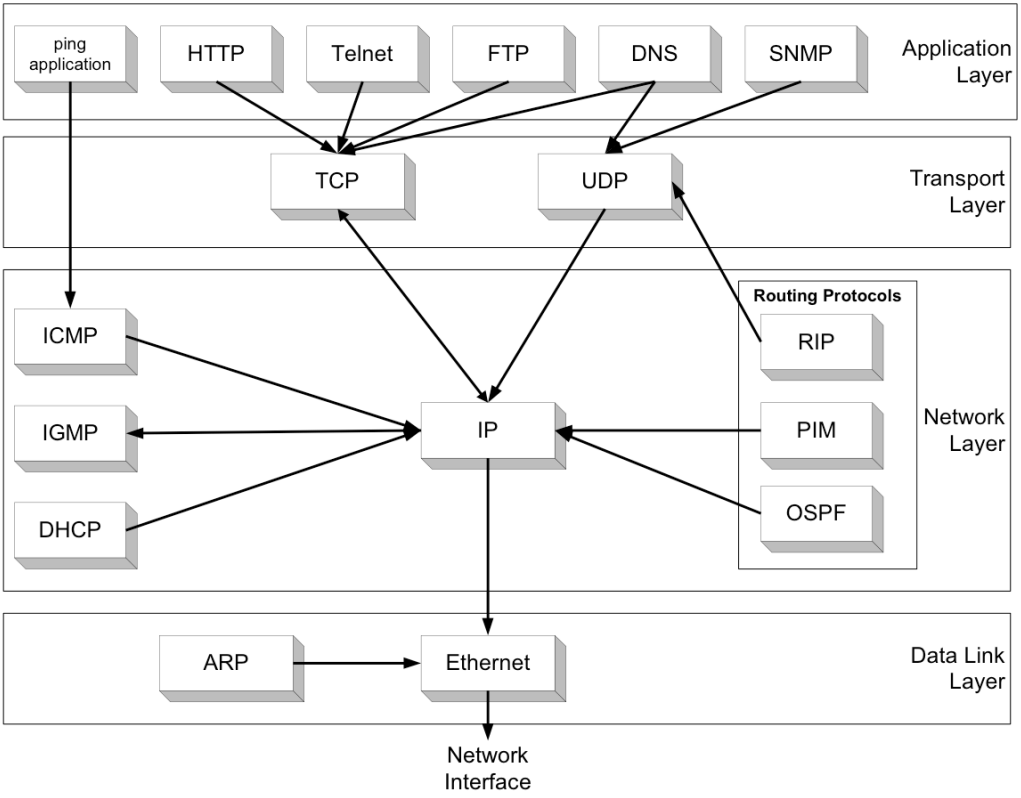
Транспортный уровень TCP/IP.



Протоколы транспортного уровня устанавливаются на конечных узлах и включают протоколы:

- пользовательских дейтаграмм (User Datagram Protocol, UDP), RFC 768;
- управления передачей (Transmission Control Protocol, TCP), RFC 793;
- передача с управлением потоков (Stream Control Transmission Protocol, SCTP), RFC 4960
- управления перегрузками (Datagram Congestion Control Protocol, DCCP), RFC 4340.

Сравнение возможностей протоколов транспортного уровня.

Параметр	UDP Light	UDP	DCCP	TCP	SCTP
Установка соединения	Нет	Нет	Нет	Да	Да
Надежная передача	Нет	Нет	Нет	Да	Да
Сохранение границ сообщения	Да	Да	Да	Нет	Да
Упорядоченная доставка	Нет	Нет	Нет	Да	Да
Неупорядоченная доставка	Да	Да	Да	Нет	Да
Контрольные суммы данных	Нет	Да	Да	Да	Да
Размер контрольной суммы (бит)	16	16	16	16	32
Путь MTU	Нет	Нет	Нет	Да	Да
Управление накоплением	Нет	Нет	Да	Да	Да
Многопоточность	Нет	Нет	Нет	Нет	Да
Поддержка множественных интерфейсов	Нет	Нет	Нет	Нет	Да
Связка потоков	Нет	Нет	Нет	Да	Да
Номер протокола в IP	136	17	33	6	132

1. Порты.

Задачей уровня сетевого взаимодействия IP, является передача данных между сетевыми интерфейсами в составной сети, главная задача протоколов транспортного уровня TCP и UDP заключается в передаче данных между прикладными процессами, выполняющимися на компьютерах в сети.

Каждый компьютер может выполнять несколько процессов, более того, даже отдельный прикладной процесс может иметь несколько точек входа, выступающих в качестве адресов назначения для пакетов данных. Поэтому после доставки данных на сетевой интерфейс компьютера-получателя, данные необходимо переправить конкретному процессу-получателю.

Мультиплексирование. Данные, генерируемые разными приложениями, работающими на одном конечном узле, должны быть переданы общему для всех них протокольному модулю IP для последующей отправки в сеть. Эту работу, называемую мультиплексированием, выполняют протоколы TCP и UDP (см. рис.).

Демультиплексирование. Существует и обратная задача. Процедура распределения протоколами TCP и UDP поступающих от сетевого уровня пакетов между прикладными процессами называется демультиплексированием.

Порты. Протоколы TCP и UDP ведут для каждого приложения две системные очереди: очередь данных, отправляемых этим приложением в сеть, и очередь данных, поступающих к приложению из сети. Такие системные очереди называются портами, причем входная и

выходная очереди одного приложения рассматриваются как один порт. Для идентификации портов им присваивают номера от 0 до 65535.

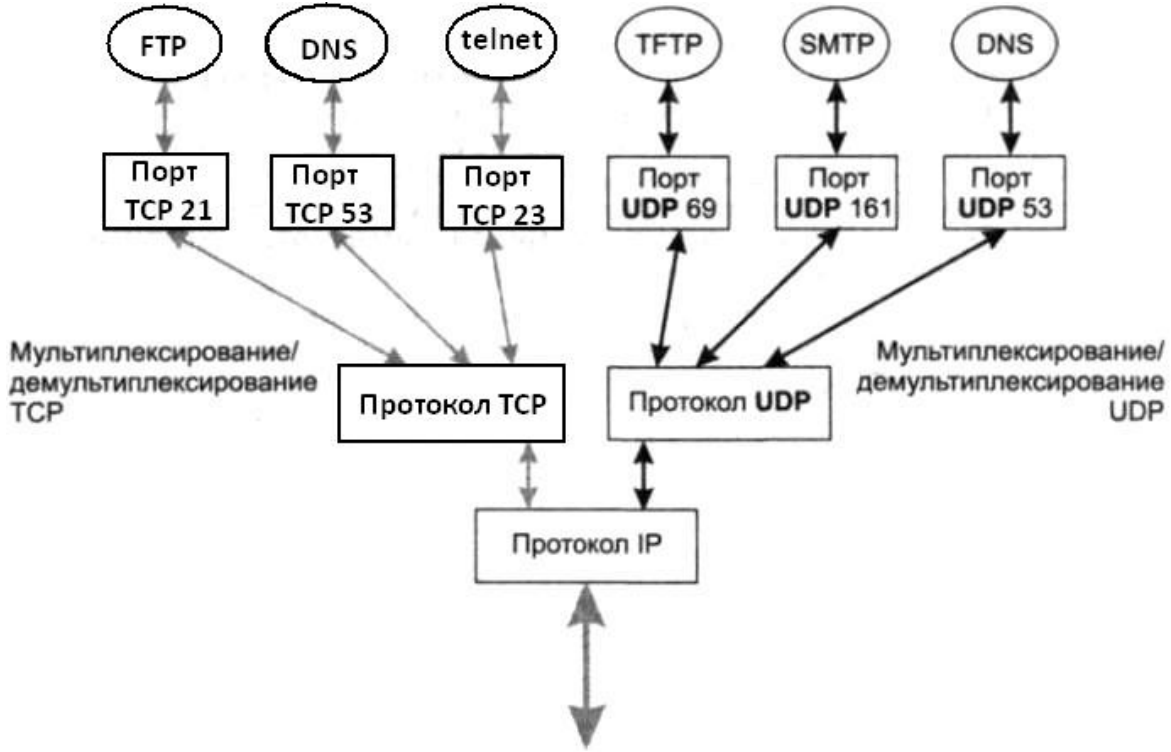


Рис. Мультиплексирование и демультиплексирование на транспортном уровне

1.1. Список свободных-занятых портов.

Специальный порт 0. Зарезервирован Internet Assigned Numbers Authority (IANA) и указывается как порт отправителя, если обратной передачи данных не предусмотрено.

Известные порты 1-1023 (2^10-1). Если процесс представляет собой популярную системную службу (FTP, telnet, HTTP, TFTP, DNS и т. п.), то за ним закрепляются стандартные назначенные номера портов, называемые также хорошо известными (well-known) портами. Эти порты зарезервированы и зарегистрированы в IANA и в стандартах Интернета (RFC 1700, RFC 3232). Назначенные номера из диапазона от 1 до 1023 являются уникальными в пределах Интернета и закрепляются за приложениями централизованно.

Регистрируемые порты 1024-49151 (2^15+2^14-1). Зарегистрированные в организации IANA службы для частных программ (Skype, ICQ)**.** Для тех приложений, которые еще не стали столь распространенными, номера портов регистрируются в IANA разработчиками этих приложений. Для Linux перечень портов для общеизвестных и зарегистрированных служб хранится в файле /etc/services, для Windows в файле %SystemRoot%\system32\drivers\etc\services,.

Динамические локальные порты 49152-65535 (2^16-1). Чаще всего выбирается OS случайно. Могут использоваться любыми программами для любых целей или операционной системой в ответ на поступление запроса от приложения.

Порт можно записать вместе с IP-адресом или DNS через двоеточие:

127.0.0.1:8080 192.169.1.10:51076 example.org:42240

На каждом компьютере операционная система ведет список занятых и свободных номеров портов. При поступлении запроса от приложения, выполняемого на данном компьютере, операционная система выделяет ему первый свободный номер. В дальнейшем все сетевые приложения должны адресоваться к данному приложению с указанием назначенного ему динамического номера порта. Такие порты называют **динамическими**. После того как приложение завершит работу, его номер возвращается в список свободных и может быть назначен другому приложению. Динамические номера являются уникальными в пределах каждого компьютера.

1.2. Порты TCP и порты UDP.

В том и другом случаях это могут быть как назначенные, так и динамические номера. Диапазоны чисел, из которых выделяются номера TCP- и UDP-портов, совпадают: от 0 до 1023 для назначенных и от 1024 до 65 535 для динамических. Однако никакой зависимости между назначенными номерами TCP- и UDP-портов нет.

Даже если номера TCP- и UDP-портов совпадают, они могут идентифицировать разные приложения. В некоторых случаях, когда приложение может обращаться по выбору к протоколу TCP или UDP (например, таким приложением является DNS), ему, исходя из удобства запоминания, назначаются совпадающие номера TCP- и UDP-портов (в данном примере — это порт 53).

Службы в SCTP и DCCP обычно используют номера портов, соответствующие их реализациям в TCP и UDP.

2. Сокеты.

Стандартные назначенные номера портов уникально идентифицируют тип приложения (FTP, или HTTP, или DNS и т. д.), однако они не могут использоваться для однозначной идентификации прикладных процессов, связанных с каждым из этих типов приложений.

Пусть, например, на одном хосте запущены две копии DNS-сервера — DNS-сервер 1, DNS-сервер 2 (см. рис.). Каждый из этих DNS-серверов имеет хорошо известный UDP-порт 53. Какому из этих серверов нужно было бы направить запрос клиента, если бы в DNS-запросе в качестве идентификатора сервера был указан только номером порта?

Чтобы снять неоднозначность в идентификации приложений, разные копии связываются с разными IP-адресами. Для этого сетевой интерфейс компьютера, на котором выполняется несколько копий приложения, должен иметь соответствующее число IP-адресов - на рисунке это IP1 и IP2. Во всех IP-пакетах, направляемых DNS-серверу 1, в качестве IP-адреса указывается IP1, а DNS-серверу 2 — адрес IP2. Поэтому показанный на рисунке пакет, в поле данных которого содержится UDP-дейтаграмма с указанным номером порта 53, а в поле заголовка задан адрес IP2, буден направлен однозначно определенному адресату — DNS-серверу 2.

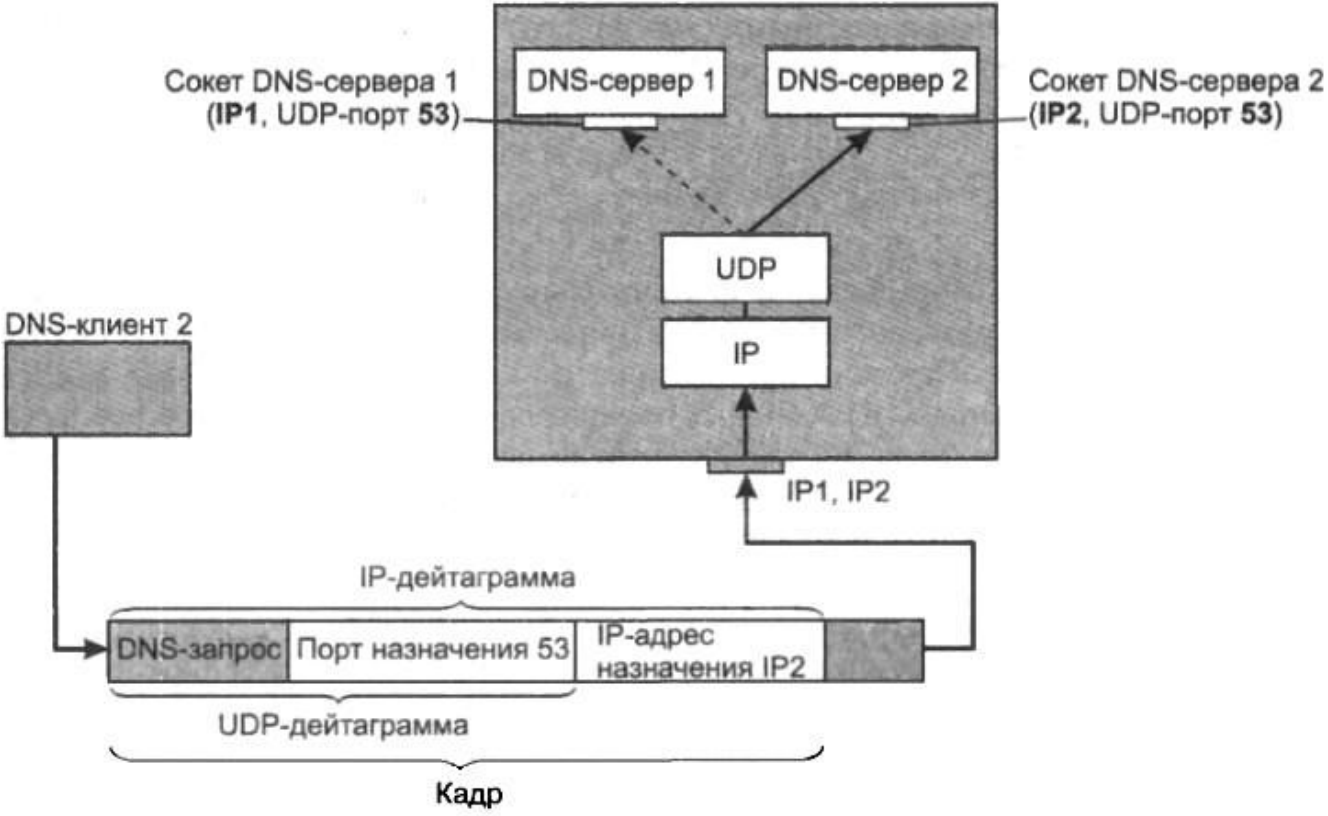


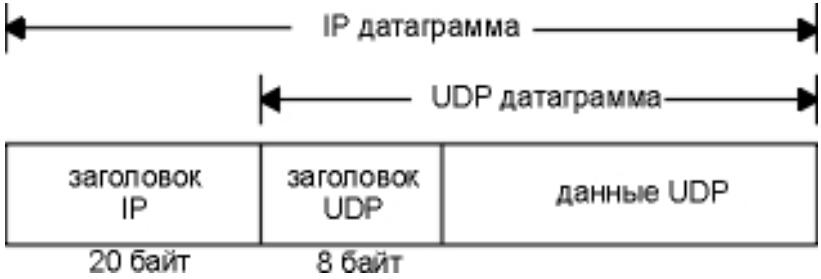
Рис. Демультиплексирование протокола UDP на основе сокетов.

3. Протокол UDP.

Автором протокола UDP является Дэвид П. Рид, UDP был создан в 1980 году.

3.1. UDP дейтаграмма и UDP заголовок.

UDP сообщения инкапсулируются и передаются в IP дейтаграмме как протокол =17, см. рис. 3.

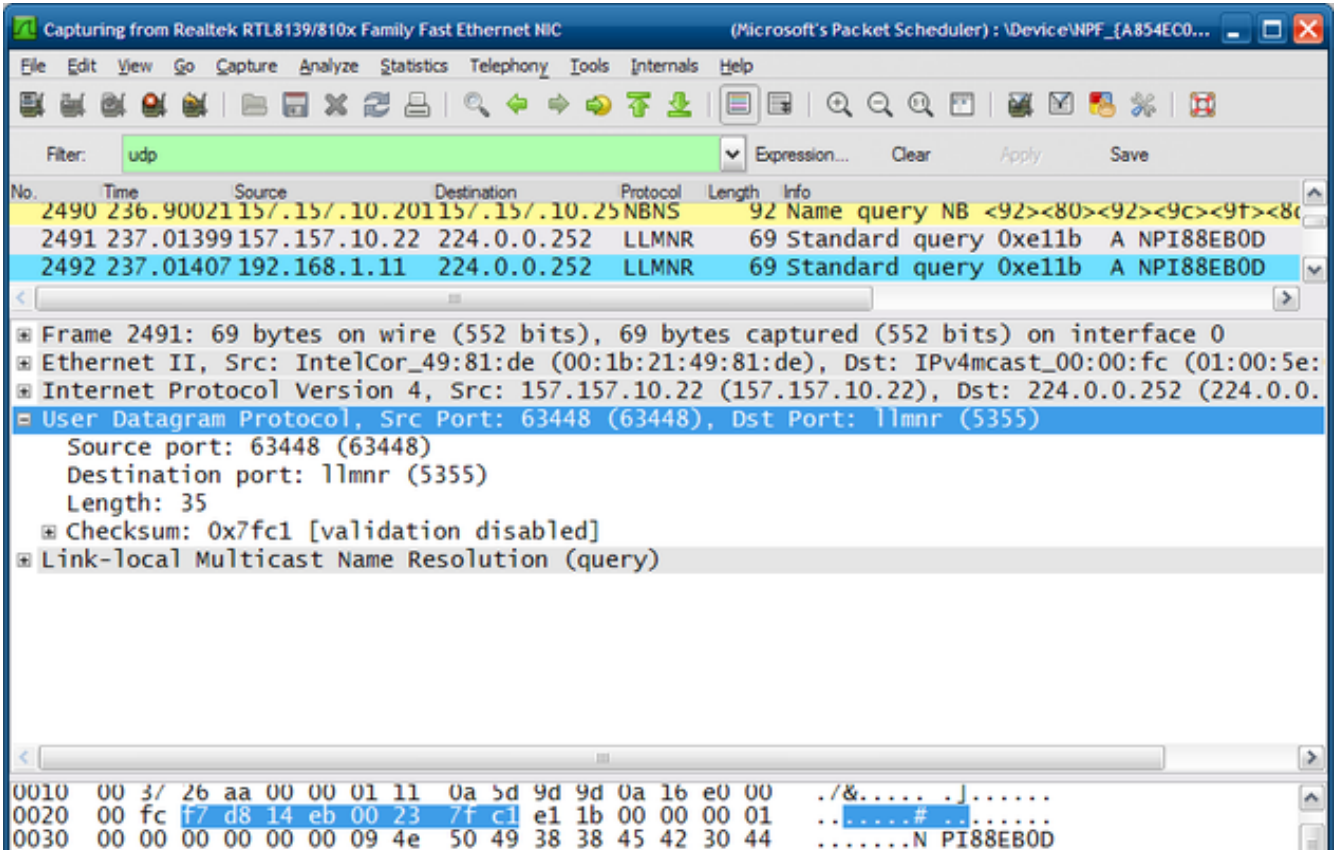


3.1.1. Формат UDP заголовка.

Octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00	**Порт отправителя**																**Порт получателя**															
04	**Длина дейтаграммы**																**Контрольная сумма**															
08-	**Данные**																															

- **Порт отправителя** - в этом поле указывается номер порта отправителя, задает порт, на который при необходимости будет посылаться ответ. Если ответа не ожидаем, то значение должно быть равным 0. Если хостом-источником является клиент, то номер порта будет, скорее всего, эфемерным. Если источником является сервер, то его порт будет одним из "хорошо известных".
- **Порт получателя** - содержит порт получателя, если клиент - хост-получатель, то номер порта эфемерный, иначе (сервер - получатель) это "хорошо известный порт".
- **Длина дейтаграммы** - поле, задающее длину всей дейтаграммы (заголовка и данных) в байтах. Минимальная длина равна длине заголовка - 8 байт. Теоретически, максимальный размер поля - 65535 байт для UDP-дейтаграммы (8 байт на заголовок и 65527 на данные), но, фактический предел для длины данных при использовании IPv4 - 65507 (помимо 8 байт на UDP-заголовок требуется еще 20 на IP-заголовок).
- **Контрольная сумма** - рассчитывается отправителем, проверяется получателем и позволяет определить любые изменения в UDP заголовке или в данных, расчитывается аналогично IP Checksum. Контрольная сумма в целях экономии ресурсов может и не расчитываться.
- **Данные** – поле переменной длины от 0 до 65507 байт.

3.1.2. Пример структуры заголовка UDP в сетевом анализатора Wireshark.



3.2. Преимущества и недостатки UDP.

UDP в отличии от TCP используется для быстрой, но, **ненадежной** транспортировки данных между TCP/IP хостами.

UDP протокол обеспечивает обслуживание без установления соединения, таким образом UDP не гарантирует доставку или проверку последовательности для любой дейтаграммы. Хост, который нуждается в надежной связи должен использовать либо протокол TCP либо программу, которая будет сама следить за последовательностью дейтаграмм и подтверждать приём каждого пакета.

Чувствительные ко времени приложения (например, видеоданные) часто используют UDP, так как предпочтительнее сбросить пакеты, чем ждать задержавшиеся пакеты, что может оказаться невозможным в системах реального времени. Также потеря одного или нескольких кадров, при передаче видеоданных по UDP, не так критична, в отличии от передачи бинарных файлов, где потеря одно пакета может привести к искажению всего файла.

Еще одним преимуществом протокола UDP являются **малые накладные расходы** при передаче, так как длина заголовка UDP составляет лишь 4 байта, а у TCP протокола - 20 байт.

UDP	TCP
Отправка *отдельных пакетов* без постоянного соединения	Отправка *потока данных* с установлением виртуального сеанса связи
Ненадежный (возможны потери, ошибки и дублирование)	*Надежный* (контролирует полную доставку всех данных потока)
Неупорядоченность (пакеты могут быть получены не в том порядке, в каком они были отправлены)	*Упорядоченность* (поступившие пакеты упорядочиваются перед передачей приложению)
Легковесность (небольшой служебный трафик)	*Тяжеловесность* (дополнительный трафик для установления соединения и контроля доставки пакетов)
Быстрая доставка данных	Ожидание доставки всех отправленных данных приводит к задержкам
Может создавать *широковещательную* рассылку	*Широковещательная* рассылка невозможна

3.3. Работа протокола UDP.

При работе на хосте-отправителе данные от приложений поступают протоколу UDP через порт в виде сообщений (см. рис.).

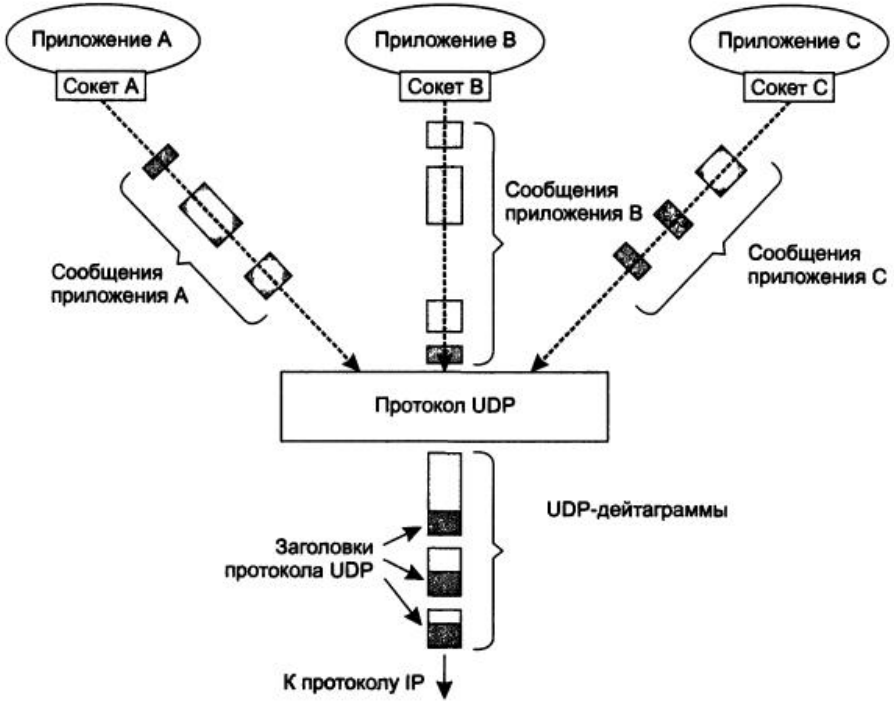


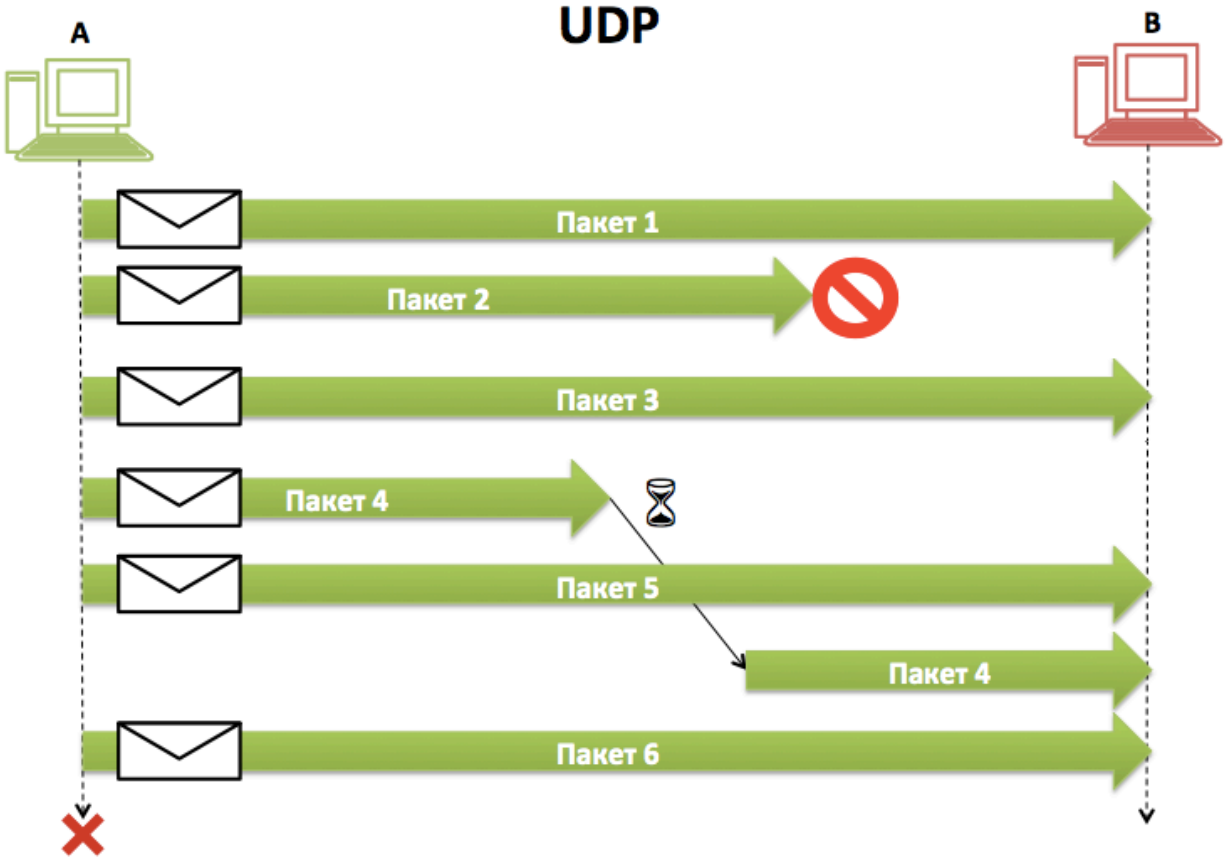
Рис. Работа протокола UDP на хосте-отправителе

Протокол UDP добавляет к каждому отдельному сообщению свой 8-байтный заголовок, формируя из этих сообщений собственные протокольные единицы, называемые UDP-дейтаграммами, и передает их нижележащему протоколу IP. В этом и заключаются его функции по мультиплексированию данных.

Каждая дейтаграмма переносит отдельное пользовательское сообщение. Сообщения могут иметь различную длину, не превышающую однако длину поля данных протокола IP, которое, в свою очередь, ограничено размером кадра технологии нижнего уровня. Поэтому если буфер UDP переполняется, то сообщение приложения отбрасывается.

Протокол UDP не сложен, его функции сводятся к простой передаче данных между прикладным и сетевым уровнями, а также примитивному контролю искажений в передаваемых данных. При контроле искажений протокол UDP только диагностирует, но не исправляет ошибку. Если контрольная сумма показывает, что в поле данных UDP-дейтаграммы произошла ошибка, протокол UDP просто отбрасывает поврежденную дейтаграмму (никаких ICMP при этом не посылается).

Работая на хосте-получателе, протокол UDP принимает от протокола IP извлеченные из пакетов UDP-дейтаграммы. Полученные из IP-заголовка IP-адрес назначения и из UDP-заголовка номер порта используются для формирования UDP-сокета, однозначно идентифицирующего приложение, которому направлены данные. Протокол UDP освобождает дейтаграмму от UDP-заголовка. Полученное в результате сообщение он передает приложению на соответствующий UDP-сокет. Таким образом, протокол UDP выполняет демультиплексирование на основе сокетов.


UDP
A
B
Пакет 1
Пакет 2
Пакет 3
Пакет 4
Пакет 5
Пакет 4
Пакет 6

3.4. Номера портов UDP.

Каждый порт UDP идентифицируется зарезервированным или известным номером порта. В следующей таблице показано **начало списка известных номеров портов UDP**.

Порт/протокол	Описание	Использование
0/TCP,UDP	резерв (допустимо использование в качестве значения порта источника, если отправляющий процесс не ожидает ответных сообщений)	Официально
1/TCP,UDP	TCPMUX (TCP Port Service Multiplexer) — для обслуживания нескольких служб через один TCP-порт	Официально
2/TCP,UDP	COMPRESSNET, процесс управления	Официально
3/TCP,UDP	COMPRESSNET, процесс сжатия	Официально
5/TCP,UDP	RJE (Remote Job Entry) — обслуживает отправку файлов и вывод отчётов при работе рабочей станции с мейнфреймами	Официально
7/TCP,UDP	ECHO — предназначен для тестирования связи путём отправки данных на сервер и получения от него их же в неизменном виде	Официально
9/TCP,UDP	DISCARD — предназначен для тестирования связи путём отправки данных на сервер, который отбрасывает принятое, не отправляя никакого ответа	Официально
11/TCP,UDP	SYSTAT — выдаёт список активных пользователей в операционной системе	Официально
13/TCP,UDP	DAYTIME — предназначен для тестирования связи путём получения от сервера текущих даты и времени в текстовом виде	Официально
15/TCP	не назначено; ранее — NETSTAT	Неофициально
17/TCP,UDP	QOTD (Quote of the Day)	Официально
18/TCP,UDP	MSP (Message Send Protocol)	Официально
19/TCP,UDP	CHARGEN (Character Generator)	Официально

20/TCP	FTP-DATA — для передачи данных FTP	Официально
21/TCP	FTP — для передачи команд FTP	Официально
22/TCP,UDP	SSH (Secure SHell) — криптографический сетевой протокол для безопасной передачи данных	Официально
23/TCP,UDP	Telnet — применяется для передачи текстовых сообщений в незашифрованном виде	Официально
24/TCP,UDP	PRIV-MAIL — для использования в любых частных системах пересылки почтовых сообщений	Официально
25/TCP,UDP	SMTP (Simple Mail Transfer Protocol) — применяется для пересылки почтовых сообщений в виде незашифрованного текста	Официально

3.5. Контрольная сумма UDP.

Контрольная сумма UDP рассчитывается точно так же, как контрольная сумма IP заголовка: сумма 16-битных слов с дополнением до 1, хотя существуют и отличия:

- Во-первых, UDP дейтаграмма может состоять из нечетного количества байт, тогда как при расчете контрольной суммы складываются 16-битные слова. поэтому при необходимости для расчета контрольной суммы, в конец дейтаграммы добавляются нулевой байт заполнения, но, байт заполнения не передаётся.
- Во-вторых, если сумма не сгенерирована отправителем, то поле заполняется 0. Но, если окажется, что рассчитанная контрольная сумма действительно равна 0, она предаётся как число 65535 (все биты = 1).

Если отправитель рассчитал контрольную сумму, а получатель определил наличие ошибки, UDP дейтаграмма молча уничтожается, сообщение об ошибке не генерируется.

3.6. Фрагментация IP дейтаграмм и UDP.

IP дейтаграмма (**datagram**) это блок, который передается от одного конца IP уровня к другому концу IP уровня (перед фрагментацией и после повторной сборки на конечном узле), тогда как пакет (**packet**) это блок данных, который передается между IP уровнем и канальным уровнем.

Пакет может содержать полную IP дейтаграмму или всего лишь фрагмент IP дейтаграммы, см. рисунок. Однако, в IP заголовке хранится достаточно информации для того, чтобы датаграмма была собрана корректно. Здесь мы видим подтверждение того, что заголовок любого транспортного уровня присутствует только в первом фрагменте.

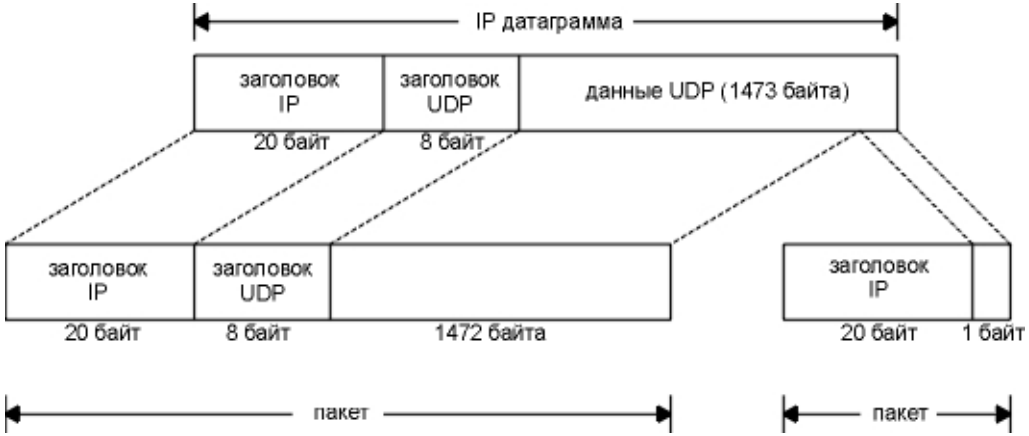


Рис. Пример IP фрагментации для UDP дейтаграмм.

3.7. Максимальный размер UDP дейтаграмм.

Теоретически максимальный размер IP дейтаграммы может составлять 65535 байт, что ограничивается 16-битным полем полной длины в IP заголовке.

В большинстве реализаций, однако, используются дейтаграммы значительно меньшего размера. Обычно играют роль четыре ограничения.

1. При длине IP заголовка равной 20 байтам и длине UDP заголовка равной 8 байтам в UDP дейтаграмме для пользовательских данных остается максимум 65507 байт.
2. Ограничение определяется реализацией ядра TCP/IP. Могут существовать характеристики реализации (или ошибки), которые ограничивают размер UDP дейтаграммы значением меньшим, чем 65507 байт.
3. Из того что IP может отправлять и принимать дейтаграммы определенного размера, не следует, что принимающее приложение готово прочитать дейтаграммы этого размера. Приложение может быть ограничено программным интерфейсом API. Сокеты API предоставляют функцию, которая может быть вызвана приложением, чтобы установить размер буферов ввода и вывода. Для UDP сокета этот размер напрямую связан с максимальным размером UDP дейтаграммы, которая может быть прочитана и записана UDP.

Ограничения ресурсов слабого хоста – память, CPU, производительность. В стандартах регламентируется, что любому хосту необходимо получать IP дейтаграммы размером по меньшей мере 576 байт.

В настоящее время современные системы предоставляют по умолчанию максимальный размер UDP дейтаграммы, которая может быть прочитана или записана, равный 8192 байтам. (Эта значение установлено в 8192, потому что именно столько по умолчанию читается и записывается системой NFS.)

Но, большинство приложений UDP разработаны таким образом, чтобы ограничивать свои приложения в размере 512 байт данных или меньше, чтобы уложиться в это ограничение. Например, RIP всегда посылает в дейтаграмме меньше чем 512 байт. Это же самое ограничение мы найдем и в других UDP приложениях: DNS, TFTP, BOOTP и SNMP.

3.8. Где применяется UDP.

- Интернет-телефония VoIP
- Интернет-телевидение IPTV
- трансляция аудио и видео онлайн (стримы и др.)
- онлайн-игры
- другие системы реального времени
- служебная информация и управление сетями, в том числе DNS (узнать IP по доменному имени), DHCP (выдача IP-адреса)

3.9. UDP Lite.

UDP Lite (облегченный UDP) — сеансовый протокол без установки соединения, весьма похожий на UDP. В отличие от UDP, в котором защищены контрольной суммой (checksum) или все пакеты или ни один из них, UDP Lite допускает возможность частичных контрольных сумм, которые покрывают только часть датаграммы, и таким образом возможна доставка частично поврежденных пакетов. Это было создано для мультимедийных протоколов (например, Voice over IP), у которых прием пакета с частично поврежденной полезной нагрузкой считается более предпочтительным вариантом, нежели не получить пакет вовсе.

UDP Lite использует другой номер протокола: 136.

Так как большинство современных протоколов канальных уровней защищают передаваемые данные достаточно надежным алгоритмом CRC и отбрасывают поврежденные фреймы, то эффективное использование UDP Lite требует от канального уровня «осведомленности» о передаваемой информации сетевого уровня. В связи с тем, что на данный момент не существует стеков TCP/IP, реализующих подобное междууровневое (или кросс-уровневое) взаимодействие, эффективное использование UDP Lite в настоящий момент требует специально модифицированных драйверов.

Поддержка UDP lite была добавлена в ядро Linux версии 2.6.20 и FreeBSD версии 10.1.

4. Упражнения.

1. Представьте себе Ethernet через который передаётся UDP дейтаграмма с 8192 байтами пользовательских данных. Сколько фрагментов будет передано и какова будет длина смещения для каждого фрагмента?
2. Продолжая предыдущий пример, представьте себе, что эти дейтаграммы затем передаются в SLIP канал с MTU равным 552 байта. Вам необходимо помнить, что количество данных в каждом фрагменте (все кроме IP заголовка) должно быть кратно 8 байтам. Сколько фрагментов передано и каковы смещение и длина каждого фрагмента?
3. Приложение, использующее UDP, посылает дейтаграмму, которая фрагментирована на 4 части. Представьте себе, что фрагменты 1 и 2 достигли своего пункта назначения, тогда как фрагменты 3 и 4 были потеряны. Приложение отрабатывает тайм-аут, а затем, через 10 секунд, повторяет передачу UDP дейтаграммы. Эта дейтаграмма фрагментируется точно так же, как и при первой передаче (то же смещение и та же длина). Теперь представьте, что фрагменты 1 и 2 потеряны, однако фрагменты 3 и 4 достигли своего пункта назначения. Таймер повторной сборки на принимающем хосте установлен в 60 секунд, поэтому когда фрагменты 3 и 4 прибыли на конечный пункт назначения, фрагменты 1 и 2 из первой передачи еще не были отброшены. Может ли получатель собрать IP дейтаграмму из этих четырех фрагментов?