Транспортный уровень 2

UDP (англ. *User Datagram Protocol* — протокол пользовательских датаграмм)

- С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных. Протокол был разработан Дэвидом П. Ридом в 1980 году и официально определён в RFC 768.
- UDP использует простую модель передачи, без неявных «рукопожатий» для обеспечения надёжности, упорядочивания или целостности данных. Таким образом, UDP предоставляет ненадёжный сервис, и датаграммы могут прийти не по порядку, дублироваться или вовсе исчезнуть без следа. UDP подразумевает, что проверка ошибок и исправление либо не необходимы, либо должны исполняться в приложении. Чувствительные ко времени приложения часто используют UDP, так как предпочтительнее сбросить пакеты, чем ждать задержавшиеся пакеты, что может оказаться невозможным в системах реального времени.

- Природа UDP как протокола без сохранения состояния также полезна для серверов, отвечающих на небольшие запросы от огромного числа клиентов, например DNS и потоковые мультимедийные приложения вроде IPTV, Voice over IP, протоколы туннелирования IP и многие онлайн-игры.

- UDP-приложения используют датаграммные сокеты для установки соединения между хостами. Приложение связывает сокет с его конечной точкой передачи данных, которая является комбинацией IP-адреса и порта службы. Порт — это программная структура, определяемая номером порта — 16-битным целочисленным значением (то есть от 0 до 65535). Порт 0 зарезервирован, хотя и является допустимым значением порта источника в случае, если процесс-отправитель не ожидает ответных сообщений.

- IANA разбила номера портов на три группы.
- Порты с номерами от 0 до 1023 используются для обычных, хорошо известных служб. В Unix-подобных операционных системахдля использования таких портов необходимо разрешение суперпользователя.
- Порты с номерами от 1024 до 49151 предназначены для зарегистрированных IANA служб.
- Порты с 49152 по 65535 — динамические и могут быть использованы для любых целей, поскольку официально не разработаны для какой-то определённой службы. Они также используются как эфемерные (временные) порты, на которых запущенное на хосте программное обеспечение может случайным образом выбрать порт для самоопределения. По сути, они используются как временные порты в основном клиентами при связи с серверами.

Структура пакета

UDP — минимальный ориентированный на обработку сообщений протокол транспортного уровня, задокументированный в RFC 768.

UDP не предоставляет никаких гарантий доставки сообщения для протокола верхнего уровня и не сохраняет состояния отправленных сообщений. По этой причине UDP иногда называют Unreliable Datagram Protocol (англ. — Ненадёжный протокол датаграмм).

UDP обеспечивает многоканальную передачу (с помощью номеров портов) и проверку целостности (с помощью контрольных сумм) заголовка и существенных данных. Надёжная передача в случае необходимости должна реализовываться пользовательским приложением.

Биты	0 - 15	16 - 31
0-31	Порт отправителя (Source port)	Порт получателя (Destination port)
32-63	Длина датаграммы (Length)	Контрольная сумма (Checksum)
64-...	Данные (Data)	

- **Порт отправителя**
- В этом поле указывается номер порта отправителя. Предполагается, что это значение задаёт порт, на который при необходимости будет посылаться ответ. В противном же случае, значение должно быть равным 0. Если хостом-источником является клиент, то номер порта будет, скорее всего, эфемерным. Если источником является сервер, то его порт будет одним из «хорошо известных».
- **Порт получателя**
- Это поле обязательно и содержит порт получателя. Аналогично порту отправителя, если клиент — хост-получатель, то номер порта эфемерный, иначе (сервер — получатель) это «хорошо известный порт».
- **Длина датаграммы**
- Поле, задающее длину всей датаграммы (заголовка и данных) в байтах. Минимальная длина равна длине заголовка — 8 байт. Теоретически, максимальный размер поля — 65535 байт для UDP-датаграммы (8 байт на заголовок и 65527 на данные). Фактический предел для длины данных при использовании IPv4 — 65507 (помимо 8 байт на UDP-заголовок требуется ещё 20 на IP-заголовок).
- В Jumbogram-ах IPv6 пакеты UDP могут иметь больший размер. Максимальное значение составляет 4 294 967 295 байт (2^32 — 1), из которых 8 байт соответствуют заголовку, а остальные 4 294 967 287 байт — данным.

- **Контрольная сумма**
- Поле контрольной суммы используется для проверки заголовка и данных на ошибки. Если сумма не сгенерирована передатчиком, то поле заполняется нулями. Поле не является обязательным для IPv4.
- **Расчёт контрольной суммы**
- Метод для вычисления контрольной суммы определён в RFC 768.
- Перед расчётом контрольной суммы UDP-сообщение дополняется в конце нулевыми битами до длины, кратной 16 битам (псевдозаголовок и добавочные нулевые биты не отправляются вместе с сообщением). Поле контрольной суммы в UDP-заголовке во время расчёта контрольной суммы *отправляемого* сообщения принимается нулевым.
- Для расчёта контрольной суммы псевдозаголовок и UDP-сообщение разбивается на слова (1 слово = 2 байта (октета) = 16 бит). Затем рассчитывается поразрядное дополнение до единицы суммы всех слов с поразрядным дополнением. Результат записывается в соответствующее поле в UDP-заголовке.
- Нулевое значение контрольной суммы зарезервировано и означает, что датаграмма не имеет контрольной суммы. В случае, если вычисленная контрольная сумма получилась равной нулю, поле заполняют двоичными единицами.
- При получении сообщения получатель считает контрольную сумму заново (уже учитывая поле контрольной суммы), и, если в результате получится двоичное число из шестнадцати единиц (то есть 0xffff), то контрольная сумма считается сошедшейся. Если сумма не сходится (данные были повреждены при передаче), датаграмма уничтожается.

- **Пример расчёта контрольной суммы**
- Для примера рассчитаем контрольную сумму нескольких 16-битных слов: 0x398a, 0xf802, 0x14b2, 0xc281. Находим их сумму с поразрядным дополнением. 0x398a + 0xf802 = 0x1318c → 0x318d
  0x318d + 0x14b2 = 0x0463f → 0x463f
  0x463f + 0xc281 = 0x108c0 → 0x08c1
  Теперь находим поразрядное дополнение до единицы полученного результата:
- 0x08c1 = 0000 1000 1100 0001 → 1111 0111 0011 1110 = 0xf73e или, иначе — 0xffff − 0x08c1 = 0xf73e. Это и есть искомая контрольная сумма.
- Различие между IPv4 и IPv6 в данных, используемых для вычисления контрольной суммы.

Псевдозаголовок для IPv4

Если UDP работает над IPv4, контрольная сумма вычисляется при помощи псевдозаголовка, который содержит некоторую информацию из заголовка IPv4. Псевдозаголовок не является настоящим IPv4-заголовком, используемым для отправления IP-пакета. В таблице приведён псевдозаголовок, используемый только для вычисления контрольной суммы.

Биты	0 — 7	8 — 15	16 — 23	24 — 31
0	Адрес источника			
32	Адрес получателя			
64	Нули	Протокол	Длина UDP	
96	Порт источника		Порт получателя	
128	Длина		Контрольная сумма	
160+	Данные			

Адреса источника и получателя берутся из IPv4-заголовка. Значения поля «Протокол» для UDP равно 17 (0x11). Поле «Длина UDP» соответствует длине заголовка и данных.

Вычисление контрольной суммы для IPv4 необязательно, если она не используется, то значение равно 0.

- **Псевдозаголовок для IPv6**
  При работе UDP над IPv6 контрольная сумма обязательна. Метод для её вычисления был опубликован в RFC 2460:
  При вычислении контрольной суммы опять используется псевдозаголовок, имитирующий реальный IPv6-заголовок:

Биты	0 — 7	8 — 15	16 — 23	24 — 31
0	Адрес источника			
32				
64				
96				
128	Адрес получателя			
160				
192				
224				
256	Длина UDP			
288	Нули			Следующий заголовок
320	Порт источника		Порт получателя	
352	Длина		Контрольная сумма	
384+	данные			

Адрес источника такой же, как и в IPv6-заголовке. Адрес получателя — финальный получатель; если в IPv6-пакете не содержится заголовка маршрутизации (Routing), то это будет адрес получателя из IPv6-заголовка, в противном случае, на начальном узле, это будет адрес последнего элемента заголовка маршрутизации, а на узле-получателе — адрес получателя из IPv6-заголовка. Значение «Следующий заголовок» равно значению протокола — 17 для UDP. Длина UDP — длина UDP-заголовка и данных.

- Надёжность и решения проблемы перегрузок
- Из-за недостатка надёжности приложения UDP должны быть готовы к некоторым потерям, ошибкам и дублированиям. Некоторые из них (например, TFTP) могут при необходимости добавить элементарные механизмы обеспечения надёжности на прикладном уровне.
- Но чаще такие механизмы не используются UDP-приложениями и даже мешают им. Потоковые медиа, многопользовательские игры в реальном времени и VoIP — примеры приложений, часто использующих протокол UDP. В этих конкретных приложениях потеря пакетов обычно не является большой проблемой. Если приложению необходим высокий уровень надёжности, то можно использовать другой протокол (TCP) или erasure codes.

- Более серьёзной потенциальной проблемой является то, что в отличие от TCP, основанные на UDP приложения не обязательно имеют хорошие механизмы контроля и избежания перегрузок. Чувствительные к перегрузкам UDP-приложения, которые потребляют значительную часть доступной пропускной способности, могут поставить под угрозу стабильность в Интернете.
- Сетевые механизмы были предназначены для того, чтобы свести к минимуму возможные эффекты от перегрузок при неконтролируемых, высокоскоростных нагрузках. Такие сетевые элементы, как маршрутизаторы, использующие пакетные очереди и техники сброса, часто являются единственным доступным инструментом для замедления избыточного UDP-трафика. DCCP (англ. Datagram Congestion Control Protocol — протокол контроля за перегрузками датаграмм) разработан как частичное решение этой потенциальной проблемы с помощью добавления конечному хосту механизмов для отслеживания перегрузок для высокоскоростных UDP-потоков вроде потоковых медиа.

- Многочисленные ключевые Интернет-приложения используют UDP, в их числе — DNS (где запросы должны быть быстрыми и состоять только из одного запроса, за которым следует один пакет ответа), Простой Протокол Управления Сетями (SNMP), Протокол Маршрутной Информации (RIP), Протокол Динамической Конфигурации Узла (DHCP).
- Голосовой и видеотрафик обычно передается с помощью UDP. Протоколы потокового видео в реальном времени и аудио разработаны для обработки случайных потерь пакетов так, что качество лишь незначительно уменьшается вместо больших задержек при повторной передаче потерянных пакетов. Поскольку и TCP, и UDP работают с одной и той же сетью, многие компании замечают, что недавнее увеличение UDP-трафика из-за этих приложений реального времени мешает производительности TCP-приложений вроде систем баз данных или бухгалтерского учета. Так как и бизнес-приложения, и приложения в реальном времени важны для компаний, развитие качества решений проблемы некоторыми рассматривается в качестве важнейшего приоритета.

SCTP (англ. *Stream Control Transmission Protocol* — «протокол передачи с управлением потоком»)

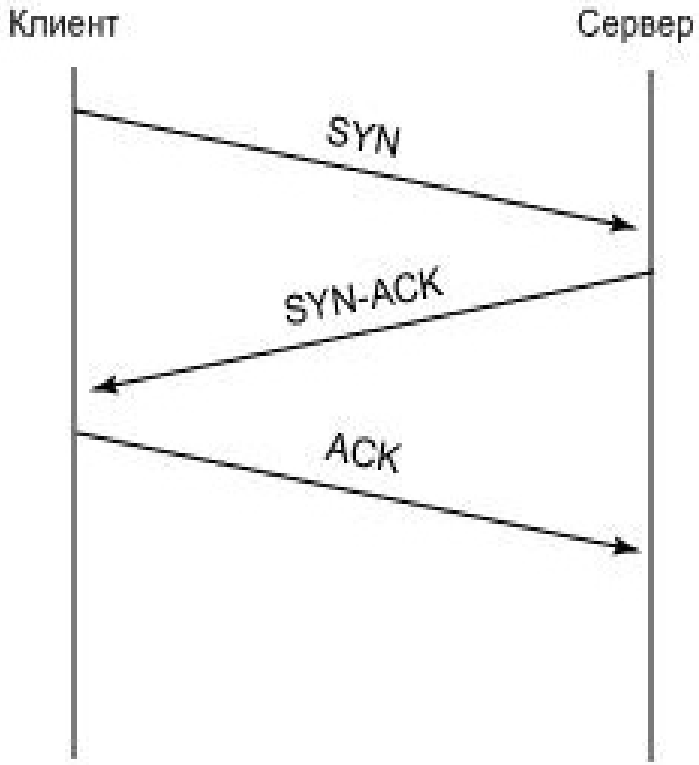
- протокол транспортного уровня в компьютерных сетях, появившийся в 2000 году в IETF. RFC 4960 описывает этот протокол, а RFC 3286 содержит техническое вступление к нему.
- Как и любой другой протокол передачи данных транспортного уровня, SCTP работает аналогично TCP или UDP. Будучи более новым протоколом, SCTP имеет несколько нововведений, таких как многопоточность, защита от SYN-flood атак, синхронное соединение между двумя хостами по двум и более независимым физическим каналам (multi-homing).

Безопасное установление соединения

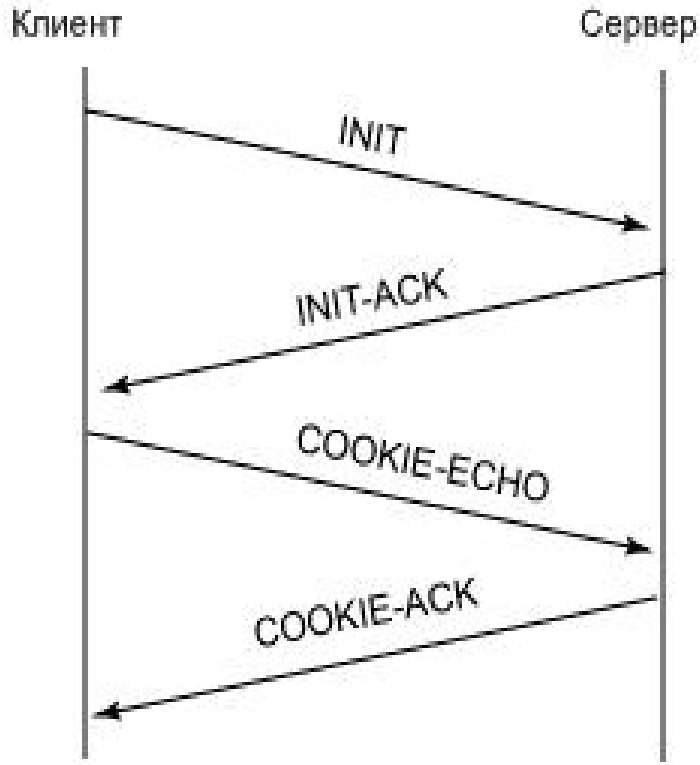
- Создание нового подключения в протоколах TCP и SCTP происходит при помощи механизма подтверждения (квитирования) пакетов. В протоколе TCP данная процедура получила название трехэтапное квитирование (three-way handshake). Клиент посылает пакет SYN (сокр. Synchronize). Сервер отвечает пакетом SYN-ACK (Synchronize-Acknowledge). Клиент подтверждает прием пакета SYN-ACK пакетом ACK. На этом процедура установления соединения завершается.
- Протокол TCP имеет потенциальную уязвимость, обусловленную тем, что нарушитель, установив фальшивый IP-адрес отправителя, может послать серверу множество пакетов SYN. При получении пакета SYN сервер выделяет часть своих ресурсов для установления нового соединения. Обработка множества пакетов SYN рано или поздно, затребует все ресурсы сервера и сделает невозможным обработку новых запросов. Такой вид атак называется «отказ в обслуживании» (Denial of Service (DoS)).

Четырехэтапное квитирование

- Протокол SCTP защищен от подобных атак с помощью механизма четырехэтапного квитирования (four-way handshake) и вводом маркера (cookie). По протоколу SCTP клиент начинает процедуру установления соединения посылкой пакета INIT. В ответ сервер посылает пакет INIT-ACK, который содержит маркер (уникальный ключ, идентифицирующий новое соединение). Затем клиент отвечает посылкой пакета COOKIE-ECHO, в котором содержится маркер, посланный сервером. Только после этого сервер выделяет свои ресурсы новому подключению и подтверждает это отправлением клиенту пакета COOKIE-ACK.
-
- Для решения проблемы задержки пересылки данных при выполнении процедуры четырехэтапного квитирования в протоколе SCTP допускается включение данных в пакеты COOKIE-ECHO и COOKIE-ACK.



Процедура установления соединения (трехэтапное квитирование) в протоколе TCP



Процедура установления соединения (четырехэтапное квитирование) в протоколе SCTP

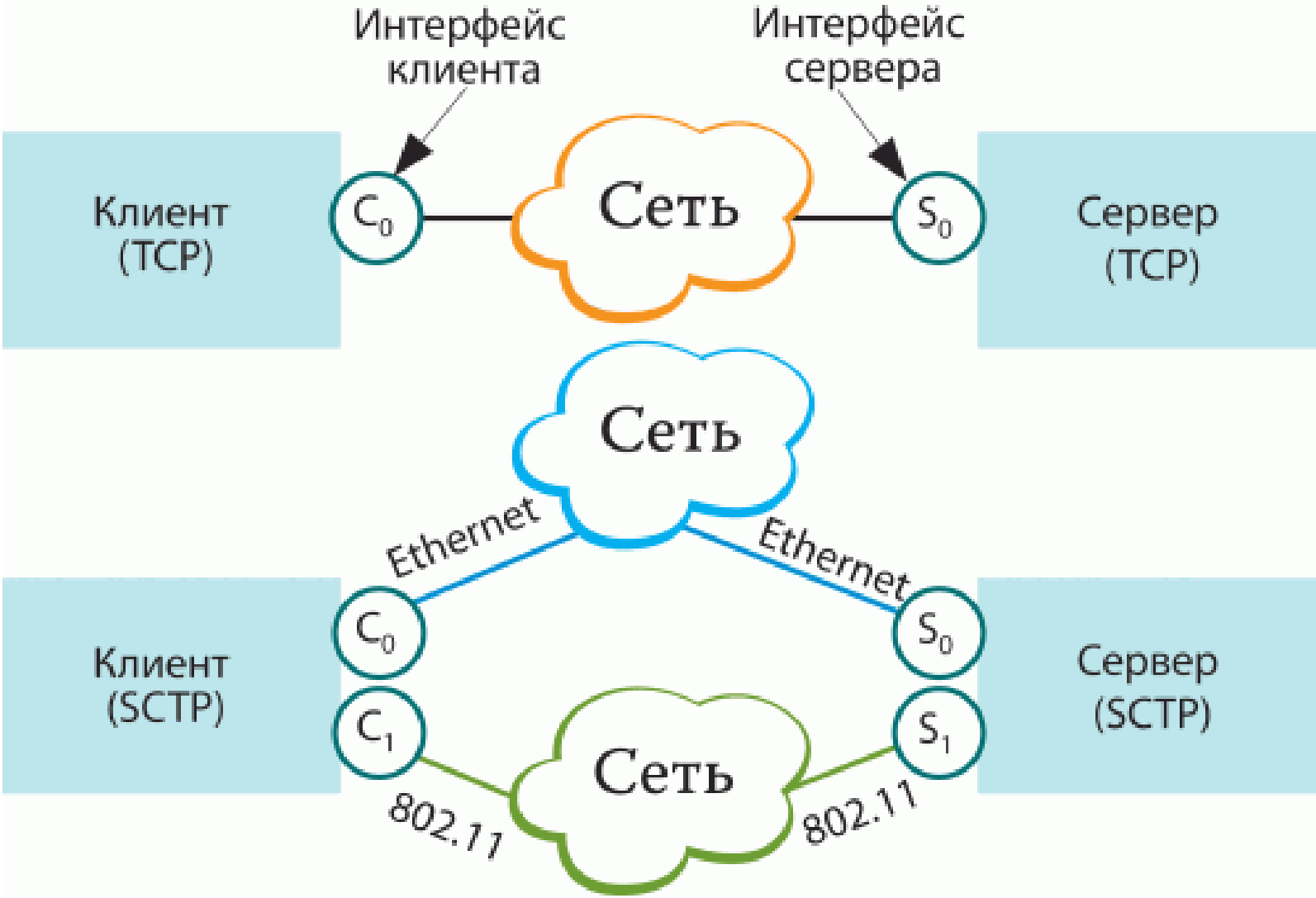
Поэтапное завершение передачи данных

- Рассмотрим отличия между процедурой закрытия сокетов протокола SCTP и процедурой частичного закрытия (half-close) протокола TCP.
- В протоколе TCP возможна ситуация частичного закрытия соединения, когда один узел закончил передачу данных (выполнив посылку пакета FIN), но продолжает принимать данные по этому соединению. Другой узел может продолжать передавать данные до тех пор, пока сам не проведёт закрытие соединения на своей стороне. Состояние частичного закрытия используется приложениями крайне редко, поэтому разработчики протокола SCTP посчитали нужным заменить его последовательностью сообщений для разрыва существующей ассоциации. Когда узел закрывает свой сокет (посылает сообщение SHUTDOWN), оба корреспондента должны прекратить передачу данных, при этом разрешается лишь обмен пакетами, подтверждающими прием ранее отправленных данных.

Многопоточность

- TCP управляет последовательностью байт: данные, посланные приложением-отправителем, должны поступать приложению-получателю строго в том же порядке (в то время как протокол IP способен поменять последовательность пакетов; кроме того, пропавшие пакеты посылаются повторно и обычно прибывают к получателю с нарушением последовательности; для борьбы с этими явлениями данные накапливаются в буфере). SCTP может транспортировать данные между двумя точками одновременно по нескольким потокам *сообщений*. В противоположность к TCP, SCTP обрабатывает целые *сообщения*, а не обычные байтыинформации. Это означает, что если отправитель отсылает серверу *сообщение*, состоящее из 100 байт за первый шаг, а за ним ещё 50 байт, то получатель за первый шаг получит именно первые 100 байт в первом сообщении, а только затем и только 50 байт на второй операции чтения из сокета.

- Термин «многопоточность» (англ. *multi-streaming*) обозначает способность SCTP параллельно передавать по нескольким независимым потокам *сообщений*. Например, мы передаем несколько фотографий через HTTP-приложение (например, браузер). Можно использовать для этого связку из нескольких TCP-соединений, однако также допустимо *SCTP-ассоциация* (англ. *SCTP-association*), управляющее несколькими потоками сообщений для этой цели.
- TCP достигает правильного порядка байт в потоке, абстрактно назначая порядковый номер каждой отосланной единице, а упорядочивая принятые байты, используя назначенные порядковые номера, по мере их прибывания. С другой стороны, SCTP присваивает различные порядковые номера сообщениям, посылаемым в конкретном потоке. Это разрешает независимое упорядочивание сообщений по разным потокам. Так или иначе, многопоточность является опцией в SCTP. В зависимости от желаний пользовательского приложения, сообщения могут быть обработаны не в порядке их отправления, а в порядке их поступления.


Интерфейс клиента
Интерфейс сервера
Клиент (TCP)
C_0
Сеть
S_0
Сервер (TCP)
Сеть
Ethernet
Ethernet
Клиент (SCTP)
C_0
C_1
S_0
S_1
Сеть
802.11
802.11
Сервер (SCTP)

Достоинства

- Использование множественных интерфейсов (англ *Multihoming*)
  Допустим, у нас есть два хоста. И хотя бы один из них имеет несколько сетевых интерфейсов, и соответственно несколько IP-адресов. В TCP, понятие «соединение» означает обмен данными между двумя точками, в то время, как в SCTP имеет место концепция «ассоциации» (англ. *association*), обозначающая всё происходящее между двумя хостами

- **Причины появления**
- Протокол TCP предоставляет основные средства для передачи данных по сети Internet по надежному пути. Однако TCP накладывает некоторые ограничения на транспорт данных:
- TCP предоставляет надежную передачу данных в строгой последовательности. Тем не менее одни приложения требуют передачу без управления и контроля последовательности, а другие будут вполне удовлетворены частичной упорядоченностью данных. Оба этих случая страдают из-за ненужных задержек, связанных с восстановлением и упорядочиванием нарушенных последовательностей TCP.
- Природа TCP ориентирована на поток байт, что вызывает неудобства. Приложения вынуждены самостоятельно добавлять собственные маркеры в пакеты, чтобы распараллелить передачу собственных сообщений, а также использовать дополнительные ухищрения, чтобы убедиться в том, что целое сообщение было доставлено за определенное время.
- Ограниченные рамки возможностей TCP-сокетов ещё более усложняют задачу предоставления возможности параллельной передачи информации к хостам по нескольким каналам связи (см. *multi-homing* выше).
- TCP относительно уязвим к атакам класса «Отказ в обслуживании» (DoS), таким как SYN-flood.

Безопасность

- SCTP был разработан с некоторыми функциями позволяющими повысить безопасность, такими как «4-х кратное рукопожатие» (по сравнению с «трёхкратным рукопожатием» в TCP), чтобы предотвратить SYN-flood атаки, и больших Cookie для проверки подлинности ассоциации.
- Надежность была одним из ключевых аспектов разработки безопасности протокола SCTP. Multi-homing позволяет ассоциации оставаться открытой, даже если некоторые используемые маршруты и интерфейсы стали недоступны. Это имеет особое значение для SIGTRAN, который используя SCTP, передаёт сообщения и сервисы протоколов ОКС-7 поверх IP сети, что требует сильной устойчивости во время отключений линков для поддержания телекоммуникационных услуг, даже при серьёзных аномалиях в сети.
- Шифрование не является частью оригинального дизайна SCTP.

- **Формирование кадров сообщения**
- При формировании кадров сообщения обеспечивается сохранение границ сообщения в том виде, в котором оно передается сокету; это означает, что если клиент посылает серверу 100 байт, за которыми следуют 50 байт, то сервер воспринимает 100 байт и 50 байт за две операции чтения. Точно так же функционирует протокол UDP, это является особенностью протоколов, ориентированных на работу с сообщениями.
- В противоположность им протокол TCP обрабатывает неструктурированный поток байт. Если не использовать процедуру формирования кадров сообщения, то узел сети может получать данные по размеру больше или меньше отправленных. Такой режим функционирования требует, чтобы для протоколов, ориентированных на работу с сообщениями и функционирующих поверх протокола TCP, на прикладном уровне был предоставлен специальный буфер данных и выполнялась процедура формирования кадров сообщений (что потенциально является сложной задачей).
- Протокол SCTP обеспечивает формирование кадров при передаче данных. Когда узел выполняет запись в сокет, его корреспондент с гарантией получает блок данных того же размера.

- Каждый пакет состоит из двух основных разделов:
- Общий заголовок, который занимает первые 12 байт.
- Блоки данных, которые занимают оставшуюся часть пакета.
- Каждый блок имеет идентификатор типа, занимающий один байт. Таким образом, возможно определение не более 255 различных типов блоков. RFC 4960 определяет список типов блоков, всего на данный момент определено 15 типов. Остальная часть блока состоит из поля длины размером в 2 байта (максимальная длина, которя может содержаться в данном поле, равна 65535 байтам) и, собственно, данных. Если размер блока не кратен 4-м байтам, то он заполняется нулями до размера, кратного 4-м байтам.

Биты	**Биты 0-7**	**8-15**	**16-23**	**24-31**
+0	Порт источника		Порт назначения	
32	Тег проверки			
64	Контрольная сумма			
96	Тип 1 блока	Флаги 1 блока	Длина 1 блока	
128	Данные 1 блока			
…	…			
…	Тип N блока	Флаги N блока	Длина N блока	
…	Данные N блока			


Порт отправителя
Порт получателя
Тег проверки
Контрольная сумма
Фрагмент 1
Фрагмент 2
Фрагмент 3
Тип
Флаг
Длина
Номер транспортной последовательности (TSN)
Идентификатор потока
Номер последовательности потока (SSN)
Идентификатор протокола
Данные пользователя

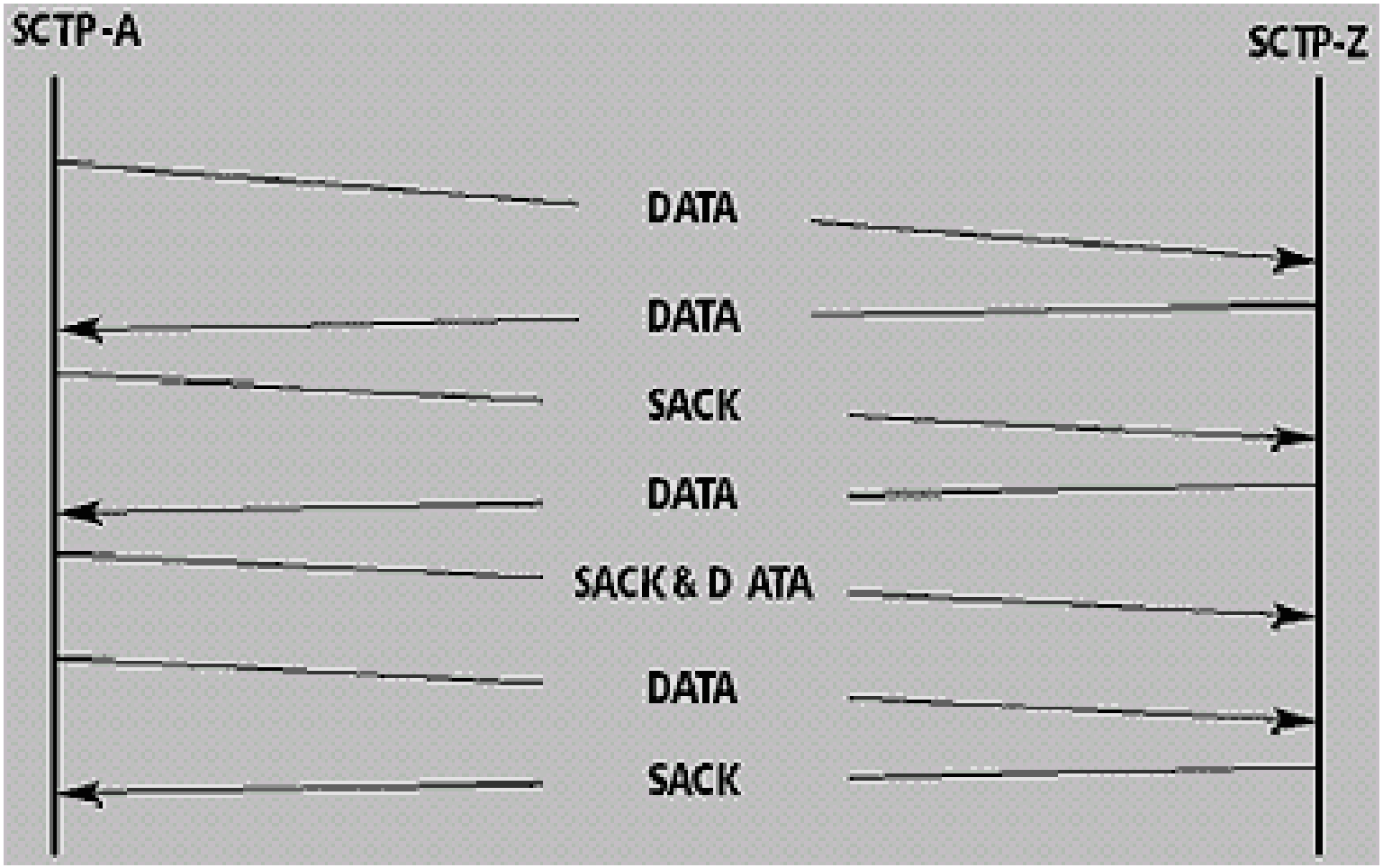
- Еще один связанный с SCTP аспект, который может вызвать путаницу, — это различие между надежной и упорядоченной доставкой. При работе с TCP эти два аспекта неразрывно связаны, поскольку все данные надежно доставляются (к примеру, утерянные пакеты передаются повторно) хосту-получателю и предоставляются приложению в той последовательности, в какой они передавались. Для этого TCP использует номер последовательности в заголовке каждого пакета.

- SCTP разделяет эти два аспекта на независимые функции. Номер последовательности в передаче в заголовке SCTP гарантирует, что все сообщения надежно доставляются на хост-получатель, но SCTP предусматривает ряд вариантов того, в каком порядке представлять сообщения приложению-получателю. Это может быть номер потока в последовательности в пакете SCTP, применяемых для упорядочивания сообщений по потокам, или передача данных приложению по мере их появления на хосте-получателе. И опять-таки этот подход позволяет устранить задержку, вызванную блокировкой вследствие неправильного порядка доставки пакетов.

- Каждый из типов фрагмента включает в себя информацию заголовка TLV, который содержит тип фрагмента, флаги обработки доставки и длину поля. Кроме того, перед фрагментом DATA будет размещаться пользовательская информация о полезной нагрузке, состоящей из номера транспортной последовательности (TSN — transport sequence number), идентификатора потока, номера последовательности потока (SSN — stream sequence number).

- TSN и SSN — два разных номера последовательности для каждого фрагмента DATA. TSN используется для обеспечения надежности каждой ассоциации, а SSN для упорядочивания по потокам. Идентификатор потока отмечает отдельные сообщения в каждом потоке.

- **Повтор передачи**
- Повторная передача блоков DATA может быть обусловлена (a) тайм-аутом, определяемым таймером повтора (retransmission timer) или (b) получением SACK, показывающих что блок DATA не был получен адресатом. Для снижения вероятности насыщения повтор передачи блоков DATA ограничивается. Значение тайм-аута для повтора (RTO) устанавливается на основе оценки времени кругового обхода и уменьшается экспоненциально с ростом частоты потери сообщений. Для активных ассоциаций с почти постоянным уровнем трафика DATA причиной повтора скорей всего будут сообщения SACK, а не тайм-аут. Для снижения вероятности ненужных повторов используется правило 4 SACK, в соответствии с которым повтор передачи происходит только по четвертому SACK, указывающему на пропуск блока данных. Это позволяет предотвратить повторы передачи, вызванные нарушением порядка доставки.


SCTP-A
SCTP-Z
DATA
DATA
SACK
DATA
SACK & D ATA
DATA
SACK

- Хост SCTP посылает избранные подтверждения (фрагменты SACK) в ответ на каждый пакет SCTP, сопровождающий фрагменты DATA. Сообщение SACK полностью описывает состояние получателя так, что отправитель может принимать решение о повторной передаче в зависимости от того, что ему уже удалось получить. SCTP поддерживает алгоритмы быстрой повторной передачи и повторной передачи с тайм-аутом, аналогичные тем, которые применяются в TCP.

- За небольшим исключением большинство типов фрагментов можно объединить вместе в один пакет SCTP.

Сбой в пути

- Поддерживается счетчик для числа повторов передачи по конкретному адресу получателя без подтверждения успешной доставки.
- Когда значение этого счетчика достигает заданного порога (конфигурационный параметр), адрес объявляется неактивным и протокол SCTP начинает использовать другой адрес для передачи блоков DATA.
- Кроме того, по всем неиспользуемым (дополнительным) адресам периодически передаются специальные блоки Heartbeat и поддерживается счетчик числа блоков Heartbeat, переданных без возврата соответствующего Heartbeat Ack.
- Когда значение счетчика достигает заданного порога (параметр конфигурации), соответствующий адрес объявляется неактивным. Блоки Heartbeat передаются по неактивным адресам до тех пор, пока не будет получено сообщение Ack, говорящее о восстановлении активности адреса.
- Частота передачи блоков Heartbeat определяется значение RTO и дополнительной задержкой, которая позволяет передавать блоки Heartbeat без помех для пользовательского трафика.

- **Отказ в конечной точке**
- Для всех адресов получателя поддерживается общий счетчик числа повторов или блоков Heartbeat, переданных удаленной точке без получения от нее соответствующего подтверждения (Ack). Когда значение счетчика достигает заданного порога (параметр конфигурации) конечная точка декларируется как недостижимая и ассоциация SCTP закрывается.

Протокол **RTP** (англ. *Real-time Transport Protocol*)

- работает на транспортном уровне и используется при передаче трафика реального времени. Протокол был разработан Audio-Video Transport Working Group в IETF и впервые опубликован в 1996 году какRFC 1889, и заменён RFC 3550 в 2003 году.
- Протокол RTP переносит в своём заголовке данные, необходимые для восстановления голоса или видеоизображения в приёмном узле, а также данные о типе кодирования информации (JPEG, MPEG и т. п.). В заголовке данного протокола, в частности, передаются временная метка и номер пакета. Эти параметры позволяют при минимальных задержках определить порядок и момент декодирования каждого пакета, а также интерполировать потерянные пакеты.

- RTP не имеет стандартного зарезервированного номера порта. Единственное ограничение состоит в том, что соединение проходит с использованием чётного номера, а следующий нечётный номер используется для связи по протоколу RTCP. Тот факт, что RTP использует динамически назначаемые адреса портов, создаёт ему трудности для прохождения межсетевых экранов, для обхода этой проблемы, как правило, используется STUN-сервер.
- Установление и разрыв соединения не входит в список возможностей RTP, такие действия выполняются сигнальным протоколом(например, RTSP или SIP протоколом).

- RTP был разработан как протокол реального времени, из конца в конец (end-to-end), для передачи потоковых данных. В протокол заложена возможность компенсации джиттера и детектирования нарушения последовательности пакетов данных — типичных событий при передаче через IP-сети. RTP поддерживает передачу данных для нескольких адресатов через Multicast. RTP рассматривается как основной стандарт для передачи голоса и видео в IP-сетях и совместно с кодеками.
- Приложения, формирующие потоки реального времени, требуют своевременной доставки информации и для достижения этой цели могут допустить некоторую потерю пакетов. Например, потеря пакета в аудио-приложении может привести к доле секунды тишины, которая может быть незаметна при использовании подходящих алгоритмов скрытия ошибок. Протокол TCP, хотя и стандартизирован для передачи RTP, как правило не используется в RTP-приложениях, так как надежность передачи в TCP формирует временные задержки. Вместо этого, большинство реализаций RTP базируется на UDP. Кроме этого, существуют другие спецификации для транспортных протоколов SCTP и DCCP, но они мало распространены.

- Компоненты протокола
- Спецификация RTP описывает два под-протокола:
- Протокол передачи данных, RTP, который взаимодействует с передачей данных реального времени. Информация, предоставляемая посредством этого протокола включает тайм-стемп (для синхронизации), последовательный номер (для детектирования потери и дублирования пакетов) и формат полезной нагрузки, который определяет формат кодирования данных.
- Протокол контроля, RTCP, используемый для определения качества обслуживания (QOS), обратной связи и синхронизации между медиа-потоками. Занимаемая полоса пропускания RTCP — мала в сравнении с RTP, обычно около 5 %.
- Управляющий сигнальный протокол, такой как SIP, H.323, MGCP или H.248. Сигнальные протоколы управляют открытием, модификацией и закрытием RTP-сессий между устройствами и приложениями реального времени.
- Управляющий протокол описания медиа, такой как Session Description Protocol.

- Сессии
- RTP-сессия устанавливается для каждого потока мультимедиа. Сессия состоит из IP-адреса и пары портов для RTP и RTCP. Например, аудио и видео потоки будут иметь различные RTP-сессии, позволяющие приемнику для этого выделить конкретный поток. Порты, которые образуют сессию, связываются друг с другом средствами других протоколов, таких как SIP (содержащий в своих сообщениях протокол SDP) и RTSP (используя SDP в методе Setup). В соответствии со спецификацией, RTP не имеет стандартного зарезервированного номера порта. Единственное ограничение состоит в том, что соединение проходит с использованием чётного номера, а следующий нечётный номер используется для связи по протоколу RTCP. RTP и RTCP обычно используют непривилегированные UDP-порты (16k-32k), но могут использовать и другие протоколы, поскольку сам протокол RTP независим от транспортного уровня.

DCCP (англ. *Datagram Congestion Control Protocol*)

- протокол транспортного уровнямодели OSI, разрабатываемый IETF. Принят в качестве стандарта в марте 2006 года. Он предоставляет механизмы для отслеживания перегрузок в сети, избегая возможности использования механизмов прикладного уровня. Этот протокол не гарантирует доставку информации в нужном порядке.
- DCCP очень эффективен для приложений, в которых данные, пришедшие не вовремя, становятся бесполезными. Например: потоковое медиа-вещание, онлайн игры и интернет-телефония. Главная особенность этих приложений состоит в том, что старые сообщения очень быстро становятся бесполезными, поэтому лучше получить новое сообщение, чем пытаться переслать старое. Но на данный момент большинство таких приложений самостоятельно реализуют отслеживание перегрузок, а в качестве протоколов передачи используются TCP или UDP.
- Протокол DCCP доступен в ядре Linux с версии 2.6.14 и улучшается с каждым выпуском.

Протокол **RDP** (англ. *Reliable Data Protocol*)

- разработан для обеспечения надежной передачи данных между пакетно-ориентированными приложениями. Изначально он был разработан для приложений, реализующих удаленную загрузку данных и удаленное устранение неполадок, однако его можно использовать и в других приложениях, требующих надежной передачи сообщений. Существуют две версии RDP, описанные в спецификациях RFC-908 и RFC-1151 соответственно.

Протокол **RUDP** (англ. *Reliable User Datagram Protocol*)

- основанный на протоколе RDP, разработан для передачи телефонных сигналов через IP-сети. Этот протокол не стандартизирован, он не имеет официальной спецификации.
- Протоколы RDP и RUDP используются в тех случаях, когда нельзя использовать UDP из-за его ненадежности, а использование TCP влечет за собой слишком высокую сложность процесса передачи данных.
- В отличие от UDP, RDP и RUDP поддерживают следующие функции:
- подтверждение доставки пакетов
- повторная отправка потерянных пакетов
- управление потоками передачи данных
- RDP обеспечивает прикладной уровень надежной службой передачи сообщений. Интерфейс протокола преобразует данные пользователя в сообщения. Сообщения, в свою очередь, в ходе обмена данными между RDP и IP преобразуются в сегменты данных и далее в дейтаграммы.