ПРОТОКОЛЫ ТРАНСПОРТНОГО УРОВНЯ UDP, TCP И SCTP: достоинства и недостатки

А.В.Лейкин, ст. преподаватель СПбГУТ

В статье рассматриваются три основных протокола транспортного уровня: UDP, TCP и SCTP, их преимущества и недостатки. Анализируется логика работы протоколов и ситуации, при которых предпочтительно использовать тот или иной протокол. Также приведено соответствие стека TCP/IP модели OSI и примеры приложений, использующих данные протоколы.

В течение почти 20 лет пользователи, имевшие дело со стеком TCP/IP, работали с одним из двух транспортных протоколов: TCP и UDP. Однако наступило время, когда длянекоторые приложений потребовалась функциональность, выходящую за рамки той, которую в состоянии предоставить эти два протокола. В 2000 году организация IETF (Internet Engineering Task Force) одобрила в качестве стандарта протокол передачи с управлением потоком SCTP (Stream Control Transmission Protocol), который стал активно внедряться в сетях не так давно.

Протокол SCTP был создан в рамках проекта, начатого рабочей группой SIGTRAN, и в первую очередь предназначался для транспортировки сигнальной информации ОКС-7 по IP-сетям. Строгие требования ОКС-7 к параметрам потерь и соблюдению очередности следования сообщений привели к созданию нового транспортного протокола, свободного от недостатков UDP и TCP [1]. IETF предлагает использовать его в качестве протокола транспортного уровня общего назначения, объединяющего функции TCP и UDP над уровнем IP, поскольку и другие приложения могут использовать некоторые возможности этого протокола.

Как уже было отмечено, все три протокола работают на транспортном уровне эталонной модели взаимодействия открытых систем ВОС (OSI), основная задача которого – реализация сквозной связи между узлами сети, а также при необходимости управление потоком и предотвращение перегрузок сети. Транспортный уровень компенсирует ненадежность, присущую нижним уровням, за счет обработки ошибок, которые вызваны искажением данных, потерей пакетов и их доставкой не по порядку. Протоколы UDP, TCP и SCTP входят в стек TCP/IP, где также работают на транспортном уровне. Соответствие стека модели ВОС показано на рис.1.

Протокол UDP

Протокол дейтаграмм пользователя UDP (User Datagram Protocol) был описан в документе RFC 768 и принят IETF в 1980 году [2]. Он не ориентирован на создание соединения, его главное отличие – отсутствие гарантии доставки и поддержки упорядоченности передаваемых сообщений до места назначения, порядок которых

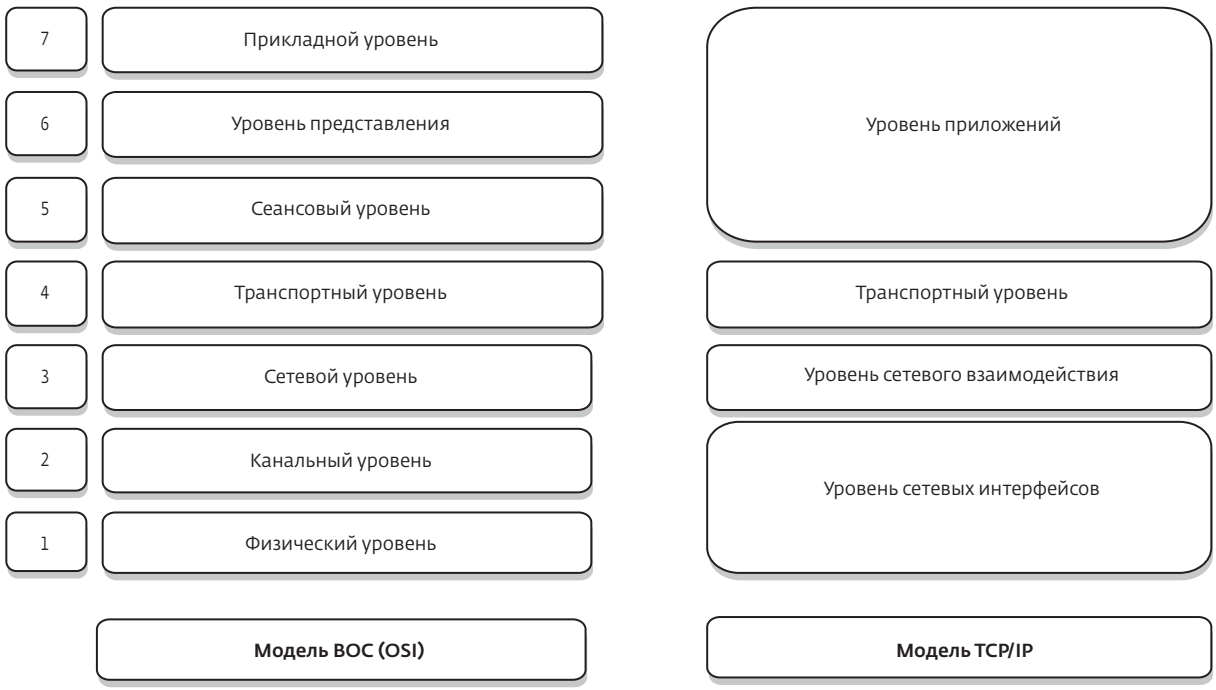

Рис.1. Соответствие стека TCP/IP модели ВОС (OSI)

может быть изменен из-за особенностей работы и капризов IP-сети. Эти отличия – следствия логики работы протокола. Приложение, использующее UDP, формирует один пакет, который передается в IP-дейтаграмме. Если дейтаграмма дублируется в сети, то на принимающий узел могут быть доставлены два ее экземпляра. Если же клиент UDP отправляет две дейтаграммы в одно и то же место назначения, то их порядок может быть изменен сетью, и они будут доставлены с нарушением исходного порядка. Поэтому в приложениях, использующих UDP, разработчики должны реализовывать функции, в некоторой степени компенсирующие ненадежность этого протокола: тайм-ауты, повторную передачу, обработку потерянных дейтаграмм и порядковые номера для сопоставления ответов запросам. Этот подход позволяет протоколу гораздо быстрее и эффективнее доставлять данные для приложений, которым требуется большая пропускная способность сети связи или малое время доставки данных. Но из-за отсутствия контроля перегрузок, возрастающего объема неконтролируемой высокоскоростной нагрузки и использования общей сетевой инфраструктуры с другими сервисами создается реальная опасность перегрузки сети, которая ведет к значительному падению ее производительности. Частично данную проблему призван решить относительно молодой протокол DCCP (Datagram Congestion Control Protocol), описанный в RFC 4340. Этот протокол – альтернатива UDP для приложений, которым необходим сервис одноадресной негарантированной доставки дейтаграмм, высокая скорость работы и реализованные на транспортном уровне механизмы для отслеживания перегрузок в сети без необходимости создавать их на уровне приложений. К сожалению, объем статьи не позволяет остановиться на нем более подробно.

Протокол TCP

Протокол управления передачей TCP (Transmission Control Protocol), разработанный по заказу агентства перспективных исследовательских программ ARPA, был опубликован

в документе RFC 793 [3] в сентябре 1981 года, а уже в 1983 полностью заменил собой протокол NCP (Network Control Protocol) в сети ARPANET (прародителе Интернета) и сегодня стал основным протоколом для передачи данных.

Протокол TCP перед началом передачи данных в обязательном порядке устанавливает соединение и обеспечивает использующим его приложениям надежный упорядоченный двухсторонний байтовый поток. Он поддерживает отправку и прием подтверждений, обработку тайм-аутов, повторную передачу, управление потоком и прочие возможности, которые описаны в ряде документов (RFC 1323, 2581, 2988, 3390 и 5681).

Протокол TCP предлагает приложениям сервис надежной и упорядоченной передачи. Все отправленные данные подлежат обязательному подтверждению встречной стороной, причем формируются подтверждения не для каждого конкретного успешно полученного пакета, а для всех данных от начала посылки до некоторого порядкового номера. Если подтверждение не приходит в течение времени RTO (Retransmission Time Out), то протокол TCP автоматически передает данные повторно и перезапускает таймер вновь. Величина таймера RTO динамически меняется и зависит от времени двухсторонней задержки, определяемой с помощью специальных алгоритмов, типа сети и конкретной реализации протокола. Суммарное время повторных попыток отправки данных в среднем может занимать до 10 минут. Разумеется, TCP не может гарантировать получение данных адресатом, поскольку это в принципе невозможно. Если осуществить доставку невозможно, TCP уведомляет об этом пользователя, прекращает попытки повторной передачи и разрывает соединение. TCP можно условно считать протоколом, надежным на все 100%: он обеспечивает доставку данных или же уведомление о неудаче.

Упорядочивание данных осуществляется привязкой некоторого порядкового номера к каждому отправляемому байту. Предположим, что приложение записывает 2048 байт в сокет TCP, что приводит к отправке двух сегментов: первый из них содержит данные с порядковыми номерами 1–1024, второй – с номерами 1025–2048. Если какой-либо сегмент приходит вне очереди, то принимающий TCP перед отправкой данных приложению заново упорядочит сегменты, используя их порядковые номера. Если TCP получает дублированные данные, то он может их определить и отбросить [6].

Для обеспечения процедуры управления потоком (Flow Control) TCP всегда сообщает встречному узлу, какое буферное пространство он выделил для приема данных, и отправляющий узел не может превышать этого ограничения. Эта процедура получила название "метод скользящего окна". В любой момент времени окно соответствует свободному пространству в буфере получателя. Данный метод гарантирует, что отправитель не переполнит этот буфер, а также позволяет оптимизировать и ускорить процесс передачи больших объемов данных. Окно изменяется динамически по мере считывания принимающим приложением данных из буфера, а значение размера передается отправителю вместе с сообщением о подтверждении. Если принимающий буфер TCP заполнен, то возможна ситуация, при которой размер окна станет нулевым. В этом случае отправитель вынужден ждать, пока получатель не считает данные из буфера. При необходимости данные можно "протолкнуть", используя функцию PUSH, которая запускается установкой в сообщении флага PSH. Тогда все данные с таким флагом и данные в буфере получателя будут переданы принимающему приложению.

Приложение, использующее TCP, имеет возможность отправлять и принимать данные в обоих направлениях на заданном соединении в любой момент времени. Иначе говоря, TCP может работать в полнодуплексном режиме и должен отслеживать порядковые номера и размеры окна для каждого направления потока данных. После установления двухстороннего соединения оно может быть преобразовано в одностороннее.

Протокол TCP, поддерживающийся практически всеми приложениями Интернета, за прошедшие годы был значительно усовершенствован для обеспечения надежности и производительности в сетях различной емкости и качества. Тем не менее, в нем сохранились свойства, которые делают его неподходящим для таких задач, как передача сигнальных сообщений в VoIP-сетях или асинхронная обработка на базе транзакций [3]. TCP требует наличия службы доставки со строго упорядоченной передачей для всех данных, пересылаемых между двумя хостами. Это слишком серьезное ограничение для приложений, которые допускают как последовательную (частичное упорядочивание), так и непоследовательную доставку сообщений. TCP трактует каждую передачу данных

как неструктурированную последовательность байтов и не хранит никаких неявных структур в передаваемых потоках данных. Приложения, которые обрабатывают отдельные сообщения, должны добавлять в поток байтов границы сообщений и отслеживать их.

Основанный на механизме TCP-сокетов API-интерфейс не поддерживает множественную адресацию, из-за чего приложение может связать только один IP-адрес другого узла с конкретным TCP-соединением. Если интерфейс, назначенный этому IP-адресу, отключается, то TCP-соединение прерывается, и его необходимо устанавливать заново, что вносит существенные задержки, особенно критичные для приложений реального времени [7].

Когда следует использовать UDP вместо TCP и почему? Мы осознано ставим этот вопрос до рассмотрения протокола SCTP, так как задача сравнения протокола UDP с протоколом TCP или SCTP ничем не отличается в виду противопоставления ненадежного протокола протоколам гарантированной доставки.

UDP может использоваться для приложений широковещательной и многоадресной передачи (отправки одного пакета нескольким получателям вместо отправки копий пакета через несколько соединений TCP). Если требуется какая-либо форма защиты от потерь и нарушения порядка следования сообщений, то соответствующая функциональность должна быть добавлена приложениями как на клиентской, так и на серверной стороне. Однако приложения часто используют широковещательную и многоадресную передачу, когда некоторое количество потерь вполне допустимо (например, потеря аудио- или видеопакетов). Имеются приложения, требующие надежной доставки, например пересылка файлов при помощи многоадресной передачи, но в каждом конкретном случае разработчик должен решить, компенсируется ли выигрыш в производительности, получаемый за счет использования многоадресной передачи, дополнительным усложнением приложения для обеспечения надежности соединений.

UDP может использоваться для простых приложений запрос-ответ, так как не требует установки и разрыва соединения и поэтому позволяет осуществить обмен запросом и ответом всего в двух пакетах при условии, что пакет не превышает размер MTU (Maximum Transmission Unit), используемый в данной сети на канальном уровне. Но тогда функция обнаружения ошибок должна быть встроена в приложение. Как минимум это означает включение подтверждений, тайм-аутов и механизма повторных передач. Если запросы и ответы имеют разумный размер, то управление потоком бывает не существенно для обеспечения надежности.

UDP не следует использовать для передачи большого количества данных (например, для передачи файлов). В этом случае управление потоком с помощью окна, предотвращение переполнения и медленный старт должны быть встроены в приложение вместе с перечисленными выше функциями. Эти механизмы, осуществляемые отправителем, служат для определения текущей пропускной способности сети и позволяют контролировать ситуацию при ее перегрузке. Опыт, накопленный еще до того, как эти алгоритмы были реализованы в конце 80-х годов, показывает, что протоколы, не снижающие скорость передачи, усугубляют перегрузку сети.

Протокол SCTP

SCTP создает двустороннюю ассоциацию между двумя конечными точками и дает возможность работы с несколькими потоками каждой паре конечных точек, а также обеспечивает поддержку концепции многоинтерфейсного узла на транспортном уровне. Пример такой архитектуры можно видеть на рис.2.

Изначально SCTP проектировался с учетом потребностей растущего рынка IP-телефонии и предназначался, в частности, для передачи сигнальных сообщений ОКС-7 через Интернет. Сервисы, предоставляемые SCTP, имеют много общего с сервисами TCP и UDP. Протокол SCTP описывается в RFC 4960 [4], а введение в SCTP приводится в RFC 3286. Несмотря на принципиальную разницу между SCTP и TCP, для приложения интерфейс "точка–точка" почти ничем не отличается от интерфейса TCP. Подобно TCP, протокол SCTP обеспечивает приложениям, взаимодействующим по IP-сети, транспортную службу с гарантией доставки и сохранением порядка следования пакетов. Протокол унаследовал многие функции, разработанные для TCP за последние три десятилетия, в том числе возможности контроля перегрузки и восстановления утерянных пакетов. В действительности любое приложение, работающее по протоколу TCP, можно перевести на SCTP без потери функциональности. Рассмотрим основные свойства протокола SCTP.

Подобно TCP, протокол SCTP предоставляет приложениям надежную передачу сообщений, упорядочение данных, управление передачей и двухстороннюю связь. Соединение по протоколу SCTP между клиентом и сервером называется ассоциацией (association), так как это многопотоковый протокол, позволяющий задать несколько IP-адресов и один порт для каждой стороны соединения. Термин "ассоциация" используется вместо слова "соединение" намеренно, потому что соединение всегда устанавливается между двумя IP-адресами, а ассоциация означает взаимодействие двух систем, которые могут иметь по несколько адресов. Каждая из сторон ассоциации в данном контексте называется конечной точкой (endpoint). Наличие у нее нескольких IP-адресов позволяет обеспечить дополнительную устойчивость в случае отказа сети. Избыточные IP-адреса конечной точки могут соответствовать собственному соединению с поставщиком услуг сети Интернет ISP. В такой конфигурации SCTP позволит обойти проблему, возникшую на одном из адресов, благодаря переключению на другой адрес, заранее связанный с данной ассоциацией SCTP. Для сокращения задержек, вызванных переключением с первичного направления на альтернативные, используется механизм контроля работоспособности, который получил название "сердцебиение" (heartbeat). Пока идет передача данных по первичному направлению, протокол SCTP посылает пакеты контроля работоспособности на адреса, находящиеся в режиме ожидания. Протокол декларирует, что IP-адрес будет отключен, как только он достигнет порогового значения невозвращенных подтверждений о работоспособности. Подобной устойчивости можно достичь и в TCP, если воспользоваться протоколами маршрутизации. Например, BGP-соединения внутри домена (iBGP) часто используют адреса, назначаемые виртуальному интерфейсу маршрутизатора в качестве сторон соединения TCP. Протокол маршрутизации домена гарантирует использование любого доступного пути между двумя маршрутизаторами, что невозможно, если используемые адреса принадлежат интерфейсу в сети, где возникли проблемы. Функция множественной адресации SCTP позволяет узлам, а не только маршрутизаторам, использовать аналогичный подход, причем даже с подключениями через разных провайдеров, что невозможно при использовании TCP с маршрутизацией. Выбранный подход делает сервер уязвимым для ситуаций, когда клиент открывает ассоциацию, но никаких данных не передает. Для такого клиента будут выделены ресурсы, которые он не использует, а неудачное стечение обстоятельств может привести к DoS-атаке со стороны неактивных клиентов. Для предотвращения подобных ситуаций в SCTP была добавлена функция автоматического закрытия ассоциаций (autoclose). Она позволяет конечной точке SCTP задавать максимальную длительность бездействия ассоциации, которая считается таковой, если по ней не передаются никакие данные ни в одном направлении. Если длительность бездействия превышает установленное ограничение, ассоциация автоматически закрывается реализацией протокола.

SCTP предоставляет разграничение отдельных записей в передаваемом потоке сообщений.

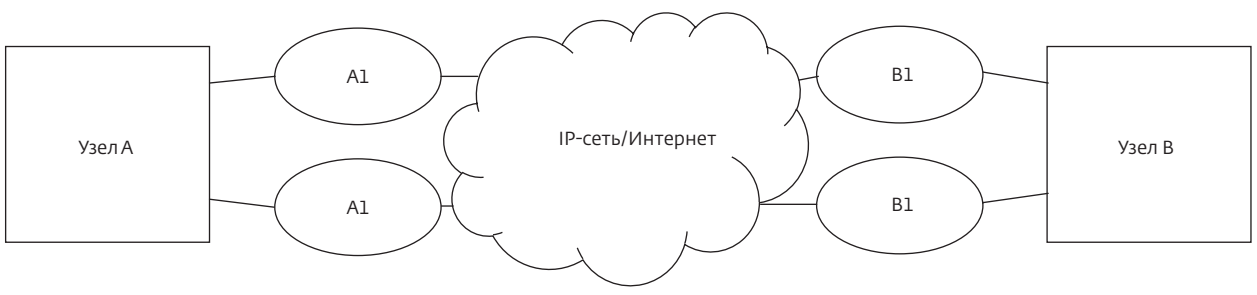

Рис.2. Пример многоинтерфейсной архитектуры

В отличие от TCP, протокол SCTP ориентирован не на поток байтов, а на сообщения. Он обеспечивает упорядоченную доставку данных и сохраняет границы сообщений в пакетах приложения, размещая сообщения в одной или нескольких структурах данных SCTP, называемых "фрагментами" (chunk). Несколько сообщений могут объединяться в один фрагмент, а длинное сообщение может быть сегментировано сразу по нескольким фрагментам.

Обладая возможностью формирования нескольких потоков сообщений между конечными точками ассоциации, протокол SCTP позволяет контролировать для каждого из них надежность и порядок следования сообщений. Благодаря этому свойству устраняется возможная блокировка линии типа head-of-line, присущая протоколу TCP, так как утрата сообщения в одном из потоков не блокирует доставку сообщений по другим. Этот подход прямо противоположен тому, что имеется в TCP, где потеря единственного байта блокирует доставку всех последующих байтов по соединению до тех пор, пока ситуация не будет исправлена.

С помощью SCTP приложения могут использовать различные модели доставки, в том числе строгий порядок передачи (как TCP), частичное упорядочивание (по потокам) и неупорядоченную доставку (как UDP). Это было сделано, поскольку некоторые приложения не нуждаются в сохранении порядка сообщений при передаче их по сети. Раньше приложению, использующему TCP для обеспечения надежности, приходилось мириться с задержками, вызванными блокированием очереди и необходимостью упорядоченной доставки (хотя само приложение в ней может и не нуждаться). На это стоит обратить особое внимание, так как для SCTP существует различие между надежной и упорядоченной доставкой. При работе с TCP эти два свойства неразрывно связаны, поскольку все данные надежно доставляются (например, утерянные пакеты передаются повторно) узлу-получателю и предоставляются приложению в той последовательности, в какой они передавались. Напротив, в SCTP эти свойства между собой не связаны. Номер последовательности в заголовке SCTP гарантирует, что все сообщения надежно доставляются узлу-получателю, но SCTP предусматривает ряд вариантов того, в каком порядке представлять сообщения приложению-получателю. Это может быть номер потока, применяемый для упорядочивания сообщений по потокам, или передача данных приложению по мере их появления на узле-получателе. И опять-таки этот подход позволяет устранить задержку, вызванную блокировкой вследствие неправильного порядка доставки пакетов [6]. Для восстановления утраченных пакетов используется схема выборочного подтверждения, унаследованная из TCP. Поддерживая обратную связь с отправителем, приемник SCTP сообщает, какие пакеты необходимо отсылать повторно, если они были утеряны. Благодаря сервису частичной надежности отправитель получает возможность указывать время жизни каждого сообщения. Если частичная надежность поддерживается обоими узлами, то недоставленные вовремя данные могут сбрасываться транспортным уровнем, а не приложением, даже если они были переданы и утеряны. Таким образом, оптимизируется передача данных в условиях загруженных линий.

Для контроля перегрузки используются стандартные методики, впервые применявшиеся еще в TCP, в том числе медленный старт, предотвращение перегрузки и быстрая повторная передача. Приложения SCTP могут, таким образом, получать свою долю сетевых ресурсов, сосуществуя с приложениями TCP в одной сети.

Когда SCTP оказывается предпочтительнее TCP?

Хотя протокол SCTP и разрабатывался для передачи сигнальных сообщений ОКС-7 через IP-сеть, в процессе разработки область его применения значительно расширилась. Фактически он превратился в общецелевой транспортный протокол. Являясь эволюционным развитием протокола TCP, SCTP поддерживает почти все его функции и значительно расширяет их новыми сервисам транспортного уровня. Эти сервисы имеют важное значение для передачи сообщений телефонной сигнализации через сеть IP и в то же время могут обеспечивать ряд преимуществ для других приложений, которым требуется надежный транспортный механизм с высоким уровнем производительности.

SCTP лишен двух особенностей TCP. Одна из них – состояние неполного закрытия соединения на своей стороне. Это состояние возникает, когда приложение закрывает свой конец соединения, но разрешает встречной

стороне отправлять данные и принимает их. Приложение входит в это состояние для того, чтобы сообщить собеседнику, что отправка данных завершена. Приложения очень редко используют эту возможность, поэтому при разработке SCTP решено было не заботиться о ее поддержке. Также не поддерживается обработка внеочередных данных (urgent data). Для доставки срочных данных в SCTP можно использовать отдельный поток, хотя это и не позволяет в точности воспроизвести поведение TCP в данной ситуации.

Протокол SCTP обеспечивает явную поддержку многоинтерфейсных узлов (см. рис.2), что позволяет значительно повысить уровень надежности ассоциации и уменьшить задержки, возникающие в случае отказа в доступе или сбоев в магистральной сети. Но при этом действующая редакция протокола SCTP [4] не поддерживает распределение нагрузки (load sharing) на альтернативные направления, поэтому данная возможность лишь обеспечивает избыточность направлений передачи для повышения уровня надежности. В IETF сейчас ведутся работы в данном направлении, и документ, имеющий версию 7 и статус экспериментального, был принят в октябре 2013 года [5].

Наконец, TCP-узлы восприимчивы к атакам типа "отказ в обслуживании" DoS (Denial of Service), что вызвано несовершенством механизма установления соединения в TCP, получившего название троекратного рукопожатия (three-way handshake). Для таких атак характерны своего рода "штормы", огромное количество пакетов TCP SYN, сигнализирующих ничего не подозревающему хосту о том, что отправитель хочет установить с ним TCP-соединение. Хост-получатель резервирует память и отвечает на запрос сообщениями SYN ACK. Когда атакующая система не возвращает сообщения ACK, необходимые для завершения троекратной процедуры установки TCP-соединения, ресурсы хоста, подвергнувшегося атаке, остаются не освобожденными. Поэтому он оказывается не готов к обслуживанию легитимных запросов на установку TCP-соединения. При разработке SCTP этот недостаток был устранен в механизме четырехкратного рукопожатия. В него добавили необходимость прохождения процедуры cookie, только после этого сервер резервирует необходимые ресурсы.

Сравнение возможностей протоколов

Результаты анализа, который был проведен в данной статье, приведены в таблице.

В заключение можно сделать ряд выводов.

Протокол UDP не обеспечивает надежность доставки и не имеет никаких механизмов подтверждения передачи. Однако существуют приложения, в которых UDP использовать целесообразнее, чем TCP или SCTP. Обладая высокой скоростью передачи и простотой реализации, UDP идеально подходит для многоадресного трафика, сервисов потоковой передачи аудио- и видеоинформации, многопользовательских игр в реальном времени и некоторых протоколов сигнализации, использующихся в сетях VoIP. При реализации механизмов повышения надежности на уровне приложения, эта задача может решаться на канальном уровне, компенсируя недостатки протокола за счет создания избыточной полосы пропускания. Уже сейчас на уровне доступа по медным и оптическим линиям связи можно использовать каналы со скоростью до 10 Гбит/с, что казалось фантастикой во времена разработки первых протоколов транспортного уровня.

В отличие от UDP, TCP более сложный в реализации потоковый протокол, обладающий рядом недостатков, которые мы рассмотрели выше. Как и TCP, SCTP обеспечивает надежность передачи, но помимо этого он позволяет задавать границы сообщений, обеспечивает поддержку множественной адресации на транспортном уровне и предлагает расширенные возможности этого уровня, выходящие за рамки тех, которые могут сейчас предоставить TCP и UDP. Многие функции TCP поддерживаются и в SCTP: уведомление о приеме, повторная передача утерянных данных, сохранение последовательности данных, оконное управление передачей, медленный старт и алгоритмы предотвращения перегрузки, а также выборочные уведомления. Протокол SCTP позволяет приложению настраивать транспортный уровень по своим потребностям, причем настройка выполняется для каждой ассоциации в отдельности. Эта гибкость в сочетании с универсальным набором значений по умолчанию (для приложений, не нуждающихся в тонкой настройке транспортного уровня) дает приложению преимущества, которые оно не могло получить при работе с TCP. Протокол SCTP активно внедряется на сетях связи в более широкой области, чем та, для которой он создавался, но, по причине пассивности разработчиков приложений и операционных систем, не теми темпами, на которые рассчитывали его создатели.

Сравнительные характеристики протоколов транспортного уровня UDP, TCP и SCTP

Функция	UDP	TCP	SCTP
Установка соединения	Нет	Да	Да
Поддержка дуплексного режима	Да	Да	Да
Надежная передача	Нет	Да	Да
Частично надежная передача	Нет	Нет	Опционально
Упорядоченная доставка	Нет	Да	Да
Неупорядоченная доставка	Да	Нет	Да
Контроль потока	Нет	Да	Да
Управление потоком	Нет	Да	Да
Уведомление о перегрузке	Нет	Да	Да
Выборочное подтверждение	Нет	Опционально	Да
Сохранение границ сообщения	Да	Нет	Да
Обнаружение пути наименьшего MTU	Нет	Да	Да
Сегментация блоков данных приложения	Нет	Да	Да
Сборка блоков данных приложения	Нет	Да	Да
Многопоточность	Нет	Нет	Да
Поддержка множественных интерфейсов	Нет	Нет	Да
Защита от SYN-flood атак	–	Нет	Да
Поддержка полуоткрытых соединений	Нет	Да	Нет
Проверка достижимости	Нет	Да	Да
Проверка имитации заголовка	Да	Да	Нет (используются vtags)
Контроль работоспособности	Нет	Нет	Да

ЛИТЕРАТУРА:

1. Гольдштейн А.Б., Гольдштейн Б.С. Softswitch. – СПб.: БХВ-Санкт-Петербург, 2006.
2. Postel J., User datagram protocol, STD 6, RFC 768, August 1980.
3. Postel J., ed., Transmission control protocol – DARPA Internet Program Protocol Specification", RFC 793, USC/Information Sciences Institute, September 1981.
4. Stewart R., Stream control transmission protocol, RFC 4960, September 2007.
5. Amer P., Becke M., Dreibholz T. Load sharing for the stream control transmission protocol (SCTP), October 07, 2013
6. Стивенс У.Р., Феннер Б., Рудофф Э.М. UNIX: разработка сетевых приложений. 3-е изд. – СПб.: Питер, 2007.
7. Стюарт Р., Метц К. SCTP. Новый транспортный протокол для TCP/IP. – Открытые системы, 2002, №2.