда и как сбрасывать плохо работающие субпотоки. Пакет RST для этой цели **следует** сопровождать опцией MP_TCPRST (параграф 3.6) с кодом ошибки Unacceptable performance.

3.3.7. Контроль перегрузок

Субпотоки в соединении MPTCP имеют разные окна насыщения. Для обеспечения беспристрастности в «пробках» и при объединении сетевых ресурсов нужно связать окна каждого субпотока, чтобы направить большую часть трафика в незагруженные каналы. Один из алгоритмов решение такой задачи представлен в [RFC6356], он не обеспечивает совершенного объединения ресурсов, но «безопасен» в том смысле, что его легко развернуть в современной сети Internet. В данном случае подразумевается, что на любом пути субпоток не занимает больше пропускной способности, чем занял бы отдельный поток по этому маршруту и алгоритм можно применять в узких местах сети, не мешая обычным потокам TCP.

Можно представить реализацию в MPTCP разных контроллеров перегрузки, нацеленных на достижение своих свойств в части объединения ресурсов, беспристрастности и стабильности, а также качества обслуживания, надежности доставки и отказоустойчивости. Независимо от применяемого алгоритма, MPTCP нацелен на реализацию механизма контроля перегрузок, обеспечивающего достаточную для принятия верных решений информацию, включающую время и место потери пакетов для каждого субпотока.

3.3.8. Политика для субпотока

В рамках локальной реализации MPTCP хост может применять любую политику передачи трафика по множеству путей. В типичном случае, где нужно обеспечить максимальную пропускную способность, для передачи данных одновременно применяются все доступные пути с согласованным контролем перегрузок [RFC6356]. Однако возможны и иные подходы. Одним из вариантов является «все или ничего», когда второй путь держится в резерве на случай отказа первого, но возможен также вариант с «переполнением», когда второй путь подключается при перегрузке первого. Выбор скорей всего будет связан с расходами на использование канала, но может основываться и на таких свойствах, как задержка или ее вариации, если стабильность (задержек или полосы) важнее пропускной способности. Требования приложений подробно рассмотрены в [RFC6897].

Для эффективного выбора решения отправителю нужны полные сведения о «стоимости» пути, наличие которых маловероятно. Было бы желательно получать сведения о предпочтительных путях от приемной стороны, поскольку та зачастую является многодомной и может платить за расход полосы входящих потоков. Для решения этой задачи опция MP_JOIN (параграф 3.2) включает флаг B, позволяющий хосту указать своему партнеру, какой путь следует считать резервным и применять лишь при отказах других субпотоков (субпоток, где получатель указал B=1, **не следует** применять для передачи данных, пока имеются доступные субпотоки с B=0).

При изменении набора доступных путей хост может сообщить партнеру, например, указав, что субпоток, выступавший как резервный, теперь становится приоритетным среди оставшихся субпотоков. Опция MP_PRIO, показанная на рисунке 11, позволяет изменить флаг B для передавшего эту опцию субпотока.

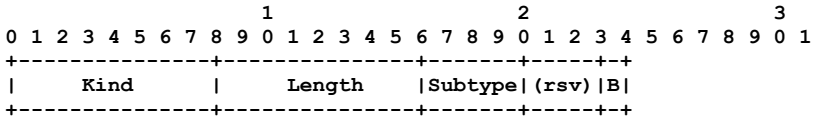


Рисунок 11. Опция MP_PRIO.

Другим вариантом применения MP_PRIO является установка флага B в субпотоке для прекращения его использования перед закрытием и удалением REMOVE_ADDR (параграф 3.4.2), например, для поддержки режима make-before-break, где добавляются новые субпотоки перед закрытием ранее применявшихся.

Следует отметить, что флаг B является запросом получателя данных лишь к их отправителю и тому **следует** прислушиваться к таким запросам. Однако хост не может предполагать, что отправитель выполнит запрос, поскольку локальная политика или технические ограничения могут переопределять запросы MP_PRIO. Отметим, что сигнал применяется в одном направлении и поэтому отправитель может продолжать использование субпотока для передачи данных, даже указав партнеру B=1.

3.4. Обмен информацией об адресах (поддержка путей)

Термин «управление путями» (path management) применяется для обмена информацией о дополнительных путях между хостами, которые в данном случае обеспечиваются наличием у хостов множества адресов. Дополнительная информация об архитектуре MPTCP приведена в [RFC6182].

В протоколе применяется два метода обмена информацией о путях, которые могут работать совместно. Первый метод заключается в прямой организации новых субпотоков (параграф 3.2), когда инициатор имеет дополнительный адрес. Во втором методе (см. ниже) адреса явно передаются другому хосту, чтобы тот мог инициировать новые субпотоки. Механизмы дополняют друг друга и первый из них неявный и проще, а второй сложнее, но явный и более устойчив к отказам. Совместно эти методы позволяют менять адреса «на лету» (это позволяет работать через NAT, поскольку не требует знать адрес отправителя), а также обмениваться ранее неизвестными адресами и адресами другого семейства (например, IPv4 и IPv6).

Ниже перечислено несколько типичных операций протокола.

- Соединение MPTCP исходно организовано между адресом и портом A1 хоста A и адресом и портом B1 хоста B. Если A является многодомным и многоадресным, он может начать новый субпоток с адреса A2 на B1, передав SYN с опцией MP_JOIN от A2 к B1, используя ранее объявленный B маркер для этого соединения. Если B является многодомным, он может попытаться организовать новый субпоток от B2 к A1, используя ранее объявленный маркер A. В обоих случаях SYN передается в порт, уже используемый принимающим хостом в этом соединении.

- Одновременно (или после ожидания) передается опция ADD_ADDR (параграф 3.4.1) через имеющийся субпоток для информирования получателя о дополнительных адресах отправителя. Получатель может использовать эти сведения для организации нового субпотока по полученному адресу. В нашем примере A передает опцию ADD_ADDR, сообщающую B адрес и порт A2. Совместное использование опции в SYN и опции ADD_ADDR, включая тайм-ауты, определяется реализацией и может зависеть от локальной политики.

- Если субпоток A2-B1 организован, хост B может использовать Address ID в опции MP_JOIN для сопоставления этого адреса с опцией ADD_ADDR, которая также прибудет в имеющемся субпотоке и B будет знать о наличии субпотока A2-B1, игнорируя ADD_ADDR. Если же B не получил A2-B1 MP_JOIN SYN, но принял ADD_ADDR, он может попытаться организовать новый субпоток с одного или нескольких своих адресов на адрес A2. Это позволяет создавать субпотоки, когда один из хостов расположен за NAT.

Возможны иные способы применения этих двух механизмов, например, для указания адресов из другого семейства, которое доступно лишь через опцию ADD_ADDR.

3.4.1. Анонсирование адресов

Опция ADD_ADDR в MPTCP анонсирует дополнительные адреса (возможно и порты), доступные на хосте (рисунок 12). Опцию можно использовать в любой момент действующего соединения в зависимости от желания отправителя разрешить множество путей и/или при появлении доступных путей. Как и для других опций MPTCP, отправитель **должен** выполнить стандартные проверки пригодности TCP, например, [RFC5961], до начала действия.

Каждый адрес имеет идентификатор Address ID, который может служить для однозначного указания адреса в соединении при его удалении. Address ID можно также применять для идентификации опций MP_JOIN (параграф 3.2), относящихся к этому адресу, даже при работе через трансляторы адресов. Address ID **должен** однозначно указывать адрес для отправителя опции (в рамках соединения), механизм выделения идентификаторов задает реализация.

Address ID, полученные в MP_JOIN или ADD_ADDR, **следует** сохранять на приемной стороне в структуре данных, которая содержит все отображения Address ID на реальный адрес в соединении (указывается парой маркеров). Таким способом сохраняются сопоставления Address ID, наблюдаемого адреса отправителя и пары маркеров для будущей обработки данных управления для соединения. Отметим, что реализация **может** по своему усмотрению отбрасывать входящие анонсы адресов, например, для предотвращения обновлений состояния или по причине невозможности применить адрес (например, анонс адреса IPv6 хосту IPv4). Поэтому хост **должен** считать анонсы адресов «мягким» состоянием и **может** периодически обновлять их. Отметим, что реализация **может** кэшировать такие анонсы, даже если они сейчас не актуальны, но могут быть применены в будущем, например, адреса IPv4 при использовании IPv6 с ожиданием доступности IPv4 DHCP.

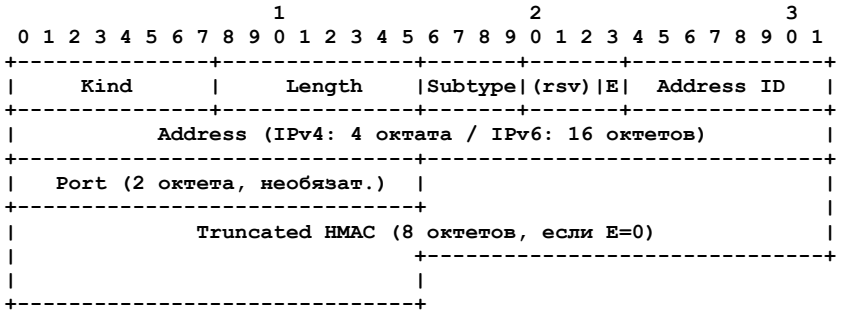


Рисунок 12. Опция ADD_ADDR.

Опция показана на рисунке 12 с указанием размера адресов IPv4. Для IPv6 размер адреса будет 16 октетов (вместо 4).

Два октета, указывающие используемый порт TCP, не обязательны и их наличие определяется по размеру опции. Хотя предполагается, что в ьошьшинстве случаев будет применяться тот же порт, который использован в начальном соединении (например, порт 80 сохранится для всех субпотоков, как и эфемерный порт у клиента), могут возникать ситуации (например, балансировка трафика по портам), где требуется явно указать другой порт. Если порт не задан, MPTCP **следует** пытаться пдключиться к указанному адресу через тот же порт, который уже используется субпотоком, где был передан сигнал ADD_ADDR (см. параграф 3.9).

Параметр Truncated HMAC в этой опции содержит 64 младших (справа) битов HMAC, согласованных и рассчитанных так же, как описано для MP_JOIN в параграфе 3.2. Для этой спецификации MPTCP задан лишь один алгоритм хэширования HMAC [RFC2104], использующий SHA-256 [RFC6234]. Как и в случае MP_JOIN, ключом для алгоритма HMAC при передаче сообщения от хоста Host A будет Key-A, за которым следует Key-B, а при передаче от хоста B - Key-B, затем Key-A. Это ключи, обмен которыми выполнен при начальном согласовании MP_CAPABLE. Сообщением для HMAC являются Address ID, IP-адрес и порт, которые предшествуют HMAC в опции ADD_ADDR. Если порт не указан в опции ADD_ADDR, в HMAC будут включаться два октета со значением 0. Назначение HMAC состоит в предотвращении вставки сигналов ADD_ADDR не уполномоченными элементами для перехвата соединения. Дополнительно HMAC предотвращает замену адреса в процессе передачи, если промежуточному элементу не известен ключ. Если хост получает опцию ADD_ADDR, для которой не удается проверить (подтвердить) HMAC, **следует** игнорировать эту опцию.

После субтипа (перед Address ID) указывается набор из 4 флагов. Данная спецификация задает лишь самый правый флаг E, остальные в настоящее время не используются и **должны** устанавливаться в 0 при передаче и игнорироваться при получении. Флаг E служит для обеспечения надежности этой опции. Поскольку опция часто передается в «чистых» сегментах ACK, ее доставка не гарантирована. Поэтому получатель свежей опции ADD_ADDR (где E=0) будет

передавать ту же опцию назад, но без включения HMAC и с E=1 для индикации получения. В соответствии с локальной политикой отсутствие этого типа «эхо» может указывать исходному отправителю опции ADD_ADDR на необходимость ее повтора.

В связи с широким распространением NAT вполне вероятно, что хост может попытаться анонсировать не маршрутизируемые адреса [RFC1918]. Запрещать такое поведение нежелательно, поскольку возможны ситуации, когда оба хоста имеют дополнительные интерфейсы в одну частную (с немаршрутизируемыми адресами) сеть и хост **может** анонсировать такие адреса. Согласование MP_JOIN для создания нового субпотока (параграф 3.2) обеспечивает механизмы для минимизации угроз безопасности. Сообщение MP_JOIN содержит 32-битовый маркер, который однозначно указывает соединение принимающему хосту. Получив неизвестный маркер, хост будет отвечать пакетом RST. В редком случае, когда (чужой) маркер оказывается действительным на принимающем хосте, огранизация субпотока будет продолжена, но для проверки подлинности потребуется обмен HMAC. Этот обмен будет завершаться отказом, что обеспечивает достаточную защиту от ситуаций, когда 2 несоединенных хоста случайно создают новый субпоток по сигнализации частного адреса. Дополнительное рассмотрение вопросов безопасности, связанных с сообщениями ADD_ADDR которые, ошибочно или злонамеренно создают новые попытки MP_JOIN, рассмотрены в разделе 5.

Хосту, получившему ADD_ADDR, но потерпевшему неудачу при соединении с данным адресом IP и портом, **не следует** продолжать попытки подключения к этому адресу и порту в данном соединении. Отправитель, желающий инициировать новую попытку входящего соединения с ранее анонсированной парой адрес-порт, может обновить информацию ADD_ADDR, передав опцию снова.

Хост может передать сообщение ADD_ADDR с уже выделенным значением Address ID, но для него **должен** использоваться адрес, уже назначенный Address ID. Новая опция ADD_ADDR может указывать тот же или иной порт. Если номер порта отличается, принимающему хосту **следует** попытаться организовать новый субпоток для этого адреса и порта.

Хост, желающий заменить Address ID, **должен** сначала удалить имеющийся идентификатор (параграф 3.4.2).

При нормальной работе MPTCP маловероятно наличие достаточного места в опциях TCP для включения ADD_ADDR вместе с опциями порядковых номеров данных (параграф 3.3.1). Поэтому предполагается отправка реализацией MPTCP опций ADD_ADDR в отдельных сегментах ACK. Однако, как было отмечено выше, реализации MPTCP **недопустимо** считать дубликаты ACK с любой опцией MPTCP, за исключением DSS, индикацией перегрузки [RFC5681] и реализации **не следует** передавать для сигнальных целей более 2 дубликатов ACK подряд.

3.4.2. Удаление адресов

Если в течение срока действия соединения MPTCP анонсированный ранее адрес стал недействительным (например, интерфейс отключили или IPv6 перестал быть предпочтительным), хосту, на которой это влияет, **следует** анонсировать такую ситуацию, чтобы партнер мог удалить субпотоки, связанные с этим адресом. Даже если адрес не используется в соединении MPTCP, но был ранее анонсирован, реализации **следует** анонсировать его удаление. Хост также **может** анонсировать, что действительный адрес IP больше не следует использовать, например, для выполнения процедуры make-before-break (сначала организовать, затем прерывать).

Отзыв адресов выполняется с помощью опции REMOVE_ADDR (рисунок 13), которая удаляет добавленный ранее адрес (список адресов) из соединения и прервывает субпотоки, использующие удаляемый адрес.

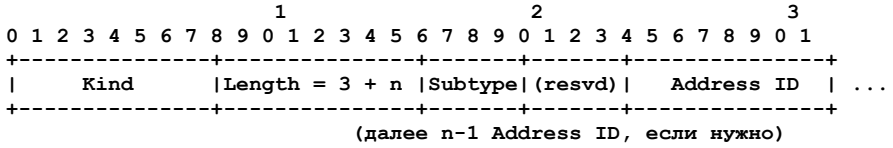


Рисунок 13. Опция REMOVE_ADDR.

В целях безопасности получивший опцию REMOVE_ADDR хост должен убедиться в том, что затрагиваемый опцией путь (или пути) больше не используется, прежде чем инициировать закрытие. Получателю REMOVE_ADDR **следует** сначала инициировать отправку по этому пути TCP keepalive [RFC1122] и при получении отклика удалять путь **не следует**. Если обнаружится, что путь сохраняет активность, получившему опцию хосту **следует** прекратить использование указанного адреса для будущих соединений, но ответственность за разрыв субпотока ложится на хост, передавший REMOVE_ADDR. Перед удалением адреса запрашивающая сторона **может** также передать MP_PRIO (параграф 3.3.8) для запроса прекращения использования пути. **Должны** также выполняться обычные проверки TCP для субпотока (например, проверка корректности порядковых номеров и номеров ACK). Реализация может использовать отказы при таких проверках как часть обнаружения попыток вторжения или записывать событие в системный журнал.

После отправки и приема (если не получено отклика keepalive) этого сообщения **следует** инициировать передачу обоими хостами пакетов RST в соответствующие субпотоки (если это возможно) в качестве «любезности», чтобы позволить промежуточным устройствам очистить состояния до сброса локальных состояний.

Удаление адреса выполняется в соответствии с Address ID для учета наличия NAT и других промежуточных устройств, меняющих адрес отправителя. Если значение Address ID неизвестно, получатель будет игнорировать запрос.

Продолжающий работать субпоток **должен** закрываться с использованием обмена FIN как в обычном TCP, а не с помощью этой опции (см. параграф 3.3.3).

3.5. Опция MP_FASTCLOSE

В обычном TCP имеются средства отправки сигнала RST для резкого закрытия соединения. В MPTCP обычный сигнал RST работает лишь на уровне субпотока, закрывая лишь его и не влияя на другие субпотоки. Соединение MPTCP остается активным на уровне данных, чтобы разрешить передачу обслуживания в режиме break-before-make (создать после прерывания). Поэтому требуется обеспечить «сброс» на уровне MPTCP, позволяющий резко закрыть соединение MPTCP целиком. Это делается с помщью опции MP_FASTCLOSE.

MP_FASTCLOSE служит для информирования партнера о закрытии соединения и прекращении восприятия данных. Причины использования MP_FASTCLOSE зависят от реализации. В обычном TCP не разрешается отправка RST, пока состояние соединения засинхронизировано [RFC0793]. Тем не менее, реализации позволяют отправлять RST в таком состоянии, например, при нехватке ресурсов операционной системы. В таких случаях MPTCP следует передавать MP_FASTCLOSE (рисунок 14).

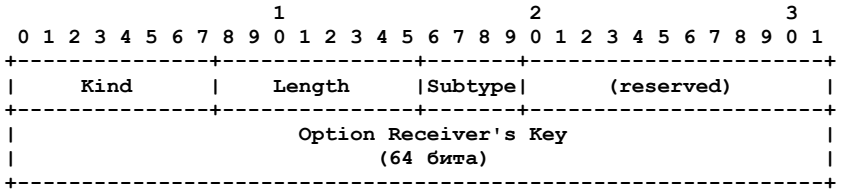


Рисунок 14. Опция MP_FASTCLOSE.

Если хост A хочет закрыть соединение MPTCP, он может сделать это двумя способами, описанными ниже.

Вариант A (ACK)

Хост A передает ACK с опцией MP_FASTCLOSE в субпоток, используя ключ хоста B, объявленный при начальном согласовании соединения. Во все остальные субпотоки хоста A передает обычный TCP RST для закрытия субпотоков. Затем хост A переходит в состояние FASTCLOSE_WAIT.

Вариант R (RST)

Хост A передает RST с общией MP_FASTCLOSE во все субпотоки, используя ключ хоста B, объявленный при начальном согласовании соединения. Хост A может незамедлительно разорвать все субпотоки соединение.

Если хост A выбрал вариант A и передал ACK с опцией MP_FASTCLOSE, соединение проходит указанные ниже стадии.

- При получении хостом B пакета ACK с опцией MP_FASTCLOSE, содержащей действительный ключ, этот хост (B) отвечает в том же субпотоке пакетом TCP RST и закрывает все субпотоки, передавая в них TCP RST. После этого хост B может закрыть соединение MPTCP целиком (с прямым переходом в состояние CLOSED).
- Как только хост A получит TCP RST для остающихся субпотоков, он будет закрывать эти субпотоки и разрывать соединение целиком (переход из состояния FASTCLOSE_WAIT в CLOSED). Если хост A вместо TCP RST получает MP_FASTCLOSE , это говорит об одновременной попытке обеих сторон закрыть соединение. Хосту A следует ответить сигналом TCP RST и разорвать соединение.
- Если хост A не получает TCP RST в отклике на свою опцию MP_FASTCLOSE в течение тайм-аута повтора RTO (для субпотока, где была передана опция MP_FASTCLOSE), ему **следует** повторить MP_FASTCLOSE. Чтобы предотвратить слишком долгое удержание соединения, число повторов передачи **следует** ограничить (предел зависит от реализации), **рекомендуется** значение 3. Если хост A не получил в ответ TCP RST, ему **следует** передать TCP RST с опцией MP_FASTCLOSE, когда он освобождает состояние, чтобы очистить состояния на промежуточных устройствах.

Если хост A решит закрыть соединение с помощью варианта R и передаст RST с опцией MP_FASTCLOSE, хост B при получении RST с опцией MP_FASTCLOSE, содержащей действительный ключ, разрывает все субпотоки путем передачи TCP RST. После этого хост B может закрыть соединение MPTCP цкликом (переходит в состояние CLOSED).

3.6. Сброс субпотока

Реализации MPTCP может также потребоваться передать обычный TCP RST для форсированного закрытия субпотока. Хост передает TCP RST, чтобы закрыть субпоток или отвергнуть попытку создания субпотока (MP_JOIN). Чтобы принимающий хост мог значть причину закрытия или отклонения субпотока, в пакет TCP RST **можно** включить опцию MP_TCPRST (рисунок 15). Хост **может** использовать эту информацию для принятия решения, например, о повторении попытки снова создать субпоток сразу или позднее, а также отказе от нее.

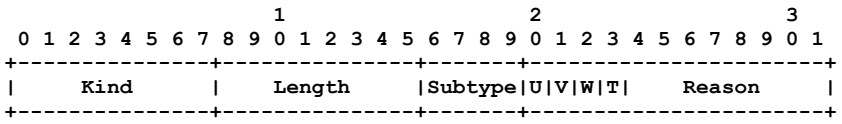


Рисунок 15. Опция MP_TCPRST.

Опция MP_TCPRST содержит код причины, который позволяет отправителю опции предоставить больше информации о причине прерывания субпотока. В 12-итовом поле опции первые 4 бита зарезервированы для флагов (в настоящее время определен лишь 1), а остальные октеты служат для указания причины прерывания субпотока, на основе чего получатель **может** вывести информацию о применимости этого пути.

Флаг T используется отправителем для указания временного (Transient, T = 1) или постоянного (Permanent, T = 0) состояния ошибки. Если ошибка считается временной отправителем сегмента RST, получатель этого сегмента **может** попытаться создать субпоток для этого соединения снова, используя тот же путь. Время, которое следует выждать перед повтором, зависит от реализации, но **следует** учитывать свойства отказа, заданные представленным кодом причины. Если ошибка считается постоянной, получателю сегмента RST **не следует** повторять попытки организации субпотока для данного соединения по этому пути. Флаги U, V и W эта спецификация не задает, резервируя их на будущее. Реализации этого документа **должны** сбрасывать (0) эти флаги, а получатели **должны** игнорировать их.

8-битовое поле Reason указывает код причины прерывания субпотока.

Неуказанная ошибка (0x00)

Применяется по умолчанию и предполагает, что субпоток более не доступен. Наличие этого кода показывает, что сигнал RST был подан понимающим MPTCP устройством.

Ошибка, связанная с MPTCP (0x01)

Ошибка обнаружена при обработке опций MPTCP. Это обычный код в случае отправки RST для закрытия субпотока по причине недействительного отклика.

Нехватка ресурсов (0x02)

Этот код говорит о нехватке у передавшего его хоста ресурсов для поддержки прерываемого субпотока.

Административный запрет (0x03)

Запрошенный поток запрещен правилами передавшего код хоста.

Слишком много ожидающих данных (0x04)

Указывает что слишком много данных, которые требуется передать через прерываемый поток, уже подтверждено в других субпотоках. Это может быть обусловлено кратковременной недоступностью пути и эффективней будет сбросить субпоток и начать заново, нежели повторно передавать данные из очереди.

Неприемлемая производительность (0x05)

Производительность субпотока слишком мала по сравнению с другими субпотоками соединения Multipath TCP.

Влияние промежуточного устройства (0x06)

Обнаружены помехи со стороны промежуточного устройства, делающие сигнализацию MPTCP недействительной (например, несовпадение контрольной суммы).

3.7. Возврат к TCP

Иногда на пути могут присутствовать промежуточные устройства, препятствующие работе MPTCP. Пртокол MPTCP разрабатывался с учетом разных промежуточных устройств (раздел 6), однако сохраняется возможность отказа отдельных субпотоков работать в рамках требований MPTCP. К таким случаям относится потеря опций MPTCP и изменение содерживаго пакетов пути. При возникновении таких событий требуется «откат» к обычной, безопасной работе, что можно сделать путем возврата к оычному протоколу TCP или удаления проблемного субпотока.

При старте соединения MPTCP (т. е. в первом субпотоке) важно обеспечить полную совместимость пути с MPTCP и передачу каждому хосту всех требуемых опций MPTCP. При описанном в параграфе 3.1 согласовании **следует** возвращаться к обычному TCP, если любое из сообщений SYN не включает опций MPTCP. Так же следует (и желательно) поступать, если хост не поддерживает MPTCP или путь не позволяет передать опции MPTCP. При попытке присоединения к имеющемуся соединению MPTCP (параграф 3.2) следует закрывать субпоток с помощью опции MP_JOIN, если путь пне поддерживает MPTCP и опции MPTCP не приходят в SYN.

Однако следует рассмотреть еще ожин важный случай, когда опции MPTCP приходят в пакетах SYN, но не приходят в обычных сегментах. Если субпоток является первым в соединении и находящиеся в процессе передачи данные являются смежными (непрерывными), проблему можно решить с помощью приведенных ниже правил.

- Отправитель **должен** включать опцию DSS с Data Sequence Mapping в каждый сегмент, пока один из переданных сегментов не будет подтвержден с опцией DSS, содержащей Data ACK. При получении такого подтверждения отправитель будет знать о прохождении опции DSS в обоих направлениях и сможет передавать эту опцию реже (не в каждом сегменте).

- Однако при получении ACK (не просто для SYN) без опции DSS с Data ACK, отправитель узнает об отсутствии поддержки MPTCP на пути. Если это происходит в дополнительном субпотоке (т. е. начатом с опцией MP_JOIN), хост **должен** закрыть поток с помощью пакета RST, в который **следует** включить опцию MP_TCPRST (параграф 3.6) с кодом причины "Middlebox interference".

- В случае получения ACK в первом потоке (открытом с опцией MP_CAPABLE) до создания дополнительных субпотоков, реализация **должна** отказаться от MPTCP и рернуться к протоколу TCP. Отправитель передает одно финальное отображение Data Sequence Mapping с нулевым значением Data-Level Length, указывающим бесконечное отображение (для информирования другой стороны в случае отбрасывания опций лишь в одном направлении), а затем возвращается к передаче данных в одном субпотоке без опций MPTCP.

- Если субпоток прерывается в процессе работы, например, при смене маршрутизации с переходом на путь, не поддерживающий опции MPTCP, после обнаружения этого (по заполнению буфера передачи на уровне субпотока или по отсутствию отображений для отправки DATA_ACK), субпоток **следует** считать оборванным и закрывать с помощью RST, поскольку данные невозможно доставить на уровень приложения и нет возможности надежно передать сигнала возврата к TCP. В пакет RST **следует** включать опцию MP_TCPRST (параграф 3.6) с кодом причины "Middlebox interference".

Эти правила должны охватывать все случаи отказов на прямом или обратном пути, назависимо от того, кто первым отправил данные (сервер или клиент).

До сих пор рассматривалась потеря опций MPTCP в начале или в процессе работы соединения. Как указано в параграфе 3.3, каждая часть данных, для которой имеется отображение, охватывается контрольной суммой, если применение контрольных сумм согласовано. Этот механизм служит для обнаружения изменений, которые промежуточные устройства могли внести в данные (добавление, удаление, изменение). При любом изменении данных в пути контрольная сумма не совпадет. Применение контрольных сумм позволяет также обнаружить изменение размера данных в субпотоке, которое делает отображение Data Sequence Mapping недействительным. Отправитель больше не знает, с каким порядковым номером в субпотоке будет работать получатель (промежуточные устройства будут передавать фиктивные ACK) и не может указывать новые отображения. Кроме допустимого на программном уровне изменения данных возможны изменения при пересечении границ сегмента MPTCP, повреждающие данные. Поэтому никаким данным от начала сегмента, в котором не совпала контрольная сумма, доверять нельзя.

Отметим, что при работе без контрольных сумм этот механизм откьа нельзя применять, пока нет сигнала от выше- или нижележащего уровня, информирующего реализацию MPTCP об изменении данных (payload).

При использовании множества субпотоков находящиеся в процессе передачи субпотоком данные могут не быть непрерывной частью общего потока, распределеяемого по разным субпотокам. Из-за упомянутых выше проблем невозможно определить внесенные в данные изменения (в частности, изменения нумерации в субпотоках). Поэтому невозможно восстановить субпоток и затронутые изменениями потоки нужно немедленно закрывать с помощью RST, включающего опцию MP_FAIL (рисунок 16), которая указывает порядковый номер в начале сегмента (определяется по Data Sequence Mapping) с несовпадением контрольной суммы. Отметим, что опция MP_FAIL требует использования полного 64-битового номера даже в случаях передачи 32-битовых номеров в сигналах DSS для пути.

Получатель опции **должен** отбросить все данные после указанного номера и **недопустима** отправка для них DATA_ACK (параграф 3.3.6).

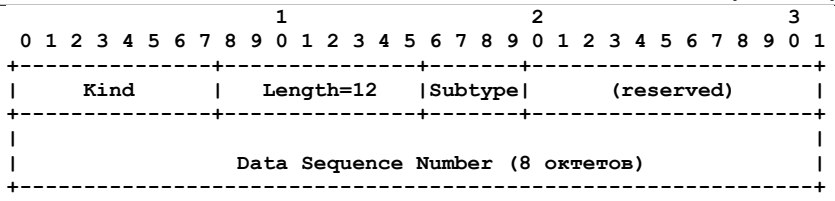


Рисунок 16. Опция MP_FAIL.

Особым случаем является отказ при проверке контрольной суммы в единственном субпотоке. Если известно, что неподтвержденные данные, которые еще в сети, непрерывны (обычно это врено в случае с одним субпотоком), можно применить бесконечное отображение к потоку без необходимости сначала закрыть его, по существу отключая все последующие сигналы MPTCP. В этом случае при обнаружении получателем ошибки в контрольной сумме при единственном субпотоке, он будет возвращать опцию MP_FAIL в пакете ACK на уровне субпотока, указывая порядковый номер данных (не субпотока) в начале сегмента, где найдена ошибка. Отправитель получит эту информацию и при непрерывности неподтвержденных данных будет сообщать о бесконечном отображении. Это отображение будет опцией DSS (параграф 3.3) для первого нового пакета, содержащей отображение Data Sequence Mapping, указанное задним числом и задающее начало порядковых номеров субпотока в наиболее свежем сегменте, который был (как известно) доставлен без повреждений (т. е. для него получено DATA_ACK). С этого момента изменения данных промежуточным устройством не будут влиять на MPTCP, поскольку поток данных эквивалентен обычно сессии TCP. Хотя в теории путь может быть поврежден лишь в одном направлении (и сигнал MP_FAIL будет влиять токо на одно направление), для упрощения реализации получатель MP_FAIL все равно **должен** отвечать опцией MP_FAIL в обратном направлении и полностью возвращаться к обычной сессии TCP.

В редких случаях, когда данные не являются непрерывными (применяется лишь один субпоток, но в нем повторно передаются данные из недавно неаккуратно закрытого субпотока), получатель **должен** закрыть субпоток сигналом RST с опцией MP_FAIL. Получатель опции **должен** отбросить все данные после указанного порядкового номера. Отправитель **может** попытаться создать новый субпоток, относящийся к тому же соединению и в этом случае ему **следует** незамедлительно перевести этот субпоток в режим с одним путем, установив бесконечное отображение Data Sequence Mapping. Это отображение будет начинаться с порядкового номера данных из опции MP_FAIL.

После передачи отправителем бесконечного отображение он **должен** использовать лишь ACK субпотока для очистки своего буфера передачи. Это обусловлено тем, что Data ACK могут утратить выравнивание с ACK при вставке или удалении данных промежуточным устройством. Получателю **следует** прекратить генерацию Data ACK после приема бесконечного отображения.

При «откате» соединения с помощью бесконечного отображения данные может передавать лишь один субпоток и в ином случае получатель не сможет упорядочить данные. На практике это означает, что все потоки MPTCP, кроме одного, будут прерваны. Когда MPTCP возвращается к обычному TCP, последующий переход к MPTCP **недопустим**.

Следует подчеркнуть, что MPTCP не пытается пердотвратить использование промежуточных устройств, которые могут менять данные. Знающие о протоколе MPTCP промежуточные устройства могут поддерживать такую функциональность и даже менять контрольные суммы.

3.8. Обработка ошибок

В дополнение к описанному выше механизму «отката» может потребоваться обработка стандартных ошибок TCP специфическими для MPTCP сопособами. Изменение семантики, например, релевантность RST, рассматривается в разделе 4. По возможности сохраняется обычное для TCP поведение. Ниже приведен список ошибок и соответствующее поведение MPTCP.

- При неизвестном маркере в опции MP_JOIN (а также отказе при проверке HMAC для опции MP_JOIN в ACK или отсутствии MP_JOIN в отклике SYN/ACK) передается RST (аналогично поведению TCP для неизвестного порта).
- При выходе DSN за пределы окна (при нормальной работе) данные отбрасываются без передачи Data ACK.
- Запросы на удаление неизвестных Address ID просто игнорируются.

3.9. Эвристика

Имеется много эвристических подходов, которые нужны для повышения производительности и развертывания, но не требуются для корректности протокола. Такие вопросы рассматриваются в этом параграфе. Обсуждение буферизации и поведения окна приема приведено в параграфах 3.3.4 и 3.3.5, а повтороной передачи - в параграфе 3.3.6.

3.9.1. Использование портов

При типичных вариантах работы реализации MPTCP **следует** использовать порты, которые уже применяются. Иными словами, портом назначения в SYN с опцией MP_JOIN **следует** указывать тот же удаленный порт, который был задан для первого субпотока в соединении. Локальному порту для таких SYN **следует** совпадать с локальным портом для первого субпотока (поэтому реализации **следует** резервировать эфемерные порты для всех локальных адресов IP), хотя в некоторых случаях это невозможно. Эта стратегия предназначена для обеспечения максимальной вероятности прохождения пакетов SYN через МСЭ или NAT у получателя и предотвращения путаницы для программ сеетвого мониторинга.

Однако во многих случаях хост пожелает указать конкретный порт, который следует использовать. Эта возможность обеспечивается опцией ADD_ADDR, описанной в параграфе 3.4.1. Таким образом, можно использовать множество субпотоков между парой адресов с балансировкой нагрузки через разные порты (например, с некоторыми реализациями ECMP [RFC2992]).

3.9.2. Отложенный запуск и симметрия субпотоков

Многие соединения TCP краткосрочны и включают лишь несколько сегментов, поэтому они ничего не выиграют от использования MPTCP. Поэтому нужен эвристический подход для решения вопроса о применении MPTCP. Экспериментальные развертывания показали возможность применения MPTCP в разных сценариях, поэтому реализациям, вероятно, придется придется учитывать такие факторы, как тип передаваемого трафика и продолжительность сессии. Эту информацию может сообщать прикладной уровень.

Для стандартного трафика TCP предлагаемая эвристика общего назначения, которую **можно** использовать, описана ниже.

Если у хоста есть данные, буферизованные для партнера (это подразумевает, что приложение получило запрос на такие данные), хост создает 1 субпоток для каждого начального окна буферизованных данных. При этом следует учитывать ограничение скорости добавления новых субпотоков, а также ограничение числа потоков для одного соединения. Хост может менять эти значения на основе свединий о трафике и характеристиках пути. Отметим, что такой эвристики может оказаться недостаточно. Трафик многих распространенных приложений (например, загрузка файлов) является асимметричным и многодомный хост может оказаться клиентом, который никогда не заполнит свои буферы и на основании такой эвристики просто не будет применять MPTCP. Здесь помогут расширенные API, позволяющие приложениям указать требования к трафику, которые помогут принять решение.

Можно применить дополнительную эвристику на основе времени с созданием дополнительных субпотоков по истечении заданного интервала. Это решило бы отмеченную выше проблему, а также обеспечило бы отказоустойчивость для долгосрочных низкоскоростных соединений.

Другая проблема заключается в возможных попытках одновременной организациями хостами субпотока для одной пары адресов, ведущих к неэффективному использованию ресурсов. При использовании для всех субпотоков одной пары портов, как рекомендовано выше, эта проблема решается стандартной логикой TCP при одновременной организации соединения, в результате чего между парой адресов будут создаваться единственный субпоток. Если хост не поддерживает одновременное открытие TCP, **рекомендуется** вносить некий элемент случайности для времени ожидания при создании новых субпотоков. Если же хост сообщает о возможности использовать дополнительные порты (например, при поддержке ECMP на пути), такая эвристика работать не будет.

Здесь рассмотрены некоторые факторы, которые следует учитывать в эвристике MPTCP, но это не обязатель.

3.9.3. Обработка отказов

Требования к обработке неожиданных сигналов в MPTCP приведены в параграфе 3.8. Однако имеются случаи отказов, когда хосту приходится выбирать поведение. Например, в параграфе 3.1 указано, что хосту **следует** попытаться вернуться к обычным TCP SYN после одного или нескольких отказов MPTCP SYN в соединении. Хост может хранить системный кэш для такой информации, чтобы отказаться от MPTCP для определенного целевого хоста и даже для интерфейса, при повторяющихся отказах MPTCP. Продолжительность кэширования будет зависеть от реализации.

Другой случай может возникать при отказе согласования MP_JOIN. В параграфе 3.8 сказано, что некорректное согласование **должно** приводить к закрытию субпотока командой RST. Хост с активной системой предотвращения вторжений может заблокировать пакеты MP_JOIN от источника при возникновении множественных отказов MP_JOIN. С точки зрения инициатора при отказе MP_JOIN **не следует** пытаться организовать соединение с тем же адресом IP и портом в течение срока едйствия соединения, если другой хост не обновит информацию с помощью опции ADD_ADDR. Отметим, что опция ADD_ADDR является лишь информационной и не гарантирует попытки соединения от другого хоста.

Кроме того, реализация может узнать из множества соединений, что с некоторыми интерфейсами или адресатами постоянно связаны отказы и решить, что по умолчанию для них не следует пытаться использовать MPTCP. interfaces or addresses. Может также изучаться поведение субпотоков, в которых регулярно возникают отказы при работе с целью временного исключения соответствующих путей.

4. Вопросы семантики

Для поддержки работы по нескольким путям семантика некоторых понятий TCP была изменена (см. ниже).

Sequence number - порядковый номер
: Порядковый номер (в заголовке) TCP относится к субпотоку. Чтобы получатель мог упорядочить данные приложения, применяется дополнительная нумерация данных на уровне всего соединения. В этом пространстве начальный сегмент SYN и финальный сегмент DATA_FIN занимают по 1 октету. Это сделано для того, чтобы эти сигналы подтверждались на уровне соединения. Применяется явное отображение порядковых номеров данных на номера в субпотоках, передаваемое через опции TCP в пакетах данных.

ACK - подтверждение
: Поле ACK в заголовке TCP подтверждает лишь порядковый номер в субпотоке, а не в пространстве номеров данных. реализациям **не следует** пытаться вывести номера данных из ACK в субпотоке. Это разделяет обработку на уровнях соединения и субпотоков конечным хостом.

Duplicate ACK - дубликат подтверждения
: Дубликат ACK с любой сигнализацией MPTCP (кроме опции DSS) **недопустимо** считать сигналом перегрузки. Чтобы ограничить шансы не понимающим MPTCP элементом ложно трактовать дубликаты ACK как индикацию перегрузки, MPTCP **не следует** передавать подряд более двух дубликатов ACK с опциями MPTCP (кроме DSS).

Receive Window - окно приема
: Окно приема в заголовке TCP указывает размер свободного пространтсва в буфере на уровне соединения (в отличие от пространства субпотока) на приемной стороне. Семантика похожа на обычный TCP, но окно приема должно интерпретироваться передающей стороной относительно порядкового номера в DATA_ACK, а не в заголовке ACK субпотока. Это сохраняет привычный контроль перегрузок. Отметим, что некоторые промежуточные устройства могут менять окно приема, поэтому хосту **следует** использовать максимальное из недавно наблюдавшихся в субпотоках значение, а также поддерживать окна на уровне субпотоков для их обработки.

FIN - завершение

Флаг FIN в заголовке TCP применяется лишь к субпотоку, где он был передан, а не к соединению в целом. На уровне соединения роль FIN играет опция DATA_FIN.

RST - сброс

Флаг RST в заголовке TCP применяется лишь к субпотоку, где он был передан, а не к соединению в целом. Опция MP_FASTCLOSE обеспечивает функциональность Fast Close (RST) на уровне соединения MPTCP.

Address List - список адресов

Поддержка списка адресов (информации о доступных адресах IP на локальном и удаленном хосте) обеспечивается на уровне соединения (а не субпотока, хоста или пары взаимодействующих хостов). Это позволяет приложению задавать локальные правила на уровне соединения. Добавление адреса в соединение (явное в сообщении ADD_ADDR или неявное в опции MP_JOIN) не влияет на другие соединения между хостами.

5-tuple - квинтет

Квинтет (5-tuple) из протокола и пар локального и удаленного адресов и портов, представляемый API ядра уровню приложения без поддержки множества путей, который применяется в первом субпотоке, даже если позднее субпоток был закрыт и удален из соединения. Этот и другие связанные API вопросы рассматриваются в [RFC6897].

5. Вопросы безопасности

Как указано в [RFC6181], добавление поддержки нескольких путей в TCP создает множество новых классов угроз. Для их предотвращения в [RFC6182] представлен набор требований к защитным решениям для MPTCP. Основная цель защиты MPTCP состоит в том, чтобы ьыть «не хуже» обычного TCP. Основные требования защиты указаны ниже.

- Механизм проверки соответствия сторон согласования субпотока участникам исходного соединения.
- Проверка возможности приема партнером трафика по новому адресу до начала его применения в соединении.
- Защита от повторного использования, т. е. контроль «свежести» запросов на удаление и добавление.

Для решеения этих задач в MPTCP служит алгоритм согласования с хешированием, описанный в параграфах 3.1 и 3.2.

защита соединения MPTCP зависит от ключей, совместно применяемых й раз при старте первого субпотока и больше не передаваемых через сеть (если не применяется механизм Fast Close, параграф 3.5). Для упрощения демультиплексирования и предотвращения передачи криптографического материала в новых субпотоках применяется усеченный хэш этих ключей как идентификатор соединения (token). Конкатенация ключей служит ключом для создания кодов HMAC, используемых при создании субпотоков, для проверки подлинности сторон и возможности получения партнером трафика, переданного по новому адресу. Replay-атаки все еще возможны, если применяются лишь ключи, поэтому в согласовании используются одноразовые случайные числа (nonce) на обеих сторонах — это обеспечивает разные значения HMAC в каждом согласовании. Рекомендации по генерации случайных чисел для ключей приведены в [RFC4086] и рассмотрены в параграфе 3.1. Значения nonce действительны в течение попытки организовать соединение TCP. Коды HMAC служат также для защиты опции ADD_ADDR от угроз, указанных в [RFC7430].

Использование битов возможности шифрования в начальном согласовании соединения для задания конкретного алгоритма позволяет в будущем развернуть дополнительные алгоритмы шифрования. Тем не менее, это согласование уязвимо для активных атак с понижением уровня защиты, если злоумышленник может менять биты возможностей шифрования в откликах от получателя для выбора более слабого алгоритма. Представленный в этом документе механизм обеспечивает защиту от всех лавинных атак и перехватов, рассмотренных в [RFC6181].

Предложенный в параграфе 3.1 вариант согласования версии при использовании разными версиями MPTCP общего формата позволяет расположенному на пути злоумышленнику возможность теоретической атаки для снижения версии. Поскольку протоколы v1 и v0 используют разное согласование, такая атака потребует от клиента повтороной организации соединения с версией v0 и поддержки этой версии сервером. Отметим, что атакующий в пути будет иметь доступ к необработанным данным, обходящие все другие механизмы защиты на уровне TCP. Как отмечено в Приложении E, этот документ задает удаление поля AddrID [RFC6824] из опции MP_PRIO (параграф 3.3.8), что исключает возможность теоретической атаки, когда злоумышленник может перевести субпоток в состояние резервного.

При нормальной работе обычные механизмы защиты TCP (такие как контроль попадания порядковых номеров в окно) обеспечивают такой же уровень защиты от атак на отдельные субпотоки, как для обычных соединений TCP. Реализации будут создавать дополнительные буферы (по сравнению с TCP) для сборки данных на уровне соединения. Иземенение размеров окна позволяет снизить риск DoS-атак, нацеленных на истощение ресурсов.

В параграфе 3.4.1 отмечено, что хост может анонсировать свои приватные (не маршрутизируемые) адреса, но они могут относится также к другим хостам сети получателя. Согласование (параграф 3.2) предотвращает организацию субпотоков с некорректными хостами, однако такие анонсы могут вызывать нежелательный трафик согласований TCP. Это свойство MPTCP может стать целью DoS-эксплойтов, когда злонамеренный участник соединения MPTCP побуждает получателя соединяться с другими хостами сети. Поэтому реализациям следует рассмотреть эвристические подходы (параграф 3.9) для снижения таких влияний на стороне клиента и сервера.

Для дополнительной защиты от вредоносных сообщений ADD_ADDR, передаваемых злоумышленником, не находящимся на пути передачи, ADD_ADDR включает код HMAC, использующий ключи, заданные при согласовании. Это предотвращает нарушение работы MPTCP за счет вставки в поток обманнных ADD_ADDR.

Небольшой риск теоретически может быть связан с повторным использованием ключей, но для реализации replay-атаки требуется соответствие ключей отправителя и получателя, а также случайных значений получателя в согласовании MP_JOIN (параграф 3.2).

Хотя эта спецификация определяет «промежуточное» решение по защите, отвечающее критериям, указанным в начале этого параграфа, и анализу угроз в [RFC6181], усложнение атак очевидно и будущей версии MPTCP наверняка потребуется более строгая защита. Имеется несколько способов улучшения защиты MPTCP, часть которых совместима с описанным в этом документе протоколом MPTCP, другие могут оказаться несовместимыми. На данный момент лучшим решением является обретение опыта работы с текущим подходом и проверка точности анализа угроз.

Возможные способы повышения уровня безопасности MPTCP перечислены ниже.

- Выбор нового криптографического алгоритма для MPTCP, согласуемого в MP_CAPABLE. При развертывании реализации в контролируемой среде, где возможны дополнительные допущения, такие как возможность серверов сохранять состояние в процессе согласования TCP, становится возможным применение более строгий криптографических алгоритмов.
- Задание способа защищенной передачи данных в MPTCP без изменения сигнальной части протокола.
- Определение защиты, требующей дополнительного пространства опций, возможно в сочетании с предложением «длинных опций» для расширения пространства опций TCP (например, [TCPLO]) или на основе текущего подхода со вторым этапом защиты на основе опций MPTCP.
- Пересмотр решения рабочей группы об использовании для сигнализации MPTCP лишь опций TCP и поиск возможности сигнализации также в данных TCP (payload).

При разработке MPTCP предусмотрено несколько методов идентификации новых механизмов защиты, включая:

- доступные флаги MP_CAPABLE (рисунок 4);
- доступные субтипы в опции MPTCP (рисунок 3);
- поле Version в MP_CAPABLE (рисунок 4).

6. Взаимодействие с промежуточными устройствами

TCP с множеством путей разработан для развертывания в реальном мире и предполагает «разумное» поведение имеющихся промежуточных устройств (ПУ). В этом разделе описано несколько типичных вариантов отказов, связанных с ПУ, и показана их обработка в Multipath TCP. Затем указаны решения Multipath TCP для адаптации к разным ПУ.

Основной проблемой является использование новой опции TCP. ПУ следует пересылать пакеты с неизвестными опциями без их изменения, но это делают не все. Предполагается наличие ПУ, вырезающих опции и передающих данные, копирующих одну опцию в разные сегменты (например, при сегментации) или отбрасывающих опции при слиянии сегментов. MPTCP использует одну новую опцию TCP (Kind) и все сообщения указываются значениями субтипа (раздел 7). Это снижает вероятность передачи лишь некоторых опций, используемых MPTCP, и ключевыми вопросами становятся различие путей и присутствие флага SYN.

Пакеты MPTCP SYN в первом субпотоке соединения включают опцию MP_CAPABLE (параграф 3.1). При ее отбрасывании MPTCP **следует** возвращаться к обычному TCP. Если отбрасываются пакеты с опцией MP_JOIN (параграф 3.2), путь просто не используется.

Если ПУ вырезает опции, не меняя остального, для MPTCP это безопасно. Если опция MP_CAPABLE отброшена на прямом или обратном пути, хост-инициатор просто вернется к TCP, как описано в параграфе 3.1 и показано на рисунке 17.

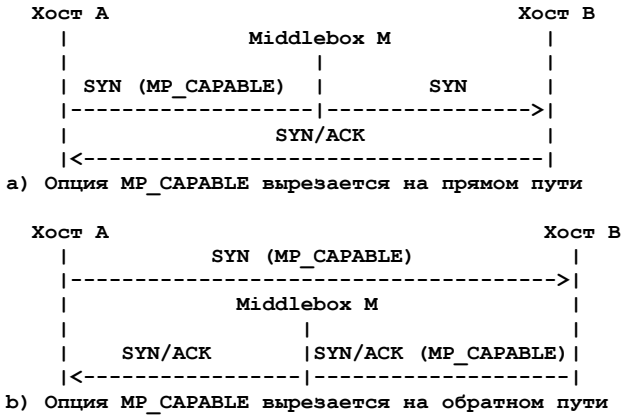


Рисунок 17. Соединение с вырезанием опции.

Пакеты SYN в субпотоках включают опцию MP_JOIN. Если эта опция вырезается на обратном пути, пакет с точки зрения хоста B становится обычным SYN. В зависимости от наличия на указанном порту прослушивающего сокета хост B будет отвечать пакетом SYN/ACK или RST (отказ в соединении для субпотока). Когда хоста A получить SYN/ACK, он ответит пакетом RST по причине отсутствия в SYN/ACK опции MP_JOIN и маркера. В любом случае организация субпотока завершается отказом и не влияет на соединение MPTCP в целом.

Рассмотрим поток данных MPTCP, предполагая его корректную организацию, означающую нормальное прохождение опций в пакетах SYN через ПУ. Если при этом ПУ не использует ресегментации и объединения сегментов TCP, потоки Multipath TCP будут работать без проблем.

Однако опции могут вырезаться из пакетов данных, как отмечено в параграфе 3.7. При частично вырезании опций MPTCP поведение протокола становится неопределенным. Если теряются некоторые Data Sequence Mapping, соединение будет работать, пока существует сопоставление с данными субпотоков (например, если передано несколько сопоставлений). Если часть пространства номеров субпотоков выпадает из сопоставления, субпотоки считаются поврежденными и закрываются с использованием описанного в параграфе 3.7 процесса. MPTCP охраняет работоспособность и при потере некоторых sData ACK, но производительность будет снижаться по мере роста числа вырезаемых опций. Возникновение таких ситуаций на практике не предполагается, поскольку большинство ПУ пропускают или удаляют все опции.

Завершим раздел описанием классов ПУ, их поведения и элементов MPTCP, позволяющих работать через такие ПУ. Проблемы, связанные с отбрасыванием пакетов с опциями и вырезанием опций, рассмотрены выше и здесь не обсуждаются.

NAT (Трансляторы сетевых адресов и портов)

Трансляторы [RFC3022] меняют адрес отправителя (зачастую и порт) в пакетах. Это означает, что хост не знает свой публичный адрес для сигнализации в MPTCP. Поэтому MPTCP разрешает неявное добавление адресов через опцию MP_JOIN и механизм согласования обеспечивает для попыток соединения с приватными адресами [RFC1918] (поскольку они аутентифицированы) организацию субпотоков лишь с нужными хостами. Явное удаление адресов выполняется по Address ID и позволяет не знать адрес отправителя.

Прокси-ускорители (Performance Enhancing Proxies или PEP)

PEP [RFC3135] могут заранее передавать пакеты ACK для повышения производительности. Однако MPTCP полагается на точность сигналов контроля перегрузки от конечного хоста, а не понимающие MPTCP устройства PEP не могут обеспечить таких сигналов. Поэтому MPTCP будет возвращаться к TCP или закрывать проблемный субпоток (параграф 3.7).

Нормализаторы трафика

Нормализаторы [norm] могут не допускать пропусков в порядковых номерах, а также кэшировать пакеты и повторно передавать данные. MPTCP в линии выглядит как обычный TCP и не передает разные данные с одним и тем же номером в субпотоке. В случае повторной передачи данные пойдут в том же субпотоке TCP, даже если они дополнительно повторены на уровне соединения с использованием другого субпотока.

Межсетевые экраны

МСЭ [RFC2979] могут использовать случайные начальные порядковые номера (Initial Sequence Number или ISN) для соединений TCP. В MPTCP используется относительная нумерация в Data Sequence Mapping для обхода этого. Подобно трансляторам NAT, МСЭ не разрешают множество входных соединений, поэтому MPTCP поддерживает передачу адресов (ADD_ADDR), позволяющую многоадресным хостам пригласить патнера, находящегося за МСЭ или NAT соединяться с ним через дополнительны интерфейс.

Системы детектирования и предотвращения вторжений (IDS/IPS)

Эти системы наблюдают в потоках пакетов определенные шаблоны и проверяют содержимое на предмет угроз из сети. Для MPTCP может потребоваться организация дополнительных путей и знающием о MPTCP устройствам IDS или IPS потребуется считывать маркеры MPTCP для сопоставления данных из множества субпотоков, чтобы получить полную картину трафика между устройствами. Без такого изменения IDS не будут видеть полной картины трафика и это увеличит риск прохождения вредоносных пакетов (ложный пропуск) и ошибочного отнесения пакетов к вредоносным (ложное срабатывание).

ПУ прикладного уровня

Такие устройства, как МСЭ с анализом содержимого, могут менять данные в субпотоках, например, переписывая URI в трафике HTTP. MPTCP будет замечать такие изменения при использовании контрольных сумм и закрывать соответствующие субпотоки, если другие субпотоки можно использовать. При влиянии на все субпотоки MPTCP будет возвращаться к TCP, позволяя ПУ менять содержимое пакетов. Осведомленные о MPTCP промежуточные устройства должны быть способны «подстраивать» содержимое и метаданные MPTCP, чтобы соединения не разрывались.

Кроме того, все классы ПУ могут влиять на трафик TCP, как указано ниже.

- Многие классы МУ могут удалять опции TCP или отбрасывать пакеты с неизвестными опциями. Предполагается, что начального обмена SYN с опциями TCP достаточно для определения свойств пути. Если такие пакеты не проходят, MPTCP будет возвращаться к обычному TCP.
- При разбиении и объединении сегментов (например, для выгрузки сегментации TCP) опции могут копироваться между пакетами или вырезаться. Data Sequence Mapping в MPTCP включает относительные порядковые номера субпотоков вместо использования порядковых номеров в сегменте. Это обеспечивает независимость от нумерации в пакетах.
- Некоторые ПУ могут сокращать окно приема на уровне субпотока. MPTCP использует на уровне данные максимальное из окон субпотоков, сохраняя окна самих субпотоков.

7. Взаимодействие с IANA

Этот документ отменяет [RFC6824], поэтому обновлено несколько реестров IANA, ссылающихся на отмененный документ. Эти вопросы рассмотрены в последующих параграфах.

7.1. Номера опций TCP

Агентство IANA обновило реестр TCP Option Kind Numbers, указав этот документ для Multipath TCP (таблица 1).

Таблица 1. Номер опции TCP.

Тип	***Размер***	***Назначение***	***Документ***
30	N	Multipath TCP (MPTCP)	RFC 8684

7.2. Субтипы опции MPTCP

Субреестр 4-битовых субтипов опций MPTCP Option Subtypes в реестре Transmission Control Protocol (TCP) Parameters, был определен в [RFC6824]. Поскольку [RFC6824] относится к категории Experimental RFC, а не Standards Track RFC и в субреестр не было добавлено других записей, указывающих на [RFC6824], агентство IANA заменило имеющийся реестр содержимым таблицы 2 с представленным ниже примечанием.

> Этот реестр задает субтипы опций MPTCP для MPTCP v1, отменяющего экспериментальный протокол MPTCP v0. Субтипы MPTCP v0 указаны в [RFC6824].

Таблица 2. Субтипы опции MPTCP.

Значение	***Символ***	***Назначение***	***Документ***
0x0	MP_CAPABLE	Поддержка множества путей	RFC 8684, параграф 3.1
0x1	MP_JOIN	Добавление в соединение	RFC 8684, параграф 3.2
0x2	DSS	Data Sequence Signal (Data ACK и Data Sequence Mapping)	RFC 8684, параграф 3.3
0x3	ADD_ADDR	Добавление адреса	RFC 8684, параграф 3.4.1

0x4	REMOVE_ADDR	Удаление адреса	RFC 8684, параграф 3.4.2
0x5	MP_PRIO	Смена приоритета субпотока	RFC 8684, параграф 3.3.8
0x6	MP_FAIL	Возврат к TCP	RFC 8684, параграф 3.7
0x7	MP_FASTCLOSE	Ускоренное закрытие	RFC 8684, параграф 3.5
0x8	MP_TCPRST	Сброс субпотока	RFC 8684, параграф 3.6
0xf	MP_EXPERIMENTAL	Резерв для частного применения	

Значения 0x9 - 0xe в настоящее время не выделены, а 0xf зарезервировано для частных экспериментов (ее использование может быть задано в будущих спефикациях). Выделение значений в этом реестре происходит по процедуре Standards Action, определенной в [RFC8126]. Назначение включает символьное имя субтипа MPTCP, значение и ссылку на спецификацию.

7.3. Алгоритмы согласования MPTCP

Субреестр MPTCP Handshake Algorithms в реестре Transmission Control Protocol (TCP) Parameters, был определен в [RFC6824]. Поскольку [RFC6824] относится к категории Experimental RFC, а не Standards Track RFC и в субреестр не было добавлено других записей, указывающих на [RFC6824], агентство IANA заменило имеющийся реестр содержимым таблицы 3 с представленным ниже примечанием.

> Этот реестр задает субтипы опций MPTCP для MPTCP v1, отменяющего экспериментальный протокол MPTCP v0. Субтипы MPTCP v0 указаны в [RFC6824].

Таблица 3. Алгоритмы согласования MPTCP.

Флаг	***Назначение***	***Документ***
A	Требуется контрольная сумма	RFC 8684, параграф 3.1
B	Расширяемость	RFC 8684, параграф 3.1
C	Не пытаться организовать новые субпотоки с этого адреса отправителя	RFC 8684, параграф 3.1
D-G	Не выделены	
H	HMAC-SHA256	RFC 8684, параграф 3.2

Отметим, что значения битов D - H могут зависеть от бита B с учетом определения параметра Extensibility в будущих спецификациях (см. параграф 3.1).

Выделение значений в этом реестре происходит по процедуое Standards Action, заданной в [RFC8126]. Назначение включает значение флага, символьное имя алгоритма и ссылку на спецификацию.

7.4. Коды причин MP_TCPRST

Субреестр MPTCP MP_TCPRST Reason Codes создан IANA в реестре Transmission Control Protocol (TCP) Parameters для кодов причины в сообщениях MP_TCPRST (параграф 3.6). Начальные значения указаны в таблице 4, а выделение новых значений будет происходить по процедуре Specification Required, заданной в [RFC8126]. Назначение включает значение кода, краткое описание и ссылку на спецификацию. Максимальное значение кода - 0xff.

Таблица 4. Коды причин MP_TCPRST.

Код	***Значение***	***Документ***
0x00	Неуказанная ошибка	RFC 8684, параграф 3.6
0x01	Ошибка MPTCP	RFC 8684, параграф 3.6
0x02	Нехватка ресурсов	RFC 8684, параграф 3.6
0x03	Административный запрет	RFC 8684, параграф 3.6
0x04	Слишком много ожидающих данных	RFC 8684, параграф 3.6
0x05	Неприемлемая производительность	RFC 8684, параграф 3.6
0x06	Влияние промежуточных усттройств	RFC 8684, параграф 3.6

В качестве рекомендации для назначенных экспертов [RFC8126] отметим, что назначения не следует отвергать, если нет дефицита в пространстве номеров, если новое назначение явно отличается от имеющихся и сопровождается рекомендациями для разработчиков по части отправки и обработки этого кода.

8. Литература

8.1. Нормативные документы

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

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, DOI 10.17487/RFC5961, August 2010, <https://www.rfc-editor.org/info/rfc5961>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

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

8.2. Дополнительная литература

[deployments] Bonaventure, O. and S. Seo, "Multipath TCP Deployments", IETF Journal 2016, November 2016, <https://www.ietfjournal.org/multipath-tcp-deployments/>.

[howhard] Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and M. Handley, "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP", Usenix Symposium on Networked Systems Design and Implementation 2012, April 2012, <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/raiciu>.

[norm] Handley, M., Paxson, V., and C. Kreibich, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics", Usenix Security Symposium 2001, August 2001, <https://www.usenix.org/legacy/events/sec01/full_papers/handley/handley.pdf>.

[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>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, DOI 10.17487/RFC2018, October 1996, <https://www.rfc-editor.org/info/rfc2018>.

[RFC2979] Freed, N., "Behavior of and Requirements for Internet Firewalls", RFC 2979, DOI 10.17487/RFC2979, October 2000, <https://www.rfc-editor.org/info/rfc2979>.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, DOI 10.17487/RFC2992, November 2000, <https://www.rfc-editor.org/info/rfc2992>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <https://www.rfc-editor.org/info/rfc3022>.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <https://www.rfc-editor.org/info/rfc3135>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[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>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC6181] Bagnulo, M., "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6181, DOI 10.17487/RFC6181, March 2011, <https://www.rfc-editor.org/info/rfc6181>.

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

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <https://www.rfc-editor.org/info/rfc6356>.

[RFC6528] Gont, F. and S. Bellovin, "Defending against Sequence Number Attacks", RFC 6528, DOI 10.17487/RFC6528, February 2012, <https://www.rfc-editor.org/info/rfc6528>.

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

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

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

[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>.

[RFC7430] Bagnulo, M., Paasch, C., Gont, F., Bonaventure, O., and C. Raiciu, "Analysis of Residual Threats and Possible Fixes for Multipath TCP (MPTCP)", RFC 7430, DOI 10.17487/RFC7430, July 2015, <https://www.rfc-editor.org/info/rfc7430>.

[RFC8041] Bonaventure, O., Paasch, C., and G. Detal, "Use Cases and Operational Experience with Multipath TCP", RFC 8041, DOI 10.17487/RFC8041, January 2017, <https://www.rfc-editor.org/info/rfc8041>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[TCPLO] Ramaiah, A., "TCP option space extension", Work in Progress, Internet-Draft, draft-ananth-tcpm-tcpoptext-00, 26 March 2012, <https://tools.ietf.org/html/draft-ananth-tcpm-tcpoptext-00>.

Приложение A. Использование опций TCP

Пространство опций TCP ограничено по причине малого размера поля Data Offset в заголовке TCP (4 бита), задающего размер заголовка TCP в 32-битовых. Стандартный заголовок TCP занимает 20 байтов и для опций остается лишь 40, из которых многие уже заняты такими опциями, как временные метки и SACK.

Было рассмотрено общепринятое использование опций TCP в пакетах SYN, данных и «чистых» подтверждениях ACK и обнаружено достаточно места для включения описанных в документе опций.

Пакеты SYN обычно включают опции MSS[1] (4 байта), масштаб окна (3 байта), возможность использовать SACK (2 байта) и временную метку (10 байтов), что в сумме дает 19 байтов. Некоторые операционные системы дополняют каждую опцию до границы слова, что может увеличить размер до 24 байтов (известно, что Windows XP и Mac OS X используют лополнение, Linux - нет). В результате оптимистичная оценка дает 21 свободный байт или 16 при использовании дополнения. В любом случае для пакетов SYN опция MP_CAPABLE (12 байтов) или MP_JOIN (12 или 16 байтов) помещается в заголовок.

Отметим, что, благодаря применению 64-битовых номеров на уровне данных, MPTCP не требует опции timestamp для защиты от переполнения (wrap) порядковых номеров (с помощью механизма PAWS[2] [RFC7323]), поскольку достижение максимального номера значительно менее вероятно. Этот вопрос оставлен для дальнейшего изучения.

Пакеты TCP обычно содержат временную метку, занимающую 10 байтов (12 с заполнением). Это оставляет для других опций 30 байтов (28 при заполнении). Опция DSS меняет размер в зависимости от (1) включения Data Sequence Mapping, DATA_ACK или обоих, (2) передачи 4- или 8-октетных номеров и (3) наличия контрольной суммы. Максимальный размер опции DSS составляет 28 байтов, что позволяет ее включать в оставшееся пространство. Однако, если соединение не является сразу двухсторонним и широкополосным, такой размер опции в каждом пакете маловероятен.

Для опции DSS не требуется включать всякий раз Data Sequence Mapping и DATA_ACK, а во многих случаях можно чередовать их (пока отображение включает данные для передачи в следующем пакете). Можно также чередовать 4- и 8-байтовые номера в каждой опции.

При организации субпотока и соединения опций MPTCP передается также в третьем пакете (ACK). MP_CAPABLE (20 байтоы) или MP_JOIN (24 байта) помещается в оставшееся место заголовка.

«Чистые» ACK в TCP обычно включают лишь временную метку (10 байтов). Протоколу Multipath TCP здесь обычно требуется указать лишь DATA_ACK (не более 12 байтов). Иногда пакеты ACK включают информацию SACK. В зависимости от числа потерянных пакетов SACK может занимать все пространство опций. Если нужно включить DATA_ACK, вероятно потребуется снизить число блоков SACK. Однако присутствие DATA_ACK вряд ли потребуется при использовании SACK, поскольку до повторной передачи хотя бы некоторых блоков SACK кумулятивное подтверждение ACK на уровне соединения не перемещается вперед (или делает это при повторах по другому пути, который можно использовать для передачи DATA_ACK).

Опция ADD_ADDR может иметь размер от 16 до 30 байтов в зависимости от (1) используемого протокола (IPv4 или IPv6) и (2) присутствия номера порта. Маловероятно, что такая сигнализация поместится в пакет данных, хотя при наличии места ее могжно включить. Рекомендуется не применять дубликаты ACK с другими данными или опциями для передачи этих сигналов. Отметим, что это послужило причиной того, что дубликаты ACK не считаются в MPTCP признаком перегрузки.

Приложение B. TCP Fast Open и MPTCP

Экспериментальное расширение TCP Fast Open (TFO) [RFC7413] разработано для обеспечения возможности передачи данных на один интервал RTT раньше, чем в обычном TCP. Это считается важным преимуществом, поскольку широко распространены очень короткие соединения, особенно для запросов и откликов HTTP. Механизм реализуется путем отправки сегмента SYN с данными приложения и обеспечения принимающей стороне возможности передать ответные данные вслед за SYN/ACK. [RFC7413] защищает этот механизм за счет использования новой опции TCP, включающей значение cookie, согласованное в предыдущем соединении.

При использовании TFO с протоколом iMPTCP следует учитывать два важных момента, рассмотренных ниже.

B.1. TFO Cookie в MPTCP

При первом подключении инициатора TFO к прослушивающей стороне он не может сразу включить данные в SYN по соображениям безопасности [RFC7413]. Вместо этого он запрашивает cookie для использования в последующих соединениях. Это делается с помощью опции запроса и возврата TCP, размером 2 и 6-18 байтов (в зависимости от размера cookie).

TFO и MPTCP можно объединить, если общий размер всех опций в заголовке TCP не превысит 40 байтов.

- В пакетах SYN протокол MPTCP использует 4-байтовую опцию MP_CAPABLE. Суммарный размер опций MPTCP и TFO составляет 6 байтов. При типичном для TCP наборе опций SYN в 19 байтов (24 при выравнивании по границе слова) остается достаточно места для опции MP_CAPABLE и запроса TFO cookie.
- В пакетах SYN + ACK протокол MPTCP использует 12-байтовую опцию MP_CAPABLE, но опция TFO может достигать 18 байтов. Поскольку размер поля опций явно превышается, принимающей стороне для решения проблемы остается лишь сокращение размера cookie. Например, при использовании 19 байтов для классических опций TCP максимальный размер cookie составит 7 байтов. Отметим, что для пакета SYN такое же ограничение применимо к последующим соединениям (поскольку инициатор возвращает значение cookie). Если снижение размера cookie не приемлемо с точки зрения безопасности, принимающая сторона может сократить другие опции TCP, опустив временные метки TCP (как указано в Приложении A).

B.2. Отображение порядковых номеров при TFO

В фазе организации соединения TCP протокол MPTCP использует обмен ключами, применяемыми для генерации начальных порядковых номеров IDSN. В частности, SYN с опцией MP_CAPABLE занимает первый октет пространства номером. При использовании TFO одним из способов передачи данных в SYN является неявное отображение DSS, охватывающее сегмент SYN (поскольку в пакете SYN нет места для включения опции DSS). Проблема этого подхода заключается в том, что при изменении промежуточным устройством данных TFO этого не увидит MPTCP по причине отсутствия контрольной суммы DSS. Например, знающее TCP (но не MPTCP) промежуточное устройство может вставить байты в начало потока, скорректировав контрольную сумму TCP и порядковые номера. При неявном

[1]Maximum Segment Size - максимальный размер сегмента.
[2]Protection Against Wrapped Sequences - защита для случаев перехода порядкового номера через максимум.

отображении эта информация будет давать инициатору и принимающей стороне разное отображение DSS и это невозможно увидеть по причине отсутствия контрольной суммы DSS.

Для решения проблемы данные TFO не должны считаться частью пространства порядковых номеров соединения. SYN с опцией MP_CAPABLE по-прежнему занимает первый октет пространства, но затем второй октет занимает первый байт данных, не относящихся к TFO. Это гарантирует, что при согласованном использовании контрольной суммы DSS все данные в пространстве номеров соединения будут учтены в контрольной сумме. Отметим также, что это не влечет потери функциональности, поскольку данные TFO передаются лишь в первом субпотоке до попыток создания новых субпотоков в соединении.

B.3. Примеры организации соединений

Ниже приведено несколько примеров организации соединения при использовании TFO с протоколом MPTCP.

Перед отправкой инициатором данных вместе с SYN он должен запросить у принимающей стороны значение cookie, как показано на рисунке 18. Отметим, что на этом и последующих рисунках номер и размер указаны в форме Seq(Length), например, S. 0(0). Это делается путем обычного использования опций TFO и MPTCP.

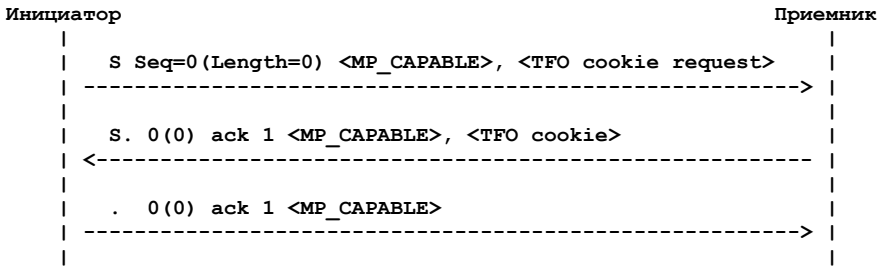


Рисунок 18. Запрос cookie.

После этого полученное значение cookie можно применять в TFO, как показано на рисунке 19. В этом примере инициатор сначала передает 20 байтов в SYN. Принимающая сторона сразу же отвечает 100 байтами, следующими за SYN-ACK, на которые инициатор отвечает еще 20 байтами. Отметим, что последний сегмент на рисунке имеет номер TCP 21, тогда как DSS субпотока имеет номер 1 (поскольку данные TFO не являются частью пространства порядковых номеров данных, как отмечено в Приложении B.2).

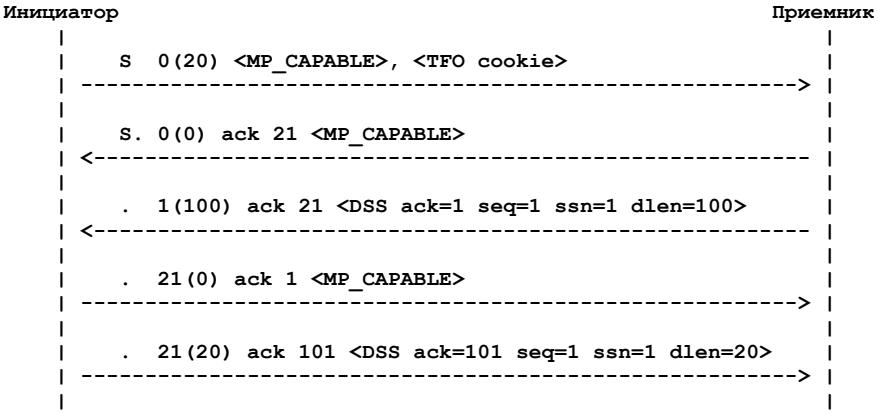


Рисунок 19. Приемник с поддержкой TFO.

На рисунке 20 принимающая сторона не поддерживает TFO. Инициатор видит отсутствие состояния на приемной стороне (нет ACK для данных) и передает MP_CAPABLE в третьем пакете, чтобы приемная сторона могла создать контекстс MPTCP в конце процесс согласования. Сейчас при повторе данных TFO они становятся частью Data Sequence Mapping, поскольку фактически передаются после организации соединения.

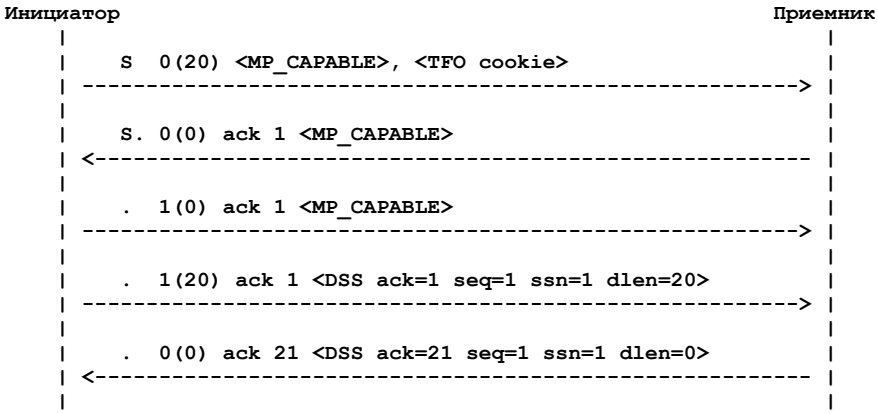


Рисунок 20. Приемник без поддержки TFO.

Возможно также подтверждение приемной стороной лишь части данных TFO, как показано на рисунке 21. Инициатор просто повторит недостающие данные вместе с отображением DSS.

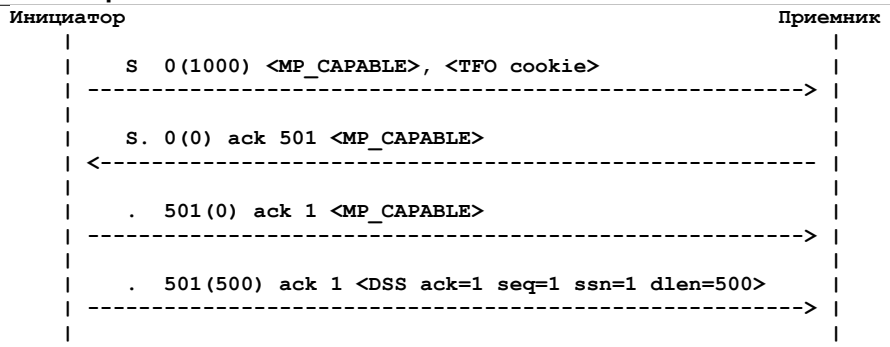


Рисунок 21. Частичное подтверждение данных.

Приложение C. Блоки управления

Концептуально соединение MPTCP можно представить как блок управления протоколом MPTCP (protocol control block или PCB), содержащий несколько переменных, которые отражают процесс организации и состояние соединения MPTCP и набор связанных с этим блоков управления TCP для созданных в соединении субпотоков.

В RFC 793 [RFC0793] задано несколько переменных состояния и здесь по возможности применяется терминология RFC 793 для описания переменных состояния MPTCP.

C.1. Блок управления MPTCP

Блок управления MPTCP содержит описанные ниже переменные состояния на уровне соединения в целом.

C.1.1. Аутентификация и метаданные

Local.Token (32 бита)

Маркер, выбранный локальным хостом для данного соединения MPTCP. Маркер должен быть уникальным для каждого организованного хостом соединения MPTCP и генерируется из локального ключа.

Local.Key (64 бита)

Ключ, переданный локальным хостом для этого соединения MPTCP.

Remote.Token (32 бита)

Маркер, выбранный удаленным хостом для данного соединения MPTCP (генерируется из удаленного ключа).

Remote.Key (64 бита)

Ключ, выбранный локальным хостом для этого соединения MPTCP.

MPTCP.Checksum (flag)

Флаг, устанавливаемый (true) при установке любой из сторон бита A в опции MP_CAPABLE, переданной при организации соединения. В противном случае флаг сброшен (false). Установка этого флага требует расчета контрольной суммы во всех опциях DSS.

C.1.2. Передающая сторона

SND.UNA (64 бита)

Порядковый номер следующего байта для подтверждения на уровне соединения MPTCP. Переменная обновляется при получении опции DSS с DATA_ACK.

SND.NXT (64 бита)

Порядковый номер передаваемого следующим байта. Переменная служит значением DSN в опции DSS.

SND.WND (32 бита)

Окно передачи. 32 бита при использовании функций RFC 7323 и 16 - в остальных случаях. MPTCP поддерживает окно передачи на уровне MPTCP, которое является общим для всех субпотоков. Субпотоки применяют SND.WND для расчета значения SEQ.WND в каждом передаваемом сегменте.

C.1.3. Приемная сторона

RCV.NXT (64 бита)

Порядковый номер байта, ожидаемого в соединении MPTCP. Переменная изменяется при получении данных с нарушением порядка. Значение RCV.NXT применяется для значений DATA_ACK, передаваемых в общии DSS всех субпотоков.

RCV.WND (32 бита)

Окно приема на уровне соединения, которое принимает максимальное из значений RCV.WND среди всех субпотоков. 32 бита при использовании функций RFC 7323 и 16 - в остальных случаях.

C.2. Блоки управления TCP

Блок управления MPTCP содержит также список блоков управления TCP, связанных с соединением MPTCP.

Отметим, что блоки управления TCP всех субпотоков не включают переменных RCV.WND и SND.WND, поскольку они поддерживаются на уровне соединения, а не субпотока.

В каждом из блоков управления TCP поддерживаются указанные ниже переменные.

C.2.1. Передающая сторона

SND.UNA (32 бита)

Порядковый номер следующего байта, который будет подтверждаться в субпотоке. Переменная обновляется при получении в субпотоке каждого подтверждения TCP.

SND.NXT (32 бита)

Порядковый номер следующего байта, который будет передан в субпотоке. SND.NXT служит для установки SEG.SEQ при передаче следующего сегмента.

C.2.2. Приемная сторона

RCV.NXT (32 бита)

Порядковый номер следующего байта, который ожидается в субпотоке. Переменная обновляется при получении сегментов с нарушением порядка. Значение RCV.NXT копируется в поле SEG.ACK следующего сегмента, передаваемого в субпоток.

RCV.WND (32 бита)

Окно приема на уровне субпотока, обновляемое значением поля окна из сегментов, принятых в субпотоке. 32 бита при использовании функций RFC 7323 и 16 - в остальных случаях.

Приложение D. Конечный автомат состояний

На рисунке 22 показан конечный автомат (Finite State Machine или FSM) для закрытия соединения. Рисунок показывает взаимодействие сигнала DATA_FIN (указан на рисунке флагом DFIN в DATA_ACK) (1) с FIN на уровне потоков и (2) переключение break-before-make (прервать до организации) между субпотоками.

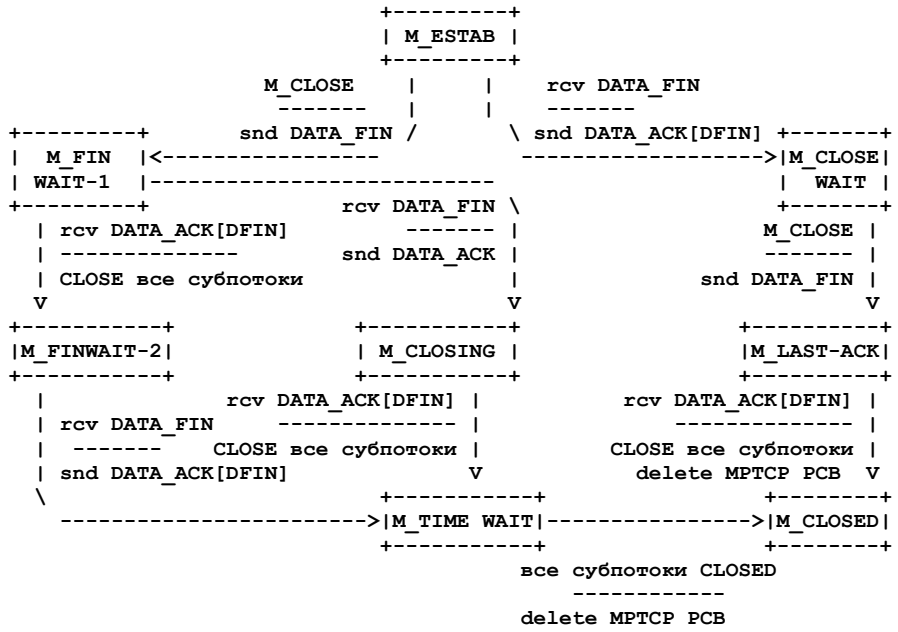


Рисунок 22. Конечный автомат для закрытия соединения.

Приложение E. Отличия от RFC 6824

В этом приложении перечислены основные технические различия между [RFC6824], где задан протокол MPTCP v0 и данным документом, который отменяет [RFC6824] и задает MPTCP v1. Отметим, что новая спецификация не совместима с [RFC6824].

- Документ включает опыт реализация, развертывания и экспериментов, собранный в документе Use Cases and Operational Experience with Multipath TCP [RFC8041] и статье в IETF Journal Multipath TCP Deployments [deployments].
- Инициирование соединения с обменом опцией MPTCP MP_CAPABLE отличается от [RFC6824]. SYN больше не содержит ключ инициатора для сокращения размера опции MP_CAPABLE в пакете SYN и предотвращения дублирования передачи ключевого материала.
- Это также обеспечивает надежную доставку ключа для MP_CAPABLE за счет разрешения его передачи вместе с данными и соответствующего использования встроенного механизма гарантированной доставки TCP. Если у инициатора еще нет данных для передачи, опция MP_CAPABLE с ключами будет повторена в первом пакете данных. Если другая сторона начнет передачу первой, наличие опции DSS неявно подтвердит получения MP_CAPABLE.
- В поле Flags опции MP_CAPABLE выделен бит C, указывающий, что отправитель опции не будет воспринимать дополнительные субпотоки MPTCP по адресу и порту. Это повышает эффективность, например, при размещении отправителя за NAT.
- В поле Flags опции MP_CAPABLE бит H указывает использование алгоритма HMAC-SHA256 (вместо HMAC-SHA1).
- Инициирование соединения также определяет процедуру согласования версий для реализаций, поддерживающих v0 [RFC6824] и v1 (данный документ).
- Применяется алгоритм HMAC-SHA256 (взамен HMAC-SHA1), обеспечивающий лучшую защиту. Алгоритм служит для генерации маркеров в MP_JOIN и ADD_ADDR, а также для установки IDSN.
- Имеется новая опция на уровне субпотока для указания причины отправки RST в субпотоке MP_TCPRST (параграф 3.6), которая может помочь реализации при решении вопроса о повторении попыток соединения.
- Опция MP_PRIO (параграф 3.3.8), служащая для изменения приоритета субпотока, больше не включает поле AddrID, которое позволяло изменить приоритет для другого субпотока. Однако было выяснено, что опция может применяться для MITM-атак[1] для перенаправления трафика на свой путь, а опция MP_PRIO не включает маркера или иного механизма защиты.

[1]Man-in-the-middle - перехват и изменение данных в пути с участием человека.

- Опция ADD_ADDR (параграф 3.4.1), служащая для информирования партнера о другом возможном адресе, отличается в нескольких аспектах. Сейчас она включает HMAC для добавленного адреса, что улучшает защиту. Кроме того, добавлена надежная доставка опции ADD_ADDR - поле IPVer заменено полем флагов, один из которых (E) используется как «эхо» и хост может сообщить о получении опции.
- Этот документ описывает дополнительный вариант процедуры Fast Close за счет передачи опции MP_FASTCLOSE в RST через все субпотоки. Это позволяет хосту сразу разорвать субпотоки и соединение.
- Агентство IANA зарезервировало субтип опции MPTCP со значением 0xf для частных применений (Private Use, параграф 7.2). Этот документ не задает способа использования данного значения.
- Добавлено Приложение B с обсуждением одновременного использования опций MPTCP и TFO в одном пакете.

Благодарности

Авторы признательные Sebastien Barre и Andrew McDonald за существенный вклад в этот документ.

Спасибо Iljitsch van Beijnum, Lars Eggert, Marcelo Bagnulo, Robert Hancock, Pasi Sarolahti, Toby Moncaster, Philip Eardley, Sergio Lembo, Lawrence Conroy, Yoshifumi Nishida, Bob Briscoe, Stein Gjessing, Andrew McGregor, Georg Hampel, Anumita Biswas, Wes Eddy, Alexey Melnikov, Francis Dupont, Adrian Farrel, Barry Leiba, Robert Sparks, Sean Turner, Stephen Farrell, Martin Stiemerling, Gregory Detal, Fabien Duchene, Xavier de Foy, Rahul Jadhav, Klemens Schragel, Mirja Kühlewind, Sheng Jiang, Alissa Cooper, Ines Robles, Roman Danyliw, Adam Roach, Eric Vyncke, Ben Kaduk за рецензии и вклад в работу.

Адреса авторов

Alan Ford
Pexip
Email: alan.ford@gmail.com

Costin Raiciu
University Politehnica of Bucharest
Splaiul Independentei 313
Bucharest
Romania
Email: costin.raiciu@cs.pub.ro

Mark Handley
University College London
Gower Street
London
WC1E 6BT
United Kingdom
Email: m.handley@cs.ucl.ac.uk

Olivier Bonaventure
Université catholique de Louvain
Pl. Ste Barbe, 2
1348 Louvain-la-Neuve
Belgium
Email: olivier.bonaventure@uclouvain.be

Christoph Paasch
Apple, Inc.
Cupertino, CA
United States of America
Email: cpaasch@apple.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru