Network Working Group
Request for Comments: 4340
Category: Standards Track

E. Kohler
UCLA
M. Handley
UCL
S. Floyd
ICIR

March 2006

Протокол DCCP

Datagram Congestion Control Protocol (DCCP)

Статус документа

В этом документе содержится спецификация протокола, предложенного сообществу Internet. Документ служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола вы можете узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться без ограничений.

Авторские права

Аннотация

DCCP[1] представляет собой транспортный протокол, который обеспечивает двухсторонние индивидуальные (unicast) соединения для передачи дейтаграмм с контролем насыщения, но без гарантий доставки. Протокол DCCP подходит для приложений, которые передают достаточно большие объёмы данных и могут получить преимущества в результате контроля выбора между своевременностью и надёжностью.

Оглавление

[1]Datagram Congestion Control Protocol - протокол контроля насыщения при передаче дейтаграмм.

Список таблиц

1. Введение

DCCP представляет собой транспортный протокол, который обеспечивает двухсторонние индивидуальные соединения для передачи дейтаграмм с контролем насыщения, но без гарантий доставки. В частности, DCCP обеспечивает следующие возможности:

- поддержка потоков дейтаграмм без гарантий доставки;
- гарантированное согласование параметров при организации и разрыве соединений;
- гарантированное согласование опций, включая выбор подходящего механизма контроля насыщения;
- механизмы, позволяющие серверам предотвратить сохранение состояния для неподтвержденных попыток организации соединений и завершённых соединений;
- контроль насыщения, включающий механизмы ECN[1] [RFC3168] и ECN Nonce [RFC3540];
- механизм подтверждений, обеспечивающий сигналы о потере пакетов и информацию ECN; подтверждения передаются с таким уровнем надёжности, который требуется механизмом контроля насыщения (включая полную гарантию доставки);
- дополнительные механизмы, обеспечивающие (с высоким уровнем надёжности) передающее приложение информацией о доставке пакетов получателю, установке маркеров ECN, повреждении или отбрасывании пакетов;
- механизм определения PMTU[2] [RFC1191].
- выбор модулей механизма контроля насыщения; в настоящее время поддерживаются два механизма - контроль насыщения в стиле TCP[3] [RFC4341] и TFRC[4] [RFC4342]; протокол DCCP легко расширяется с точки зрения поддержки новых механизмов контроля насыщения для unicast-пакетов.

Протокол DCCP предназначен для потоковых приложений и других задач, которые могут получить преимущества за счёт возможности выбора между величиной задержки и надёжной доставкой с соблюдением порядка. Протокол TCP не подходит для таких приложений, поскольку используемые в нем механизмы контроля насыщения и обеспечения гарантий упорядоченной доставки могут приводить к произвольно долгим задержкам. Протокол UDP избавляется от длительных задержек, но приложения UDP, поддерживающие контроль насыщения, делают это на свой страх и риск. Протокол DCCP поддерживает средства контроля насыщения, включая ECN, для потоков дейтаграмм без гарантий доставки, но и без произвольно долгих задержек, присущих TCP. Протокол также поддерживает организацию, и согласование параметров, а также разрыв соединений с гарантированной доставкой.

2. Обоснование задач

Одной из задач при разработке протокола DCCP было не оставлять потоковым приложениям UDP причин для отказа от перехода на DCCP после того, как протокол будет развернут. Для решения этой задачи при создании DCCP минимизировались издержки, связанные как с заголовками пакетов, так и с дополнительной нагрузкой на процессоры конечных хостов. В протоколе DCCP реализован лишь минимальный набор функций, не включающий упреждающей

[1]Explicit Congestion Notification – явное уведомление о насыщении.
[2]Path Maximum Transmission Unit – максимальный размер передаваемого блока.
[3]TCP-like Congestion Control.
[4]TCP-Friendly Rate Control.

коррекции ошибок (FEC[1]), частичных гарантий доставки[2] и поддержки множества потоков - такие функции при необходимости могут быть реализованы «поверх» DCCP.

Для различных приложений подходят разные формы контроля насыщения. Например, для сетевых игр может оказаться желательным быстрое использование доступной полосы сети, тогда как для потоковых приложений более важной может оказаться стабильность скорости передачи данных (стремительное изменение скорости может вызывать нежелательные сбои на уровне пользовательского интерфейса типа пауз в воспроизведении звука или щелчков при прослушивании). Протокол DCCP позволяет приложениям выбирать подходящий механизм контроля насыщения из числа поддерживаемых. Один из вариантов - TCP-like Congestion Control - снижает вдвое размер окна насыщения в ответ на отбрасывание или маркировку пакетов, как это делается в TCP. Использующие этот механизм приложения будут быстро реагировать на изменение доступной полосы, но должны быть устойчивы к внезапному изменению размера окна насыщения, характерному для TCP. Второй вариант - TCP-Friendly Rate Control (TFRC) [RFC3448] – представляет собой основанный на выравнивании механизм контроля насыщения, который минимизирует внезапные изменения скорости передачи. Другие варианты могут добавляться в протокол по мере стандартизации новых механизмов контроля насыщения.

DCCP также позволяет для трафика с негарантированной доставкой безопасно использовать ECN. Интерфейс UDP API[3] в ядре может не позволять приложениям устанавливать для пакетов UDP флаг поддержки ECN, поскольку данный API не может гарантировать, что приложения будут корректно детектировать насыщение и реагировать на него подобающим образом. Интерфейсы DCCP API не вызывают таких проблем, поскольку сам протокол DCCP поддерживает контроль насыщения.

Протокол не требует использовать механизм CM[4] [RFC3124], позволяющий поддерживать между отправителем и получателем множество потоков с единым контролем насыщения. Существующий механизм CM могут использовать только приложения, имеющие собственную сквозную систему оповещения о потере пакетов, чего нельзя сказать о большинстве приложений, использующих протокол UDP. Кроме того, в CM непросто организовать поддержку множества механизмов контроля насыщения, а также механизмов, поддерживающих информацию о потере и маркировке пакетов на стороне получателя, а не отправителя. Протоколу DCCP следует обеспечивать возможность использования CM в тех случаях, когда это желательно для приложения, но мы не видим никаких преимуществ от развёртывания DCCP на базе CM.

Для протокола DCCP выбраны описанные в этом документе механизмы, которые подойдут для всех приложений, нуждающихся в контроле насыщения для потоков индивидуальных (unicast) дейтаграмм без обеспечения гарантий доставки. Однако механизмы контроля насыщения, принятые в настоящее время для DCCP и описанные о отдельных документах (Congestion Control ID Profile[5]) [RFC4341, RFC4342], могут приводить к возникновению проблем для некоторых приложений, включая широкополосные интерактивные видеосистемы. Такие приложения смогут использовать DCCP после стандартизации подходящего профиля контроля насыщения.

3. Уровни требований

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с [RFC2119].

3.1. Числа и поля

Все многобайтовые значения в DCCP (такие, как номера портов, порядковые номера и аргументы опций) передаются с использованием сетевого порядка следования байтов[6] (сначала передаётся старший байт).

В документе используются термины «левый» и «правый» применительно к битовым полям. Слева располагаются более значимые биты, а справа – менее значимые.

Случайные числа используются в DCCP для обеспечения безопасности и выбирать их следует с учётом рекомендаций [RFC4086].

Все операции с порядковыми номерами в DCCP используют циклическую арифметику по модулю 2^{48} при сравнении числовых значений. Такая арифметика позволяет сохранить отношения упорядоченности номеров в случаях перехода от 2^{48}-1 к нулевому значению. Стратегия реализации порядковых номеров в DCCP похожа на другие варианты использования циклической арифметики, включая порядковые номера TCP [RFC793] и DNS [RFC1982]. Имеет смысл хранить порядковые номера DCCP в старших 48 битах 64-битовых целых чисел, заполняя младшие 16 битов нулями. Такой подход согласуется с общепринятым методом сравнения для циклической арифметики, когда проверка выполнения условия A < B выполняется путём проверки выполнения неравенства (A - B) < 0 с использованием арифметики дополнения до двух. Резервные поля заголовков DCCP **должны** устанавливаться в 0 на стороне отправителя и игнорироваться получателями, если явно не задано иное. Это обеспечит совместимость с будущими расширениями протокола. В частности, для процессоров DCCP **недопустимо** сбрасывать соединения DCCP на основании лишь того, что резервное поле имеет отличное от нуля значение [RFC3360].

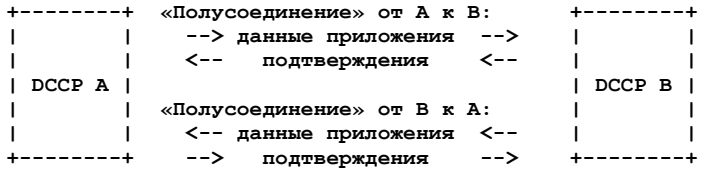
3.2. Участники соединения

Каждое соединение DCCP организуется между двумя хостами, которые мы будем обычно называть DCCP A и DCCP B. Каждое соединение инициируется одним из хостов, мы будем называть этот хост клиентом. Другая сторона на начальном этапе находится в пассивном состоянии и называется сервером. Термин «конечная точка DCCP» будет использоваться применительно к любой из двух сторон соединения (клиент и сервер). Термин «процессор DCCP» в

[1] Forward error correction.
[2] Semi-reliability.
[3] Application Programming Interface.
[4] Congestion Manager - менеджер насыщения (перегрузки).
[5] Профиль контроля насыщения.
[6] Network byte order.

общем случае обозначает любой хост, от которого может потребоваться обработка заголовков DCCP, т. е. включает конечные точки соединения и промежуточные узлы (middlebox) типа МСЭ[1] или систем трансляции адресов.

Соединения DCCP являются двухсторонними – данные могут передаваться от любой из конечных точек другой точке. Это означает, что данные и подтверждения могут передаваться одновременно в обоих направлениях. Логически, однако, соединение DCCP состоит из двух раздельных «односторонних» соединений, называемых «полусоединениями». Каждое полусоединение включает данные приложения, передаваемые одной конечной точкой, и встречный поток подтверждений от другой конечной точки, как показано на рисунке.



Хотя логически полусоединения различаются, на практике они перекрываются – пакет DCCP-DataAck, например, содержит данные приложения, относящиеся к одному полусоединению, и подтверждение, относящееся к другому.

В контексте одного полусоединения термины HC-Sender и HC-Receiver обозначают конечные точки, передающие прикладные данные и подтверждения, соответственно. Например, DCCP A будет HC-Sender, а DCCP B - HC-Receiver для полусоединения от A к B.

3.3. Признаки

Атрибуты соединения, согласованные двумя конечными точками, называются признаками протокола DCCP[2]. Многие параметры соединений DCCP, включая механизмы контроля насыщения, используемые для пары полусоединений, определяются признаками. Конечные точки согласуют признаки путём обмена опциями согласования[3] в заголовках DCCP.

Признаки DCCP идентифицируются их номерами и конечной точкой. Обозначение F/X представляет признак с номером F, связанный с конечной точкой X. Каждый корректный номер признака, таким образом, соответствует двум признакам, которые согласуются раздельно и могут иметь разные значения. Две конечные точки знают значения каждого корректного признака и согласны использовать его. DCCP A является «местоположением[4]» для всех признаков F/A и «удалённой стороной[5]» для признаков F/B.

3.4. Время кругового обхода

Измерение времени кругового обхода в DCCP выполняется механизмами контроля насыщения - различные механизмы могут измерять это время по-разному или не измерять вовсе. Однако сам протокол DCCP время от времени использует значение времени кругового обхода (в частности, для задания начальных значений некоторых таймеров). Каждая реализация DCCP определяет принятое по умолчанию время кругового обхода для использования в тех случаях, когда оценка этого значения не представляется возможной. По умолчанию не следует устанавливать для этого параметра значения меньше 0,2 сек (разумно консервативное значение для соединений TCP в Internet). Поведение протокола, описываемое в терминах «времени кругового обхода», реально относится к оценке RTT[6], сделанной неким механизмом CCID[7], или (при отсутствии возможности оценки) принятому по умолчанию значению времени кругового обхода.

Максимальное время жизни сегмента MSL[8] - это максимальное время, в течение которого пакет может оставаться в сети. Для протокола DCCP значение MSL следует делать равным аналогичному значению для протокола TCP, которое обычно составляет 2 минуты.

3.5. Ограниченность защиты

DCCP не обеспечивает защиты от атакующих, которые способны совать свой нос в процесс организации соединения или иным путём узнавать порядковые номера. Приложениям, которым требуется сильная защита, следует использовать IPsec [RFC2401]. В зависимости от уровня требуемой защиты может оказаться достаточным использование криптографических методов на уровне приложения. Эти вопросы более подробно рассматриваются в параграфах 7.5.5 и 18.

3.6. Отказоустойчивость

Реализации DCCP будут следовать принятому в TCP общему принципу отказоустойчивости - «be conservative in what you do, be liberal in what you accept from others[9]» [RFC793].

4. Обзор

Динамика соединений вышележащих уровней для DCCP в точности соответствует протоколу TCP. Каждое соединение включает три фазы – инициирование соединения, включая трехэтапное согласование, передача данных и разрыв соединения. Данные через соединение могут передаваться обоими путями. Подтверждения позволяют отправителю определить количество теряемых данных и предотвратить возникновение перегрузок в сети. Протокол DCCP обеспечивает семантику негарантированной доставки дейтаграмм, а не принятую в TCP семантику битовых потоков с гарантией доставки. Приложение должно упаковывать свои данные в явные кадры и при необходимости повторять

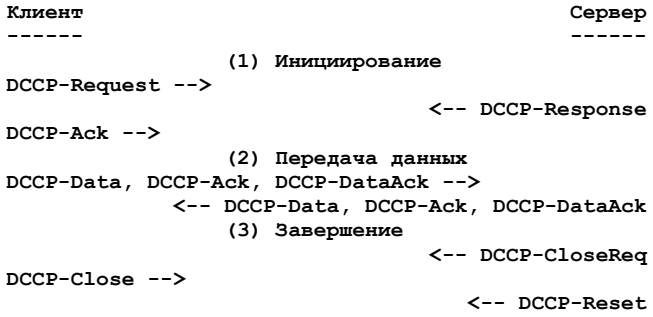
[1]Межсетевой экран – firewall. *Прим. перев.*
[2]DCCP feature – признак (свойство, характерная черта) DCCP.
[3]Feature negotiation option – опция согласования признаков (свойств).
[4]В оригинале - “feature location“. *Прим. перев.*
[5]В оригинале - “feature remote“. *Прим. перев.*
[6]Round-trip time – время кругового обхода.
[7]Congestion Control ID – идентификатор механизма контроля насыщения. *Прим. перев.*
[8]Maximum segment lifetime.
[9]Быть требовательным по отношению к собственным действиям и великодушным при восприятии действий других.

передачу таких кадров. Может оказаться полезным представление DCCP как TCP без семантики байтовых потоков и гарантий доставки или как UDP с контролем насыщения, согласованием параметров и подтверждениями.

4.1. Типы пакетов

Для реализации своих функций протокол DCCP использует 10 типов пакетов. Например, каждая попытка организации нового соединения начинается с передачи клиентом пакета DCCP-Request. Пакет DCCP-Request напоминает пакеты TCP SYN, но, поскольку DCCP-Request является специальным типом пакетов, не существует возможности передачи неожиданной комбинации флагов типа SYN+FIN+ACK+RST в TCP.

Обычно в течение срока жизни соединения используется восемь типов пакетов, как показано на рисунке. Отметим, что трехэтапное согласование[1] происходит как при организации, так и при разрыве соединения.



Два оставшихся типа пакетов применяются для ресинхронизации после случаев потери большого числа пакетов.

Каждый пакет DCCP начинается с базового заголовка[2] фиксированного размера. Отдельные типы пакетов включают дополнительные поля заголовка с фиксированными размерами. Например, пакеты DCCP-Ack включают номер подтверждения. Опции DCCP и данные приложения следуют после заголовка фиксированного размера.

Ниже перечислены типы пакетов.

DCCP-Request
Передаётся клиентом для организации соединения (первый шаг трехэтапного согласования).

CPD-Response
Передаётся сервером в ответ на пакет DCCP-Request (второй шаг трехэтапного согласования).

DCCP-Data
Служит для передачи данных приложения.

DCCP-Ack
Используется для передачи только подтверждения (без данных).

DCCP-DataAck
Используется для передачи данных приложения вместе с прицепленным подтверждением.

DCCP-CloseReq
Передаётся сервером в качестве запроса к клиенту на разрыв соединения.

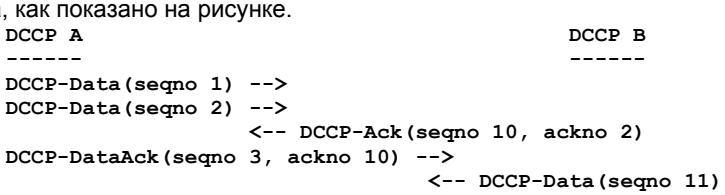
DCCP-Close
Используется клиентом для завершения соединения; в ответ передаётся пакет DCCP-Reset.

DCCP-Reset
Используется для завершения соединения (нормального или аварийного).

DCCP-Sync, DCCP-SyncAck
Служат для ресинхронизации порядковых номеров после потери большого числа пакетов.

4.2. Упорядочивание пакетов

Каждый пакет DCCP включает порядковый номер, что позволяет детектировать потерю пакетов и сообщать о ней. В отличие от порядковых номеров TCP, учитывающих байты, порядковые номера DCCP просто увеличиваются на 1 для каждого следующего пакета, как показано на рисунке.



Каждый пакет DCCP увеличивает значение порядкового номера, независимо от наличия в этом пакете данных приложения. Например, пакеты DCCP-Ack, содержащие только подтверждение, также увеличивают порядковый номер. Второй пакет DCCP B на рисунке использует порядковый номер 11, поскольку номер 10 был использован для подтверждения. Такая нумерация позволяет конечным точкам детектировать потерю пакетов, включая пакеты подтверждения. Это также означает, что конечные точки могут терять синхронизацию порядковых номеров после потери большого числа пакетов. Для восстановления синхронизации служат пакеты DCCP-Sync и DCCP-SyncAck (параграф 7.5).

Поскольку DCCP поддерживает семантику без гарантии доставки, здесь не используется повтор передачи и кумулятивные подтверждения в стиле TCP не имеют смысла. Поле Acknowledgement Number (номер подтверждения) в DCCP содержит наибольшее значение порядкового номера из принятых пакетов, а не минимальное значение порядкового номера, который ещё не получен. Отдельные опции показывают порядковые номера пакетов, которые не были получены.

[1] Three-way initiation handshake.
[2] Generic header.

4.3. Состояния

Конечные точки DCCP проходят различные состояний в течение срока существования соединения; эти состояния можно разделить на три фазы – организация соединения, обмен данными и завершение соединения. На рисунке справа показан типовой пример смены состояний для клиента и сервера.

```
Клиент                                                  Сервер
------                                                  ------
                        (0) Нет соединения
CLOSED                                                  LISTEN

                        (1) Инициирование
REQUEST         DCCP-Request -->
                                    <-- DCCP-Response   RESPOND
PARTOPEN        DCCP-Ack или DCCP-DataAck -->

                        (2) Передача данных
OPEN             <-- DCCP-Data, Ack, DataAck -->        OPEN

                        (3) Разрыв
                                    <-- DCCP-CloseReq   CLOSEREQ
CLOSING         DCCP-Close -->
                                       <-- DCCP-Reset   CLOSED
TIMEWAIT
CLOSED
```

Всего возможно девять состояний, которые перечислены ниже в порядке возрастания[1]. Следовательно, запись state >= CLOSEREQ означает, что соединение находится в состоянии state = CLOSEREQ, state = CLOSING или state = TIMEWAIT. В главе 8 приводится более детальное описание всех состояний.

CLOSED
Нет соединения.

LISTEN
Сокеты сервера находятся в пассивном состоянии прослушивания. Состояния LISTEN и CLOSED не связываются с каким-либо конкретным соединением DCCP.

REQUEST
Сокет клиента переходит в это состояние из состояния CLOSED после передачи пакета DCCP-Request с целью организации соединения.

RESPOND
Сокет сервера переходит в это состояние из состояния LISTEN после получения от клиента пакета DCCP-Request.

PARTOPEN
Сокет клиента переходит в это состояние из состояния REQUEST после получения от сервера пакета CPD-Response. Это состояние является третьим шагом трехэтапного согласования. Клиент может передавать данные приложений из этого состояния, но он **должен** включать во все такие пакеты номер подтверждения.

OPEN
Это состояние является центральной частью соединения DCCP. Сокеты клиента и сервера переходят в это состояние из состояний PARTOPEN и RESPOND, соответственно. Иногда мы будет говорить о состояниях SERVER-OPEN и CLIENT-OPEN, указывающих на состояние OPEN для клиента и сервера, соответственно.

CLOSEREQ
Сокет сервера переходит в это состояние из состояния SERVER-OPEN для того, чтобы запросить у клиента завершение соединения и переход в состояние TIMEWAIT.

CLOSING
Сокеты клиента и сервера переходят в это состояние для завершения соединения.

TIMEWAIT
Сокет клиента или сервера остаётся в этом состоянии в течение времени 2MSL (4 минуты) после разрыва соединения было разорвано для того, чтобы предотвратить ошибки, связанные с доставкой старых пакетов. Состояние TIMEWAIT устанавливается только на одной стороне (другая сразу же переходит в состояние CLOSED) и сервер может запросить переход клиента в состояние TIMEWAIT, используя для этого пакет типа DCCP-CloseReq.

4.4. Механизмы контроля насыщения

Для соединений DCCP осуществляется контроль насыщения, но в отличие от TCP, приложения DCCP могут сами выбирать механизм контроля. Фактически два полусоединения могут находиться под управлением разных механизмов. Механизмы контроля насыщения задаются однобайтовыми идентификаторами CCID[2]. Конечные точки согласуют свои значения CCID при организации соединения. Каждый идентификатор CCID описывает, как HC-Sender ограничивает скорость пакетов данных, а HC-Receiver передаёт информацию о насыщении в подтверждениях и т. д. В настоящее время определены значения CCID 2 и 3, а значения CCID 0, 1, 4 - 255 зарезервированы для использования в будущем.

CCID 2 обозначает механизм TCP-like Congestion Control[3], который похож на используемый протоколом TCP контроль насыщения. Отправитель поддерживает окно насыщения и передаёт пакеты, пока это окно не будет заполнено. Доставка пакетов подтверждается получателем. Индикация насыщения происходит путём отбрасывания пакетов и передачи уведомлений ECN[4] [RFC3168]. Откликом на возникновение перегрузки (насыщения) является уменьшение размера окна насыщения вдвое. Подтверждения в CCID 2 содержат порядковые номера всех полученных пакетов в некотором окне, что напоминает механизм селективного подтверждения (SACK) [RFC2018].

CCID 3 обозначает механизм TFRC[5] - основанный на выравнивании способ контроля насыщения, предназначенный для более мягкого, нежели в CCID 2 управления скоростью передачи в случаях перегрузки. Отправитель поддерживает значение скорости передачи, которое обновляется с использованием принятой от получателя оценки потери и

[1]Номеров состояний. *Прим. перев.*
[2]Congestion Control ID - идентификатор контроля насыщения. *Прим. перев.*
[3]Контроль насыщения в стиле TCP.
[4]Explicit Congestion Notification – явное уведомление о насыщении. *Прим. перев.*
[5]TCP-Friendly Rate Control – дружественный к TCP контроль скорости.

маркировки пакетов. CCID 3 отличается от TCP при рассмотрении коротких интервалов времени, но ведёт себя подобно TCP в течение более продолжительного периода.

В главе 10 более детально описаны CCID протокола DCCP. Поведение CCID 2 и 3 определено в отдельных документах [RFC4341, RFC4342].

4.5. Опции согласования признаков

Конечные точки DCCP используют опции Change и Confirm для согласования значений признаков. Согласование признаков почти всегда происходит в процессе согласования параметров на этапе организации соединения, но может начаться и в любой другой момент.

Всего существует четыре опции согласования признаков - Change L, Confirm L, Change R и Confirm R. Опции L передаются той стороной, с которой связан признак (feature location), а опции R передаются другой стороной (feature remote). Опция Change R говорит поддерживающей признак стороне: «измените значение признака как указано далее». Поддерживающая этот признак сторона отвечает опцией Confirm L, означающей: «значение изменено». Некоторые признаки позволяют указывать в опции Change R множество значений, задаваемых в порядке предпочтения, как показано на рисунке.

```
Клиент                                          Сервер
------                                          ------
Change R(CCID, 2) -->
                                  <-- Confirm L(CCID, 2)
            * согласие с CCID/Server = 2 *

Change R(CCID, 3 4) -->
                            <-- Confirm L(CCID, 4, 4 2)
            * согласие с CCID/Server = 4 *
```

Оба обмена опциями согласуют значение признака CCID/Server, который определяет CCID для полусоединения от сервера к клиенту. Во втором обмене клиент запрашивает у сервера использование CCID 3 или CCID 4, причём значение 3 является предпочтительным. Сервер выбирает значение 4 и показывает свой список предпочтений «4 2».

Опции Change L и Confirm R используются для согласования значений признаков по инициативе держателя признака (feature location). В показанном на рисунке примере сервер запрашивает для CCID/Server значение 3 или 2 (предпочтительно 3) и клиент соглашается со значением 3.

```
Клиент                                          Сервер
------                                          ------
                                <-- Change L(CCID, 3 2)
Confirm R(CCID, 3, 3 2)  -->
             * согласие с CCID/Server = 3 *
```

В главе 6 приводится дополнительная информация по вопросу согласования признаков, включая стратегию повтора, обеспечивающую гарантированное согласование.

4.6. Отличия от TCP

Отличия DCCP от протокола TCP кроме уже упомянутых включают следующие:

- Большее пространство для опций (до 1008 байтов на PMTU).
- Иной формат подтверждений. Идентификатор CCID для соединения определяет объем информации, передаваемой в подтверждениях. Например, в CCID 2 (TCP-like) передаётся примерно 1 подтверждение на каждые 2 пакета и в каждом подтверждении должны точно указываться полученные пакеты. В CCID 3 (TFRC) передаётся примерно 1 подтверждение за период кругового обхода и подтверждение должно покрывать по крайней мере протяжённость последнего интервала потерянных пакетов.
- Защита от DoS-атак[1]. Некоторые механизмы помогают ограничить количество состояний, которые могут заставить поддерживать на сервере DCCP клиенты с некорректным поведением. Опция Init Cookie аналогична SYN Cookies [SYNCOOKIES] в TCP и позволяет предотвратить атаки типа SYN-flood[2]. Только одна из конечных точек может сохранять состояние TIMEWAIT; пакет DCCP-CloseReq, который может быть передан только сервером, передаёт это состояние клиенту. Различные ограничители скорости позволяют предотвратить атаки, которые могут вызвать ненужную загрузку процессора или генерацию пакетов.
- Возможность различать разные типы потери пакетов. Опция Data Dropped (параграф 11.7) позволяет конечной точке объявить о том, что пакет был отброшен по причине его повреждения, переполнения приёмного буфера и т. п. Такая возможность облегчает исследования в направлении поиска методов реагирования на потери пакетов, не связанные с насыщением (хотя в настоящее время такие потери вызывают обычный отклик, характерный для насыщения).
- Подтверждаемость. В TCP пакет может подтверждаться только один раз при помещении принятой информации во входную очередь с гарантией доставки приложению. В DCCP такой подход не имеет смысла, поскольку приложение может, например, запросить отбрасывание пакетов из начала приёмного буфера. Пакет DCCP может быть подтверждён после успешной обработки его заголовка. Более точно, пакет становится подтверждаемым на этапе 8, описанного в параграфе 8.5 псевдокода обработки. Подтверждение не гарантирует доставки данных, однако опция Data Dropped может использоваться позднее для передачи информации об отбрасывании пакетов данных приложения.
- Отсутствие окна приёма. DCCP является протоколом с контролем насыщения, а не контролем потоков.
- Не возникает дубликатов при организации соединений. Каждое соединение имеет одного клиента и один сервер.

[1] Denial of Service – атаки, направленные на отказ служб.
[2] Поток пакетов с флагом SYN, запрашивающих организацию нового соединения.

- Отсутствие полуоткрытых состояний. DCCP не имеет состояний, соответствующих FINWAIT и CLOSEWAIT в TCP, когда одна сторона уже явно закрыла соединение, а вторая ещё держит его в активном состоянии. Однако использование опции Data Dropped с кодом 1 (Application Not Listening, см. параграф 11.7) может приводить к возникновению подобного эффекта.

4.7. Пример соединения

Ниже перечисляются этапы типичного соединения DCCP (описание является информационным, а не нормативным).

```
     Клиент                                        Сервер
     ------                                        ------
0.   [CLOSED]                                    [LISTEN]
1.   DCCP-Request -->
2.                                  <-- DCCP-Response
3.   DCCP-Ack -->
4.   DCCP-Data, DCCP-Ack, DCCP-DataAck -->
               <-- DCCP-Data, DCCP-Ack, DCCP-DataAck
5.                                  <-- DCCP-CloseReq
6.   DCCP-Close -->
7.                                     <-- DCCP-Reset
8.   [TIMEWAIT]
```

1. Клиент направляет серверу пакет DCCP-Request, задающий номера портов на стороне клиента и сервера, запрашиваемые услуги и все признаки, которые нужно согласовать (включая идентификатор механизма CCID, который клиент предлагает использовать серверу). Клиент может прицепить к пакету DCCP-Request запрос приложения, но сервер вправе игнорировать этот запрос.

2. Сервер передаёт клиенту пакет CPD-Response, показывающий готовность к работе с клиентом. Этот отклик содержит все признаки, которые желательно согласовать, и может также включать опции Init Cookie, которые будут «охватывать» всю эту информацию и должны возвращаться клиентом для завершения процедуры организации соединения.

3. Клиент передаёт серверу пакет DCCP-Ack, подтверждающий получение пакета CPD-Response. Этот пакет подтверждает начальный порядковый номер сервера и возвращает опции Init Cookie из пакета CPD-Response. Пакет может также продолжать согласование признаков. Клиент может добавить запрос прикладного уровня в пакет DCCP-DataAck.

4. Сервер и клиент обмениваются пакетами DCCP-Data, подтверждениями DCCP-Ack и, возможно, пакетами DCCP-DataAck, содержащими данные, с прицепленными к ним подтверждениями. Если у клиента нет данных для передачи серверу, последний будет передавать пакеты DCCP-Data и DCCP-DataAck, а клиент - только пакеты DCCP-Ack (однако клиент не может передавать пакеты DCCP-Data до получения от сервера по крайней мере одного пакета, отличного от CPD-Response).

5. Сервер передаёт пакет DCCP-CloseReq, запрашивающий закрытие соединения.

6. Клиент передаёт пакет DCCP-Close, подтверждающий закрытие.

7. Сервер передаёт пакет DCCP-Reset с кодом сброса (Reset) 1, «закрывает» соединение и сбрасывает его состояние. Пакеты DCCP-Reset являются частью процедуры нормального завершения соединений, описанной в параграфе 5.6.

8. Клиент получает пакет DCCP-Reset и сохраняет состояние[1] в течение двух сроков максимального времени жизни сегмента (2MSL), что позволяет оставшимся пакетам покинуть сеть.

При закрытии соединения по инициативе клиента процедура несколько меняется:

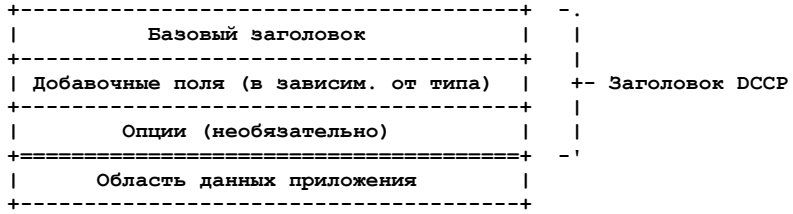
5b. Клиент передаёт пакет DCCP-Close, закрывающий соединение.

6b. Сервер передаёт пакет DCCP-Reset с кодом Reset 1, «закрывает» соединение и сбрасывает его состояние.

7b. Клиент получает пакет DCCP-Reset и удерживает состояние[1] в течение периода 2MSL, чтобы оставшиеся пакеты могли покинуть сеть.

5. Формат пакетов

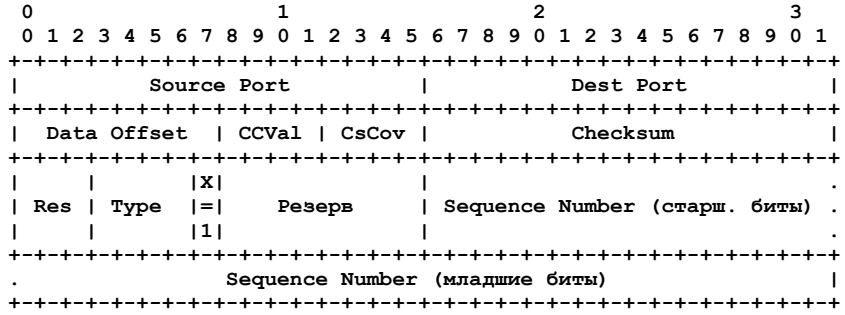
Заголовок DCCP может иметь размер от 12 до 1020 байтов. Начальная часть заголовка использует одинаковую семантику для всех определённых в настоящее время типов пакетов. Далее могут размещаться дополнительные поля фиксированных размеров, требуемые конкретным типом пакетов, а за ними - список опций переменной длины. Данные приложения размещаются после заголовка. В некоторых типах пакетов содержимое после заголовка может игнорироваться.



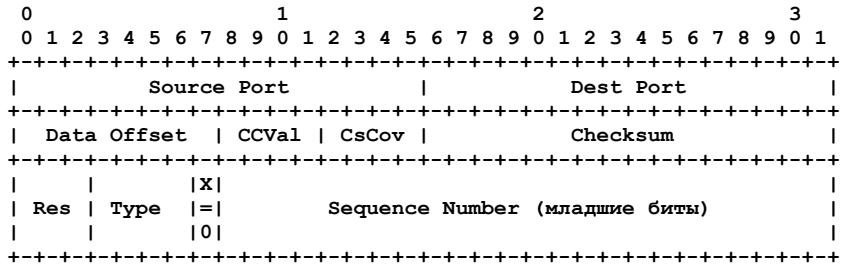
[1]TIMEWAIT. *Прим. перев.*

5.1. Базовый заголовок

Базовый заголовок DCCP может принимать различные формы в зависимости от значения бита X[1]. Если X = 1, поле Sequence Number имеет размер 48 битов, а размер базового заголовка составляет 16 байтов, как показано на рисунке.



Если же бит X имеет нулевое значение, передаются только младшие 24 бита порядкового номера (Sequence Number) и общий размер базового заголовка уменьшается до 12 байтов, как показано на рисунке.



Поля базового заголовка описаны ниже.

Source Port и Destination Port: каждое по 16 битов

Эти поля идентифицируют соединение, подобно аналогичным полям в TCP и UDP. Поле Source Port представляет номер порта конечной точки, передающей пакет, а Destination Port - номер порта на приёмной стороне. При организации соединения клиенту **следует** использовать случайное значение номера порта на своей стороне для снижения вероятности атак.

DCCP API следует трактовать номера портов подобно принятым для протоколов TCP и UDP трактовкам таких номеров. Например, машинам, разделяющим порты на привилегированные и непривилегированные для протоколов TCP и UDP, следует так же относиться к портам DCCP.

Data Offset: 8 битов

Задаёт смещение от начала заголовка DCCP до начала данных приложения в 32-битовых словах. Получатель **должен** игнорировать пакеты, в которых значение Data Offset меньше минимального размера заголовка для данного типа пакетов DCCP.

CCVal: 4 бита

Используется HC-Sender CCID. Например, отправитель CCID для полусоединения от A к B, который активен на DCCP A, **может** передать своему адресату 4 бита информации на пакет, кодируя эту информацию в CCVal. Отправитель **должен** устанавливать CCVal = 0, если его HC-Sender CCID не задаёт иного, а получатель **должен** игнорировать поле CCVal, если его HC-Receiver CCID не задаёт иного.

Checksum Coverage (CsCov): 4 бита

Поле Checksum Coverage определяет часть пакета, которая используется для расчёта контрольной суммы (поле Checksum). Эта часть всегда включает заголовок и опции DCCP, но прикладные данные могут быть исключены из расчёта контрольной суммы полностью или частично. Такое исключение может повысить производительность приложений, устойчивых к ошибкам в данных, при работе на шумных каналах (см. главу 9).

Checksum: 16 битов

Контрольная сумма Internet для заголовка DCCP (включая опции), псевдозаголовка сетевого уровня и, в зависимости от значения поля Checksum Coverage, части (включая нулевую) или всех прикладных данных (см. главу 9).

Reserved (Res): 3 бита

Отправитель **должен** устанавливать для этого поля нулевые значения во всех битах, а получатель **должен** игнорировать это поле.

Type: 4 бита

Это поле задаёт тип пакета. Возможные значения приведены в таблице 1.

Таблица 1. Типы пакетов DCCP

Тип	*Название*
0	DCCP-Request
1	CPD-Response
2	DCCP-Data
3	DCCP-Ack
4	DCCP-DataAck
5	DCCP-CloseReq
6	DCCP-Close
7	DCCP-Reset
8	DCCP-Sync
9	DCCP-SyncAck
10 - 15	Резерв

[1]Extended Sequence Numbers – флаг использования расширенных порядковых номеров.

Получатели **должны** игнорировать все пакеты с зарезервированными номерами типа. Таким образом, обработка и подтверждение пакетов зарезервированных типов являются **недопустимыми**.

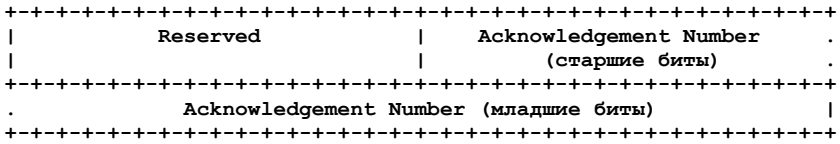
Extended Sequence Numbers (X): 1 бит

Установка этого флага указывает на использование расширенного базового заголовка с 48-битовыми порядковыми номерами и номерами подтверждений. Пакеты DCCP-Data, DCCP-DataAck и DCCP-Ack **могут** устанавливать для флага X значения 1 или 0. Все пакеты DCCP-Request, CPD-Response, DCCP-CloseReq, DCCP-Close, DCCP-Reset, DCCP-Sync и DCCP-SyncAck **должны** использовать X = 1; конечные точки **должны** игнорировать такие пакеты, если в них установлено X = 0. На высокоскоростных соединениях **следует** устанавливать X = 1 во всех пакетах для обеспечения более надёжной защиты от атак с угадыванием порядковых номеров (см. параграф 7.6).

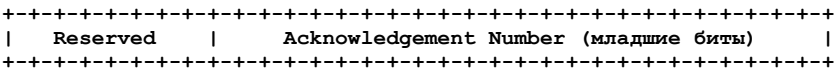
Sequence Number: 48 или 24 бита

Уникальный идентификатор пакета в последовательности все пакетов, переданных этим отправителем в данном соединении. Значение Sequence Number увеличивается на единицу для каждого следующего пакета, включая пакеты DCCP-Ack, не содержащие данных приложений (см. раздел 7).

Все определённые в настоящее время типы пакетов, за исключением DCCP-Request и DCCP-Data содержат подзаголовок Acknowledgement Number в форме четырёх или восьми байтов, следующих сразу после базового заголовка. Формат этого подзаголовка для случая X=1 показан на рисунке.



При X=0 передаются только 24 младших бита Acknowledgement Number и подзаголовок имеет иной формат, как показано на рисунке.



Reserved: 16 или 8 битов

Отправитель **должен** устанавливать для этого поля нулевые значения во всех битах, а получатель **должен** игнорировать это поле.

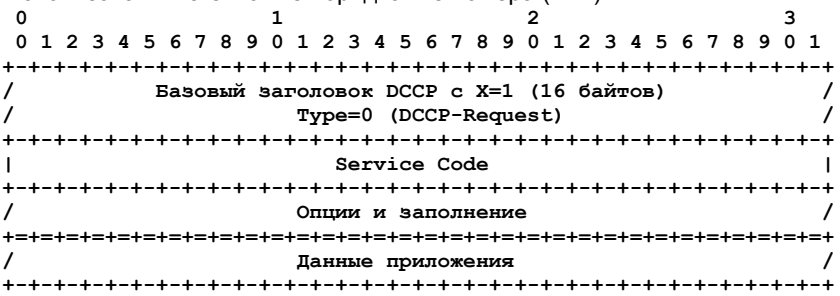
Acknowledgement Number: 48 или 24 бита

В общем случае содержит значение GSR[1] для всех подтверждаемых пакетов. Пакет является подтверждаемым тогда и только тогда, когда его заголовок успешно обработан получателем (см. дополнительную информацию в параграфе 7.4). Вместе с номером подтверждения могут использоваться такие опции, как Ack Vector (параграф 11.4), обеспечивающие точную информацию о доставленных пакетах.

Для пакетов DCCP-Sync и DCCP-SyncAck не требуется устанавливать Acknowledgement Number = GSR (см. параграф 5.7).

5.2. Пакеты DCCP-Request

Клиент инициирует соединение DCCP, передавая пакет DCCP-Request. Такие пакеты **могут** содержать данные приложения и **должны** использовать 48-битовые порядковые номера (X=1).



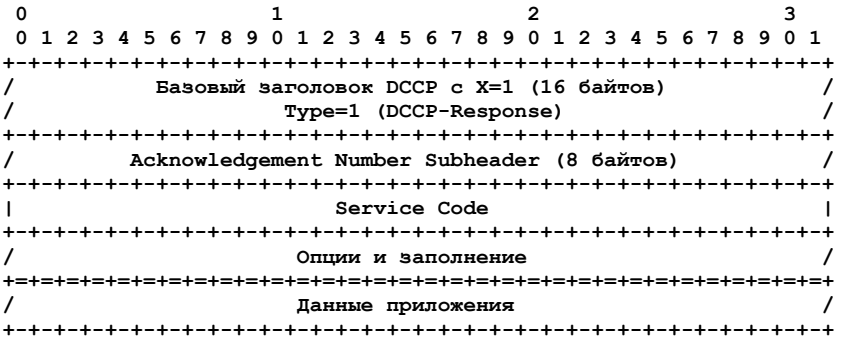
Service Code: 32 бита

Описывает сервис прикладного уровня, к которому хочет подключиться клиентское приложение. Коды сервиса предназначены для того, чтобы обеспечить информацию о прикладном протоколе, который будет использоваться в данном соединении, - эта информация помогает промежуточным устройствам и не требует полагаться на использование общепринятых номеров портов для служб (см. параграф 8.1.2).

5.3. Пакеты CPD-Response

Сервер отвечает на корректные пакеты DCCP-Request пакетом CPD-Response. Это является второй фазой трехэтапного согласования при организации соединения. Пакеты CPD-Response **могут** содержать данные приложения и **должны** использовать 48-битовые порядковые номера (X=1).

[1]Greatest Sequence Number Received – максимальный порядковый номер принятого пакета.



Acknowledgement Number: 48 битов

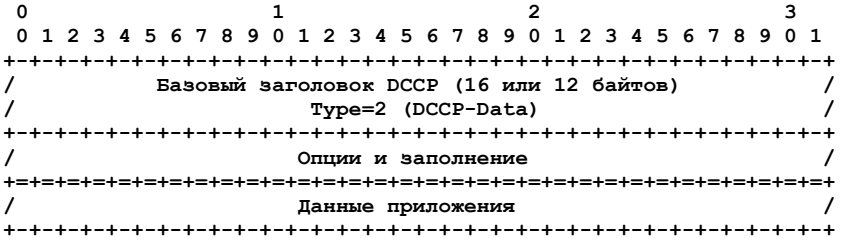
Содержит значение GSR. Поскольку пакеты CPD-Response передаются только на этапе организации соединения, значение этого поля всегда будет равно значению Sequence Number из принятого пакета DCCP-Request.

Service Code: 32 бита

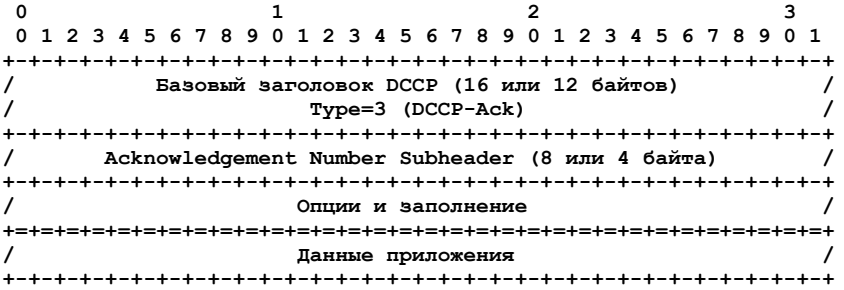
Должно совпадать со значением Service Code в соответствующем пакете DCCP-Request.

5.4. Пакеты DCCP-Data, DCCP-Ack и DCCP-DataAck

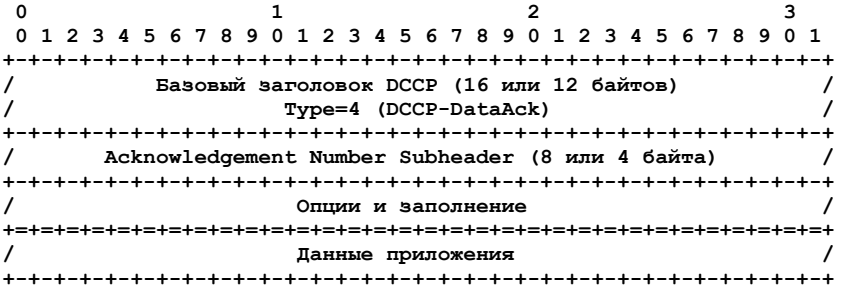
Основной этап каждого соединения DCCP связан с передачей данных и использует пакеты типов DCCP-Data, DCCP-Ack и DCCP-DataAck. Эти пакеты **могут** использовать 24-битовые порядковые номера в зависимости от значения признака Allow Short Sequence Numbers (параграф 7.6.1). Пакеты DCCP-Data передают данные приложений без подтверждений.



Пакеты DCCP-Ack не включают данных приложений, но содержат номер подтверждения (Acknowledgement Number). Такие пакеты служат исключительно для передачи подтверждений о доставке.



Пакеты DCCP-DataAck содержат как данные, так и Acknowledgement Number. Подтверждающая информация цепляется после данных. Формат таких пакетов показан на рисунке ниже.



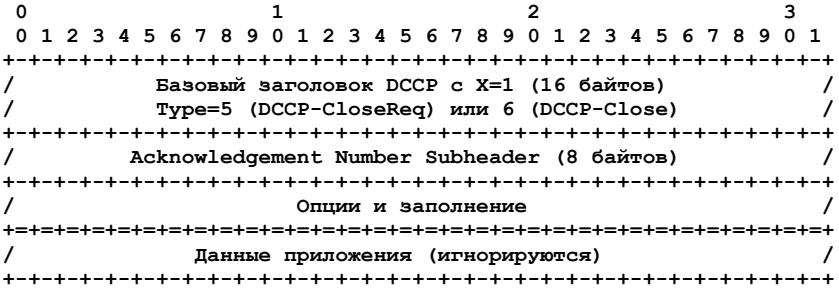
Пакеты DCCP-Data и DCCP-DataAck могут иметь область данных приложения нулевого размера, что показывает передачу приложением дейтаграммы нулевой длины. Этим пакеты указанных типов отличаются от пакетов DCCP-Request и CPD-Response, для которых нулевой размер области данных приложения говорит об отсутствии у приложения данных для передачи. API **следует** передавать информацию о получении дейтаграмм нулевой длины принимающему приложению.

Пакет DCCP-Ack **может** иметь область данных приложения ненулевого размера, которая будет дополнять пакеты DCCP-Ack до желаемой длины. Получатель **должен** игнорировать данные, принятые в пакетах DCCP-Ack.

Пакеты DCCP-Ack и DCCP-DataAck часто включают дополнительные опции подтверждения, такие, как Ack Vector, которые могут требоваться используемому механизму контроля насыщения.

5.5. Пакеты DCCP-CloseReq и DCCP-Close

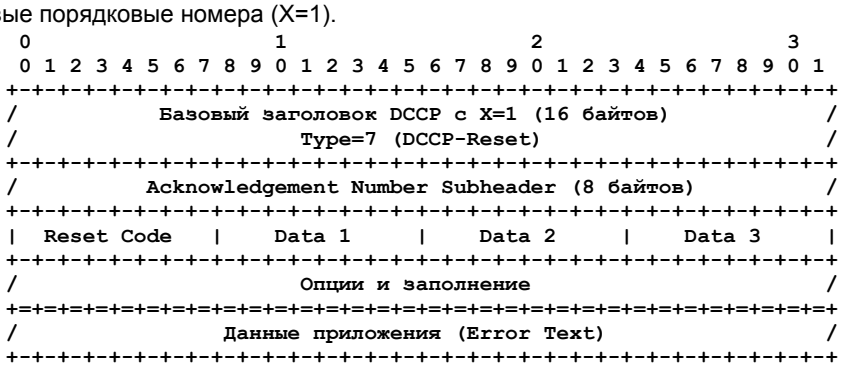
Пакеты DCCP-CloseReq и DCCP-Close начинают процесс согласования при нормальном завершении соединения. Клиент или сервер может передать пакет DCCP-Close, который будет вызывать передачу пакета DCCP-Reset. Сервер может также передавать пакет DCCP-CloseReq, который показывает желание сервера закрыть соединение, но нежелание сохранять на своей стороне состояние TIMEWAIT. Оба типа пакетов **должны** использовать 48-битовые порядковые номера (X=1).



Как и DCCP-Ack, пакеты DCCP-CloseReq и DCCP-Close **могут** содержать область данных приложения, отличного от 0 размера. Эти данные **должны** игнорироваться на принимающей стороне.

5.6. Пакеты DCCP-Reset

Пакеты DCCP-Reset служат для безусловного разрыва соединений. Нормальное завершение соединений также использует пакеты DCCP-Reset, но эти пакеты могут также применяться в других случаях, включая получение некорректного порядкового номера и/или некорректные отклики ECN Nonce Echo. Пакеты DCCP-Resets **должны** использовать 48-битовые порядковые номера (X=1).



Reset Code: 8 битов

Указывает причину, по которой отправитель пакета разрывает соединение DCCP.

Data 1, Data 2, Data 3: по 8 битов каждое

Поля Data обеспечивают дополнительные данные о причине разрыва соединения DCCP. Трактовка этих полей определяется значением Reset Code.

Область данных приложения: Error Text

При наличии поля Error Text оно содержит понятное для человека текстовое сообщение в кодировке Unicode UTF-8 (предпочтительно на английском языке) с более детальным описанием причины разрыва соединения. Например, пакет DCCP-Reset с Reset Code = 11 (Aggression Penalty) может содержать поле Error Text вида «Aggression Penalty: Received 3 bad ECN Nonce Echoes, assuming misbehavior[1]».

Определённые в настоящее время значения Reset Code перечислены ниже. Если явно не указано иное, поля Data 1, 2 и 3 **должны** устанавливаться отправителем в 0, а получатель должен игнорировать эти поля в пакетах DCCP-Reset. Коды сопровождаются описаниями конкретных ситуаций, которые будут вызывать передачу каждого значения Reset Code, однако описание таких ситуаций не является исчерпывающим.

0, Unspecified - причина не указана

Показывает отсутствие трактовки Reset Code. Использование Reset Code = 0 **не рекомендуется** - отправителю следует выбирать код, показывающий причину разрыва соединения.

1, Closed - закрытие соединения

Нормальное завершение работы соединения (см. параграф 8.3).

2, Aborted - прерывание соединения

Передающая конечная точка отказывается от соединения по причине того, что в нем ничего не происходит (см. параграфы 8.1.1 и 8.1.5).

3, No Connection - нет соединения

Соединения не существует (см. параграф 8.3.1).

4, Packet Error - ошибочный пакет

Получен корректный пакет неожиданного типа. Например, пакет DCCP-Data с корректной контрольной суммой в заголовке и порядковым номером был получен соединением, находящимся в состоянии REQUEST (см. параграф 8.3.1). Поле Data 1 содержит тип пакета-нарушителя (т. е., при получении ошибочного пакета типа 2 поле Data 1 будет иметь значение 2).

5, Option Error - ошибка в опциях

Опция была ошибочной и эта ошибка достаточно серьёзна для того, чтобы вызвать сброс соединения (см. параграфы 6.6.7, 6.6.8 и 11.4). Поле Data 1 содержит тип ошибочной опции, а Data 2 и Data 3 - два первых байта поля опции (или 0, если опция использовала менее 2 байтов данных).

6, Mandatory Error - ошибка в Mandatory

Передающая сторона не может обработать опцию O, которой непосредственно предшествовала опция Mandatory. Поля Data показывают тип и данные опции O с использованием такого же формата, как для Reset Code 5, Option Error (см. параграф 5.8.2).

7, Connection Refused - соединение отвергнуто

Значение Destination Port не соответствует порту, открытому для прослушивания. Этот код передаётся только в пакетах DCCP-Request (см. параграф 8.1.3).

8, Bad Service Code - некорректный код сервиса

Значение Service Code не совпадает с кодом сервиса, который связан в Destination Port. Этот код передаётся только в пакетах DCCP-Request (см. параграф 8.1.3).

[1] Наказание за агрессивное поведение - получены 3 некорректных сигнала ECN Nonce Echo, что говорит о недопустимом поведении.

9, Too Busy - сервер занят

Сервер слишком занят для того, чтобы принимать новые соединения. Этот код передаётся только в пакетах DCCP-Request (см. параграф 8.1.3).

10, Bad Init Cookie - некорректное значение Init Cookie

Клиент не вернул значение Init Cookie или указал некорректное значение (см. параграф 8.1.4).

11, Aggression Penalty - наказание за агрессию

Конечная точка определила некорректность поведения другой стороны в плане контроля насыщения (см. параграф 12.3).

12-127, зарезервированы

Резервные значения следует трактовать как Reset Code 0, Unspecified.

128-255, Связанные с CCID коды

Семантика этих кодов зависит от используемого в соединении CCID (см. параграф 10.3). Получателю следует трактовать неизвестные коды этого диапазона как Reset Code 0, Unspecified.

Сводка определённых в настоящее время кодов приведена в таблице 2.

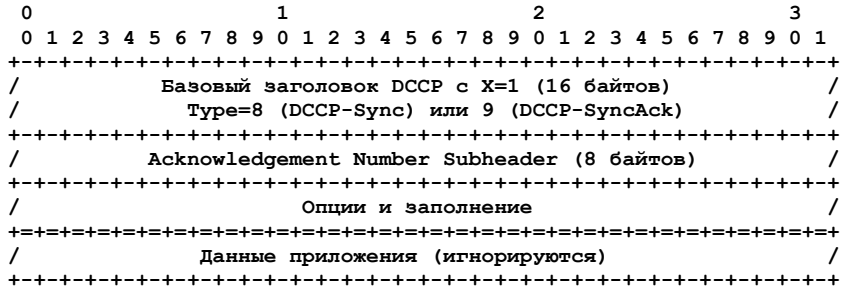
Таблица 2. Коды DCCP Reset.

Код сброса	*Имя*	*Данные 1*	*Данные 2 и 3*
0	Unspecified	0	0
1	Closed	0	0
2	Aborted	0	0
3	No Connection	0	0
4	Packet Error	Тип пакета	0
5	Option Error	Номер опции	Данные опции
6	Mandatory Error	Номер опции	Данные опции
7	Connection Refused	0	0
8	Bad Service Code	0	0
9	Too Busy	0	0
10	Bad Init Cookie	0	0
11	Aggression Penalty	0	0
12 - 127	Резерв	0	0
128 - 255	Коды, связанные с CCID	0	0

Опции в пакетах DCCP-Reset обрабатываются до разрыва соединения. Это означает, что некоторые комбинации опций (в частности, включающие опцию Mandatory) могут заставлять конечную точку отвечать на корректный пакет DCCP-Reset другим пакетом DCCP-Reset. Это может вести к всплеску передачи таких пакетов (reset storm), поскольку после сброса первой конечной точкой соединения второй пакет DCCP-Reset будет игнорироваться.

5.7. Пакеты DCCP-Sync и DCCP-SyncAck

Пакет DCCP-Sync помогают конечным точкам DCCP восстановить синхронизацию порядковых номеров после потери множества пакетов, а также восстановить соединения из полуоткрытого состояния. Каждый принятый корректный пакет DCCP-Sync вызывает незамедлительную передачу пакета DCCP-SyncAck. Оба типа пакетов **должны** использовать 48-битовые порядковые номера (X=1).



Поле Acknowledgement Number имеет в пакетах DCCP-Sync и DCCP-SyncAck специальную семантику. Во-первых, для пакетов, соответствующих DCCP-Sync поле Acknowledgement Number не должно быть подтверждением. Таким образом, для получателя **недопустимо** предполагать, что пакет был обработан, лишь на основании того, что его номер указан в поле Acknowledgement Number пакета DCCP-Sync. Это отличается от трактовки данного поля для остальных типов пакетов, где поле Acknowledgement Number по определению соответствует подтверждаемому пакету. Во-вторых, поле Acknowledgement Number в любом пакете DCCP-SyncAck **должно** соответствовать полю Sequence Number подтверждаемого пакета DCCP-Sync. В случаях изменения порядка это значение может отличаться от GSR.

Как и DCCP-Ack, пакеты DCCP-Sync и DCCP-SyncAck **могут** иметь поле данных приложения с отличным от нуля размером, которое получатель пакета **должен** игнорировать. Дополненные пакеты DCCP-Sync могут быть полезны для выполнения процедур Path MTU discovery (см. раздел 14).

5.8. Опции

Любой пакет DCCP может содержать опции, которые размещаются в конце заголовка DCCP. Размер каждой опции кратен 8 битам. Отдельные опции не дополняются для выравнивания по 32-битовой границе и каждая опция может начинаться на границе любого байта. Однако полный набор опций **должен** дополняться соответствующим числом байтов для выравнивания по 32-битовой границе; для добавочных байтов **должна** использоваться опция Padding. Все имеющиеся в пакете опции учитываются при расчёте контрольной суммы заголовка.

Первый байт каждой опции определяет её тип. Опции типов 0 - 31 являются однобайтовыми. В остальных опциях второй байт указывает размер опции. Размер учитывает два первых байта, определяющих тип и размер опции, а также поле данных опции, следовательно значение поля размера во всех случаях должно быть не меньше 2.

Опции **должны** обрабатываться последовательно, начиная с первой опции в заголовке пакета. Опции неизвестных типов **должны** игнорироваться. Так же **должны** игнорироваться опции некорректного размера (значение поля размера опции меньше 2 или больше оставшейся части пространства опций в заголовке пакета) вместе с любым дополнительным пространством опций вслед за опцией с некорректно указанной длиной. Если явно не указано иное, разные экземпляры одной опции в заголовке пакета **должны** обрабатываться независимо. Для некоторых опций это может означать, что будет использоваться только последнее корректное значение данной опции в заголовке пакета.

Определённые на сегодняшний день опции перечислены в таблице 3.

Таблица 3. Опции DCCP

Опция			***DCCP-Data?***	***Описание***
Тип	***Размер***	***Значение***		
0	1	Padding	+	5.8.1
1	1	Mandatory	-	5.8.2
2	1	Slow Receiver	+	11.6
3 - 31	1	Резерв	-	
32	переменный	Change L	-	6.1
33	переменный	Confirm L	-	6.2
34	переменный	Change R	-	6.1
35	переменный	Confirm R	-	6.2
36	переменный	Init Cookie	-	8.1.4
37	3 - 8	NDP Count	+	7.7
38	переменный	Ack Vector [Nonce 0]	-	11.4
39	переменный	Ack Vector [Nonce 1]	-	11.4
40	переменный	Data Dropped	-	11.7
41	6	Timestamp	+	13.1
42	6/8/10	Timestamp Echo	+	13.3
43	4/6	Elapsed Time	-	13.2
44	6	Data Checksum	+	9.2
45 - 127	переменный	Резерв		
128 - 255	переменный	Опции, связанные с CCID	-	10.3

Не все опции подходят для конкретного типа пакетов. Например, поскольку опция Ack Vector интерпретируется относительно номера подтверждения, она не может использоваться в пакетах DCCP-Request и пакетах DCCP-Data, не содержащих поля Acknowledgement Number. Если опция включена в неподходящий тип пакета, в общем случае она **должна** игнорироваться; такие случаи рассматриваются при описании отдельных опций. В таблице указаны ограничения общего значения - если в колонке DCCP-Data? указано «-», соответствующая опция **должна** игнорироваться при её получении в пакетах DCCP-Data (в параграфе 7.5.5 рассматриваются причины этого).

Опции с некорректными значениями **должны** игнорироваться, если явно не указано иное. Например, любая опция Data Checksum со значением 4 в поле размера **должна** игнорироваться, поскольку все корректные опции Data Checksum имеют размер 6.

В этом параграфе описываются две базовых опции - Padding и Mandatory. Остальные опции рассматриваются позднее.

5.8.1. Опция Padding

```
+--------+
|00000000|
+--------+
  Type=0
```

Padding представляет собой однобайтовую опцию no-operation, используемую для заполнения между другими опциями или после них. Если размер остальных опций пакета не кратен 32 битам, опцию Padding **требуется** включать для выравнивания размера пространства опций по границе слова, заданной полем Data Offset. Заполнение может также использоваться между опциями (например, для выравнивания следующей опции по 32-битовой границе). Нет гарантий использования этой опции на стороне отправителя, поэтому получатели должны быть готовы к обработке опций, который начинаются не на границе слова.

5.8.2. Опция Mandatory

```
+--------+
|00000001|
+--------+
  Type=1
```

Mandatory - однобайтовая опция, которая маркирует непосредственно следующую за ней опцию, как обязательную для исполнения. Предположим, что следующей опцией является O. Тогда опция Mandatory не будет иметь эффекта, если приёмная конечная точка DCCP понимает и обрабатывает опцию O. Если же эта конечная точка не может понять или обработать опцию O, она **должна** будет сбросить соединение, используя Reset Code 6, Mandatory Failure. Например, конечная точка будет сбрасывать соединение в тех случаях, когда она не понимает тип опции O; понимает тип, но не понимает данные, данные некорректны для этого типа опции; опция O относится к типу согласования признаков, а данная конечная точка не понимает указанный номер признака; данная конечная точка понимает опцию, но не может выполнить предполагаемых этой опцией действий. Приведённый список не является исчерпывающим и в описаниях отдельных опций могут указываться другие ситуации, в которых конечной точке следует сбрасывать соединение, а также ситуации, в которых этого делать не следует.

Опцию Mandatory **недопустимо** включать в пакеты DCCP-Data и при получении опции Mandatory в пакетах DCCP-Data она **должна** игнорироваться.

Соединение считается ошибочным и его следует сбросить с использованием Reset Code 5, Option Error, если опция O отсутствует (т. е., опция Mandatory является последней в списке опций), или опция O также является Mandatory.

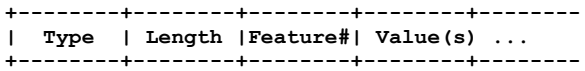
Однако комбинация опций Mandatory и Padding является корректной и **должна** обрабатываться как два байта заполнения (Padding).

В параграфе 6.6.9 более подробно описаны опции согласования признака Mandatory.

6. Согласование признаков

Четыре опции DCCP - Change L, Confirm L, Change R и Confirm R - используются для согласования значение признаков. Опции Change инициируют согласование, а опции Confirm завершают его. Опции L передаются держателем признака, а опции R относятся к удалённому признаку. Для опций Change используется повтор передачи с целью обеспечения гарантий доставки.

Все эти опции используют однотипный формат, показанный на рисунке. Первый байт данных опции задаёт номер признака, а следующие байты содержат одно или несколько значений этого признака. Точный формат значения признака определяется типом признака (см. параграф 6.3).

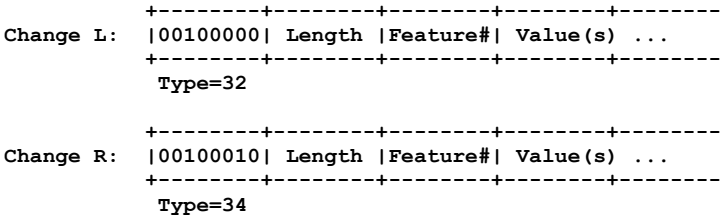


Номер признака и тип опции (L или R) совместно обеспечивают уникальную идентификацию признака, к которому относится данная опция. Точный формат значений зависит от типа признака.

Опции согласования признаков **недопустимо** передавать в пакетах DCCP-Data и полученные в таких пакетах опции согласования признаков **должны** игнорироваться.

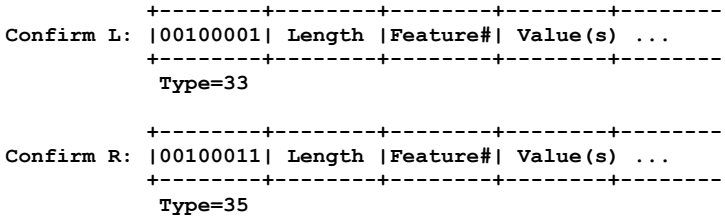
6.1. Опции Change

Опции Change L и Change R инициируют согласование признака. Выбор опции для использования зависит от места расположения признака. Для начала согласования признака F/A конечная точка DCCP A будет передавать опцию Change L, а для начала согласования признака F/B – опцию Change R. Передача опций Change повторяется до тех пор, пока не будет получен отклик. Опции содержат по крайней мере одно значение, поэтому размер опции равен, как минимум, 4.



6.2. Опции Confirm

Опции Confirm L и Confirm R завершают согласование признаков и передаются в ответ на опции Change R и Change L, соответственно. **Недопустима** генерация этих опций иначе, чем в ответ на получение опции Change. Опции Confirm не требуется передавать повторно, поскольку при необходимости будет повторно передана опция Change. Первый байт значения опции Confirm содержит номер признака из соответствующей опции Change. За ним следует выбранное значение признака (Value) и может также указываться список предпочтений отправителя.



Если конечная точка получает некорректную опцию Change (с неизвестным номером признака или недопустимым значением), она будет возвращать “пустую” опцию Confirm, содержащую номер вызвавшего проблему признака, но не содержащую значения. Такие опции имеют размер 3.

6.3. Правила согласования

Правила согласования определяют, как из двух наборов предпочтительных значений для данного признака выбирается однозначный результат. Правила определяются только номером признака. Каждое правило должно включать свойство, которое позволяет однозначно определить содержимое опций Change, переданных двумя конечными точками.

Все существующие в настоящий момент признаки DCCP используют одно из двух правил согласования – SP (приоритет сервера) или NN (не согласуется).

6.3.1. Приоритет сервера

Значение признака представляет собой строку фиксированного размера (определяется номером признака). Каждая опция Change содержит список значений в порядке предпочтения, начиная с самого предпочтительного значения. Каждая опция Confirm содержит подтверждаемое значение, за которым может следовать список предпочтений передающей опцию точки. Таким образом, текущее значение признака будет в общем случае дважды появляться в поле данных опции Confirm – один раз в качестве выбранного значения, а другой – в списке предпочтений.

Для согласования списков предпочтений выбирается первая опция из списка предпочтений сервера, которая присутствует также в списке предпочтений клиента. Если совпадающего значения нет, **недопустимо** изменять значение признака и опция Confirm будет подтверждать прежнее значение признака (если опция Change не помечена, как Mandatory; см. параграф 6.6.9).

6.3.2. Согласование не используется

Значение признака представляет собой строку байтов. Каждая опция содержит ровно одно значение признака. Держатель признака (feature location) сообщает о новом значении, передавая опцию Change L. Другая сторона (feature remote) **должна** принимать любое значение, отвечая с помощью опции Confirm R, содержащей новое значение. В ответ на некорректную опцию **должна** возвращаться пустая опция Confirm R (если опция Change L не была помечена, как Mandatory; см. параграф 6.6.9). Опции Change R и Confirm L **недопустимо** передавать для несогласуемых признаков (см. параграф 6.6.8). Несогласуемые признаки используют механизм согласования лишь для обеспечения гарантий доставки.

6.4. Номера признаков

Определяемые в этом документе признаки перечислены в таблице 4.

Таблица 4. Номера признаков DCCP.

Номер	Значение	Правило	Начальное значение	Обязательный	Описание
0	Резерв	SP			
1	Congestion Control ID (CCID)	NN	2	+	10
2	Allow Short Seqnos	SP	0	+	7.6.1
3	Sequence Window	NN	100	+	7.5.2
4	ECN Incapable	SP	0	-	12.1
5	Ack Ratio	NN	2	-	11.3
6	Send Ack Vector	SP	0	-	11.5
7	Send NDP Count	SP	0	-	7.7.2
8	Minimum Checksum Coverage	SP	0	-	9.2.1
9	Check Data Checksum	SP	0	-	9.3.1
10 - 127	Резерв				
128 - 155	Связанные с CCID признаки				10.3

Правило используется для согласования значений этого признака (SP – приоритет сервера, NN – признак не согласуется).

Начальное значение – начальное значение признака. Для всех признаков начальные значения известны.

Обязательный - значение «+» указывает признаки, которые **должны** поддерживаться каждой реализацией DCCP. Значение «-» указано для признаков, которые подобны расширениям (см. главу 15), и можно без опаски отвечать на опцию Change для такого признака пустой опцией Confirm. Естественно, механизмы CCID могут требовать поддержки определённых признаков - например, протокол DCCP, реализующий CCID 2, **должен** поддерживать признаки Ack Ratio и Send Ack Vector.

6.5. Примеры согласования признаков

На рисунке показаны три примера согласования признаков, связанных с сервером. Два первых примера используются для согласования признака Congestion Control ID, а последний - для согласования признака Ack Ratio.

```
                Клиент                    Сервер
                ------                    ------
1. Change R(CCID, 2 3 1)  -->
   ("2 3 1" – список предпочтений клиента)
2.                        <--  Confirm L(CCID, 3, 3 2 1)
                          (3 – согласованное значение;
                          "3 2 1" – список предпочтений сервера)
            * согласие с CCID/Server = 3 *


1.                  XXX   <--  Change L(CCID, 3 2 1)
2.                        Повтор передачи:
                          <--  Change L(CCID, 3 2 1)
3. Confirm R(CCID, 3, 2 3 1)  -->
            * согласие с CCID/Server = 3 *


1.                        <--  Change L(Ack Ratio, 3)
2. Confirm R(Ack Ratio, 3)  -->
            * согласие с Ack Ratio/Server = 3 *
```

Следующий пример показывает одновременное согласование признаков.

```
                Клиент                    Сервер
                ------                    ------
  1a. Change R(CCID, 2 3 1)  -->
   b.                        <--  Change L(CCID, 3 2 1)
  2a.                        <--  Confirm L(CCID, 3, 3 2 1)
   b. Confirm R(CCID, 3, 2 3 1)  -->
              * согласие с CCID/Server = 3 *
```

Ниже описано кодирование байтов для некоторых опций Change и Confirm. Все представленные опции передаются конечной точкой DCCP A.

Change L(CCID, 2 3) = 32,5,1,2,3
DCCP B следует изменить значение CCID/A (признак 1, правило приоритета сервера); предпочитаемые DCCP A значения признака - 2 и 3, в указанном порядке.

Change L(Sequence Window, 1024) = 32,9,3,0,0,0,0,4,0
DCCP B следует изменить значение Sequence Window/A (признак 3, без согласования), установив для него 6-байтовую строку 0,0,0,0,4,0 (1024).

Confirm L(CCID, 2, 2 3) = 33,6,1,2,2,3
DCCP A меняет значение CCID/A на 2; предпочтительными значениями являются 2 и 3 в указанном порядке.

Empty Confirm L(126) = 33,3,126
DCCP A не поддерживает признак с номером 126 или DCCP B предлагает некорректное значение для признака 126/A.

Change R(CCID, 3 2) = 34,5,1,3,2
DCCP B следует изменить значение CCID/B; предпочитаемые DCCP A значения - 3 и 2 в указанном порядке.

Confirm R(CCID, 2, 3 2) = 35,6,1,2,3,2
DCCP A меняет значение CCID/B на 2; предпочтительными значениями являются 3 и 2 в указанном порядке.

Confirm R(Sequence Window, 1024) = 35,9,3,0,0,0,0,4,0
DCCP A меняет значение Sequence Window/B на 6-байтовую строку 0,0,0,0,4,0 (1024).

Empty Confirm R(126) = 35,3,126
DCCP A не поддерживает признак с номером 126 или предложенное DCCP B значение признака 126/B некорректно.

6.6. Обмен опциями

Для обмена опциями согласования признаков существует несколько базовых правил.

1. Каждая опция Change, переданная с соблюдением порядка, даёт в ответ опцию Confirm.
2. Передача опций Change повторяется до тех пор, пока в ответ не будет получена опция Change.
3. Опции согласования признаков обрабатываются строго по нарастанию значений порядковых номеров пакетов.

Остальная часть этой главы посвящена более детальному описанию этих правил.

6.6.1. Нормальный обмен

Опции Change генерируются в тех случаях, когда конечная точка DCCP хочет изменить значение того или иного признака. Обычно это происходит в начале соединения, хотя изменения могут возникать и в процессе работы. Мы говорим, что конечная точка «генерирует» или «передаёт» опцию Change L или Change R, понимая, что на самом деле эта опция просто включается в пакет. Конечная точка может присоединить опцию к имеющемуся пакету (например, DCCP-Request) или создать специальный пакет согласования признака (обычно DCCP-Ack или DCCP-Sync), который будет служить лишь для передачи опции. Пакеты согласования признаков контролируются соответствующим механизмом контроля насыщения. Например, DCCP A может передать пакет DCCP-Ack или DCCP-Sync для согласования признака лишь с том случае, когда CCID для полусоединения от B к A будет позволять передачу DCCP-Ack. Кроме того, конечной точке **следует** генерировать не более одного пакета согласования признаков за период кругового обхода.

При получении опции Change L или Change R конечная точка DCCP проверяет включенный в неё список предпочтений, согласует с собственными предпочтениями и возвращает в ответ опцию Confirm R или Confirm L, соответственно, информирующую партнёра о новом значении или о том, что признак не был понят. Каждая опция Change, полученная без нарушения порядка, **должна** приводить к передаче соответствующей опции Confirm и любой пакет, содержащий опцию Confirm, должен включать Acknowledgement Number (см. параграф 6.6.4, описывающий детектирование и обработку нарушения порядка доставки опций Change). Созданные опции Confirm могут присоединяться к имеющемуся пакету (например, CPD-Response или DCCP-SyncAck) или включаться в специально создаваемый пакет согласования признака, как было описано выше.

Передающая Change конечная точка **должна** дождаться получения ответной опции Confirm и только после этого изменять значение признака. Передающая опцию Confirm конечная точка изменяет значение признака сразу же после передачи опции Confirm.

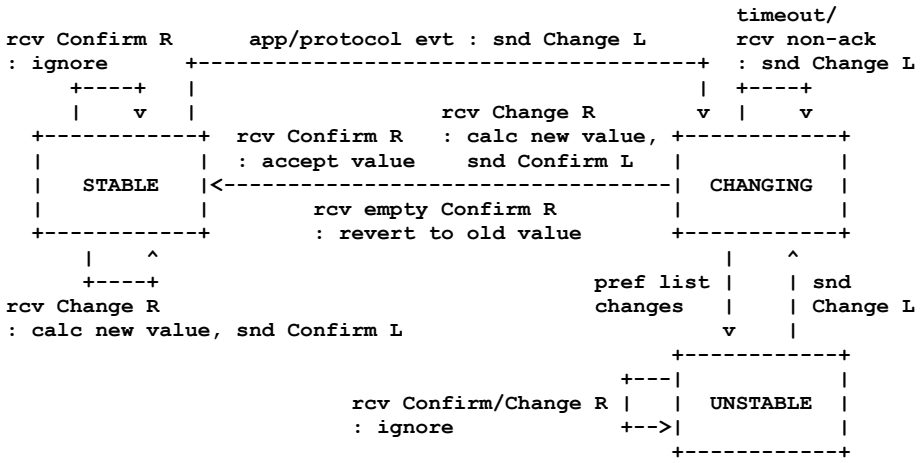
Пакет **может** содержать более одной опции согласования признаков и возможно даже включать две опции, относящиеся к одному признаку. Как обычно опции будут обрабатываться в порядке их следования в заголовке.

6.6.2. Обработка полученных опций

Для каждого признака конечная точка DCCP может находиться в одном из трёх состояний.

Стабильное состояние (STABLE) является нормальным состоянием каждого признака - обе конечные точки знают значение признака и считают, что другая сторона согласна с этим значением. Конечная точка переходит в состояние CHANGING после передачи первой опции Change для признака и возвращается в состояние STABLE после получения соответствующей опции Confirm. Нестабильное состояние (UNSTABLE) показывает, что конечная точка, находящаяся в состоянии CHANGING, изменила свой список предпочтений, но ещё не передала опцию Change с новым списком.

Смена состояния признака в точке его реализации происходит в соответствии с приведённым выше рисунком. На этой схеме не учитываются вопросы корректности порядковых номеров и опций - они отдельно описываются приведённым ниже псевдокодом.



Поддерживающей признак стороне **следует** использовать приведённый здесь псевдокод, который соответствует диаграмме состояний, для реагирования на каждую корректную опцию согласования признаков, полученную в корректном пакете, не относящемся к непосредственной передаче данных. Псевдокод содержит обозначения P.seqno и P.ackno, относящиеся к свойствам пакета, O.type и O.len, относящиеся к свойствам опции, FGSR и FGSS, относящиеся к свойствам соединения и обеспечивающие обработку случаев нарушения порядка доставки в соответствии с параграфом 6.6.4. F.state обозначает состояние признака (STABLE, CHANGING, UNSTABLE), а F.value – его значение.

Сначала проверяется, что данный признак известен (параграф 6.6.7).

```
Если F неизвестно
   Если опция помечена, как Mandatory,   /* параграф 6.6.9 */
      Сброс соединения (Reset) и возврат
```

```
      Иначе, если O.type == Change R
         Передача пустой опции Confirm L в будущем пакете
      Возврат
```

Далее проверяется порядок доставки (параграф 6.6.4).

```
   Если F.state == UNSTABLE или P.seqno <= FGSR или (O.type == Confirm R и P.ackno < FGSS)
      Игнорировать опцию и вернуться
```

После этого проверяется опция Change R.

```
   Если O.type == Change R
      Если значение опции корректно   /* параграф 6.6.8 */
         Рассчитать новое значение
         Передать опцию Confirm L в будущем пакете
         Установить F.state := STABLE
      Иначе, если опция была помечена, как Mandatory
         Сброс соединения (Reset) и возврат
      Иначе
         Передача пустой опции Confirm L в будущем пакете
         /* Сохраняется существующее состояние. Если состояние CHANGING, данная
            конечная точка позднее будет повторять передачу своей опции Change L. */
```

Далее обрабатывается опция Confirm R (только в состоянии CHANGING).

```
   Если F.state == CHANGING и O.type == Confirm R
      Если O.len > 3   /* непустая опция */
         Если значение опции корректно
            Установить F.value := новое значение
         Иначе
            Сброс соединения (Reset) и возврат
      Установить F.state := STABLE
```

Варианты этой диаграммы состояний и псевдокода используются также на удалённой стороне - в этом случае просто меняются местами обозначения L и R, чтобы они относились к опциям Change R и Confirm L.

6.6.3. Потеря и повторная передача пакетов

Пакеты с опциями Change и Confirm могут теряться или задерживаться в сети. Следовательно, опции Change будут передаваться повторно для обеспечения гарантий доставки. Этот процесс называется повтором передачи, хотя это и не является повтором в полной мере, поскольку повторные опции Change передаются в новых пакетах с иными порядковыми номерами.

Конечная точка, находящаяся в состоянии CHANGING, передаёт новую опцию Change, после того, как она решит, что не получила отклика от другой конечной точки. Новая опция Change не обязана совпадать с оригиналом, поскольку защита от нарушения порядка доставки обеспечит согласование на базе наиболее свежей опции.

Конечная точка, находящаяся в состоянии CHANGING, **должна** продолжать повтор передачи опций Change, пока не будет получен тот или иной ответ или соединение не будет разорвано.

Конечным точкам **следует** использовать таймер с экспоненциальным ростом для принятия решения о повторе передачи опций Change (пакеты, генерируемые специально для согласования признаков, **должны** использовать такой таймер). Начальное значение для таймера должно быть не меньше одного периода кругового обхода и его следует увеличивать не менее, чем до 64 секунд. Процедура backoff защищает от задержки согласования в результате использования алгоритмов проверки порядковых номеров, описанных в следующем параграфе. Как обычно, конечные точки могут добавлять опции Change в пакеты, которые уже приготовлены к передаче, или создавать новые пакеты для доставки этих опций. Все новые пакеты контролируются имеющим к ним отношение механизмом контроля насыщения.

Опции Confirm никогда не передаются повторно, но передающая опцию Confirm конечная точка **должна** генерировать опцию Confirm после приёма. каждой опции Change, доставленной без нарушения порядка.

6.6.4. Нарушение порядка доставки

Изменение порядка доставки пакетов через сеть может приводить к получению опций Change и Confirm в неожиданном порядке. Конечные точки **должны** игнорировать опции согласования признаков, которые были получены с нарушением порядка возрастания значений Sequence Number. Далее в этом параграфе рассматриваются два алгоритма, соответствующих этому требованию.

Первый алгоритм использует две переменных для порядковых номеров, которые поддерживает каждая из конечных точек соединения.

FGSR - Feature Greatest Sequence Number Received - максимальный порядковый номер, полученный для признаков

> Максимальное значение порядкового номера полученных корректных пакетов, содержащих по крайней мере одну опцию согласования признаков (Change и/или Confirm). В качестве начального значения используется ISR - 1.

FGSS - Feature Greatest Sequence Number Sent – максимальный порядковый номер, переданный для признаков

> Максимальное значение порядкового номера полученных корректных пакетов, содержащих по крайней мере одну новую опцию Change. Опция Change считается новой тогда и только тогда, когда она была сгенерирована в процессе перехода из состояния STABLE или UNSTABLE в состояние CHANGING. Опции Change, сгенерированные в состоянии CHANGING, являются повторами и **должны** в точности совпадать с переданной ранее новой опцией, что обеспечивает устойчивость к нарушению порядка доставки. Начальным значением FGSS является ISS.

Каждая конечная точка принимает решение об обработке полученных опций согласования признаков на основании проверки двух условий для порядковых номеров.

1. Если порядковый номер пакета не превышает значения FGSR, эта опция Change **должна** игнорироваться.
2. Если порядковый номер пакета не превышает FGSR и пакет не имеет поля Acknowledgement Number или значение этого поля меньше FGSS, данная опция Confirm **должна** игнорироваться.

В другом варианте алгоритма конечная точка **может** поддерживать раздельные переменные FGSR и FGSS для каждого признака. Значение FGSR(F/X) будет равно наибольшему порядковому номеру, полученному в пакетах, содержащих опции Change или Confirm для данного признака F/X. Значение FGSS(F/X) определяется аналогично. Этот алгоритм требует больше переменных, но он немного лучше подходит для случаев обработки множества перекрывающихся согласований признаков. **Может** использоваться любой из этих алгоритмов, но **рекомендуется** использовать первый алгоритм с переменными FGSR и FGSS для соединения в целом.

Одним из следствий этих правил является то, что конечная точка в состоянии CHANGING будет игнорировать все опции Confirm, которые не подтверждают последнюю переданную опцию Change. Это гарантирует, что согласие будет достигнуто с учётом наиболее свежего списка предпочтений конечной точки.

6.6.5. Смена предпочтений

Конечным точкам разрешается в любое время изменять свой список предпочтений. Однако, конечная точка, меняющая этот список из состояния CHANGING, **должна** перейти в состояние UNSTABLE. Она будет возвращаться в состояние CHANGING после передачи опции Change с новым списком предпочтений. Это обеспечивает достижение согласия на основе последних списков предпочтений. Без перехода в состояние UNSTABLE одновременное согласование (когда конечные точки начинают независимо согласовывать значение одного признака) может приводить к прерыванию согласования признака конечной точкой, предполагающей, что признак имеет другое значение.

6.6.6. Одновременное согласование

Две конечных точки могут одновременно начать согласование одного признака после чего конечная точка, находящаяся в состоянии CHANGING, получит опцию Change с тем же значением. Получение таких опций Change может действовать как отклик на исходные опции Change. Находящаяся в состоянии CHANGING конечная точка **должна** проверить список предпочтений в полученной опции Change, сравнить его со своим списком предпочтений (который указан в сгенерированной опции Change) и сгенерировать соответствующую опцию Confirm. После этого конечная точка может переходить в состояние STABLE.

6.6.7. Неизвестные признаки

Конечные точки могут получать опции Change, указывающие номера неизвестных данной точке признаков (например, когда расширенная реализация DCCP работает с обычным DCCP). Конечные точки **должны** отвечать на неизвестные опции Change пустой опцией Confirm (т. е., опцией Confirm, не содержащей данных), которая информирует конечную точку, находящуюся в состоянии CHANGING о том, что признак не был понят. Однако, если опция Change была помечена как обязательная (Mandatory), соединение **должно** быть разорвано (см. параграф 6.6.9).

При получении пустой опции Confirm для того или иного признака конечная точка в состоянии CHANGING **должна** вернуться в состояние STABLE, сохранив значение признака неизменным. В главе 15 предлагается использовать принятое по умолчанию значение для любых расширенных признаков, которые недоступны.

Некоторые признаки должны пониматься всеми реализациями DCCP (см. параграф 6.4). Конечной точке в состоянии CHANGING **следует** сбрасывать соединение (с передачей Reset Code 5, Option Error) при получении пустой опции Confirm для такого признака.

Поскольку опции Confirm генерируются лишь в ответ на получение опции Change, конечная точка никогда не должна получать опции Confirm, относящиеся к непонятным ей признакам. Независимо от этого, конечные точки **должны** игнорировать такие опции при их получении.

6.6.8. Некорректные опции

Конечная точка DCCP может получить опцию Change или Confirm для известного признака, содержащую список, в котором одно или несколько значений неизвестны данной точке. Некоторые (но не все) из таких опций являются некорректными в зависимости от относящегося к опции правила согласования. Примеры таких случаев даны ниже.

- Все признаки имеют ограниченный размер и опции с некорректным полем размера являются некорректными. Например, признак Ack Ratio принимает 16-битовые значения, поэтому корректная опция Confirm R(Ack Ratio) имеет размер 5.
- Некоторые несогласуемые признаки имеют ограничения по диапазону значений. Признак Ack Ratio принимает 2-байтовые, отличные от нуля целые числа, поэтому опция Change L(Ack Ratio, 0) никогда не может быть корректной. Отметим, что признаки с приоритетом сервера не ограничиваются по диапазону значений, поскольку неизвестные значения обрабатываются, как нечто само собой разумеющееся.
- Любая опция Confirm, выбирающая некорректное значение на основе двух списков предпочтений и соответствующего правила согласования, является некорректной.

Однако неожиданные опции Confirm (указывающие на неизвестные номера признаков или не относящиеся к текущему согласованию признаков) не являются некорректными, хотя и игнорируются получателем.

Конечная точка, получившая некорректную опцию Change, **должна** ответить на неё соответствующей пустой опцией Confirm. Конечная точка, получившая некорректную опцию Confirm, **должна** разорвать соединение с использованием Reset Code 5, Option Error.

6.6.9. Согласование обязательных признаков

Опции Change может предшествовать опция Mandatory (параграф 5.8.2). Опции Mandatory Change обрабатываются как обычные опции Change с единственным отличием в том, в перечисленных ниже случаях соединение должно разрываться с использованием Reset Code 6, Mandatory Failure вместо передачи ответной опции Confirm. Соединение **должно** разрываться, если:

- номер признака в опции Change непонятен;
- значение опции Change некорректно и в обычных условиях получатель вернул бы в ответ пустую опцию Confirm;

- для признаков с приоритетом сервера в двух списках предпочтений отсутствуют совпадающие элементы.

В остальных случаях соединение не разрывается. В частности, защита от нарушения порядка доставки может приводить к игнорированию опции Mandatory Change без сброса соединения.

Опции Confirm ведут себя одинаково, независимо от наличия флага Mandatory.

7. Порядковые номера

Протокол DCCP использует порядковые номера для упорядочивания пакетов, детектирования потерь и дубликатов, защиты от атак, а также для предотвращения полуоткрытых соединений и доставки очень старых пакетов. Каждый пакет содержит поле Sequence Number, а большинство пакетов включает также номер подтверждения - Acknowledgement Number.

Порядковые номера DCCP базируются на счёте числа пакетов. Т. е., значения полей Sequence Number, генерируемых каждой конечной точкой увеличиваются на 1 для каждого следующего пакета и используют арифметику с модулем 2^{48}. Даже пакеты DCCP-Ack, DCCP-Sync и прочие пакеты, не содержащие пользовательских данных, приводят к увеличению значений Sequence Number. Поскольку протокол DCCP не гарантирует доставки, в нем не используется механизма передачи повторов, но повтор передачи таких пакетов, как DCCP-Request, также ведёт к возрастанию значения Sequence Number. Это позволяет реализациям DCCP детектировать сетевые дубликаты, повторы и потерю подтверждения, что существенно отличается от поведения протокола TCP.

7.1. Переменные

Конечные точки DCCP поддерживают набор переменных для порядковых номеров каждого соединения.

ISS[1] - начальный порядковый номер, переданный этой конечной точкой.

Это значение совпадает с полем Sequence Number в первом, переданном через данное соединение, пакете DCCP-Request или CPD-Response.

ISR[2] - начальный порядковый номер, полученный от другой конечной точки.

Значение этой переменной совпадает со значением поля Sequence Number в первом, принятом через данное соединение, пакете DCCP-Request или CPD-Response.

GSS[3] - наибольший порядковый номер, переданный этой конечной точкой.

Максимальное значение переданного порядкового номера с учётом циклической арифметики.

GSR[4] - наибольший порядковый номер, полученный от удалённой точки.

Максимальное значение порядкового номера, полученное от другой точки в подтверждаемых пакетах (см. параграф 7.4).

GAR[5] - наибольший номер номер подтверждения, полученный от другой точки.

Максимальный номер подтверждения, полученный от другой точки в подтверждаемых пакетах, не относящихся к числу DCCP-Sync.

На основе этих примитивов поддерживаются также дополнительные переменные.

SWL[6] и **SWH**[7] - минимальное и максимальное значение окна порядковых номеров.

Эти параметры определяют границы корректного окна для порядковых номеров в принимаемых пакетах.

AWL[8] и **AWH**[9] - минимальное и максимальное значение окна номеров подтверждений.

Эти параметры определяют границы корректного окна для номеров подтверждений в принимаемых пакетах.

7.2. Начальные порядковые номера

Конечные точки устанавливают начальные порядковые номера в первых передаваемых пакетах DCCP-Request и CPD-Response. Начальные порядковые номера **должны** выбираться для предотвращения двух проблем:

- доставка старых пакетов, когда застрявшие в сети пакеты старых соединений соединений будут передаваться новому соединению с такими же адресами и номерами портов;
- атаки с предсказанием порядковых номеров, когда атакующий пытается угадать порядковые номера, которые могут использоваться в будущих соединениях [M85].

Эти проблемы похожи на аналогичные проблемы TCP и реализациям DCCP **следует** использовать принятую в TCP стратегию для предотвращения проблем [RFC793, RFC1948]. Остальная часть параграфа посвящена более детальному рассмотрению этой стратегии.

Для решения первой проблемы реализация протокола **должна** гарантировать, что начальные значения для данного квартета <source address, source port, destination address, destination port> не перекрываются с порядковыми номерами для недавнего соединения с таким же квартетом. Термин «недавний» в этом контексте относится к пакетам,

[1] Initial Sequence Number Sent.
[2] Initial Sequence Number Received.
[3] Greatest Sequence Number Sent.
[4] Greatest Sequence Number Received.
[5] Greatest Acknowledgement Number Received.
[6] Sequence Number Window Low.
[7] Sequence Number Window High.
[8] Acknowledgement Number Window Low.
[9] Acknowledgement Number Window High.

переданным в течение удвоенного максимального срока жизни сегмента (4 минуты). Реализации **должны** также гарантировать, что младшие 24 бита начального порядкового номера не перекрываются с младшими 24 битами недавних порядковых номеров (если реализация не предполагает отказ от использования коротких порядковых номеров, как описано в параграфе 7.6). Реализация, имеющая состояние для недавнего соединения с таким же квартетом, может явно выбрать подходящий начальный порядковый номер. В остальных случаях начальные номера могут выбираться на основе того или иного таймера типа 4-х микросекундного таймера, используемого TCP [RFC793]. Могут потребоваться два отдельных таймера, один из которых будет использоваться для старших 24 битов порядкового номера, а второй – для 24 младших битов номера.

Для решения второй проблемы реализация **должна** обеспечивать каждому квартету адресов и портов независимое пространство начальных порядковых номеров. Тогда при организации нового соединения не будет предоставляться никакой информации о начальных порядковых номерах других соединений того же хоста. В соответствии с [RFC1948] это достигается путём добавления криптографической хэш-функции и секретного значения к квартету номеров портов и адресов при выборе каждого начального порядкового номера. Для выбора секретного значения [RFC1948] рекомендует использовать комбинацию неких случайных данных [RFC4086], задаваемую администратором парольную фразу, IP-адрес конечной точки и время загрузки этой точки, но вполне достаточно и действительно случайных данных. Следует аккуратно относиться к смене секретного значения, поскольку такая смена будет изменять все пространства начальных порядковых номеров, что может привести к совпадению пространства начальных номеров для некого квартета с недавно переданными порядковыми номерами для того же квартета. Для предотвращения подобных проблем конечная точка может запоминать состояние «мёртвых» соединения для каждого квартета или устанавливать паузу продолжительностью 2MSL после смены секретного значения.

7.3. Пауза (Quiet Time)

Конечные точки DCCP, подобно конечным точкам TCP, должны с осторожностью относиться к организации соединений при загрузке хоста. В частности, **недопустимо** передавать пакеты, чьи порядковые номера близки к порядковым номерам пакетов, остававшихся в сети перед загрузкой. Простейшим способом выполнения этого правила для конечных точек DCCP является отказ от передачи каких-либо пакетов в течение периода MSL (2 минуты) после загрузки.

Другой механизм основан на запоминании недавних порядковых номеров до перезагрузки и резервирование 8 (или около того) старших битов начальных порядковых номеров для постоянно работающего счётчика, значение которого уменьшается на 2 при каждой перезагрузке хоста. Этот механизм требует запрета на использование коротких порядковых номеров (см. параграф 7.6.1.).

7.4. Номера подтверждений

Кумулятивные подтверждения не имеют смысла для протокола, не обеспечивающего гарантий доставки. Следовательно, в DCCP поле Acknowledgement Number имеет не такой смысл, как в TCP.

Принятые пакеты классифицируются, как подтверждаемые тогда и только тогда, когда их заголовок успешно обработан принимающим модулем DCCP. В терминах псевдокода, рассматриваемого в параграфе 8.5, полученный пакет становится подтверждаемым, когда принимающая сторона достигает этапа 8. Это означает, например, что все подтверждаемые пакеты имеют корректные значения контрольной суммы заголовка и порядкового номера. Передаваемое конечной точкой значение Acknowledgement Number **должно** совпадать со значением GSR[1] для передающей конечной точки на момент подтверждения для всех типов пакетов, за исключением DCCP-Sync и DCCP-SyncAck.

«Подтверждаемость» не связана с обработкой данных. Например, данные приложений из подтверждаемых пакетов могут отбрасываться в результате их повреждения или переполнения приёмного буфера. Опции Data Dropped при необходимости используются для передачи информации о таких событиях, позволяя механизмам контроля насыщения различать потери в сети и потери в конечной точке. Этот вопрос рассматривается в параграфах 11.4 и 11.7.

Номера подтверждений для пакетов DCCP-Sync и DCCP-SyncAck отличаются - поле Acknowledgement Number в пакете DCCP-Sync соответствует принятому пакету, который не обязан быть подтверждаемым (в частности, это может быть пакет восстановления синхронизации, опции которого не обрабатываются). Поле Acknowledgement Number пакета DCCP-SyncAck всегда соответствует подтверждаемому пакету DCCP-Sync и может иметь значение меньше GSR в случаях нарушения порядка доставки.

7.5. Корректность номеров и синхронизация

Любая конечная точка DCCP может получать пакеты, которые в действительности не относятся к текущему соединению. Например, из сети могут приходить старые пакеты или другая конечная точка может аварийно завершить работу, оставив полуоткрытое соединение.

DCCP, подобно TCP, использует проверку порядковых номеров для детектирования таких ситуаций. Пакеты, в которых порядковые номера и/или номера подтверждений выходят за пределы допустимого диапазона, называются пакетами с некорректными номерами и не обрабатываются как обычно.

В отличие от TCP, протокол DCCP требует использования механизма синхронизации для восстановления порядковых номеров при потере большого числа пакетов. Одна из конечных точек может в период потери передать так много пакетов, что когда один из таких пакетов достигнет другой конечной точки, та сочтёт его порядковый номер некорректным. Для восстановления синхронизации порядковых номеров используется согласование с помощью пакетов DCCP-Sync и DCCP-SyncAck.

7.5.1. Окна порядковых номеров и номеров подтверждений

Каждая конечная точка DCCP определяет окна корректности, которые являются подмножествами пространства порядковых номеров и номеров подтверждений. Эти окна соответствуют пакетам, которые конечная точка ожидает получить в течение нескольких последующих периодов кругового обхода. Окна Sequence Number и Acknowledgement

[1] Greatest Sequence Number Received – максимальный принятый порядковый номер. *Прим. перев.*

Number всегда содержат значения GSR и GSS, соответственно. Ширина окна контролируется признаками Sequence Window для двух полусоединений.

Окно корректности значений Sequence Number для пакетов от DCCP B представляет собой интервал номеров [SWL, SWH]. Это окно всегда содержит значение GSR для пакетов с корректными порядковыми номерами, принятых от DCCP B. Окно имеет ширину W пакетов, где W является значениям признака Sequence Window/B. Одна четверть номеров из окна (с округлением в меньшую сторону) имеет значения не более GSR, а три четверти номеров превышают GSR (эта асимметрия предполагает, что потеря множества пакетов в сети более вероятна, нежели существенное нарушение порядка их доставки).

```
некорректн.|    корректные порядковые номера     |некорректн.
<---------*|*===========*======================*|*--------->
      GSR -|GSR + 1 -   GSR                GSR +|GSR + 1 +
 floor(W/4)|floor(W/4)                ceil(3W/4)|ceil(3W/4)
           = SWL                           = SWH
```

Окно корректности номеров подтверждений для пакетов от DCCP B представляет собой интервал [AWL, AWH]. Верхняя граница окна (AWH) равна значению GSS (Greatest Sequence Number Sent – максимальный переданный порядковый номер) DCCP A; ширина окна W' представляет собой значение признака Sequence Window/A.

```
некорректн.|     корректные порядковые номера    |некорректн.
<---------*|*================================*|*--------->
   GSS - W'|GSS + 1 - W'                    GSS|GSS + 1
           = AWL                          = AWH
```

Значения SWL и AWL изначально задаются так, чтобы они были не меньше начальных принятых и переданных порядковых номеров, соответственно:

```
SWL := max(GSR + 1 - floor(W/4), ISR),
AWL := max(GSS + 1 - W', ISS).
```

Такое присвоение значений **должно** происходить только в начале соединения (долгоживущие соединения могут достигать максимального значения порядковых номеров и сбрасывать номера в 0, поэтому значения могут быть меньше ISR или ISS и такое присвоение **недопустимо** в этом случае).

7.5.2. Признак Sequence Window

Признак Sequence Window/A определяет ширину окна корректности порядковых номеров, используемого DCCP B, и ширину окна корректности номеров подтверждений, используемого DCCP A. Конечная точка DCCP A передаёт опцию Change L(Sequence Window, W) для уведомления DCCP B о том, что признак Sequence Window/A имеет значение W.

Sequence Window является признаком номер 3 и не требует согласования. Он принимает 48-битовые (6 байтов) значения, подобно порядковым номерам DCCP. Опции Change и Confirm для признаков Sequence Window имеют, следовательно, размер 9 байтов. Новые соединения начинаются со значения Sequence Window 100 для обеих конечных точек. Минимально допустимое значение признака Window - Wmin = 32, а максимальное - Wmax = 2^{46} - 1 = 70368744177663. Опции Change, предлагающие значение Sequence Window за пределами указанного диапазона, являются некорректными и **должны** обрабатываться подобающим образом.

Подходящее значение Sequence Window/A должно отражать число пакетов, которые DCCP предполагает находящимися в пути. Это может прогнозировать только DCCP A. Слишком малые значения повышают для конечной точки риск потери синхронизации в результате потери группы пакетов, а ещё меньшие значения могут сделать невозможным обмен данными независимо от наличия потерь. С другой стороны, слишком большие значения повышают риск захвата соединений (оценка этого риска проводится в параграфе 7.5.5). Одной из хороших рекомендаций может служить установка для Sequence Window значения, примерно в 5 раз превышающего максимальное число пакетов, которые предполагается получать за один период кругового обхода. Конечным точкам **следует** при необходимости передавать опции Change L(Sequence Window) в течение периода работы соединения. Кроме того, для конечной точки **недопустимо** постоянно передавать за один период кругового обхода число пакетов, превышающее значение Sequence Window для этой точки (т. е., для точки DCCP A **недопустимо** постоянно передавать более Sequence Window/A пакетов за период RTT).

7.5.3. Правила проверки корректности номеров

Корректность порядкового номера зависит от типа принятого пакета. Приведённая здесь таблица показывает проверки порядковых номеров и номеров подтверждений, применимые к каждому типу. Пакеты, прошедшие проверку считаются корректными с точки зрения нумерации. Многие из проверок относятся к окнам корректности порядковых номеров [SWL, SWH] и номеров подтверждений [AWL, AWH], определенным в параграфе 7.5.1.

Тип пакета	Проверка поряд. номера	Проверка номера подтвер.
DCCP-Request	SWL <= seqno <= SWH (*)	неприменимо
DCCP-Response	SWL <= seqno <= SWH (*)	AWL <= ackno <= AWH
DCCP-Data	SWL <= seqno <= SWH	неприменимо
DCCP-Ack	SWL <= seqno <= SWH	AWL <= ackno <= AWH
DCCP-DataAck	SWL <= seqno <= SWH	AWL <= ackno <= AWH
DCCP-CloseReq	GSR < seqno <= SWH	GAR <= ackno <= AWH
DCCP-Close	GSR < seqno <= SWH	GAR <= ackno <= AWH
DCCP-Reset	GSR < seqno <= SWH	GAR <= ackno <= AWH
DCCP-Sync	SWL <= seqno	AWL <= ackno <= AWH
DCCP-SyncAck	SWL <= seqno	AWL <= ackno <= AWH

(*) - проверка неприменима, если соединение находится в состоянии LISTEN или REQUEST.

В общем случае пакеты являются корректными с точки зрения нумерации, если порядковые номера и номера подтверждений попадают в соответствующие окна корректности [SWL, SWH] и [AWL, AWH]. Исключения из этих правил перечислены ниже:

- Поскольку пакеты DCCP-CloseReq, DCCP-Close и DCCP-Reset завершают соединение, они не могут иметь порядковых номеров, которые меньше или равны GSR, или номеров подтверждений, которые меньше GAR.

- Порядковые номера в пакетах DCCP-Sync и DCCP-SyncAck не подвергаются строгой проверке. Эти пакеты существуют, в частности, для того, чтобы конечные точки могли восстановить синхронизацию порядковых номеров. Строгая проверка порядковых номеров в этих пакетах препятствовала бы синхронизации номеров.

Мягкая проверка номеров для пакетов DCCP-Sync и DCCP-SyncAck позволяет продолжать работу после необычных событий типа аварийного завершения работы конечной точки или потери большого числа пакетов, но в отсутствие необычных событий (при нормальном обмене данными) такая мягкость не требуется. Поэтому реализациям DCCP **следует** использовать приведённые здесь более строгие проверки для активных соединений (соединение рассматривается как активное, если через него были получены корректные пакеты от другой конечной точки в течение трёх предыдущих периодов кругового обхода).

Тип пакета	Проверка поряд. номера	Проверка номера подтвер.
DCCP-Sync	SWL <= seqno <= SWH	AWL <= ackno <= AWH
DCCP-SyncAck	SWL <= seqno <= SWH	AWL <= ackno <= AWH

Наконец, конечная точка **может** применять приведённые здесь более строгие проверки для пакетов DCCP-CloseReq, DCCP-Close и DCCP-Reset, дополнительно снижающие вероятность успеха при атаках вслепую с использованием таких пакетов.

Поскольку эти проверки могут приводить к дополнительным издержкам на синхронизацию и замедлять закрытие соединения при потере пакетов, их следует рассматривать как экспериментальные.

Тип пакета	Проверка поряд. номера	Проверка номера подтвер.
DCCP-CloseReq	seqno == GSR + 1	GAR <= ackno <= AWH
DCCP-Close	seqno == GSR + 1	GAR <= ackno <= AWH
DCCP-Reset	seqno == GSR + 1	GAR <= ackno <= AWH

Отметим, что проверка корректности нумерации является лишь одной из проверок, используемых по отношению к полученным пакетам.

7.5.4. Обработка пакетов с некорректными номерами

При получении пакетов с некорректной нумерацией, конечные точки выполняют перечисленные ниже действия.

- Все пакеты DCCP-Sync и DCCP-SyncAck с некорректными номерами **должны** игнорироваться.

- Пакеты DCCP-Reset с некорректной нумерацией **должны** вызывать передачу в ответ пакета DCCP-Sync (возможно с ограничением скорости передачи). Пакеты отклика **должны** использовать новое значение Sequence Number и, таким образом, будут увеличивать значение GSS. Однако значение GSR изменяться не будет, поскольку принятый пакет имеет некорректную нумерацию. Значения Acknowledgement Number в откликах **должны** быть равны GSR.

- Все остальные пакеты с некорректными номерами **должны** вызывать передачу в ответ похожих пакетов DCCP-Sync, отличающихся лишь тем, что значение Acknowledgement Number **должно** совпадать со значением Sequence Number в пакете с некорректным номером.

При получении пакета DCCP-Sync с некорректным номером партнерская конечная точка (скажем, DCCP B) **должна** обновить свою переменную GSR и передать в ответ пакет DCCP-SyncAck. Номер подтверждения в пакете DCCP-SyncAck будет равен порядковому номеру в пакете DCCP-Sync, который может отличаться от GSR. При получении этого пакета DCCP-SyncAck, который будет иметь корректную нумерацию, поскольку он подтверждает пакет DCCP-Sync, конечная точка DCCP A будет обновлять свою переменную GSR и синхронизация номеров между конечными точками восстановится. Как исключение из этого правила конечная точка в состоянии REQUEST **должна** отвечать пакетом DCCP-Reset вместо DCCP-SyncAck. Это нужно для сброса полуоткрытого соединения DCCP A.

Для защиты от DoS-атак реализациям DCCP **следует** вводить ограничение скорости передачи пакетов DCCP-Sync в ответ на приём пакетов с некорректной нумерацией, чтобы передавалось не более 8 пакетов DCCP-Sync в секунду.

Конечным точкам DCCP **недопустимо** обрабатывать пакеты с некорректной нумерацией, за исключением, возможно, генерации пакетов DCCP-Sync. Например, **не допускается** обработка опций. Конечная точка **может** временно сохранять пакеты с некорректными номерами, если они могут позднее стать корректными - это может снижать воздействие потери множества пакетов за счёт доставки приложению большего числа пакетов. В частности, конечная точка **может** сохранять пакеты с некорректной нумерацией на время до 2 периодов кругового обхода. Если в течение срока сохранения соответствующее окно корректности порядковых номеров изменится так, что номера в пакетах станут корректными, конечная точка **может** снова обработать эти пакеты.

Отметим, что пакеты DCCP-Reset с некорректными номерами вызывают генерацию пакетов DCCP-Sync. Это происходит потому, что конечная точка в несинхронизированном состоянии (CLOSED, REQUEST, LISTEN) может не иметь информации, достаточной для генерации подходящего пакета DCCP-Reset с первой попытки. Например, если партнерская точка находится в состоянии CLOSED и получает пакет DCCP-Data, она не может определить корректное значение Sequence Number для включения в генерируемый пакет DCCP-Reset (поскольку пакет DCCP-Data не имеет поля Acknowledgement Number). Пакеты DCCP-Sync, генерируемые в ответ на такие некорректные пакеты сброса, обслуживаются как отвод (challenge) и содержат информацию, которой достаточно для генерации подходящего пакета DCCP-Reset. Однако новый пакет DCCP-Reset может содержать значение Reset Code, отличающееся от кода сброса в исходном пакете DCCP-Reset (новым кодом может быть, например, Reset Code 3, No Connection). Конечной точке **следует** использовать, по возможности, информацию из исходного пакета DCCP-Reset.

7.5.5. Атаки на порядковые номера

Порядковые номера и номера подтверждений являются основным способом защиты DCCP от атак. Атакующий, который не способен предсказать или угадать порядковые номера, не сможет легко манипулировать соединениями DCCP или захватывать их, поэтому требования типа осторожного выбора начальных порядковых номеров избавляют от большинства серьёзных атак.

Однако атакующий имеет возможность передавать множество пакетов со случайно выбранными значениями порядковых номеров и номеров подтверждений. Если одна из таких попыток увенчается успехом, атакующий сможет разорвать соединение или создать другие проблемы. Простейший вариант такой атаки описан ниже.

- Передача пакетов DCCP-Data со случайными значениями поля Sequence Number. Если один из таких пакетов попадает в окно корректных порядковых номеров, данные из этого пакета могут быть помещены в поток данных приложения.
- Передача пакетов DCCP-Sync со случайными значениями порядковых номеров и номеров подтверждений. Если один из таких пакетов попадёт в окно корректных номеров подтверждений, получатель будет соответствующим образом сдвигать своё окно корректных порядковых номеров, теряя синхронизацию с корректной конечной точкой (возможно, навсегда).

Для того, чтобы любая из таких атак оказалась успешной, атакующий должен подобрать корректные номера портов для отправителя и получателя. Кроме того, соединение будет неактивным с точки зрения атаки DCCP-Sync, если атакуемая сторона использует более строгие проверки для активных соединений, рекомендованные в параграфе 7.5.3.

Для оценки вероятности успеха атаки обозначим число пакетов, которые передаёт атакующий, как N, окно порядковых номеров - W, а размер порядковых номеров (24 или 48 битов) - L. Наилучшей стратегией атаки будет равномерное распределение порядковых номеров в используемых для атаки пакетах по всему пространству номеров. В этом случае вероятность попадания пакета в окно корректных порядковых номеров составляет $P = WN/2^L$.

Вероятность успеха атаки DCCP-Data при использовании коротких порядковых номеров составляет $P = WN/2^{24}$. При W = 100 атакующему потребуется передать более 83 000 пакетов, чтобы вероятность успеха достигла 50%. Для сравнения простейшая атака на TCP путём передачи пакетов SYN со случайными порядковыми номерами, которые будут приводить к сбросу соединения при попадании номера в окно, при размере окна W = 8760 (общепринятое значение по умолчанию) и L = 32 потребует более 245 000 пакетов для достижения вероятности успеха 50%.

Для быстрых соединений обычно используется большее значение W, что при фиксированном значении N повышает вероятность успешной атаки. Если эта вероятность становится недопустимо высокой при L = 24, конечной точке **следует** отказаться от использования коротких порядковых номеров с помощью признака Allow Short Sequence Numbers (см. параграф 7.6.1). Однако допустимый предел вероятности успешной атаки зависит от приложения. Некоторые приложения (например, те, которые могут работать при повреждении данных) являются достаточно устойчивыми к атакам путём вставки дополнительных данных.

Для атак DCCP-Sync L = 48, поскольку пакеты DCCP-Sync используют только длинные порядковые номера. Кроме того, вероятность успешной атаки снижается вдвое за счёт того, что корректна только половина пространства порядковых номеров. Следовательно, вероятность успеха таких атак существенно ниже. При большом значении W = 2000 пакетов для достижения 50% вероятности успеха потребуется более 10^{11} пакетов.

Атаки с использованием пакетов DCCP-Ack, DCCP-DataAck, DCCP-CloseReq, DCCP-Close и DCCP-Reset сложнее, поскольку требуют одновременно угадать порядковый номер и номер подтверждения. Вероятность успешной атаки для этих типов пакетов составляет $P = WXN/2^{2L}$, где W – размер окна порядковых номеров, X – размер окна номеров подтверждений, а N и L имеют такой же смысл, как раньше.

Поскольку атаки DCCP-Data при использовании коротких номеров сравнительно просты, протокол DCCP включает методы предотвращения сброса соединений и иных вредных воздействий в результате таких атак. В частности, все опции, обработка которых может вести к сбросу соединения игнорируются, если они получены в пакетах DCCP-Data.

7.5.6. Примеры обработки порядковых номеров

В первом примере DCCP A и DCCP B восстанавливают синхронизацию порядковых номеров после потери большого числа пакетов, приведшей к тому, что порядковые номера DCCP A вышли за пределы окна номеров DCCP B.

```
               DCCP A                                          DCCP B
               (GSS=1,GSR=10)                                  (GSS=10,GSR=1)
                           -->   DCCP-Data(seq 2)     XXX
                                   ...
                           -->   DCCP-Data(seq 100)   XXX
                           -->   DCCP-Data(seq 101)           -->  ???
                                                                 некорректный поряд. номер;
                                                                 передать Sync
                  OK       <--   DCCP-Sync(seq 11, ack 101)   <--
                                                                 (GSS=11,GSR=1)
                           -->   DCCP-SyncAck(seq 102, ack 11)   -->   OK
               (GSS=102,GSR=11)                                (GSS=11,GSR=102)
```

Во втором примере соединение DCCP восстанавливается после простой атаки вслепую.

```
DCCP A                                          DCCP B
(GSS=1,GSR=10)                                  (GSS=10,GSR=1)
          *Атакующий*  -->  DCCP-Data(seq 10^6)  -->  ???
                                                некорректный порядковый номер; передать Sync
   ???      <--   DCCP-Sync(seq 11, ack 10^6)  <-- некорр. номер подтверждения; игнорировать
(GSS=1,GSR=10)                                  (GSS=11,GSR=1)
```

Заключительный пример показывает восстановление из полуоткрытого соединения.

```
DCCP A                                          DCCP B
(GSS=1,GSR=10)                                  (GSS=10,GSR=1)
(Crash)
CLOSED                                              OPEN
REQUEST     -->   DCCP-Request(seq 400)         -->   ???
!!          <--   DCCP-Sync(seq 11, ack 400)    <--   OPEN
REQUEST     -->   DCCP-Reset(seq 401, ack 11)   -->   (Abort)
REQUEST                                             CLOSED
REQUEST     -->   DCCP-Request(seq 402)         -->   ...
```

7.6. Короткие порядковые номера

Порядковые номера DCCP имеют длину 48 битов. Такое большое пространство порядковых номеров защищает соединения DCCP от атак вслепую (таких, как вставка пакетов DCCP-Reset в существующие соединения). Однако в пакетах DCCP-Data, DCCP-Ack и DCCP-DataAck, которые составляют основную часть любого соединения DCCP, могут использоваться сокращённые номера, задаваемые только младшими 24 битами соответствующего порядкового номера или номера подтверждения. Принимающая конечная точка будет восстанавливать 48-битовую нумерацию, используя приведённый здесь псевдокод.

```
procedure Extend_Sequence_Number(S, REF)
   /* S - 24-битовый порядковый номер из заголовка пакета.
      REF - соответствующее 48-битовое значение:
      GSS, если S является номером подтверждения и GSR, если S - порядковый номер. */
   Установить REF_low := младшие 24 бита REF
   Установить REF_hi  := старшие 24 бита REF
   Если REF_low (<) S          /* циклическое сравнение с модулем 2^24 */
        И S |<| REF_low,       /* обычное, нециклическое сравнение */
      Возвратить (((REF_hi + 1) mod 2^24) << 24) | S
   Иначе, если S (<) REF_low И REF_low |<| S,
      Возвратить (((REF_hi - 1) mod 2^24) << 24) | S
   Иначе,
      Возвратить (REF_hi << 24) | S
```

Два разнотипных сравнения в первом условии позволяют детектировать переход через границу пространства порядковых номеров в младших битах. Циклическое сравнение REF_low (<) S возвращает позитивный результат тогда и только тогда, когда значение (S - REF_low), вычисленное с использованием арифметики дополнения до 2 и представленное в виде целого числа без знака, не превышает 2^{23} (mod 2^{24}). В таких случаях старшие биты инкрементируются или декрементируются подобающим образом.

7.6.1. Признак Allow Short Sequence Numbers

Конечные точки могут требовать, чтобы во всех пакетах использовались длинные порядковые номера, оставляя для признака Allow Short Sequence Numbers принятое по умолчанию значение 0. Это может снизить риск вставки неуместных данных в соединение. DCCP A передаёт опцию Change L(Allow Short Seqnos, 1) для индикации своего желания передавать пакеты с короткими порядковыми номерами.

Признак Allow Short Sequence Numbers имеет номер 2 и относится к признакам с приоритетом сервера. Признак принимает однобайтовые логические значения. Если Allow Short Seqnos/B имеет нулевое значение, для DCCP B **недопустимо** передавать пакеты с короткими порядковыми номерами, а точка DCCP A **должна** игнорировать все полученные пакеты с короткими порядковыми номерами. Значения признака, превышающие 1, являются резервными. Новые соединения организуются с Allow Short Sequence Numbers = 0 для обеих конечных точек.

7.6.2. Когда не следует использовать короткие порядковые номера

Использование коротких порядковых номеров снижает скорость, которой можно безопасно достигнуть для соединения DCCP, а также повышает риски иных атак, включая вставку данных вслепую. Для очень скоростных соединений DCCP и соединений с большими окнами порядковых номеров (параграф 7.5.2) **не следует** использовать короткие порядковые номера в пакетах данных. Риск, связанный с атаками, был рассмотрен выше. Здесь мы рассмотрим скоростные ограничения.

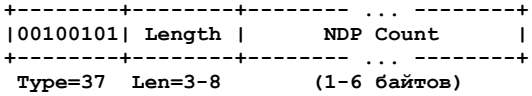
Механизм проверки корректности нумерации предполагает, что из сети не приходит очень старых данных. В частности, предполагается, что в сети должны быть отброшены все пакеты в момент достижения верхней границы пространства порядковых номеров и начала новой нумерации с нуля. Это ограничивает максимальную скорость соединений, которая может быть достигнута без опаски. Предположим, что максимальное время жизни сегмента равно MSL, P определяет средний размер пакета DCCP в битах, а L – размер порядкового номера (24 или 48 битов). Тогда максимальная скорость соединения будет равна

$$R = P*(2^L)/2MSL$$

Для принятого по умолчанию значения MSL (2 минуты), 1500-байтовых пакетов DCCP и коротких порядковых номеров максимальная безопасная скорость будет составлять приблизительно 800 Мбит/с. Хотя значение 2 минуты является очень большим для MSL, в любой сети, где может достигаться подобная скорость для таких мелких пакетов, использование длинных порядковых номеров при прочих равных условиях позволяет безопасно повышать скорость до 14 петабит/с[1].

7.7. Опция NDP Count и детектирование потери данных приложения

Порядковые номера DCCP увеличиваются на 1 для каждого следующего пакета, включая пакеты, не содержащие данных приложения. Это позволяет использовать порядковые номера DCCP для детектирования потери пакетов в сети (но не для детектирования потери данных приложения). Опция NDP[2] Count сообщает размер каждого блока потерянных в сети пакетов, не содержащих данных приложения. Это даёт приёмной стороне DCCP возможность надёжного определения случаев, когда потерянный блок пакетов включает данные приложения.



Если для передающей точки DCCP признак Send NDP Count имеет значение 1 (см. ниже), эта точка **должна** передавать опцию NDP Count в каждом пакете, который непосредственно следует за пакетом, не содержащим данных. К числу пакетов, не содержащих данных приложения относятся DCCP-Ack, DCCP-Close, DCCP-CloseReq, DCCP-Reset, DCCP-Sync и DCCP-SyncAck. Остальные типы пакетов, а именно DCCP-Request, CPD-Response, DCCP-Data и DCCP-DataAck рассматриваются как пакеты данных, хотя пакеты DCCP-Request и CPD-Response могут и не содержать данных приложения.

[1]Приставка “пета” означает 10^{15}. *Прим. перев.*
[2]non-data packet - пакеты, не содержащие данных приложения. *Прим. перев.*

Значение, хранящееся в NDP Count, равно числу последовательных пакетов без данных в группе пакетов, непосредственно предшествующих данному пакету. Пакеты без опции NDP Count рассматриваются как пакеты с NDP Count = 0.

Опция NDP Count может содержать от 1 до 6 байтов данных. **Следует** использовать минимальный размер поля данных опции, которого достаточно для того, чтобы поместить значение NDP Count.

С помощью опции NDP Count получатель может точно сказать лишь наличие в блоке потерянных пакетов по крайней мере одного пакета данных. Например, получатель не всегда может сказать, были ли в числе потерянных пакеты, не содержащие данных.

7.7.1. Использование NDP Count

Предположим, что K последовательных порядковых номеров отсутствуют в результате потери некого блока пакетов и признак Send NDP Count включён. В этом случае можно говорить, что блок потерянных пакетов включал какие-либо данные, если пакет, следующий за потерянным блоком, содержит значение опции NDP Count, которое не меньше K.

Предположим для примера, что конечная точка передала последовательность пакетов, не содержащих данные (Nx), и пакетов данных (Dx).

```
N0  N1  D2  N3  D4  D5  N6  D7  D8  D9  D10 N11 N12 D13
```

Для этих пакетов значения опции NDP Count будут:

```
N0  N1  D2  N3  D4  D5  N6  D7  D8  D9  D10 N11 N12 D13
-   1   2   -   1   -   -   1   -   -   -   -   1   2
```

Значения NDP Count не будет полезной для приложений, поддерживающих собственный механизм нумерации в заголовках своих пакетов.

7.7.2. Признак Send NDP Count

Признак Send NDP Count позволяет конечным точкам DCCP согласовать необходимость использования опции NDP Count в пакетах. DCCP A передаёт опцию Change R(Send NDP Count, 1), чтобы просить DCCP B передавать в заголовках опции NDP Count.

Признак Send NDP Count имеет номер 7 и выбирается с приоритетом сервера. Признак принимает однобайтовые логические значения. Конечная точка DCCP B **должна** передавать опции NDP Count, как описано выше, если Send NDP Count/B = 1, хотя она **может** передавать опции NDP Count и в случае Send NDP Count/B = 0. Значения признака, превышающие 1, являются резервными. Новые соединения организуются с Send NDP Count = 0 для обеих точек.

8. Обработка событий

В этой главе описываются смены состояний DCCP и порядок передачи пакетов. Отметим, что согласование признаков осуществляется параллельно со сменами состояний соединения, описанными здесь.

8.1. Организация соединения

Фаза организации соединения DCCP включает трехэтапное согласование - клиент передаёт начальный запрос DCCP-Request, сервер отвечает на него пакетом CPD-Response и клиент в заключение передаёт серверу пакет подтверждения (обычно DCCP-Ack или DCCP-DataAck). Клиент переходит из состояния REQUEST в состояние PARTOPEN, а потом в состояние OPEN; сервер переходит из состояния LISTEN сначала в состояние RESPOND, а потом в состояние OPEN.

```
   Состояние клиента                          Состояние сервера
       CLOSED                                       LISTEN
1.     REQUEST    -->       Request         -->
2.                <--       Response        <--     RESPOND
3.     PARTOPEN   -->     Ack, DataAck       -->
4.                <--  Data, Ack, DataAck   <--     OPEN
5.     OPEN       <->  Data, Ack, DataAck   <->     OPEN
```

8.1.1. Запрос клиента

Когда клиент решает организовать соединения, он переходит в состояние REQUEST, выбирает начальный порядковый номер (параграф 7.2) и передаёт пакет DCCP-Request, используя выбранный порядковый номер для данного сервера.

Пакеты DCCP-Request в общем случае включают опции согласования признаков для различных параметров соединения, такие, как идентификаторы механизмов контроля насыщения для каждого из полусоединений. Эти пакеты могут также включать данные приложения, но клиенту следует принимать во внимание, что сервер может не принять такие данные.

Клиенту в состоянии REQUEST **следует** использовать таймер для экспоненциального увеличения интервала передачи повторных пакетов DCCP-Request при отсутствии отклика от сервера. Первый повтор следует передавать приблизительно через 1 секунду и далее экспоненциально увеличивать интервал повтора не менее, чем до 64 секунд. Конечная точка может также использовать стратегию передачи повторов, принятую для пакетов TCP SYN. В каждом новом пакет DCCP-Request порядковый номер **должен** увеличиваться на 1 и все пакеты должны содержать такие же значения в поле Service Code и поле данных приложения, как и в исходном запросе DCCP-Request.

Клиент может прервать передачу запросов DCCP-Request по истечении того или иного времени (например, 3 минут). В таких случаях **следует** передать серверу пакет DCCP-Reset с Reset Code 2, Aborted для сброса состояния сервера в тех случаях, когда один или несколько запросов всё-таки были доставлены серверу. Клиент в состоянии REQUEST никогда не получает начальный порядковый номер от своего партнёра, поэтому в пакете DCCP-Reset должно быть установлено Acknowledgement Number = 0.

Клиент переходит из состояния REQUEST в состояние PARTOPEN при получении от сервера пакета CPD-Response.

8.1.2. Коды сервиса

Каждый пакет DCCP-Request содержит 32-битовый код сервиса, определяющий службу прикладного уровня, к который пытается подключиться клиент. Значения Service Code должны соответствовать прикладным службам и протоколам. Примерами могут служить коды сервиса для организации соединений SIP или аудио-сервиса RTP. Промежуточные устройства могут использовать поля Service Code для идентификации приложений, работающих через нестандартный порт (предполагается, что заголовок DCCP не зашифрован).

Конечные точки **должны** связывать значение Service Code с каждым сокетом DCCP, открытым активно или пассивно. Приложения, в общем случае, будут предоставлять значения Service Code. Каждый активный сокет **должен** иметь в точности одно значение Service Code. Пассивные сокеты **могут** (по выбору приложения) связываться с несколькими значениями кодов сервиса. Это позволяет множеству приложений или разным версиям одного приложения прослушивать один порт с использованием значения Service Code для выбора нужного приложения. Если пакет DCCP-Request содержит значение Service Code, которое не соответствует ни одному из кодов сервиса для данного порта, сервер **должен** отвергнуть запрос, передавая пакет DCCP-Reset с Reset Code 8, Bad Service Code. Промежуточное устройство также **может** передать такой пакет DCCP-Reset в ответ на запросы, значение Service Code в которых представляется неподходящим.

Значения Service Code не предназначены для исключительного использования DCCP и распределяются агентством IANA. В соответствии с правилами [RFC2434] большинство значений Service Code выделяется в порядке поступления запросов[1] с соблюдением приведённых здесь рекомендаций.

- Значения Service Code выделяются по одному или небольшими блоками. Для выделения значения Service Code **требуется** краткое описание сервиса на английском языке, но спецификаций, предложенных стандартов или чего-то иного не требуется. IANA поддерживает связь значений Service Code с соответствующими описаниями.
- Пользователи запрашивают конкретные значения кода сервиса. Мы предлагаем представлять код запрашиваемого сервиса с использованием формата «SC:», описанного ниже. Таким образом Frobodyne Plotz Protocol может соответствовать Service Code = 17178548426 или «SC:fdpz». Каноническая интерпретация поля Service Code является числовой.
- Для значений Service Code, байты которых входят в набор {32, 45-57, 65-90}, при выделении значений используется процедура Specification Required[2] (т. е., такие значения Service Code используются для международных стандартов, спецификаций со статусом standards-track, IETF и в иных случаях). Указанный набор содержит символы цифр, пробела, знака подчёркивания, '-', '.' и '/' в коде ASCII.
- Значения Service Code со старшим байтом 63 (ASCII-код символа ?) зарезервированы для частного использования (Private Use).
- Значение Service Code = 0 представляет отсутствие значимого кода обслуживания и его **недопустимо** выделять для иного применения.
- Значение 4294967295 обозначает некорректный код обслуживания. Серверы **должны** отвергать все пакеты DCCP-Request с таким значением Service Code, передавая в ответ на них пакет DCCP-Reset с Reset Code 8, Bad Service Code.

Эта схема распределения значений Service Code основана на распределении 4-байтовых идентификаторов ресурсов Macintosh, блоков PNG, а также таблиц TrueType и OpenType.

В текстовом представлении рекомендуется указывать значения Service Code в одной из трёх форм с префиксом (в коде ASCII) SC, за которым следует двоеточие «:» или знак равенства «=». Ниже описана интерпретация этих форм.

SC:

Показывает код сервиса, представляемый с использованием подмножества символов ASCII. После двоеточия указывается от 1 до 4 символов, которые могут быть буквами, цифрами и знаками «-_+.*/?@» (не включая символы кавычек). Десятичные коды этих символов относятся к набору {42-43, 45-57, 63-90, 95, 97-122}. Значение Service Code рассчитывается путём дополнения строки пробелами (символ с кодом 32) справа до 4 байтов и интерпретации полученной строки как 32-битового целого числа со старшим байтом в начале.

SC=

Задаёт значение Service Code в форме десятичного числа. После знака равенства может указываться любое количество десятичных цифр, которое задаёт код обслуживания. Значение Service Code = 4294967294 является некорректным.

SC=x или SC=X

Задаёт значение Service Code в форме шестнадцатеричного числа. После символа «x» или «X» может указываться любое количество шестнадцатеричных цифр (регистр символов не имеет значения), задающих код обслуживания. Значение Service Code = 4294967294 является некорректным.

Таким образом, Service Code со значением 1717858426 можно представить в форме SC:fdpz, SC=1717858426 или SC=x6664707A.

8.1.3. Отклик сервера

Во второй фазе трехэтапного согласования сервер переходит из состояния LISTEN в состояние RESPOND и передаёт клиенту сообщение CPD-Response. В этой фазе сервер зачастую будет задавать признаки, которые он хочет использовать, из числа запрошенных клиентом признаков или в дополнение к ним. Среди опций указывается и механизм контроля насыщения, который сервер предполагает использовать.

Сервер **может** ответить на пакет DCCP-Request пакетом DCCP-Reset, означающим отказ в соединении. Подходящие для этого случая значения Reset Code включают 7, Connection Refused (когда значение Destination Port в пакете DCCP-Request не соответствует порту DCCP, открытому для прослушивания), 8, Bad Service Code (значение Service Code в пакете DCCP-Request не соответствует коду сервиса, зарегистрированному для порта) и 9, Too Busy (когда сервер

[1]Процедура «First Come First Served».
[2]Требуется спецификация.

слишком занят и не может ответить на запрос). Серверу **следует** ограничивать скорость, с которой могут генерироваться пакеты отказа (например, не более 1024 отказов в секунду).

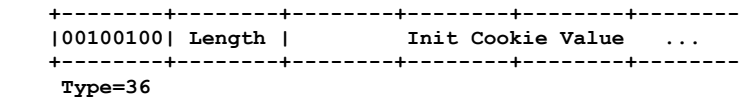
Серверу **не следует** повторять передачу пакетов CPD-Response, поскольку при отсутствии отклика клиент будет повторно передавать пакет DCCP-Request (отметим, что «повторный» пакет DCCP-Request будет отличаться от исходного, по крайней мере, порядковым номером, что позволяет серверу отличать повтор пакетов от их дублирования в сети). Сервер будет детектировать принадлежность повторных пакетов DCCP-Request к существующему соединению, используя номера портов отправителя и получателя. Каждый корректный пакет DCCP-Request, принятый сервером в состоянии RESPOND, **должен** вызывать генерацию нового ответного пакета CPD-Response. В каждом новом пакете CPD-Response поле Sequence Number **должно** увеличиваться на 1, а пакет **должен** включать те же данные приложения, которые были включены в исходный пакет CPD-Response.

Для сервера **недопустимо** принимать более одного элемента данных приложения в пакете DCCP-Request для каждого соединения. В частности, для пакета CPD-Response, передаваемого в ответ на повторный пакет DCCP-Request с данными приложения, **следует** устанавливать опцию Data Dropped, которая говорит об отбрасывании данных из повторного пакета DCCP-Request с Drop Code 0, Protocol Constraints. Исходный пакет DCCP-Request также **следует** указывать в опции Data Dropped как Normal Block (если сервер принял данные или данных не было) или Drop Code 0, Drop Block (если сервер отказался от приёма. этих данных).

Опции Data Dropped и Init Cookie полезны, в частности, для пакетов CPD-Response (параграфы 11.7 и 8.1.4).

Сервер переходит из состояния RESPOND в состояние OPEN при получении от клиента корректного пакета DCCP-Ack, завершающего трехэтапное согласование. Сервер **может** также перейти из состояния RESPOND в состояние CLOSED по тайм-ауту не менее 4MSL (8 минут). В таких случаях серверу **следует** передавать клиенту пакет DCCP-Reset с Reset Code 2, Aborted для сброса состояния клиента.

8.1.4. Опция Init Cookie



Опция Init Cookie позволяет серверу DCCP предотвратить смену состояния до завершения процесса трехэтапного согласования, подобно TCP SYN cookies [SYNCOOKIES]. Сервер помещает Service Code, номер своего порта и все опции, полученные в пакетах DCCP-Request и CPD-Response в специальную переменную cookie. Обычно значение cookie шифруется с использованием ключа, известного только серверу, и включает криптографическую контрольную сумму или «магическое значение», которые позволяют проверить корректность дешифровки. Когда сервер получает назад значение cookie в отклике от клиента, он может дешифровать это значение и установить нужное состояние. До этого сервер может сохранять состояние LISTEN.

Опцию Init Cookie **недопустимо** передавать в пакетах DCCP-Request или DCCP-Data. Любые опции Init Cookie, полученные в пакетах DCCP-Request и DCCP-Data или после организации соединения (состояние соединения >= OPEN), должны игнорироваться. Сервер **может** включить опции Init Cookie в свой пакет CPD-Response. В этом случае клиент **должен** возвращать такие же опции Init Cookie с соблюдением их порядка во всех последующих пакетах DCCP, пока один из таких пакетов не будет подтверждён (завершение трехэтапного согласования) или соединение не будет разорвано. В результате этого для клиента **недопустимо** использование пакетов DCCP-Data до завершения процедуры трехэтапного согласования или сброса соединения. Опции Init Cookie в пакете от клиента **должны** в точности соответствовать опциям, полученным в пакете DCCP-Request, указанном значением Acknowledgement Number в пакете клиента. Серверу **следует** использовать для Init Cookie такой формат, который допускает проверку на предмет подделки. Серверу **следует** отвечать на поддельные опции Init Cookie сбросом соединения с Reset Code 10, Bad Init Cookie.

В спецификации протокола не требуется точное описание реализации Init Cookie, поскольку значение Init Cookie непонятно клиенту и проблем взаимодействия возникнуть не может. В качестве примера может служить шифрование (с использованием закрытого ключа) порядковых номеров и номеров подтверждений, значения Service Code, номеров портов, всех опций из пакета DCCP-Request и соответствующего ему пакета CPD-Response, случайного значения salt и «магического значения». При получении возвращённого клиентом значения Init Cookie сервер будет дешифровать его, проверять корректность с использованием «магического значения», порядковых номеров и номеров портов. Если полученное значение корректно, сервер будет открывать соответствующий сокет с использованием опций.

Каждая отдельная опция Init Cookie может включать до 253 байтов данных и сервер может включать в пакет более одной опции Init Cookie для увеличения объёма информации.

8.1.5. Завершение согласования

Получив от сервера пакет CPD-Response, клиент переходит из состояния REQUEST в состояние PARTOPEN и завершает процедуру трехэтапного согласования, передавая серверу пакет DCCP-Ack. Клиент остаётся в состоянии PARTOPEN до тех пор, пока он не обретёт уверенность в том, что сервер получил тот или иной пакет, переданный клиентом из состояния PARTOPEN (начальный пакет DCCP-Ack или следующие за ним пакеты). Клиенты, желающие передавать данные из состояния PARTOPEN, **должны** делать это с помощью пакетов DCCP-DataAck, а не DCCP-Data. Это связано с тем, что пакеты DCCP-Data не включают номера подтверждения, поэтому сервер не может понять из пакета DCCP-Data, получил ли клиент его отклик CPD-Response. Более того, если пакет CPD-Response включает опцию Init Cookie, значение Init Cookie **должно** включаться во все пакеты, передаваемые из состояния PARTOPEN.

Один пакет DCCP-Ack, переданный при переходе в состояние PARTOPEN, может, естественно, быть потерян в сети. Клиенту **следует** обеспечить проверку доставки таких пакетов. Предпочтительным механизмом является использование таймера (приблизительно на 200 мсек), запускаемого при передаче каждого пакета из состояния PARTOPEN. Если отсчёт для таймера закончен, а клиент сохраняет состояние PARTOPEN, ему следует генерировать новый пакет DCCP-Ack и и заново запустить таймер. Если состояние PARTOPEN сохраняется дольше 4MSL (8 минут), **следует** сбросить соединение с Reset Code 2, Aborted.

Клиент переходит из состояния PARTOPEN в состояние OPEN при получении от сервера корректного пакета, отличного от CPD-Response, DCCP-Reset или DCCP-Sync.

8.2. Передача данных

В основной части соединения, связанной с обменом данными, клиент и сервер находятся в состоянии OPEN.

DCCP A передаёт пакеты DCCP-Data и DCCP-DataAck точке DCCP B в результате событий прикладного уровня на хосте A. Для этих пакетов используется контроль насыщения с помощью механизма CCID для полусоединения от A к B. Для пакетов же DCCP-Ack, передаваемых DCCP A, контроль насыщения осуществляется механизмом CCID для полусоединения от B к A. В общем случае DCCP A будет добавлять подтверждающую информацию в пакеты DCCP-Data, когда это возможно, создавая пакеты DCCP-DataAck. Пакеты DCCP-Ack используются в тех случаях, когда нет данных для передачи от DCCP A к DCCP B или состояние насыщения для CCID от A к B не позволяет передавать данные.

Пакеты DCCP-Sync и DCCP-SyncAck также могут использоваться в фазе обмена данными. Некоторые ситуации, вынуждающие генерировать пакеты DCCP-Sync, описаны в параграфе 7.5. Важным отличием пакетов DCCP-Sync от остальных типов пакетов является то, что получение DCCP-Sync вызывает немедленную отправку подтверждения. При получении корректного пакета DCCP-Sync конечная точка DCCP **должна** незамедлительно сгенерировать и передать отклик DCCP-SyncAck (подчиняясь всем ограничениям скорости передачи). Поле Acknowledgement Number пакета DCCP-SyncAck **должно** совпадать со значением поля Sequence Number в принятом пакете DCCP-Sync.

Отдельные реализации DCCP могут инициировать согласование признаков только после перехода в состояние OPEN. В этих случаях передача данных может оказаться недопустимой в течение некоторого времени. Данные, полученные в этот период, **следует** отбрасывать и уведомлять об этом с помощью опции Data Dropped Drop Block с Drop Code 0, Protocol Constraints (см. параграф 11.7).

8.3. Разрыв соединения

При разрыве соединений DCCP используется согласование, включающее необязательный пакет DCCP-CloseReq, а также пакеты DCCP-Close и DCCP-Reset. Сервер переходит из состояния OPEN (возможно, через промежуточное состояние CLOSEREQ) в состояние CLOSED; клиент переходит из состояния OPEN в состояние CLOSING, затем в состояние TIMEWAIT и, по истечении периода ожидания 2MSL (4 минуты), в состояние CLOSED.

Последовательность пакетов DCCP-CloseReq, DCCP-Close, DCCP-Reset используется в тех случаях, когда сервер хочет закрыть соединение, не удерживая себя в состоянии TIMEWAIT:

```
  Состояние клиента                          Состояние сервера
     OPEN                                    OPEN
1.             <--       CloseReq       <--   CLOSEREQ
2.   CLOSING   -->        Close          -->
3.             <--        Reset         <--   CLOSED (LISTEN)
4.   TIMEWAIT
5.   CLOSED
```

Более короткая последовательность используется для разрыва соединения по инициативе клиента.

```
  Состояние клиента                          Состояние сервера
     OPEN                                    OPEN
1.   CLOSING   -->        Close          -->
2.             <--        Reset         <--   CLOSED (LISTEN)
3.   TIMEWAIT
4.   CLOSED
```

Если сервер решить сохранить у себя состояние TIMEWAIT, используется последовательность:

```
  Состояние клиента                          Состояние сервера
     OPEN                                    OPEN
1.             <--        Close          <--   CLOSING
2.   CLOSED    -->        Reset          -->
3.                                           TIMEWAIT
4.                                           CLOSED (LISTEN)
```

Во всех случаях получатель пакета DCCP-Reset удерживает соединение в состоянии TIMEWAIT. Как и в TCP, состояние TIMEWAIT сохраняется конечной точкой в течение 2MSL (4 минуты), после чего соединение закрывается. Период ожидания гарантирует, что не будет организовано новое соединение с такими же адресами и номерами портов, пока в сети могут оставаться пакеты разорванного соединения.

Далее описывается процедура согласования при разрыве соединения. Получатель корректного пакета DCCP-CloseReq **должен** ответить на него пакетом DCCP-Close. Получатель корректного пакета DCCP-Close **должен** передать в ответ пакет DCCP-Reset с Reset Code 1, Closed. Получатель корректного пакета DCCP-Reset, который одновременно является отправителем пакета DCCP-Close (и, возможно, получателем пакета DCCP-CloseReq), будет сохранять состояние TIMEWAIT для закрываемого соединения.

Пакет DCCP-Reset завершает каждое соединение DCCP, независимо от того, является закрытие соединения нормальным (по запросу приложения с Reset Code 1, Closed) или аварийным. В отличие от протокола TCP, использующего два разных механизма разрыва соединений (FIN и RST), DCCP одинаково закрывает соединения во всех случаях. Это сделано по той причине, что некоторые аспекты разрыва соединения совершенно не зависят от того, каким способом (нормально или аварийно) завершилось соединение. Например, конечной точке, получившей корректный пакет DCCP-Reset, **следует** сохранить для соединения состояние TIMEWAIT. Процессоры, которые должны отличать нормальное закрытие соединения от аварийного, могут проверить значение Reset Code. Реализации DCCP обычно переходят в состояние CLOSED после передачи пакета DCCP-Reset.

Конечные точки, находящиеся в состояниях CLOSEREQ и CLOSING, **должны** повторять передачу пакетов DCCP-CloseReq и DCCP-Close, соответственно, пока не выйдут из этого состояния. Для таймера повтора передачи следует устанавливать начальное значение так, чтобы выйти за 2 периода кругового обхода и повторять передачу не реже одного раза в течение 64 секунд, если не получен требуемый отклик.

Передавать пакеты DCCP-CloseReq и переходить в состояние CLOSEREQ разрешается только серверу. При получении пакета DCCP-CloseReq сервер **должен** ответить пакетом DCCP-Sync или игнорировать пакет DCCP-CloseReq.

Пакеты DCCP-Data, DCCP-DataAck и DCCP-Ack, полученные в состоянии CLOSEREQ или CLOSING **можно** обрабатывать или игнорировать.

8.3.1. Аварийное завершение

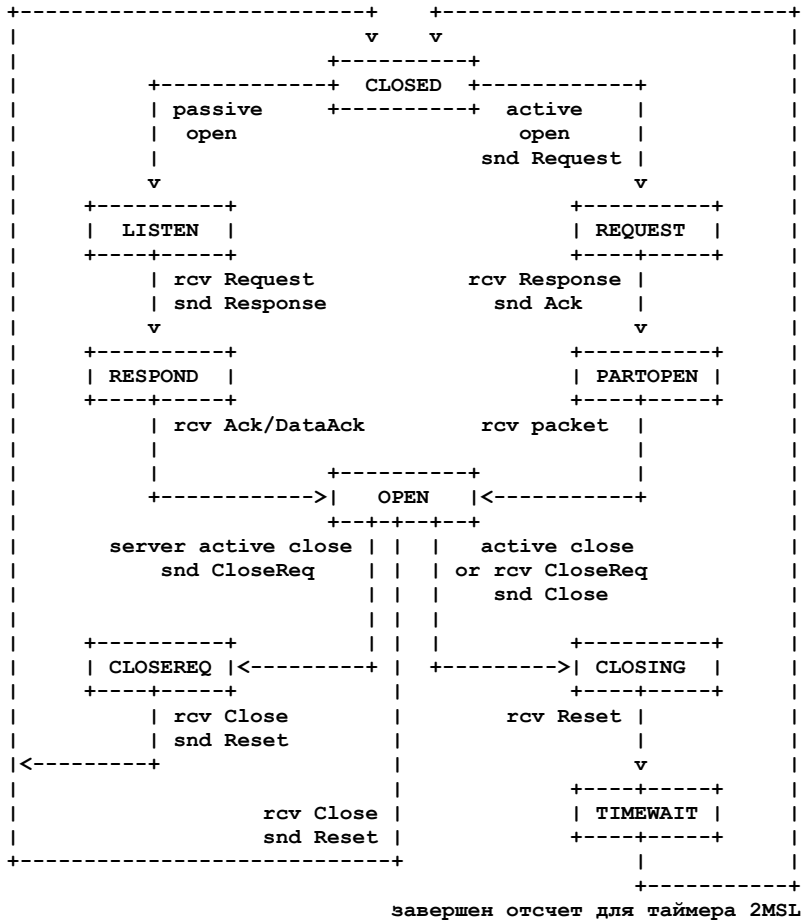
Конечные точки DCCP генерируют пакеты DCCP-Reset для аварийного разрыва соединения; генерация пакетов DCCP-Reset возможна из любого состояния. Пакеты сброса, переданные из состояния CLOSED, LISTEN или TIMEWAIT, используют Reset Code 3, No Connection, если явно не указано иное. Пакеты сброса, переданные из состояния REQUEST или RESPOND, используют Reset Code 4, Packet Error, если явно не указано иное.

Конечным точкам DCCP, находящимся в состоянии CLOSED, LISTEN или TIMEWAIT, может потребоваться генерация пакета DCCP-Reset в ответ на полученный от партнёра пакет. Поскольку для этих состояний нет соответствующих переменных с порядковыми номерами, порядковый номер и номер подтверждения для пакета DCCP-Reset (R) берутся из полученного пакета P, как показано ниже.

1. Если существует P.ackno, устанавливается R.seqno := P.ackno + 1. Иначе R.seqno := 0.
2. Устанавливается R.ackno := P.seqno.
3. Если пакет использует короткий порядковый номер (P.X == 0), для старших 24 битов R.seqno и R.ackno устанавливается значение 0.

8.4. Диаграмма состояний DCCP

Большинство смен состояний, описанных выше, можно показать на диаграмме состояний. Рисунок является лишь иллюстрацией, а в качестве определений следует использовать текст, приведённый в параграфе 8.5 и других местах спецификации. Например, имеются не показанные на рисунке переходы из всех состояний кроме CLOSED в состояние TIMEWAIT, связанные с получением корректного пакета DCCP-Reset.



8.5. Псевдокод

В этом параграфе представлен алгоритм, описывающий этапы обработки, через которые должна пройти конечная точка DCCP при получении пакета. Реализация DCCP не обязана в точности следовать описанному здесь алгоритму, но любая реализация **должна** генерировать видимые эффекты, в точности соответствующие приведенному здесь псевдокоду за исключением тех случаев, когда в данном документе явно разрешено иное поведение.

Принятый пакет обозначается P, сокет - S. Переменными сокета являются:

S.SWL - нижний предел окна порядковых номеров;

S.SWH - верхний предел окна порядковых номеров;

S.AWL - нижний предел окна номеров подтверждений;

S.AWH - верхний предел окна номеров подтверждений;

S.ISS –– начальный порядковый номер в переданном пакете;

S.ISR - начальный порядковый номер в принятом пакете;

S.OSR - порядковый номер в первом пакете, принятом в состоянии OPEN;

S.GSS - максимальный порядковый номер в переданных пакетах;

S.GSR - максимальный порядковый номер в принятых пакетах;

S.GAR - максимальный корректный номер подтверждения, принятый в пакетах, отличных от Sync; инициализируется значение S.ISS

Операция Send packet (передача пакета) всегда использует значение S.GSS и увеличивает его на 1.

Этап 1: Базовая проверка заголовка

```
/* Этот этап проверяет корректность формата пакетов. Не прошедшие проверку пакеты
   игнорируются без передачи в ответ пакетов Reset */
Если пакет короче 12 байтов, он отбрасывается с возвратом управления.
Если тип P.type непонятен, пакет отбрасывается с возвратом управления.
Если P.Data Offset меньше размера фиксированного заголовка для данного типа пакетов или
      больше размера пакета, пакет отбрасывается с возвратом управления.
Если P.type не является Data, Ack или DataAck и P.X == 0 (пакет имеет короткий
      порядковый номер), пакет отбрасывается с возвратом управления.
Если контрольная сумма заголовка некорректна, пакет отбрасывают с возвратом управления.
Если значение P.CsCov слишком велико для размера пакета,
      пакет отбрасывается с возвратом управления.
```

Этап 2: Проверка номеров портов и обработка состояния TIMEWAIT

```
/* Flow ID представляет собой квартет <src addr, src port, dst addr, dst port> */
Ищем flow ID в таблице и определяем соответствующий сокет.
Если сокета нет или S.state == TIMEWAIT,
/* Порядковый номер и номер подтверждения берутся из полученного пакета; см. 8.3.1. */
   Генерируется пакет Reset(No Connection), пока не будет P.type == Reset
   Отбросить пакет с возвратом управления.
```

Этап 3: Обработка состояния LISTEN

```
Если S.state == LISTEN,
   Если P.type == Request или P содержит корректную опцию Init Cookie,
      /* Требуется просмотр опций пакета для проверки Init Cookie. Однако на этом этапе
         обрабатываются только опции Init Cookie, а остальные опции – на этапе 8.
         Данное сканирование нужно выполнять только при наличии опций Init Cookie */
      /* Создать новый сокет и переключиться на него */
      Установить S := новый сокет для данной пары портов
      S.state = RESPOND
      Выбрать S.ISS (начальный порядковый номер) или установить из опций Init Cookie
      Инициализировать S.GAR := S.ISS
      Установить S.ISR, S.GSR, S.SWL, S.SWH из пакета или опций Init Cookie
      Продолжить с S.state == RESPOND
      /* Пакет Response будет генерироваться на этапе 11 */
   Иначе,
      Генерировать Reset(No Connection), пока не будет P.type == Reset
      Отбросить пакет с возвратом управления.
```

Этап 4: Подготовка порядковых номеров в состоянии REQUEST

```
Если S.state == REQUEST,
   Если (P.type == Response или P.type == Reset) и S.AWL <= P.ackno <= S.AWH,
      /* установить переменные нумерации, соответствующие другой точке, чтобы  пакет P
         прошёл проверки на этапе 6 */
      Установить S.GSR, S.ISR, S.SWL, S.SWH
      /* Обработка Response продолжается на этапе 10; Reset – на этапе 9 */
   Иначе,
      /* Только пакеты Response и Reset корректны в состоянии REQUEST */
      Генерировать Reset(Packet Error)
      Отбросить пакет с возвратом управления.
```

Этап 5: Подготовка порядковых номеров для Sync

```
Если P.type == Sync или P.type == SyncAck,
   Если S.AWL <= P.ackno <= S.AWH и P.seqno >= S.SWL,
      /* P является корректным, поэтому переменные нумерации соответственно обновляются.
         После этого P передаётся для проверки этапа 6. При необходимости на этапе 15
         генерируется SyncAck */
      Обновить S.GSR, S.SWL, S.SWH
   Иначе,
      Отбросить пакет с возвратом управления.
```

Этап 6: Проверка порядковых номеров

```
Если P.X == 0 и соответствующий признак Allow Short Sequential Numbers = 0,
   /* Пакет имеет короткий порядковый номер, но такие номера не разрешены */
   Отбросить пакет с возвратом управления.
Иначе, если P.X == 0,
   Расширить P.seqno и P.ackno до 48 битов с использованием процедуры параграфа 7.6
Пусть LSWL = S.SWL и LAWL = S.AWL
Если P.type == CloseReq или P.type == Close или P.type == Reset,
   LSWL := S.GSR + 1, LAWL := S.GAR
Если LSWL <= P.seqno <= S.SWH и (P.ackno не существует или LAWL <= P.ackno <= S.AWH),
   Обновить S.GSR, S.SWL, S.SWH
   Если P.type != Sync,
```

```
         Обновить S.GAR
   Иначе,
      Если P.type == Reset,
         Передать пакет Sync, подтверждающий S.GSR
      Иначе,
         Передать пакет Sync, подтверждающий P.seqno
      Отбросить пакет с возвратом управления.
```

Этап 7: Проверка неожиданных типов пакетов

```
   Если (S.is_server и P.type == CloseReq)
        или (S.is_server и P.type == Response)
        или (S.is_client и P.type == Request)
        или (S.state >= OPEN и P.type == Request и P.seqno >= S.OSR)
        или (S.state >= OPEN и P.type == Response и P.seqno >= S.OSR)
        или (S.state == RESPOND and P.type == Data),
      Передать пакет Sync, подтверждающий P.seqno
      Отбросить пакет с возвратом управления.
```

Этап 8: Обработка опций и маркировка подтверждаемости

```
   /* Обработка опций не описывается здесь. Некоторые опции (такие, как Mandatory) могут
      приводить к сбросу соединения при котором этапы 9 и далее не будут выполняться */
   Пометить пакет как подтверждаемый (в терминах Ack Vector – Received или Received ECN)
```

Этап 9: Обработка Reset

```
   Если P.type == Reset,
      Сбросить соединение
      S.state := TIMEWAIT
      Установить таймер TIMEWAIT
      Отбросить пакет с возвратом управления.
```

Этап 10: Обработка состояния REQUEST (вторая часть)

```
   Если S.state == REQUEST,
     /* Если мы пришли сюда, P является корректным пакетом Response от сервера (см. этап 4)
         и следует перейти в состояние PARTOPEN. В PARTOPEN нужно передать Ack, не
         передавать пакетов данных, периодически повторять передачу Ack, всегда включая
         все опции Init Cookie из пакета Response */
      S.state := PARTOPEN
      Установить таймер PARTOPEN
      Продолжить с S.state == PARTOPEN
      /* Этап 12 будет передавать пакет Ack, завершающий трехэтапное согласование */
```

Этап 11: Обработка состояния RESPOND

```
   Если S.state == RESPOND,
      Если P.type == Request,
         Передать Response, возможно с опциями Init Cookie
         если были переданы опции Init Cookie,
            Удалить S и возвратить управление
            /* Этап 3 будет создавать другой сокет, по завершении клиентом 3-этапного
               согласования */
      Иначе,
         S.OSR := P.seqno
         S.state := OPEN
```

Этап 12: Обработка состояния PARTOPEN

```
   Если S.state == PARTOPEN,
      Если P.type == Response,
         Передать Ack
      Иначе, если P.type != Sync,
         S.OSR := P.seqno
         S.state := OPEN
```

Этап 13: Обработка CloseReq

```
   Если P.type == CloseReq и S.state < CLOSEREQ,
      Генерировать Close
      S.state := CLOSING
      Установить таймер CLOSING
```

Этап 14: Обработка Close

```
   Если P.type == Close,
      Генерировать Reset(Closed)
      Разорвать соединение
      Отбросить пакет с возвратом управления.
```

Этап 15: Обработка Sync

```
   Если P.type == Sync,
      Генерировать SyncAck
```

Этап 16: Обработка данных

```
   /* В этой точке все данные приложения из P могут передаваться прикладной программе.
      Однако приложение НЕ ДОЛЖНО получать данных более, чем из одного пакета
      Request или Response */
```

9. Контрольные суммы

DCCP использует контрольную сумму заголовка для защиты заголовков пакетов от повреждений. В общем случае эта контрольная сумма учитывает также все данные приложения. Однако приложения DCCP могут запросить расчёт контрольной суммы заголовка с учётом лишь части данных, а в некоторых случаях и без учёта данных приложения. В таких случаях канальный уровень может снизить уровень защиты неучтенных в контрольной сумме частей пакетов

DCCP. Для некоторых шумных каналов и приложений, устойчивых к повреждению данных, отказ от учёта данных приложения в контрольной сумме может существенно повысить скорость доставки и общую производительность.

Область покрытия контрольной суммы может оказывать влияние также на работу механизма контроля насыщения. Пакет с повреждёнными данными приложения, которые были учтены в контрольной сумме, трактуются как потерянные. Это может приводить к жёсткой реакции на потери со стороны механизма контроля насыщения отправителя, который может в результате необоснованно применять санкции к соединениям через каналы с высоким фоновым уровнем ошибок. Комбинация уменьшенного покрытия для контрольной суммы заголовка и опций Data Checksum может приводить к тому, что конечные точки будут сообщать о повреждённых пакетах, не отбрасывая их, и будут использовать для уведомления опции Data Dropped и Drop Code 3 (см. параграф 11.7). Такой подход может давать приложениям некоторые преимущества. Однако нужны дополнительные исследования для определения подходящего отклика на повреждение данных, которое может в некоторых случаях напоминать насыщение. В настоящее время отклик на повреждённые пакеты происходит так же, как отклик на потерю пакетов.

Опция Data Checksum, которая использует сильный алгоритм CRC, позволяет конечным точкам детектировать повреждение данных приложения. В этом случае можно использовать API для предотвращения доставки приложению повреждённых пакетов даже в тех случаях, когда конечная точка получает из канала повреждённые данные при использовании уменьшенного покрытия для контрольной суммы заголовка. Однако снижение покрытия контрольной суммы заголовка для приложений, которым требуются корректные данные, в настоящее время рассматривается как экспериментальное. Это связано с тем, что сумма потерь и повреждений для пакетов с неполным покрытием контрольной суммы может быть существенно выше, нежели для пакетов, в которых контрольная сумма учитывает все данные, хотя доля потерянных пакетов в общем случае будет ниже. Реальное поведение будет зависеть от канала и для окончательного выбора требуется дополнительный опыт и исследования.

Снижение области покрытия контрольной суммой создаёт некоторые проблемы безопасности, рассмотренные в параграфе 18.1. Приложение B содержит дополнительную информацию по вопросу выбора области покрытия контрольной суммы. Реализации DCCP с неполным покрытием для контрольной суммы берут своё начало от UDP-Lite [RFC3828].

9.1. Поле Checksum в заголовке

DCCP использует алгоритм расчёта контрольных сумм TCP/IP. Поле Checksum в базовом заголовке DCCP (см. параграф 5.1) представляет собой 16-битовое дополнение до 1 суммы 16-битовых дополнений до 1 всех 16-битовых слов заголовка DCCP, опция DCCP, псевдозаголовка, взятого из заголовка сетевого уровня и, в зависимости от значения поля Checksum Coverage, части или всех данных приложения. При расчёте контрольной суммы значение поля Checksum принимается нулевым. Если пакет содержит нечётное количество байтов в полях, используемых для расчёта контрольной суммы, справа к последнему байту добавляется 8 нулевых битов для формирования полного 16-битового слова при вычислении контрольной суммы. Этот байт заполнения не передаётся через сеть.

Псевдозаголовок вычисляется так же, как для протокола TCP. Для IPv4 он имеет размер 96 битов и включает адреса IPv4 для получателя и отправителя, номер протокола IP для DCCP (дополняется слева 8 битами с нулевым значением) и размер DCCP в 16-битовых словах (длина заголовка DCCP вместе с опциями плюс размер всех данных), как описано в параграфе 3.1 [RFC793]. Для IPv6 псевдозаголовок имеет размер 320 и включает адреса IPv6 для отправителя и получателя, размер DCCP в 32-битовых словах, номер протокола IP для DCCP (дополняется слева 24 битами с нулевым значением), как описано в параграфе 8.1 [RFC2460].

Пакеты с некорректной контрольной суммой заголовка **должны** игнорироваться. В частности, **недопустимо** обрабатывать опции таких пакетов.

9.2. Поле заголовка Checksum Coverage

Поле Checksum Coverage в базовом заголовке DCCP (см. параграф 5.1) задаёт, какая часть пакета учитывается при вычислении контрольной суммы в поле Checksum:

CsCov = 0

Поле Checksum учитывает заголовок DCCP, опции DCCP, псевдозаголовок сетевого уровня и все данные приложения в пакете, возможно дополненные справа нулями для выравнивания по границе 16-битового слова.

CsCov = 1-15

Поле Checksum учитывает заголовок DCCP, опции DCCP, псевдозаголовок сетевого уровня и начальные (CsCov-1)*4 байтов данных приложения.

Таким образом, при CsCov = 1 данные приложения в контрольной сумме не учитываются совсем. Значение (CsCov-1)*4 **должно** быть не более общего размера данных приложения в пакете. Пакеты с некорректными значениями CsCov **должны** игнорироваться. В частности, **недопустима** обработка опций из таких пакетов. Толкование значений поля, отличных от 0 и 1, следует считать экспериментальным.

Отличные от 0 значения указывают, что допустимо повреждение в части или всех данных приложения, содержащихся в пакете DCCP. Фактически DCCP не сможет даже обнаружить повреждение данных в областях, не покрываемых контрольной суммой заголовка, если не используется опция Data Checksum. Приложениям не следует принимать каких-либо допущений о корректности полученных данных, не учитываемых в контрольной сумме и, следовательно, при необходимости такого контроля вводить свой механизм проверки корректности данных.

Прикладному интерфейсу DCCP следует позволять передающим приложениям предлагать значение CsCov для передаваемых пакетов, используя по умолчанию 0 (полное покрытие). Признак Minimum Checksum Coverage, описанный ниже, позволяет конечной точке отвергнуть доставку данных в пакетах с частичным покрытием для контрольной суммы - по умолчанию принимаются данные приложения только с полным покрытием. Нижележащие уровни, которые поддерживают частичное детектирование ошибок, **могут** использовать поле Checksum Coverage как рекомендацию, где не требуется искать возможные ошибки. Нижележащие уровни **должны** использовать строгий механизм детектирования для обнаружения ошибок по крайней мере в важных частях пакета и отбрасывания

повреждённых пакетов. К важным частям относятся байты, расположенные между первым байтом заголовка IP и последним байтом, указанным полем Checksum Coverage.

Более детальное обсуждение вопросов частичного покрытия для контрольной суммы в части прикладного уровня и интерфейса с нижележащим уровнем приводится в [RFC3828].

9.2.1. Признак Minimum Checksum Coverage

Признак Minimum Checksum Coverage позволяет конечной точке DCCP определить будет ли партнёр принимать пакеты со сниженным значением Checksum Coverage. Например, DCCP A передаёт опцию Change R(Minimum ChecksumCoverage, 1) точке DCCP B, чтобы проверить будет ли B принимать пакеты с Checksum Coverage = 1.

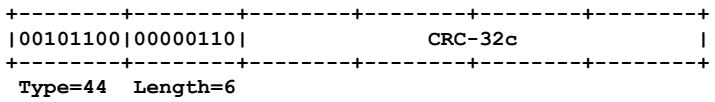
Признак Minimum Checksum Coverage имеет номер 8 и устанавливается с приоритетом сервера. Признак принимает однобайтовые целочисленные значения от 0 до 15; значения от 16 и выше зарезервированы. Признак Minimum Checksum Coverage/B отражает значения Checksum Coverage, которые DCCP B считает неприемлемыми. Пусть Minimum Checksum Coverage/B = MinCsCov. Тогда

- если MinCsCov = 0, DCCP B будет принимать только пакеты с CsCov = 0;
- если MinCsCov > 0, DCCP B будет также принимать пакеты с CsCov >= MinCsCov.

DCCP B **может** отказаться от обработки данных приложения из пакетов с недопустимым значением Checksum Coverage. О таких пакетах **следует** сообщать с помощью опций Data Dropped (параграф 11.7) с Drop Code 0, Protocol Constraints. Новые соединения начинаются с Minimum Checksum Coverage = 0 для обеих сторон.

9.3. Опция Data Checksum

Опция Data Checksum содержит 32-битовую контрольную сумму CRC-32c для данных приложения в пакете DCCP.



Передающая точка DCCP рассчитывает значение CRC для байтов, содержащихся в области данных приложения, и помещает полученное значение в поле данных опции. Для расчёта значения опции Data Checksum используется такой же алгоритм CRC-32c, который применяется в SCTP [RFC3309]. Отметим, что значение контрольной суммы CRC-32c для 0 байтов данных равно 0. Контрольная сумма заголовка DCCP будет учитывать значение опции Data Checksum, поэтому контрольную сумму данных требуется вычислять заранее, до расчёта контрольной суммы заголовка.

Конечная точка DCCP, получившая пакет с опцией Data Checksum, **должна** или **может** проверить контрольную сумму данных приложения - выбор определяется значением признака Check Data Checksum, описанного ниже. При проверке контрольной суммы последняя рассчитывается с использованием того же алгоритма CRC-32c, который применялся для расчёта значения поля Data Checksum. Если значение рассчитанной контрольной суммы не совпадает с полученным, конечная точка может реагировать одним из двух способов, описанных ниже.

- Принимающее приложение может принимать заведомо повреждённые данные через некий специальный дополнительный интерфейс API. В этом случае данные из пакета **должны** передаваться приложению вместе с информацией о повреждении данных. Более того, принимающая конечная точка **должна** сообщить о доставке повреждённого пакета на передающую сторону с помощью опции Data Dropped (Drop Code 7, Delivered Corrupt).
- В остальных случаях принимающая конечная точка **должна** отбросить данные приложения и сообщить о повреждении на передающую сторону с помощью опции Data Dropped (Drop Code 3, Corrupt).

В обоих случаях пакет считается подтверждаемым (поскольку обработка заголовка была проведена) и, следовательно, будет подтверждаться с использованием эквивалента Ack Vector для состояния Received или Received ECN Marked.

Хотя опция Data Checksum предназначена для пакетов, содержащих данные приложения, её можно включать и в другие пакеты (такие, как DCCP-Ack, DCCP-Sync и DCCP-SyncAck). Получателю **следует** рассчитывать для области данных таких пакетов контрольную сумму CRC-32c, как это делается для пакетов DCCP-Data и других пакетов с данными приложения. Если значения CRC не совпадают, об это **требуется** сообщить с использованием опции Data Dropped (Drop Code 3), хотя данные в любом случае не доставляются приложению.

9.3.1. Признак Check Data Checksum

Признак Check Data Checksum позволяет конечной точке DCCP определить будет ли её партнёр проверять опции Data Checksum. Точка DCCP A передаёт обязательную (Mandatory) опцию Change R(Check Data Checksum, 1) точке DCCP B, чтобы потребовать от той проверки опций Data Checksum (в противном случае соединение будет сбрасываться).

Признак Check Data Checksum имеет номер 9 и устанавливается с приоритетом сервера. Этот признак принимает однобайтовые логические значения. Конечная точка DCCP B **должна** проверять все полученные опции Data Checksum, если Check Data Checksum/B = 1, хотя она **может** проверять эти опции и в случае Check Data Checksum/B = 0. Значения 2 и больше для этого признака являются резервными. Новые соединения организуются с Check Data Checksum = 0 для обеих сторон.

9.3.2. Использование контрольных сумм

Каналы Internet обычно достаточно строго проверяют целостность передаваемых пакетов [RFC3828, RFC3819]. По умолчанию заголовок DCCP содержит Checksum Coverage = 0 (полное покрытие). Однако значение Checksum Coverage может быть и отличным от нуля. Задавая частичное покрытие для контрольной суммы заголовка, приложение показывает свою устойчивость к повреждению данных, не защищённых контрольной суммой. Понимая это, канальный уровень может снизить строгость детектирования и/или корректировки ошибок при передаче незащищённых частей. В результате это приведёт к существенному росту вероятности повреждения данных при передаче. Опция Data Checksum позволяет с очень высокой вероятностью детектировать такие повреждения.

10. Контроль насыщения

Каждый механизм контроля насыщения, поддерживаемый протоколом DCCP, имеет идентификатор CCID, который представляет собой целое число от 0 до 255. При организации соединения (а в некоторых случаях и в процессе работы соединения) конечные точки выбирают механизм контроля насыщения путём согласования значений признаков CCID. Признак CCID имеет номер 1. Значение CCID/A равно идентификатору CCID, используемому для полусоединения от A к B. DCCP B передаёт опцию Change R(CCID, K) для того, чтобы запросить у DCCP A использование CCID K для своих пакетов данных. Признак устанавливается с приоритетом сервера, поэтому опции согласования CCID могут содержать множество подходящих значения CCID, отсортированных в порядке предпочтения. Например, опция Change R(CCID, 2 3 4) запрашивает у получателя использование CCID 2 для своих пакетов и показывает, что подходят также CCID 3 и CCID 4 (это соответствует последовательности байтов 35, 6, 1, 2, 3, 4: опция Change R (35), размер опции (6), идентификатор признака (1), значения CCID (2, 3, 4)). Аналогично, опция Confirm L(CCID, 2, 2 3 4) говорит её получателю, что отправитель использует для своих пакетов CCID 2, но приемлемы также значения CCID 3 и CCID 4.

Определённые в настоящее время значения CCID перечислены в таблице 5.

Таблица 5. Идентификаторы механизмов контроля насыщения DCCP.

CCID	*Название*	*Документ*
0-1	Резерв	[RFC4341]
2	TCP-like Congestion Control	[RFC4342]
3	TCP-Friendly Rate Control	
4 - 255	Резерв	

Новые соединения начинаются с использования обеими точками CCID 2. Если такой выбор неприемлем для конечной точки DCCP, она **должна** передать с флагом Mandatory опцию Change(CCID) в своих первых пакетах.

Все CCID, стандартизованные для использования с DCCP, будут соответствовать механизмам контроля насыщения, предварительно стандартизованным IETF. Мы предполагаем, что в течение некоторого времени все такие механизмы будут дружественны к TCP, но это не является требованием DCCP.

Реализациям DCCP, предназначенным для решения задач общего назначения, таким, как реализации протокола в ядрах операционных систем общего назначения, **следует** поддерживать по крайней мере CCID 2. Широкое распространение CCID 2 обеспечит взаимодействие, хотя отдельные приложения могут запрещать использование этого механизма.

10.1. Контроль насыщения в стиле TCP

CCID 2 (TCP-like Congestion Control – контроль насыщения в стиле TCP) обозначает механизм контроля насыщения с аддитивным увеличением AI и мультипликативным уменьшением MD (AIMD[1]), в соответствии с моделью, используемой для протокола TCP, включая окно насыщения, замедленный старт, таймауты и т. п. [RFC2581]. Механизм CCID 2 обеспечивает максимальное использование полосы в течение продолжительного периода, совместимое со сквозным контролем насыщения, но уменьшает вдвое окно насыщения в ответ на каждый факт возникновения перегрузки. Это ведёт к внезапным изменениям скорости, характерным для TCP. Приложениям следует использовать CCID 2, если они предпочитают использовать максимально доступную полосу для установившейся скорости. Это зачастую относится к приложениям, которые не доставляют данные непосредственно пользователю.

Например, гипотетическое приложение для передачи файлов на основе DCCP, использующее управление повторной передачей на прикладном уровне в случаях потери пакетов, может предпочесть CCID 2 по сравнению с CCID 3. Для интерактивных сетевых игр также может оказаться предпочтительным использование CCID 2.

Подробное описание CCID 2 содержится в [RFC4341].

10.2. Контроль насыщения TFRC

CCID 3 обозначает механизм контроля насыщения TFRC[2], основанный на выравнивании. Механизм TFRC разработан для обеспечения разумно беспристрастного распределения полосы между одновременными потоками в стиле TCP. Разумная беспристрастность означает, что скорость потока не более, чем вдвое отличается от скорости потока TCP при таких же условиях. Однако для TFRC характерны существенно более низкие вариации изменения пропускной способности во времени по сравнению с TCP и это делает механизм CCID 3 более предпочтительным по сравнению с CCID 2 для потоковых приложений, где важна относительно стабильная скорость передачи.

Подробное описание CCID 3 содержится в [RFC4342]. Механизм контроля насыщения TFRC изначально был описан в [RFC3448].

10.3. Опции, признаки и коды сброса, связанные с CCID

Половина значений типов опций, номеров признаков и Reset Code зарезервирована для использования с конкретными CCID. Механизмам CCID могут потребоваться новые опции (например, для обмена информацией о подтверждениях или скорости); например, резервирование пространства опций позволяет механизмам CCID создавать опции без изменения в пространстве глобальных опций протокола. Например, опция 128 может иметь разное значение для полусоединения с CCID 4 и полусоединения с CCID 8. Специфические для CCID опции и признаки никогда не будут конфликтовать с глобальными опциями и признаками, которые могут появиться в новых версиях данной спецификации.

Любой пакет может содержать информацию, имеющую значение для одного из полусоединений, поэтому относящиеся к CCID опции, признаки и коды сброса явно указывают полусоединение, к которому они относятся.

- Опции с номерами от 128 до 191 относятся к числу опций, передаваемых от отправителя (HC-Sender) к получателю (HC-Receiver) в данном полусоединении, а опции с номерами от 192 до 255 передаются от получателя к отправителю.

[1]Additive Increase – аддитивное увеличение, Multiplicative Decrease – мультипликативное уменьшение.
[2]TCP-Friendly Rate Control - дружественное к TCP управление скоростью.

- Значения Reset Code от 128 до 191 показывают, что отправитель (HC-Sender) сбросил соединение (скорей всего в результате проблем с подтверждениями, переданными HC-Receiver). Значения Reset Code от 192 до 255 показывают, что получатель (HC-Receiver) сбросил соединение (скорей всего в результате проблем с пакетами данных, переданными HC-Sender).

- Номера от 128 до 191 используются для признаков, поддерживаемых на стороне отправителя (HC-Sender); номера от 192 до 255 - для признаков, поддерживаемых на стороне получателя (HC-Receiver). Поскольку опции Change L и Confirm L для признаков передаются той стороной, на которой реализован признак, ясно, что любая опция Change L(128) передаётся со стороны HC-Sender, а любая опция Change L(192) - со стороны HC-Receiver. Аналогично, опция Change R(128) передаётся со стороны HC-Receiver, а опция Change R(192) - со стороны HC-Sender.

В качестве примера рассмотрим соединение DCCP, где полусоединение от A к B использует CCID 4, а полусоединение от B к A - CCID 5. На рисунке показана выборка связанных с CCID опций для обоих полусоединений.

Пакет	Опция	Полусоед.	CCID
A > B	128	от A к B	4
A > B	192	от B к A	5
A > B	Change L(128, ...)	от A к B	4
A > B	Change R(192, ...)	от A к B	4
A > B	Confirm L(128, ...)	от A к B	4
A > B	Confirm R(192, ...)	от A к B	4
A > B	Change R(128, ...)	от B к A	5
A > B	Change L(192, ...)	от B к A	5
A > B	Confirm R(128, ...)	от B к A	5
A > B	Confirm L(192, ...)	от B к A	5
B > A	128	от B к A	5
B > A	192	от A к B	4
B > A	Change L(128, ...)	от B к A	5
B > A	Change R(192, ...)	от B к A	5
B > A	Confirm L(128, ...)	от B к A	5
B > A	Confirm R(192, ...)	от B к A	5
B > A	Change R(128, ...)	от A к B	4
B > A	Change L(192, ...)	от A к B	4
B > A	Confirm R(128, ...)	от A к B	4
B > A	Confirm L(192, ...)	от A к B	4

Использование связанных с CCID опций и признаков в процессе согласования **не рекомендуется**, поскольку в момент обработки таких опций сложно предсказать, какое значение CCID будет выбрано. Например, если пакет DCCP-Request содержит последовательность опций Change L(CCID, 3), 128, связанная с CCID опция 128 может обрабатываться CCID 3 (если сервер поддерживает CCID 3) или принятым по умолчанию CCID 2 (если сервер не поддерживает CCID 3). Однако можно безопасно включать связанные с CCID опции совместно с опцией Mandatory. Например, если пакет DCCP содержит последовательность опций Mandatory, Change L(CCID, 3), 128, тогда опция 128 будет обработана CCID 3 или соединение просто будет сброшено.

Серверы, которые не поддерживают принятый по умолчанию механизм CCID 2, могут никогда не получать связанных с CCID 2 опций в пакетах DCCP-Request (поскольку сервер **должен** передавать опции Mandatory Change(CCID) в своём пакете DCCP-Response, связанные с CCID опции во всех остальных пакетах не будут относиться к CCID 2). Сервер **должен** трактовать такие опции как непонятные. Таким образом, он будет сбрасывать соединение, встретив опцию Mandatory, связанную с CCID, или запрос на согласование признака передавая пустую опцию Confirm для необязательной опции Change по изменению связанного с CCID признака и игнорируя остальные связанные с CCID опции.

10.4. Требования к профилям CCID

Каждый документ CCID Profile **должен** удовлетворять по крайней мере перечисленным ниже требованиям:

- Профиль **должен** включать имя и номер описываемого механизма CCID.

- Профиль **должен** описывать условия, в которых этот механизм явно будет полезен. Зачастую лучшим способом будет являться сравнение с существующими CCID.

- Профиль **должен** перечислять и описывать все связанные с данным CCID опции, признаки, значения Reset Code; **следует** также указать, какие из опций и признаков общего назначения, описанных в настоящем документе, имеют отношение к данному CCID.

- Все новые механизмы подтверждения **должны** включать способ передачи сигналов ECN Nonce Echo отправителю.

- Профиль **должен** описывать формат пакетов данных, всех опций, которые следует включать, и установку поля Ccval в заголовке.

- Профиль **должен** описывать формат пакетов подтверждения, включая опции, которые следует помещать в эти пакеты.

- Профиль **должен** определять контроль насыщения для пакетов данных. Это описание включает отклики на факты насыщения, простои, периоды вносимых приложением ограничений, а также опции DCCP Data Dropped и Slow Receiver. Механизмам CCID, которые реализуют контроль насыщения на уровне отдельных пакетов, **следует** указывать, как размер пакетов влияет на принимаемые механизмом контроля решения.

- Профиль **должен** указывать, когда генерируются подтверждения и как контролируется насыщение для них.

- Профиль **должен** определять, когда использующий CCID отправитель считается статичным.

- Профиль **должен** указывать, нужно ли подтверждать подтверждения CCID и как часто это следует делать.

10.5. Состояние насыщения

Большинство алгоритмов контроля насыщения принимает во внимание предысторию при определении текущего значения допустимой скорости. В CCID 2 характеристики насыщения включают окно насыщения и меру числа пакетов, остающихся в сети. В CCID 3 характеристики включают продолжительность недавних интервалов потерь. Оба механизма CCID используют оценку времени кругового обхода. Характеристики насыщения зависят от пути через сеть и становятся некорректными при смене пути. Поэтому отправителям и получателям DCCP **следует** сбрасывать свои характеристики насыщения, заново запуская механизм контроля из состояния slow start или его эквивалента, при существенном изменении сквозного пути через сеть. Например, конечной точке, которая передаёт или принимает сообщение Mobile IPv6 Binding Update [RFC3775], **следует** сбросить свои характеристики насыщения для всех соответствующих соединений DCCP.

Реализация DCCP **может** также сбрасывать свои характеристики насыщения при изменении CCID (т. е., при смене согласованного механизма CCID или установке нового значения признака). Таким образом, соединение в загруженной среде может уклоняться от сквозного контроля насыщения путём частого повтора согласования CCID, как оно может делать это путём открытия новых соединений в той же сессии. Такое поведение запрещено. Для его предотвращения реализации DCCP **могут** ограничивать скорость изменения CCID (например, отказываясь от замены значения признака CCID чаще одного раза в минуту).

11. Подтверждения

Контроль насыщения требует от получателей передачи отправителю информации о потерянных пакетах или пакетах с метками ECN. Получатели DCCP **должны** сообщать о любой замеченной перегрузке, как определено в соответствующем профиле CCID. Каждый механизм CCID указывает, когда должны передаваться подтверждения, какие опции они должны использовать и т. п. Для подтверждений DCCP также используется контроль насыщения, хотя от потока подтверждений не требуется ничего сверх подобия TCP friendly; каждый CCID определяет контроль насыщения для пакетов подтверждения.

Большинство подтверждений использует опции DCCP. Например, в полусоединении с CCID 2 (TCP-like) получатель передаёт подтверждающую информацию с использованием опции Ack Vector. В этом параграфе описываются опции общего назначения для подтверждений и показано, как подтверждения используют эти опции в условиях общего типа. Полное описание механизмов подтверждения, используемых каждым CCID, приводится в спецификации профиля CCID.

Опции подтверждений, такие, как Ack Vector, зависят от номера подтверждения DCCP и, таким образом, могут передаваться лишь в пакетах, где может присутствовать поле Acknowledgement Number. Опции подтверждения, полученные в пакетах иного типа, а именно, DCCP-Request и DCCP-Data, **должны** игнорироваться. Однако не все опции подтверждения являются обязательными для каждого пакета, содержащего поле Acknowledgement Number.

11.1. Подтверждения подтверждений и односторонние соединения

Протокол DCCP разрабатывался для работы с односторонними и двухсторонними потоками данных, а также для соединений, которые могут переходить из одного такого состояния в другое. Однако подтверждения, требуемые для однонаправленных потоков данных, существенно отличаются от подтверждений для двухсторонних соединений. В частности, однонаправленным потокам нужно заботиться о подтверждении подтверждений.

Проблема подтверждения подтверждений возникает из-за того, что некоторые механизмы подтверждения используют гарантированную доставку. Например, получатель HC-Receiver, использующий CCID 2 (TCP-like Congestion Control), передаёт подтверждения Ack Vector, содержащие подтверждающую информацию, с полной гарантией доставки. Точке HC-Sender следует время от времени информировать точку HC-Receiver о получении подтверждений. Если этого не делать, HC-Receiver может повторить передачу данных Ack Vector с самого начала соединения (для каждого пакета DCCP-Ack)! Однако следует отметить, что сами по себе подтверждения подтверждений не обязательно должны доставляться с гарантией - когда подтверждения подтверждений теряются, HC-Receiver будет просто поддерживать и периодически заново передавать старые состояния, связанные с подтверждениями. Следовательно не возникает необходимости в «подтверждении подтверждения подтверждений».

При двухстороннем обмене данными все требуемые подтверждения подтверждений автоматически включаются в нормальные подтверждения доставки данных. Для односторонних соединений, однако, получатель DCCP не передаёт данных, поэтому отправитель обычно не шлёт подтверждений. Следовательно, механизм CCID соответствующего полусоединения должен явно говорить, когда и как отправителю (HC-Sender) следует генерировать подтверждения подтверждений.

В качестве примера рассмотрим двухстороннее соединение, в котором обе половины используют один механизм CCID (2 или 3), и DCCP B перестаёт передавать данные, т. е., соединение становится односторонним.

> DCCP B прекращает передачу данных и шлёт DCCP A только пакеты DCCP-Ack. В случае CCID 2 (TCP-like Congestion Control) DCCP B использует Ack Vector для надёжной доставки информации о получении пакетов. Как сказано выше, точка DCCP A должна время от времени подтверждать «чистые» подтверждения от DCCP B, чтобы точка B могла освобождать старые состояния Ack Vector. Например, A может передавать пакет DCCP-DataAck взамен DCCP-Data. Однако в случае CCID 3 состояния подтверждений в общем случае «привязаны», поэтому точке A не требуется подтверждать подтверждения от точки B.
>
> При односторонней связи один механизм CCID (например, CCID соединения от A к B) управляет всеми подтверждениями DCCP в плане их содержимого, частоты и т. п. Для двухсторонних соединений CCID полусоединения от A к B управляет подтверждениями DCCP B (включая подтверждения подтверждений DCCP A), а CCID полусоединения от B к A управляет подтверждениями A.
>
> DCCP A переключает свою картину подтверждений из двухсторонней в одностороннюю, когда узнает, что точка DCCP B перестала передавать данные (статична). Обратное переключение в двухсторонний режим происходит тогда, когда точка A должна подтвердить хотя бы один пакет DCCP-Data или DCCP-DataAck от точки DCCP B.

Каждый механизм CCID определяет способ детектирования статичности для этого CCID и способ организации подтверждения подтверждений для односторонних соединений. CCID полусоединения от B к A определяет, когда DCCP B становится статичным. Обычно это происходит по истечении определённого периода, в течение которого точка B не передаёт ни одного пакета данных; в CCID 2, например, этот период равен большему из двух значений - 0,2 и два интервала кругового обхода. CCID от A к B определяет, как DCCP A обеспечивает подтверждения подтверждений после того, как точка DCCP B становится статичной.

11.2. Добавление подтверждений

Данные подтверждения от A к B **могут «**прицепляться» к данным, передаваемым точкой DCCP B, если это не задерживает доставку подтверждения больше, чем дозволено CCID от A к B. Однако для данных подтверждения зачастую требуется более 4 байтов. Большой объем данных подтверждения, добавленных перед большим блоком данных может привести к тому, что размер пакета превысит максимально допустимое значение. В этом случае DCCP B **следует** передавать раздельные пакеты DCCP-Data и DCCP-Ack или подождать (но не слишком долго) отправки более мелкой дейтаграммы.

Использование «прицепа[1]» является общепринятым для DCCP A, когда полусоединение от B к A становится статичным, т. е., когда DCCP A просто подтверждает подтверждения DCCP B. Существует три причины подтверждения подтверждений DCCP B - чтобы позволить DCCP B освободить информацию о ранее подтверждённых пакетах данных от A, уменьшить размер будущих подтверждений и управлять скоростью, с которой подтверждения будут передаваться. Поскольку все эти причины являются второстепенными, DCCP A может обычно дождаться пакета данных для добавления к нему данных подтверждения. Если точка DCCP B захочет получить подтверждение раньше, она может передать пакет DCCP-Sync.

Все ограничения на добавление подтверждений в пакеты данных описываются в соответствующих профилях CCID.

11.3. Признак Ack Ratio

Признак Ack Ratio позволяет отправителю (HC-Sender) влиять на скорость, с которой получатели (HC-Receiver) генерируют пакеты DCCP-Ack, влияя таким образом на загрузку обратного пути. Это отличается от протокола TCP, который в настоящее время не контролирует насыщение для трафика, содержащего «чистые» подтверждения. Контроль насыщения на пути возврата с помощью Ack Ratio не пытается быть похожим на TCP - он просто старается предотвратить возможное насыщение и обеспечить более эффективное, нежели в TCP, поведение для случаев потери или маркировки большого числа пакетов на пути возврата.

Ack Ratio применяется к CCID, чьи получатели (HC-Receiver) слишком часто подтверждают пакеты данных. Значение Ack Ratio/A равно округлённому отношению числа пакетов данных, переданных DCCP A, к числу пакетов DCCP-Ack, переданных DCCP B. Более высокие значения Ack Ratio соответствуют более редкой передаче пакетов DCCP-Ack; отправитель поднимает Ack Ratio, когда путь возврата перегружается, и снижает Ack Ratio при свободном пути возврата. Каждый профиль CCID определяет способ контроля насыщения на пути подтверждений и, в частности, использование Ack Ratio. Например, CCID 2 использует Ack Ratio для контроля насыщения на пути подтверждений, CCID 3 не использует. Однако каждый признак Ack Ratio имеет определённое значение, независимо от того, используется ли это значение соответствующим CCID.

Признак Ack Ratio имеет номер 5 и устанавливается без согласования. Признак принимает двухбайтовые целочисленные значения. Значение Ack Ratio/A = 4 означает, что DCCP B будет передавать по крайней мере один пакет подтверждения на каждые 4 пакета данных, переданные DCCP A. Точка DCCP A передаёт опцию Change L(Ack Ratio) для того, чтобы уведомить DCCP B о допустимом числе подтверждений. Значение Ack Ratio = 0 говорит, что соответствующее полусоединение не использует Ack Ratio для контроля за частотой подтверждений. Новые соединения начинаются с Ack Ratio 2 для обеих точек; это значение обеспечивает поведение подтверждений, аналогичное отложенным подтверждениям в TCP.

Значения Ack Ratio следует трактовать, как рекомендации, а не жёсткие требования. Управляемое с помощью Ack Ratio поведение системы подтверждения доставки похоже на поведение подтверждений TCP в отсутствии насыщения и несколько более консервативно при возникновении перегрузки на пути возврата. Соответствие этим задачам является более важным, нежели точная реализация Ack Ratio. В частности, следует выполнять ряд рекомендаций:

- Получатели **могут** добавлять подтверждающую информацию в пакеты данных, создавая для этого пакеты DCCP-DataAck. Признак Ack Ratio не относится к подтверждениям, добавленным в пакеты данных. Однако, если пакет данных слишком велик, чтобы добавлять к нему подтверждение, или частота передачи данных ниже, чем предлагает значение Ack Ratio для подтверждений, точке DCCP B **следует** передавать «чистые» пакеты DCCP-Ack для обеспечения генерации одного подтверждения на Ack Ratio принятых пакетов данных.

- Получатели **могут** поднять частоту передачи подтверждений вместо передачи подтверждения сразу же по получении пакета данных. Получателям, повышающим частоту передачи подтверждений, **следует** поднимать её приблизительно до уровня Ack Ratio, а также **следует** включать опции Elapsed Time (параграф 13.2), чтобы помочь отправителю рассчитать время кругового обхода.

- Получателям **следует** поддерживать таймеры отложенных подтверждений (подобно TCP), с помощью которых подтверждения пакетов могут задерживаться не более, чем на T секунд. Задержка позволяет получателю собрать дополнительные пакеты для подтверждения и снизить связанный с подтверждениями объем служебного трафика. По прошествии времени T подтверждение передаётся без дополнительного ожидания. По умолчанию для T следует устанавливать значение 0,2 секунды, принятое в большинстве реализаций TCP. Это может приводить к более частой передаче подтверждений, нежели задаёт значение Ack Ratio.

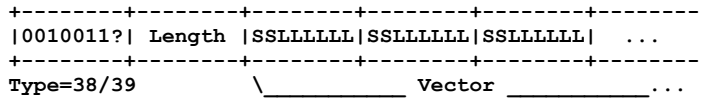
- Получателям **следует** передавать подтверждения незамедлительно по получении для пакетов с маркировкой ECN Congestion Experienced и пакетов с нарушением порядка доставки, что может говорить о потере пакетов. Однако нет необходимости в передаче незамедлительных подтверждений чаще одного раза за период кругового обхода.

[1] В оригинале «piggybacking». *Прим. перев.*

- Получатели **могут** игнорировать значение Ack Ratio, если они осуществляют свой контроль насыщения для подтверждений. Например, получатель, который знает частоту потери и маркировки пакетов DCCP-Ack, может по своему усмотрению поддерживать частоту передачи подтверждений в стиле TCP. Такие получатели **должны** обеспечивать постоянное наличие информации о частоте потери и маркировки пакетов или возвращаться к использованию Ack Ratio в случаях нехватки такой информации, которая может возникать в период статичности получателя.

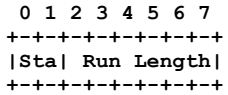
11.4. Опции Ack Vector

Опции Ack Vector обеспечивает групповое кодирование истории получения пакетов данных клиентом. Каждый байт вектора даёт состояние пакета данных в истории потерь и число предшествующих пакетов с таким же состоянием. Вид опции показан на рисунке.



Два варианта опции Ack Vector (тип 38 и 39) отличаются лишь значениями, которые они подразумевают для ECN Nonce Echo (см. параграф 12.2).

Сам вектор представляет собой последовательность байтов, кодируемую, как показано на рисунке.



Состояние (Sta) занимает два старших бита и может принимать одно из значений, указанных в таблице 6.

Таблица 6. Состояния DCCP Ack Vector.

Состояние	***Название***
0	Received
1	Received ECN Marked
2	резерв
3	Not Yet Received

Термин ECN marked относится к пакетам, с кодом ECN 11, CE (Congestion Experienced); о пакетах, принятых с таким кодом ECN, **должно** передаваться уведомление с использованием State 1, Received ECN Marked. О пакетах, полученных с кодами ECN 00, 01 или 10 (Non-ECT, ECT(0) и ECT(1), соответственно), **должно** передаваться уведомление с использованием State 0, Received.

Поле Run Length (младшие шесть битов каждого байта) указывает количество последовательных пакетов в состоянии State. Значение Run Length = 0 говорит о том, что соответствующее состояние относится только к одному пакету, значение Run Length = 63 говорит, что состояние относится к 64 последовательным пакетам. При числе пакетов 65 и более должны использоваться несколько байтов.

Первый байт первой опции Ack Vector относится к пакету, указанному полем Acknowledgement Number, а следующие байты относятся к более старым пакетам. **Недопустимо** передавать опции Ack Vector в пакетах DCCP-Data и DCCP-Request, которые не имеют поля Acknowledgement Number, и все опции Ack Vector, полученные в таких пакетах, **должны** игнорироваться.

Опция Ack Vector, содержащая десятичные значения 0,192,3,64,5, для которой поле Acknowledgement Number имеет десятичное значение 100, показывает, что:

пакет 100 был получен (Acknowledgement Number 100, State 0, Run Length 0);

пакет 99 был потерян (State 3, Run Length 0);

пакеты 98, 97, 96 и 95 были получены (State 0, Run Length 3);

пакет 94 имел маркер ECN (State 1, Run Length 0);

пакеты 93, 92, 91, 90, 89, 88 были получены (State 0, Run Length 5).

Одна опция Ack Vector может подтверждать до 16192 пакетов данных. Если нужно подтвердить больше пакетов, нежели можно указать в 253 байтах опции Ack Vector, можно передать множество опций Ack Vector; вторая опция Ack Vector начинается там, где заканчивается первая и т. д..

Состояния Ack Vector подчиняются двум ограничениям общего значения (эти принципы **следует** использовать также для других механизмов подтверждения; состояния Ack Vector просто позволяют упростить разъяснение принципов):

1. Пакеты, для которых указывается State 0 или State 1, **должны** быть подтверждаемыми - их опции должны быть обработаны принимающим стеком DCCP. Любые данные из пакетов не обязательно доставляются приложению - фактически они могут быть просто отброшены.

2. Пакеты, для которых указывается State 3, **должны** быть неподтверждаемыми. Согласование признаков и опции таких пакетов **недопустимо** обрабатывать и таким пакетам **недопустимо** ставить в соответствие значение Acknowledgement Number.

О пакетах, отброшенных в приёмном буфере приложения, **должны** передаваться уведомления, как о Received или Received ECN Marked (состояния 0 и 1), в зависимости от состояния ECN; значения ECN Nonce из таких пакетов **должны** включаться в Nonce Echo. Опция Data Dropped информирует отправителя о том, что некоторые пакеты, указанные, как принятые, на самом деле были отброшены на уровне приложения.

Одну или несколько опций Ack Vector, которые (совместно) сообщают о состоянии пакета с порядковым номером меньше ISN (начальный порядковый номер), **следует** рассматривать, как некорректные. Принимающей точке DCCP **следует** игнорировать такие опции или сбросить соединение с использованием Reset Code 5, Option Error. Ни одна из

опций Ack Vector не может относиться к пакетам, которые ещё не были переданы, что можно проверить с помощью Acknowledgement Number, как описано в параграфе 7.5.3, но в результате атак, ошибок в реализациях или некорректного поведения опция Ack Vector может показывать, что пакет был принят до его реальной доставки. В параграфе 12.2 описаны методы обнаружения таких случаев и реакция отправителя на них. Пакеты, которые не были включены ни в одну опцию Ack Vector, отправителю **следует** трактовать как ещё не принятые (not yet received, State 3).

В Приложении A приведено не являющееся нормативным описание деталей обработки подтверждений DCCP в контексте абстрактной реализации Ack Vector.

11.4.1. Согласованность Ack Vector

Отправитель DCCP обычно будет получать многочисленные подтверждения для некоторых отправленных им пакетов данных. Например, HC-Sender может получить два пакета DCCP-Ack с опциями Ack Vector, каждый из которых будет содержать информацию о порядковом номере 24 (информация о порядковом номере обычно повторяется в каждом подтверждении, пока HC-Sender не подтвердит получение подтверждения; в этом случае HC-Receiver передаёт подтверждения быстрее, чем HC-Sender подтверждает их получение). В идеальном мире две опции Ack Vector всегда будут согласованы. Однако существует множество причин, по которым согласования может не быть.

- Получатель HC-Receiver принял пакет 24 в промежутке между передачей своих подтверждений и в первом он указал, что пакет 24 не принят (State 3), а во втором указал статус ECN (State 0 или 1).
- Получатель HC-Receiver принял пакет 24 в промежутке между передачей своих подтверждений, а порядок доставки этих подтверждений был нарушен в сети. В этом случае пакеты будут представляться как сменившие State 0 или 1 на State 3.
- В сети возникли дубликаты пакета 24 и один или несколько таких дубликатов имеют маркер ECN. Это может представляться как переход между состояниями 0 и 1.

Чтобы справиться с такими ситуациями реализации DCCP на стороне HC-Sender **следует** объединять множество полученных состояний Ack Vector, как показано на рисунке.

Прежнее состояние	Полученное состояние 0	1	3
0	0	0/1	0
1	1	1	1
3	0	1	3

Для чтения таблицы выбирается строка, соответствующая старому состоянию пакета, и колонка, соответствующая состоянию пакета в недавно полученной опции Ack Vector; пересечение покажет новое состояние пакета. Если старое состояние было 0 (принят немаркированный пакет), а новое – 1 (пакет принят с маркером ECN), в качестве нового состояния следует установить 0 или 1. Реализация HC-Sender будет безразлична к изменению порядка доставки подтверждений, если она выберет для данной ячейки в качестве нового состояния 1.

Получателю (HC-Receiver) следует собирать информацию о принятых пакетах, как показано на рисунке.

Сохраненное состояние	Принятый пакет 0	1	3
0	0	0/1	0
1	0/1	1	1
3	0	1	3

Эта таблица отличается от таблицы для отправителя лишь тем, что когда сохранённое состояние равно 1, а полученное - 0, получатель может сменить сохранённое состояние на 0.

HC-Sender **может** выбрать отказ от старой информации, собранной из полученных от HC-Receiver опций Ack Vector; в этом случае отправитель **должен** игнорировать полученные недавно от HC-Receiver подтверждения для тех старых пакетов. Представляется более разумным сохранение информации Ack Vector, чтобы отправитель HC-Sender мог отказаться от своей реакции на предполагаемое насыщение, когда «потерянный» пакет неожиданно найдётся (смена State 3 на State 0).

11.4.2. Покрытие Ack Vector

Можно разбить пакеты, переданные от HC-Sender получателю HC-Receiver, на 4 почти непрерывные группы. Начиная с наиболее старых пакетов это будут:

1. пакеты, уже подтверждённые HC-Receiver, для которых HC-Receiver знает, что отправитель HC-Sender определённо получил подтверждения;
2. пакеты, уже подтверждённые HC-Receiver, для которых HC-Receiver не имеет уверенности в том, что отправитель HC-Sender уже получил подтверждения;
3. пакеты, которые получатель HC-Receiver ещё не подтвердил;
4. пакеты, ещё не полученные HC-Receiver.

Объединение групп 2 и 3 называется окном подтверждения (Acknowledgement Window). В общем случае каждая опция Ack Vector, генерируемая HC-Receiver, будет покрывать все окно Acknowledgement Window - подтверждения Ack Vector являются кумулятивными (это упрощает поддержку Ack Vector на стороне HC-Receiver; см. Приложение A). По мере доставки пакетов это окно будет расширяться справа и сужаться слева. Окно расширяется по причине доставки новых

пакетов и сужается в результате того, что номера подтверждений HC-Sender будут подтверждать более ранние подтверждения, перемещая пакеты из группы 2 в группу 1.

11.5. Признак Send Ack Vector

Признак Send Ack Vector позволяет точкам DCCP согласовать использование опции Ack Vector для передачи информации о насыщении. Опции Ack Vector обеспечивают подробную информацию о потерях пакетов и позволяют отправителю проинформировать прикладную программу на своей стороне о том, что некоторые пакеты были отброшены. Признак Send Ack Vector является обязательным для некоторых CCID и необязательным для остальных.

Признак Send Ack Vector имеет номер 6 и устанавливается с приоритетом сервера. Признак принимает однобайтовые логические значения. Точка DCCP A **должна** передавать опции Ack Vector в своих подтверждениях, если Send Ack Vector/A = 1, хотя она **может** передавать опции Ack Vector даже в случае Send Ack Vector/A = 0. Значения признака 2 и более являются резервными. Новые соединения начинаются с Send Ack Vector 0 для обеих сторон. DCCP B передаёт опцию Change R(Send Ack Vector, 1) точке DCCP A для запроса у A передачи опций Ack Vector в пакетах подтверждений.

11.6. Опция Slow Receiver

HC-Receiver передаёт опцию Slow Receiver своему отправителю, чтобы сказать о возникновении проблем при приёме данных от того. Отправителю HC-Sender **не следует** повышать скорость передачи в течение приблизительно одного периода кругового обхода после получения пакета с опцией Slow Receiver. По истечении одного периода кругового обхода действие опции Slow Receiver прекращается и HC-Sender может повышать скорость. Поэтому получателю HC-Receiver **следует** продолжать передачу опций Slow Receiver, если ему требуется предотвратить повышение скорости передачи в течение продолжительного срока. Опция Slow Receiver не говорит о насыщении в сети и отправителю HC-Sender не требуется снижать скорость передачи (при необходимости получатель может вынудить отправителя к снижению скорости путём отбрасывания пакетов с передачей опции Data Dropped или без неё, а также путём установки фиктивных маркеров ECN). Интерфейсам API следует позволять принимающим приложениям устанавливать опцию Slow Receiver, а передающим приложениям - определять наличие этой опции в пакетах.

Опция Slow Receiver является однобайтовой, как показано на рисунке.

```
+--------+
|00000010|
+--------+
 Type=2
```

Опция Slow Receiver не указывает причину, по которой получатель не может справиться с поток данных от отправителя. К таким причинам может относиться нехватка буферной ёмкости, перегрузка процессора или ограничения прикладной программы. Передающее приложение может реагировать на получение опции Slow Receiver снижением скорости передачи на прикладном уровне или иным способом.

Однако передающему приложению не следует реагировать на получение опции Slow Receiver передачей большего объёма данных. Хотя оптимальной реакцией на ограниченность ресурсов процессора приёмной стороны будет снижение уровня сжатия и передача большего объёма данных (данные с высоким уровнем сжатия могут загружать процессор на приёмной стороне значительно сильнее, чем больший расход памяти при передаче данных с менее сильным сжатием), этот вариант изменения формата следует согласовывать на уровне приложений, а не с помощью опции Slow Receiver.

Опция Slow Receiver частично реализует функции окна приёма TCP.

11.7. Опция Data Dropped

Опция Data Dropped показывает, что данные приложения из одного или множества полученных пакетов на самом деле не были доставлены приложению. Кроме того, Data Dropped сообщает причину отбрасывания данных - возможно они оказались повреждёнными или получатель неспособен принимать информацию с той скоростью, которую установил отправитель и данные были отброшены приёмным буфером. С помощью опции Data Dropped конечные точки DCCP могут отличать разные варианты потери пакетов. Это отличается от протокола TCP, в котором информация о потере пакетов единообразна.

Если явно не указано иное, механизмы контроля насыщения DCCP **должны** реагировать на опции Data Dropped, как будто пакеты были помечены в сети маркерами ECN CE. Мы предполагаем, что опция Data Dropped позволит исследовать отклики в условиях сильного насыщения на повреждения и отбрасывание конечными точками пакетов, но механизмы CCID **должны** консервативно реагировать на опции Data Dropped, пока это поведение не стандартизовано. В параграфе 11.7.2 обсуждаются отклики на перегрузки для всех определённых в настоящее время значений Drop Code.

Если данные приложения из полученного пакета отбрасываются по одной или нескольким перечисленным ниже причинам, об этом **следует** уведомлять с помощью опций Data Dropped. Кроме того, отправитель **может** сообщать, как о принятых только о тех пакетах, данные приложений из которых не были отброшены, но в этих случаях **недопустима** обработка опций из тех пакетов, которые не считаются принятыми.

Вид данных опции показан на рисунке.

```
+--------+--------+--------+--------+--------+--------
|00101000| Length | Block  | Block  | Block  |  ...
+--------+--------+--------+--------+--------+--------
Type=40          \___________ Vector ___________ ...
```

Область Vector содержит последовательности байтов, называемые блоками, каждая из которых кодируется в соответствии с одним из двух показанных на рисунке вариантов.

```
 0 1 2 3 4 5 6 7              0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+            +-+-+-+-+-+-+-+-+
|0| Run Length  |    или     |1|DrpCd|Run Len|
+-+-+-+-+-+-+-+-+            +-+-+-+-+-+-+-+-+
 Нормальный блок              Отброшенный блок
```

Первый байт опции Data Dropped относится к пакету, указанному полем Acknowledgement Number, а последующие байты относятся к более старым пакетам. Опции Data Dropped **недопустимо** включать в пакеты DCCP-Data и DCCP-Request, которые не включают поля Acknowledgement Number, и все опции Data Dropped, полученные в таких пакетах, **должны** игнорироваться.

Нормальные блоки, старший бит которых имеет значение 0, показывают, что все пакеты, указанные полем Run Length, были доставлены приложению. Отброшенные блоки, старший бит которых имеет значение 1, показывают, что пакеты, указанные в поле Run Len, не были доставлены приложению. Трехбитовое поле DrpCd[1] указывает, что произошло (в общем случае данные из пакета не были доставлены приложению). Пакеты, указанные как «ещё не полученные», **должны** включаться в нормальные блоки; пакеты, не включённые ни в одну из опций Data Dropped, трактуются, как будто они были включены в Normal Block. Определённые значения кодов отбрасывания для отброшенных блоков показаны в таблице 7.

Таблица 7. Коды отбрасывания пакетов DCCP.

Код причины	***Название***
0	Protocol Constraints
1	Application Not Listening
2	Receive Buffer
3	Corrupt
4 - 6	Резерв
7	Delivered Corrupt

Ниже приводится более детальное описание причин.

0 - данные из пакета были отброшены в результате протокольных ограничений. Например, данные были включены в пакет DCCP-Request, но принимающее приложение не поддерживает такую возможность, или данные были включены в пакет с недопустимо низким значением Checksum Coverage.

1 - данные были отброшены по той причине, что приложение больше не находится в режиме прослушивания (см. параграф 11.7.2).

2 - данные из пакета были отброшены в приёмном буфере, возможно в результате его переполнения (см. параграф 11.7.2).

3 - данные из пакета были отброшены по причине их повреждения (см параграф 9.3).

7 - данные из пакета оказались повреждёнными, но были доставлены приложению (см. параграф 9.3).

В качестве примера рассмотрим пакет, прибывающий с Acknowledgement Number 100, опцией Ack Vector, сообщающей, что все пакеты получены, и опцией Data Dropped, содержащей десятичные значения 0,160,3,162.

пакет 100 был получен (Acknowledgement Number 100, Normal Block, Run Length 0).

пакет 99 был отброшен в приёмном буфере (Drop Block, Drop Code 2, Run Length 0).

пакеты 98, 97, 96, 95 были получены (Normal Block, Run Length 3).

пакеты 95, 94, 93 были отброшены в приёмном буфере (Drop Block, Drop Code 2, Run Length 2).

Значения Run length, превышающие 128 (для Normal Block) или 16 (для Drop Block), должны кодироваться с использованием нескольких блоков. Одна опция Data Dropped может подтверждать до 32384 нормальных блоков, хотя получателю **не следует** передавать опцию Data Dropped, когда все относящиеся к делу пакеты помещаются в нормальные блоки. Если требуется подтвердить больше пакетов, нежели можно поместить в 253 байта опции Data Dropped, можно передавать множество опций Data Dropped. Вторая опция будет начинаться там, где закончится первая и т. д.

Одна или множество опций Data Dropped вместе могут сообщать о состоянии большего числа пакетов, нежели было передано, менять состояние для пакетов, вступать в противоречие с Ack Vector или эквивалентными опциями (например, сообщая о пакетах, которые ещё не доставлены, как об отброшенных в приёмном буфере). Такие опции **следует** рассматривать как некорректные. Принимающей стороне DCCP **следует** игнорировать такие опции или сбрасывать соединение с Reset Code 5, Option Error.

Прикладному интерфейсу DCCP следует позволять принимающему приложению задавать значения Drop Code, соответствующие полученным пакетам. Например, это позволит приложениям рассчитывать свои контрольные суммы, но продолжать уведомлять об отбрасывании пакетов в результате повреждения с помощью опции Data Dropped. Интерфейсу **не следует** позволять приложениям снижать «значимость» Drop Code; например, приложению не следует позволять менять для пакета состояние с «повреждён при доставке» (Drop Code 7) на «доставлен нормально» (нет Drop Code).

Информация Data Dropped передаётся с гарантией доставки. Т. е., конечной точке **следует** продолжать передачу опций Data Dropped до получения подтверждения, показывающего, что соответствующие опции были обработаны. В терминах Ack Vector каждому подтверждению следует включать опции Data Dropped, которые покрывают все окно подтверждений (Acknowledgement Window, см. параграф 11.4.2), хотя в тех случаях, когда все пакеты из этого окна помещаются в нормальный блок, опция не требуется.

11.7.1. Отклик на Data Dropped и обычный отклик на перегрузку

При решении вопроса об отклике на отдельное подтверждение или набор подтверждений, содержащих опции Data Dropped, механизм контроля насыщения **должен** принимать во внимание отброшенные пакеты, пакеты с маркерами ECN CE (включая маркированные пакеты, которые указаны в опции Data Dropped) и пакеты, отдельно указанные в Data Dropped. Для основанных на использовании окна механизмов контроля насыщения корректные отклики определены ниже.

[1]Drop Code - код отбрасывания.

Обозначим старое окно W. Независимо рассчитаем новое окно W_new1, которое предполагает отсутствие пакетов Data Dropped (W_new1 содержит только обычный отклик на насыщение), и новое окно W_new2, которое предполагает отсутствие потерянных или помеченных пакетов (W_new2 содержит только отклик Data Dropped). Мы полагаем, что Data Dropped рекомендует снижение размера окна насыщения, поэтому W_new2 < W.

Тогда реальное новое окно W_new **недопустимо** делать больше минимального из двух значений W_new1 и W_new2, а отправитель **может** объединить два отклика, установив

```
W_new = W + min(W_new1 - W, 0) + min(W_new2 - W, 0)
```

Детальное описание этой процедуры задаётся в спецификациях профилей CCID. Механизмы контроля насыщения, не использующие окно, **должны** вести себя аналогично. Точное поведение опять-таки определяется профилем CCID.

11.7.2. Отдельные значения Drop Code

Drop Code 0, Protocol Constraints[1] не указывает на какой-либо тип насыщения, поэтому механизму CCID на стороне отправителя **следует** реагировать на пакеты с Drop Code 0, как на принятые пакеты (с маркером ECN CE или без него, как подходит). Однако передающей конечной точке **не следует** передавать данные, пока она не убедится, что протокольные ограничения более не применимы.

Drop Code 1, Application Not Listening[2] означает, что приложение на той стороне, которая передала опции, более не принимает данные. Например, сервер мог закрыть своё приёмное полусоединение для получения новых данных после приёма от клиента запроса на завершение. Это будет ограничивать число состояний сервера для входящих данных и, следовательно, снижать риск повреждений в результате некоторых атак на службы. Опцию Data Dropped с Drop Code 1 **следует** передавать всякий раз, когда полученные данные игнорируются в результате того, что приложение прекратило прослушивание входящей информации. После того, как конечная точка передала для пакета уведомление с Drop Code 1, ей **следует** передавать этот код для всех последующих пакетов в данном полусоединении. После получения конечной точкой уведомления с Drop State 1, ей **следует** ожидать, что другая сторона больше не предполагает получение данных и их **не следует** передавать.

Drop Code 2, Receive Buffer показывает, что насыщение произошло на приёмном хосте. Например, если буфер сокета ядра, работающий по принципу отбрасывания из конца, слишком заполнен, чтобы принимать данные, следует передавать уведомление с Drop Code 2. Для буферов с отбрасыванием из начала или более сложных вариантов буферизации для сокета ядра, уведомлять об отбрасывании пакетов следует с Drop Code 2. Реализации DCCP могут также предоставлять интерфейс API, с помощью которого приложения могут помечать полученные пакеты, как Drop Code 2, показывая, что приложение выходит за допустимые пределы приёмного буфера в пользовательском пространстве (однако в общем случае не будет пользы от передачи уведомлений об отбрасывании пакетов с Drop Code 2 после того, как пройдёт более 2 периодов кругового обхода; HC-Sender может забыть за это время состояния подтверждения для пакетов, поэтому уведомление Data Dropped не будет оказывать влияния). На каждый пакет, о котором получено уведомление с Drop Code 2, **следует** уменьшать мгновенную скорость передачи на один пакет за период кругового обхода, если скорость больше одного пакета за время RTT. Каждый профиль CCID определяет специфический для данного CCID механизм, который обеспечивает это снижение скорости.

В настоящее время другие значения Drop Code (а именно, Drop Code 3, Corrupt; Drop Code 7, Delivered Corrupt и резервные значения 4-6) **должны** заставлять соответствующий механизм CCID вести себя, как при получении пакетов с маркерами ECN CE.

12. Явное уведомление о насыщении

Протокол DCCP полностью поддерживает ECN [RFC3168]. Каждый CCID задаёт для конечных точек способ отклика на маркировку ECN. Более того, DCCP, в отличие от TCP, позволяет отправителю контролировать скорость генерации подтверждений (с помощью опций типа Ack Ratio); поскольку для подтверждений также применяется контроль насыщения, они также квалифицируются как ECN-Capable Transport.

Каждый профиль CCID описывает, как CCID взаимодействует с ECN для трафика данных и для «чистых» пакетов подтверждения. Отправителю **следует** устанавливать флаг ECN-Capable Transport в заголовках IP своих пакетов, если признак получателя ECN Incapable для соответствующего CCID не запрещает это.

Остальная часть этой главы описывает признак ECN Incapable и взаимодействие ECN Nonce с опциями подтверждения, такими, как Ack Vector.

12.1. Признак ECN Incapable

Конечные точки DCCP поддерживают ECN по умолчанию, но признак ECN Incapable позволяет конечной точке отвергнуть использование явных уведомлений о насыщении. Использовать этот признак **не рекомендуется**. Несовместимость с ECN не позволит использовать возможные преимущества ECN и лишит отправителей возможности использования ECN Nonce для проверки корректности поведения получателей. Стек DCCP **может**, следовательно, оставить признак ECN Incapable нереализованным, действуя так, будто все соединения поддерживают ECN. Отметим, что недопустимое взаимодействие МСЭ, обманывающее реализации ECN в TCP [RFC3360], использует биты заголовка TCP, а не биты ECN в заголовке IP; нам не известно промежуточных устройств, которые будут блокировать поддерживающие ECN пакеты DCCP, разрешая передачу несовместимых с ECN пакетов DCCP.

Признак ECN Incapable имеет номер 4 и устанавливается с приоритетом сервера. Признак принимает однобайтовые логические значения. Точка DCCP A **должна** быть способна читать биты ECN в заголовках IP полученных кадров, когда ECN Incapable/A = 0 (это не зависит от возможности установки данной точкой битов ECN в передаваемых ею кадрах). Точка DCCP A будет передавать опцию Change L(ECN Incapable, 1) точке DCCP B, чтобы проинформировать ту о том, что A не может читать биты ECN. Если ECN Incapable/A = 1, все пакеты от DCCP B **должны** передаваться, как несовместимые с ECN. Новые соединения начинаются с ECN Incapable 0 (т. е., ECN поддерживается) для обеих сторон. Значения признака 2 и более зарезервированы.

[1] Протокольные ограничения.
[2] Приложение не прослушивает входящие данные.

Если точка DCCP не поддерживает ECN, она **должна** передавать обязательную опцию Change L(ECN Incapable, 1) другой конечной точке до получения от той подтверждения в опции Confirm R(ECN Incapable, 1) или разрыва соединения. Более того, **недопустимо** принимать какие-либо данные, пока другая точка не передаст опцию Confirm R(ECN Incapable, 1). **Следует** передавать в подтверждении опцию Data Dropped с Drop Code 0 (протокольные ограничения), если другая точка шлёт данные до передачи требуемой опции.

12.2. Сигналы ECN Nonce

Предотвращения перегрузок не будет и получатель будет иногда брать данные быстрее, если отправитель не говорит о возникшем насыщении. Таким образом, получатель может оказаться заинтересованным в фальсификации данных подтверждения, показывающей отброшенные или маркированные пакеты, как доставленные нормально. Эта проблема более серьёзна для DCCP, нежели для TCP, поскольку протокол TCP обеспечивает гарантированную доставку и скрыть от TCP потерю пакетов без нарушения работы приложения сложнее.

Сигналы ECN Nonce представляют собой механизм общего назначения, предотвращающий сокрытие ECN (или потери пакетов). Два значения (01 и 10) двухбитового поля ECN в заголовке показывают ECN-Capable Transport. Второй код (10) используется, как ECN Nonce. В общем случае отправитель случайно выбирает один из этих двух кодов для включения в передаваемые пакеты, запоминая последовательность этих кодов. В каждом подтверждении получатель указывает число полученных сигналов ECN Nonce. Это называется ECN Nonce Echo. Поскольку маркировка ECN и отбрасывание пакетов удаляют ECN Nonce, получатель, желающий скрыть маркировку или отбрасывание пакетов, может с вероятностью 50% угадать состояние ECN Nonce и избежать наказания за обман. Отправитель может жёстко реагировать на несоответствие ECN Nonce, вплоть до сброса соединения. Поле ECN Nonce Echo не обязательно должно быть целым числом - достаточно 1 бита для обнаружения подмены с вероятностью 50% и вероятность обнаружения обмана экспоненциально возрастает по мере увеличения числа переданных пакетов [RFC3540].

В DCCP поле ECN Nonce Echo включается в опции подтверждения. Например, опция Ack Vector существует в 2 вариантах - Ack Vector [Nonce 0] (опция 38) и Ack Vector [Nonce 1] (опция 39), - соответствующих двум значениям однобитового поля ECN Nonce Echo. Значение Nonce Echo для данной опции Ack Vector равно однобитовой сумме (чётность или «исключающее ИЛИ») значений ECN nonce для пакетов, которые Ack Vector показывает, как принятые без маркеров ECN. Таким образом, лишь пакеты, указанные, как State 0 (принятые корректные пакеты без маркера ECN), принимаются во внимание при вычислении суммы. Каждая опция Ack Vector содержит достаточную информацию, чтобы отправитель мог определить, каким должно быть значение Nonce Echo. Отправитель может заново рассчитать значение Nonce Echo, сравнить с полученным и принять меры при наличии расхождения (опция Ack Vector может сообщать состояние ECN Nonce для каждого пакета, но это будет ограничивать сжимаемость опции без обеспечения дополнительной защиты).

Каждому отправителю DCCP **следует** устанавливать биты ECN Nonce в передаваемых пакетах и запоминать пакеты, имеющие nonce. Когда отправитель детектирует несоответствие ECN Nonce Echo, он ведёт себя, как описано в следующем параграфе. Каждый получатель DCCP **должен** рассчитывать и использовать корректное значение для бита ECN Nonce Echo при отправке опций подтверждения.

Несовместимость с ECN, указанная признаком ECN Incapable, обрабатывается следующим образом - конечная точка, передающая пакеты не поддерживающему ECN получателю, **должна** передавать свои пакеты как несовместимые с ECN, а получатель, который не поддерживает ECN, **должен** использовать нулевое значение для всех битов ECN Nonce Echo.

12.3. Наказания за агрессию

Конечные точки DCCP имеют несколько механизмов детектирования некорректного поведения в связи с контролем насыщения. Например,

- отправитель может обнаружить несоответствие ECN Nonce Echo, говорящее, возможно, о некорректном поведении получателя;
- получатель может определить, реагирует ли отправитель на сигналы о насыщении или опцию Slow Receiver;
- конечная точка может быть способна детектировать, что её партнёр сообщает недопустимо малые значения Elapsed Time (параграф 13.2).

Конечной точке, обнаружившей предположительно некорректное поведение в части контроля насыщения, **следует** попытаться проверить корректность поведения партнёра. Например, передающая сторона может отправить пакет, в котором поле заголовка ECN имеет значение Congestion Experienced, 11 - если отправитель не сообщит о маркировке пакета, это будет говорить о его некорректном поведении.

После обнаружения предположительно некорректного поведения отправителю **следует** реагировать на него, как при получении от партнёра. уведомления о маркировке ECN для одного или нескольких недавних пакетов. Получателю, обнаружившему предположительно некорректное поведение, **следует** сообщить о наличии маркеров ECN в одном или нескольких недавних пакетах, которые на самом деле не имели маркеров. Кроме того, в таких случаях отправитель может реагировать, как при получении опции Slow Receiver, а получатель может передавать опции Slow Receiver. Допустимы и другие меры, способные снизить скорость передачи данных. Точка, обнаружившая грубые случаи некорректного поведения, **может** также сбросить соединение с использованием Reset Code 11, Aggression Penalty.

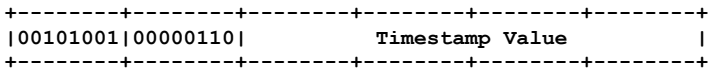
Однако несоответствие ECN Nonce и другие сигналы могут быть результатом неумышленных действий, ошибок в реализации или атак. В частности, успешная атака DCCP-Data (параграф 7.5.5) может приводить к тому, что получатель будет передавать некорректные значения ECN Nonce Echo. Поэтому сброс соединений и другие жёсткие меры **следует** применять только в крайних случаях, когда агрессивное поведение наблюдается в течение многих периодов кругового обхода.

13. Опции синхронизации

Опции Timestamp, Timestamp Echo и Elapsed Time помогают конечным точка DCCP явно измерять время кругового обхода.

13.1. Опция Timestamp

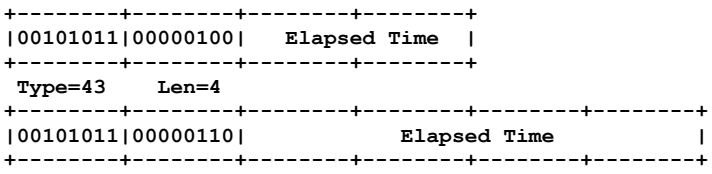
Эту опцию можно использовать в пакетах DCCP любого типа. Размер опции составляет 6 байтов.



Четыре байта данных опции позволяют включить в пакет временную метку. Метка представляет собой 32-битовое целое число, которое монотонно возрастает с течением времени (каждые 10 мксек на 1). При такой скорости значение Timestamp Value будет достигать максимума и сбрасываться в 0 приблизительно каждые 11,9 часа. Конечным точка не требуется измерять время с точностью 10 мксек; например, конечная точка, предпочитающая измерять время с дискретностью 1 мсек, может просто передавать значения Timestamp Value, кратные 100. Точное время, соответствующее Timestamp Value = 0, не задаётся - Timestamp Value имеет смысл лишь относительно значений Timestamp Value в пакетах, переданных через данное соединение. Точке DCCP, принявшей опцию Timestamp, **следует** включить в следующий передаваемый пакет опцию Timestamp Echo.

13.2. Опция Elapsed Time

Эта опция может включаться в любые пакеты DCCP, имеющие поле Acknowledgement Number; при получении опции в других пакетах, она **должна** игнорироваться. Опция показывает время, прошедшее с момента получения подтверждаемого пакета, который указывает поле Acknowledgement Number. Опция может занимать 4 или 6 байтов в зависимости от значения Elapsed Time. Опция Elapsed Time помогает корректировать оценку времени кругового обхода, когда интервал между приёмом и подтверждением пакета может быть достаточно велик (например, в CCID 3, где подтверждения передаются достаточно редко).

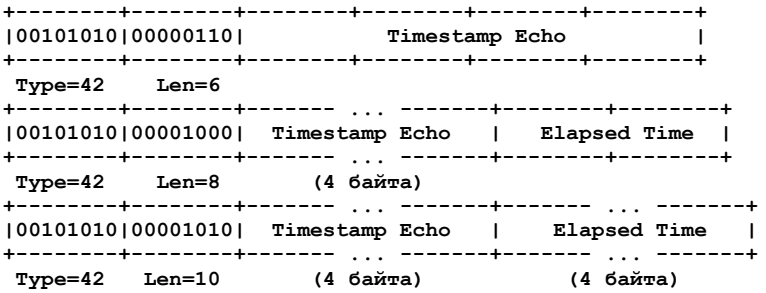


Поле данных опции, Elapsed Time, представляет оценку нижней границы времени, прошедшего с момента приёма подтверждаемого пакета в сотых долях миллисекунды (10 мксек). Если Elapsed Time меньше 0,5 сек, **следует** использовать 4 байтовый вариант опции. Значения Elapsed Time больше 0,65535 сек **должны** передаваться с использованием 6-байтового формата. Специальное значение Elapsed Time = 4294967295, которое соответствует приблизительно 11,9 часа, используется для представления времени, превышающего 42949,67294 сек. Для конечных точек DCCP **недопустимо** передавать значения Elapsed Time, существенно превышающее реальное время задержки между приёмом и подтверждением пакета. Соединение **может** быть сброшено с Reset Code 11, Aggression Penalty, если одна из сторон определит, что другая указала слишком большое значение Elapsed Time.

Единица измерения Elapsed Time (10 мксек) была выбрана, как компромисс между двумя противоречивыми требованиями. С одной стороны, обеспечивается достаточная гранулярность для снижения ошибок округления при использовании в скоростных средах ЛВС. С другой стороны, в два байта можно поместить информацию о большинстве разумных значений задержки.

13.3. Опция Timestamp Echo

Эту опцию можно использовать в любых пакетах DCCP при получении хотя бы одного пакета с опцией Timestamp. В общем случае конечной точке DCCP следует передавать одну опцию Timestamp Echo для каждой принятой опции Timestamp, делая это так, как будет удобно. Размер опции может составлять от 6 до 10 байтов, в зависимости от величины включаемого в неё значения Elapsed Time.



Первые 4 байта данных опции (поле Timestamp Echo) содержат поле Timestamp Value из полученной последней опции Timestamp. Обычно эта опция берётся из последнего принятого пакета (указываемого полем Acknowledgement Number, если оно присутствует), но это может быть и предшествующий пакет. Каждая принятая опция Timestamp в общем случае ведёт к передаче в ответ одной опции Timestamp Echo. Если точка получила множество опций Timestamp с момента передачи последнего пакета, она **может** игнорировать все опции Timestamp, используя только одну - из пакета с максимальным порядковым номером. **Можно** также включить в пакет множество опций Timestamp Echo, каждая из которых будет соответствовать своей опции Timestamp.

Значение Elapsed Time, подобно опции Elapsed Time, показывает время, прошедшее с момента получения опции, для которой передаётся ответная опция. Это время **должно** измеряться в сотых долях миллисекунды (10 мксек). Значение Elapsed Time предназначено для того, чтобы помочь отправителю опции Timestamp вычесть из времени кругового обхода время обработки на стороне приёма опции Timestamp. Эта возможность может оказаться особо важной для CCID, в которых подтверждения передаются достаточно редко и может пройти много времени между получением опции Timestamp и передачей ответной опции Timestamp Echo. Отсутствие поля Elapsed Time трактуется, как Elapsed Time = 0. **Следует** использовать наименьший по размеру вариант опции, позволяющий поместить значение Elapsed Time.

14. Максимальный размер пакета

Реализация DCCP **должна** поддерживать значение максимального размера пакета (MPS[1]), разрешённого для каждой активной сессии DCCP. Значение MPS определяется максимальным размером пакета, разрешаемым механизмом контроля насыщения (CCMPS[2]), максимальным размером пакета, поддерживаемым используемыми для сессии путями (PMTU[3]) [RFC1191] и размером заголовков IP и DCCP.

Прикладному интерфейсу DCCP **следует** разрешать приложению определять текущее значение MPS для DCCP. В общем случае реализация DCCP будет отвергать передачу всех пакетов, размер которых превышает MPS, возвращая приложению соответствующее сообщение об ошибке. Интерфейс DCCP **может** разрешать приложению запрашивать фрагментацию для пакетов, размер которых превышает PMTU, но не превышает CCMPS (пакеты, размер которых превышает CCMPS, **должны** отвергаться в любом случае). Фрагментацию **не следует** использовать по умолчанию, поскольку она снижает устойчивость к ошибкам - весь пакет будет отбрасываться при потере единственного фрагмента. Приложение обычно будет более устойчиво к ошибкам, если станет использовать пакеты, размер которых меньше PMTU.

Сообщаемое приложению значение MPS **следует** выбирать с учётом ожидаемого размера заголовка и опций DCCP. Если приложение предоставляет данные так, что при добавлении к ним опций, которые желает включать реализация DCCP, размер пакета будет превышать MPS, реализации следует передавать опции в отдельном пакете (таком, как DCCP-Ack) или уменьшить значение MPS, отбросить данные и вернуть приложению соответствующее сообщение об ошибке.

14.1. Измерение PMTU

Каждая точка DCCP **должна** отслеживать текущее значение PMTU для каждого из своих соединений, за исключением соединений IPv4, в которых приложения запросили фрагментацию пакетов. При инициализации PMTU **следует** использовать значение MTU для интерфейса, через который будут передаваться пакеты. Для инициализации MPS используется меньшее из значений PMTU и CCMPS (если оно существует).

Классический механизм определения PMTU использует нефрагментируемые пакеты. В IPv4 такие пакеты используют флаг DF[4], а в IPv6 все пакеты становятся нефрагментируемыми после того, как покинут исходный хост. Как сказано в [RFC1191], когда маршрутизатор получает пакет с флагом DF, размер которого превышает значение MTU для следующего отрезка пути, он возвращает отправителю сообщение ICMP Destination Unreachable, в котором поле Code показывает, что размер нефрагментируемого пакета слишком велик для пересылки (сообщение Datagram Too Big). Когда реализация DCCP получает сообщения Datagram Too Big, она снижает своё значение PMTU до значения поля Next-Hop MTU в полученном сообщении ICMP. Если в сообщении указано MTU = 0, отправитель выбирает значение PMTU на основе алгоритма, описанного в главе 7 [RFC1191]. Если полученное в сообщении значение MTU превышает текущее значение PMTU, сообщение Datagram Too Big игнорируется, как описано в [RFC1191] (мы полагаем, что это может вызывать проблемы для конечных точек DCCP, расположенных за некоторыми типами МСЭ).

Реализация DCCP может разрешать приложению время от времени запрашивать повторное определение PMTU. Это будет приводить к установке для PMTU значения MTU выходного интерфейса. Частоту таких запросов **следует** ограничивать (например, не более одного запроса в течение 2 сек.).

Отправитель DCCP **может** трактовать получение сообщения ICMP Datagram Too Big, как индикацию того, что пакет не был потерян в результате насыщения и, в целях контроля насыщения, **может** игнорировать сигнал получателя DCCP о том, что пакет не был получен. Однако в этом случае отправитель DCCP **должен** проверять биты ECN в заголовке пакета IP, содержащегося в сообщении ICMP, и игнорировать уведомление получателя только в том случае, когда эти биты ECN показывают, что пакет не сталкивался с насыщением до того, как попал на маршрутизатор, неспособный переслать пакет по причине превышения MTU.

Реализации DCCP **следует** всякий раз, когда это возможно, проверять, что сообщения ICMP Datagram Too Big действительно сгенерированы маршрутизаторами, поскольку атакующие могут снижать PMTU до недопустимо малых значений. Простейшим способом является проверка соответствия значения Sequence Number инкапсулированного в сообщении ICMP заголовка порядковым номерам пакетов, недавно переданных отправителем (в соответствии с современными спецификациями маршрутизаторам следует возвращать полный заголовок DCCP и данные из пакета, общим размером до 576 байтов [RFC1812] или минимальное значение IPv6 MTU [RFC2463], хотя не обязательно возвращать более 64 битов [RFC792]; любое количество данных, превышающее 128 битов, будет включать поле Sequence Number). Сообщения ICMP Datagram Too Big с некорректными или отсутствующими порядковыми номерами можно игнорировать или снижать в ответ на них значение PMTU лишь на короткое время. Если получено 3 или более разрозненных сообщения Datagram Too Big и другая конечная точка DCCP сообщает о том, что потеряно более 3 пакетов, реализации DCCP **следует** предполагать наличие плохо настроенного маршрутизатора и принимать указанное в сообщениях ICMP значение PMTU или (для IPv4) разрешать фрагментацию пакетов.

DCCP также допускает нарастающие пробы PMTU [PMTUD], при которых конечная точка DCCP начинает передавать мелкие пакеты с флагом DF и постепенно увеличивает размер пакетов до первой потери. Этот механизм не требует обработки сообщений ICMP. Пакеты DCCP-Sync лучше всего подходят для нарастающих проб, поскольку передача таких пакетов не связана с риском потери данных. Реализация DCCP помещает произвольную информацию в область данных DCCP-Sync, создавая пакеты нужного размера. Поскольку каждый корректный пакет DCCP-Sync вызывает незамедлительную генерацию ответного пакета DCCP-SyncAck, конечная точка сможет легко зафиксировать потерю пакета.

14.2. Поведение отправителя

Отправителю DCCP **следует** передавать пакеты нефрагментируемыми, как описано выше, за исключением следующих случаев.

---

[1] Maximum packet size.
[2] Congestion Control Maximum packet size.
[3] Path Maximum Transmission Unit - максимальный размер передаваемого блока для пути через сеть.
[4] Don't Fragment - не фрагментировать.

- На соединениях IPv4, где приложения запросили фрагментацию, отправителю **следует** передавать пакеты со сброшенным флагом DF.

- На соединениях IPv6, где приложения запросили фрагментацию, отправителю **следует** использовать специальное расширение для фрагментации пакетов, превышающих по размеру значение PMTU, в блоки подходящего размера (блоки, естественно, являются нефрагментируемыми).

- Нежелательно определять PMTU на этапе начального согласования параметров соединения, поскольку процесс организации соединения может не представлять размера пакетов, которые будут использоваться в соединении, а выполнение процедур определения MTU для пути через сеть может слишком затянуть процесс организации соединения. Таким образом, пакеты DCCP-Request и DCCP-Response **следует** передавать как фрагментируемые. Кроме того, пакеты DCCP-Reset **следует** передавать как фрагментируемые, хотя обычно они достаточно малы и не требуют фрагментации. Для соединений IPv4 такие пакеты **следует** передавать со сброшенным флагом DF, а для соединений IPv6 пакеты **следует** заранее фрагментировать до размера, не превышающего значение MTU для выходного интерфейса.

Если реализация DCCP снижает значение PMTU, а передающее приложение не запросило фрагментацию и размер пакета превышает новое значение MPS, интерфейс API **должен** отвергнуть передачу пакета и возвратить соответствующее сообщение об ошибке. Приложению после этого следует, используя API, запросить новое значение MPS. Ядро может содержать в буфере передачи пакеты, размер которых меньше старого значения MPS, но превышает новое значение. Эти пакеты **можно** передать как фрагментируемые или отбросить, но **недопустимо** передавать эти пакеты с запретом фрагментации.

15. Совместимость с будущими версиями

В будущих версиях протокола DCCP могут появляться новые опции и признаки. Приведённые ниже простые рекомендации позволят обеспечить совместимость расширенных версий DCCP с обычными реализациями DCCP.

- Для процессоров DCCP **недопустимо** использование карательных мер в отношении непонятных опций и признаков. Например, процессору DCCP **недопустимо** сбрасывать соединение, если некоторые поля, указанные в этой спецификации, как резервные, имеют отличные от 0 значения. Вместо этого процессор DCCP **должен** игнорировать такие поля. Обязательные опции или признаки являются единственным исключением - если опция Mandatory предшествует непонятной опции или признаку, соединение **должно** быть сброшено.

- Процессоры DCCP **должны** допускать возможность присутствия неизвестных значений, которые могут появляться при согласовании известных признаков. Для признаков, устанавливаемых с приоритетом сервера, неизвестные значения обрабатываются, как сами собой разумеющиеся - поскольку стандартный (не расширенный) список предпочтений DCCP не будет содержать неизвестных значений, результат обязательно будет из числа известных. Реализация DCCP **должна** возвращать пустую опцию Confirm при получении неизвестного значения для признака, не использующего согласования.

- Каждое расширение DCCP **следует** контролировать с помощью того или иного признака. По умолчанию для этого признака **следует** использовать значение extension not available[1]. Если расширенная реализация DCCP хочет использовать это расширение, ей **следует** предпринять попытку изменить значение этого признака с помощью опции Change L или Change R. Стандартные (не расширенные) реализации DCCP будут игнорировать опцию и для признака сохранится значение extension not available.

В главе 19 перечислены значения, выделенные для экспериментов и тестирования DCCP.

16. Промежуточные устройства

В данной главе описаны свойства DCCP, которые следует принимать во внимание (включая рассмотрение тех частей пакетов, которые не следует изменять в промежуточных устройствах) межсетевым экранам, системам трансляции адресов и иным промежуточным устройствам. Задачей является привлечение внимания к тем аспектам DCCP, которые могут быть полезны или опасны для промежуточных узлов, а также к существенным отличиям от протокола TCP.

Поле Service Code в пакетах DCCP-Request обеспечивает информацию, которая может быть полезна промежуточным устройствам, учитывающим состояние соединений. На основании Service Code промежуточная точка может определить используемый в соединении протокол, не опираясь на номера портов. Промежуточные устройства могут блокировать соединения, пытающиеся получить доступ к нежелательным службам, передавая пакет DCCP-Reset с Reset Code 8, Bad Service Code. Промежуточным узлам не следует менять значения поля Service Code, если они реально не подменяют службу, к которой обращается соединение.

Поля Source Port и Destination Port расположены в пакетах там же, где находятся соответствующие поля TCP и UDP, что может упрощать некоторые реализации промежуточных устройств.

Рассмотрение совместимости с будущими версиями в главе 15 применимо и к промежуточным устройствам. В частности, промежуточным устройствам в общем случае не следует использовать карательные меры по отношению к непонятным признакам и опциям.

Изменение полей Sequence Number и Acknowledgement Number в DCCP является более утомительным и опасным, нежели изменение порядковых номеров TCP. Промежуточные устройства, добавляющие или удаляющие пакеты из соединений DCCP, должны как минимум изменить опции подтверждения, такие, как Ack Vector, и связанные с CCID опции типа Loss Interval для TFRC. На поддерживающих ECN соединениях промежуточным устройствам нужно отслеживать информацию ECN Nonce для добавляемых или удаляемых пакетов, так, чтобы соответствующие опции подтверждений продолжали сохранять корректные значения ECN Nonce Echo, поскольку в противном случае соединение может быть сброшено за «агрессивное поведение». Мы, следовательно, рекомендуем промежуточным узлам не изменять поток пакетов путём добавления или удаления пакетов.

Отметим, что необходимость менять порядковые номера DCCP, основанные на учёте пакетов, возникает гораздо реже необходимости изменения порядковых номеров TCP, учитывающих байты; например, промежуточный узел может

[1]Расширение недоступно.

изменить содержимое пакета, не меняя его порядкового номера. В TCP изменение порядкового номера требуется для протоколов типа FTP, которые используют в потоке данных адреса переменной длины. Если такое приложение реализовать на базе DCCP, промежуточные точки могут просто увеличивать или уменьшать размер соответствующих пакетов без изменения их порядковых номеров. Это может относиться и к фрагментации пакетов.

Промежуточные точки могут, естественно, сбрасывать активные соединения. Очевидно, что для этого требуется вставка пакетов в один или оба потока данных, но сложностей тут не возникает.

Протокол DCCP не совсем дружелюбен к «слиянию соединений» [SHHP00], при котором клиентские попытки организации соединений перехватываются, но позднее могут «приращиваться» к соединениям с внешними серверами путём манипуляций с порядковыми номерами. Для слияния соединений нужно обеспечить по крайней мере согласие со всеми относящимися к делу признаками, что может потребовать повторного согласования.

Содержимое этой главы не следует воспринимать как рекламу промежуточных устройств с учётом состояния соединений.

17. Связь с другими спецификациями

17.1. RTP

Протокол RTP[1] [RFC3550] в настоящее время используется на базе транспорта UDP многими приложениями, подходящими для DCCP (например, потоковое вещание). Следовательно, важно рассмотреть отношения между протоколами DCCP и RTP и, в частности, вопрос о необходимости внесения изменений в RTP при переходе на транспорт DCCP взамен UDP.

Имеются два потенциальных источника дополнительных издержек при использовании RTP на основе DCCP - дублирование подтверждающей информации и дублирование порядковых номеров. Совместно они добавляют чуть больше 4 байтов на пакет по сравнению с использованием RTP на основе UDP и устранение избыточности не будет значительно сокращать издержки.

Рассмотрим сначала подтверждения. Оба протокола RTP и DCCP передают отправителю уведомления о скорости потери пакетов с помощью пакетов RTCP SR/RR[2] и опций подтверждения DCCP. Эти механизмы обратной связи являются в некоторой степени избыточными. Однако пакеты RTCP SR/RR содержат информацию, которой нет в подтверждениях DCCP (например, interarrival jitter[3]), а подтверждения DCCP содержат информацию, отсутствующую в RTCP (например, ECN Nonce Echo). Ни один из механизмов обратной связи не заменяет полностью другой механизм.

Передача обоих типов информации не повышает значительно расходы по сравнению с передачей одного (любого) типа. Отчёты RTCP могут передаваться сравнительно редко (в среднем 1 раз в течение 5 сек. для узкополосных потоков). В DCCP некоторые механизмы обратной связи достаточно «дороги» (опции Ack Vector, например, передаются часто и содержат много данных), а другие сравнительно дёшевы - подтверждения CCID 3 (TFRC) занимают 16 или 32 байта опций однократно в течение периода кругового обхода (более редкая передача отчётов снизит эффективность реакции механизма контроля насыщения на потери). Мы, следовательно, делаем вывод, что издержки на передачу подтверждений при использовании RTP на базе DCCP незначительно превышают издержки для случая RTP-over-UDP (по крайней мере, для CCID 3).

Одна явная избыточность может быть устранена на уровне приложения. Подробные уведомления о потере для каждого пакета, передаваемые в RTCP Extended Reports Loss RLE Block [RFC3611], могут быть получены из опций DCCP Ack Vector (обратное неверно, поскольку Loss RLE Block не содержит информации ECN). Поскольку реализациям DCCP следует обеспечивать API для доступа приложений к данным Ack Vector, приложения RTP на базе DCCP могут запрашивать опции DCCP Ack Vector или RTCP Extended Report Loss RLE Blocks, но не оба сразу.

Рассмотрим избыточность порядковых номеров в пакетах данных. Вложенный заголовок RTP содержит 16-битовый порядковый номер RTP. Большинство данных будет передаваться в пакетах DCCP-Data; пакеты DCCP-DataAck и DCCP-Ack обычно не требуются. Заголовок DCCP-Data имеет размер 12 байтов без учёта опций и включает 24-битовый порядковый номер. Это на 4 байта превышает размер заголовка UDP. Любые опции пакета будут дополнительно увеличивать издержки, хотя многие механизмы CCID (например, CCID 3, TFRC) не требуют использования опций в большинстве пакетов данных.

Порядковый номер DCCP невозможно получить из порядкового номера RTP, поскольку в DCCP нумерация увеличивается для каждого пакета, независимо от наличия в нем данных приложения. Порядковый номер RTP также невозможно получить из номера DCCP [RFC3550]. Более того, удаление порядкового номера RTP не даст экономии пространства заголовка, поскольку в том используется выравнивание. Мы, следовательно, рекомендуем при передаче трафика RTP на базе транспорта DCCP использовать стандартные заголовки. 4 дополнительных байта заголовка являются вполне приемлемой платой за механизмы контроля насыщения DCCP и доступ к ECN. Конечным точкам, которым действительно требуется снижение размера заголовков, следует воспользоваться той или иной схемой компрессии заголовков.

17.2. Congestion Manager и мультиплексирование

Поскольку DCCP не обеспечивает гарантированной доставки с сохранением порядка, множество субпотоков приложения может мультиплексироваться в одно соединение DCCP без снижения производительности. Следовательно, для DCCP не требуется обеспечивать поддержку для множества субпотоков. В этом протокол отличается от SCTP [RFC2960].

Некоторые приложения могут захотеть совместно использовать данные о состоянии контроля насыщения для множества потоков DCCP с одинаковыми адресами отправителя и получателя. Функционально это может обеспечить Congestion Manager [RFC3124] - базовый элемент мультиплексирования. Однако CM не будет полностью поддерживать DCCP без внесения изменений - например, он не может достаточно элегантно обслуживать множество механизмов контроля насыщения.

[1] Real-Time Transport Protocol — транспортный протокол для приложений, работающих в реальном масштабе времени. *Прим. перев.*
[2] RTP Control Protocol Sender/Receiver Report.
[3] Флуктуации интервала доставки пакетов.

18. Вопросы безопасности

DCCP не обеспечивает криптографической защиты. Приложениям, которым требуются криптографические методы защиты (целостность, аутентификация, конфиденциальность, контроль доступа и защита от повторного использования пакетов), следует применять IPsec или сквозные средства защиты иного типа; одним из таких вариантов может служить протокол Secure RTP [RFC3711].

Тем не менее, протокол DCCP подразумевает защиту от некоторых классов атак - атакующие не могут захватить соединение (неожиданно разорвать соединение или вынудить конечную точку принять данные атакующего взамен подлинной информации) без точного подбора порядковых номеров. Таким образом, при качественном выборе конечной точкой начального порядкового номера, атакующий DCCP должен сунуть свой нос в пакет данных для получения какой-либо вероятности успеха. Проверка корректности порядковых номеров обеспечивает надёжные гарантии. В параграфе 7.5.5 подробно рассматривается вопрос защиты порядковых номеров. Это средство защиты требует лишь выбора случайных чисел для DCCP в соответствии с рекомендациями [RFC4086].

DCCP также обеспечивает механизмы ограничения возможных последствий некоторых атак на отказ служб. Эти механизмы включают опции Init Cookie (параграф 8.1.4), пакеты DCCP-CloseReq (параграф 5.5), код отбрасывания Application Not Listening (параграф 11.7.2), ограничения на обработку опций, способных приводить к сбросу соединения (параграф 7.5.5), ограничения на обработку некоторых сообщений ICMP (параграф 14.1) и различные ограничения скорости, которые позволяют серверам избавиться от избыточных вычислений или генерации ненужных пакетов (параграфы 7.5.3, 8.1.3 и др.).

DCCP не обеспечивает защиты от атакующих, способных просматривать пакеты данных.

18.1. Вопросы безопасности для неполных контрольных сумм

Возможность использования неполных контрольных сумм оказывает своё влияние на безопасность и, в частности, на взаимодействие с механизмами аутентификации и шифрования. Это влияние для протокола DCCP такое же, как для UDP-Lite и позволяет адаптировать соответствующий текст из спецификации протокола UDP-Lite [RFC3828].

Когда пакеты DCCP содержат отличное от 0 значение поля Checksum Coverage, не учитываемая в контрольной сумме часть пакета может быть изменена в пути. Это вступает в противоречие с базовой идеей большинства механизмов аутентификации - считать аутентификацию успешной, если пакет не был изменён при передаче. До тех пор, пока не будет механизма аутентификации, способного работать только с частью пакета, аутентификация всегда будет давать сбои для пакетов DCCP с неполной контрольной суммой, если неучтенная в контрольной сумме часть пакета будет повреждена.

Проверка целостности IPsec (Encapsulation Security Protocol - ESP или Authentication Header - AH) применяется (как минимум) ко всем данным пакета IP. Повреждение любого бита данных будет приводить к отбрасыванию получателем IPsec пакета DCCP целиком, даже если повреждение связано с не учитываемой в контрольной сумме частью данных приложения DCCP.

Когда IPsec используется с шифрованием ESP для данных, канал не способен определить конкретный транспортный протокол пересылаемого пакета путём просмотра данных IP. В этом случае канал **должен** обеспечить стандартную проверку целостности, которая применяется ко всему пакету IP и содержащимся в нем данным. Неполные контрольные суммы DCCP в этом случае не дают никаких преимуществ.

Допускается использование шифрования (например, на транспортном или прикладном уровне). Отметим, что отказ от проверки целостности может в некоторых условиях приводить к риску потери конфиденциальности [B98].

Если несколько битов зашифрованного пакета повреждено, дешифровка обычно добавляет ошибок и пакет становится слишком повреждённым, чтобы его можно было использовать. Такое поведение характерно для многих современных механизмов шифрования. Существуют способы потокового шифрования, которые не приводят к распространению ошибки. Правильное применение потоковых шифров может быть достаточно сложным, особенно при отсутствии проверки аутентификации [BB01]. В частности, атакующий может внести предсказуемые искажения в получаемый после расшифровки текст, даже при отсутствии возможности дешифровки.

19. Согласование с IANA

Агентство IANA выделило для протокола DCCP значение IP Protocol Number = 33.

DCCP вводит восемь наборов чисел, значения которых должны распределяться IANA. Мы указываем правила распределения (такие, как Standards Action), описанные в [RFC2434], и большинство реестров резервирует часть значений для экспериментов и тестирования [RFC3692]. Кроме того, DCCP требует создания реестра IANA Port Numbers для регистрации портов DCCP (см. параграф 19.9). IANA может без ограничений обращаться к DCCP Expert Reviewer с вопросами по любым реестрам, независимо от принятой для них политики распределения для получения разъяснений или при возникновении проблем с распределением.

19.1. Реестр типов пакетов

Каждая запись в реестре DCCP Packet Types содержит типа пакета (целое число от 0 до 15), имя типа (например, DCCP-Request) и ссылку на RFC, в котором определён данный тип. Изначально реестр заполняется значениями из таблицы 3 (параграф 5.1). Данный документ определяет типы 0 - 9, а тип 14 резервируется для экспериментов и тестирования.

Типы пакетов 10 - 13 и 15 в настоящее время являются резервными. Распределение этих значений должно осуществляться на основе процедуры Standards Action, которая требует просмотра и одобрения IESG, а также публикации RFC со статусом standards-track.

19.2. Реестр кодов сброса

Каждая запись в реестре DCCP Reset Codes содержит код сброса (Reset Code), задаваемый целым числом от 0 до 255, краткое описание кода (например, No Connection) и ссылку на RFC, где определяется Reset Code. Изначально реестр

заполняется значениями из таблицы 3 (параграф 5.6). Данный документ определяет коды 0 – 11 и резервирует коды 120 - 126 для экспериментов и тестирования. Коды 12 - 119 и 127 в настоящее время являются резервными. Их следует распределять на основе процедуры IETF Consensus, требующей публикации RFC (статус standards track не требуется) с просмотром и одобрением IESG. Коды 128 - 255 резервируются для связанных с CCID реестров; каждый документ CCID Profile описывает управление соответствующим реестром.

19.3. Реестр типов опций

Каждая запись реестра опций DCCP содержит тип опции (целое число от 0 до 255), имя опции (например, Slow Receiver) и ссылку на RFC, где определён тип опции. Реестр изначально заполняется значениями из таблицы 5 (параграф 5.8). данный документ выделяет для типов опций значения 0 - 2 и 32 – 44, а опции типов 31 и 120 - 126 для экспериментов и тестирования. Опции типов 3 - 30, 45 – 119 и 127 в настоящее время зарезервированы. Их следует распределять на основе процедуры IETF Consensus, требующей публикации RFC (статус standards track не требуется) с просмотром и одобрением IESG. Опции типов 128 - 255 резервируются для связанных с CCID реестров; каждый документ CCID Profile описывает управление соответствующим реестром.

19.4. Реестр номеров признаков

Каждая запись в реестре признаков DCCP содержит номер признака (целое число от 0 до 255), его имя (например, ECN Incapable) и ссылку на RFC, где определяется признак. Реестр изначально заполняется значениями из таблицы 4 (параграф 6). Данный документ выделяет значения для признаков с номерами 0 - 9, а признаки 120 - 126 резервируются для экспериментов и тестирования. Признаки с номерами 10-119 и 127 являются резервными. Их следует распределять на основе процедуры IETF Consensus, требующей публикации RFC (статус standards track не требуется) с просмотром и одобрением IESG. Признаки с номерами 128 - 255 резервируются для связанных с CCID реестров; каждый документ CCID Profile описывает управление соответствующим реестром.

19.5. Реестр идентификаторов контроля насыщения

Каждая запись в реестре идентификаторов DCCP CCID содержит значение CCID (целое число от 0 до 255) имя механизма CCID (например, TCP-like Congestion Control) и ссылку на RFC, где определяется CCID. Реестр изначально заполняется значениями из таблицы 6 (глава 10). Значения CCID 2 и 3 выделены для опубликованных к настоящему моменту профилей, а значения 248 – 254 резервируются для экспериментов и тестирования. Значения CCID 0, 1, 4 – 247 и 255 в настоящее время являются резервными. Их следует распределять на основе процедуры IETF Consensus, требующей публикации RFC (статус standards track не требуется) с просмотром и одобрением IESG.

19.6. Реестр состояний Ack Vector

Каждая запись реестра DCCP Ack Vector States содержит Ack Vector State (целое число от 0 до 3), имя состояния (например, Received ECN Marked) и ссылку на RFC, где определяется состояние. Реестр изначально заполняется значениями из таблицы 7 (параграф 11.4). Данный документ определяет состояния 0, 1 и 3. Состояние с кодом 2 в настоящее время является резервным и должно распределяться на основании процедуры Standards Action, требующей просмотра и одобрения IESG, а также публикации RFC со статусом standards-track.

19.7. Реестр значений Drop Code

Каждая запись реестра DCCP Drop Codes содержит значение Data Dropped Drop Code 9целое число от 0 до 7) название Drop Code (например, Application Not Listening) и ссылку на RFC, где определяется Drop Code. Реестр изначально заполняется значениями из таблицы Ошибка: источник перекрёстной ссылки не найден (параграф 11.7). данный документ выделяет значения Drop Code 0 - 3 и 7. Коды 4 - 6 в настоящее время зарезервированы. Их следует распределять на основе процедуры Standards Action, требующей просмотра и одобрения IESG, а также публикации RFC со статусом standards-track.

19.8. Реестр кодов сервиса

Каждая запись в реестре Service Codes содержит значение Service Code (целое число от 0 до 4294967294); краткое описание сервиса на английском языке и может также включать ссылку на RFC или иной доступный документ, определяющий Service Code. В реестре следует указывать десятичное представление значений Service Code. Когда код может быть представлен в формате «SC:», согласно правилам параграфа 8.1.2, в реестре следует также указывать соответствующее представление в коде ASCII без префикса «SC:». Таким образом, число 1717858426 будет дополнительно представляться строкой «fdpz». Значения кодов сервиса не связаны исключительно с протоколом DCCP. Значение Service Code 0 зарезервировано (оно представляет отсутствие значимого кода сервиса), а значения 1056964608 – 1073741823 (старший байт соответствует символу ASCII «?») зарезервированы для приватного использования. Отметим, что значение 4294967295 не является корректным кодом сервиса. Большая часть оставшихся значений Service Code распределяется в порядке получения запросов (процедура First Come First Served), без обязательной публикации RFC; исключения перечислены в параграфе 8.1.2. данный документ выделяет одно значение Service Code 1145656131 (DISC). Это значение соответствует сервису discard, которые отбрасывает все полученные данные и ничего не передаёт в ответ.

19.9. Реестр номеров портов

Службы DCCP могут использовать контактные номера портов для предоставления услуг неизвестным абонентам, как это делается в TCP и UDP. Агентству IANA, следовательно, нужно открыть реестр номеров портов для DCCP, используя приведённые ниже правила, которые мы разработали с учётом существующих процедур регистрации номеров портов.

Номера портов делятся на три группы. Общеизвестные порты имеют номера от 0 до 1023, зарегистрированные порты - от 1024 до 49151, а динамически выделяемые и предназначенные для приватного использования - от 49152 до 65535. Общеизвестные и зарегистрированные номера портов предназначены для использования серверными приложениями, которым нужен принятый по умолчанию номер порта. В большинстве систем общеизвестные номера портов могут использоваться только системными (или принадлежащими пользователю root) процессами или программами,

запущенными от имени привилегированных пользователей, а зарегистрированные номера могут применяться обычными пользовательскими процессами и программами, запущенными от имени непривилегированных пользователей. Порты Dynamic и Private предназначены для временного использования, включая порты на клиентской стороне соединений, портов согласуемых независимо от соединения и тестирования приложений до регистрации для них выделенного порта. Следовательно, номера таких портов **недопустимо** регистрировать.

Для регистрации в реестре Port Numbers портов DCCP следует принимать номера из диапазонов Well Known и Registered. Порты Well Known и Registered Ports **не следует** использовать без регистрации. Хотя в некоторых случаях (например, при переносе UDP-приложений на DCCP) представляется естественным начать использование порта DCCP до завершения регистрации, мы подчёркиваем, что IANA не гарантирует регистрацию конкретного запрошенного значения для портов Well Known и Registered. Регистрацию номера порта следует запрашивать как можно раньше.

При каждой регистрации порта нужно указывать следующие данные:

- Краткое имя порта, содержащее только буквы латиницы (A-Z и a-z), цифры (0-9) и символы «-_+./*» (не включая кавычек).
- Номер порта, для которого запрашивается регистрация.
- Краткое описание назначения порта на английском языке. Это описание **должно** включать один или несколько разделённых пробелами текстовых дескрипторов Service Code, именующих порт с соответствующим значением Service Code (см. параграф 8.1.2).
- Имя и контактная информация персоны или организации, регистрирующей номер порта. По возможности следует указать также ссылку на документ, определяющий использование порта. Если регистрацией занимается рабочая группа IETF, достаточно указать название группы, но рекомендуется включать и контактную информацию.

Тем, кто регистрирует номер порта рекомендуется при подаче документов на регистрацию следовать приведённым ниже рекомендациям.

- Конкретное имя **не следует** регистрировать более, чем для одного порта DCCP.
- Имя порта, зарегистрированное для UDP **может** быть зарегистрировано и для DCCP. При такой регистрации **следует** использовать такой же номер порта, который зарегистрирован для данного имени в UDP.
- Перед регистрацией порта **следует** определить его конкретное назначение. Например, зарегистрированные порты UDP **не следует** бесцельно переносить в регистрацию портов для DCCP.
- Имена портов, связанные обычно с TCP и/или SCTP, **не следует** регистрировать для DCCP, поскольку имя такого порта косвенно предполагает транспорт с гарантированной доставкой. Например, мы рекомендуем не регистрировать порта с именем http для протокола DCCP. Однако, если подобная регистрация имеет смысл (т. е., имеется конкретное назначение для данного порта), при регистрации порта DCCP **следует** использовать тот же номер, который уже зарегистрирован для другого протокола.
- Допускается множественная регистрация для одного номера порта DCCP, если значения Service Code не совпадают.

Этот документ регистрирует один порт (в качестве модели регистрации).

```
discard    9/dccp    Discard SC:DISC
# IETF dccp WG, Eddie Kohler <kohler@cs.ucla.edu>, [RFC4340]
```

Сервис discard, который принимает соединения DCCP через порт 9, отбрасывает все полученные данные приложения и не передаёт никаких данных в ответ. Таким образом, порт discard в DCCP аналогичен порту TCP discard и может служить для проверки работоспособности стека DCCP.

20. Благодарности

Благодарим Jitendra Padhye за его помощь при подготовке ранних версий этой спецификации.

Спасибо Junwen Lai и Arun Venkataramani, которые, в качестве интернов ICIR, создавали прототип реализации DCCP. В частности, по рекомендации Junwen Lai старый механизм согласования признаков был забракован и заменён заново созданным механизмом. Отклики Arun Venkataramani позволили улучшить Приложение A.

Благодарим сотрудников и интернов ICIR и бывшего ACIRI, а также членов рабочих групп End-to-End Research и Transport Area за их отклики по DCCP. Особая благодарность просматривавшим документ экспертам - Greg Minshall, Eric Rescorla и Magnus Westerlund - за подробные комментарии и найденные проблемы, а также Rob Austein и Steve Bellovin за письменные и устные комментарии. Отдельно благодарим Aaron Falk, который руководил рабочей группой в процессе разработки спецификации.

Мы также благодарим тех, кто внёс свои комментарии и предложения через DCCP BOF, рабочую группу и почтовые конференции, включая Damon Lanphear, Patrick McManus, Colin Perkins, Sara Karlberg, Kevin Lai, Bernard Aboba, Youngsoo Choi, Pengfei Di, Dan Duchamp, Lars Eggert, Gorry Fairhurst, Derek Fawcus, David Timothy Fleeman, John Loughney, Ghyslain Pelletier, Hagen Paul Pfeifer, Tom Phelan, Stanislav Shalunov, Somsak Vanit-Anunchai, David Vos, Yufei Wang и Michael Welzl. В частности отметим многочисленные подробные отклики Colin Perkins, предложенную Michael Welzl опцию Data Checksum, отклики Gorry Fairhurst во вопросам контрольных сумм, а также модель Colored Petri Net [VBK05], предложенную Somsak Vanit-Anunchai, Jonathan Billington и Tul Kongprakaiwoot и позволившую обнаружить некоторые проблемы при обмене сообщениями.

Приложение A. Реализация Ack Vector

В этом приложении обсуждаются детали обработки подтверждений DCCP в контексте абстрактной реализации Ack Vector. Приложение является информативным, а не нормативным.

Первая часть нашей реализации работает на стороне HC-Receiver и, следовательно, подтверждает пакеты данных. Эта часть генерирует опции Ack Vector. Реализация имеет следующие характеристики:

- не более 1 байта состояния на пакет подтверждения;
- время O(1) затрачивается на обновление состояния при доставке нового пакета (нормальная ситуация);
- кумулятивные подтверждения;
- быстрое удаление старого состояния.

Базовая структура данных представляет собой кольцевой буфер, содержащий информацию о подтверждённых пакетах. Каждый байт этого буфера содержит состояние и групповой код (run length); состояние может принимать значения 0 (пакет получен), 1 (пакет имеет маркер ECN) или 3 (пакет ещё не получен). Начало буфера находится слева. Реализация поддерживает 5 переменных в дополнение к содержимому буфера:

- переменные buf_head и buf_tail показывают активную часть буфера;
- buf_ackno содержит значение Acknowledgement Number самого свежего подтверждённого пакета в буфере; этой переменной соответствует указатель head;
- buf_nonce представляет собой однобитовую сумму (чётность или «Исключающее ИЛИ») значений ECN Nonce, полученных во всех пакетах, подтверждаемых буфером со статусом State 0.

Вид буфера подтверждений показан на рисунке.

```
                  +-----------------------------------------------------------------+
                  |S,L|S,L|S,L|S,L|S,L|   |   |   |S,L|S,L|S,L|S,L|S,L|S,L|S,L|S,L|S,L|
                  +-----------------------------------------------------------------+
                            ^                   ^
                       buf_tail     buf_head, buf_ackno = A     buf_nonce = E

                     <=== buf_head и buf_tail перемещаются в этом направлении <===
```

Каждая пара «S,L» представляет байт State/Run length. Мы будем показывать только активную часть кольцевого буфера и добавлять комментарии, показывающие Acknowledgement Number для последнего активного байта в буфере. Пример показан ниже.

```
  +-----------------------------------------------+
A |S,L|S,L|S,L|S,L|S,L|S,L|S,L|S,L|S,L|S,L|S,L|S,L| T    BN[E]
  +-----------------------------------------------+
```

Здесь переменная buf_nonce имеет значение E, а buf_ackno = A. Будем использовать этот буфер в качестве рабочего примера.

```
   +---------------------------+
10 |0,0|3,0|3,0|3,0|0,4|1,0|0,0| 0    BN[1]   [Example Buffer]
   +---------------------------+
```

Словами содержимое буфера можно описать так:

пакет 10 был получен (начало буфера имеет порядковый номер 10, состояние 0 и run length 0);

пакеты 9, 8 и 7 ещё не получены (три следующих байта показывают состояние 3 и run length 0.)4

пакеты 6, 5, 4, 3 и 2 были получены;

пакет 1 имеет маркер ECN;

пакет 0 был получен.

Однобитовая сумма значений ECN Nonce для пакетов 10, 6, 5, 4, 3, 2 и 0 равна 1.

Кроме того HC-Receiver нужно сохранять некоторые данные о недавно переданных значениях Ack Vector. Для этого каждый пакет, содержащий Ack Vector, запоминается в 4 переменных:

- ack_seqno содержит значение Sequence Number, использованное для пакета; это порядковый номер HC-Receiver;
- ack_ptr - значение buf_head в момент подтверждения;
- ack_runlen - групповой код, находящийся в байте буфера данных с позицией buf_head в момент подтверждения;
- ack_ackno - значение Acknowledgement Number, использованное для пакета; это порядковый номер HC-Sender; поскольку подтверждения являются кумулятивными, один номер полностью задаёт всю информацию о пакете, подтверждаемом данным значением Ack Vector;
- ack_nonce - 1-битовая сумма ECN Nonce для всех пакетов со State 0 в буфере между позициями buf_head и ack_ackno, включительно; изначально эта переменная равна значению Nonce Echo в Ack Vector подтверждения (или, если пакет подтверждения содержит несколько опций Ack Vector, «исключающее ИЛИ» для всех Ack Vector); значение переменной меняется по мере удаления данных о старых подтверждениях (сближение ack_ptr и buf_head) и прибытия старых пакетов (смена State 3 или State 1 на State 0).

A.1. Получение пакетов

В этом параграфе рассматривается, как HC-Receiver обновляет буфер подтверждений по мере доставки пакетов от HC-Sender.

A.1.1. Новые пакеты

Когда прибывает пакет с Sequence Number > buf_ackno, HC-Receiver обновляет buf_head (перемещая соответствующим образом влево), buf_ackno (сюда помещается значение Sequence Number из нового пакета), и,

возможно, buf_nonce (в зависимости от ECN Nonce в принятом пакете) в дополнение к обновлению самого буфера. Например, если пакет 11 от HC-Sender прибыл с маркером ECN, показанный выше Example Buffer будет переходить в состояние, показанное на рисунке справа (изменения отмечены звёздочками).

```
** +***---------------------------+
11 |1,0|0,0|3,0|3,0|3,0|0,4|1,0|0,0| 0    BN[1]
** +***---------------------------+
```

Если состояние пакета совпадает с состоянием в голове буфера, HC-Receiver может увеличить значение run length (вплоть до максимального). Например, если пакет 11 от HC-Sender прибудет без маркера ECN и со значением ECN Nonce 0, Example Buffer может перейти в состояние, показанное на рисунке.

```
** +--*-----------------------+
11 |0,1|3,0|3,0|3,0|0,4|1,0|0,0| 0    BN[1]
** +--*-----------------------+
```

Порядковый номер вновь прибывшего пакета может, естественно, не совпасть с ожидаемым номером. В таких случаях HC-Receiver будет вводить для пропущенных пакетов состояние State 3. Если отсутствует несколько пакетов, HC-Receiver можно ввести множество байтов с run length = 0 или один байт с соответствующим значением группового кода. Использование отдельного байта для каждого пропущенного пакета упрощает обновление при поступлении какого-либо из пропущенных пакетов. Например, если пакет 12 от HC-Sender прибудет с ECN Nonce 1, Example Buffer перейдёт в состояние, показанное на рисунке.

```
** +*******-----------------------------+         *
12 |0,0|3,0|0,1|3,0|3,0|3,0|0,4|1,0|0,0| 0    BN[0]
** +*******-----------------------------+         *
```

Кольцевой буфер может переполняться, когда HC-Sender передаёт данные с очень высокой скоростью, а подтверждения от HC-Receiver не успевают доходить до HC-Sender или тот забывает подтверждать эти подтверждения (в результате HC-Receiver не может сбросить старые состояния). В этом случае полусоединению HC-Receiver следует сжать буфер (увеличивая, по возможности, значения run length), переместить состояния в буфер большего размера или, в качестве крайней меры, отбрасывать все принятые пакеты без их обработки, пока буфер не освободится.

A.1.2. Старые пакеты

Когда прибывает пакет с Sequence Number S <= buf_ackno, HC-Receiver будет просматривать таблицу в поисках байта, соответствующего S (структуры с индексированием могут упростить такой поиск). Если номер S ранее считался потерянным (State 3) и указан в байте с run length 0, HC-Receiver может просто изменить состояние для этого байта. Например, если пакет в от HC-Sender поступил с ECN Nonce 0, Example Buffer будет иметь вид, показанный на рисунке.

```
   +--------*------------------+
10 |0,0|3,0|0,0|3,0|0,4|1,0|0,0| 0    BN[1]
   +--------*------------------+
```

Если пакет S не был указан среди потерянных или отсутствует в таблице, пакет возможно является возникшим в сети дубликатом и его следует игнорировать (состояние маркера ECN для нового пакета может отличаться от состояния, указанного в буфере; описание таких ситуаций приведено в параграфе 11.4.1). Если для номера S в буфере используется группа с отличным от нуля значением run length, может потребоваться перераспределение буфера с целью освобождения места для одного или двух байтов.

Могут также потребоваться операции с полями ack_nonce при доставке старого пакета. В частности, когда S переходит из состояния 3 или 1 в состояние 0 и S имеет ECN Nonce 1, реализации следует изменить значение битов ack_nonce для всех подтверждений с ack_ackno >= S.

При использовании описанной структуры данных невозможна смена состояния 0 на состояние 1, поскольку буфер не сохраняет отдельных значений ECN Nonce.

A.2. Отправка подтверждений

Когда получателю HC-Receiver нужно генерировать подтверждение, содержимое буфера можно просто скопировать в одну или несколько опций Ack Vector. Копирование Ack Vector может не обеспечивать максимального сжатия. Например, показанный выше Example Buffer содержит три байта 3,0 подряд, которые можно объединить в один байт 3,2. HC-Receiver может, следовательно, выполнять сжатие буфера перед копированием или в процессе копирования.

В каждое подтверждение, передаваемое получателем HC-Receiver **следует** включать все состояния из буфера. Таким образом, подтверждения являются кумулятивными.

Если подтверждение помещается в одну опцию Ack Vector, значение Nonce Echo этой опции просто будет равно buf_nonce. Для множества опций Ack Vector требуется больше операций. Значения Ack Vector следует расщепить в точках, соответствующих предыдущим подтверждениям, поскольку сохранённые поля ack_nonce обеспечивают достаточно информации для расчёта корректных значений Nonce Echo. Реализации **следует**, подтверждать данные по крайней мере один раз на 253 байта буферного пространства (иначе не будет возможности расчёта Nonce Echo).

Для каждого передаваемого подтверждения HC-Receiver будет добавлять новую запись. Значение ack_seqno будет равно порядковому номеру HC-Receiver, использованному для пакета подтверждения; ack_ptr = buf_head; значение ack_runlen будет равно значению run length из байта buf_head; ack_ackno = buf_ackno; ack_nonce will = buf_nonce.

A.3. Очистка состояния

Некоторые из пакетов от HC-Sender будут включать номера подтверждений, которые подтверждают подтверждения HC-Receiver. При получении такого подтверждения HC-Receiver находит запись R с соответствующим значением ack_seqno и выполняет следующие операции:

- если значение run length в байте R.ack_ptr byte превышает R.ack_runlen, значение run length уменьшается на R.ack_runlen + 1 и устанавливается buf_tail = R.ack_ptr; иначе устанавливается buf_tail = R.ack_ptr + 1;
- если R.ack_nonce = 1, меняется значение бита buf_nonce и ack_nonce для всех последующих записей о подтверждениях;
- удаляется запись R и все предшествующие ей записи.

(HC-Receiver может сохранять часть старой информации на случай получения пакетов, считавшихся потерянными). Предположим, что получатель HC-Receiver, сохраняющий Example Buffer, уже передал 2 подтверждения:

1. ack_seqno = 59, ack_runlen = 1, ack_ackno = 3, ack_nonce = 1.

2. ack_seqno = 60, ack_runlen = 0, ack_ackno = 10, ack_nonce = 0.

Далее предположим, что HC-Receiver получает от HC-Sender пакет DCCP-DataAck с Acknowledgement Number 59. Этот пакет говорит получателю HC-Receiver, что отправитель HC-Sender принял и обработал всю информацию из пакета от HC-Receiver с номером 59. Этот пакет подтверждает полученный от HC-Sender пакет 3 и HC-Sender имеет от HC-Receiver подтверждения для пакетов 0, 1, 2 и 3. Вид буфера Example Buffer показан на рисунке.

```
                +------------------*+ *       *
             10 |0,0|3,0|3,0|3,0|0,2| 4    BN[0]
                +------------------*+ *       *
```

Значение run length для «хвостового» байта было изменено, поскольку пакет 3 был учтен в этом байте. Поскольку значение R.ack_nonce было равно 1, значение бита buf_nonce изменяется, как и значения битов ack_nonce для последующих подтверждений (в данном случае запись HC-Receiver Ack 60 не показана; для неё значение ack_nonce меняется на 1). HC-Receiver может также удалить сохранённую информацию для HC-Receiver Ack 59 и всех предшествующих подтверждений.

Осторожная реализация может предпринять попытку обеспечения разумной устойчивости к смене порядка доставки. Воспользуемся снова примером Example Buffer, предположив, что пакет 9 приходит с нарушениям порядка доставки. Вид буфера для этого случая показан на рисунке.

```
                +----*----------------------+
             10 |0,0|0,0|3,0|3,0|0,4|1,0|0,0| 0   BN[1]
                +----*----------------------+
```

Опасность заключается в том, что HC-Sender может подтвердить предыдущее подтверждение от HC-Receiver (номер 60), которое говорит о том, что пакет 9 не был получен, до того, как HC-Receiver получит шанс на передачу нового подтверждения, указывающего получение пакета 9. Следовательно, по получении пакета 9 HC-Receiver может изменить подтверждающую запись, как показано ниже:

1. ack_seqno = 59, ack_ackno = 3, ack_nonce = 1.

2. ack_seqno = 60, ack_ackno = 3, ack_nonce = 1.

Т. е., пакет Ack 60 сейчас трактуется подобно дубликату пакета Ack 59. Это будет предотвращать перемещение хвоста буфера за пакет 9, пока HC-Receiver не узнает, что полусоединение HC-Sender увидело вектор Ack Vector, показывающий доставку пакета.

A.4. Обработка подтверждений

При получении подтверждения HC-Sender в общем случае следует внимательно относится к номерам пакетов, которые были отброшены и/или помечены маркером ECN. Эта информация считывается из опций Ack Vector. Кроме того, следует проверить корректность ECN Nonce (как было сказано в параграфе 11.4.1, может возникнуть потребность в сохранении более детальной информации о подтверждённых пакетах на случай смены состояния пакетов между подтверждениями или получения запроса о доставке пакета от прикладной программы).

HC-Sender может также подтверждать подтверждения от HC-Receiver для удаления старых состояний Ack Vector (поскольку подтверждения Ack Vector доставляются гарантированно, получатель HC-Receiver должен поддерживать информацию Ack Vector и повторно передавать её, пока не будет уверенности, что она получена HC-Sender). Достаточно использования простого алгоритма - поскольку подтверждения Ack Vector являются кумулятивными, один номер подтверждения говорит HC-Receiver, как много подтверждающих данных было получено. В предположении что HC-Receiver не передаёт данных, полусоединение HC-Sender может гарантировать, что по крайней мере 1 раз за период кругового обхода оно будет передавать пакет DCCP-DataAck, подтверждающий приём последнего пакета DCCP-Ack. Естественно, что полусоединению HC-Sender требуется подтверждать подтверждения от HC-Receiver только в том случае, когда HC-Sender передаёт также данные. Если HC-Sender не передаёт данных, состояние Ack Vector для HC-Receiver будет неизменным и не возникает необходимости в его сжатии. Отправитель HC-Sender должен отслеживать отбрасывание и маркировку ECN в полученных пакетах DCCP-Ack, чтобы можно было корректировать скорость передачи (например, с помощью Ack Ratio) подтверждений от HC-Receiver в случаях возникновения перегрузок.

Если другое полусоединение не является статичным (т. е., HC-Receiver передаёт данные в направлении HC-Sender, возможно используя другой механизм CCID), подтверждений в этом полусоединении достаточно будет для очистки состояния HC-Receiver.

Приложение B. Применение неполных контрольных сумм

Было много дискуссий по поводу утилиты, позволяющей отправителю DCCP ограничивать область покрытия для контрольной суммы, чтобы не учитывать все данные из пакета. В этом приложении собраны некоторые соображения по данному вопросу.

Многие из приложений, которые предполагается реализовать на основе DCCP, обеспечивают устойчивость к некоторому уровню потери данных или используют транспорт с гарантированной доставкой. Некоторые из этих приложений (например, аудио) устойчивы также к повреждению данных. Такие устойчивые к ошибкам приложения предпочтут получить от DCCP повреждённые пакеты, нежели эти пакеты будут просто отбрасываться протоколом. Это связано отчасти с контролем насыщения - DCCP не может указать разницу между пакетами, отброшенными в результате повреждения и перегрузок, поэтому протокол должен снижать скорость передачи при любых потерях. Такое поведение может привести к снижению полосы соединения до неприемлемых значений. Для некоторых сетевых технологий повреждение данных не зависит от загрузки сети или, по крайней мере, не всегда коррелирует с уровнем насыщения. Следовательно, повреждение пакетов (если заголовок и опции DCCP не повреждены) не должно вызывать существенного снижения скорости передачи, которое может вызываться механизмом контроля насыщения.

Протокол DCCP позволяет учитывать в контрольной сумме весь пакет, только заголовок DCCP или заголовок и часть данных приложения. Если приложение неустойчиво к частичному повреждению данных, следует использовать контрольную сумму для всего пакета. Если же приложение предпочитает доставку частично повреждённых данных полному отбрасыванию таких пакетов, контрольная сумма может учитывать только часть пакета (с обязательным включением заголовка DCCP). Кроме того, если приложение хочет отделить контрольную сумму заголовка DCCP от контрольной суммы данных, оно может сделать это с помощью опции Data Checksum. Это позволяет DCCP отбрасывать повреждённые данные приложения без нарушения работы системы контроля насыщения.

Таким образом, с точки зрения приложений использование неполных контрольных сумм представляется привлекательным. Однако польза от применения неполных контрольных сумм будет будет зависеть от того, как частично повреждённые пакеты будут доставляться получателю. Если контрольные суммы канального уровня (CRC) всегда ведут к отбрасыванию повреждённых пакетов, польза от применения неполных контрольных сумм будет только для случаев повреждения пакетов в маршрутизаторах и иных местах, не контролируемых на основе CRC канального уровня. Не удалось прийти к согласию по оценке вероятности того, что новые сетевые технологии для каналов с высоким уровнем ошибок будут использовать строгую контрольную сумму CRC для всего пакета. DCCP может адаптировать приложения к использованию таких каналов, но сложно предсказать, насколько это будет актуально для будущих технологий канального уровня.

Кроме того, неполные контрольные суммы несовместимы с механизмами аутентификации на уровне IP (такими, как IPsec AH), которые используют криптографические хэш-функции для всего пакета. В результате, если требуется одновременно с неполными контрольными суммами обеспечивать криптографическую аутентификацию, последняя должна выполняться для данных приложения. Возможным вариантом будет появление протокола, подобного Secure RTP. Однако такая аутентификация на «прикладном уровне» не защищает согласование опций и машину состояний DCCP от обманных пакетов. Альтернативой может служить использование IPsec ESP и шифрование заголовков DCCP для защиты от атак с проверкой корректности заголовков DCCP для аутентификации источника пакета, содержащего заголовок (владелец корректного ключа). Такая схема защищает от повторного использования пакетов (благодаря порядковым номерам DCCP), она не обеспечивает достаточной защиты от некоторых MITM-атак[1], поскольку данные пакета не связаны с его заголовком. Таким образом, аутентификация на уровне приложения может потребовать связи с IPsec ESP или иным механизмом, обеспечивающим комплексное защитное решение. Связанные с этим издержки могут оказаться неприемлемыми для некоторых приложений, которым подходят неполные контрольные суммы.

В целом авторы надеются, что неполные контрольные суммы DCCP позволят использовать некоторые функции, реализация которых без этого может оказаться затруднительной. Хотя с поддержкой неполных контрольных сумм не связано значительных расходов и сложностей, их применение в настоящее время не представляется целесообразным. Практика покажет насколько неполные контрольные суммы полезны на самом деле.

Нормативные документы

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3309] Stone, J., Stewart, R., and D. Otis, "Stream Control Transmission Protocol (SCTP) Checksum Change", RFC 3309, September 2002.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, January 2004.

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.

Дополнительная литература

[B98] Bellovin, S. M., "Cryptography and the Internet", CRYPTO '98 (LNCS 1462), pp 46-55, August 1988.

[BB01] Bellovin, S.M. and M. Blaze, "Cryptographic Modes of Operation for the Internet", 2nd NIST Workshop on Modes of Operation, August 2001.

[M85] Morris, R. T., "A Weakness in the 4.2BSD Unix TCP/IP Software", Computer Science Technical Report 117[2], AT&T Bell Laboratories, Murray Hill, NJ, February 1985.

[PMTUD] Mathis, M. and J. Heffner, "Path MTU Discovery", Work in Progress[3], March 2006.

[RFC792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1948] Bellovin, S., "Defending Against Sequence Number Attacks", RFC 1948, May 1996.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

[1] man-in-the-middle attack - атака основанная на перехвате пакетов с участием человека. *Прим. перев.*
[2] Копию статьи можно найти на сайте pdos.csail.mit.edu/~rtm/papers/117.pdf, а перевод доступен на сайте. *Прим. перев.*
[3] Работа опубликована в RFC 4821. *Прим. перев.*

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, October 1996.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2463] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

[RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.

[RFC3360] Floyd, S., "Inappropriate TCP Resets Considered Harmful", BCP 60, RFC 3360, August 2002.

[RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.

[RFC3540] Spring, N., Wetherall, D., and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces", RFC 3540, June 2003.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC4086] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, March 2006.

[RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, March 2006.

[SHHP00] Spatscheck, O., Hansen, J.S., Hartman, J.H., and L.L. Peterson, "Optimizing TCP Forwarder Performance", IEEE/ACM Transactions on Networking 8(2):146-157, April 2000.

[SYNCOOKIES] Bernstein, D.J., "SYN Cookies", http://cr.yp.to/syncookies.html (по состоянию на март 2006).

[VBK05] Vanit-Anunchai, S., Billington, J., and T. Kongprakaiwoot, "Discovering Chatter and Incompleteness in the Datagram Congestion Control Protocol", FORTE 2005, pp 143-158, October 2005.

Адреса авторов

Eddie Kohler
4531C Boelter Hall
UCLA Computer Science Department
Los Angeles, CA 90095
USA
EMail: kohler@cs.ucla.edu

Mark Handley
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
UK
EMail: M.Handley@cs.ucl.ac.uk

Sally Floyd
ICSI Center for Internet Research
1947 Center Street, Suite 600
Berkeley, CA 94704
USA
EMail: floyd@icir.org

Перевод на русский язык
Николай Малых
nmalykh@protokols.ru

Полное заявление авторских прав

Интеллектуальная собственность

IETF не принимает какой-либо позиции в отношении действительности или объема каких-либо прав интеллектуальной собственности (Intellectual Property Rights или IPR) или иных прав, которые, как может быть заявлено, относятся к

реализации или использованию описанной в этом документе технологии, или степени, в которой любая лицензия, по которой права могут или не могут быть доступны, не заявляется также применение каких-либо усилий для определения таких прав. Сведения о процедурах IETF в отношении прав в документах RFC можно найти в BCP 78 и BCP 79.

Копии раскрытия IPR, предоставленные секретариату IETF, и любые гарантии доступности лицензий, а также результаты попыток получить общую лицензию или право на использование таких прав собственности разработчиками или пользователями этой спецификации, можно получить из сетевого репозитория IETF IPR по ссылке http://www.ietf.org/ipr.

IETF предлагает любой заинтересованной стороне обратить внимание на авторские права, патенты или использование патентов, а также иные права собственности, которые могут потребоваться для реализации этого стандарта. Информацию следует направлять в IETF по адресу ietf-ipr@ietf.org.

Подтверждение

Финансирование функций RFC Editor обеспечено IETF Administrative Support Activity (IASA).