Протокол DCCP
Datagram Congestion Control Protocol (DCCP)

Статус документа
В этом документе содержится спецификация протокола, предложенного сообщству 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
Copyright (C) The Internet Society (2006).

Тезисы
DCCP 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 объемы данных и могут получить преимущества в результате контроля выбора между своевременностью и надежностью.

Оглавление

1. Введение............................................................................................................................................................................. 3
2. Обоснование задач.......................................................................................................................................................... 4
3. Уровни требований.......................................................................................................................................................... 4
3.1. Числа и поля............................................................................................................................................................. 4
3.2. Участники соединения........................................................................................................................................... 5
3.3. Признаки.................................................................................................................................................................. 5
3.4. Время кругового обхода......................................................................................................................................... 5
3.5. Ограниченность защиты........................................................................................................................................ 5
3.6. Отказоустойчивость............................................................................................................................................... 5
4. Обзор.............................................................................................................................................................................. 6
4.1. Типы пакетов............................................................................................................................................................ 6
4.2. Упорядочивание пакетов..................................................................................................................................... 6
4.3. Состояния.............................................................................................................................................................. 7
4.4. Механизмы контроля насыщения....................................................................................................................... 7
4.5. Опции согласования признаков.......................................................................................................................... 8
4.6. Отличия от TCP.................................................................................................................................................... 8
4.7. Пример соединения............................................................................................................................................... 9
5. Формат пакетов.......................................................................................................................................................... 9
5.1. Базовый заголовок................................................................................................................................................. 10
5.2. Пакеты DCCP-Request................................................................................................................................„.. 11
5.3. Пакеты DCCP-Response................................................................................................................................... 11
5.4. Пакеты DCCP-Data, DCCP-Ack и DCCP-DataAck............................................................................................... 12
5.5. Пакеты DCCP-CloseReq и DCCP-Close.............................................................................................................. 12
5.6. Пакеты DCCP-Reset............................................................................................................................................. 12
5.7. Пакеты DCCP-Sync и DCCP-SyncAck................................................................................................................. 14
5.8. Опции.................................................................................................................................................................. 14
5.8.1. Опция Padding.................................................................................................................................................. 15
5.8.2. Опция Mandatory....................................................................................................................................... 15
6. Согласование признаков........................................................................................................................................ 15
6.1. Опции Change....................................................................................................................................................... 16
6.2. Опции Confirm................................................................................................................................................... 16
6.3. Правила согласования........................................................................................................................................... 16
6.3.1. Приоритет сервера........................................................................................................................................... 16
6.3.2. Согласование не используется...................................................................................................................... 16
6.4. Номера признаков............................................................................................................................................ 16
6.5. Примеры согласования признаков .................................................. 17
6.6. Обмен опциями ................................................................. 17
6.6.1. Нормальный обмен ....................................................... 18
6.6.2. Обработка полученных опций ......................................... 18
6.6.3. Потеря и повторная передача пакетов ......................... 19
6.6.4. Нарушение порядка доставки ........................................ 19
6.6.5. Смена предпочтений ..................................................... 20
6.6.6. Одновременное согласование ........................................ 20
6.6.7. Неизвестные признаки ................................................... 20
6.6.8. Некорректные опции .................................................... 20
6.6.9. Согласование обязательных признаков ......................... 20
7. Порядковые номера ............................................................. 21
7.1. Переменные ................................................................. 21
7.2. Сигналы ECN Nonce ....................................................... 21
7.3. Пауза (Quiet Time) ........................................................ 22
7.4. Номера подтверждений ..................................................... 22
7.5. Корректность номеров и синхронизация ............................. 22
7.5.1. Окна порядковых номеров и номеров подтверждений .......... 22
7.5.2. Признак Sequence Window ........................................... 23
7.5.3. Правила проверки корректности номеров ..................... 23
7.5.4. Обработка пакетов с некорректными номерами ................. 24
7.5.5. Атаки на порядковые номера ......................................... 24
7.5.6. Примеры обработки порядковых номеров ..................... 25
7.6. Короткие порядковые номера ............................................. 26
7.6.1. Признак Allow Short Sequence Numbers ......................... 26
7.6.2. Когда не следует использовать короткие порядковые номера ........................................ 26
7.7. Опция NDP Count и детектирование потери данных приложения ........................................ 26
7.7.1. Использование NDP Count ........................................... 27
7.7.2. Признак Send NDP Count ............................................. 27
8. Обработка событий ............................................................ 27
8.1. Организация соединения .................................................... 27
8.1.1. Запрос клиента ........................................................ 27
8.1.2. Коды сервиса ........................................................... 28
8.1.3. Отклик сервера ........................................................ 28
8.1.4. Опция Init Cookie ..................................................... 29
8.1.5. Завершение согласования ............................................. 29
8.2. Передача данных .......................................................... 30
8.3. Разрыв соединения ....................................................... 30
8.3.1. Аварийное завершение .............................................. 31
8.4. Диаграмма состояний DCCP ........................................... 31
8.5. Псевдокод ................................................................. 31
9. Контрольные суммы .......................................................... 34
9.1. Поле Checksum в заголовке ............................................. 34
9.2. Поле заголовка Checksum Coverage .............................. 34
9.2.1. Признак Minimum Checksum Coverage ....................... 35
9.3. Опция Data Checksum .................................................. 35
9.3.1. Признак Check Data Checksum ................................... 35
9.3.2. Использование контрольных сумм ................................ 35
10. Контроль насыщения ....................................................... 36
10.1. Контроль насыщения в стиле TCP ................................. 36
10.2. Контроль насыщения TFRC ........................................ 36
10.3. Опции, признаки и коды сброса, связанные с CCID ............ 36
10.4. Требования к профилям CCID ....................................... 37
10.5. Состояние насыщения .................................................. 37
11. Подтверждения ............................................................... 38
11.1. Подтверждения подтверждений и односторонние соединения ........................................ 38
11.2. Добавление подтверждений ............................................ 38
11.3. Признак Ack Ratio ..................................................... 39
11.4. Опции Ack Vector ..................................................... 39
11.4.1. Согласованность Ack Vector ..................................... 40
11.4.2. Покрытие Ack Vector .............................................. 41
11.5. Признак Send Ack Vector ............................................ 41
11.6. Опция Slow Receiver .................................................. 41
11.7. Опция Data Dropped ................................................... 42
11.7.1. Отклик на Data Dropped и обычный отклик на перегрузку ........................................ 43
11.7.2. Отдельные значения Drop Code .................................. 43
12. Явное уведомление о насыщении ........................................ 44
12.1. Признак ECN Incapable ............................................... 44
12.2. Сигналы ECN Incapable .............................................. 44
12.3. Назначения за агрессию .............................................. 45
13. Опции синхронизации ....................................................... 45
13.1. Опция Timestamp ..................................................... 45
13.2. Опция Elapsed Time .................................................. 45
13.3. Опция Timestamp Echo ............................................. 46
14. Максимальный размер пакета ............................................ 46
14.1. Измерение PMTU ..................................................... 46
14.2. Поведение отправителя ............................................... 47
15. Совместимость с будущими версиями ................................... 47
16. Промежуточные устройства.................................................................48
17. Связь с другими спецификациями.....................................................48
17.1. RTP.................................................................................................48
17.2. Congestion Manager и мультиплексирование...............................49
18. Вопросы безопасности.......................................................................49
18.1. Вопросы безопасности для неполных контрольных сумм..............49
19. Согласование с IANA..........................................................................50
19.1. Реестр типов пакетов.................................................................50
19.2. Реестр кодов сброса.................................................................50
19.3. Реестр типов опций.................................................................50
19.4. Реестр номеров признаков..........................................................50
19.5. Реестр идентификаторов контроля насыщения............................50
19.6. Реестр состояний Ack Vector.....................................................50
19.7. Реестр значений Drop Code.......................................................51
19.8. Реестр кодов сервиса.................................................................51
19.9. Реестр номеров портов..............................................................51
20. Благодарности......................................................................................52
Приложение А. Реализация Ack Vector................................................52
A.1. Получение пакетов........................................................................53
A.1.1. Новые пакеты...........................................................................53
A.1.2. Старые пакеты..........................................................................53
A.2. Отправка подтверждений..............................................................54
A.3. Очистка состояния.........................................................................54
A.4. Обработка подтверждений.............................................................54
Приложение В. Обоснование применения неполных контрольных сумм ..................................................................................55
Нормативные документы........................................................................55
Дополнительная литература.....................................................................56

Список таблиц
Таблица 1: Типы пакетов DCCP ...............................................................11
Таблица 2: Коды DCCP Reset.................................................................15
Таблица 3: Опции DCCP.........................................................................16
Таблица 4: Номера признаков DCCP....................................................18
Таблица 5: Идентификаторы механизмов контроля насыщения DCCP 39
Таблица 6: Состояния DCCP Ack Vector...............................................43
Таблица 7: Коды отбрасывания пакетов DCCP ...................................45

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 неподтверждённых попыток организации соединений и завершённых соединений;
- контроль насыщения, включающий механизмы ECN1 [RFC3168] и ECN Nonce [RFC3540];
- механизм подтверждений, обеспечивающий сигналы о потере пакетов и информацию ECN; подтверждения передаются с таким уровнем надежности, который требуется механизму контроля насыщения (включая полную гарантию доставки);
- дополнительные механизмы, обеспечивающие (с высоким уровнем надежности) передающее приложение информацией о доставке пакетов получателю, установке маркеров ECN, повреждении или отбрасывании пакетов;
- механизм определения PMTU2 [RFC1191].
- выбор модулей механизма контроля насыщения; в настоящее время поддерживаются два механизма - контроль насыщения в стиле TCP3 [RFC4341] и TFR4 [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 преимущество за счет возможности выбора между высокой задержкой и надёжной доставкой с соблюдением порядка. Протокол TCP не подходит для таких приложений, поскольку используемый в нем механизм контроля насыщения и обеспечения гарантий упорядоченной доставки может 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.

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.
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 коррекции ошибок (FEC1), частичных гарантий доставки2 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 остаться стабильность скорости передачи данных (стремительное изменение скорости может вызывать нежелательные сбои на уровне пользовательского интерфейса типа паузы 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. Используя этот механизм приложения могут настраивать размер окна насыщения, подобно тому как это делается в приложениях, использующих протокол TCP. Второй вариант - TCP-Friendly Rate Control (TFRC) [RFC3448] – представляет собой основанный на выравнивании механизма 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 API3 в адре 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 CM4 [RFC3124], позволяющий поддерживать между отправителем и получателем множество потоков с единым контролем насыщения. Существующий механизм CM может 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 развертывания 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, применявшиеся для DCCP и описанные в отдельных документах (Congestion Control ID Profile5) [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 (SHOULD 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, поряжковое число и аргументы опций) передаются с использованием сетевого порядка следования байтов6 (сначала передается 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 учетом рекомендаций [RFC4086].

Все операции с порядковыми номерами в DCCP используют циклическую арифметику по модулю 2^48 при сравнении чисел поддерживаемых. 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 в TCP и DNS [RFC1982]. Имеет смысл хранить порядковые номера DCCP в старших 48 битах 64-битовых целых чисел, заполняя младшие 16 бит нулями. Такой подход согласуется с общепринятым методом сравнения для циклической арифметики, когда проверка выполнения условия A < B выполняется путем 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].

1Forward error correction.
2Semi-reliability.
3Application Programming Interface.
4Congestion Manager - менеджер насыщения (перегрузки).
5Congestion Control ID Profile.
6Network byte order.
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 общем случае обозначает любой хост, от которого может потребоваться обработка заголовков DCCP, т. е. включает конечные точки соединения и промежуточные узлы (middlebox) типа МСЭ 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.

В контексте DCCC термин 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 соединений, согласованные двумя конечными точками, называются признаками протокола DCCP. Многие параметры соединений DCCP, включая механизмы контроля насыщения, используемые для пары полусоединений, определяются признаками. Конечные точки согласуют признаки путём обмена опциями согласования 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» для всех признаков F/A и «удаленной стороной» 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, описанное в терминах времени кругового обхода, реально относится к оценке RTT, сделанной неким механизмом CCID, или (при отсутствии возможности оценки) принятому по умолчанию значению времени кругового обхода.

Максимальное время жизни сегмента MSL - 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 путем 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 принципу в TCP общему принципу отказоустойчивости - «be conservative in what you do, be liberal in what you accept from others» [RFC793].

4. Обзор

Динамика соединений вышележащих уровней для DCCP в точности соответствует протоколу TCP. Каждое соединение состоит из трех фаз – инициирование соединения, включая трехэтапное согласование, передача данных и разрыв

1Межсетевой экран – firewall. Прим. перев.
2DCCP feature – признак (свойство, характерная черта) DCCP.
3Feature negotiation option – опция согласования признаков (свойств).
4В оригинале - “feature location”. Прим. перев.
5В оригинале - “feature remote”. Прим. перев.
6Round-trip time – время кругового обхода.
7Congestion Control ID – идентификатор механизма контроля насыщения. Прим. перев.
8Maximum segment lifetime.
9Быть требовательным по отношению к собственным действиям и великодушным при восприятии действий других.
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 согласование1 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 заголовка2 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**
Передается клиентом для организации соединения (первый шаг трехэтапного согласования).

**DCCP-Response**
Передается 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**
Передается сервером в качестве запроса к клиенту на разрыв соединения.

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

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

**DCCP-Sync, DCCP-SyncAck**
Служат для ресинхронизации порядковых номеров после потери большего 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.

<table>
<thead>
<tr>
<th>Пакет DCCP A</th>
<th>Пакет DCCP B</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCCP-Data(seqno 1) →</td>
<td>DCCP-Ack(seqno 10, ackno 2) →</td>
</tr>
<tr>
<td>DCCP-Data(seqno 2) →</td>
<td>DCCP-Data(seqno 11) →</td>
</tr>
</tbody>
</table>

1Three-way initiation handshake.
2Generic header.
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 еще не получен. Отдельные опции показывают порядковые номера пакетов, которые не были получены.

4.3. Состояния

Конечные точки DCCP проходят различные состояния 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.

Всего возможно девять состояний, которые перечислены ниже в порядке возрастания

<table>
<thead>
<tr>
<th></th>
<th>Клиент</th>
<th>Сервер</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CLOSED</td>
<td>LISTEN</td>
</tr>
<tr>
<td>1</td>
<td>REQUEST</td>
<td>DCCP-Request --&gt; DCCP-Response</td>
</tr>
<tr>
<td></td>
<td>PARTOPEN</td>
<td>DCCP-Ack или DCCP-DataAck --&gt;</td>
</tr>
<tr>
<td>2</td>
<td>OPEN</td>
<td>&lt;-- DCCP-Data, Ack, DataAck --&gt;</td>
</tr>
<tr>
<td></td>
<td>CLOSING</td>
<td>&lt;-- DCCP-CloseReq</td>
</tr>
<tr>
<td></td>
<td>TIMEWAIT</td>
<td>&lt;-- DCCP-Reset</td>
</tr>
<tr>
<td></td>
<td>CLOSED</td>
<td></td>
</tr>
</tbody>
</table>

Все состояния, которые перечислены ниже в порядке возрастания

1 Номеров состояний. Прим. перев.
2 Congestion Control ID - идентификатор контроля насыщения. Прим. перев.
Отличия DCCP от протокола TCP.

Энциклопедия сетевых протоколов
Перевод RFC 4340

ССИД 2 обозначает механизм TCP-like Congestion Control\(^1\), который похож на используемый протоколом TCP контроль насыщения. Отправитель поддерживает окно насыщения и передает пакеты, пока это окно не будет заполнено. Доставка пакетов подтверждается получателем. Индикация насыщения происходит путем отбрасывания пакетов и передачи уведомлений ECN\(^2\) [RFC3168]. Отклоняя 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\(^3\) - 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 маркировки пакетов. CCID 3 отличается от TCP при рассмотрении коротких интервалов времени, но ведет 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.


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

Опции Change L и Confirm L 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.

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:

- Больше 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) передается примерно 1 подтверждение на каждые 2 пакета и в каждом подтверждении должно точно указываться полученные пакеты. В CCID 3 (TFRC) передается примерно 1 подтверждение за период кругового обхода и подтверждение должно покрывать по крайней мере протяженность последнего интервала потерянных пакетов.
- Защита от DoS-атак\(^4\). Некоторые механизмы помогают ограничить количество состояний, которые могут заставить подождать на сервере DCCP клиенты с некорректным поведением. Опция Init Cookie аналогична SYN Cookies [SYNCOOKIES] в TCP и позволяет предотвратить атаки типа SYN-flood\(^5\). Только одна из конечных точек может содержать состояние TIMEWAIT; пакет DCCP-CloseReq, который может быть передан только сервером, передает 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 приемного 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 приемного буфера. Пакет DCCP может быть подтвержден после успешной обработки его заголовка. Более точно, пакет становится подтверждаемым на этапе 8.

---

1. Контроль насыщения в стиле TCP.
2. Explicit Congestion Notification – явное уведомление о насышении. Прим. перев.
3. TCP-Friendly Rate Control – дружественный к TCP контроль скорости.
4. Denial of Service – атаки, направленные на отказ служб.
5. Potok пакетов с флагом SYN, запрашивающих организацию нового соединения.

Pr Incidения сетевых протоколов
Translation of RFC 4340

---

WWW.protocols.ru
Отсутствие окна приема. DCCP является протоколом с контролем насыщения, а не контролем потоков.

- Отсутствие дубликатов при организации соединений. Каждое соединение имеет одного клиента и один сервер.

- Отсутствие полуоткрытых состояний. DCCP не имеет состояний, соответствующих FINWAIT и CLOSEWAIT в TCP, когда одна сторона уже явно закрыла соединение, а вторая еще 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).

1. Клиент направляет серверу пакет DCCP-Request, задающий номер 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 прикрепить к пакету DCCP-Request запрос приложения, но сервер вправе игнорировать этот запрос.

2. Сервер передает клиенту пакет DCCP-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. Сервер передает клиенту пакет DCCP-Reset, подтверждающий получение пакета DCCP-Response. Этот пакет подтверждает начальный порядковый номер сервера и возвращает опции Init Cookie из пакета DCCP-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 прицепленным к ним подтверждением. Если у клиента нет данных для передачи серверу, последний будет передавать пакеты DCCP-Data и DCCP-DataAck, а клиент - только пакеты DCCP-Data до получения от сервера по крайней мере одного пакета, отличного от DCCP-DataAck.

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

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

7. Сервер передает пакет DCCP-Reset с кодом сброса (Reset). Сервер передает пакет DCCP-Close, пользователя услуг пакеты DCCP-Data и сервера, которые были согласованы в пакете DCCP-Response.

8. Клиент получает пакет DCCP-Reset и сохраняет состояние в течение двух сроков максимального времени жизни сегмента (2MSL), что позволяет остальным пакетам пойти вдоль.

При закрытии соединения по инициативе клиента процедура несколько меняется:

5b. Клиент передает пакет DCCP-ResetClose, закрывающий соединение.

6b. Сервер переходит к состоянию LISTEN (TIMEWAIT) и сбрасывает его состояние.

7b. Клиент получает пакет DCCP-Reset и удерживает состояние в течение периода 2MSL, чтобы оставшиеся пакеты могли пойти вдоль.

5. Формат пакетов

Заголовок DCCP может иметь размер от 12 до 1020 байтов. Начальная часть заголовка использует одинаковую схематику для всех определенных 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.

5.1. Базовый заголовок

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

1TIMEWAIT. Прим. перев.
2Extended Sequence Numbers – флаг использования расширенных порядковых номеров.
10

**Энциклопедия сетевых протоколов**

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 ниже.

Один бит Source Port и один бит Destination Port описаны ниже.

Поля базового заголовка описаны ниже.

Поле Checksum Coverage определяет часть пакета, которая используется для расчета контрольной суммы (поле Checksum). Эта часть всегда включает заголовок и опции DCCP, но прикладные данные могут быть исключены из расчета 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 бит

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 байтах, а получатель должен игнорировать это поле.

**Type:** 4 бита

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

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, DCCP-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 надежной защиты от атак с угадыванием порядковых номеров (см. параграф 7.6).

**Sequence Number:** 48 или 24 бита

---

### Таблица 1. Типы пакетов DCCP

<table>
<thead>
<tr>
<th>Тип</th>
<th>Название</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DCCP-Request</td>
</tr>
<tr>
<td>1</td>
<td>DCCP-Response</td>
</tr>
<tr>
<td>2</td>
<td>DCCP-Data</td>
</tr>
<tr>
<td>3</td>
<td>DCCP-Ack</td>
</tr>
<tr>
<td>4</td>
<td>DCCP-DataAck</td>
</tr>
<tr>
<td>5</td>
<td>DCCP-CloseReq</td>
</tr>
<tr>
<td>6</td>
<td>DCCP-Close</td>
</tr>
<tr>
<td>7</td>
<td>DCCP-Reset</td>
</tr>
<tr>
<td>8</td>
<td>DCCP-Sync</td>
</tr>
<tr>
<td>9</td>
<td>DCCP-SyncAck</td>
</tr>
<tr>
<td>10-15</td>
<td>Резерв</td>
</tr>
</tbody>
</table>
Все определенные в настоящем стандарте типы пакетов, за исключением DCCP-Request и DCCP-Response, содержат базовый заголовок DCCP с X=1 (16 байтов) / Type = 0 (DCCP-Request) / Service Code / Данные приложения / Опции и заполнение.

В общем случае содержат значение GSR1 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 устанавливатьAcknowledgement 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 поддерживаться от использования общепринятых номеров портов для служб (см. параграф 8.1.2).

### 5.3. Пакеты DCCP-Response

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

**Acknowledgement Number:** 48 битов

Содержит значение GSR. Поскольку пакеты DCCP-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

---

1. Greatest Sequence Number Received – максимальный порядковый номер принятого пакета.
Пакеты DCCP-Ack не включают данных приложения, но содержат номер подтверждения (Acknowledgement Number). Такие пакеты служат исключительно для передачи подтверждений о доставке.

Пакеты DCCP-Reset служат для некорректного порядкового номера пакеты DCCP-Reset, но эти пакеты нормальное завершение безусловного разрыва соединений. Пакеты DCCP-Reset, которые показывают желание сервера закрыть соединение, но нежелание сохранять на своей стороне состояние TIMEWAIT.

Пакеты DCCP-DATA и DCCP-DATAck могут иметь область данных приложения нулевого размера, что показывает передачу приложением дейтаграммы нулевой длины. Этим пакетам указанных типов отличаются от пакетов DCCP-Request и DCCP-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 нулевого размера, которая будет дополнять пакеты DCCP-Ack до желаемой длины. Получатель должен игнорировать данные, принятые в пакетах DCCP-Ack.

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-CloseReset. Сервер может также передавать пакет DCCP-CloseReq, который показывает желание сервера закрыть соединение, но нежелание сохранять на своей стороне состояние TIMEWAIT.

5.6. Пакеты DCCP-Reset
Пакеты DCCP-Reset служат для безусловного разрыва соединений.

Ответ на подтверждение пакета DCCP-ACK также состоит из 8 бит, но оно не содержит подтверждения.

Reset Code: 8 битов

---

Перевод RFC 4340

www.protocols.ru
Перевод RFC 4340

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».

Определенные 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 серьезна 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 передается 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 передается только в пакетах DCCP-Request (см. параграф 8.1.3).

9, Too Busy - сервер занят

Сервер слишком занят для того, чтобы принимать новые соединения. Этот код передается 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.

1 Наказание за агрессивное поведение - получены 3 некорректных сигнала ECN Nonce Echo, что говорит о недопустимом поведении.
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 storms), поскольку после сброса первой конечной точки соединения второй пакет DCCP-Reset будет игнорироваться.

5.7. Пакеты DCCP-Sync и DCCP-SyncAck

Пакет DCCP-Sync помогает конечным точкам DCCP восстановить синхронизацию порядковых номеров после потери множества пакетов, а также выстраивает порядок следования пакетов из полуоткрытого состояния. Каждый принятый корректный пакет DCCP-Sync вызывает незамедлительную передачу пакету 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 подтверждено. Таким образом, он должен быть обработан, либо 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 искривать. Дополненные пакеты DCCP-Sync могут быть полезны для выполнения процедуры Path MTU discovery (см. главу 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 допускаются для выравнивания по 32-битовой границе и каждая опция может быть дополнена соответствующим числом байтов для выравнивания по 32-битовой границе; для добавочных байтов должна использоваться опция Padding. Все имеющиеся в пакете опции учитываются при расчете контрольной суммы заголовка.

Первый байт каждой опции определяет ее 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 после неизвестной опции 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.

Определенные в главе 4 длины опции перечислены в таблице 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 рассматриваятся при описании отдельных опций. В таблице указаны ограничения общего значения - если в колонке DCCP-Data? указан ‘-‘, соответствующая опция должна игнорироваться при ее получении в пакетах DCCP-Data (а 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.
5.8.1. Опция Padding

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

5.8.2. Опция Mandatory

Но предположим, что следующей опцией является O. Тогда опция Mandatory не будет иметь эффекта, если приемная 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. Приведенный 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 значения признаков. Опции Change инициируют согласование, а опции Confirm завершают его. Опции L передаются держателем признака, а опции R относятся к удаленному 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 задает номер | Type | Length | Feature# | Value(s) ... |

на рисунке. Первый байт данных опции задает номер | Type | Length | Feature# | Value(s) ... |

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.

Таблица 3: Опции DCCP

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

Энциклопедия сетевых протоколов www.protocols.ru
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 опцией 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 предпочитений сервера, которая присутствует также в списке предпочитений 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 принять любое значение, отвечая с помощью опции Confirm, 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.

<table>
<thead>
<tr>
<th>Номер</th>
<th>Значение</th>
<th>Правило</th>
<th>Начальное значение</th>
<th>Обязательный</th>
<th>Описание</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Резерв</td>
<td>SP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Congestion Control ID (CCID)</td>
<td>NN</td>
<td>2</td>
<td>+</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Allow Short Seqnos</td>
<td>SP</td>
<td>0</td>
<td>+</td>
<td>7.6.1</td>
</tr>
<tr>
<td>3</td>
<td>Sequence Window</td>
<td>NN</td>
<td>100</td>
<td>+</td>
<td>7.5.2</td>
</tr>
<tr>
<td>4</td>
<td>ECN Incapable</td>
<td>SP</td>
<td>0</td>
<td>-</td>
<td>12.1</td>
</tr>
<tr>
<td>5</td>
<td>Ack Ratio</td>
<td>NN</td>
<td>2</td>
<td>-</td>
<td>11.3</td>
</tr>
</tbody>
</table>
Перевод RFC 4340

<table>
<thead>
<tr>
<th>Номер</th>
<th>Значение</th>
<th>Правило</th>
<th>Начальное значение</th>
<th>Обязательный</th>
<th>Описание</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Send Ack Vector</td>
<td>SP</td>
<td>0</td>
<td>-</td>
<td>11.5</td>
</tr>
<tr>
<td>7</td>
<td>Send NDP Count</td>
<td>SP</td>
<td>0</td>
<td>-</td>
<td>7.7.2</td>
</tr>
<tr>
<td>8</td>
<td>Minimum Checksum Coverage</td>
<td>SP</td>
<td>0</td>
<td>-</td>
<td>9.2.1</td>
</tr>
<tr>
<td>9</td>
<td>Check Data Checksum</td>
<td>SP</td>
<td>0</td>
<td>-</td>
<td>9.3.1</td>
</tr>
<tr>
<td>10 - 127</td>
<td>Резерв</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128 - 155</td>
<td>Связанные с CCID признаки</td>
<td></td>
<td></td>
<td></td>
<td>10.3</td>
</tr>
</tbody>
</table>

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 текущей опцией Confirm. Естественно, механизмы CCID могут требовать поддержки определенных признаков - например, протокол DCCP, реализующий CCID 2, должен поддерживать признаки Ack Ratio и Send Ack Vector.

6.5. Примеры согласования признаков

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

Следующий пример (рисунок слева) показывает одновременное согласование признаков.

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

**Клиент**

**Сервер**

<table>
<thead>
<tr>
<th>Пример</th>
<th>Опция</th>
<th>Справка</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Change R(CCID, 2 3 1) --*</td>
<td>32,5,1,2,3</td>
</tr>
<tr>
<td>2.</td>
<td>Confirm L(CCID, 3, 3 2 1)</td>
<td>32,9,3,0,0,0,0,4,0</td>
</tr>
</tbody>
</table>

DCCP A меняет значение CCID/A на 2; предпочтительными значениями являются 2 и 3 в указанном порядке.

**Клиент**

**Сервер**

<table>
<thead>
<tr>
<th>Пример</th>
<th>Опция</th>
<th>Справка</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a.</td>
<td>Change R(CCID, 2 3 1) --*</td>
<td>32,9,3,0,0,0,0,4,0</td>
</tr>
<tr>
<td>b.</td>
<td>Confirm R(Ack Ratio, 3)</td>
<td>32,9,3,0,0,0,0,4,0</td>
</tr>
</tbody>
</table>

DCCP A меняет значение Ack Ratio/A на 3; предпочтительными значениями являются 3 в указанном порядке.

6.6. Обмен опциями

Для обмена опциями согласования признаков существует несколько базовых правил.

1. Каждая опция Change, переданная с соблюдением порядка, дает в ответ опцию Confirm.
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 «передает» 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 тем 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 нее список предпочтений, согласует с собственными предпочтениями и возвращает в ответ опцию Confirm L или Confirm R, соответственно, информирующую партнеру 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, содержащей опцию Confirm, должен включать Acknowledgement Number (см. параграф 6.6.4, описывающий детектирование и обработку нарушения порядка доставки опций Change). Созданные опции Confirm могут присоединять к имеющемуся пакету (например, DCCP-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 трех состояний.

- **(STABLE)** - нормальное состояние (включает в себя STABLE и CHANGING), когда признак не изменяется, а изменение признака возможно.
- **(UNSTABLE)** - неустойчивое состояние, когда изменение признака неизвестно.
- **(CHANGING)** - состояние признака, когда признак изменяется.

**Схема изменения состояния признака**

```
-----------
|         |
|         |
|         |
```

**Опции Change**

```
1. rcv Confirm R: ignore +---+ |
   | v | rcv Change R v v
2. | UNSTABLE to CHANGING +---+|
   | Changes | snd Confirm L v
3. | Change L: ignore +---+ |
```

**Опции Confirm**

```
1. rcv Confirm R: calc new value, +---+ |
   | v | rcv Change R v v
2. changes | snd Confirm L v
3. | STABLE to UNSTABLE +---+|
```

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

**Псевдокод**

```java
1. 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 F.segno <= 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 удаленной 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 в ответ на отсутствие гарантий доставки. Этот процесс называется повтором передачи, хотя это и
не является повтором в полном смысле, поскольку повторные опции Change передаются в новых пакетах с иными
порядковыми номерами.
Конечная точка, находящаяся в состоянии CHANGING, передает 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 менее 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 приема 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 возрастаания значений Sequence Number. Далее в этом параграфе рассматриваются два алгоритма,
соответствующих этому требованию.
Первый алгоритм использует две переменные 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 FGSS 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 большие 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 **должна** игнорировать все опции Confirm, которые не подтверждают последнюю переданную опцию Change. Это гарантирует, что согласие будет достигнуто с учетом наиболее свежего списка предпочтений конечной точки.

### 6.6.6. Смена предпочтений

Конечным токам 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 признак к конечной точкой, предполагающей, что признак имеет другое значение.

### 6.6.6. Одновременное согласование

Две конечные точки могут одновременно начать согласование одного признака после чего конечная точка, находящаяся в состоянии CHANGING, получит опцию Change с тем же значением. Получение таких признаков из конечной точки действовать как отказ 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 получить 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 в ответ на получение опции Confirm (т. е., опции 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 отношения к опциям Change или Confirm. 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-битовые, 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 обрабатывается 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 нее 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;

20 www.protocols.ru
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 счете 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 механизм передачи повторов, но повтор передачи таких пакетов, как DCCP-Request, также ведет 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** - начальный порядковый номер, переданный этой конечной точкой.

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

**ISR** - 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 DCCP-Response.

**GSS** - наибольший порядковый номер, переданный этой конечной точкой.

Максимальное значение переданного порядкового номера с учетом циклической арифметики.

**GSR** - наибольший порядковый номер, полученный от удаленной точки.

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

**GAR** - 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** и **SWH** - минимальное и максимальное значение окна порядковых номеров.

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

**AWL** и **AWH** - 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 DCCP-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

---

1Initial Sequence Number Sent.
2Initial Sequence Number Received.
3Greatest Sequence Number Sent.
4Greatest Sequence Number Received.
5Greatest Acknowledgement Number Sent.
6Greatest Acknowledgement Number Received.
7Sequence Number Window Low.
8Sequence Number Window High.
9Acknowledgement Number Window Low.
10Acknowledgement Number Window High.
для недавнего соединения с таким же квартетом. Термин «недавний» в этом контексте относится к пакетам, 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 бит 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 путем добавления криптографической хэш-функции и секретного значения к квартету номеров портов и адресов при выборе каждого начального порядкового номера. Для выбора секретного значения [RFC1948] рекомендуется 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 изменению этого 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 первые младшие 8 битов пространства начальных порядковых номеров, которое занимается обоим клиентам 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 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 счетчика, 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 гарантии 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 передаваемые пакеты имеют корректные значения контрольной суммы заголовка и порядкового номера. Передаваемое конечной точкой значение Acknowledgement Number должно совпадать со значением GSR1 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 поражения или переполнения приемных буферов. Опция 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 обрабатывались). 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 втереться в сетевой 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 первые младшие 8 битов пространства начальных порядковых номеров, которое занимается обоим клиентам 2MSL после смены секретного значения.

6) Fast Initial Connect (FIC)

7.5.1. Окна порядковых номеров и номеров подтверждений

Каждая конечная точка DCCP определяет окна корректности, которые являются подмножеством пространства порядковых номеров и номеров подтверждений. Эти окна соответствуют пакетам, которые конечная точка ожидает

1Greatest Sequence Number Received – максимальный принятый порядковый номер. Прим. перев.
В общем случае пакеты являются корректными с точки зрения нумерации, если порядковые номера и номера подтверждений в сети более вероятная, нежели существенное нарушение порядка их доставки.

Верхняя граница окна (AWH) равна GSS (Greatest Sequence Number Sent – максимальный переданный порядковый номер) DCCP A; ширина окна W представляет собой значение признака Sequence Window/A.

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

\[
\begin{align*}
\text{SWL} & = \max (GSR + 1 - \lfloor W/4 \rfloor, ISR), \\
\text{AWL} & = \max (GSS + 1 - W, ISS).
\end{align*}
\]

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 передает 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 отличаться числом пакетов, которые DCCP предполагает находящимися в пути. Это может прогнозироваться только DCCP A. Слишком малые значения повышают для конечной точки риск потери синхронизации в результате потери группы пакетов, а еще меньше 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 следовать 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 принявшего пакета.

<table>
<thead>
<tr>
<th>Тип пакета</th>
<th>Проверка поряд. номера</th>
<th>Проверка номера подтвержд.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCCP-Request</td>
<td>SWL &lt;= seqno &lt;= SWH (*)</td>
<td>nеприменимо</td>
</tr>
<tr>
<td>DCCP-Response</td>
<td>SWL &lt;= seqno &lt;= SWH (*)</td>
<td>AWL &lt;= ackno &lt;= AWH</td>
</tr>
<tr>
<td>DCCP-Data</td>
<td>SWL &lt;= seqno &lt;= SWH</td>
<td>nеприменимо</td>
</tr>
<tr>
<td>DCCP-Ack</td>
<td>SWL &lt;= seqno &lt;= SWH</td>
<td>AWL &lt;= ackno &lt;= AWH</td>
</tr>
<tr>
<td>DCCP-DataAck</td>
<td>SWL &lt;= seqno &lt;= SWH</td>
<td>AWL &lt;= ackno &lt;= AWH</td>
</tr>
<tr>
<td>DCCP-CloseReq</td>
<td>GSR &lt; seqno &lt;= SWH</td>
<td>GAR &lt;= ackno &lt;= AWH</td>
</tr>
<tr>
<td>DCCP-Close</td>
<td>GSR &lt; seqno &lt;= SWH</td>
<td>GAR &lt;= ackno &lt;= AWH</td>
</tr>
<tr>
<td>DCCP-Reset</td>
<td>GSR &lt; seqno &lt;= SWH</td>
<td>GAR &lt;= ackno &lt;= AWH</td>
</tr>
<tr>
<td>DCCP-Sync</td>
<td>SWL &lt;= seqno &lt;= SWH</td>
<td>AWL &lt;= ackno &lt;= AWH</td>
</tr>
<tr>
<td>DCCP-SyncAck</td>
<td>SWL &lt;= seqno &lt;= SWH</td>
<td>AWL &lt;= ackno &lt;= AWH</td>
</tr>
</tbody>
</table>

(*) - 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 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 конечные точки от необходимости использовать новые значения порядковых номеров.

Наконец, конечная точка может выполнить проверку порядкового номера в пакете DCCP-Sync с некорректным порядковым номером.

Все пакеты DCCP-Sync и DCCP-SyncAck с некорректным порядковым номером должны игнорироваться.

При получении пакета DCCP-Sync с некорректным порядковым номером партнерская конечная точка (скажем, DCCP В) должна обновить свою переменную GSR и передать в ответ пакет DCCP-SyncAck. Номер подтверждения в пакете DCCP-SyncAck будет равен порядковому номеру пакета 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 А будет обновлять свою переменную GSR и синхронизация номеров между конечными точками восстановится. Как исключение из этого правила конечная точка в состоянии REQUEST может отвечать пакетом DCCP-Reset вместо DCCP-SyncAck. Это нужно для сброса полуткрытого соединения DCCP A.

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

Некоторым конечным точкам DCCP недопустимо обрабатывать пакеты с некорректной нумерацией, за исключением, возможно, генерации пакетов DCCP-Sync. Например, не допускается обработка пакетов с некорректными номерами, если они могут позднее стать корректными - это может негативно сказаться на производительности. Конечная точка может сохранять пакеты с некорректной нумерацией на время до 2 периодов кругового обхода. Если в течение этого периода пакет станет корректным, конечная точка может снова обработать его.

Отметим, что пакеты DCCP-Reset с некорректными номерами вызывают генерацию пакетов DCCP-Reset. Это происходит потому, что конечная точка в неработающем соединении (CLOSED, REQUEST, LISTEN) может не иметь информации, достаточной для генерации подходящего пакета DCCP-Reset.

Конечным точкам DCCP следует использовать приведенные здесь таблицы для определения порядковых номеров и номеров подтверждений, которые они должны использовать в ответ на некорректные пакеты.

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 серьезных 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 попадет 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 попадет в окно корректных номеров подтверждений, получатель будет соответствующим образом сдвигать свое окно корректных порядковых номеров, теряя синхронизацию с корректной конечной точкой.

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 передает атакующий, как Н, 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.

Вероятность успеха атаки DCCP-Data при использовании коротких порядковых номеров составляет P = W/N2. При W = 100 атакующему потребуется передать более 83 000 пакетов, чтобы вероятность успеха достигла 50%. Для сравнения простейшая атака на TCP путем 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 путем 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 счет 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 1011 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, 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 прочих вредных воздействий в результате такого рода 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 восстанавливают DCCP-Data (seq 2) XXX DCCP B (GSS=10,GSR=1)

синхронизацию

после порядковых номеров

пакетов, приведшей к большому числу

tому, что порядковые номера DCCP A вышли за пределы окна порядковых номеров DCCP B.

Во втором примере соединение DCCP восстанавливается после простой атаки вслепую.

DCCP A (GSS=1,GSR=10)

**Атакующий** --> DCCP-Data(seq 10^6) --> ???

??<-- DCCP-Sync(seq 11, ack 10^6) <-- некорректный порядк. номер; передать Sync

(Назначенный) пример показывает восстановление из полуоткрытого соединения.

DCCP A (GSS=1,GSR=10) DCCP B (GSS=10,GSR=1)

(Crash) CLOSED

REQUEST --> DCCP-Request(seq 400) --> ???

!! <-- DCCP-Sync(seq 11, ack 400) <-- OPEN

REQUEST --> DCCP-Reset(seq 401, ack 11) --> (Abort)

REQUEST <-- DCCP-Request(seq 402) --> ...

Перевод RFC 4340 Энциклопедия сетевых протоколов www.protocols.ru
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 сокращенные номера, задаваемые только младшими 24 битами соответствующего порядкового номера или номера подтверждения. Принимая конечная точка будет восстанавливать 48-битную нумерацию, используя приведенный здесь псевдокод.

procedure Extend_Sequence_Number(S, REF)
    /* S – 24-битовый порядковый номер из заголовка пакета.
       REF – соответствующее 48-битное 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 */
    Иначе, если S (<) REF_low И REF_low |<| S,
        Возвратить (((REF_hi - 1) mod 2^24) << 24) | S
    Иначе,
        Возвратить (((REF_hi + 1) mod 2^24) << 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 декретируются подобным 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 принятого по умолчанию значение 0. Это может снизить риск вставки неуместных данных в соединение. DCCP A передает 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 неуместных данных в соединение. DCCP A передает опцию Change L(Allow Short Seqnos, 1) для индикации своего желания передавать пакеты с короткими порядковыми номерами.

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

\[ R = P \times (2^L) / 2MSL \]

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, 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, максимальная безопасная скорость будет составлять приблизительно 800 Мбит/с.

7.7. Опция NDP Count и детектирование потери данных приложения

По умолчанию DCCP увеличивается 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 дает приемной стороне DCCP возможность надежного 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, DCCP-Response, DCCP-Data и DCCP-Ack, non-data packet - пакеты, не содержащие данных приложения. Прим. перев.
DataAck рассматриваются как пакеты данных, хотя пакеты DCCP-Request и DCCP-Response могут и не содержать данных приложения.

Значение, хранимое 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 некоего блока пакетов и признак Send NDP Count включен. В этом случае можно говорить, что блок потерянных пакетов включает 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).

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 будут полезной для приложений, поддерживающих собственный механизм нумерации в заголовках своих пакетов.

### 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 передает начальный запрос DCCP-Request, сервер отвечает на него пакетом DCCP-Response и клиент в заключение передает пакет DCCP-Request, используя выбранный порядковый номер для данного сервера.

<table>
<thead>
<tr>
<th>Состояние клиента</th>
<th>Состояние сервера</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSED</td>
<td>LISTEN</td>
</tr>
<tr>
<td>1. REQUEST</td>
<td>Request</td>
</tr>
<tr>
<td>2. &lt;&lt;-- Response</td>
<td>--&lt;-- RESPOND</td>
</tr>
<tr>
<td>3. PARTOPEN</td>
<td>Ack, DataAck</td>
</tr>
<tr>
<td>4. &lt;&lt;-- Data, Ack, DataAck</td>
<td>&lt;&lt;-- OPEN</td>
</tr>
<tr>
<td>5. OPEN</td>
<td>Data, Ack, DataAck &lt;&lt;-- OPEN</td>
</tr>
</tbody>
</table>

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 повторения повторов, принятую для пакетов TCP SYN. В каждом новом пакете 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 пакета 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 все-таки были доставлены серверу. Клиенту в состоянии REQUEST всегда получат начальный порядковый номер от своего партнера, поэтому в пакете DCCP-Reset должен быть установлено Acknowledgement Number = 0.

Клиент переходит из состояния REQUEST в состояние PARTOPEN при получении от сервера пакета DCCP-Response.
8.1.2. Коды сервиса

Каждый пакет DCCP-Request содержит 32-битовый код сервиса, определяющий службу прикладного уровня, к которой 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).


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 запросов1 с соблюдением приведенных 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.
- Значения Service Code со старшим битом 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-битовых идентификаторов ресурсов Macintosh, блоков PNG, а также таблиц TrueType и OpenType.

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


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

**SC=x** или **SC=X** задает 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 17178548426 можно представить в форме SC:fdpz, SC=17178548426 или SC=x6664707A.

8.1.3. Отклик сервера

Во второй фазе трехэтапного согласования сервер переходит из состояния LISTEN в состояние RESPOND и передает клиенту сообщение DCCP-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 предпочитает использовать.

---
1Процедура «First Come First Served».
2Требуется спецификация.
Сервер может ответить на пакет DCCP-Request пакетом DCCP-Reset, означающим отказ в соединении. Подходящее 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 слишком занят и не может ответить на запрос). Серверу следует ограничивать скорость, с которой могут генерироваться пакеты отказа (например, не более 1024 откликов в секунду).

Серверу не следует повторять передачу пакетов DCCP-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, должен вызывать генерацию нового ответного пакета DCCP-Response. В каждом новом пакете DCCP-Response поле Sequence Number должно увеличиваться на 1, а пакет должен включать те же данные приложения, которые были включены в исходный пакет DCCP-Response.

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

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

Сервер переходит из состояния RESERVED в состояние OPEN при получении от клиента корректного пакета DCCP-Ack, завершающего трехэтапное согласование. Сервер может также перейти из состояния RESERVED 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. Сервер помещает значение Service Code, номер своего порта и все опции, полученные в пакетах DCCP-Request и DCCP-Response в специальную переменную cookie. Обычно значение cookie шифруется с использованием ключа, известного только серверу, и включает критографическую 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 packets или после организации соединения (состояние соединения => OPEN), должны игнорироваться. Сервер может включать опцию Init Cookie в свой пакет DCCP-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 подтвержден (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 проблема интероперабельности возникнуть не может. В качестве примера может служить вариант, 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 проблема интероперабельности возникнуть не может. В качестве примера может служить вариант, который допускает проверку на предмет подделки. Серверу следует отвечать на поддельные опции Init Cookie сбросом соединения с Reset Code 10, Bad Init Cookie.

8.1.5. Завершение соединения

Получив от сервера пакет DCCP-Response, клиент переходит из состояния REQUEST в состояние PARTOPEN и завершает процедуру трехэтапного согласования, передавая серверу пакет DCCP-Ack. Клиент остается в состоянии PARTOPEN до тех пор, пока он не обретет 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-Data, а не DCCP-Data. Это связано с тем, что пакеты DCCP-Data не включают номера подтверждений, поэтому сервер не может понять из пакета DCCP-Data, получил ли клиент его отклики DCCP-Response. Более того, если пакет DCCP-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 отсчет для таймера закончен, а клиент сохраняет состояние PARTOPEN, ему следует генерировать...
новый пакет DCCP-Ack и затем запустить таймер. Если состояние PARTOPEN сохраняется более 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 DCCP-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 передает 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.

Пакеты DCCP-Close и DCCP-Sync 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:

<table>
<thead>
<tr>
<th>Состояние клиента</th>
<th>Состояние сервера</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>1. CLOSING</td>
<td>--&lt;-- CloseReq</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSEREQ</td>
</tr>
<tr>
<td>2. TIMEWAIT</td>
<td>--&lt;-- Reset</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSED (LISTEN)</td>
</tr>
<tr>
<td>3. CLOSED</td>
<td>--&lt;-- CloseReq</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSEREQ</td>
</tr>
<tr>
<td>4. TIMEWAIT</td>
<td>--&lt;-- Reset</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSED (LISTEN)</td>
</tr>
<tr>
<td>5. CLOSED</td>
<td>--&lt;-- CloseReq</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSEREQ</td>
</tr>
</tbody>
</table>

Более короткая последовательность используется для разрыва соединения по инициативе клиента.

<table>
<thead>
<tr>
<th>Состояние клиента</th>
<th>Состояние сервера</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>1. CLOSING</td>
<td>--&lt;-- Close</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSED (LISTEN)</td>
</tr>
<tr>
<td>2. TIMEWAIT</td>
<td>--&lt;-- Reset</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- TIMEWAIT</td>
</tr>
<tr>
<td>3. CLOSED</td>
<td>--&lt;-- Close</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSED (LISTEN)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Состояние клиента</th>
<th>Состояние сервера</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>1. CLOSED</td>
<td>--&lt;-- Close</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSING</td>
</tr>
<tr>
<td>2. TIMEWAIT</td>
<td>--&lt;-- Reset</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- TIMEWAIT</td>
</tr>
<tr>
<td>3. CLOSED</td>
<td>--&lt;-- Close</td>
</tr>
<tr>
<td></td>
<td>--&lt;-- CLOSED (LISTEN)</td>
</tr>
</tbody>
</table>

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.
Сокеты являются:

Принятый пакет обозначается P, сокет явно разрешено иное поведение. Случаях, когда в данном документе псевдокоду за исключением тех соответствующие приведены-ному здесь видимые эффекты, в точности реализации описанному здесь алгоритму, но любая не обязана в точности следовать получении пакета. Реализация DCCP должна пройти конечная точка DCCP при обработке, через которые должна передачи пакетов 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.

**8.3.1. 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 партнера 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.

8.4. Диаграмма состояний DCCP

Большинство смен состояний, описанных выше, можно показать на диаграмме состояний. Рисунок является лишь иллюстрацией, а в качестве определения следует использовать текст, приведенный 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 работы, 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 приведен-ному 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;
**Энциклопедия сетевых протоколов**

**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 отбрасывается 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> */

Идем flow ID в таблице и определям 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
Выбираем S.ISS (начальный порядковый номер) или установив 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 прошел 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 передается 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.
Перевод RFC 4340

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.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 PAROPEN. В PAROPEN нужно передать Ack, не передавать пакетов данных, периодически повторять передачу Ack, всегда включая все опции Init Cookie из пакета Response */
   S.state := PAROPEN
   Установить таймер PAROPEN
   Продолжить с S.state == PAROPEN
   /* 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 PAROPEN

Если S.state == PAROPEN,
   Если P.type == Response,
      Передать Ack
   Иначе, если P.type != Sync, S.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)
   Рассогласовать 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. */
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 расчет контрольной суммы заголовка с учетом лишь части данных, а в некоторых случаях и без учета 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 учета 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 поврежденными данными приложения, которые были учтены в контрольной сумме, трактуются как потерянный. Это может приводить к жесткой 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 поврежденных 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 отклик на повреждение данных, которое может в некоторых случаях напоминать нарушение. В настоящее время отклик на поврежденные 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 поврежденных пакетов даже в тех случаях, когда конечная точка получает из канала поврежденные 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 потеря 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 создает некоторые проблемы безопасности, рассмотренные в параграфе 18.1. Приложение В содержит дополнительную информацию по вопросу выбора области покрытия контрольной суммы. Реализация DCCP с неполным покрытием для контрольной суммы берет свое начало от UDP-Lite [RFC3828].

9.1. Поле Checksum в заголовке

DCCP использует алгоритм расчета 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 всех 16-битовых слов заголовка DCCP, опции DCCP, псевдозаголовка, взятого из заголовка сетевого уровня и, в зависимости от значения поля Checksum Coverage, части или всех данных приложения. При расчете контрольной суммы значение поля Checksum принимается нулевым. Если пакет содержит нечетное количество байтов в полях, используемых для расчета контрольной суммы, справа к последнему байту добавляется 8 нулевых битов для формирования полного 16-битового слова при вычислении контрольной суммы. Этот байт заполнения не передается 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 плес 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) задает, 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 учитывается 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.

Отличие от 0 значения указывает, что допустимо повреждение в части или всех данных приложения, содержащихся в пакете DCCP. Фактически DCCP не сможет даже обнаружить повреждение данных в областях, не покрытых 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 предложить 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 поврежденных 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 партнер принимать пакеты со сниженным значением Checksum Coverage. Например, DCCP A передает 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 значение Checksum Coverage, которое 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 опции Data Dropped (напарпд 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.

Предварающая точка DCCP рассчитывает значение CRC для байтов, содержащихся в области данных приложения, и помещает полученное значение в поле данных опции. Для расчета 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 расчета 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 расчета 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 принять заведомо поврежденные 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 поврежденного 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 сообщит 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 приложения, ее можно включать и в другие пакеты (такие, как DCCP-Ack, DCCP-Sync и DCCP-SyncAck). Получатель должен 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 этом следует 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 ее партнер проверять опции Data Checksum. Точка DCCP A передает обязательную (Mandatory) опцию Change R(Check Data Checksum, 1) точке DCCP B, чтобы проверить от той проверки опций Data Checksum (в противном случае соединение будет сброшено).

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 защищенных контрольной суммой. Понимая это, канальный уровень может снизить строгость детектирования и/или корректировки ошибок при передаче незащищенных частей. В результате это приведет 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 путем согласования значений признаков CCID. Признак CCID имеет номер 1. Значение CCID/A равно идентификатору CCID, используемому для полусоединения от A к B. DCCP передает 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 значений 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, 3, 4) говорит ее получателю, что отправитель использует для своих пакетов CCID 2, но приемлемы также значения CCID 3 и CCID 4.

Определенные в настоящее время значения CCID перечислены в таблице 5.

<table>
<thead>
<tr>
<th>CCID</th>
<th>Название</th>
<th>Документ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>Резерв</td>
<td>[RFC4341]</td>
</tr>
<tr>
<td>2</td>
<td>TCP-like Congestion Control</td>
<td>[RFC4342]</td>
</tr>
<tr>
<td>3</td>
<td>TCP-Friendly Rate Control</td>
<td></td>
</tr>
<tr>
<td>4-255</td>
<td>Резерв</td>
<td></td>
</tr>
</tbody>
</table>

Таблица 5: Идентификаторы механизмов контроля насыщения DCCP

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 интероперабельность, 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 (AIMD1), 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 ведет 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 TFRC2, основанный на выравнивании. Механизм TFRC разработан для обеспечения разумно беспропорционального распределения полосы между одновременными потоками в стиле TCP. Разумная беспропорциональность 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 вначале 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.

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

<table>
<thead>
<tr>
<th>Пакет</th>
<th>Опция</th>
<th>Полусоед.</th>
<th>CCID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &gt; B</td>
<td>128</td>
<td>от A к B</td>
<td>4</td>
</tr>
<tr>
<td>A &gt; B</td>
<td>192</td>
<td>от B к A</td>
<td>5</td>
</tr>
</tbody>
</table>

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), CCID может обрабатывать CCID 3 (если сервер поддерживает CCID 3) или принятим 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), CCID 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 опций. Передавая опции Mandatory Change(CCID) в своем 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, прямо, периоды вносимых приложениям ограничений, а также опции DCCP Data Dropped и Slow Receiver. Механизм 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 применяет во внимание предыдущий 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
после того, как точка DCCP B становится статичной. Обычно это происходит по истечении определенного периода, в течение которого точка DCCP B перестала передавать данные, т. е., соединение становится односторонним.

В качестве примера рассмотрим двухстороннее соединение, в котором обе половины используют 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 B к A управляет подтверждениями DCCP B (включая подтверждения подтверждений DCCP A), а CCID полусоединения от B к A управляет подтверждениями DCCP A.

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.

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.

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

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 подтверждений, полученные в пакетах одного типа, а именно, DCCP-Request и DCCP-Data, должны быть определены точно. Однако не все опции подтверждений являются обязательными для каждого пакета, содержащего поле Acknowledgement Number.

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), передает 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 повторно передать данному 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 получения. 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 шлет DCCP A только пакеты DCCP-Ack. В случае CCID 2 (TCP-like Congestion Control) DCCP B использует Ack Vector для надежной 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 передать пакет DCCP-DataAck взамен DCCP-Data. Однако в случае CCID 3 время подтверждений в общем случае «привязаны», поэтому точке A не требуется подтверждать подтверждение от точки B.

Опции подтверждений включают в себя метки ECN. Получатели DCCP должны обрабатывать метки ECN, но могут выбрасывать пакеты без меток ECN, если это не требуется. Например, получатель DCCP может выбросить пакет DCCP-Data, если поле ECN содержит значение 1. Получатель DCCP должен проверять, что поле ECN в пакете DCCP-Data не равно нулю, и если это так, должен вернуть пакет DCCP-Ack с соответствующим значением поля ECN.
11.2. Добавление подтверждений

Данные подтверждения от A к B могут «прицепляться» к данным, передаваемым точкой DCCP B, если это не задерживает доставку подтверждения, чем дозволено CCID от A к B. Однако для данных подтверждений 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 следает передавать раздельные пакеты DCCP-Data и DCCP-Ack или подождать (но не слишком долго) отправки более мелкой дейтаграммы.

Использование «прицепа1» 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 подтвержденных 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 округленному 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 перегружен. Каждый признак 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 определенное значение, независимо от того, используется ли это значение соответствующим CCID.

Признак Ack Ratio имеет номер 5 и устанавливается без согласования. Признак принимает двухбайтные 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 передает 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 обоих точек; это значение обеспечивает поведение подтверждений, аналогичное отложенным подтверждениям в TCP.

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

- Пользователи 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.

- Пользователи могут поднять частоту передачи подтверждений вместо передачи подтверждения сразу же по получении пакета данных. Получателям, повышающим частоту передачи подтверждений, следует поднимать ее 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 передается 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 задает 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.

- 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.

1В оригинале «piggybacking». Прим. перев.
11.4. Опции Ack Vector

Опции Ack Vector обеспечивают групповое копирование истории получения пакетов данным клиентом. Каждый байт вектора дает 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 справа.

<table>
<thead>
<tr>
<th>Состояние</th>
<th>Название</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Received</td>
</tr>
<tr>
<td>1</td>
<td>Received ECN Marked</td>
</tr>
<tr>
<td>2</td>
<td>резерв</td>
</tr>
<tr>
<td>3</td>
<td>Not Yet Received</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Состояние (Sta)</th>
<th>Run Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7</td>
<td></td>
</tr>
</tbody>
</table>

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).

Она опция Ack Vector может подтверждать до 16192 пакетов данных. Если нужно подтвердить большие 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 подтверждений; состояние 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 приемном 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.

Один или несколько опций Ack Vector, которые (совместно) сообщают о состоянии пакета с порядковым номером меньше ISN (начальный порядковый номер), следует рассматривать, как некорректные. Приемная сторона принимает такие пакеты в соответствии с принципами DCCP.

Опция Reset Code 5, Option Error. Ни одна из опций Ack Vector не может относиться к пакетам, которые еще не были переданы, что можно проверить с помощью Acknowledgement Number, как описано в параграфе 7.5.3, 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 еще не принятые (not yet received, State 3).

Приложении А 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 передает...
- 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 пакетов был нарушен в сети. В этом случае пакеты будут представляться как смещенные 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 справляться с такими ситуациями реализации DCCP на стороне HC-Sender следует объединять множество полученных состояний Ack Vector, как показано на рисунке справа.

Для чтения таблицы выбирайтесь 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 (принятый 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 выбирает для данной ячейки в качестве нового состояния 1.

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

Полученное состояние

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>+---+---+---+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+---+---+---+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+---+---+---+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Состояние

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>+---+---+---+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+---+---+---+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+---+---+---+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Например, полученный пакет имеет состояние (0/1/1 0), а получатель (HC-Receiver) получает пакет с новым состоянием (0/1/1 0). В этом случае HC-Receiver может выбрать отказ от старой информации, собранной из полученных от HC-Receivers опций Ack Vector, и восстановить состояние, как показано на рисунке слева.

Для чтения таблицы выбирается строка, соответствующая старому состоянию пакета, и колонка, соответствующая номеру подтверждений. Признак Send Ack Vector имеет пленку в виде (0/1/0 1), 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 выбирает для данной ячейки в качестве нового состояния 1.

11.4.2. Покрытые 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 подтвержденные HC-Receiver, для которых HC-Receiver знает, что отправитель HC-Sender определенно получил подтверждения;
2. пакеты, уже подтвержденные HC-Receiver, для которых HC-Receiver не имеет уверенности в том, что отправитель HC-Sender уже получил подтверждения;
3. пакеты, которые получатель HC-Receiver еще не подтвердил;
4. пакеты, еще не полученные HC-Receiver.

Объединение групп 2 и 3 называется окном подтверждений (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 А). По мере доставки пакетов это окно будет расширяться справа и сужаться слева. Окно расширяется по причине доставки новых подтверждений, 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 передает 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 передает опцию Slow Receiver своему отправителю, чтобы сказать о возникновении проблем при приеме 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-Receiver может повышать скорость. Поэтому получатель 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
показаны в таблице будто они были включены в Normal Block. Определенные значения кодов отбрасывания для отброшенных блоков должны данных из пакета не были доставлены приложению). Пакеты, указанные как “еще не полученные”, в поле Run Len, не были доставлены приложению.

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

Ниже приведено более детальное описание причин.

0 — данные из пакета были отброшены в результате протокольных ограничений. Например, данные были включены в пакет DCCP-Request, но принимаемое 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 приемном буфере, возможно в результате его переполнения (см. параграф 11.7.2).

3 — данные из пакета были отброшены по причине их пережатия (см. параграф 9.3).

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

Опция Slow Receiver не указывает причину, по которой получатель не может справляться с поток данных от отправителя. С такими причинами может относиться нехватка буферной емкости, перегрузка процессора или ограничения прикладной программы. Передающее приложение может реагировать на отбрасывание пакеты Slow Receiver снижением скорости передачи на прикладном уровне или иным способом.

Однако передающему приложению не следует реагировать на получение опции Slow Receiver передачей большого объема данных. Хотя оптимальной реакцией на ограниченность ресурсов процессора приемной стороны будет снижение уровня сжатия и передача большего объема данных (данные с высоким уровнем сжатия могут загружен процессор на приемной стороне значительно сильнее, чем большей 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 функцию окна приема 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 поврежденными или получатель неспособен принимать информацию с той скоростью, которую установил отправитель и данные были отброшены приемным 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 помещены 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.

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 недопустимо обработка опций из тех пакетов, которые не считаются принятыми.

Вид данных опции показан на рисунке справа.

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

Опция Slow Receiver частично реализует функцию окна приема TCP.

Таблица 7: Коды отбрасывания пакетов DCCP

<table>
<thead>
<tr>
<th>Код причины</th>
<th>Название</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Protocol Constraints</td>
</tr>
<tr>
<td>1</td>
<td>Application Not Listening</td>
</tr>
<tr>
<td>2</td>
<td>Receive Buffer</td>
</tr>
<tr>
<td>3</td>
<td>Corrupt</td>
</tr>
<tr>
<td>4 - 6</td>
<td>Резерв</td>
</tr>
<tr>
<td>7</td>
<td>Delivered Corrupt</td>
</tr>
</tbody>
</table>

1Drop Code - код отбрасывания.
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 приемном буфере (Drop Block, Drop Code 2, Run Length 0).

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

Пакеты 94, 93 были отброшены в приемном 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 подтвердить до 32384 нормальных блоков, хотя получатель не следует передавать опцию Data Dropped, когда все относящиеся к делу пакеты помещаются в нормальные блоки. Если требуется подтвердить больше пакетов, нужно 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 большого числа пакетов, нежели было передано, менять состояние для пакетов, вступать в противоречие с Ack Vector или эквивалентными опциями (например, сообщая о пакетах, которые еще не доставлены, как об отброшенных в приемном 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 сбрасывать «значимость» Drop Code; например, приложение не следует позволять менять для пакета состояние с «поврежден при доставке» (Drop Code 7) на «доставлен нормально» (нет Drop Code).

Информация Data Dropped передается 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 переезду

При решении вопроса об отклое 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.

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 задается 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 Constraints1 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 подходят). Однако передающая конечной точке не следует передавать данные, пока она не убедится, что протокольные ограничения более не применимы.

Drop Code 1, Application Not Listening2 означает, что приложение на той стороне, которая передала опции, более не принимает данные. Например, сервер мог закрыть свое приемное 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. Опции Data Dropped с Drop Code 1 следует передавать всякий раз, когда полученные данные игнорируются в результате того, что приложение прекратило прослушивание входящих информаций. 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 приемном 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 выходят за допустимые пределы приемного буфера в пользовательском пространстве (однако в общем случае не будет пользы от передачи уведомлений об отбрасывании пакетов с Drop

1Протокольные ограничения.
2Приложение не прослушивает входящие данные.
12. Явное уведомление о насыщении
Протокол DCCP полностью поддерживает ECN [RFC3168]. Каждый CCID задает для конечных точек способ отклика на маркерову 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 None 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 None 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 MSЭ, обманывающее реализации ECN в TCP [RFC3360], использует биты заголовка TCP, а не биты ECN в заголовке IP; нам не известны 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 поддерживаются) для обеих сторон. Значения признака 2 и более зарезервированы.

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 подтверждение опцию Data Dropped с Drop Code 0 (протокольные ограничения), если другая точка шлет данные до передачи требуемой опции.

12.2. Сигналы ECN None
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, показывающих отброшенные или маркированные пакеты, как доставленные нормально. Эта проблема более серьезна для DCCP, нежели для TCP, поскольку протокол TCP обеспечивает гарантированную доставку и скрывать от TCP потерю пакетов удаляет ECN Nonce, получатель, желающий скрыть маркировку или отбрасывание пакетов, должен выбрасывать пакеты и запоминать их в памяти. В DCCP, в отличие от TCP, маркер ECN может быть забыт отправителем и маркер ECN Nonce может быть забыт получателем. Это обстоятельство делает нецелесообразным широкое использование ECN Nonce в качестве уведомления о насыщении.

Сигналы ECN Nonce представляют собой простой способ уведомления о насыщении, 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 устанавливает значение полей ECN Nonce в заголовке кадра, как показано на рисунке. В DCCP, в отличие от TCP, маркер ECN может быть забыт отправителем и маркер ECN Nonce может быть забыт получателем. Это обстоятельство делает нецелесообразным широкое использование ECN Nonce в качестве уведомления о насыщении, предотвращающего сокрытие ECN (или потери пакетов).
13. Опции синхронизации

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

13.1. Опция Timestamp

<table>
<thead>
<tr>
<th>Type=41</th>
<th>Length=6</th>
</tr>
</thead>
<tbody>
<tr>
<td>00101001</td>
<td>00000110</td>
</tr>
</tbody>
</table>

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. Конечные точки DCCP, принявшие опцию Timestamp, могут сбросить соединение с использованием Reset Code 11, Aggression Penalty.

13.2. Опция Elapsed Time

<table>
<thead>
<tr>
<th>Type=43</th>
<th>Len=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>00101011</td>
<td>00000110</td>
</tr>
</tbody>
</table>

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 содержит 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 приемом 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 приема подтверждаемого пакета в сотых долях миллисекунд (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 4294967294 сек. Для конечных точек DCCP недопустимо передавать значения Elapsed Time, существенно превышающие реальное время задержки между приемом 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.
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 нее значения Elapsed Time.

Первые 4 байта данных опции (поле Timestamp Echo) содержат поле Timestamp Value из полученной последней опции Timestamp. Обычно эта опция берется 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 ведет 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. Можете 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 передается 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 приема 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\(^1\)), разрешенного 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 приложение значение MPS следует выбирать с учетом 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, реализация 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 будет 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 указано 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 сокращает свое 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 указано в [RFC1191] (мы полагаем, что это может вызывать проблемы для конечных точек DCCP, расположенных за некоторыми типами МСЭ).

\(^1\)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.
Реализация DCCP может разрешать приложению время от времени запрашивать повторное определение PMTU. Это будет приводить к установке старого значения MTU выходного интерфейса. Часто 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 снизить 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 маршрутизатором следует возвращать полный заголовок DCCP и данные из пакета, общим размером до 576 битов [RFC1812] или минимальное значение IPv6 MTU [RFC2463], хотя не обязательно возвращать более 64 битов [RFC722]; 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 снизить в ответ на их значение PMTU лишь на короткое время. Если получено 3 или более разрешенных сообщений Datagram Too Big и другая конечная точка DCCP сообщает о том, что потеряно более 3 пакетов, реализация DCCP следует предполагать наличие плохо настроенного маршрутизатора и принимать указанное в сообщениях ICMP значение PMTU или (для IPv4) разрешать фрагментацию пакетов.

Для соединений IPv4, где приложения запросили фрагментацию, отправителю следует передавать пакеты нефрагментируемыми, как описано выше, за исключением следующих случаев.

- 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, превышающих по размеру значение PMTU, в блохах подходящего размера (блоки, естественно, являются нефрагментируемыми).
- Нежелательно определять PMTU на этапе начального согласования параметров соединения, поскольку процесс организации соединения может не представить разметку пакетов, которые будут использоваться в соединении, и выполнение процедуры 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 значения 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 отвернуть передачу пакета и возвратить соответствующее сообщение об ошибках. 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. Приведенные 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 обрабатывать 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 предписывает 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 если бы задуманные - но не считаются стандартными (не расширенные) список предположений 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.
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, участвующим в составлении сообщений. 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 некелятным 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 сообщение.

Поле 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 GoID опции типа Loss Interval для TFRC. На поддерживаемых ECN соединениях промежуточным устройствам нужно отслеживать информацию ECN Nonce для добавляемых или удалаемых 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 путем добавления или удаления пакетов. Отметим, что необходимость менять порядковые номера DCCP, основанные на учете пакетов, возникает гораздо реже необходимости изменения порядковых номеров TCP, учитывающих байты; например, промежуточный узел может 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 реализовано 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 путем 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 учетом состояния соединений.

17. Связь с другими спецификациями

17.1. RTP

Протокол RTP2 [RFC3550] в настоящее время используется на базе транспорта UDP многими приложениями, поддерживающими 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 агбитржей 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 задержки.

Рассмотрим сначала подтверждения. Оба протокола RTP и DCCP передают отправителю уведомления о скорости потери пакетов с помощью пакетов RTCP SR/RR2 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 jitter4), а подтверждения DCCP содержат информацию, отсутствующую в RTCP (например, ECN Nonce Echo). Ни один из механизмов обратной связи не заменяет полностью другой механизм.

1 Расширение недоступно.
2 Real-Time Transport Protocol — транспортный протокол для приложений, работающих в реальном масштабе времени. Прим. перев.
4 Флуктуации интервала доставки пакетов.
Передача обоих типов информации не повышает значительно расходы по сравнению с передачей одного (любого) типа. Отчеты RTP могут передаваться сравнительно редко (в среднем 1 раз в течение 5 сек. для узкополосных потоков). В DCCP некоторые механизмы обратной связи достаточно «длинны» (опции Ack Vector, например, передаются часто и содержат много данных), а другие сравнительно дешевы - подтверждения CCID 3 (TFRC) занимают 16 или 32 байта опций однократно в течение периода кругового обхода (более редкая передача отчетов 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. Вложеный 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 учета 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 субпотоков DCCP и обеспечивать группы с одинаковыми адресами отправителей и получателей. 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 субпотоков DCCP и обеспечивать группы с одинаковыми адресами отправителей и получателей.

### 18. Вопросы безопасности

DCCP не обеспечивает криптографической защиты. Приложениям, которым требуются криптографические методы защиты (целостность, аутентификация, конфиденциальность, контроль доступа и защита от повторного использования), 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 вынуть конечную точку принять данные атакующего взамен подланной информации) без точного подбора порядковых номеров. Таким образом, при качественном выборе ключевой точки начального порядкового номера, атакующий DCCP должен сунуть свой нос в пакет данных для получения какой-либо вероятности успеха. Проверка корректности порядковых номеров обеспечивает надежные 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 службы. 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 свое 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 учитывающее 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 изменен 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 неутечная 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 путем 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 сломаным поврежденным, 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 аутентичности [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 тип пакета (целое число от 0 до 15), имя типа (например, DCCP-Request) и ссылку на RFC, в котором определен данный тип. Изначально реестр заполняется значениями из таблицы 1 (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), задаваемым 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 2 (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 определен тип опции. Реестр изначально заполняется значениями из таблицы 3 (параграф 5.8). Данный документ выделяет значения для типов опций 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 признак (целое число от 0 до 255), имя признака (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 5 (глава 10). Значения CCID 2 и 3 выделены для опубликованных к настоящему моменту профилей, а значения 248 – 254 резервируются для экспериментов и тестирования. Значение 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 6 (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 (целое 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 7 (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 1145665131 (DISC). Это значение соответствует сервису discard, которые отбрасывает все полученные данные и ничего не передает 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 приведенные ниже правила, которые мы разработали с учетом существующих процедур регистрации номеров портов.

Номера портов делятся на три группы. Общедоступные 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. Общедоступные и зарегистрированные номера портов предназначены для использования серверными приложениями, которым нужен принятый по умолчанию номер порта. В большинстве систем общедоступные 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 тестирующих 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 подчеркиваем, 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 разделенных 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 приведенным 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.

Энциклопедия сетевых протоколов
www.protocols.ru
Энциклопедия сетевых протоколов

- 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 передает 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 заменен 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 просматрившим 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.


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 отклики 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 подтвержденных 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 еще 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 подтвержденного пакета в буфере; этой переменной соответствует указатель head;
- buf_nonce представляет собой однобитовую сумму (четность или «Исключающее ИЛИ») значений ECN Nonce, полученных во всех пакетах, подтверждаемых буфером со статусом State 0.

Вид буфера подтверждений показан на рисунке справа.

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.

```
+-----------------------------------------------+
+-----------------------------------------------+
```

Здесь переменная buf_nonce имеет значение E, a buf_ackno = A. Будем использовать этот буфер в качестве рабочего примера.
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 еще не получены (три следующих байта показывают состояние 3 и run length 0);
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 должен 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 задает 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 акк 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 (переменная buf_head 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 иметь следующий вид (

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

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 прибьют примеру Example Buffer будет иметь следующий вид (изменим отмеченные звездочками).

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

0, Example Buffer может переместиться в состояние, показанное на рисунке справа. Порядковый номер вновь прибывшего пакета может, ** +*+------------------------+         *
естественно, не совпадать с ожидающим номером. В 12 |0,0|3,0|3,0|3,0|3,0|0,4|1,0|0,0| 0
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 прибьют с ECN Nonce 1, Example Buffer перейдет в состояние, показанное на рисунке справа. Кольцевой буфер может переполниться, когда HC-Sender передает 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, когда пакеты прибывают меньше чем общая загрузка памяти буфера, HC-Receiver может просто изменить состояние для этого байта. Например, если пакет 12 от HC-Sender прибьют с ECN Nonce 1, Example Buffer будет иметь вид, показанный на рисунке справа.

A.1.2. Старые пакеты

Когда прибывает пакет с Sequence Number S <= buf_ackno, HC-Receiver будет просматривать таблицу в поисках байта, 10 |0,0|3,0|0,0|3,0|0,4|1,0|0,0| 0
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 12 от HC-Sender прибьют с ECN Nonce 0, Example Buffer будет иметь вид, показанный на рисунке справа.

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 полем 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, реализация 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.

### А.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. 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. Значение Ack Vector следует расцепить в точках, соответствующих предыдущим подтверждениям, поскольку сохраненные поля ack_nonce обеспечивают достаточное информирование для расчета корректных значений Nonce Echo. Реализация следует, подтверждая данные по крайней мере один раз на 253 байта буферного пространства (иначе не будет возможности расчета 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.

### А.3. Очистка состояния

Некоторые из пакетов от HC-Sender будут включать номера подтверждений, которые подтверждаются подтверждениями 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

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

DCCP-DataAck с Acknowledgement Number 59. Этот пакет говорит 10 [0,0,3,0,3,0,3,0,2,1] 4 BN[0] получателю HC-Receiver, что отправитель HC-Sender принял и

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

получаю информацию из пакета от HC-Receiver с номером 59. Этот пакет подтверждает полученный от HC-Receiver пакет и HC-Sender имеет от HC-Receiver подтверждения для пакетов 0, 1, 2 и 3. Вид буфера Example Buffer показан на рисунке справа.

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 S.ack_nonce для последующих подтверждений (в данном случае запись HC-Receiver Ack 60 не показана; для нее значение S.ack_nonce меняется на 1). HC-Receiver может также удалить сохраненную информацию для HC-Receiver Ack 59 и всех предшествующих подтверждений.

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

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

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:

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.
Практика покажет насколько неполные контрольные суммы полезны на самом деле. Связано значительных расходов и сложностей, их применение в настоящее время не представляется целесообразным. Реализация которых без этого может оказаться затруднительной. Хотя с поддержкой неполных контрольных сумм не связаны с его заголовком. Таким образом, аутентификация на уровне приложения может потребовать связи с порядковым номерам DCCP, она не обеспечивает достаточной защиты от некоторых MITM-атак. DCCP от обманных пакетов. Альтернативой может служить использование IPsec ESP и шифрование заголовков DCCP. Однако такая аутентификация на «прикладном уровне» не защищает от повторного использования подтверждений или получения запроса о доставке пакета от прикладной программы.

HC-Sender также подтверждает подтверждение от HC-Receiver для удаления старых состояний Ack Vector (поскольку подтверждения Ack Vector достигаются гарантированно, получатель HC-Receiver должен поддерживать информацию Ack Vector и повторно передавать ее, 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 передает данных, полуосоединение HC-Sender может гарантировать, что по крайней мере 1 раз за период кругового обхода оно будет передавать пакет DCCP-Ack, подтверждающий прием последнего пакета DCCP-Ack. Естественно, что полуосоединение HC-Sender требуется подтверждения подтверждения от HC-Receiver только в том случае, когда HC-Sender передает также данные. Если HC-Sender не передает данных, состояние Ack Vector для HC-Receiver будет неизменным и не возникает необходимости в его сжатии. Отправитель HC-Sender должен отслеживать отбрасывание и маркуровку 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 полуосоединение не является статичным (т. е., HC-Receiver передает данные в направлении HC-Sender, возможно используя другой механизм CCID), подтверждений в этом полуосоединении достаточно будет для очистки состояния HC-Receiver.

Приложение B. Обоснование применения 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 поврежденные пакеты, нежели эти пакеты будут просто отбрасываться протоколом. Это связано с тем, что контрольная сумма DCCP не может указать разницу между пакетами, отброшенными в результате повреждения и перегрузки, 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 предпочтает доставку частично поврежденных 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 поврежденные 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 зависеть от того, как частино поврежденные пакеты будут достигаться получателем. Если контрольные суммы канального уровня (CRC) всегда ведут к отбрасыванию поврежденных пакетов, польза от примененных 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 хеш-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 машины состояний DCCP от обмана. Альтернативой может служить использование IPsec ESP и шифрование 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, 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 связаны значительных расходов и сложностей, их применение в настоящее время не представляется целесообразным. Практика показывает насколько неполные контрольные суммы полезны на самом деле.

1man-in-the-middle attack – атака основанная на перехвате пакетов с участием человека. Прим. перев.
Дополнительная литература


Перевод RFC 4340


Фредди Коулер, Эдди Коулер, Е., и Дж. Падхи, "Профиль для Датаграммы Конgestion Control Protocol (DCCP) Конgestion Control ID 3: TCP-Friendly Rate Control (TFRC)"; RFC 4342, Март 2006.


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@protocols.ru

Полное заявление авторских прав

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Интеллектуальная собственность

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights.

Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

1Перевод этой статьи имеется на сайте www.protocols.ru. Прим. перев.
The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Подтверждение

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