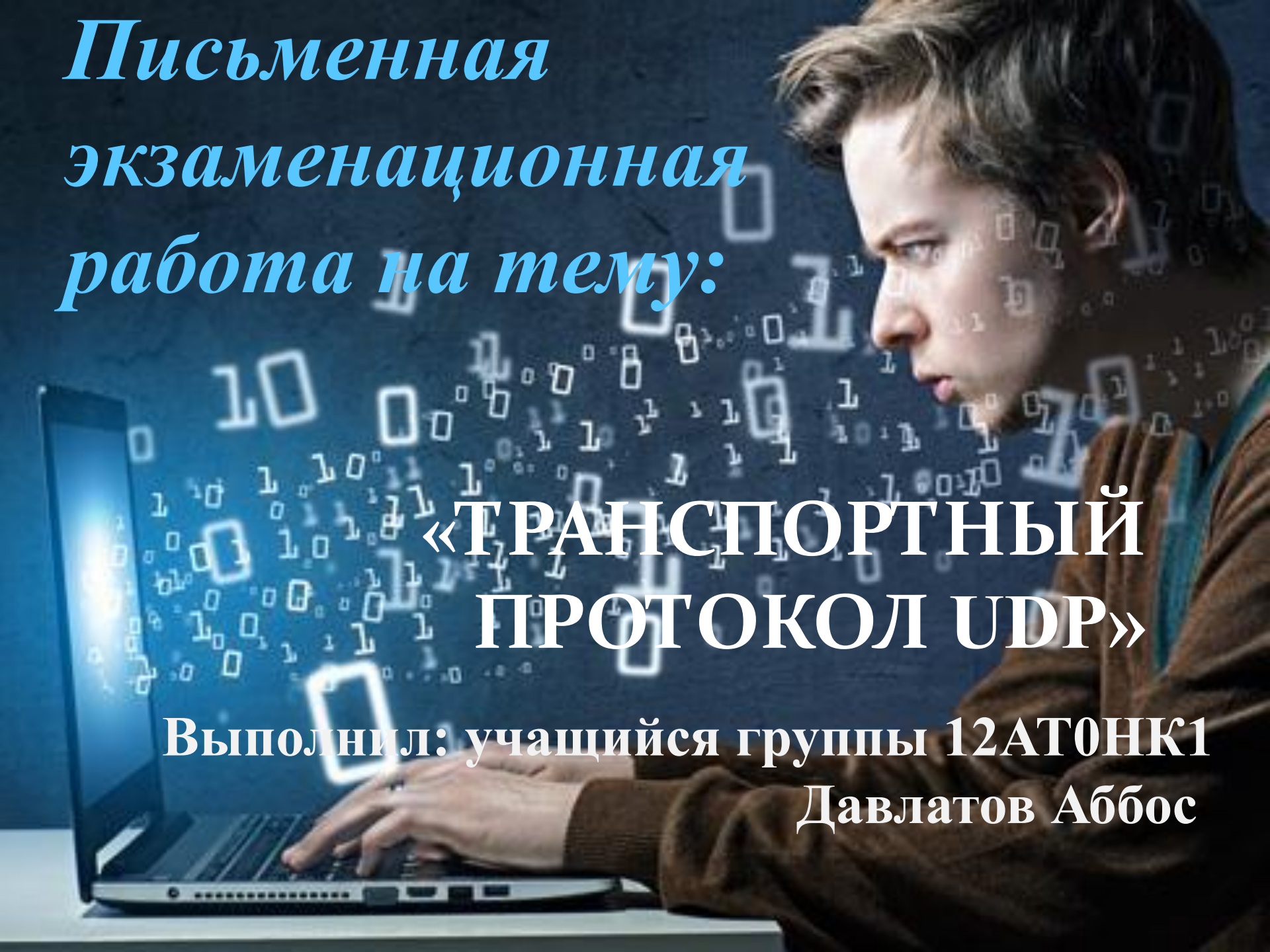


Письменная экзаменационная работа на тему:

«ТРАНСПОРТНЫЙ ПРОТОКОЛ UDP»

Выполнил: учащийся группы 12АТ0НК1
Давлатов Аббос

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

UDP (англ. User Datagram Protocol — протокол пользовательских датаграмм) — один из ключевых элементов TCP/IP, набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных. Протокол был разработан Дэвидом П. Ридом в 1980 году и официально определён в RFC 768.

2. ОСНОВНАЯ ЧАСТЬ

Номера портов (port numbers) указывают на отправляющий и принимающий процессы. TCP и UDP используют номер порта назначения для демультиплексирования данных, поступающих от IP. Так как IP осуществляет демультиплексирование входящих IP датаграмм для TCP и UDP (с использованием значения протокола в IP заголовке), TCP просматривает номера портов TCP, а UDP - номера портов UDP. Номера портов TCP независимы от номеров портов UDP.

Несмотря на подобную независимость, если зарезервированный сервис предоставляется обоими TCP и UDP, обычно выбирается один и тот же номер порта для обоих транспортных уровней. Это сделано для удобства, а не по требованию протоколов.

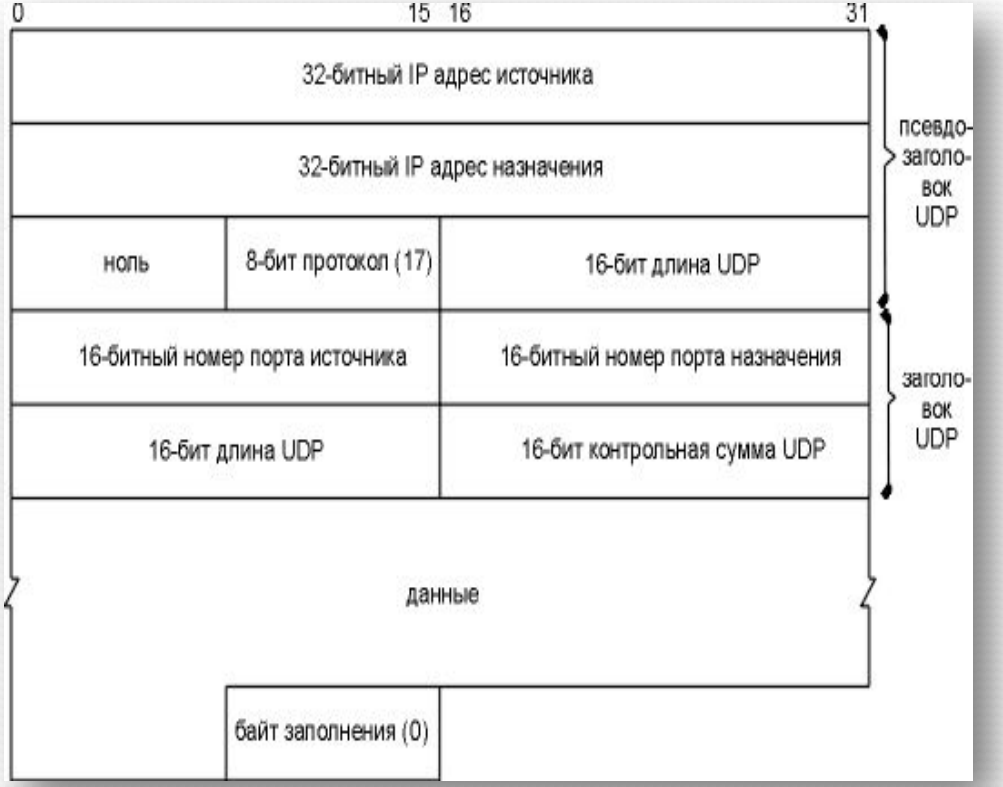
2.1. UDP

Номера портов (port numbers) указывают на отправляющий и принимающий процессы. TCP и UDP используют номер порта назначения для демультиплексирования данных, поступающих от IP. Так как IP осуществляет демультиплексирование входящих IP датаграмм для TCP и UDP (с использованием значения протокола в IP заголовке), TCP просматривает номера портов TCP, а UDP - номера портов UDP. Номера портов TCP независимы от номеров портов UDP.

Несмотря на подобную независимость, если зарезервированный сервис предоставляется обоими TCP и UDP, обычно выбирается один и тот же номер порта для обоих транспортных уровней. Это сделано для удобства, а не по требованию протоколов.

2.2. Контрольная сумма UDP

В UDP и TCP существуют 12-байтовые псевдозаголовки (в UDP датаграммах и TCP сегментах) только для расчета контрольной суммы. Псевдозаголовки содержат в себе определенные поля из IP заголовка. Все это сделано для двойной проверки того, что данные достигли того пункта назначения, которому предназначались (IP не принимает датаграммы, которые не адресованы для данного хоста, и не сможет передать UDP датаграммы, предназначенные для другого верхнего уровня). На рисунке 11.3 показан псевдозаголовок и UDP датаграмма.



На этом рисунке мы специально показали датаграмму с нечетной длиной, в этом случае требуется дополнительный байт для расчета контрольной суммы. Обратите внимание на то, что длина UDP датаграммы, при расчете контрольной суммы, появляется дважды.

Если рассчитанная контрольная сумма равна 0, она хранится как все единичные биты (65535), эти значения эквивалентны в арифметике с поразрядным дополнением (дополнение единицы) (ones-complement). Если переданная контрольная сумма равна 0, это означает, что отправитель не рассчитал контрольную сумму.

2.3. Простой пример

Мы воспользуемся программой sock, чтобы сгенерировать несколько UDP датаграмм, которые мы просмотрим с использованием tcpdump:

bsdi % **sock -v -u -i -n4 svr4 discard**
connected on 140.252.13.35.1108 to 140.252.13.34.9

bsdi % **sock -v -u -i -n4 -wo svr4 discard**
connected on 140.252.13.35.1110 to 140.252.13.34.9

2.4. Фрагментация IP

Как мы указали в разделе "MTU" главы 2, физический сетевой уровень обычно имеет ограничение максимального размера фрейма, который может быть передан. Когда IP уровень получает IP датаграмму, которую необходимо отправить, он определяет, на какой локальнй интерфейс отправляется датаграмма (или маршрутизируется), и запрашивает интерфейс, чтобы тот сообщил размер своего MTU. IP сравнивает MTU с размером датаграммы и, если необходимо, осуществляет фрагментацию. Фрагментация может быть осуществлена как на отправляющем хосте, так и на промежуточном маршрутизаторе.

Когда IP датаграмма фрагментирована, она не собирается вновь до тех пор, пока не достигнет конечного пункта назначения. (Для некоторых других сетевых протоколов процесс повторной сборки отличается от описанного выше, при этом повторная сборка осуществляется на маршрутизаторе следующей пересылки, а не в конечном пункте назначения.) На уровне IP сборка осуществляется в конечном пункте назначения. Это сделано для того, чтобы сделать фрагментацию и повторную сборку прозрачной для транспортных уровней (TCP и UDP), хотя это может вести к некоторой потере производительности. Существует вероятность, что фрагмент датаграммы будет снова фрагментирован (возможно даже несколько раз). Информации, которая содержится в IP заголовке вполне достаточно для фрагментации и повторной сборки.

2.5 ICMP ошибки о недоступности (требуется фрагментация)

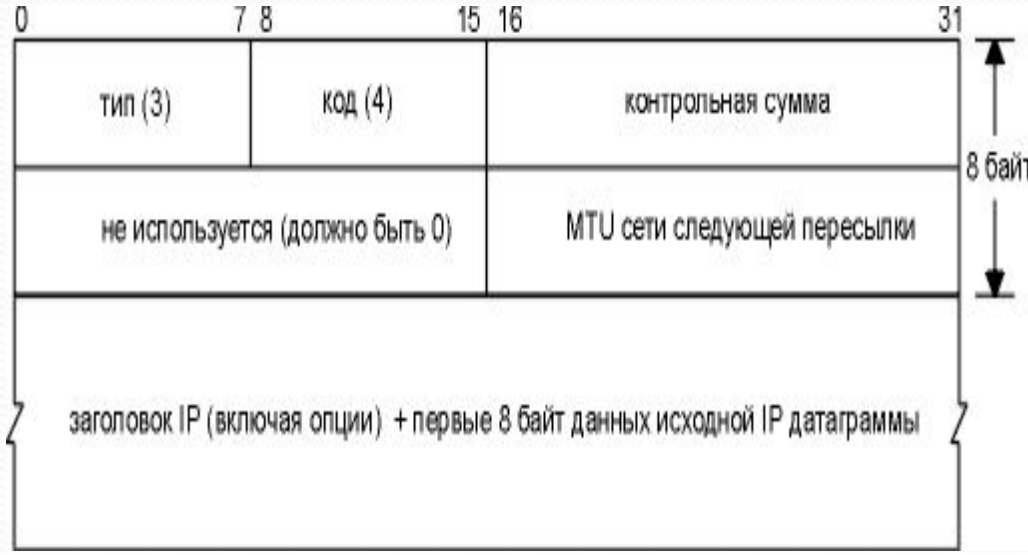
Необходимо обсудить еще одну ICMP ошибку о недоступности. Она генерируется, когда маршрутизатор принимает датаграмму, которую необходимо фрагментировать, однако в IP заголовке установлен флаг "не фрагментировать" (DF). Эта ошибка может быть использована программой, которой необходимо определить минимальный MTU в маршруте до пункта назначения - что называется механизмом определения транспортного MTU (path MTU discovery) (глава 2, раздел "Транспортный MTU").

На рисунке 11.9 показан формат ICMP ошибки о недоступности для данного случая. Он отличается от формата, потому что биты 16-31 во втором 32-битном слове могут содержать MTU следующей пересылки, вместо того чтобы быть установленными в 0.

Рисунок 11.9 ICMP ошибка о недоступности, когда необходима фрагментация, однако установлен бит "не фрагментировать".

Если маршрутизатор не поддерживает этот новый формат ICMP ошибки, MTU следующей пересылки устанавливается в о.

Новые требования к маршрутизаторам Router Requirements RFC [Almquist 1993] указывают на то, что маршрутизатор должен генерировать эту новую форму, когда выдает ICMP сообщение о недоступности.



24.6 Определение транспортного MTU с использованием Traceroute

Так как большинство систем не поддерживают функцию определения транспортного MTU, мы доработаем программу traceroute так, чтобы она могла определять транспортный MTU. Мы отправим пакет с установленным битом "не фрагментировать" (don't fragment). Размер первого отправляемого пакета будет равен MTU исходящего интерфейса. Когда вернется ICMP ошибка "не могу фрагментировать" (can't fragment), мы уменьшим размер пакета. Если маршрутизатор, отправивший ICMP ошибку, поддерживает новую версию, которая включает MTU исходящего интерфейса в ICMP сообщение, мы используем полученное значение; иначе мы попробуем следующий меньший MTU. Как утверждает RFC 1191 [Mogul and Deering 1990], существует ограниченное количество значений MTU, наша программа имеет таблицу возможных значений и просто перейдет на следующее меньшее значение.

Мировой Internet

Модифицированная версия traceroute была запущена несколько раз на различные хосты по всему миру. С ее помощью было достигнуто 15 стран (включая Антарктику), при этом были использованы различные трансатлантические и транстихоокеанские каналы. Однако, до того как сделать это, мы увеличили MTU SLIP канала с дозвоном между нашей подсетью и маршрутизатором netb (рисунок 11.12) до 1500, как в Ethernet.

Из 18 раз, когда была запущена программа, только в двух случаях траспортный MTU был меньше чем 1500. Один трансатлантический канал имел MTU равный 572 (странное значение, которое даже не приведено в списке в RFC 1191), и маршрутизатор не вернул ICMP ошибку в новом формате. Еще один канал между двумя маршрутизаторами в Японии не обрабатывал фреймы размером 1500 байт, также маршрутизатор не вернул ICMP ошибку в новом формате. После того как MTU было уменьшено до 1006, все заработало.

Вывод, который мы можем сделать из этих экспериментов, заключается в том, что большинство (но не все) глобальных сетей в настоящее время могут обрабатывать пакеты больше чем 512 байт. Использование характеристики поиска транспортного MTU позволяет приложениям работать значительно продуктивнее, пользуясь большими MTU.

2.7.2 Определение транспортного MTU при использовании UDP

Давайте рассмотрим взаимодействие между приложением, использующим UDP, и механизмом определения транспортного MTU. Нам необходимо посмотреть, что произойдет в том случае, когда приложение отправляет датаграмму, которая слишком велика для некоторого промежуточного канала.

Пример

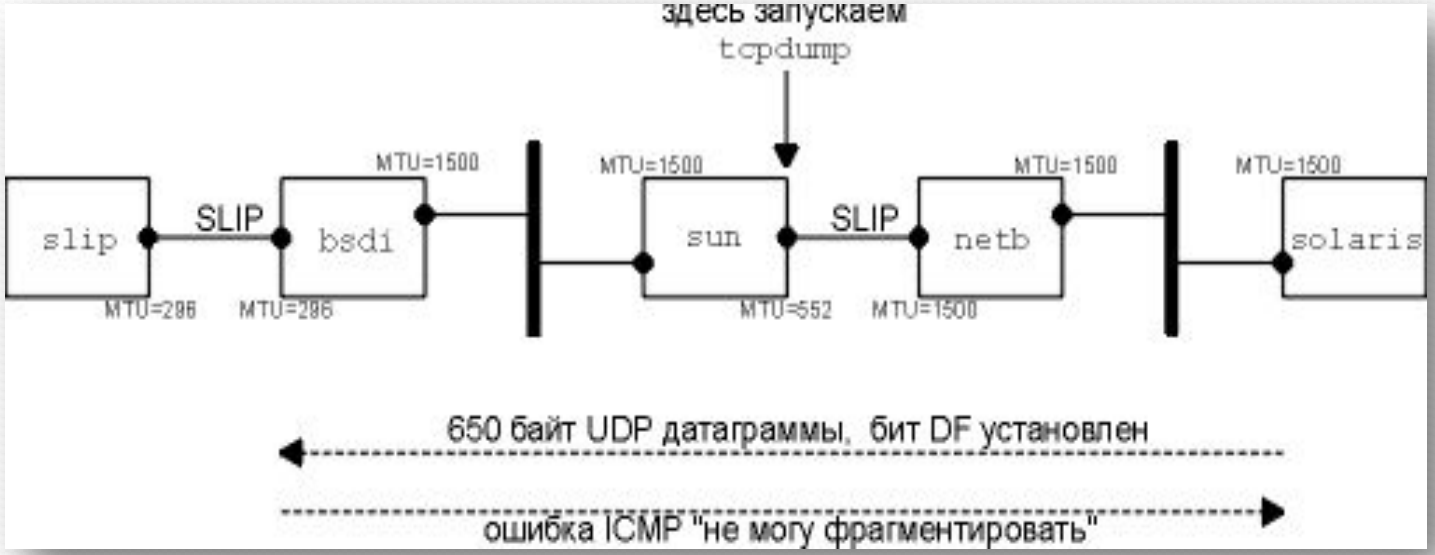
Так как единственная система, которая поддерживает механизм определения транспортного MTU, это Solaris 2.x, мы используем ее как хост источник, чтобы отправить датаграмму размером 650 байт на slip. Так как хост slip находится позади SLIP канала с MTU равным 296, любая UDP датаграмма больше чем 268 байт (296 - 20 - 8) с установленным битом "не фрагментировать" должна вызвать ICMP ошибку "не могу фрагментировать" с маршрутизатора bsdi. На рисунке 11.13 показана топология и MTU каналов.

Следующая команда генерирует десять UDP датаграмм размером 650 байт с интервалом в 5 секунд:

solaris % **sock -u -i -n10 -w650 -p5 slip discard**

На рисунке 11.14 показан вывод команды tcpdump. Когда этот пример был запущен, маршрутизатор bsdi был сконфигурирован таким образом, чтобы не возвращать MTU следующей пересылки как часть ICMP ошибки "не могу фрагментировать".

Первая посланная датаграмма с установленным битом DF (строка 1) генерирует ожидаемую ошибку от маршрутизатора bsdi (строка 2). Что интересно, следующая датаграмма, также посланная с установленным битом DF (строка 3), генерирует ту же самую ICMP ошибку (строка 4). Мы ожидали, что эта датаграмма будет послана с выключенным битом DF.



Требования к хостам Host Requirements RFC требуют от реализаций, чтобы они предотвращали лавинообразную рассылку ARP запросов (повторная отправка ARP запросов для одного и того же IP адреса с большой частотой). Рекомендуемая максимальная частота составляет один раз в секунду. Здесь мы видим шесть ARP запросов в течение 4,3 миллисекунды. Требования к хостам Host Requirements RFC требуют, чтобы ARP сохранил по крайней мере один пакет, и это должен быть самый последний пакет. Это как раз то, что мы видели здесь.

Следующая необъяснимая аномальность заключается в том, что svr4 отправил назад семь ARP откликов, а не шесть.

И последнее, про что хочется сказать, tcpdump работал еще 5 минут после того, как вернулся последний ARP отклик, ожидая увидеть, как svr4 пошлет назад ICMP ошибку "время истекло в течение повторной сборки" (time exceeded during reassembly). ICMP сообщение так и не появилось.

2.8 Взаимодействие между UDP и ARP

Используя UDP, мы можем рассмотреть очень интересное взаимодействие между UDP и типичной реализацией ARP.

Мы используем программу sock, чтобы сгенерировать одну UDP датаграмму с 8192 байтами данных. Мы ожидаем, что в этом случае будет сгенерировано шесть Ethernet фрагментов. Также, перед запуском программы, мы убедимся в том, что ARP кэш пуст, поэтому перед тем как будет отправлен первый фрагмент, должен произойти обмен ARP запросом и откликом.

bsdi % **arp -a** проверяем, что ARP кэш пуст
bsdi % **sock -u -i -n1 -w8192 svr4 discard**

Мы ожидаем, что первая датаграмма вызовет отправку ARP запроса. Следующие пять фрагментов, которые генерируются IP, ставят перед нами два вопроса, на которые мы можем ответить, только воспользовавшись tcpdump: будут ли готовы к отправке оставшиеся фрагменты, перед тем как будет получен ARP отклик, если так, что будет делать ARP с этими несколькими пакетами направляемыми на конкретный пункт назначения, пока ожидается ARP отклик?

2.9. Максимальный размер UDP датаграммы

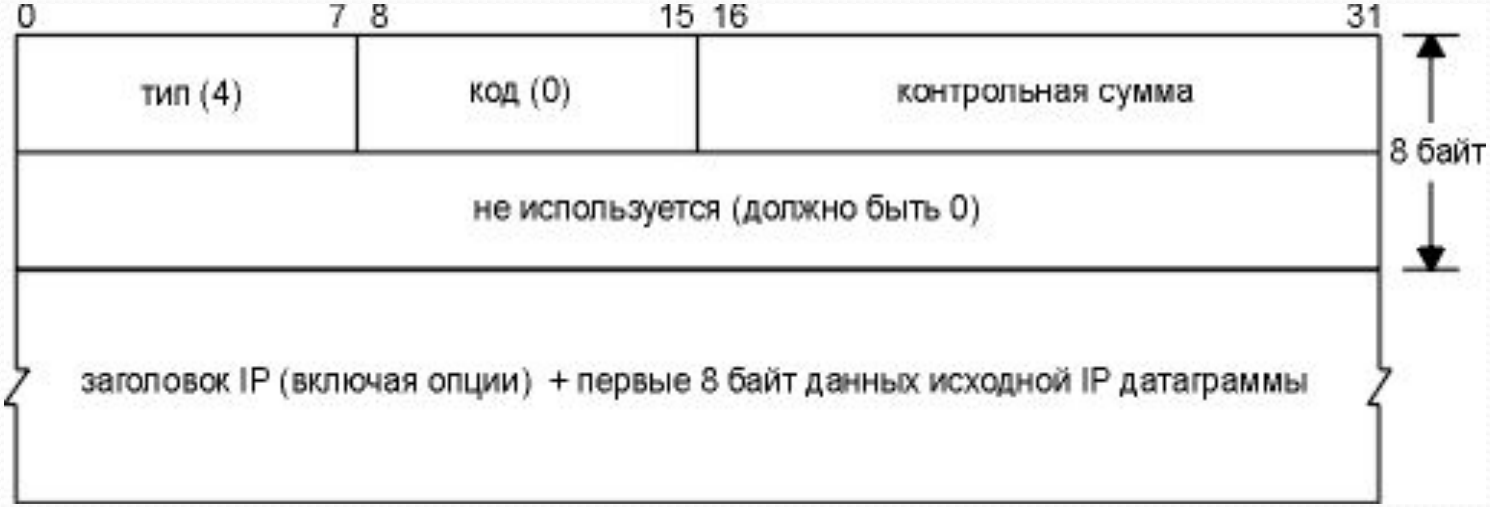
Теоретически максимальный размер IP датаграммы может составлять 65535 байт, что ограничивается 16-битным полем полной длины в IP заголовке. При длине IP заголовка равной 20 байтам и длине UDP заголовка равной 8 байтам в UDP датаграмме для пользовательских данных остается максимум 65507 байт. В большинстве реализаций, однако, используются датаграммы значительно меньшего размера.

ICMP ошибка подавления источника

Воспользовавшись UDP, можно сгенерировать ICMP ошибку "подавление источника" (source quench). Эта ошибка может быть сгенерирована системой (маршрутизатором или хостом), когда она принимает датаграммы быстрее, чем эти датаграммы могут быть обработаны. Обратите внимание на выражение "могут быть". Система не требует послать подавление источника, даже если буферы переполнены и датаграммы отбрасываются.

На рисунке 11.18 показан формат ICMP ошибки подавления источника. Мы имеем идеальную возможность сгенерировать подобную ошибку в нашей тестовой сети. Мы можем посылать датаграммы с bsdi на маршрутизатор sun по Ethernet, причем эти датаграммы должны быть смаршрутизированы через SLIP канал. Так как SLIP канал примерно в тысячу раз медленнее чем Ethernet, мы легко можем переполнить буфер. Следующая команда посылает 100 датаграмм размером 1024 байта с хоста bsdi через маршрутизатор sun на solaris. Мы отправляем датаграммы на стандартный discard сервис, где они будут игнорированы:

bsdi % **sock -u -i -w1024 -n100 solaris discard**



3. ЗАКЛЮЧЕНИЕ

UDP это простой протокол. Официальная спецификация RFC 768 [Postel 1980] состоит всего лишь из трех страниц. Сервисы, которые он предоставляет пользовательским процессам, находящиеся над и позади IP, это номера портов и необязательные контрольные суммы. Мы использовали UDP, чтобы просмотреть расчет контрольных сумм и посмотреть, как осуществляется фрагментация.

Затем мы рассмотрели ICMP ошибку о недоступности, которая является частью новой характеристики определения транспортного MTU. Мы рассмотрели определение транспортного MTU с использованием Traceroute и UDP. Также рассмотрен процесс взаимодействия UDP и ARP.

Мы убедились, что ICMP ошибка подавления источника может быть послана системой, которая принимает IP датаграммы быстрее, чем может обработать. Существует возможность легко генерировать эти ICMP ошибки с использованием UDP.

СПАСИБО ЗА ВНИМАНИЕ!!!