Транспортные протоколы

План

- **Функции транспортного уровня**
- **Концепция портов. Мультиплексирование и демультиплексирование**
- **Протокол UDP**
- **Протокол TCP**

Транспортный уровень

- Транспортный уровень (*Transport Layer*) – обеспечивает приложениям или верхним уровням стека – прикладному, представления и сеансовому – передачу данных с той степенью надежности, которая им требуется. Модель OSI определяет пять классов транспортного сервиса от низшего класса о до высшего класса 4. Эти виды сервиса отличаются качеством предоставляемых услуг: срочностью, возможностью восстановления прерванной связи, наличием средств мультиплексирования нескольких соединений между различными прикладными протоколами через общий транспортный протокол, а главное – способностью к обнаружению и исправлению ошибок передачи, таких как искажение, потеря и дублирование пакетов.

- Выбор класса сервиса транспортного уровня определяется, с одной стороны, тем, в какой степени задача обеспечения надежности решается самими приложениями и протоколами более высоких, чем транспортный, уровней. С другой стороны, этот выбор зависит от того, насколько надежной является система транспортировки данных в сети, обеспечиваемая уровнями, расположенными ниже транспортного, - сетевым, канальным и физическим.

- Так, если качество каналов передачи связи очень высокое и вероятность возникновения ошибок, не обнаруженных протоколами более низких уровней, невелика, то разумно воспользоваться одним из облегченных сервисов транспортного уровня, не обремененных многочисленными проверками, квитированием и другими приемами повышения надежности. Если же транспортные средства нижних уровней очень ненадежны, то целесообразно обратиться к наиболее развитому сервису транспортного уровня, который работает, используя максимум средств для обнаружения и устранения ошибок, включая предварительное установление логического соединения, контроль доставки сообщений по контрольным суммам и циклической нумерации пакетов, установление тайм-аутов доставки и т.п.

Функции транспортного уровня

- преобразование транспортного адреса в сетевой;
- межоконечное мультиплексирование транспортных соединений в сетевые;
- установление и разрыв транспортных соединений;
- межоконечное упорядочение блоков данных по отдельным соединениям;
- межоконечное обнаружение ошибок и необходимый контроль качества услуг;
- межоконечное восстановление после ошибок;
- межоконечное сегментирование, объединение и сцепление;
- межоконечное управление потом данных по отдельным соединениям;
- супервизорные функции;
- передача срочных транспортных сервисных блоков данных.

Транспортный уровень стека TCP/IP может предоставлять вышележащему уровню два типа сервиса:

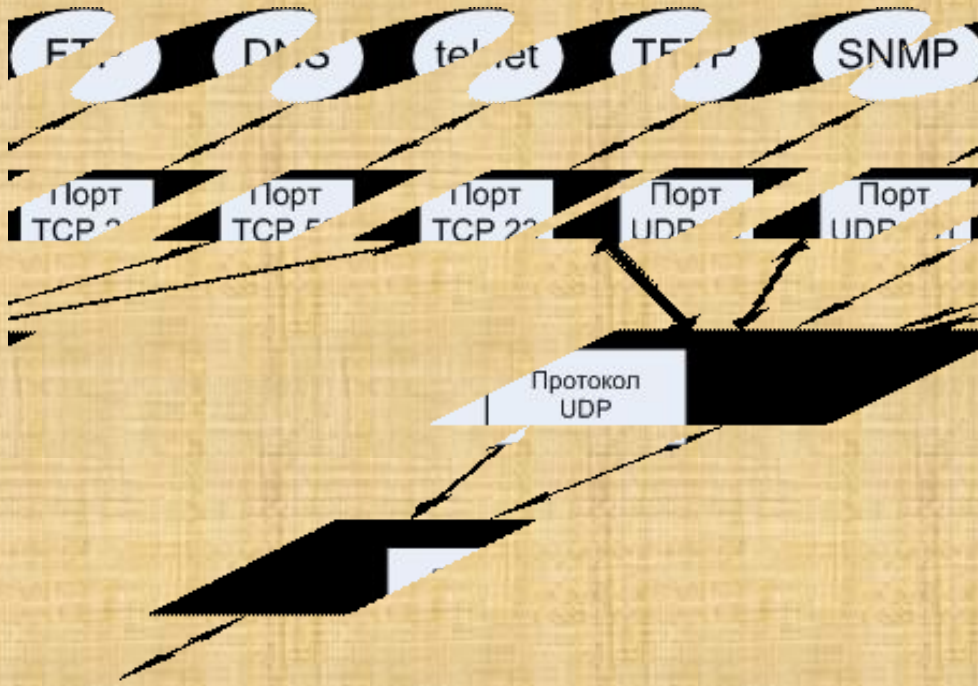
- гарантированную доставку обеспечивает протокол управления передачей *(Transmission Control Protocol, TCP*);
- доставку по возможности, или с максимальными усилиями, обеспечивает протокол пользовательских дейтаграмм (*User Datagram Protocol, UDP*).

Мультиплексирование и демультиплексирование

- Каждый компьютер может выполнять несколько процессов, более того, прикладной процесс тоже может иметь несколько точек входа, выступающих в качестве адреса назначения для пакетов данных. Поэтому после того, как пакет средствами протокола IP доставлен на сетевой интерфейс компьютера-получателя, данные необходимо переправить конкретному процессу получателю.

- Существует и обратная задача: пакеты, которые отправляют в сеть разные приложения, работающие на одном конечном узле, обрабатываются общим для них протоколом IP. Следовательно, в стеке должно быть предусмотрено средство «сбора» пакетов от разных приложений для передачи протоколу IP. Эту работу выполняют протоколы TCP и UDP.

- Процедура приема данных протоколами TCP и UDP, поступающих от нескольких различных прикладных служб, называется ***мультиплексированием.***
- Обратная процедура – процедура распределения протоколами TCP и UDP поступающих от сетевого уровня пакетов между набором высокоуровневых служб – называется ***демультиплексированием***



- Протоколы TCP и UDP ведут для каждого приложения две очереди:

- очередь пакетов, поступающих к данному приложению из сети;

- очередь пакетов, отправляемых данным приложением в сеть.

Пакеты, поступающие на транспортный уровень, организуются операционной системой в виде множества очередей к точкам входа различных прикладных процессов. В терминологии TCP/IP такие системные очереди называются портами, причем входная и выходная очереди одного приложения рассматриваются как один порт. Для однозначной идентификации портов им присваивают номера. Номера портов используются для адресации приложений.

Протокол UDP

- Протокол UDP, являясь дейтаграммным протоколом, реализует сервис по возможности, то есть не гарантирует доставку своих сообщений, а, следовательно, никоим образом не компенсирует ненадежность дейтаграммного протокола IP.
- Единица данных протокола UDP называется UDP-пакетом или пользовательской дейтаграммой (*user datagram*).

- Каждая дейтаграмма переносит отдельное пользовательское сообщение. Это приводит к естественному ограничению: длина дейтаграммы UDP не может превышать длины поля данных протокола IP, которое, в свою очередь, ограничено размером кадра технологии нижнего уровня.
- Поэтому если UDP-буфер переполняется, то данные приложения отбрасываются. Заголовок UDP-пакета, состоящий из четырех 2-байтовых полей, содержит поля *порт источника, порт получателя, длина UDP и контрольная сумма.*

- Рассмотрим, как протокол UDP решает задачу демультиплексирования. Казалось бы, для этой цели достаточно использовать номера портов. Кадры, несущие UDP-дейтаграммы, прибывают на сетевой интерфейс хоста, последовательно обрабатываются протоколами стека и, наконец, поступают в распоряжение протокола UDP. UDP извлекает из заголовка номер порта назначения и передает данные на соответствующий порт соответствующему приложению, то есть выполняет демультиплексирование.

Протокол TCP

Протокол TCP (*Transmission Control Protocol*) обеспечивает надежную транспортировку данных между прикладными процессами путем установления логического соединения.

Установление соединения происходит в три шага:

1. Клиент, запрашивающий соединение, отправляет серверу пакет, указывающий номер порта, который клиент желает использовать, а также код (определенное число) ISN (Initial Sequence number).
2. Сервер отвечает пакетом, содержащий ISN сервера, а также ISN клиента, увеличенный на 1.
3. Клиент должен подтвердить установление соединения, вернув ISN сервера, увеличенный на 1.

Трехступенчатое открытие соединения устанавливает номер порта, а также ISN клиента и сервера. Каждый, отправляемый TCP – пакет содержит номера TCP – портов отправителя и получателя, номер фрагмента для сообщений, разбитых на меньшие части, а также контрольную сумму, позволяющую убедиться, что при передачи не произошло ошибок.

- Информация, поступающая к протоколу TCP от протоколов более высокого уровня, рассматривается протоколом TCP как неструктурированный поток байтов.
- Поступающие данные буферизируются средствами TCP.
- Для передачи на сетевой уровень из буфера «вырезается» некоторая непрерывная часть данных, которая называется сегментом.
- Сегмент состоит из фиксированного 20-байтного заголовка (плюс необязательная часть), за которой могут следовать байты данных. Размер сегментов определяется программным обеспечением TCP. Оно может объединять в один сегмент данные, полученные в результате нескольких операций записи, или, наоборот, распределять результата одной записи между несколькими сегментами.

- Размер сегментов ограничен двумя пределами. Во-первых, каждый сегмент, включая TCP-заголовок, должен помещаться 65 515-байтное поле полезной нагрузки IP-пакета. Во-вторых, в каждой сети есть максимальная единица передачи *(MTU, Maximum Transfer Unit)*, и каждый сегмент должен помещаться в MTU. На практике размер максимальной единицы передачи составляет обычно 1500 байт (что соответствует размеру поля полезной нагрузки Ethernet), и таким образом определяется верхний предел размера сегмента.

Формат TCP-пакета

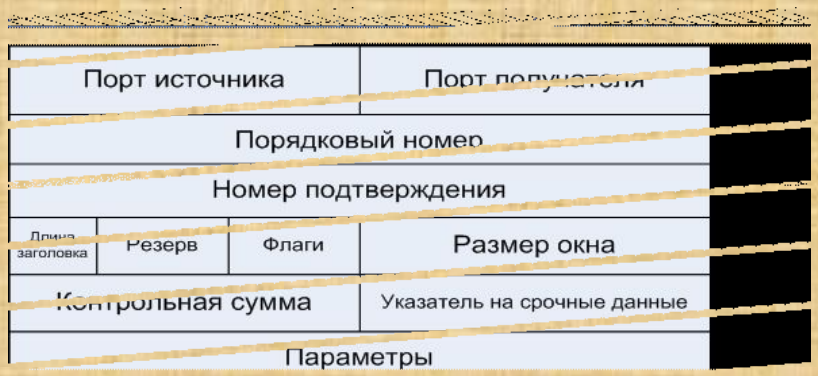


Рис. Формат заголовка сегмента TCP

- Заголовок TCP-сегмента содержит значительно больше полей, чем заголовок UDP, что отражает более развитые возможности первого протокола.

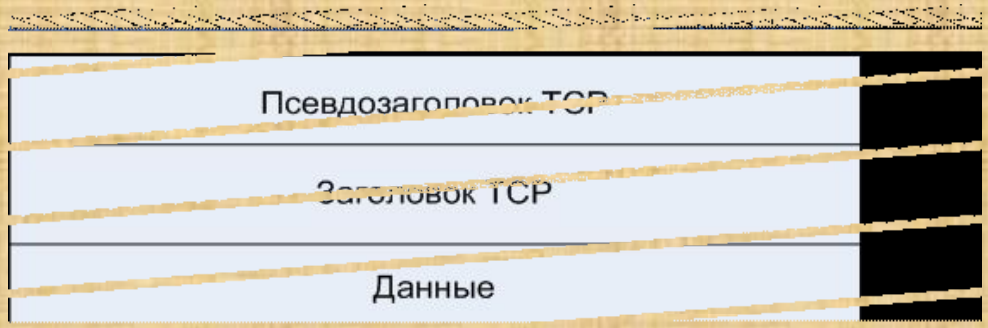


Рис. Структура пакета TCP при вычислении контрольной суммы

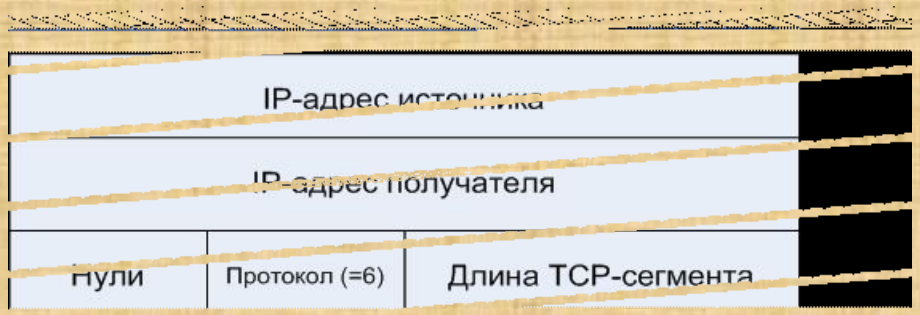


Рис. Структура псевдозаголовка пакета TCP.

Сессии TCP

- Основным отличием TCP от UDP является то, что на протокол TCP возложена дополнительная задача – обеспечить надежную доставку сообщений, используя в качестве основы ненадежный дейтаграммный протокол IP.
- Установленные на конечных узлах протокольные модули TCP решают задачу обеспечения надежного обмена данными путем установления между собой логических соединений. Благодаря логическому соединению TCP следит, чтобы передаваемые сегменты не были потеряны, не были продублированы и пришли к получателю в том порядке, в котором были отправлены.
- При установлении логического соединения модули TCP договариваются между собой о параметрах процедуры обмена данными. В протоколе TCP каждая сторона соединения посылает противоположной стороне следующие параметры:
- максимальный размер сегмента, который она готова принять;
- максимальный объем данных (возможно несколько сегментов), которые она разрешает другой стороне передавать в свою сторону. Даже если та еще не получила квитанцию на предыдущую порцию данных (размер окна);
- начальный порядковый номер байта, с которого она начинает отсчет потока данных в рамках данного соединения.

Чтобы установить соединение, одна сторона (например, сервер) пассивно ожидает входящего соединения, выполняя примитивы LISTEN (объявление о желании принять соединение) и ACCEPT (блокирование звонящего до получения попытки соединения), либо указывая конкретный источник, либо не указывая его.

Другая сторона (например, клиент) выполняет примитив CONNECT (активно пытаться установить соединение), указывая IP-адрес и порт, с которым он хочет установить соединение, максимальный размер TCP-сегмента и, по желанию, некоторые данные пользователя (например, пароль) Примитив CONNECT посылает TCP-сегмент с установленным битом SYN и сброшенным битом ACK и ждет ответа.

Когда этот сегмент прибывает в пункт назначения, TCP-сущность проверяет, выполнил ли какой-нибудь процесс примитив LISTEN, указав в качестве параметра тот же порт, который содержится в поле *порт получателя*. Если такого процесса нет, она отвечает отправкой сегмента с установленным битом RST для отказа от соединения.