КЛАСИЧЕСКИЙ ПРИВАТНЫЙ УНИВЕРСИТЕТ

РЕФЕРАТ

с дисциплины: Компьютерные сети (Локальные, корпоративные, глобальные)

на тему: «Протоколы транспортного уровня»

Выполнил студент группы:

ДИ-204 Шевченко М.Ю.

# ВВЕДЕНИЕ:

Приложение, работающее с Интернет, как правило, общается с одним из протоколов транспортного уровня TCP/IP: протоколом управления транспортировкой (TCP) или протоколом пользовательских датаграмм (UDP). Приложение строит свою работу на взаимодействии с одним из этих протоколов.

Приложения Интернет, например программа ftp, передающая файлы по сети, обычно использует TCP, так как он предлагает надежную потокоориентированную службу доставки. Приложения типа электронной почты часто пользуются TCP по той же самой причине. Не требующие особой надежности приложения типа tftp *(протокол простой передачи файлов,* trivial file transfer protocol) используют UDP. Приложения на основе протокола времени (time protocol), связывающиеся с серверами времени Интернет, могут пользоваться как тем, так и другим протоколом. Прочтя эту главу, вы будете точно знать, в каком случае может потребоваться TCP, а в каком — UDP.

Конечная цель сетевого взаимодействия заключается в передаче информации между приложением-клиентом и приложением-сервером. Понимание транспорт­ных протоколов и транспортного уровня совершенно необходимо для эффективного проектирования приложений типа клиент-сервер. Закончив чтение данной главы, вы овладеете следующими понятиями и концепциями:

* Как транспортный протокол использует порт протокола для связи с програм­мой-приложением .
* Назначение полей данных в заголовке UDP.
* Как TCP обеспечивает надежную доставку данных.
* Как TCP использует скользящее окно для увеличения пропускной способ­ности сети.
* Как модули TCP устанавливают и заканчивают соединение.
* Как TCP использует сообщения-подтверждения.
* Назначение полей данных в заголовке TCP.

Что такое транспортный уровень?

На первый взгляд кажется, что IP, служба доставки Интернет, и транспортные протоколы выполняют одинаковые функции. На самом деле IP модуль достав­ляет данные только между двумя компьютерами. Транспортный уровень и его протоколы передают данные между приложениями.

Во многих случаях ответственность транспортных протоколов за передаваемые данные такая же, как и у протокола Интернет. И вообще, большинство из того, что вы узнали относительно IP-датаграммы и IP-заголовка, в полной мере применимо и к транспортным протоколам. Вы знаете, что TCP/IP включает два транспортных протокола: протокол управления транспортировкой (собственно, транспортный протокол, TCP) и протокол пользовательских датаграмм (UDP). Ориентированный на соединение протокол управления транспортировкой для приема и передачи данных использует надежный поточно-байтовый способ доставки. Сетевое соединение устанавливается в виде виртуальной цепи. Про­токол пользовательских датаграмм ненадежен, не ориентирован на соединение, передает и принимает данные при помощи датаграмм.

Что такое порт транспортного уровня?

Понятие «порте» в терминологии TCP/IP очень похоже на IP-адрес компьютера. Только порт обозначает приложение, а IP-адрес — определенный компьютер (вернее, его сетевой интерфейс). Так же как IP-датаграммы содержат адреса источника и получателя данных, транспортные протоколы хранят номера портов источника и получателя. Если вышесказанное кажется вам несколько странным, давайте рассмотрим, что нам известно об аппаратных портах нашего персональ­ного компьютера. Может быть, вам приходилось писать программу, посылаю­щую данные на порт компьютера. Если нет, то вспомним процесс печати через параллельный или последовательный порт. Если вам приходилось пользоваться

модемом, концепция портов TCP/IP должна показаться вам еще более зна­комой.

***Примечание:*** *В* случае модема, компьютер принимает и посылает данные через последовательный порт. В случае принтера, данные обыкновенно только посылаются.

Порты персональных компьютеров имеют название и номер. Параллельные порты компьютера называются LPT1 и LPT2. Последовательные — СОМ1 и COM2. В сети Интернет номера портов протоколов только нумеруются. Парал­лельный порт LPT1 персональных компьютеров годами использовался для печати данных. Тысячи программных продуктов знают, что печатать надо через LPT1. Точно так же порт протокола Интернет ассоциируется со вполне опреде­ленным приложением или функцией.

**Что такое порт протокола Интернет?**

Мы уже упоминали, что Интернет включает протоколы для некоторых часто используемых приложений, например ftp, telnet или электронной почты. Данным приложениям Интернет присвоены так называемые *официальные* (well-known) номера портов. «Официальность» порта состоит в том, что его номер широко известен и применяется всеми компьютерами в Интернет, работающими с данным сетевым приложением или функцией.

Так же как программисты персональных компьютеров пользуются портом LPT1 для печати, программисты Интернет пользуются набором номеров портов для выполнения конкретных сетевых приложений. Например, официальный (общеизвестный) номер порта для простого протокола передачи файлов (tftp) — 69. Официальный номер порта telnet — 23. В табл. 1 приведен список широко используемых портов протоколов Интернет.

*Таблица 1. Некоторые официальные номера портов протоколов Интернет*

**Номера портов которые существуют:**

Служба эхо (Echo Protocol) 7

Время суток (Daytime Protocol) 13

Протокол передачи файлов (File Transfer Protocol) 21

Протокол удаленного терминала (Telnet Protocol) 23

Простой протокол передачи почты 25

(Simple Mail Transfer Protocol)

Точное время (Time Protocol) 37

Кто-есть-кто (Whois Protocol) 43

Протокол простой передачи файлов 69

(Trivial File Transfer Protocol)

Информация о пользователях (Finger Protocol) 79

**Как используется порт UDP?**

Протокол, не ориентированный на соединение (IP или UDP), можно сравнить с почтовой службой доставки. Если вы забыли эту аналогию, перечитайте раздел «Словарь терминов расширяется» в третьей главе. Аналогия позволяет увидеть отношение между UDP, портами и приложениями. Здесь почтовое отделение превращается в сетевой компьютер, абонентские ящики — в порты, а люди, их арендующие, — в прикладные программы.

Протокол Интернет — это сетевая служба доставки. Прежде вы думали, что IP похож на почтового работника. Теперь же IP больше походит на грузовик, развозящий письма между почтовыми отделениями, а транспортный протокол — на почтового работника, сортирующего и раскладывающего письма по абонен­тским ящикам.

Грузовик (IP) развозит письма (данные) по почтовым отделениям (сетевым компьютерам). Почтовые работники (UDP) сортируют почту по номерам або­нентских ящиков (портам). Рассортировав письма, почтовые работники (UDP) кладут письма (данные) в абонентские ящики (порты). Клиенты почтового отделения (приложения) периодически проверяют свои ящики и забирают почту. Почтовые работники (UDP) не уведомляют клиентов (приложения) о приходе свежей корреспонденции (данных), а просто размещают ее в абонент­ском ящике (порту).

Как используется порт TCP?

Поскольку TCP является надежным и ориентированным на соединение прото­колом, его способ использования портов несколько отличается от способа UDP. Например, будучи не ориентированным на соединение, UDP просто доставляет данные до определенного порта и не обеспечивает никакого соединения между передатчиком и получателем. TCP ориентирован на соединение, поэтому достав­ка данных для него — не просто передача в порт, но в первую очередь соединение. Например, приложение TCP, пожелавшее открыть несколько со­единений одновременно на одном и том же порту, может без проблем сделать это — данные не потеряются.

Мы уже говорили о том, что TCP больше похож на телефонные переговоры, чем на почтовую службу. Сейчас мы немного изменим телефонную аналогию и представим ее в следующем виде. Офис станет сетевым компьютером, номер телефона — портом, а телефонный звонок — сетевым соединением. Служащие в офисе будут представлять прикладные протоколы, а их телефонные перего­воры — обмен данными. Как и прежде, IP представляет телефонную компанию.

Служащие (прикладные протоколы), работающие в офисе (сетевом компьюте­ре), пользуются услугами телефонной компании (IP). Каждому служащему присвоен определенный телефонный номер (порт). Несколько телефонных линий (портов) остаются все время свободными, то есть ими может воспользо­ваться любой желающий. Каждое рабочее место оборудовано телефонным аппаратом, с которого можно звонить (устанавливать соединение), пользуясь любой не занятой в данный момент телефонной линией (портом) в офисе (сетевом компьютере).

Телефонная компания (IP) передает все входящие звонки в офис (сетевой компьютер), заставляя телефонные аппараты звонить. Определенный номер (порт) соответствует определенному служащему (приложению), то есть тому, кто отвечает на звонок (устанавливает соединение). В начале служащий (при­кладной протокол) с определенным номером (портом) всегда отвечает на звонок (устанавливает соединение). Далее, если служащий решает продолжить перего­воры со звонящим абонентом, он некоторое время беседует с ним по телефону (производит обмен данными).

Служащий (прикладной протокол) отвечает на звонки различным образом. Например, он может пригласить другого служащего принять участие в перего­ворах. При этом другой служащий (прикладной протокол) поднимает трубку параллельного аппарата (порта) и участвует в переговорах (устанавливает еще одно соединение по тому же самому порту). В некоторых случаях служащий не хочет занимать служебную линию связи (официальный порт) и переключает собеседника на одну из редко используемых линий (другой порт), чтобы никто не прерывал беседу (обмен данными).

Как номер порта используется в программе?

Поскольку транспортный уровень перемещает пакеты данных к прикладным программам и от них, он должен каким-то образом распознавать те программы, с которыми имеет дело. Тут на сцену и выступают номера портов. Любое приложение, независимо от того, сервер оно или клиент, имеет уникальный номер порта. Когда программа устанавливает соединение с сетью, ей присваи­вается определенный номер порта. Разрабатывая приложение-клиент, обычно не нужно беспокоиться по поводу номера его порта. Клиент может заранее и не знать его. Но совсем другое дело — приложение-сервер. Каждый раз, когда клиент посылает сообщение, транспортный уровень автоматически присваивает ему правильный номер порта в поле порта источника сообщения. В главе 19 описан процесс создания приложения-сервера, когда вы можете запросить у сети назначить ему определенный, заранее известный номер порта. Приложение-сер­вер, таким образом, может обслуживать запросы, поступающие с официальных номеров портов, перечисленных, например, в табл. .1.

Что такое протокол пользовательских датаграмм?

UDP весьма похож на IP в том смысле, что они оба ненадежные, не ориенти­рованные на соединение протоколы, пользующиеся датаграммами. Они оба переносят данные между компьютерами, однако только UDP умеет распо­знать то приложение среди многих, работающих внутри компьютера, которому предназначены данные. Как правило, сеть назначает таким приложениям опре­деленный номер порта. Итак, UDP пользуется датаграммами для доставки данных. Точно так же, как IP прицепляет к данным IP-заголовок, UDP прицеп­ляет к ним UDP-заголовок. Структура UDP-заголовка намного проще. На рис. 5.1 изображена структура UDP-датаграммы. UDP-заголовок содержит четыре поля: «порт-источник», «порт-получатель», «длина сообщения» и «конт­рольная сумма».

Длина UDP-заголовка — восемь байтов. Поля портов состоят из 16-битных целых чисел, представляющих номера портов протоколов. Поле «порт-источ­ник» содержит номер порта, которым пользуется приложение-источник данных. Поле «порт-получатель» соответственно указывает на номер порта приложения-получателя данных. Поле «длина сообщения» определяет длину (в байтах) UDP-датаграммы, включая UDP-заголовок. Наконец, поле «контрольная сумма», в отличие от контрольной суммы IP-заголовка, содержит результат суммирования всей UDP-датаграммы, включая ее данные, область которых начинается сразу после заголовка.

***Примечание:*** Несмотря на то, что контрольная сумма UDP включает область данных, подсчитывать и помещать ее в заголовок не обязательно. Такое поведение не характерно, например, для протоколов IP или TCP. Последние обязаны подсчитать и включить в свой заголовок контроль­ную сумму.

Модуль UDP отслеживает появление вновь прибывших датаграмм, сортирует их и распределяет (демультиплексирует) в соответствии с портами назначения. На рис. 5.2 показан поток данных, следующий сквозь сетевой уровень и модуль UDP к прикладным программам.


### Рис..2. Поток данных через модуль UDP

Что такое транспортный протокол?

Транспортный протокол (Transport Control Protocol, TCP) наряду с протоколом IP — один из наиболее распространенных в Интернет. Так же, как и UDP, TCP служит для передачи данных между сетевым и прикладным уровнями сетевой модели. По сравнению с UDP TCP устроен гораздо сложнее. И это понятно — ведь ему приходится обеспечивать надежную, потоковую, ориентированную на соединение службу доставки данных. Другими словами, TCP сам следит за доставкой данных, а также за правильной последовательностью передаваемых пакетов. В отличие от TCP протокол пользовательских датаграмм (UDP) доставки данных не гарантирует. Не обеспечивает он и правильной последова­тельности прихода датаграмм.

TCP пытается оптимизировать пропускную способность сети, то есть увеличи­вает производительность доставки пакетов в Интернет, насколько это вообще возможно. Для этого он динамически управляет потоком данных в соединении. Если буфер приемника данных на принимающем конце переполняется, TCP просит передающую сторону снизить скорость передачи.

Рассмотрим, каким образом протокол управления транспортировкой пользуется услугами IP для передачи данных между двумя компьютерами. Может пока­заться странным, что TCP умудряется пользоваться ненадежным IP и при этом оставаться надежным. Также может казаться удивительным, что TCP, пользуясь не ориентированным на соединение IP, остается ориентированным на соедине­ние. Наконец, как получается, что TCP доставляет данные в виде потока байтов, тогда как IP пересылает датаграммы? Последующие абзацы ответят на все эти вопросы и устранят возможные недоразумения. Постигая секреты TCP, вы должны помнить о том, что его данные всегда переносит IP. To есть данные TCP всегда упаковываются в IP-датаграммы.

Обеспечение надежности

Для обеспечения надежной доставки и правильной последовательности данных в потоке TCP пользуется подтверждениями. Как только пункт назначения принимает блок данных, он передает подтверждение о приеме источнику данных. Источнику данных это сообщение говорит: «Все в порядке. Я принял твои данные». Каждый раз при передаче сообщения модуль TCP запускает специальный таймер. По истечении заданного в нем времени и не получении подтверждения TCP повторяет попытку передать свое сообщение. На рис. 5.3 схематично показана работа такой системы.

*Рис. .3. Передача данных с простым подтверждением о доставке*

К несчастью, простое подтверждение о доставке, изображенное на рис. 5.3, работает исключительно неэффективно. Одна из сторон соединения вынуждена все время ждать появления подтверждения о доставке от другой стороны. Вскоре вы узнаете, что на самом деле TCP не использует такую простейшую схему подтверждения, при которой пакеты и подтверждения следуют по очереди друг за другом.

Что такое скользящее окно?

TCP не посылает один пакет, ожидая прихода подтверждения, чтобы послать следующий. Вместо этого он использует принцип *«скользящего* окна». Этот принцип позволяет послать несколько сообщений и только потом ожидать подтверждения. «Скользящее окно» проиллюстрировано на рис. 5.4.

Рис. .4. Скользящее окно TCP

Что такое транспортный протокол?

Выражаясь образно, TCP как бы накладывает окно на поток данных, ожидаю­щих передачи, и передает все данные, попавшие в окно. Приняв подтверждение о доставке всех данных, TCP перемещает окно дальше по потоку и передает следующие попавшие в него сообщения. Работая сразу с несколькими сообще­ниями, TCP может одновременно «выставить» их на сетевой канал и только потом ожидать прихода подтверждения. Метод скользящего окна значительно увеличивает производительность соединения, а также эффективность циклов обмена сообщениями и подтверждениями об их доставке. Рис. 5.5 иллюстрирует цикл обмена сообщение-подтверждение TCP.

*Рис. .5. Передача сообщений и подтверждений о доставке по схеме скользящего окна*

Передатчик и приемник на рис. 5.5 используют скользящее окно шириной в три пакета. То есть передатчик сначала высылает три пакета и только после ждет прихода подтверждения. Приняв подтверждение о доставке третьего последнего пакета, передатчик может посылать следующие три.

TCP регулирует полосу пропускания сети, договариваясь с другой стороной о некоторых параметрах потока данных. Причем процесс регулировки происходит на протяжении всего соединения TCP. В частности, регулировка заключается в изменении размеров скользящего окна. Если сеть загружена не сильно и вероятность столкновения данных минимальна, TCP может увеличить размер скользящего окна. При этом скорость выдачи данных на канал увеличивается и соединение становится более эффективным, поскольку через сеть проходит больше данных за одно и то же время.

Если, наоборот, вероятность столкновения данных велика, TCP уменьшает размер скользящего окна. Если размер скользящего окна, изображенного на рис. 5.4, принять равным восьми пакетам при обычном сетевом трафике, то в худших условиях, когда Интернет сильно загружен, его размер может умень­шиться до пяти. Наоборот, когда данных в сети немного, размер окна может увеличиться, например, до 10—20 пакетов.

Имейте в виду, что представленная на рис. 5.4 и описанная в предыдущих абзацах схема несколько упрощена. На самом деле TCP задает размер окна в байтах. То есть размер окна по умолчанию может равняться нескольким тысячам байтов, а не восьми, десяти и двенадцати байтам, как в предыдущем примере. Как правило, модуль TCP передает несколько сегментов, прежде чем скользя­щее окно заполнится целиком. Большинство систем в Интернет устанавливают окно равным по умолчанию 4096 байтам. Иногда размер окна равен 8192 или 16384 байтам.

Сообщение TCP

Блок данных TCP принято называть сообщением или сегментом. Оба эти термина вполне корректны и широко употребляются в литературе, посвященной Интернет. Мы, однако, по причинам, которые обсудим ниже, на протяжении всей книги будем употреблять термин *«сегмента.* TCP рассматривает свои данные в качестве однородного, неделимого потока. Тем не менее для доставки данных он вынужден использовать IP-датаграммы. Прикладной программе, однако, совершенно не обязательно знать, что ее поток данных переносят датаграммы. TCP делает этот процесс прозрачным для всех приложений, работающих через него.

Где бы вы не встретили термин *«сообщением,* относящийся к TCP, можете смело подставлять туда термин *«сегмента.* Почему? Потому, что каждое сообщение TCP, доставленное датаграммой протокола Интернет, является на самом деле ТСР-сегментом. Сегмент TCP состоит из TCP-заголовка, TCP-опций и данных, переносимых сегментом. На рис. 5.6 изображена структура сегмента TCP. Несмотря на то, что заголовок показан состоящим как бы из нескольких уровней, на самом деле он является последовательным потоком данных, длиной как минимум в 20 байтов. В табл. 5.2 кратко описано назначение каждого поля заголовка TCP. Способы применения каждого поля будут обсуждены в следующих абзацах.

*Таблица.2. Назначение полей заголовка TCP*

|  |  |
| --- | --- |
| Поле заголовка | Назначение |
| Порт-передатчик | Обозначает порт протокола приложения-источ­ника данных. |
| Порт-приемник | Обозначает порт протокола приложения-полу­чателя данных. |
| Номер последовательности | Определяет первый байт данных в области данных сегмента TCP. |
| Номер подтверждения | Определяет следующий байт данных, который приемник рассчитывает получить из входного потока. |
| Длина заголовка | Длина TCP-заголовка, измеренная в 32-разряд­ных словах. |
| Флаг URG | Если установлен, извещает принимающий модуль TCP о том, что в сегменте находятся данные для неотложной обработки. |
| Флаг АСК | Указание принимающему модулю TCP на то, что поле номер подтверждения содержит соответ­ствующие данные. |
| Флаг PSH | Требование принимающему модулю TCP передать данные приложению-получателю немедленно. |
| Флаг RST | Запрос принимающему модулю TCP сбросить соединение. |
| Флаг SYN | Запрос принимающему модулю TCP синхрони­зировать номера последовательности. |
| Флаг FIN | Сообщение принимающему модулю TCP об окон­чании передачи. |
| Размер окна | Сообщение принимающему модулю TCP о коли­честве байтов, которое способен принять модуль-передатчик. |
| Контрольная сумма TCP | Служит для обнаружения поврежденных при передаче данных. |
| Указатель на неотложные данные | Указывает на последний байт данных, требую­щих неотложной обработки, находящихся в об­ласти данных сегмента TCP. |
| Опции | Обычно используются совместно с опцией макси­мальная длина сегмента (MSS). |

Установление TCP-соединения

Чтобы обеспечить надежную передачу данных, а также правильный порядок следования сегментов, TCP использует сообщения-подтверждения о доставке. Для выполнения этих задач требуется какой-либо способ идентифицировать передаваемые данные. Также сеть должна уметь синхронизировать передавае­мые между обоими сторонами данные. Другими словами, каждая сторона должна знать, когда можно начинать передачу. Сторонам также следует иметь представление, как обозначить тот или иной сегмент. Предположим, что модуль TCP принял поврежденный пакет. Он должен иметь возможность известить модуль-передатчик о том, какой именно пакет данных нужно повторить. Перед установлением соединения обе стороны должны договориться о таком, понятном им обеим, способе отметить каждый пакет.

Определенная система обмена сообщениями-подтверждениями соединения TCP является частью более общего процесса синхронизации обмена данными. Не будет системы — соединение может не состояться. В следующих абзацах описываются поля TCP-заголовка, служащие для обмена подтверждениями. Для установления и прекращения соединения, а также для отправки и получения подтверждений TCP заголовок имеет поля «номер последовательности», «номер подтверждения» и поля флагов. Каждый раз, желая что-нибудь передать по протоколу TCP, программа-приложение запрашивает модуль TCP установить соединение. Модуль TCP в свою очередь шлет сообщение TCP с установленным флагом SYN (синхронизации) удаленному порту, с которым программа-клиент хочет установить соединение.

Флаг синхронизации указывает принимающей стороне (серверу, например), что программа-клиент желает установить соединение. Вместе с флагом SYN, сооб­щение TCP несет в себе 32-битный *номер последовательности,* размещенный модулем TCP в поле «номер последовательности». TCP-модуль сервера отвечает сегментом TCP с установленными флагом подтверждения (АСК) и *номером подтверждения.* Для того чтобы разобраться в процедуре установления соеди­нения TCP, необходимо понимать смысл номеров последовательности и подтвер­ждения. В следующем разделе процедура установления соединения и смысл соответствующих номеров изучаются более подробно.

Что такое начальный номер последовательности?

Мы уже отмечали выше, что обе стороны соединения TCP должны идентифи­цировать информацию в потоке данных для того, чтобы корректно слать и принимать подтверждения о доставке. Номер последовательности — это то, как TCP обозначает данные. Сетевые компьютеры пользуются разнообразными методами для выбора начального номера последовательности (для наших целей, однако, эти методы несущественны). Вы можете рассматривать начальный номер последовательности как случайное число. Значение начального номера последо­вательности не играет никакой роли.

Начальный номер последовательности представляет собой число, пересылаемое модулем-передатчиком TCP модулю-приемнику TCP. Передавая начальный номер, TCP-модуль как бы сообщает корреспонденту: «Эй, я хочу установить с тобой соединение. Нумерация данных в потоке от меня начинается с этого числа».

Сервер-приемник сообщения, получив запрос на установление соединения, тоже не сидит сложа руки. Он посылает обратно сообщение, содержащее его собст­венный начальный номер. Модуль TCP генерирует какое-либо число, совершен­но не зависящее от числа, посланного TCP-клиентом. Другими словами, сервер как бы сообщает: «Привет! Я получил твой запрос на TCP-соединение и высы­лаю номер, с которого начнется нумерация моих собственных данных».

Все TCP соединения являются дуплексными. Данные следуют в обоих направ­лениях одновременно. Поток данных в одном направлении совершенно не зависит от потока данных в противоположном. В силу дуплексной природы соединения оба модуля TCP должны учитывать и нумеровать данные в каждом направлении по-разному, а значит, обрабатывать два начальных номера после­довательности.

Подтверждение доставки данных

В TCP-заголовке самого первого, начального, сообщения-ответа сервера модуль TCP устанавливает два флага: синхронизации (SYN), чтобы известить модуль TCP клиента о том, что в сообщении содержится начальный номер последова­тельности сервера, и подтверждения (АСК), чтобы заставить клиента изучить содержимое поля «подтверждение».

TCP-модуль сервера использует номер последовательности, принятый от клиен­та, чтобы сконструировать на его основе собственный номер подтверждения. Номер подтверждения *всегда* указывает на номер сообщения, которое сервер рассчитывает получить следующим. Таким образом, в начальном сообщении-от­вете сервера содержится номер последовательности клиента, увеличенный на единицу.

Предположим, например, что модуль TCP клиента, запрашивающего соедине­ние, шлет номер последовательности, равный 1000. В ответ от сервера он получит начальное сообщение с числом 1001, установленным в поле подтверждения. Для модуля-клиента это выглядит, как будто сервер сказал: «Кстати, следующее сообщение, которое я от тебя ожидаю, должно иметь номер 1001».

Соединение установлено!

До передачи любых данных модуль TCP клиента, запросивший соединение, должен подтвердить начальное сообщение-ответ, пришедшее от модуля TCP сервера. То есть, когда модуль TCP клиента получает начальное сообщение-ответ от сервера, он должен посылать «подтверждение подтверждения». (На самом деле клиент подтверждает запрос сервера на синхронизацию обмена.)

Сообщение, посланное TCP-модулем клиента, тоже будет содержать установ­ленный флаг «подтверждение». В поле «номер подтверждения» TCP-модуль клиента размещает начальный номер последовательности, принятый от сервера, увеличенный на единицу. (Теперь TCP-модуль клиента не устанавливает флаг синхронизации, так как обе стороны соединения уже синхронизировались, то есть договорились о начальных номерах своих последовательностей.) Другими словами, между TCP-модулями происходит обмен данными, состоящий из *трех стадий:*

1. TCP-модуль клиента пытается установить TCP-соединение, посылая запрос на синхронизацию, содержащий среди прочего начальный номер последова­тельности.

2. TCP-модуль сервера подтверждает прием запроса на установление соедине­ния и в свою очередь шлет клиенту запрос на синхронизацию с собственным начальным номером последовательности.

3. TCP-модуль клиента подтверждает прием запроса сервера на синхрони­зацию.

Что такое номер последовательности?

Номер последовательности однозначно идентифицирует байт данных в потоке данных TCP. Как следует из названия, номера последовательности последова­тельны, то есть следуют один за другим. Однако номера последовательности разных соединений TCP необязательно начинаются с одного и того же числа. Номер последовательности, устанавливаемый в каждом сегменте TCP, иденти­фицирует первый байт в сообщении. То есть является смещением относительно начала потока данных. Номер последовательности — то же самое, что счетчик переданных байтов. Он как бы говорит: «Первый байт этого пакета — это номер байта (номер последовательности) в потоке данных».

Предположим, что программа-клиент желает передать 2000 байтов данных посредством TCP. Предположим, что произведена синхронизация и следующий номер последовательности будет равен 1251. Предположим так же, что длина данных в сегменте равна 500 байтам. Произойдут следующие события:

1. Модуль TCP клиента передает сегмент TCP, содержащий байты с 1 по 500. Номер последовательности, записанный в поле «номер последовательности» сегмента, равен 1251.

2. Модуль TCP клиента передает сегмент TCP, содержащий байты с 501 по 1000. Номер последовательности равен 1751.

3. Следующий сегмент, посланный модулем TCP клиента, содержит байты с 1001 по 1500 с номером последовательности, равным 2251.

4. Наконец, модуль TCP клиента передает данные с 1501 байта по 2000 и устанавливает номер последовательности равным 2751.

TCP-модуль сервера в нашем примере шлет следующие сообщения:

1. Приняв первый сегмент от клиента, TCP-модуль сервера отвечает пакетом-подтверждением с номером подтверждения 1751. Сервер как бы говорит:

«Эй, я принял твои данные, и следующий сегмент, который я надеюсь получить, должен содержать номер последовательности 1751».

2. Приняв второй сегмент, TCP-модуль сервера шлет номер подтверждения, установленный в 2251.

3. Приняв третий сегмент, TCP-модуль сервера шлет номер подтверждения, равный 2751.

4. Приняв четвертый сегмент, TCP-модуль сервера шлет номер подтверждения, равный 3251. (В данный момент TCP-модуль сервера не знает, что передача данных окончена — TCP-модуль клиента еще не известил об этом.)

Дуплексные сетевые службы

Как уже отмечалось, соединение TCP является *дуплексным.* Это значит, что данные следуют одновременно в обоих направлениях. Данные, следующие в одном направлении, совершенно не зависят от данных, следующих в противо­положном. Поскольку обмен данными TCP является дуплексным, TCP-модулям необходимо подсчитывать получаемые и передаваемые данные раздельно, и номера последовательности для обоих потоков будут разными.

Если вышесказанное не произвело на вас впечатления, давайте рассмотрим поток данных с точки зрения одной из сторон соединения. Предположим, что мы рассматриваем поток данных со стороны TCP-модуля клиента. С точки зрения TCP-модуля клиента номер последовательности отсчитывает или идентифици­рует данные, посылаемые клиентом в сторону TCP-модуля сервера. Номер подтверждения в сегментах, посланных TCP-модулем клиента, с его точки зрения идентифицирует данные, принятые от TCP-модуля сервера. На рис. 5.7 изображен поток данных и номера последовательности так, как они выглядят со стопоны TCP-модуля клиента.

*Рис. .7. Идентификация данных и их поток с точки зрения TCP-модуля клиента 137*

Окончание соединения TCP

Соединение TCP заканчивается обменом пакетами, состоящего из *двух стадий.* Каждая из сторон, как сервер, так и клиент, может предложить другой закончить соединение. Для этого сторона-инициатор обмена высылает пакет с установлен­ным флагом «окончание обмена» (FIN). В силу дуплексной природы протокола TCP оба потока данных независимы, и должны быть завершены по отдельности. Если вам приходилось программировать в UNIX, последнее утверждение преды­дущего абзаца может показаться подозрительным. Когда в UNIX закрывается соединение, дальнейший обмен по нему становится невозможен. Соединение TCP работает по-другому. Даже после закрытия соединения (завершения пере­дачи данных) одной из сторон соединения она в состоянии продолжать прием данных от другой стороны соединения.

Мысль принимать данные после закрытия соединения кажется странной на первый взгляд. На самом деле, установленный в пакете флаг FIN является сигналом, означающим, что одна сторона прекратила *передачу* данных. Приход сообщения-подтверждения от другой стороны означает, что обе стороны догово­рились прекратить обмен данными в одном направлении. Начиная с этого момента одна из сторон соединения будет молчаливо принимать данные, не делая никаких замечаний по их поводу, а другая молчаливо посылать, не ожидая никаких комментариев. Окончание (закрытие) TCP-соединения — двухступен­чатый процесс. Одна сторона выполняет *активное* закрытие, а другая — *пассивное.* Закрытие активно, если вызвано по инициативе данной стороны, и, наоборот, пассивно, если вызвано противоположной стороной соединения. Сто­рона, первой высылающая пакет с установленным флагом «окончание соедине­ния», является активной. Как правило, модуль TCP, принявший пакет с установленным флагом «окончание соединения», инициирует пассивное окон­чание соединения. Это просто значит, что пассивная сторона также посылает сообщение с установленным флагом окончания. Другими словами, сторона, принявшая сообщение об окончании первой, отвечает: «Хорошо, если тебе нечего больше послать, то и мне тоже нечего послать тебе». После того как обе стороны выслали друг другу сообщения об окончании и получили подтверждения о доставке этих сообщений, соединение TCP считается действительно закончен­ным (закрытым).

Что такое закрытие «наполовину»?

Вы уже знаете, что в силу дуплексной природы TCP-соединения, в то время как поток данных в одну из сторон закончен, он может сохраняться в обратном направлении. Такое окончание лишь одного потока называется закрытием «наполовину» (half-close). Лишь очень немногие приложения TCP нуждаются или используют закрытие «наполовину». Однако если в ваши планы входит разработка приложений Интернет, эта возможность может когда-нибудь и пригодиться.

Что такое заголовок TCP?

Из рис. 5.8 видно, что структура TCP-заголовка намного сложнее, нежели у UDP. В следующих абзацах изучается назначение полей, составляющих TCP-заголовок.


### Рис. 8. Структура сегмента (сообщения) TCP

Порт источника и порт получателя

16-битные поля источника и получателя однозначно определяют посылающие и Принимающие данные приложения или прикладные протоколы. Номера портов источника и получателя в совокупности с IP-адресами сетевых компьютеров (в IP-заголовке) однозначно идентифицируют любое TCP-соединение. Каждая из сторон TCP-соединения называется *сокетом* (socket). *\*

Номер последовательности

32-битное поле номера последовательности обозначает первый байт данных из области данных сегмента TCP. Оно соответствует смещению этого байта отно­сительно начала потока данных. Каждый байт в потоке данных может быть идентифицирован при помощи номера последовательности.

Номер подтверждения

32-битное поле номера подтверждения обозначает байт данных, который прини­мающая сторона рассчитывает получить следующим в потоке данных. Напри­мер, если последний принятый байт имел номер 500, модуль TCP установит номер подтверждения равным 501.

Длина заголовка

Как и в заголовке IP, поле длины заголовка TCP состоит из четырех битов, обозначающих длину заголовка, измеренную в 32-битных словах. Как и заголо­вок IP, заголовок TCP обычно имеет длину в 20 байтов. Область данных начинается сразу после заголовка TCP. Таким образом, модуль TCP определяет место, где начинаются данные и кончается заголовок, просто прибавляя поле «длина заголовка», умноженное на четыре байта (32 бита), к первому байту сегмента данных.

Флаги

Заголовок TCP содержит шесть однобитных полей флагов. С тремя из них мы уже встречались. Это флаги синхронизации (SYN), подтверждения (АСК) и флаг окончания соединения (FIN). Следующие абзацы описывают оставшиеся три.

**Флаг URG**

Данный флаг сообщает принимающему модулю TCP о том, что. указатель на данные, требующие немедленной обработки, в поле «неотложные данные» установлен, то есть указывает на них. (Модуль TCP должен обработать неот­ложные данные до того, как обрабатывать что-либо еще.)

**Флаг АСК**

Установленный флаг сообщает принимающему модулю TCP, что поле «номер подтверждения» содержит правильный номер подтверждения. Вы знаете, что данный флаг служит обеспечению надежной передачи данных.

Флаг PSH

Установленный флаг PUSH требует от принимающего модуля TCP вытолкнуть (push), то есть немедленно выслать принятый сегмент данных приложению-по­лучателю. Как правило, модуль TCP буферизует принимаемые данные. То есть он не доставляет каждый сегмент по отдельности, а ждет, пока его буфер наполнится, а затем доставляет все принятые сегменты за один раз. Флаг PSH запрещает размещать сегменты данных в буфере. Telnet, например, устанавли­вает этот флаг. Нажатия на клавиши пользователем незамедлительно попадают на сервер Telnet, с которым он работает. Такое поведение устраняет возможные задержки в выдаче эха от сервера — большинство пользователей Telnet желают сразу видеть на экране то, что они печатают.

Флаг RST

Данный флаг запрашивает у принимающего модуля TCP сброс соединения. TCP устанавливает флаг RST, если с соединением случилась какая-либо проблема. Большинство приложений просто прекращает работу, приняв этот флаг. Флаг RST может применяться в сложных разработках для контроля повреждений в сети, сбоев в работе оборудования и сетевых программ.

Флаг SYN

Флаг SYN просит принимающий модуль TCP синхронизировать номера после­довательности. Вы уже знаете, что этот флаг используется на этапе установления соединения, чтобы сообщить приемнику TCP о том, что источник готовится передать новый поток данных.

Флаг FIN

Флаг сообщает принимающему модулю TCP о том, что источник закончил передавать данные. Вы знаете, что этот флаг заканчивает соединение только в том направлении, в каком был послан. Чтобы закончить соединение полностью, принимающий модуль TCP должен также послать сообщение с установленным флагом FIN.

Размер окна

16-битное поле «размер окна» сообщает принимающему модулю TCP количество байтов, которое собирается принять передатчик. Вы уже знаете, что TCP использует скользящие окна переменной длины для увеличения производитель­ности и оптимизации пропускной способности сети. Значение данного поля определяет размер этого скользящего окна. Как правило, оно равняется несколь­ким тысячам байтов.

Контрольная сумма TCP

Как и в случае UDP, 16-битное поле контрольной суммы TCP содержит сумму, вычисленную по области данных. Протокол требует от передатчика, чтобы он включил вычисленную контрольную сумму в поле, а от приемника — чтобы он вычислил ее повторно и сравнил результаты.

***Примечание:*** Контрольные суммы UDP и TCP вычисляются похожим образом. Одна­ко в случае UDP включать контрольную сумму в датаграмму не обяза­тельно. Напротив, протокол TCP обязывает вставлять контрольную сумму в каждый переданный сегмент данных.

Указатель неотложных данных

16-битное поле указателя определяет положение байта данных в области данных сегмента TCP. Указатель и флаг неотложных данных извещают принимающий модуль TCP о том, что некоторые, требующие немедленной обработки данные находятся в сегменте и указывают модулю на них. Никто, однако, так и не дал исчерпывающего ответа на вопрос, что же такое неотложные данные. Никто не определил ответственность модуля TCP за их обработку. Даже вопрос, на что же, собственно, обращает внимание указатель на неотложные данные, требует более основательного обсуждения.

Дуглас Камер во втором издании классического труда *«Межсетевое взаимодей­ствие сетей на базе TCP/IP\** (Internetworking with TCP/IP, Volume 1, Prentice Hall, 1991) обсуждает поле «неотложные данные» в разделе 12.12, < *Данные вне основной полосы пропусканиям* (Out of Band Data). С другой стороны, Ричард Стивене в разделе 20.8 своей замечательной книги *^TCP/IP в иллюстрациях^* (TCP/IP Illustrated, Volume 1, Prentice Hall, 1994) пишет следующее:

*4... во многих сетевых приложениях данные для неотложной обработ­ки TCP неправильно называются сданными вне основной полосы пропусканиям,*

Стивене полагает, что эти приложения совершенно неоправданно смешивают разные понятия: данные вне полосы пропускания и данные для неотложной об­работки. И он пытается объяснить причину возникновения такой ситуации:

*^Путаница в связи с неотложными данными TCP и данными вне основной полосы пропускания возникает из-за того, что предпочита­емый большинством интерфейс прикладного программирования (API) сам по себе относит данные ^для неотложной обработким к категории данных <вне основного диапазонам,*

Относительно точного местоположения данных для неотложной обработки Стивене делает следующий комментарий:

*<Идет продолжительный спор о том, должен ли указатель неотлож­ных данных указывать на последний байт этих данных или на байт, следующий за последним. Первоначальный стандарт TCP позволял толковать это двояким образом, однако RFC, посвященный обяза­тельным рекомендациям для сетевых компьютеров, вносит ясность в это дело, утверждая, что указатель все-таки указывает на последний байт данных для немедленной обработки.*

*Проблема, однако, в том, что большинство реализации сетевых операционных систем (т. н. производные операционной системы Бер­кли) продолжают использовать неверное представление. Получает­ся, что вполне лояльное по отношению к стандарту TCP сетевое приложение не сможет работать с большинством остальных сете­вых компьютеров^.*

Стивене и Камер согласны друг с другом в том, что указатель неотложных данных указывает на **их** *последний* байт. Также Стивене подчеркивает, что не существует способа, позволяющего определить местонахождение *начала* неот­ложных данных. Практически единогласно утверждается, что приложению Telnet необходимо передавать неотложные данные, поскольку ему приходится обрабатывать разного рода управляющие последовательности. В настоящее время очевидно, что от употребления режима неотложных данных TCP следует воздерживаться. Нужно либо быть уверенным, что все программы, работающие с вашим приложением, ведут себя корректно, либо вообще не использовать этот режим при работе в Интернет.

Так же как и у IP, заголовок TCP содержит необязательное поле «опции» (options). В ходе установления соединения модули TCP договариваются о максимальной длине сегмента (MSS) и устанавливают соответствующую опцию. Смысл максимальной длины сегмента тот же, что и у максимальной длины передаваемого блока (MTU) физического уровня сети. Максимальная длина сегмента определяет максимальный размер сегмента, который может быть передан по соединению TCP. TCP оптимизирует пропускную способность сети, увеличивая ее производительность. Опция максимальной длины сегмента поз­воляет воспользоваться самым большим размером блока данных, который еще можно передать. Опция MSS устанавливается только в тех сообщениях, в которых уже установлен флаг SYN. Однако опция MSS не является предметом обсуждения между обоими модулями. Каждый из модулей TCP просто сообщает другому тот MSS, который он в состоянии принять. Если модуль TCP по каким-либо причинам не передает MSS, его партнер считает, что нужно пользо­ваться MSS, равным по умолчанию 536 байтам.

Что такое инкапсуляция?

Как уже замечалось, разработка программного обеспечения для Интернет в целом ненамного отличается от разработки обычного программного обеспечения. Многоуровневая структура сети и наличие протоколов TCP/IP позволяет скрыть от разработчика ненужные детали функционирования сетевых программ. Сетевые протоколы берут большинство рутинной работы на себя. Вся сложность процесса доставки данных в Интернет заключена в достаточно простой сетевой интерфейс. Прикладные данные просто передаются из программы в стек прото­колов, этот протокол передает **их** следующему и т. д. Вы знаете, что весьма полезно представлять, как происходит весь этот процесс. Однако для того, чтобы программировать приложения, достаточно знать детали, касающиеся только взаимодействия программы с верхними протоколами стека TCP/IP, перенося­щими данные, то есть интерфейс сетевого программирования.

Эта и две предыдущие главы познакомили вас с различными уровнями сети на базе TCP/IP. В этих главах также описываются интерфейсы между сетевыми уровнями и протоколами TCP/IP. Процесс перемещения данных сквозь стек протоколов на самом деле представляет собой **их** инкапсуляцию. Инкапсуляция данных — это их форматирование таким образом, чтобы они удовлетворяли тому или иному протоколу. По мере прохождения сквозь стек протоколов данные инкапсулируются в тот формат, с которым умеет работать очередной сетевой уровень. На рис. 5.9 процесс прохождения данных сквозь стек прото­колов изображен целиком.

Разработка структуры и функций приложения Интернет практически не отли­чается от разработки любого другого приложения. Решив передать информацию по Интернет, вы сначала решаете, каким протоколом удобнее воспользоваться. Данные будут инкапсулированы в выбранный протокол, отвечающий вашим требованиям. Для правильного выбора протокола необходимо знать, какие из них есть в вашем распоряжении и какими особенностями они обладают. Чтобы правильно инкапсулировать данные, необходимо представлять структуру дан­ных выбранного протокола. Надеемся, что три последние прочитанные главы помогут вам понять этот процесс и все детали, необходимые для выполнения поставленных задач.

*Рис. .9- Инкапсуляция данных по мере их прохождения сквозь стек протоколов*

Что такое прикладной уровень?

Вам наверное уже понятно, что именно происходит на прикладном сетевом уровне. Он включает в себя все, касающееся непосредственно решаемой при­кладной задачи. Другими словами, в качестве программиста приложений Ин-тернет вы, разрабатывая программу, вместе с тем конструируете и прикладной уровень.

Являясь прикладным программистом, вы, по определению, занимаетесь разра­боткой прикладных программ. Конструирование программы тесно связано с выполняемыми функциями. Например, коль скоро вам необходимо написать сетевую программу, вам требуется обменяться данными с другим приложением Интернет. Для успешной разработки приложения Интернет необходимо знать, как принимать и посылать сетевые данные. Приступая к написанию, задайте себе вопрос: «Каким образом моя программа будет обмениваться данными с Интернет?» И вы уже знаете ответ: просто посылая информацию *вниз* по *стеку протоколов.*

Ваша ответственность за доставку данных заканчивается, как только прикладная программа передаст их низлежащему протоколу. Далее каждый последующий уровень в стеке протоколов, сквозь который пойдут данные, будет выполнять свою собственную функцию: определять адрес, маршрут и транспортировать данные по Интернет. Чтобы задействовать определенный протокол, необходимо для начала знать, какие из них доступны в вашей системе. Также необходимо знать, где именно в стеке они находятся и понимать выполняемые ими функции. Как правило, прикладные программы взаимодействуют с протоколами UDP и TCP.