Глава 6
ТРАНСПОРТНЫЙ УРОВЕНЬ

В результате освоения главы 6 студент должен:

знать

- структуры заголовков протоколов транспортного уровня стека TCP/IP;
- особенности передачи данных на транспортном уровне;

уметь

- определять рамки применения различных транспортных протоколов;

владеть

- способностью анализировать данные заголовков протоколов транспортного уровня.

В главе рассмотрены основные протоколы транспортного уровня стека TCP/IP: UDP, TCP, SCTP, DCCP. В основу раздела легли материалы из источников [1; 2], а также интернет-стандарты[1].

6.1. Основная концепция протоколов транспортного уровня

На транспортном уровне организована служба надежной доставки данных для верхних уровней, использующая управление потоком и коррекцию ошибок в сквозном потоке. Некоторые реализации транспортного уровня дополнительно осуществляют сегментацию данных при их отправке и воссоздании на приемной стороне. Кроме того, транспортный уровень предоставляет приложению одно или несколько виртуальных соединений, связывающих оконечные точки.

Сегментация данных позволяет разделить большой блок данных, переданных приложением, на более мелкие фрагменты, которые способен передать сетевой уровень. На сетевом уровне выполняется инкапсуля-

[1] *Postel J.* User Datagram Protocol, RFC 768. URL: http://www.faqs.org/ rfcs/rfc768.html; *Postel J.* Transmission Control Protocol, RFC 793. URL: http://www.faqs.org/rfcs/rfc793.html; Stream Control Transmission Protocol, RFC 2960 / R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. URL: http://www.faqs.org/rfcs/rfc2960.html; *Ong L., Yoakum J.* An Introduction to the Stream Control Transmission Protocol (SCTP), RFC 3286. URL: http://www.faqs.org/rfcs/rfc3286.html; *Mogul J., Deering S.* Path MTU discovery, RFC 1191. URL: http://www.faqs.org/rfcs/rfc1191.html; *Floyd S., Handley M., Kohler E.* Problem Statement for the Datagram Congestion Control Protocol (DCCP), RFC 4336. URL: http://www.tools.ietf.org/html/rfc4336; *Kohler E., Handley M., Floyd S.* Datagram Congestion Control Protocol (DCCP), RFC 4340. URL: http://www.tools.ietf.org/html/rfc4340; *Ramakrishnan K., Floyd S., Black D.* The Addition of Explicit Congestion Notification (ECN) to IP, RFC 3168. URL: http://www.tools.ietf.org/html/rfc3168.

ция заголовков пакетов транспортного протокола и прикладных данных, а сформированный пакет передается на канальный уровень.

Управление потоком на транспортном уровне обычно сопровождается ограничением числа пакетов, которые могут быть посланы без подтверждения их приема.

На транспортном уровне семейства протоколов TCP/IP применяются два основных протокола — ориентированный на соединение протокол TCP и не требующий соединения протокол UDP.

Важной концепцией служб транспортного уровня семейства протоколов TCP/IP является концепция *портов*, представляющих собой 16-битный номер и идентифицирующих службу прикладного уровня стека протоколов TCP/IP.

6.2. Протокол UDP

Относить *протокол пользовательских датаграмм* (*User Datagram Protocol, UDP*) (см. RFC 768[1]) к транспортному уровню не вполне корректно. *UDP* ненадежен в том смысле, что доставка пакетов не гарантируется. Протокол *UDP* — это протокол *без установления соединения* (*ConnectionLess*). Он не устанавливает виртуального соединения, не осуществляет никаких повторных передач, не выполняет переупорядочивания пакетов, не управляет потоком данных. Все эти функции возложены на протоколы более высокого уровня (или приложения).

Формат заголовка пакета UDP показан на рис. 6.1. Поле данных на нем не показано. Заметим лишь, что оно выравнивается по 32-битной границе нулевыми байт-заполнителями.

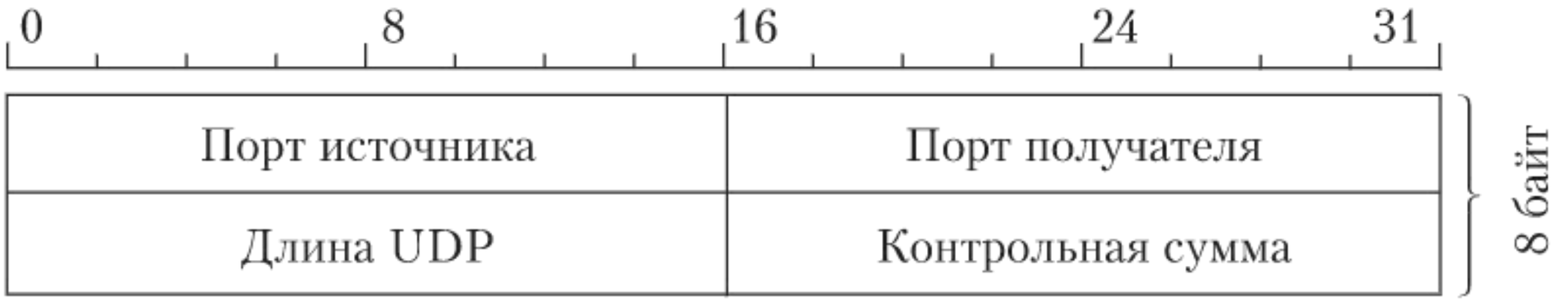

Рис. 6.1. **Формат заголовка пакета UDP**

Поля *Порт источника* (*Source Port*) размером 16 бит и *Порт получателя* (*Destination Port*) длиной 16 бит идентифицируют передающий и получающий процессы соответственно. *Длина UDP* (*Length*) в 16 бит содержит длину пакета UDP в байтах. Поле *Контрольная сумма UDP* (*Checksum*) размерностью в 16 бит содержит контрольную сумму пакета UDP, вычисляемую по всему пакету UDP с добавленным псевдозаголовком (рис. 6.2).

Во время вычисления контрольной суммы это поле выставляется в нуль, а поле данных выравнивается по 32-байтной границе нулевыми байтами. Если контрольная сумма в полученном пакете равняется нулю, то считается, что передающий уровень UDP ее не вычисляет и данные не защищены.

---

[1] *Postel J.* User Datagram Protocol, RFC 768. URL: http://www.faqs.org/rfcs/rfc768.html.

Рис. 6.2. **Структура пакета UDP при вычислении контрольной суммы**

Псевдозаголовок формируется исключительно для работы с контрольной суммой и имеет следующую структуру (рис. 6.3).

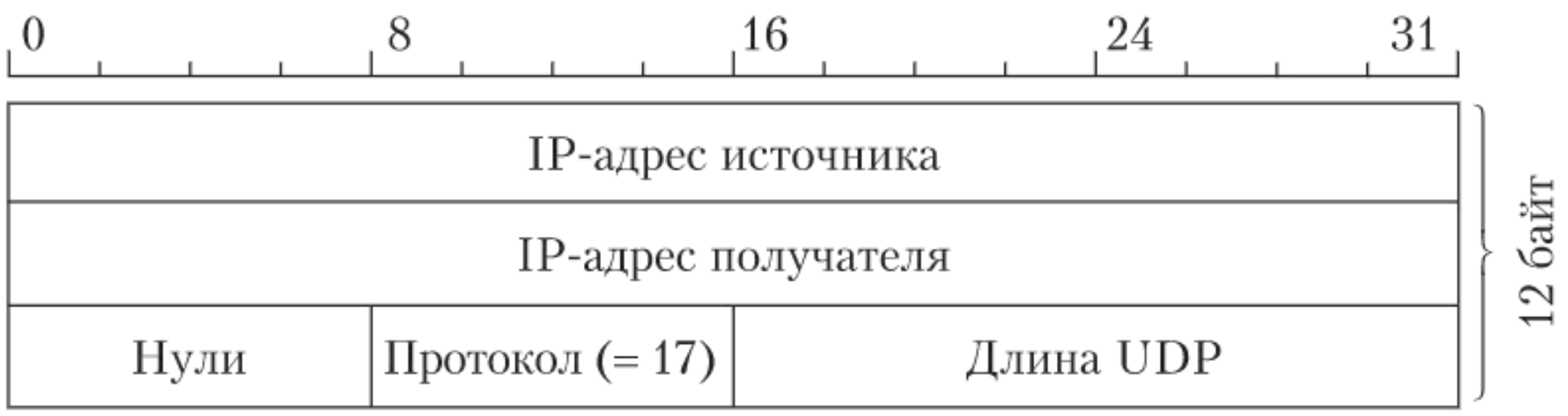

Рис. 6.3. **Структура псевдозаголовка пакета UDP**

Вначале идут поля *IP-адрес источника* в 32 бита и *IP-адрес получателя* длиной 32 бита. Далее идет зарезервированное поле в 8 бит, заполненное нулями. Поле *Протокол* (8 бит) идентифицирует протокол из заголовка пакета IP. Для UDP это значение равно 17 (см. табл. 5.1). Далее идет поле *Длина UDP* (длина 16 бит).

Защита заголовка IP несколько избыточна и делает протокол UDP (впрочем, как и TCP) неотделимым от протокола IP, хотя это и позволяет провести двойную проверку датаграмм IP, поступивших для заданного получателя.

Протоколом UDP пользуются приложения, которым нужно передавать датаграммы последовательно. Например, это такие протоколы, как протокол динамического конфигурирования хостов (*Dynamic Host Configuration Protocol, DHCP*), служба именования доменов (*Domain Name Service, DNS*), простой протокол управления сетью (*Simple Network Management Protocol, SNMP*) и др. Пользуясь UDP, приложение несет ответственность за коррекцию ошибок.

6.3. Протокол TCP

Протокол управления передачей (*Transmission Control Protocol, TCP*) является, в отличие от UDP, «настоящим» протоколом транспортного уровня, который имеет средства управления потоком и коррекции ошибок. Он ориентирован на установление соединения (RFC 793[1])[2].

---

[1] *Postel J.* Transmission Control Protocol, RFC 793. URL: http://www.faqs.org/rfcs/rfc793.html.

[2] Также в следующих RFC: 1323, 1644, 2018, 2581, 2582, 2861, 2873, 2883, 2923, 2988, 3293, 3448, 3465, 3481.

6.3.1. Формат пакета TCP

На рис. 6.4 показана структура заголовка сегмента TCP.

Рис. 6.4. **Формат заголовка пакета TCP**

Поля *Порт источника* (*Source Port*) и *Порт получателя* (*Destination Port*) в 16 бит каждый аналогичны таким же полям в заголовке пакета UDP (см. параграф 6.2) и идентифицируют процесс или приложение, использующее протокол TCP.

Поля *Порядковый номер* (*Sequence Number*) и *Номер подтверждения* (*Acknowledgement Number*) длинами 32 бита нумеруют каждый отправленный или полученный байт данных. Эти поля реализуются как целые числа без знака, которые сбрасываются, когда достигают максимального значения. Каждая сторона ведет собственную порядковую нумерацию.

Поле *Длина заголовка* (*Offset*) в 4 бита содержит размер TCP-заголовка в 32-битных словах. Эта информация необходима, так как поле *Параметры* (*Option*) может быть переменной длины. Можно сказать, что это поле задает смещение от начала сегмента до начала данных в 32-битных словах.

Следом идет неиспользуемое поле (Resrvd) длиной 6 бит. Затем идет поле *Флаги* длиной 6 бит (рис. 6.5).

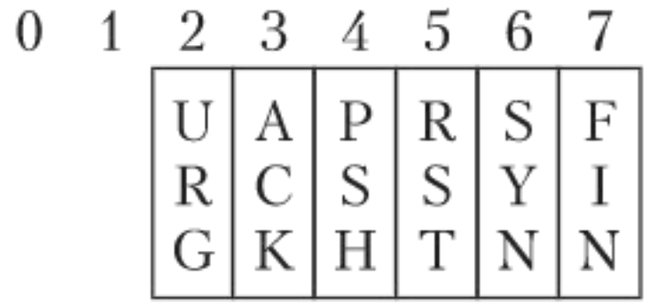

Рис. 6.5. **Поле Флаги заголовка пакета TCP**

Флаг *Указатель срочности* (*Urgent Pointer*, *URG*) устанавливается в 1 в случае использования поля *Указатель на срочные данные*. *Подтверждение* (*Acknowledgment*, *ACK*) устанавливается в 1 в случае, если поле *Номер подтверждения* (*Acknowledgement Number*) содержит данные. В противном случае это поле игнорируется.

Флаг *Выталкивание* (*Push*, *PSH*) означает, что принимающий стек TCP должен немедленно информировать приложение о поступивших данных, а не ждать, пока буфер заполниться. Большинство современных реализа-

ций TCP просто игнорируют флаг *PSH* во время приема пакетов. Этот флаг оставлен по историческим причинам.

Сброс (*Reset*, *RST*) используется для отмены соединения из-за ошибки приложения, отказа от неверного сегмента, попытки создать соединение при отсутствии затребованного сервиса. *Синхронизация* (*Synchronize*, *SYN*) устанавливается при инициировании соединения и синхронизации порядкового номера. Флаг *Завершение* (*Finished*, *FIN*) используется для разрыва соединения. Он указывает, что отправитель закончил передачу данных.

Управление потоком в протоколе TCP осуществляется при помощи скользящего окна переменного размера. Поле *Размер окна* (*Window*) длиной 16 бит содержит количество байт, которое может быть послано после байта, получение которого уже подтверждено. Если значение этого поля равно нулю, это означает, что все байты, вплоть до байта с номером *Номер подтверждения* − 1, получены, но получатель отказывается принимать дальнейшие данные. Разрешение на дальнейшую передачу выдается отправкой сегмента с таким же значением поля *Номер подтверждения* и ненулевым значением поля *Размер окна*.

Поле *Контрольная сумма TCP* (*Checksum*) в 16 бит содержит контрольную сумму пакета TCP, вычисляемую по всему пакету TCP с добавленным псевдозаголовком (рис. 6.6). Во время вычисления контрольной суммы это поле выставляется в нуль, а поле данных выравнивается по 32-байтной границе нулевыми байтами.

Рис. 6.6. **Структура пакета TCP при вычислении контрольной суммы**

Псевдозаголовок формируется исключительно для работы с контрольной суммой и имеет следующую структуру (рис. 6.7).

Рис. 6.7. **Структура псевдозаголовка пакета TCP**

Вначале идут поля *IP-адрес источника* и *IP-адрес получателя* (длинами 32 бит). Далее идет зарезервированное поле (длина 8 бит), заполненное нулями. Поле *Протокол* (8 бит) идентифицирует протокол из заголовка пакета IP. Для TCP это значение равно 6 (см. табл. 5.1). Далее идет поле

Длина TCP (в 16 бит). Поле *Указатель на срочные данные* (длина 16 бит) содержит смещение в байтах от текущего порядкового номера байта до места расположения срочных данных. Содержимым срочных данных занимаются вышестоящие уровни.

Поле *Параметры* (*Option*) (длина переменная, кратная 32 битам) содержит дополнительные поля, расширяющие возможности стандартного заголовка. Это поле зарезервировано для будущего применения и в заголовке может отсутствовать. В настоящее время определены опции:

- конец списка опций;
- никаких операций (используется для заполнения поля опции до числа октетов, кратного 4);
- максимальный размер сегмента (Maximum Segment Size, MSS), задающий верхний размер поля данных.

Данные в TCP-сегменте могут и отсутствовать, характер и формат передаваемой информации задаются исключительно прикладной программой, теоретически максимальный размер этого поля составляет в отсутствие опций 65 495 байт.

6.3.2. Установление сессии TCP

Поля *Порядковый номер* (*Sequence Number*) и *Номер подтверждения* (*Acknowledgment Number*) играют роль счетчика пакетов. При установлении сессии используется поле флагов.

Установление связи клиент-сервер осуществляется в три этапа (трехступенчатый handshake), что показано на рис. 6.8.

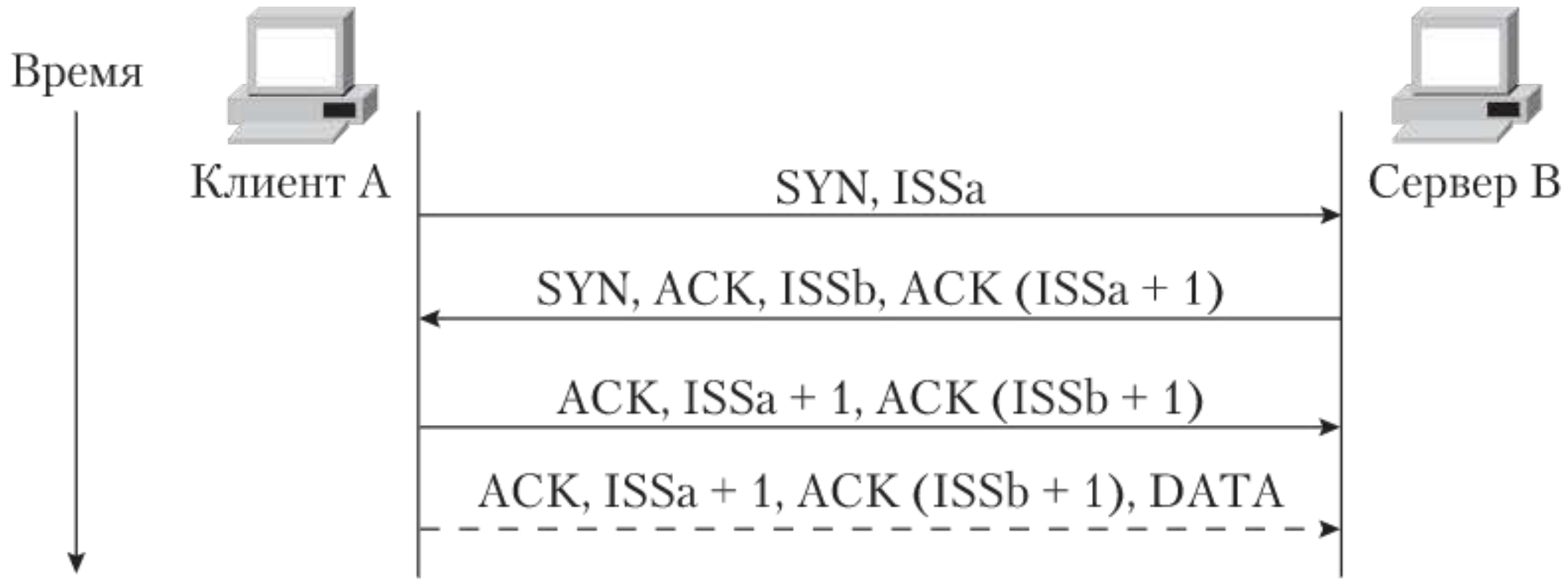

Рис. 6.8. **Трехступенчатый handshake**

Пусть хост A создает соединение с хостом B.

1. *Режим активного доступа* (*Active Open*). Клиент посылает сообщение *SYN, ISSa*, т.е. в передаваемом сообщении установлен бит SYN (Synchronize Sequence Number), а в поле *Порядковый номер* (*Sequence Number*) — начальное 32-битное значение ISSa (Initial Sequence Number).

2. *Режим пассивного доступа* (*Passive Open*). Сервер откликается, посылая сообщение *SYN, ACK, ISSb, ACK* (*ISSa + 1*), т.е. установлены биты SYN и ACK; в поле *Порядковый номер* хостом B устанавливается начальное значение счетчика — ISSb; поле *Номер подтверждения* (*Acknowledgment Number*) содержит значение ISSa, полученное в первом пакете от хоста A и увеличенное на единицу.

3. *Завершение рукопожатия.* Клиент отправляет подтверждение получения SYN-сегмента от сервера с идентификатором, равным ISN (сервера) + 1: *ACK*, *ISSa + 1*, *ACK* (*ISSb + 1*). В этом пакете установлен бит ACK, поле *Порядковый номер* содержит ISSa + 1, поле *Номер подтверждения* содержит значение ISSb + 1. Посылкой этого пакета заканчивается трехступенчатый handshake, и TCP-соединение считается установленным.

4. Теперь клиент может посылать пакеты с данными на сервер по только что созданному виртуальному TCP-каналу: *ACK*, *ISSa + 1*, *ACK* (*ISSb + 1*); *DATA*.

Из рассмотренной выше схемы создания TCP-соединения видно, что единственными идентификаторами TCP-абонентов и TCP-соединения являются два 32-битных параметра *Порядковый номер* и *Номер подтверждения*.

6.3.3. Управление потоком

Для ускорения и оптимизации процесса передачи больших объемов данных протокол TCP определяет метод управления потоком, называемый *методом скользящего окна*, который позволяет отправителю посылать очередной сегмент, не дожидаясь подтверждения о получении в пункте назначения предшествующего сегмента.

Протокол TCP формирует подтверждения не для каждого конкретного успешно полученного пакета, а для всех данных от начала посылки до некоторого порядкового номера *ACK SN* (*Acknowledge Sequence Number*). В качестве подтверждения успешного приема, например, первых n байт, высылается ACK SN = n +1: это означает, что все данные в байтовом потоке под номерами от ISN + 1 = 1 до n успешно получены.

Вместе с посылкой отправителю порядкового номера ACK SN получатель объявляет также размер окна. Это значит, что отправитель может посылать данные с порядковыми номерами от текущего ACK SN до (ACK SN + размер окна − 1), не дожидаясь подтверждения со стороны получателя. Если не будет получено новое подтверждение (новый ACK SN), отправитель будет посылать данные, пока он остается в пределах объявленного окна. После этого посылка данных будет прекращена до получения очередного подтверждения и нового размера окна.

Размер окна выбирается таким образом, чтобы подтверждения успевали приходить вовремя и остановки передачи не происходило. Размер окна может динамически изменяться получателем.

Для временной остановки посылки данных достаточно объявить нулевое окно. Но даже в этом случае через определенные промежутки времени будут отправляться сегменты с одним октетом данных. Это делается для того, чтобы отправитель гарантированно узнал о том, что получатель вновь объявил ненулевое окно, поскольку получатель обязан подтвердить получение пробных сегментов, а в этих подтверждениях он укажет также и текущий размер своего окна. В протоколе TCP скользящее окно используется для регулировки трафика и препятствия переполнению буфера.

Регулирование трафика в TCP подразумевает существование двух независимых процессов: *контроля доставки*, управляемого получателем с помощью параметра *Размер окна* (*Window*), и *контроля перегрузки*, управ-

ляемого отправителем с помощью *Окна перегрузки* (*Congestion Window*, *CWnd*) и *Порога медленного старта* (*Slow Start Threshold*, *SSThreth*).

Первый процесс отслеживает заполнение входного буфера получателя, второй — регистрирует перегрузку канала и связанные с этим потери, а также понижает интенсивность трафика. В исходный момент времени при установлении соединения CWnd делается равным одному MSS (максимальному размеру сегмента), а SSThreth — 65 535 байтам. Программа, управляющая пересылкой, никогда не пошлет больше байт, чем это задано CWnd и объявленным получателем значением *Размера окна* (*Window*). Когда получение очередного блока данных подтверждено, значение CWnd увеличивается. Если значение CWnd меньше или равно значению SSThreth, то выполняется процедура *Медленный старт*, в противном случае осуществляется подавление перегрузки. В последнем случае CWnd_i + 1 = CWnd_i + MSS/8 + (MSS*MSS)/CWnd. Если возникает состояние перегрузки канала, значение CWnd снова делается равным одному MSS. Окно перегрузки позволяет согласовать полную загрузку виртуального соединения, и текущие возможности канала, минимизируя потери пакетов при перегрузке.

Для управления потоком используется *Порог медленного старта* (*SSThresh*). При установлении соединения SSThresh = 64 Кбайт. В случае возникновения таймаута значение SSThresh становится равным CWnd/2, а само значение CWnd приравнивается MSS. Далее запускается процедура медленного старта, чтобы выяснить возможности канала. При этом экспоненциальный рост CWnd осуществляется вплоть до значения SSThresh. Когда этот уровень CWnd достигнут, дальнейший рост происходит линейно с приращением на каждом шаге, равном MSS.

6.3.4. Проблемы TCP

TCP за годы существования претерпел значительные изменения, касающиеся обеспечения надежности и производительности в сетях различной емкости и качества. Но при этом возможности TCP уже не удовлетворяют современным потребностям.

- TCP не подходит для передачи данных в VoIP-сетях или для асинхронной обработки на базе транзакций.
- TCP требует строго упорядоченной передачи данных, что не подходит для приложений, допускающих как последовательную, так и непоследовательную доставку потоков.
- TCP не структурирует последовательности передаваемых данных. Поэтому требуется добавление разграничителей сообщений.
- TCP не поддерживает множественную адресацию.

TCP-хосты восприимчивы к атакам «отказ в обслуживании» (Denial of Service, DoS) типа SYN DoS (или FIN DoS)[1].

---

[1] Хосту посылается огромное количество пакетов TCP SYN. Хост-получатель резервирует память и отвечает на запрос сообщениями SYN ACK. Когда атакующая система не возвращает сообщения ACK, необходимые для завершения процедуры установки TCP-соединения, ресурсы хоста, подвергнувшегося атаке, остаются неосвобожденными. Поэтому он оказывается не готов к обслуживанию других запросов.

6.4. Протокол SCTP

Протокол управления потоковой передачей (*Stream Control Transmission Protocol, SCTP*) (RFC 2960[1], RFC 3286[2]) можно рассматривать как дальнейшее логическое развитие протокола TCP. Как и TCP, протокол SCTP предлагает приложениям, взаимодействующим по IP-сети, ориентированную на соединения типа точка — точка транспортную службу с надежной доставкой. Протокол унаследовал часть функциональности TCP, в том числе возможность контроля перегрузки и восстановления утерянных пакетов. Любое приложение, работающее по протоколу TCP, можно перевести на SCTP без потери функциональности.

6.4.1. Формат пакета SCTP

Сообщения SCTP включают общий заголовок, за которым следует один или несколько *подпакетов* (*Chunk*), которые могут содержать данные или управляющую информацию (рис. 6.9). В заголовке (рис. 6.10) указываются номера портов отправителя и получателя, что позволяет мультиплексировать различные ассоциации SCTP на одном адресе.

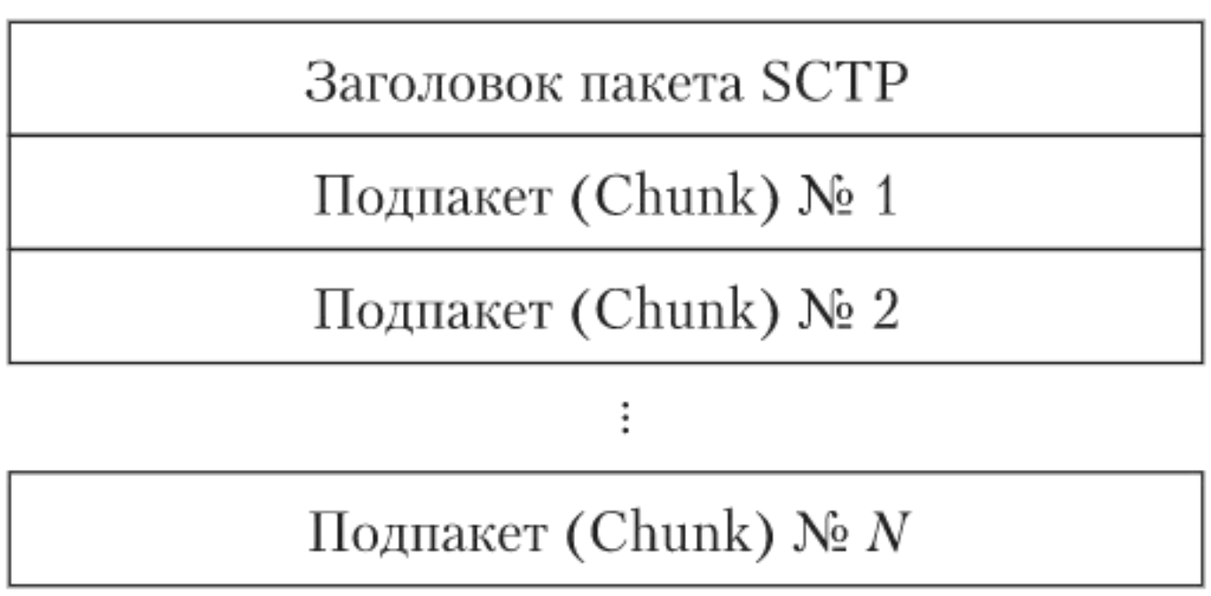

Рис. 6.9. **Формат пакета SCTP**

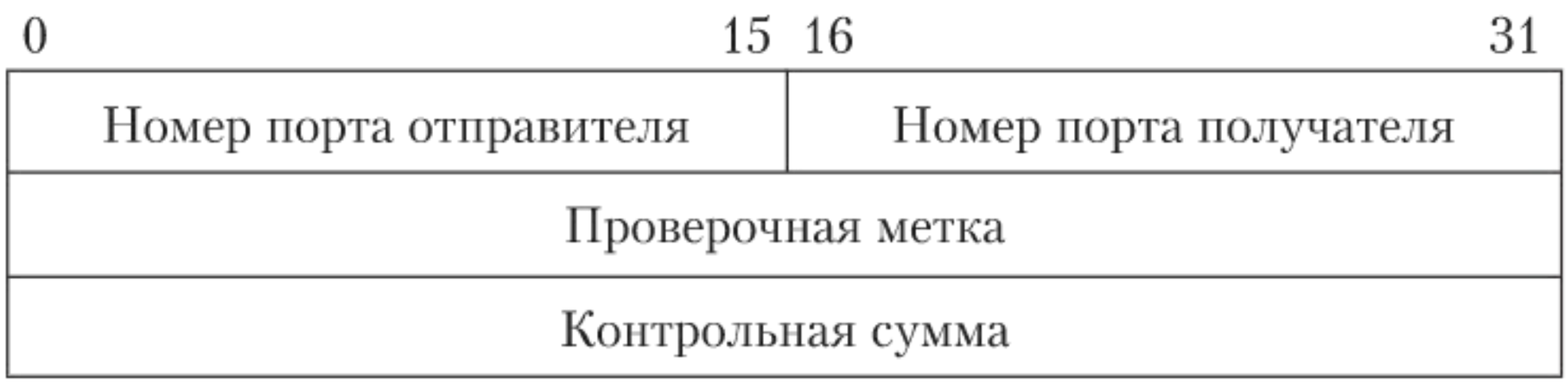

Рис. 6.10. **Формат заголовка пакета SCTP**

Проверочная метка (*Verification Tag*) в 32 бита предотвращает возможность включения в ассоциацию SCTP устаревших или фальсифицированных сообщений. *Контрольная сумма* (длина 32 бита) рассчитывается на основе полиномиального алгоритма CRC-32c и служит для выявления ошибок.

---

[1] Stream Control Transmission Protocol, RFC 2960 / R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. URL: http://www.faqs.org/rfcs/rfc2960.html.

[2] *Ong L., Yoakum J.* An Introduction to the Stream Control Transmission Protocol (SCTP), RFC 3286. URL: http://www.faqs.org/rfcs/rfc3286.html.

Формат подпакета. Каждый подпакет (фрагмент) содержит поля *Тип подпакета* (*Chunk ID*), *Флаги* (*Chunk Flags*), *Длина подпакета* (*Chunk Length*), *Данные* (*Chunk Value*) (рис. 6.11).

Рис. 6.11. **Формат подпакета SCTP**

Восьмибитное поле типа подпакета способно принимать до 255 значений (в настоящее время определены 15, а остальные зарезервированы). Если данное поле имеет нулевое значение, то это говорит о передаче *полезной информации* (*Payload Data*); в других случаях подпакет несет служебные сведения.

Второе поле — восьмибитное поле *флагов*, его использование определяется типом подпакета. Поле *длины* с разрядностью 16 бит заполняется суммарным значением длины подпакета с учетом полей заголовка.

Управляющие блоки включают различные параметры и флаги, зависящие от типа блока. Подпакеты *данных* (*DATA*) включают флаг управления сегментацией и сборкой, а также параметры TSN, Stream ID, Stream Sequence Number и Payload Protocol Identifier.

Перед фрагментом DATA размещаются *номер транспортной последовательности* (*Transport Sequence Number, TSN*), *идентификатор потока*, *номер последовательности потока* (*Stream Sequence Number, SSN*).

Номер транспортной последовательности используется для обеспечения надежности каждой ассоциации, а номер последовательности потока — для упорядочивания по потокам. Отдельные сообщения в потоке отмечаются идентификатором потока.

Информационная часть предназначена для передачи собственно данных, которые определяются типом подпакета. Согласно протоколу SCTP, размерность подпакета должна быть кратна 32 битам. В противном случае информационная часть дополняется нулевыми значениями, но в поле длины указывается истинная величина. Это позволяет на приемной стороне соединения исключить добавленные нули из передаваемых данных.

Параметр *Payload Protocol ID* включен для обеспечения возможности расширения в новых версиях протокола. Если предположить, что функции идентификации протокола и мультиплексирования по портам в будущем перестанут играть столь важную роль, как сейчас, *Payload Protocol ID* будет обеспечивать идентификацию протоколов, передаваемых с помощью SCTP без использования номера порта.

Формат сообщений SCTP обеспечивает механизм связывания множества блоков данных и управления в одно сообщение для повышения эффективности транспорта. Использованием такой *группировки* (*Bundling*) управляет приложение, поэтому группировка стартовой передачи невоз-

можна. Связывание естественным образом осуществляется при повторе передачи блоков *DATA* в целях снижения вероятности насыщения.

6.4.2. Функции SCTP

SCTP представляет собой Unicast-протокол, который обеспечивает обмен данными между двумя конечными точками.

Аналогом TCP-соединения для SCTP является ассоциация, которая устанавливается между двумя оконечными устройствами. При этом одно устройство может быть определено несколькими IP-адресами, список которых передается при установлении ассоциации. Для передачи данных через ассоциацию используются все возможные комбинации адресов пары оконечных устройств.

Отказоустойчивость в таком случае обеспечивается за счет того, что разные IP-адреса присваиваются различным интерфейсам устройств и трафик между ними передается по разным маршрутам. В случае отказа какого-либо оборудования в сети и недоступности одного или нескольких IP-адресов трафик продолжает передаваться между оставшимися адресами и разрыва SCTP-ассоциации не происходит.

Описанный выше механизм работы SCTP-ассоциации носит название *многодомности* (*SCTP Multi-Homing*).

К другим ключевым функциям протокола SCTP относятся:

• группировка различных сигнальных сообщений в одном пакете с одним SCTP/IP-заголовком (Chunk Bundling), что повышает эффективность использования полосы пропускания;

• последовательная доставка сообщений внутри различных потоков, что позволяет избежать ситуации, встречающейся при использовании протокола TCP, когда в случае потери одного пакета остальные задерживаются в буфере до успешной его перепосылки (Head-of-Line Blocking);

• использование контрольных сумм для обеспечения безошибочной передачи пакетов, а также для защиты от атак.

Протокол SCTP поддерживает ряд функций, унаследованных не только от TCP, но и от других протоколов. При этом в нем реализованы и дополнительные функции:

— *сохранение границ сообщений*. Сообщения, передаваемые SCTP, размещаются в подпакетах (или фрагментах), что дает возможность приложениям отделить одно сообщение от другого;

— *отсутствие блокировок типа head-of-line*. В отличие от TCP протокол SCTP не требует строгой упорядоченности передаваемых пакетов. Поэтому в нем отсутствует задержка, вызываемая блокировкой обслуживания, возникающей при восстановлении TCP корректной последовательности пакетов;

— *несколько режимов доставки*. SCTP может передавать данные как в строгом порядке (как TCP), так и частично упорядоченные (по потокам) и неупорядоченные вовсе (как UDP);

— *поддержка многодомности*. SCTP может переадресовывать пакеты на альтернативный IP-адрес;

— *контроль перегрузки*. SCTP использует стандартные методики, применяющиеся для контроля перегрузки в TCP, в том числе медленный старт, предотвращение перегрузки и быструю повторную передачу;

— *выборочные подтверждения*. SCTP использует схему выборочного подтверждения, унаследованную из TCP, для восстановления утраченных пакетов;

— *фрагментация пользовательских данных*. SCTP разбивает сообщения на фрагменты, чтобы *максимальный размер передаваемого элемента* (*Maximum Transfer Unit*, *MTU*) соответствовал ограничениям конкретного маршрута пересылки между взаимодействующими хостами (RFC 1191[1]);

— *механизм контроля работоспособности* (*Hearbeat*). SCTP посылает пакеты контроля работоспособности на адреса находящегося в режиме ожидания хоста, которые входят в ассоциацию. Протокол декларирует, что IP-адрес будет отключен, как только он достигнет порогового значения невозвращенных подтверждений о работоспособности;

— *защита от DoS-атак*. SCTP использует механизм cookie при инициализации ассоциации, чтобы смягчить воздействие DoS-атак.

6.4.3. Множественность потоков и варианты доставки

Название протокола SCTP обусловлено его многопотоковой природой передачи данных. Поддержка множества одновременных потоков позволяет распределить между этими потоками передаваемую информацию так, чтобы каждый из потоков обеспечивал независимую упорядоченную доставку данных. Потеря сообщения в любом из потоков оказывает влияние лишь на данный поток, не затрагивая работу других потоков данных.

Протокол TCP работает с одним потоком данных и обеспечивает сохранение порядка доставки байт из потока. Такой подход удобен для доставки файлов или записей, но он может приводить к дополнительным задержкам при потере информации в сети или нарушении порядка доставки пакетов. При возникновении подобных ситуаций протокол TCP должен дождаться доставки всех данных, требуемых для восстановления порядка.

В рамках одного соединения SCTP обеспечивает единый механизм управления потоком и контроля насыщения, что существенно снижает нагрузку на транспортный уровень.

SCTP разделяет понятия надежной и упорядоченной доставки, в то время как в TCP эти два аспекта неразрывно связаны, так как все данные надежно доставляются хосту-получателю и предоставляются приложению в той последовательности, в какой они передавались. Для этого TCP использует номер последовательности в заголовке каждого пакета.

Протокол SCTP поддерживает многопотоковую передачу за счет устранения зависимости между передачей и доставкой данных. В частности, каждый блок полезной информации типа DATA (данные) использует два набора порядковых номеров. Номер TSN управляет передачей сообщений и детектированием их потери, а пара *идентификатор потока Stream*

---

[1] *Mogul J., Deering S.* Path MTU discovery, RFC 1191. URL: http://www.faqs.org/rfcs/rfc1191.html.

ID-номер SSN используется для управления порядком доставки потребителю полученных данных.

Такая независимость механизмов нумерации позволяет получателю незамедлительно обнаруживать пропуски данных, а также видеть влияние потерянных данных на поток. Утрата сообщения вызывает появление пропуска в порядковых номерах SSN для потока, на который это сообщение оказывает влияние и не вызывает такого пропуска для других потоков. Следовательно, получатель может продолжить доставку незатронутых потоков, не дожидаясь повтора передачи утраченного сообщения.

6.4.4. Многодомность

Механизм многодомности предназначен для повышения устойчивости сети к выходам из строя интерфейсов на хосте и ускорения восстановления в случае сбоя в сети. Но эффективность этого механизма падает, если путь взаимодействия внутри ассоциации проходит через единую точку сбоя сети.

Действующий вариант SCTP не поддерживает *распределения нагрузки* (*Load Sharing*), поэтому многодомные хосты обеспечивают лишь избыточность соединений для повышения уровня надежности. Один из адресов многодомного хоста указывается в качестве *основного* (*Primary*) и используется как адрес получателя для всех блоков данных при нормальной передаче. При передаче повторных блоков данных используется один из дополнительных адресов с целью повышения вероятности доставки в конечную точку. При повторяющихся неоднократно повторах передачи принимается решение об отправке всех блоков данных с использованием альтернативного адреса, пока системе мониторинга не удастся увидеть доступность основного адреса.

Для поддержки множества интерфейсов конечные точки SCTP обмениваются списками своих адресов в процессе создания ассоциации. Каждая из конечных точек должна быть способна принимать сообщения с любого адреса, связанного с удаленным партнером; на практике некоторые операционные системы могут использовать в пакетах циклический перебор адресов отправителя, и в таких случаях прием пакетов с различных адресов является нормальной ситуацией. Для всего списка адресов конечной точки в данной сессии используется один номер порта.

Для повышения уровня безопасности требуется, чтобы некоторые отклики передавались по адресу, указанному в поле отправителя сообщения, вызвавшего отклик. Например, когда сервер получает блок INIT от клиента для инициирования SCTP-ассоциации, сервер всегда будет передавать блок INIT ACK по адресу отправителя в заголовке IP-блока INIT.

6.4.5. Установление ассоциаций

SCTP, как и TCP, ориентирован на установление соединения. Оба протокола требуют определения состояния соединения на каждом хосте. Соединение TCP определяется двумя IP-адресами и двумя номерами портов. Ассоциация SCTP определяется как набор IP-адресов + порт на каждом хосте. Любые из IP-адресов на любом хосте могут указываться в качестве

отправителя или получателя в IP-пакете, и это корректно идентифицирует ассоциацию.

Перед началом обмена данными два SCTP-хоста должны передать друг другу информацию о состоянии соединений с помощью четырехэтапной процедуры установки соединения (handshake), что показано на рис. 6.12.

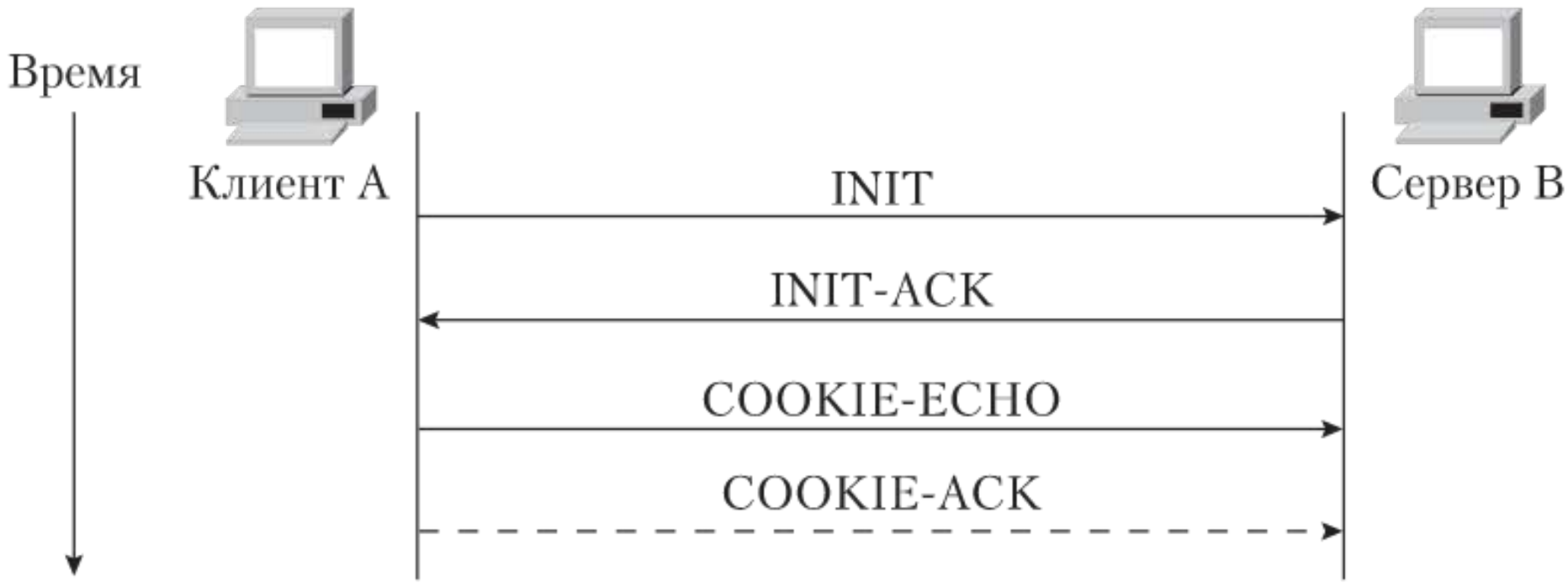

Рис. 6.12. **Четырехэтапная процедура установки соединения SCTP**

Процедура, предусмотренная протоколом SCTP, позволяет защититься от DoS-атак. Получателю сообщения о намерении установить контакт *INIT* в четырехэтапной процедуре установки соединения не требуется сохранять никакую информацию о состоянии или резервировать какие-либо ресурсы. Вместо этого он посылает в ответ сообщение *INIT-ACK*, которое включает в себя *запись состояния* (*Cookie*), содержащую всю информацию, необходимую отправителю *INIT-ACK* для того, чтобы сформировать свое состояние. Запись состояния подписывается цифровой подписью.

Оба сообщения, *INIT* и *INIT-ACK*, содержат несколько параметров, необходимых для установки начального состояния:

- список всех IP-адресов, которые станут частью ассоциации;
- номер транспортной последовательности, используемый для надежной передачи данных;
- тег инициации, который должен быть включен в каждый входящий пакет SCTP;
- число выходящих потоков, запрашиваемых каждой из сторон;
- число входящих потоков, которые способна поддерживать каждая из сторон.

После обмена этими сообщениями отправитель INIT возвращает назад запись состояния в виде сообщения COOKIE-ECHO, которое также может содержать связанные с ним пользовательские сообщения DATA. При получении COOKIE-ECHO получатель полностью меняет свое состояние и отправляет обратное сообщение COOKIE-ACK, подтверждающее завершение настройки. COOKIE-ACK также может сопровождаться пользовательскими сообщениями DATA.

6.4.6. Завершение работы ассоциации

Транспортному протоколу, ориентированному на соединение, необходим метод постепенного отключения ассоциации. SCTP использует про-

цедуру установки соединения, отличающуюся от процедуры, применяемой в TCP: конечная точка TCP может инициировать процедуру отключения, сохраняя открытым соединение и получая новые данные от другого хоста. SCTP не поддерживает такого наполовину закрытого состояния, т.е. обе стороны не могут передавать новые данные на свой более высокий уровень, если инициирована последовательность постепенного отключения.

Пусть приложение на хосте A хочет отключить и закрыть ассоциацию с хостом B. SCTP устанавливает состояние SHUTDOWN_PENDING, в котором он не будет принимать данные от приложения, но по-прежнему будет посылать новые данные, помещаемые в очередь на передачу на хост B. После подтверждения всех размещенных в очереди данных хост A посылает подпакет SHUTDOWN и устанавливает состояние SHUTDOWN_SENT.

До получения подпакета SHUTDOWN хост B уведомляет свой более высокий уровень, что прекращает принимать от него новые данные и вводит состояние SHUTDOWN_RECEIVED. Хост B передает оставшиеся данные на A, за которыми следуют фрагменты SHUTDOWN, информирующие B о появлении данных и подтверждающие, что ассоциация отключена. Как только подтверждены все данные, помещенные в очередь на хосте B, хост A посылает соответствующий фрагмент SHUTDOWN-ACK, за которым следует фрагмент SHUTDOWN-COMPLETE, завершающий отключение ассоциации.

6.5. Протокол DCCP

Протокол DCCP (*Datagram Congestion Control Protocol*)[1] является транспортным протоколом, который использует двунаправленные уникастные соединения с управлением перегрузкой для ненадежной доставки дейтаграмм.

Протокол DCCP имеет встроенную систему управления перегрузкой, включающую поддержку *уведомления о перегрузке канала* (*Explicit Congestion Notification, ECN*)[2] для ненадежных потоков дейтаграмм, исключая непредсказуемые задержки, характерные для TCP, что обеспечивает надежное согласование параметров при установлении соединения.

6.5.1. Характеристики DCCP

Протокол DCCP обладает следующими характеристиками:

- является протоколом для потоков пакетов, а не потоков байт;
- реализует поток дейтаграмм с подтверждением получения, но без повторной посылки;
- имеет ненадежный диалог установления и разрыва соединения;

---

[1] *Floyd S., Handley M., Kohler E.* Problem Statement for the Datagram Congestion Control Protocol (DCCP), RFC 4336. URL: http://tools.ietf.org/html/rfc4336; *Kohler E., Handley M., Floyd S.* Datagram Congestion Control Protocol (DCCP), RFC 4340. URL: http://www.tools.ietf.org/html/rfc4340.

[2] *Ramakrishnan K., Floyd S., Black D.* The Addition of Explicit Congestion Notification (ECN) to IP, RFC 3168. URL: http://www.tools.ietf.org/html/rfc3168.

- обеспечивает надежное согласование параметров;
- предоставляет выбор механизмов подавления перегрузки;
- является протоколом управления перегрузкой, а не протоколом управления потоками;
- имеет опции, указывающие отправителю, был ли пакет доставлен получателю, помечен ECN, поврежден или отброшен входным буфером получателя;
- осуществляет управление перегрузкой со встроенной индикацией явной перегрузки ECN;
- обладает механизмами, позволяющими серверу избежать поддержки состояний неподтвержденных попыток соединений;
- выявляет MTU пути.

6.5.2. Типы сообщений DCCP

Протокол DCCP использует девять различных типов сообщений:

— DCCP-Request инициирует соединение;

— DCCP-Response является ответом на запрос DCCP-Request;

— DCCP-Data передает данные;

— DCCP-Ack передает подтверждения о получении пакетов;

— DCCP-DataAck передает данные в сочетании с подтверждениями;

— DCCP-CloseReq запрашивает закрытие соединения;

— DCCP-Close осуществляет закрытие соединения или запускает процедуру сброса соединения (DCCP-Reset);

— DCCP-Reset осуществляет процедуру сброса соединения;

— DCCP-Sync, DCCP-SyncAck осуществляют повторную синхронизацию номеров пакетов после длительного периода потерь.

6.5.3. Формат заголовка DCCP

Базовый заголовок DCCP имеет следующий формат (рис. 6.13).

Рис. 6.13. **Формат базового заголовка DCCP**

Поля *Порт отправителя* (*Source Port*) и *Порт получателя* (*Dest Port*) длиной по 16 бит каждый идентифицируют соединение. Когда соединение формируется, клиент должен выбрать порт отправителя случайным образом, чтобы уменьшить вероятность атаки. *Смещение данных* (*Data Offset*) в 8 бит указывает смещение от начала заголовка пакета DCCP пер-

вого октета данных (выражается в 32-битных словах). Поле *CCVal* (длина 4 бита) используется отправителем CCID.

Checksum Coverage (CsCov) размерностью 4 бита определяет части пакета, которые покрываются полем *Контрольная сумма* (*Checksum*) в 16 бит, которая содержит контрольную сумму заголовка пакета DCCP (включая опции), псевдозаголовка сетевого уровня и, в зависимости от CsCov, полей данных приложений. Поле *Зарезервировано* (*Reserved*) длиной 3 бита содержит нули, получатель должен это поле игнорировать.

Тип (*Type*) в 4 бита специфицирует тип пакета. Поле *Расширенные порядковые номера* (*X*), длина которого 1 бит, равно нулю, если передаются только младшие (LSB) 24 бита порядкового номера, а базовый заголовок имеет длину 12 байт и значение 1, если в заголовке используются 48-разрядные порядковые номера. Пакеты DCCP-Data, DCCP-DataAck и DCCP-Ack могут иметь значение X, равное 0 или 1. Все пакеты DCCP-Request, DCCP-Response, DCCP-CloseReq, DCCP-Close, DCCP-Reset, DCCP-Sync и DCCP-SyncAck должны иметь X = 1.

Поле *Порядковый номер* (*Sequence Number*) длиной 48 или 24 бита идентифицирует пакет в последовательности. Номер по порядку увеличивается на 1 после посылки каждого пакета, включая пакеты DCCP-Ack, которые не несут в себе данных. После базового заголовка следует заголовок пересылаемого типа пакета.

6.5.4. Процедура взаимодействия

Процедура взаимодействия двух элементов следующая.

1. Клиент посылает серверу запрос DCCP-Request на установление соединения. Определяются номера портов клиента и сервера, запрашиваемая услуга и другие параметры соединения, включая CCID, необходимый серверу при работе с клиентом.
2. В ответ сервер посылает пакет-отклик.
3. Клиент посылает серверу подтверждение DCCP-Ack получения DCCP-отклика.
4. Далее по необходимости происходит обмен подтверждениями DCCP-Ack для согласования используемых параметров.
5. Сервер и клиент обмениваются пакетами DCCP-Data, DCCP-Ack.
6. Для закрытия соединения сервер посылает DCCP-CloseReq.
7. Для подтверждения закрытия соединения клиент посылает DCCP-Close.
8. Сервер посылает пакет DCCP-Reset, при этом состояние соединения ликвидируется.
9. Клиент получает пакет DCCP-Reset и сохраняет свое состояние в течение некоторого времени для завершения происходящих обменов.

6.5.5. Функциональность DCCP

Протокол DCCP может реализовать механизм контроля за перегрузкой, многодомность и мобильность (за счет механизма переадресации), процедуру медленного получателя (Slow Receiver). DCCP не предоставляет

криптографических гарантий безопасности, но имеет возможности противостоять некоторым видам атак благодаря используемой системе нумерации пакетов.

Резюме

1. На транспортном уровне организована служба надежной доставки данных для верхних уровней, использующая управление потоком и коррекцию ошибок в сквозном потоке.
2. Адресом транспортного уровня является номер порта — 16-битный номер, идентифицирующих службу прикладного уровня стека протоколов TCP/IP.
3. На транспортном уровне семейства протоколов TCP/IP применяются два основных протокола — ориентированный на соединение протокол TCP и не требующий соединения протокол UDP.
4. Протокол UDP — это протокол без установления соединения (ConnectionLess). Он не устанавливает виртуального соединения, не осуществляет никаких повторных передач, не выполняет переупорядочивания пакетов, не управляет потоком данных. Все эти функции возложены на протоколы более высокого уровня (или приложения).
5. Протокол TCP имеет средства управления потоком и коррекции ошибок. Он ориентирован на установление соединения, поэтому клиент обязан установить соединение с сервером до начала передачи данных TCP в любом из направлений. Управление потоком в протоколе TCP осуществляется при помощи скользящего окна переменного размера. Установление связи клиент-сервер осуществляется с помощью процедуры «трехступенчатый handshake».
6. Протокол SCTP унаследовал часть функциональности TCP, в том числе возможность контроля перегрузки и восстановления утерянных пакетов. Любое приложение, работающее по протоколу TCP, можно перевести на SCTP без потери функциональности. Аналогом TCP-соединения для SCTP является ассоциация, которая устанавливается между двумя оконечными устройствами. При этом одно устройство может быть определено несколькими IP-адресами, список которых передается при установлении ассоциации. Для передачи данных через ассоциацию используются все возможные комбинации адресов пары оконечных устройств. Отказоустойчивость обеспечивается механизмом многодомности — SCTP может переадресовывать пакеты на альтернативный IP-адрес. SCTP не требует строгой упорядоченности передаваемых пакетов.
7. SCTP разделяет понятия надежной и упорядоченной доставки. SCTP поддерживает многопотоковую передачу за счет устранения зависимости между передачей и доставкой данных. Установление связи клиент-сервер осуществляется с помощью процедуры «четырехступенчатый handshake».
8. Протокол DCCP является транспортным протоколом, который использует двунаправленные уникастные соединения с управлением перегрузкой для ненадежной доставки дейтаграмм. DCCP имеет встроенную систему управления перегрузкой, включающую поддержку уведомления о перегрузке канала (ECN), что обеспечивает надежное согласование параметров при установлении соединения. DCCP может реализовать механизм контроля за перегрузкой, многодомность и мобильность (за счет механизма переадресации), процедуру медленного получателя (Slow Receiver). DCCP не предоставляет криптографических гарантий безопасности, но имеет возможности противостоять некоторым видам атак благодаря используемой системе нумерации пакетов.

Контрольные вопросы

1. Каковы функции и услуги транспортного уровня модели ISO/OSI?
2. В чем состоят принципиальные отличия протоколов TCP и UDP?

3. Опишите схему управления потоком в протоколе TCP.
4. Опишите схему установления сессии TCP.
5. В чем заключаются основные отличия протоколов TCP и SCTP?
6. Опишите функциональность протокола SCTP.
7. В чем заключается механизм многодомности в протоколе SCTP?
8. Опишите схему установления сессии SCTP.
9. Укажите основные характеристики протокола DCCP.
10. В чем заключаются основные отличия протокола DCCP от других транспортных протоколов?
11. Опишите схему установления сессии DCCP.

Задания для самостоятельной работы

1. Объясните, для чего, кроме подсчета пакетов, могут служить поля *Порядковый номер* (*Sequence Number*) и *Номер подтверждения* (*Acknowledgment Number*) пакета TCP.
2. Исследуйте зависимость производительности протокола TCP от размеров окна.
3. Обрисуйте область применения протокола UDP, исходя его из свойств.
4. Обрисуйте область применения протокола TCP, исходя из его свойств.
5. Обрисуйте область применения протокола SCTP, исходя из его свойств.
6. Обрисуйте область применения протокола DCCP, исходя из его свойств.