Протоколы, сервисы и услуги
в Интернет и IP-сетях

Тема № 9
Протоколы транспортного уровня TCP и UDP

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2017 г.

Протокол TCP

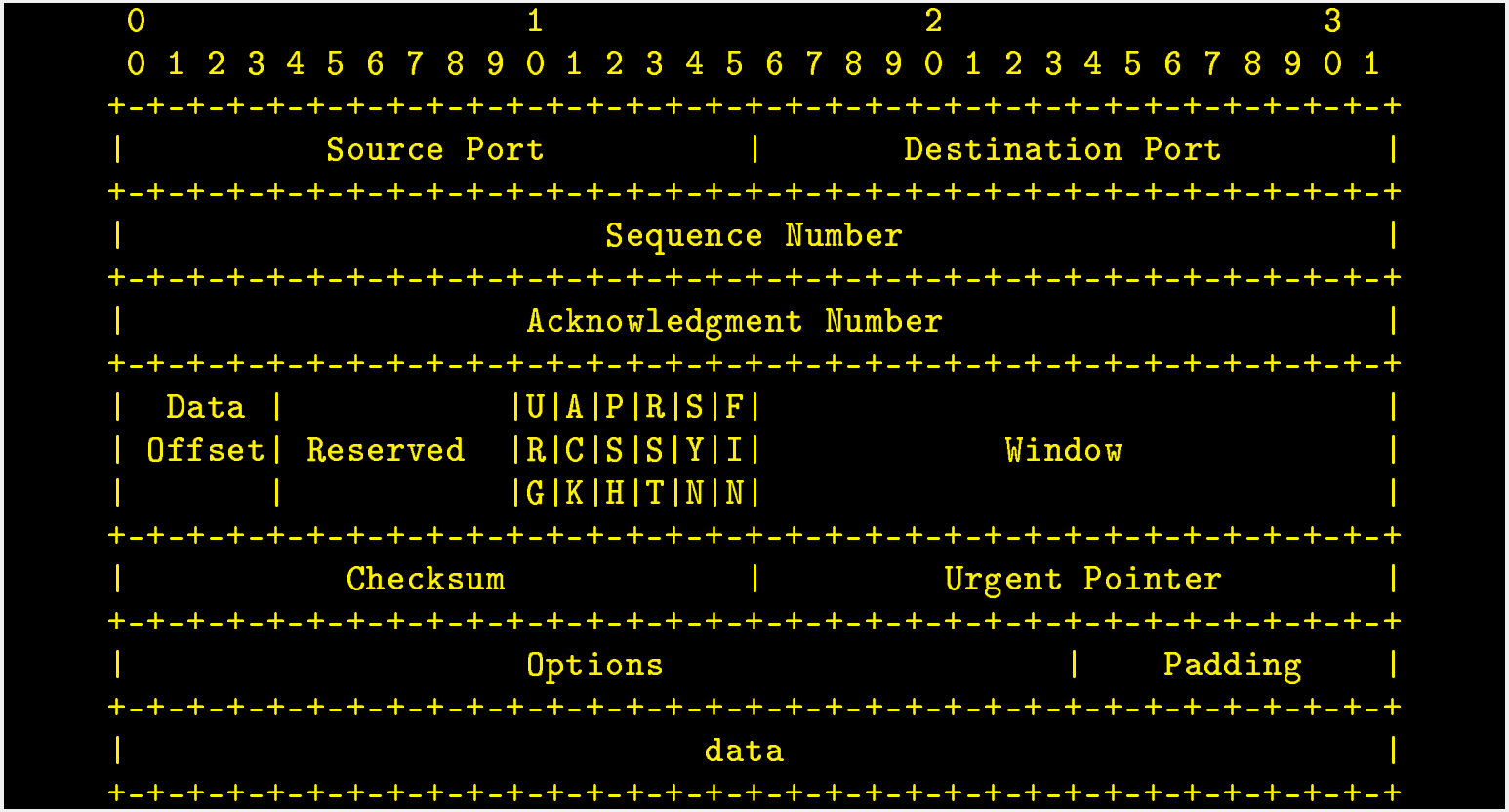
TCP (Transmission Control Protocol) — Протокол управления передачей

Один из основных протоколов транспортного уровня в стеке протоколов TCP/IP. Создан в 1981 г. Основной стандарт для протокола TCP — RFC 793 (STD 7) с обновлениями в RFC 1122, 3168, 6093, 6528.

TCP обеспечивает транспортный механизм, предоставляющий поток данных с предварительной установкой соединения, за счёт этого дающий уверенность в достоверности получаемых данных, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета. Гарантирует целостность передаваемых данных и уведомление отправителя о результатах передачи.
Реализация TCP, как правило, встроена в ядро ОС, хотя есть и реализации TCP в контексте приложения.

Когда осуществляется передача от компьютера к компьютеру через Интернет, TCP работает на верхнем уровне между двумя конечными системами, например, браузером и веб-сервером. Также TCP осуществляет надежную передачу потока байтов от одной программы на некотором компьютере к другой программе на другом компьютере. Программы для электронной почты и обмена файлами используют TCP. TCP контролирует длину сообщения, скорость обмена сообщениями, сетевой трафик.

Формат заголовка TCP



Поля заголовка TCP

Source Port. Порт источника

16-битное поле, идентифицирующее приложение клиента, с которого отправлены пакеты. Ответные данные передаются клиенту на основании этого номера.

Destination Port. Порт назначения

Идентифицирует порт (приложение/протокол), на который отправлен пакет.

Sequence Number. Номер последовательности

Номер последовательности выполняет две задачи:

1. Если установлен флаг SYN, то это начальное значение номера последовательности — ISN (Initial Sequence Number), и первый байт данных, которые будут переданы в следующем пакете, будет иметь номер последовательности, равный ISN + 1.
2. В противном случае, если SYN не установлен, первый байт данных, передаваемый в данном пакете, имеет этот номер последовательности.

Поскольку поток TCP в общем случае может быть длиннее, чем число различных состояний этого поля, то все операции с номером последовательности должны выполняться по модулю 2^{32}. Это накладывает практическое ограничение на использование TCP. Если скорость передачи коммуникационной системы такова, чтобы в течение MSL (максимального времени жизни сегмента) произошло переполнение номера последовательности, то в сети может появиться два сегмента с одинаковым номером, относящихся к разным частям потока, и приёмник получит некорректные данные.

Поля заголовка TCP

Acknowledgment Number. Номер подтверждения

Если установлен флаг ACK, то это поле содержит номер последовательности, ожидаемый получателем в следующий раз. Помечает этот сегмент как подтверждение получения.

Data Offset. Длина заголовка (смещение данных)

Это поле определяет размер заголовка пакета TCP в 4-байтных (4-октетных) словах. Минимальный размер составляет 5 слов, а максимальный — 15, что составляет 20 и 60 байт соответственно. Смещение считается от начала заголовка TCP.

Reserved. Зарезервировано

Зарезервировано (6 бит) для будущего использования и должно устанавливаться в ноль. На данный момент в RFC 3168 определены биты 5 и 6:

- CWR (Congestion Window Reduced) — «Окно перегрузки уменьшено» — флаг установлен отправителем, чтобы указать, что получен пакет с установленным флагом ECE.
- ECE (ECN-Echo) — «Эхо ECN» — указывает, что данный узел способен на ECN (явное уведомление перегрузки) и для указания отправителю о перегрузках в сети.

Поля заголовка TCP

Flags. Флаги (управляющие биты)

- URG — Поле «Указатель важности» задействовано (Urgent pointer field is significant).
- ACK — Поле «Номер подтверждения» задействовано (Acknowledgement field is significant).
- PSH — Инструктирует получателя протолкнуть данные, накопившиеся в приемном буфере, в приложение пользователя (Push function).
- RST — Оборвать соединения, сбросить буфер (очистка буфера) (Reset the connection).
- SYN — Синхронизация номеров последовательности (Synchronize sequence numbers).
- FIN — Указывает на завершение соединения (FIN bit used for connection termination).

Window. Размер окна

В этом поле содержится число, определяющее в байтах размер данных, которые отправитель готов принять.

Checksum. Контрольная сумма

Поле контрольной суммы — это 16-битное дополнение к сумме всех 16-битных слов заголовка (включая псевдозаголовок) и данных. Если сегмент, по которому вычисляется контрольная сумма, имеет длину не кратную 16-ти битам, то длина сегмента увеличивается до кратной 16-ти, за счет дополнения к нему справа нулевых бит заполнения. Биты заполнения (0) не передаются в сообщении и служат только для расчёта контрольной суммы. При расчёте контрольной суммы значение самого поля контрольной суммы принимается равным 0.

Поля заголовка TCP

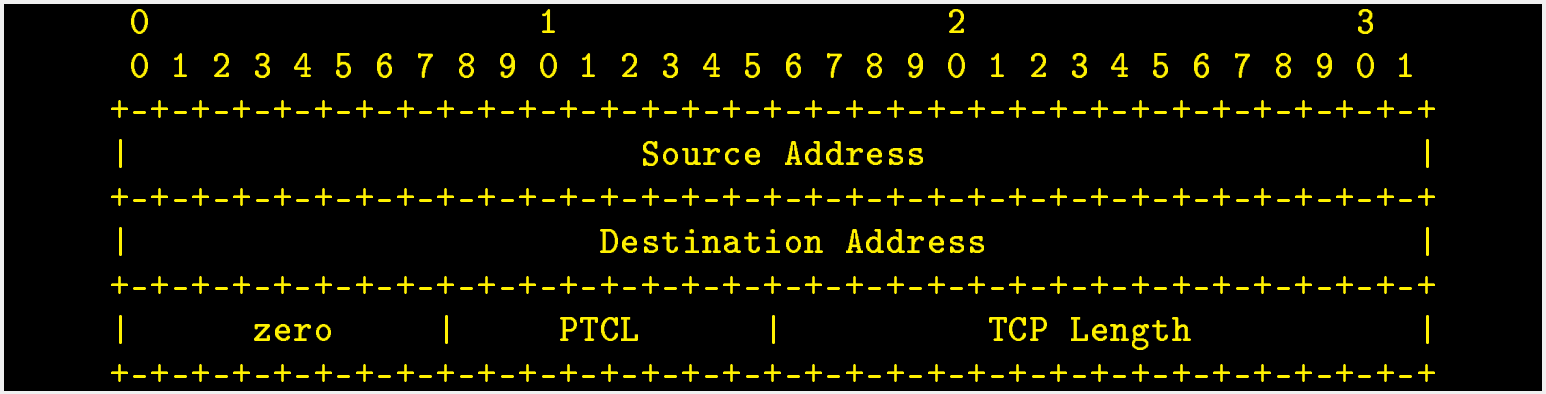
Urgent Pointer. Указатель важности

16-битовое значение положительного смещения от порядкового номера в данном сегменте. Это поле указывает порядковый номер октета, которым заканчиваются важные (urgent) данные. Поле принимается во внимание только для пакетов с установленным флагом URG.

Options. Опции

Могут применяться в некоторых случаях для расширения протокола. Иногда используются для тестирования. На данный момент в опции практически всегда включают 2 байта NOP (в данном случае `0x01`) и 10 байт, задающих временные метки (timestamps). Вычислить длину поля опции можно через значение поля смещения.

Псевдозаголовок TCP



TCP-заголовок не содержит информации об адресе отправителя и получателя, поэтому даже при совпадении порта получателя нельзя с точностью сказать, что сообщение пришло в нужное место. Поскольку назначением протокола TCP является надёжная доставка сообщений, то этот момент имеет принципиальное значение. Для того, чтобы не дублировать адресную информацию в заголовке TCP, был использован дополнительный псевдозаголовок.
Псевдозаголовок не включается в TCP-сегмент. Он используется для расчета контрольной суммы перед отправлением сообщения и при его получении (получатель составляет свой псевдозаголовок, используя адрес хоста, с которого пришло сообщение, и собственный адрес, а затем считает контрольную сумму).

Поля псевдозаголовка

Source Address — IP-адрес источника.

Dest. Address — IP-адрес получателя.

zero — Четыре нулевых бита.

PTCL — Тип протокола верхнего относительно IP уровня. В случае TCP Это поле равно `0x06`.

TCP Length — Содержит в себе длину TCP-сегмента в байтах (TCP-заголовок + данные; длина псевдозаголовка не учитывается).

Пример расчета контрольной суммы заголовка TCP

Контрольная сумма (КС, CS) заголовка TCP представляет собой 16-битовое поразрядное дополнение суммы всех 16-битовых слов заголовка, данных и псевдозаголовка. При вычислении контрольной суммы значение самого поля принимается нулевым.

Пример пакета TCP. Заголовок — зелёный. КС — синяя

```
0000: 00 50 FC 1E BF 8D 00 30 4F 0E 89 65 08 00 45 00
0010: 00 38 89 28 40 00 80 06 11 21 C0 A8 01 32 C3 13
0020: DB 88 04 50 00 15 00 4C 69 E7 3C 00 27 92 50 18
0030: 22 05 D3 39 00 00 55 53 45 52 20 61 6E 6F 6E 79
0040: 6D 6F 75 73 0D 0A
```

Псевдозаголовок

```
C0 A8 01 32
C3 13 DB 88
00 06 00 24
```

Вычисление КС заголовка TCP

1. Заголовок TCP разбивается на слова размером 16 бит (2 байта) каждое. Поле КС принимается равным нулю (не участвует в вычислении). Все полученные слова суммируются.

$$0450 + 0015 + 004C + 69E7 + 3C00 + 2792 + 5018 + 2205 + 0000 + 0000 = 14447$$

2. Аналогично считаем данные и псевдозаголовок.

$$5553 + 4552 + 2061 + 6E6F + 6E79 + 6D6F + 7573 + 0D0A = 287DA$$

$$C0A8 + 0132 + C313 + DB88 + 0006 + 0024 = 2609F$$

3. Итого: $14447 + 287da + 2609F = 62CC0$
4. Длина результата суммирования превышает 2 байта (4 шестнадцатеричные цифры), следовательно, производится перенос разряда: $0006 + 2CC0 = 2CC6$.
5. Находится поразрядное дополнение от итоговой суммы. Результат и будет контрольной суммой заголовка пакета TCP.

$$FFFF - 2CC6 = D339 = CS_{TCP}$$

Сеанс протокола TCP

Стадии TCP-соединения

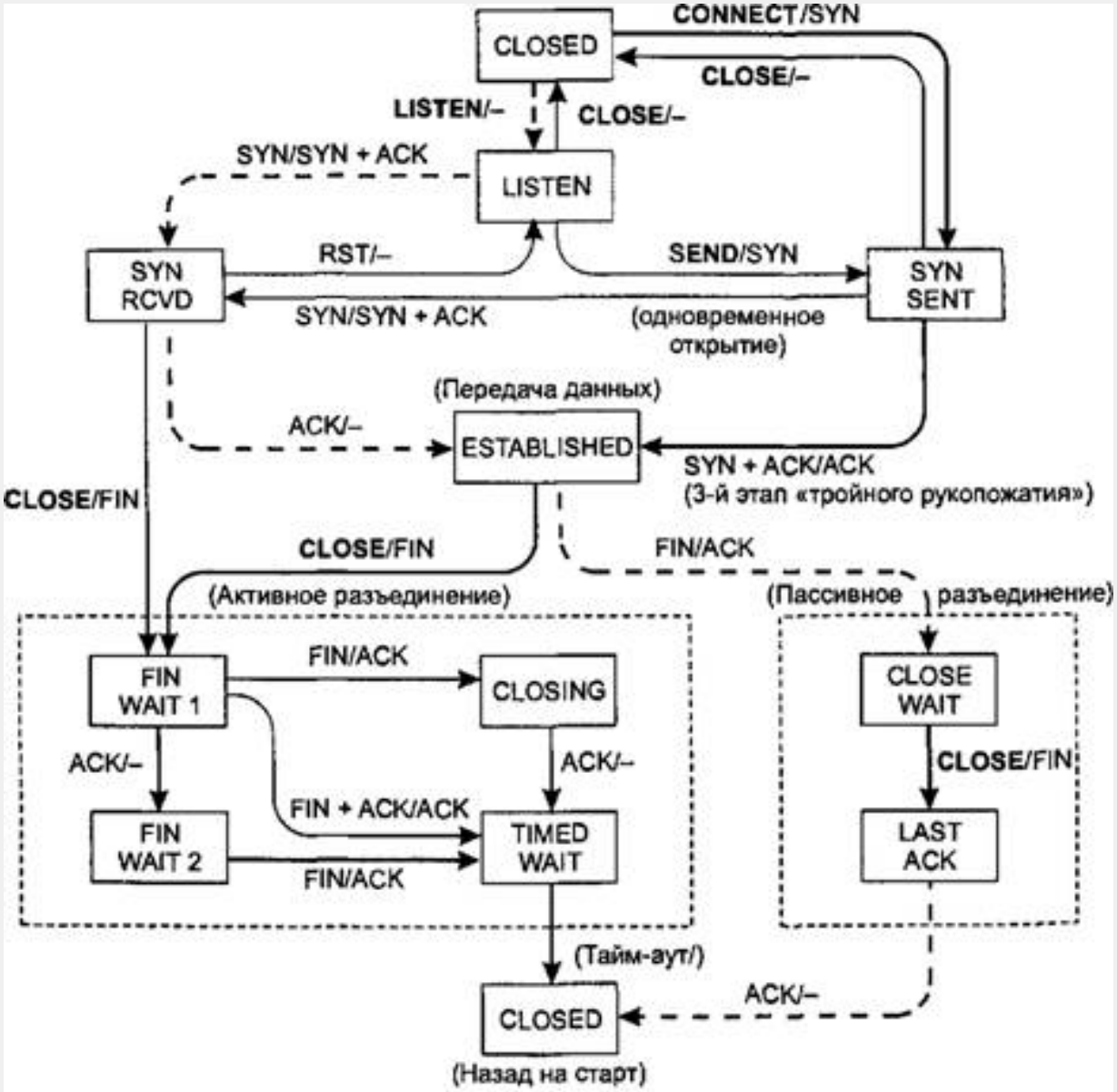
1. Установка соединения
2. Передача данных
3. Завершение соединения

Состояния сеанса TCP

CLOSED	Начальное состояние узла. Фактически фиктивное
LISTEN	Сервер ожидает запросов установления соединения от клиента
SYN-SENT	Клиент отправил запрос серверу на установление соединения и ожидает ответа
SYN-RECEIVED	Сервер получил запрос на соединение, отправил ответный запрос и ожидает подтверждения
ESTABLISHED	Соединение установлено, идёт передача данных
FIN-WAIT-1	Одна из сторон (назовём её узел-1) завершает соединение, отправив сегмент с флагом FIN
CLOSE-WAIT	Другая сторона (узел-2) переходит в это состояние, отправив, в свою очередь сегмент ACK и продолжает одностороннюю передачу
FIN-WAIT-2	Узел-1 получает ACK, продолжает чтение и ждёт получения сегмента с флагом FIN
LAST-ACK	Узел-2 заканчивает передачу и отправляет сегмент с флагом FIN
TIME-WAIT	Узел-1 получил сегмент с флагом FIN, отправил сегмент с флагом ACK и ждёт 2*MSL секунд, перед окончательным закрытием соединения
CLOSING	Обе стороны инициировали закрытие соединения одновременно: после отправки сегмента с флагом FIN узел-1 также получает сегмент FIN, отправляет ACK и находится в ожидании сегмента ACK (подтверждения на свой запрос о разъединении)

Сеанс протокола TCP

Конечный автомат TCP



Установление TCP-соединения

Процедура установления TCP-соединения называется «трехэтапным согласованием» («three way handshake»).

1. Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN.
   - Сервер получает сегмент, запоминает номер последовательности и пытается создать сокет (буферы и управляющие структуры памяти) для обслуживания нового клиента.
   - В случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK, и переходит в состояние SYN-RECEIVED.
   - В случае неудачи сервер посылает клиенту сегмент с флагом RST.
2. Если клиент получает сегмент с флагом SYN, то он запоминает номер последовательности и посылает сегмент с флагом ACK.
   - Если он одновременно получает и флаг ACK (что обычно и происходит), то он переходит в состояние ESTABLISHED.
   - Если клиент получает сегмент с флагом RST, то он прекращает попытки соединиться.
   - Если клиент не получает ответа в течение 10 секунд, то он повторяет процесс соединения заново.
3. Если сервер в состоянии SYN-RECEIVED получает сегмент с флагом ACK, то он переходит в состояние ESTABLISHED. В противном случае после тайм-аута он закрывает сокет и переходит в состояние CLOSED.

Существуют экспериментальные расширения протокола TCP, сокращающие количество пакетов при установлении соединения, например TCP Fast Open. Ранее также существовало расширение T/TCP.

Пример установления TCP-соединения

```
          TCP A                                                TCP B
1.  CLOSED                                               LISTEN
2.  SYN-SENT    → <SEQ=100><CTL=SYN>                     → SYN-RECEIVED
3.  ESTABLISHED ← <SEQ=300><ACK=101><CTL=SYN,ACK>        ← SYN-RECEIVED
4.  ESTABLISHED → <SEQ=101><ACK=301><CTL=ACK>            → ESTABLISHED
5.  ESTABLISHED ← <SEQ=301><ACK=101><CTL=ACK>            ← ESTABLISHED
```

В строке 2 TCP A начинает передачу сегмента SYN, говорящего об использовании номеров последовательности, начиная со 100. В строке 3 TCP B передает SYN и подтверждение для принятого SYN в адрес TCP A. Надо отметить, что поле подтверждения показывает ожидание TCP B приема номера последовательности 101, подтверждающего SYN с номером 100.
В строке 4 TCP A отвечает пустым сегментом с подтверждением ACK для сегмента SYN от TCP B; в строке 5 TCP B передает некоторые данные. Отметим, что номер последовательности сегмента в строке 5 совпадает с номером в строке 4, поскольку ACK не занимает пространства номеров последовательности (если это сделать, придется подтверждать подтверждения — ACK для ACK).

Передача данных в TCP

При обмене данными приемник использует номер последовательности, содержащийся в получаемых сегментах, для восстановления их исходного порядка. Приемник уведомляет передающую сторону о номере последовательности байт, до которой он успешно получил данные, включая его в поле «номер подтверждения». Все получаемые данные, относящиеся к промежутку подтвержденных последовательностей, игнорируются. Если полученный сегмент содержит номер последовательности больший, чем ожидаемый, то данные из сегмента буферизируются, но номер подтвержденной последовательности не изменяется. Если впоследствии будет принят сегмент, относящийся к ожидаемому номеру последовательности, то порядок данных будет автоматически восстановлен исходя из номеров последовательностей в сегментах.

Для того, чтобы передающая сторона не отправляла данные интенсивнее, чем их может обработать приемник, TCP содержит средства управления потоком. Для этого используется поле «окно». В сегментах, направляемых от приемника передающей стороне в поле «окно» указывается текущий размер приемного буфера. Передающая сторона сохраняет размер окна и отправляет данных не более, чем указал приемник. Если приемник указал нулевой размер окна, то передача данных в направлении этого узла не происходит, до тех пор пока приемник не сообщит о большем размере окна.

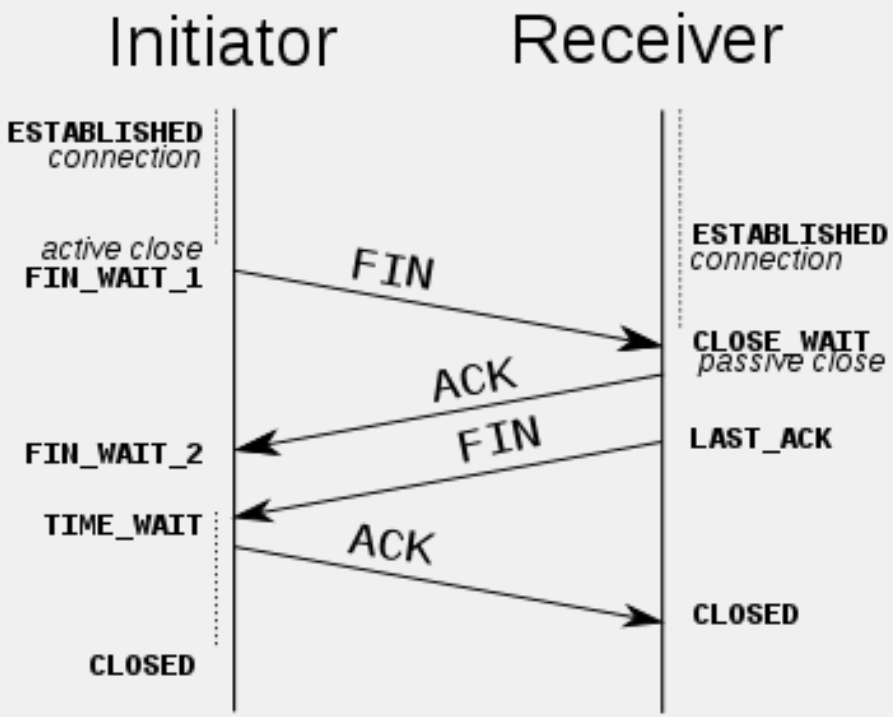
В некоторых случаях передающее приложение может явно затребовать протолкнуть данные до некоторой последовательности принимающему приложению, не буферизируя их. Для этого используется флаг PSH. Если в полученном сегменте обнаруживается флаг PSH, то реализация TCP отдает все буферизированные на текущий момент данные принимающему приложению. «Проталкивание» используется, например, в интерактивных приложениях. В сетевых терминалах нет смысла ожидать ввода пользователя после того, как он закончил набирать команду. Поэтому последний сегмент, содержащий команду, обязан содержать флаг PSH, чтобы приложение на принимающей стороне смогло начать её выполнение.

Завершение TCP-соединения

Этапы завершения соединения

1. Посылка серверу от клиента флагов FIN и ACK на завершение соединения.
2. Сервер посылает клиенту флаги ответа ACK, FIN, что соединение закрыто.
3. После получения этих флагов клиент закрывает соединение и в подтверждение отправляет серверу ACK , что соединение закрыто.

Диаграмма завершения соединения



Особенности протокола TCP

Максимальный размер сегмента

TCP требует явного указания максимального размера сегмента (MSS) в случае, если виртуальное соединение осуществляется через сегмент сети, где максимальный размер блока (MTU) менее, чем стандартный MTU Ethernet (1500 байт).
В протоколах туннелирования, таких как GRE, IPIP, а также PPPoE MTU туннель меньше чем стандартный, поэтому сегмент TCP максимального размера имеет длину пакета больше, чем MTU. Это приводит к фрагментации и уменьшению скорости передачи полезных данных. Если на каком-либо узле фрагментация запрещена, то со стороны пользователя это выглядит как «зависание» соединений. При этом «зависание» может происходить в произвольные моменты времени, а именно тогда, когда отправитель использовал сегменты длиннее допустимого размера. Для решения этой проблемы на маршрутизаторах применяются правила Firewall, добавляющие параметр MSS во все пакеты, инициирующие соединения, чтобы отправитель использовал сегменты допустимого размера.
MSS может также управляться параметрами операционной системы.

Обнаружение ошибок при передаче данных

Хотя протокол осуществляет проверку контрольной суммы по каждому сегменту, используемый алгоритм считается слабым. Так, в 2008 году ошибка в передаче одного бита, не обнаруженная сетевыми средствами, привела к остановке серверов системы Amazon Web Services.

Освобождение от расчёта контрольной суммы

Многие реализации стека TCP/IP предоставляют возможности использования аппаратной поддержки для автоматического расчёта контрольной суммы в сетевом адаптере до передачи в сеть или после приёма из сети для верификации. Это может освобождать операционную систему от использования ценных тактов процессора при вычислении контрольной суммы. Эта функция может приводить к тому, что анализаторы трафика, перехватывающие исходящие пакеты до их передачи в сетевой адаптер и не знающие о делегировании расчёта контрольной суммы сетевому адаптеру, могут сообщать об ошибке контрольной суммы в исходящих пакетах.

Протокол UDP

UDP (User Datagram Protocol)

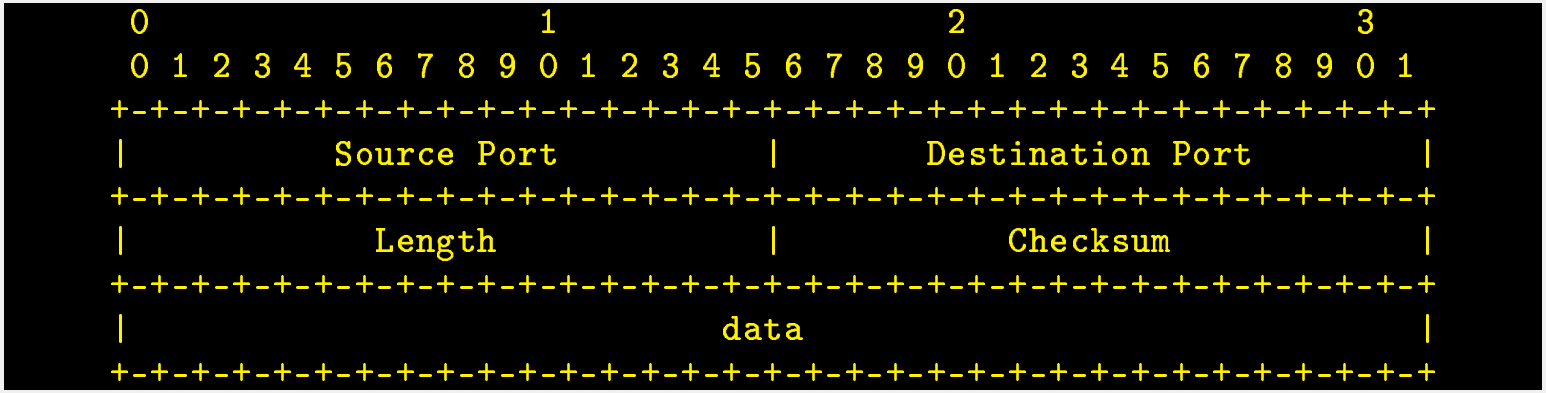
UDP позволяет посылать сообщения (датаграммы) другим хостам по IP-сети без необходимости предварительного установления соединений. Был разработан Дэвидом П. Ридом в 1980 году и официально определён в RFC 768 (STD 6).

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. По этой причине UDP иногда называют Unreliable Datagram Protocol (Ненадёжный протокол датаграмм). UDP подразумевает, что проверка ошибок и исправление либо не нужны, 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.

Структура пакета UDP. Поля заголовка UDP

Заголовок UDP



Source Port. 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, динамическим. Если источником является сервер, то его порт будет одним из зарегистрированных в IANA.

Destination Port. Порт получателя

Это поле обязательно и содержит порт получателя. Аналогично порту отправителя, если хостом-получателем является клиент, то номер порта динамический, если получатель — сервер, то это будет порт из списка IANA.

Поля заголовка UDP

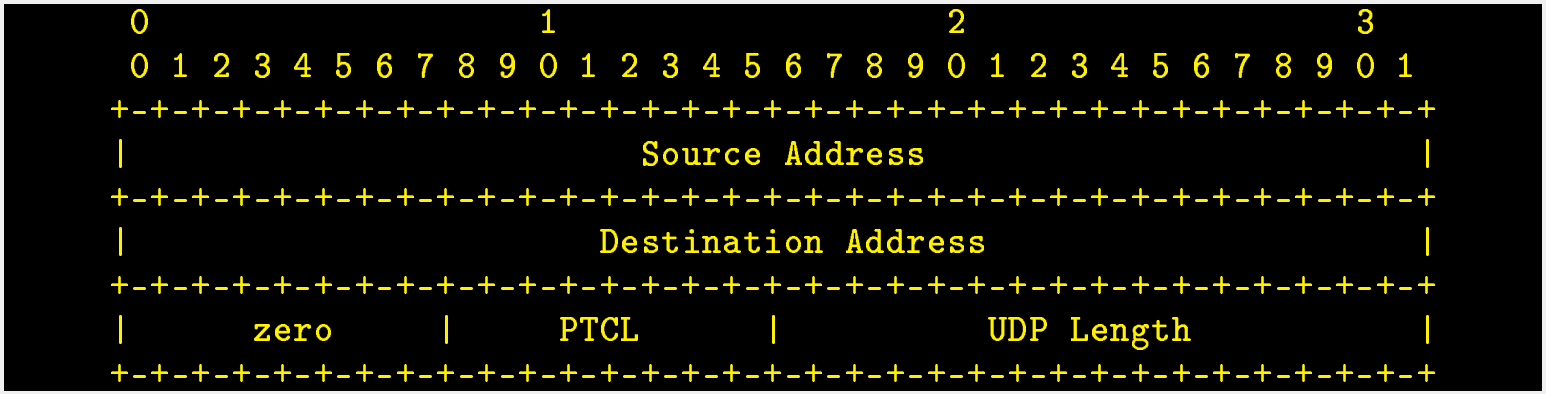
Length. Длина датаграммы

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).
На практике также следует учитывать, что если длинна IPv4 пакета с UDP будет превышать MTU (для Ethernet по умолчанию 1500 байт), то отправка такого пакета вызовет его фрагментацию, что может привести к тому, что он вообще не сможет быть доставлен, если промежуточные маршрутизаторы или конечный хост не будут поддерживать фрагментированные IP пакеты. Также в RFC 791 указывается минимальная длина IP пакета не менее 576 байт и рекомендуется отправлять IP пакеты большего размера только в том случае если вы уверены, что принимающая сторона может принять пакеты такого размера. Следовательно, чтобы избежать фрагментации UDP пакетов (и возможной их потери), размер данных в UDP не должен превышать: $MTU - (MaxIPHeaderSize) - (UDPHedaerSize) = 1500 - 60 - 8 = 1432$ байт. Для того чтобы быть уверенным, что пакет будет принят любым хостом, размер данных в UDP не должен превышать: $(MinIP - packetLength) - (MaxIPHeaderSize) - (UDPHeaderSize) = 576 - 60 - 8 = 508$ байт.
В Jumbogram 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.

Checksum. 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. Метод для вычисления контрольной суммы определён в RFC 768. Методика расчета контрольной суммы заголовка UDP полностью аналогична методике расчета контрольной суммы заголовка TCP.

Псевдозаголовок UDP



Поля псевдозаголовка

Source Address — IP-адрес источника.

Dest. Address — IP-адрес получателя.

zero — Четыре нулевых бита.

PTCL — Тип протокола верхнего относительно IP уровня. В случае UDP Это поле равно `0x11`.

UDP Length — Содержит в себе длину UDP-датаграммы в байтах (UDP-заголовок + данные; длина псевдозаголовка не учитывается). Соответствует полю Length заголовка UDP.

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 воспользоваться методами помехоустойчивого кодирования.

Более серьёзной потенциальной проблемой является то, что в отличие от TCP, основанные на UDP приложения не обязательно имеют хорошие механизмы контроля и избегания 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 (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.

Приложения и протоколы, использующие UDP

Многочисленные ключевые Интернет-протоколы 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.

Сравнение UDP и TCP

TCP

Ориентированный на соединение протокол, что означает необходимость «рукопожатия» для установки соединения между двумя хостами. Как только соединение установлено, пользователи могут отправлять данные в обоих направлениях.

- Надёжность — управляет подтверждением, повторной передачей и тайм-аутом сообщений. Производятся многочисленные попытки доставить сообщение. Если оно потеряется на пути, сервер вновь запросит потерянную часть. Нет ни пропавших данных, ни (в случае многочисленных тайм-аутов) разорванных соединений.
- Упорядоченность — если два сообщения последовательно отправлены, первое сообщение достигнет приложения-получателя первым. Если участки данных прибывают в неверном порядке, TCP отправляет неупорядоченные данные в буфер до тех пор, пока все данные не могут быть упорядочены и переданы приложению.
- Тяжеловесность — необходимо три пакета для установки соединения перед отправкой данных. Следит за надёжностью и перегрузками.
- Потоковость — данные читаются как поток байтов без особых обозначений для границ сообщения или сегментов.

UDP

Простой, основанный на сообщениях протокол без установления соединения. Связь достигается путем передачи информации в одном направлении от источника к получателю без проверки готовности или состояния получателя.

- Ненадёжный — неизвестно, достигнет ли посланное своего назначения — оно может потеряться по пути. Нет таких понятий, как подтверждение, повторная передача, тайм-аут.
- Неупорядоченность — если два сообщения отправлены одному получателю, то порядок их достижения цели не может быть предугадан.
- Легковесность — нет упорядочивания сообщений, отслеживания соединений и т. д.
- Датаграммы — пакеты посылаются по отдельности и проверяются на целостность только если они прибыли. Пакеты имеют определенные границы, которые соблюдаются после получения, то есть операция чтения на сокете-получателе выдаст сообщение таким, каким оно было изначально послано.
- Нет контроля перегрузок — UDP сам по себе не избегает перегрузок. Для приложений с большой пропускной способностью возможно вызвать коллапс перегрузок, если они не реализуют меры контроля на прикладном уровне.

Источники

- Материалы с сайта `https://wikipedia.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов. URL: `http://book.itep.ru/`
- RFC 793. Transmission Control Protocol.
- RFC 768. User Datagram Protocol.