



Урок 5

Транспортный уровень

Протоколы с гарантированной и негарантированной доставкой данных: TCP и UDP. Форматы TCP-сегмента и UDP-дейтаграммы. Сокеты. Технология перегруженного NAT(PAT). Диагностика транспортного уровня

Транспортный уровень

Типы транспортных протоколов

Идентификация процессов, порты, сокеты

Сокет (программный интерфейс)

Базовые операции сокетов для TCP

UDP

TCP

Технология подтверждений

Управление потоком

Методы управления потоком

SCTP

DCCP

Место протоколов TCP и UDP в модели OSI

Проблема нехватки IPv4 адресов и NAT

NAT

Статический NAT

Динамический NAT

Проблема нехватки IPv4-адресов

[Перегруженный NAT](#)

[Destination NAT](#)

[Практика](#)

[Настройка NAT Overload в Cisco Packet Tracer](#)

[Настройка перегруженного NAT в Linux](#)

[Практическое задание](#)

[Дополнительные материалы](#)

[Используемая литература](#)

Транспортный уровень

OSI/ISO	TCP/IP (DOD)
7. Прикладной уровень	4. Уровень приложений
6. Уровень представления	
5. Сеансовый уровень	
4. Транспортный уровень	3. Транспортный уровень
3. Сетевой уровень	2. Сетевой уровень
2. Канальный уровень	1. Уровень сетевых интерфейсов
1. Физический уровень	

На прошлых занятиях мы рассмотрели, как происходит физическая адресация и для чего она нужна, как происходит адресация на сетевом уровне. Как вы помните, на канальном уровне мы можем идентифицировать сетевые интерфейсы в пределах сети, на сетевом уровне мы можем идентифицировать хосты, даже находящиеся в других сетях (так как сетевой адрес содержит и часть, идентифицирующую сеть, и часть, идентифицирующую хост). Но нам необходимо не только определить, с какого компьютера на какой передавать информацию, но и также какому приложению отдать полученные данные. На одном компьютере могут быть запущены несколько серверов: HTTP-сервер, почтовый сервер, удаленный доступ по SSH.

Теперь мы подходим к рассмотрению к межпроцессному взаимодействию.

Итак, резюмируем.

Канальный уровень – взаимодействие между сетевыми интерфейсами.

Сетевой уровень – взаимодействие между хостами.

Транспортный уровень – взаимодействие между процессами (не важно, на одной машине или на разных они запущены). Транспортный уровень принимает данные прикладных протоколов, которые реализуются непосредственно прикладными программами, при необходимости разбивает поток данных на фрагменты (сегменты), снабжает каждый сегмент или дейтаграмму (если данные на прикладном уровне меньше MTU и используется транспортный протокол без гарантированной доставки), добавляется заголовок транспортного уровня, в котором указаны порт отправителя и порт получателя, — специальные идентификаторы, необходимые для идентификации приложения на хосте отправителя, и приложения на хосте получателя, между которыми происходит обмен сообщениями. Также заголовки содержат контрольную сумму для того, чтобы можно было

контролировать целостность данных при движении между сетями (контрольная сумма канального уровня, как вы помните, позволяет только проверить, что данные не были повреждены именно на конкретном участке пути при передаче через физическую среду, но были ли данные повреждены ранее, этот параметр не скажет). Кроме того, в зависимости от протокола в транспортном заголовке могут содержаться и другие данные, которые мы сегодня рассмотрим.

Отметим, что данный подход характерен только для стека TCP/IP, согласно модели OSI/ISO транспортный уровень связывает отправителя и получателя, обеспечивая гарантированную или негарантированную доставку, а задача идентификации приложений не привязана строго к транспортному уровню. Стоит отметить, что в стеке IPX/SPX идентификация приложений осуществлялась на сетевом уровне и адрес отправителя и получателя помимо адреса сети и адреса хоста содержал также идентификатор сокета (соответственно, для отправителя и получателя). Таким образом IPX-пакеты, или, как еще принято говорить, IPX-дейтаграммы являлись аналогом не столько IP-пакетов, сколько UDP-дейтаграмм, вложенных в IP-пакеты.

Типы транспортных протоколов

Обычно различают транспортные протоколы с установкой соединения и протоколы без установки соединения. В стеке TCP/IP, как правило, под первым в большинстве случаев подразумевается протокол TCP (Transmission Control Protocol), под вторым – UDP (User Datagram Protocol). В других стеках протоколов могут быть и иные протоколы, так в стеке IPX/SPX изначально для гарантированной доставки применялся транспортный протокол SPX (Sequenced Packet eXchange), решающий те же задачи, что и TCP в стеке TCP/IP. Также говорят, что TCP обеспечивает гарантированную доставку сообщений, а UDP — не гарантированную.

Протоколы с установкой соединения используются, когда требуется надежное соединение, когда прикладная программа не решает вопросы сборки пакетов, проверки последовательности их приема и когда последовательность приема данных и идентичность полученных данных отправленным важна. Т.е. текстовый документ или двоичный файл будет отправляться в большинстве случаев через протокол с установкой соединения (есть и исключения: так протокол TFTP — Trivial File Transfer Protocol, используемый для загрузки образа операционной системы с другой машины на бездисковые станции, и альтернативный протокол для передачи гипертекста от Корпорации добра Google – QUIC — «быстрый» самостоятельно решает задачи установки соединения и гарантированной доставки на вышестоящих уровнях модели OSI/ISO: сеансовом, представления, прикладном, работая поверх транспортного протокола UDP). Также протоколы с установкой соединения используются, когда требуется организовать двусторонний канал обмена, как правило, текстовыми сообщениями или текстовыми сообщениями и файлами, то есть осуществить работу в режиме чат, когда двое или несколько участников могут переписываться в обе стороны (например, XMPP для jabber), и как не сложно догадаться, сюда же относятся реализации виртуальных терминалов (ничто иное, как чат с «удаленной машиной», это такие протоколы как telnet и ssh, позволяющие получить доступ к консоли или оболочке удаленной машины).

Протоколы без установки соединения используются, когда требуется доставка небольших порций данных и когда потеря или порядок переданных сообщений не критичен, когда скорость передачи важнее гарантированной доставки всех сообщений. В частности, UDP используется для онлайн-трансляций, VoIP (если несколько сообщений выпадут, это приведет к искажению голоса, если же мы будем ожидать гарантированной доставки, может возникнуть задержка, которая практически неизбежна при использовании для этих целей TCP. В тоже время на качество передачи будет влиять и канал связи, в частности, низкая скорость соединения может потребовать применения соответствующих кодеков на прикладном уровне, которые уменьшают размер передаваемой

информации благодаря снижению качества голоса, т.е. роботизированный, как из радиации, голос. На эту тему можно почитать по ключевому слову «вокодеры»).

Также UDP может использоваться в реализации онлайн-игр.

Стоит отметить, что некоторые протоколы могут одновременно использовать и TCP, и UDP, но обычно для разных задач или как дополнительная опция. Наиболее характерный пример — протокол DNS (Domain Name Service), использующий для запросов-ответов UDP, а для передачи файлов доменных зон между первичным и вторичными серверами — протокол TCP.

Итак, резюмируем. В стеке TCP/IP транспортный уровень решает следующие сетевые задачи.

- Сегментирование данных, полученных от протоколов прикладного уровня, на фрагменты (TCP-сегменты для протокола TCP) для передачи по сети.
- Нумерация и упорядочивание сегментов.
- Буферизация дейтаграмм/сегментов.
- Сопоставление и адресация процессов (приложение) и сетевых запросов (создание сокетов).
- Управление интенсивностью передачи.

Основные протоколы транспортного уровня: TCP, UDP.

Также следует знать, что имеются перспективные альтернативные протоколы транспортного уровня: SCTP, DCCP.

Идентификация процессов, порты, сокет

Каждое приложение, получая доступ к сетевым услугам, использует номер порта для того, чтобы идентифицировать как удаленное приложение (если приложение обращается к серверу) либо чтобы другие приложения могли обратиться к данному серверу удаленно (если приложение само является сервером). Клиентские приложения (равно как и серверные приложения, подключаясь к другим, как клиенты, например, так работает прокси-сервер) используют динамические порты, назначаемые операционной системой. Порт — это некое число, двухбайтовый идентификатор, который идентифицирует на данном хосте приложение (как сервер, так и клиент). Не путайте с портом маршрутизатора или портом на материнской плате. На транспортном уровне порт — понятие исключительно логическое, некий номер, по которому мы идентифицируем приложение на хосте-отправителе (которому нужно вернуть ответ), и приложение на хосте-получателе (которому следует передать сообщение). В стеке TCP/IP используется два набора портов: один с гарантированной доставкой (TCP-порты, STREAM-сокеты), второй — без гарантированной доставки (UDP-порты, DGRAM-сокеты). Это два разных пространства идентификаторов, возможна ситуация, когда одно приложение использует TCP-порт, а другое UDP-порт с тем же номером. Эти два приложения не будут мешать друг другу никоим образом. С другой стороны, одно приложение для реализации услуги может использовать и TCP-порт, и UDP-порт (например, в тех случаях, когда нужна передача больших файлов или дополнительная надежность — TCP, когда же нужно отправлять короткие сообщения — UDP). Так DNS использует порт 53/UDP для коротких DNS-запросов и ответов, 53/TCP для обмена между первичным и вторичными серверами объемными файлами зон.

Номера портов, как правило, общеизвестны и зарезервированы для тех или иных сервисов. При этом не уточняется, какой используется протокол TCP или UDP, на практике же используется либо тот, либо другой, либо оба, но в зависимости от ситуации.

Два байта позволяют идентифицировать до 65545 портов (то есть на каждом хосте теоретически имеется 65545 портов для передачи данных с гарантированной доставкой, т.е. «TCP-портов» и 65545 портов для передачи данных с негарантированной доставкой, т.е. «UDP-портов»).

Порты с номерами от 1 по 1023, как правило, зарезервированы для стандартных служб. В Linux прослушивание портов до 1023 номера требует прав суперпользователя (root). Также имеется диапазон локальных портов, динамически выделяемых для клиентских приложений. Например, в Linux это могут быть порты от 32768 по 60999 (диапазон содержится в файле /proc/sys/net/ipv4/ip_local_reserved_ports).

```
cowboy@tom3:~$ cat /proc/sys/net/ipv4/ip_local_port_range
32768 60999
```

Назначение портов определяется Администрацией адресного пространства Интернет (IANA – Internet Assigned Numbers Authority, она же выдает группы IP-адресов и номера автономных систем RIR – региональным интернет-регистраторам). При этом возможно использование программным обеспечением и незарегистрированных или даже зарегистрированных для других целей портов (частым примером является использование 80 или 443 порта, предназначенных для HTTP/HTTPS службой SSH, чтобы обойти запрет на доступ к порту 22, либо вообще ко всем портам, кроме предназначенных для передачи гипертекста).

Некоторые общеизвестные номера портов.

Например.

20 – передача данных по протоколу FTP.

21 – передача команд по протоколу FTP.

22 – SSH (и заодно SFTP, являющийся надстройкой над SSH).

23 – telnet.

25 – SMTP.

53 – DNS.

67 – BOOTP, DHCP (сервер).

68 – BOOTP, DHCP (клиент).

69 – TFTP.

70 – gopher – предшественник HTTP.

80, 8080 – HTTP.

123 – NTP, SNTP.

110 – POP3.

161 – SNMP.

143 – IMAP4.

179 – BGP (Border Gateway Protocol)

443 – HTTPS (HTTP, инкапсулированный в TLS).

520 – RIP.

989 – FTPS (FTP, инкапсулированный в TLS)-данные.

990 – FTPS (FTP, инкапсулированный в TLS)-управление.

1935 – RTMP (Real Time Messaging Protocol) – используется в вебинарах (например в Clickmeeting, пример неофициального использования порта).

3306 – MYSQL.

5060 – SIP.

Также в исходящих дейтаграммах UDP-соединения в качестве исходящего порта может быть указан 0, если не предполагается получение сообщений в ответ.

Механизм портов в TCP/IP таков, что сервер должен открыть слушающий сокет (указав используемый стек, IPv4, IPX, IPv6 и тип соединения, STREAM или DGRAM), связать его с IP-адресом (может использоваться IP-адрес хоста или 0.0.0.0, если будут прослушиваться все адреса) и номером порта (в зависимости от типа соединения будет использоваться пространство TCP или UDP-номеров). Далее сервер ожидает входящих сообщений или соединений. Клиентское приложение также должно открыть сокет, но вместо того, чтобы прослушивать входящие соединения, клиентский сокет сам подключается к удаленной машине (указав свой IP-адрес, IP-адрес сервера и порт сервера, а подключившись, может ожидать и входящих сообщений после подключения и до закрытия сокета, в принципе, соединения равноправны). Локальный порт клиентскому приложению назначается автоматически операционной системой, более того, приложение номер этого порта даже и не знает.

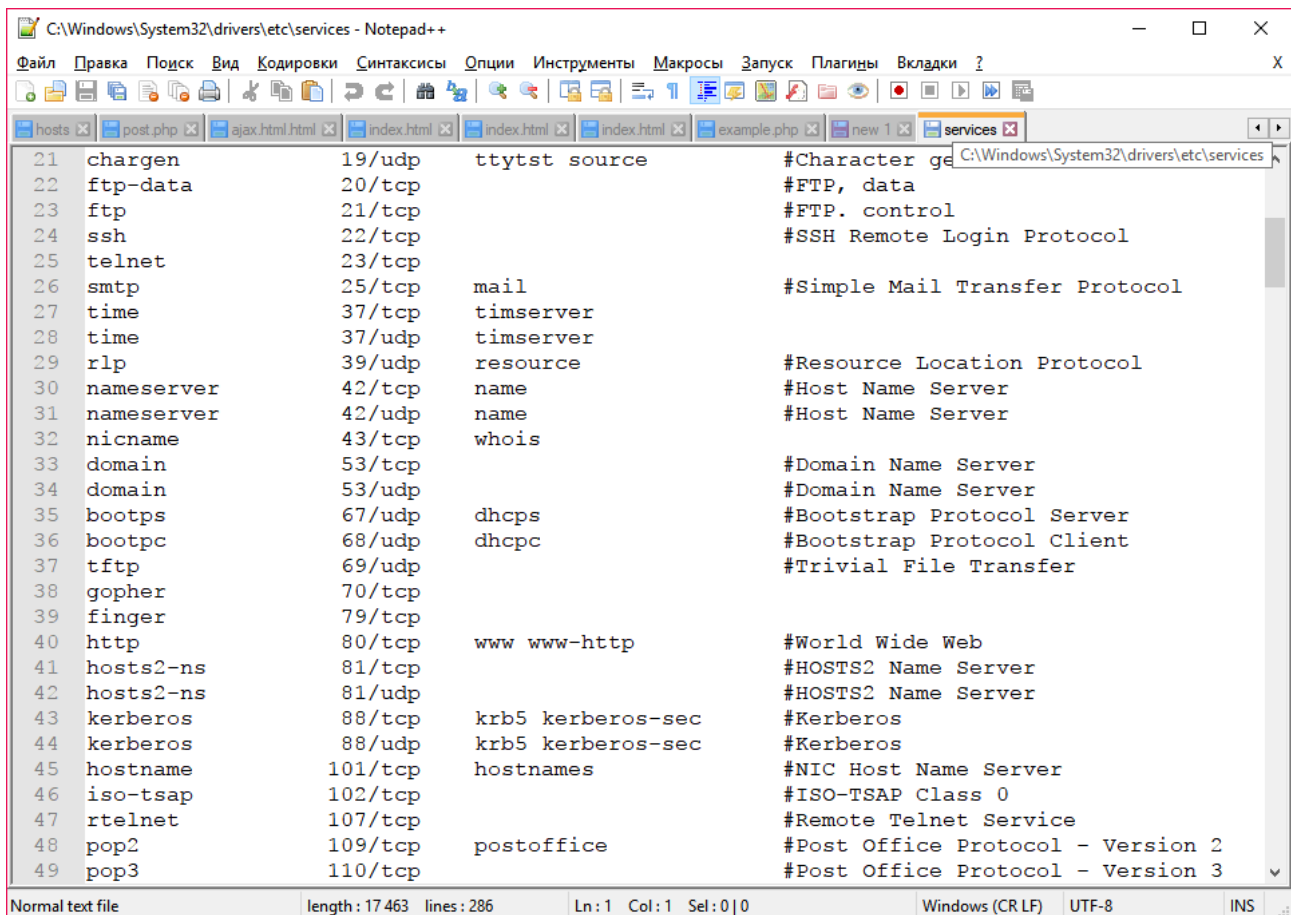
Таким образом, формально соединение является дуплексным. Интересно, что в предшественнике TCP/IP протоколе NCP (Network Control Program, использовался в ARPANET) соединения являлись симплексными и использовали по два симплексных порта, один для отправки сообщения, второй для приема. Еще интереснее, что и сейчас в стеке TCP/IP иногда используются пары портов (как в FTP, но там один для данных, другой для управления соединением), но один случай, когда используется по настоящему два разных порта для обмена сообщениями, это DHCP: сервер использует 67/UDP-порт, а клиент 68/UDP, впрочем, все равно это дуплексный режим работы (клиент отправляет с 68 порта на 67 порт сервера, а сервер с 67 на 68 клиенту). Аналогичная картинка сохранилась и в DHCPv6 (546 – клиент, 547 – сервер).

В Linux перечень портов и сервисов, для которых они зарезервированы, можно посмотреть в файле `/etc/services`.

```
$ cat /etc/services
```

```
cowboy@tom3: ~  
msp          18/udp  
chargen      19/tcp      ttytst source  
chargen      19/udp      ttytst source  
ftp-data     20/tcp  
ftp          21/tcp  
fsp          21/udp      fspd  
ssh          22/tcp      # SSH Remote Login Protocol  
ssh          22/udp  
telnet       23/tcp  
smtp         25/tcp      mail  
time         37/tcp      timserver  
time         37/udp      timserver  
rlp          39/udp      resource     # resource location  
nameserver   42/tcp      name         # IEN 116  
whois        43/tcp      nicname  
tacacs       49/tcp      # Login Host Protocol (TACACS)  
tacacs       49/udp  
re-mail-ck   50/tcp      # Remote Mail Checking Protocol  
re-mail-ck   50/udp  
domain       53/tcp      # Domain Name Server  
domain       53/udp  
mtp          57/tcp      # deprecated  
tacacs-ds    65/tcp      # TACACS-Database Service  
tacacs-ds    65/udp
```

Аналогичный файл имеется и в Windows (C:\Windows\System32\drivers\etc\services).



```
21 chargen 19/udp ttyst source #Character generator
22 ftp-data 20/tcp #FTP, data
23 ftp 21/tcp #FTP. control
24 ssh 22/tcp #SSH Remote Login Protocol
25 telnet 23/tcp
26 smtp 25/tcp mail #Simple Mail Transfer Protocol
27 time 37/tcp timserver
28 time 37/udp timserver
29 rlp 39/udp resource #Resource Location Protocol
30 nameserver 42/tcp name #Host Name Server
31 nameserver 42/udp name #Host Name Server
32 nicname 43/tcp whois
33 domain 53/tcp #Domain Name Server
34 domain 53/udp #Domain Name Server
35 bootps 67/udp dhcps #Bootstrap Protocol Server
36 bootpc 68/udp dhcpc #Bootstrap Protocol Client
37 tftp 69/udp #Trivial File Transfer
38 gopher 70/tcp
39 finger 79/tcp
40 http 80/tcp www www-http #World Wide Web
41 hosts2-ns 81/tcp #HOSTS2 Name Server
42 hosts2-ns 81/udp #HOSTS2 Name Server
43 kerberos 88/tcp krb5 kerberos-sec #Kerberos
44 kerberos 88/udp krb5 kerberos-sec #Kerberos
45 hostname 101/tcp hostnames #NIC Host Name Server
46 iso-tsap 102/tcp #ISO-TSAP Class 0
47 rtelnet 107/tcp #Remote Telnet Service
48 pop2 109/tcp postoffice #Post Office Protocol - Version 2
49 pop3 110/tcp #Post Office Protocol - Version 3
```

Порты с номерами выше 49152 называются динамическими и используются клиентским программным обеспечением.

Говорят, что для TCP и UDP-протоколов используется независимая нумерация портов. Т.е. могут независимо, не мешая друг другу, быть открыты два порта с одним и тем же номером, TCP и UDP.

Впрочем, это не совсем верно, так как кроме TCP и UDP есть другие протоколы.

Сокет (программный интерфейс)

Технология сокетов Беркли является фундаментальной для стека протоколов TCP/IP. Впервые появились в BSD UNIX в 1982 году, и основные идеи используются до сих пор.

Интерфейс сокета Беркли используется для взаимодействия между компьютерами в сети или процессами, запущенными на компьютере. Сокеты – это стандарт интерфейсов для транспортных подсистем. Различные варианты сокетов могут быть реализованы в разных ОС и языках программирования.

Сокет в Unix – интерфейс, доступ к которому похож на работу с файлом, в соответствии с концепцией Unix-way, одним из положений которых является то, что в Unix все является файлом (устройства, структуры процессора, механизмы межпроцессного взаимодействия и т.д.). Аналогично и операции сокетов изначально были подобны операциям работы с файлами (функции read() и write(), те же, что и для файлов, аналогичное закрытие с помощью close()).

Для начала приложение должно открыть сокет, используя функцию `socket()`. Функция возвращает дескриптор сокета (аналогия с файловым дескриптором, пошедшая из `unix-way`). При этом указывается, какой стек протоколов будет использоваться (IPv4, IPv6, Unix-сокеты, использующие для межпрограммного взаимодействия механизм файловых сокетов), тип используемого протокола (STREAM, DGRAM, RAW) и протокол.

Тип используемого сокета.

- STREAM – для передачи с установкой соединения. Используется в TCP и SCTP, а также для UNIX-сокетов.
- DGRAM – для передачи дейтаграмм, то есть без установки соединения. Используется в UDP и DCCP.
- RAW – сырой сокет, т.е. непосредственно работа с IP-пакетами.
- Есть и другие типы, в т.ч. позволяющие непосредственно захватывать кадры канального уровня.

Протокол.

- TCP.
- UDP.
- SCTP.
- DCCP.
- RAW (для сырых сокетов).

Работа с сокетом аналогична работе с файлом, но по факту запись в сокет приводит к передаче файла по сети (сразу или после накопления определенного количества байт в буфере), чтение происходит из буфера полученных данных.

То есть все действия сводятся к чтению или записи, сама передача данных прозрачна для разработчика.

Практика показала, что в чистом виде сокеты - нечто большее, чем файлы, потому набор операций расширился на сокет-специфичные (`send()`, `recv()` для отправки и получения).

Операция `SOCKET` создает новый сокет и записывает его в таблицу транспортной подсистемы. Параметры вызова задают тип используемого формата адресации, тип применяемого сервиса (например, надежный поток байтов) и протокол.

Например, при обращении к серверу `geekbrains.ru` на HTTP порт сокет будет выглядеть так: `5.61.239.21:80`, а ответ будет поступать на IP-адрес запросившей машины и случайный динамический порт клиента (для клиентских приложений используются динамические порты, а для серверных, как правило, фиксированные).

Базовые операции сокетов для TCP

- `SOCKET (СОКЕТ)` - создание нового сокета.
- `BIND (СВЯЗАТЬ)` - привязать локальный IP-адрес и порт.
- `LISTEN (ОЖИДАТЬ)` - слушать входящие соединения, указав размер очереди.
- `ACCEPT (ПРИНЯТЬ)` - подтвердить установление входящего соединения.
- `CONNECT (СОЕДИНИТЬ)` - инициировать процесс установления соединения на удаленную машину.

- SEND (ПОСЛАТЬ) - передать информацию по установленному соединению.
- RECEIVE (ПОЛУЧИТЬ) - принять информацию по установленному соединению.
- CLOSE (ЗАКРЫТЬ) - закрыть сеанс связи и отправить сообщение о завершение соединения.

bind() используется сервером, для того чтобы связать с сокетом прослушиваемые IP-адрес и порт.

connect() служит для того, чтобы подключиться к серверу, для чего в аргументах функции указывается (в составе структуры), с какого IP-адреса, на какой IP-адрес и какой порт мы будем подключаться.

Для перевода сокета в прослушивающее состояние (актуально для сервера) используется listen().

Для закрытия сокета используется close().

Также есть механизм для разрешения доменных имен gethostbyname(). Происходит обращение к файлу /etc/hosts к локальной DNS-службе, которая уже использует поиск по распределенной базе соответствие IP-адреса для указанного имени. Хотя DNS и протокол прикладного уровня, в данном контексте он совершенно особенный, так как использоваться независимо от того, создаем мы сокет с установкой или без установки соединения или даже сырой сокет.

Посмотреть соединения (как в Windows, так и в Linux) можно с помощью команды netstat.

```

C:\>netstat

Активные подключения

Имя      Локальный адрес      Внешний адрес      Состояние
TCP      127.0.0.1:1619        Lenovo-PC:1620     ESTABLISHED
TCP      127.0.0.1:1620        Lenovo-PC:1619     ESTABLISHED
TCP      127.0.0.1:1621        Lenovo-PC:1622     ESTABLISHED
TCP      127.0.0.1:1622        Lenovo-PC:1621     ESTABLISHED
TCP      192.168.0.104:58316   157.55.56.173:40039 ESTABLISHED
TCP      192.168.0.104:58318   13.73.158.204:https ESTABLISHED
TCP      192.168.0.104:58341   40.77.226.192:https ESTABLISHED
TCP      192.168.0.104:58381   db5sch101110533:https ESTABLISHED
TCP      192.168.0.104:58429   a23-78-70-16:https ESTABLISHED
TCP      192.168.0.104:58431   a23-78-70-16:https ESTABLISHED
TCP      192.168.0.104:59200   srv196-4-213-95:https ESTABLISHED
TCP      192.168.0.104:59220   srv212-4-213-95:https ESTABLISHED
TCP      192.168.0.104:59590   149.154.163.48:https ESTABLISHED
TCP      192.168.0.104:61029   ec2-34-253-167-3:https ESTABLISHED
^C
C:\>

```

В windows ключ -a также позволяет отобразить кроме установленных соединений и прослушиваемые (listen) порты. Похожее поведение в Linux. Дополнительно в Linux можно отобразить только прослушиваемые подключения с помощью ключа -l.

Самостоятельно посмотрите все возможности программы netstat с помощью ключа --help.

UDP

User Datagram Protocol (UDP) – протокол передачи дейтаграмм пользователя.

Сообщение UDP называется дейтаграмма.

Особенности UDP.

- Без предварительного установления соединения передача начинается сразу же.
- Негарантированная передача данных, в случае потери данные не будут повторно переданы протоколом UDP.
- Применяется преимущественно служебными протоколами такими, как: DHCP (Dynamic Host Configuration Protocol), RIP (Routing Information Protocol), SNMP (Simple Network management Protocol), NTP (Network Time Protocol) и приложениями, передающими потоковые данные (видео, музыка, голосовые звонки).

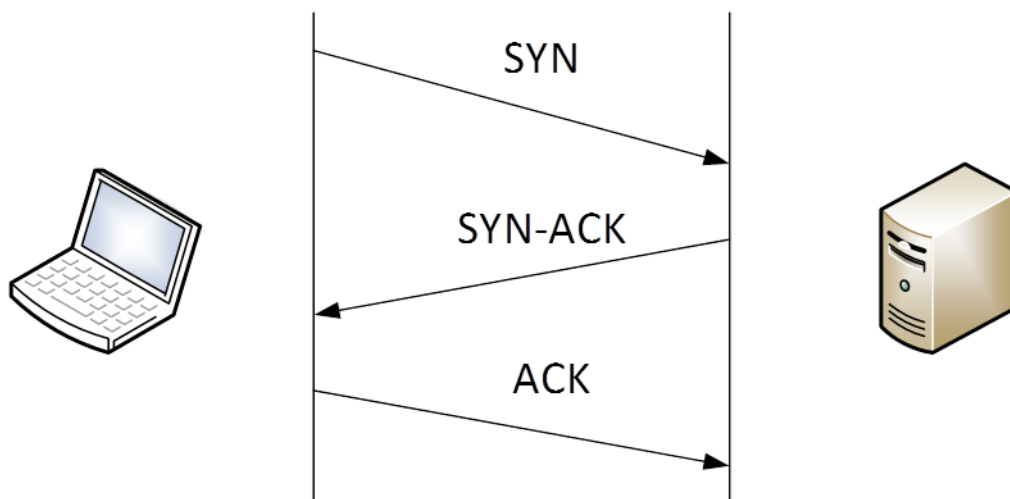


Протокол работает без установления соединения, кроме того, не используется подтверждение о доставке, что приводит к тому, что передаваемые дейтаграммами могут быть потеряны и, как следствие, это не гарантирует доставку данных. Дейтаграммы могут поступать в любой последовательности, повторяться и не доходить до адреса назначения. Это все можно отнести к минусам в отличие от протокола TCP. Плюсом является возможность начать передачу данных без установления соединения.

Пространство адресов протокола UDP отделено от TCP-портов.

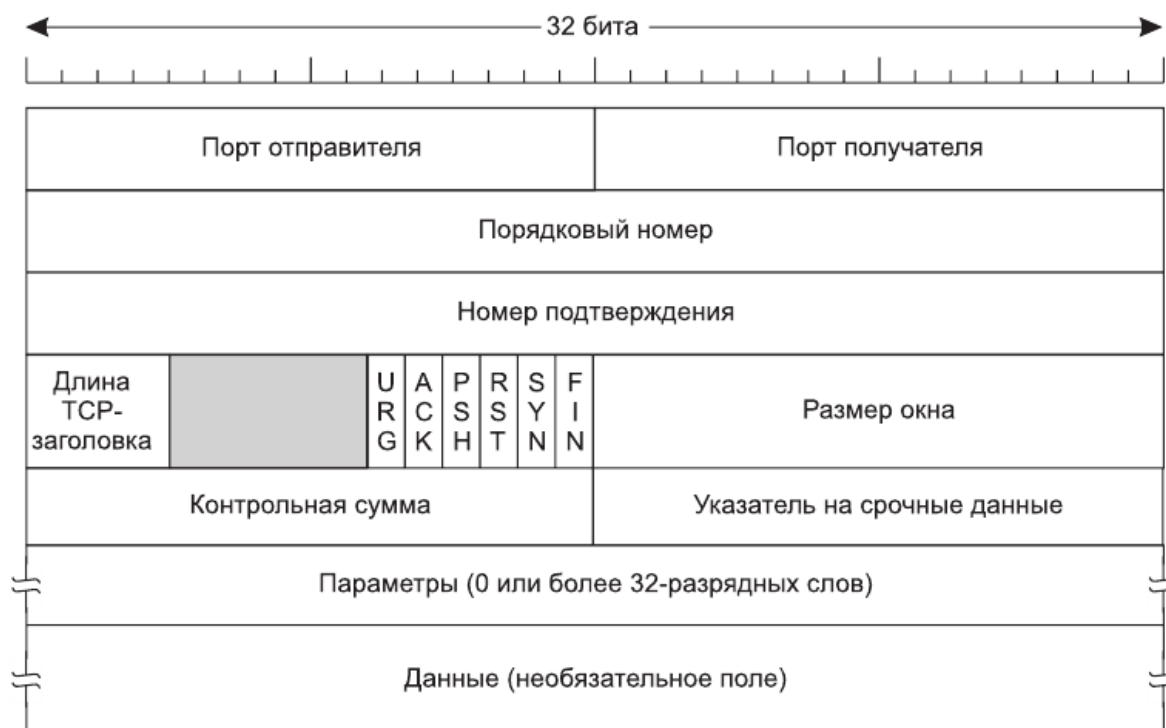
TCP

TCP является протоколом транспортного уровня, который передает поток данных, используя предварительную установку соединения, а также осуществляет контроль передачи данных путем подтверждения (квитирования). В случае потери части данных производится повторная передача данных для устранения потери, также протокол следит за дублированием информации и препятствует получению двух одинаковых пакетов, обеспечивая целостность передаваемых данных и уведомление отправителя о результатах передачи. Перечисленные особенности являются главными отличиями от протокола UDP.



До начала передачи данных клиент должен установить соединение, в котором необходимо согласовать его параметры. Это снижает скорость передачи по сравнению с UDP, но обеспечивает защиту данных.

На рисунке ниже приведена структура TCP дейтаграммы.

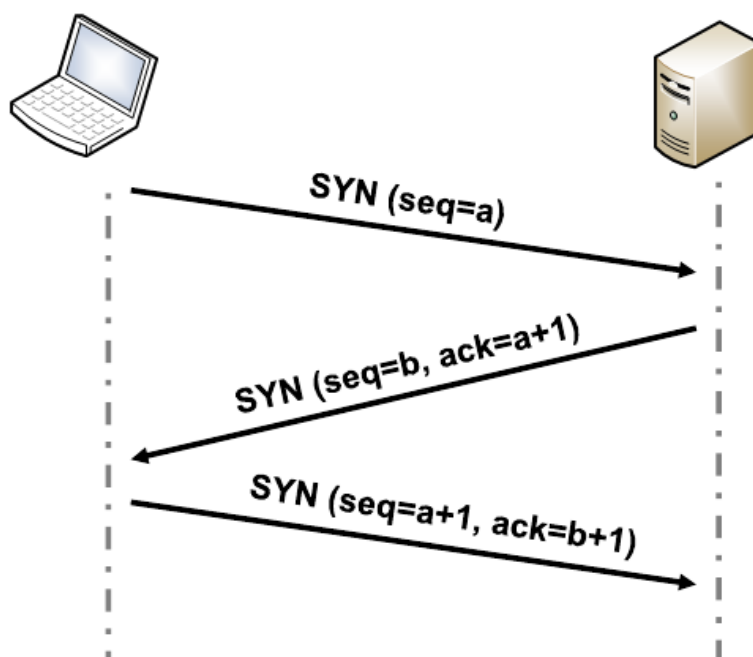


Source Port (2 байта), Destination Port (2 байта) – поля порт источника и назначения обозначают порт, на котором работает процесс-отправитель и процесс-получатель.

Sequence Number (SN) (4 байта) – числовой номер первого байта, использующийся для идентификации текущего соединения. Например, если передаются данные под порядковыми номерами с 3001 по 4000-й, тогда SN=3001.

Во время установления соединения в заголовок сегмента устанавливается бит SYN, а в поле SN устанавливается стартовое число последовательности (ISN), например, 0. В этом случае Sequence Number для данных после установления сеанса будет равен 1 (ISN+1). Следующая переданная дейтаграмма будет иметь номер 2 и т.д.

Acknowledgment Number (ACK) (32 бита) – для подтверждения или запроса дейтаграммы устанавливается бит ACK. В это поле записывается номер сегмента, который отправитель хочет запросить, также это сообщение подтверждает, что все прошлые сообщения успешно получены. Таким образом, реализуется механизм квитирования.



Длина заголовка Data Offset (4 бита) - размер TCP-заголовка в 32-битных словах.

Reserved (6 бит) – зарезервировано и заполняется нулевыми значениями.

Control Bits (6 бит) - служебные биты; при установке 1 считается, что бит установлен, содержит следующие поля.

URG (Urgent Pointer) - поле указателя срочности передачи.

ACK - поле подтверждения.

PSH – при получении TCP сегмента с флагом PSH система передает все данные, находящиеся в буфере процессу получателю, даже если буфер заполнен еще не полностью.

RST - перезапуск сеанса связи и повторное установление соединения.

SYN - запрос на установление соединения.

FIN – конец передачи данных.

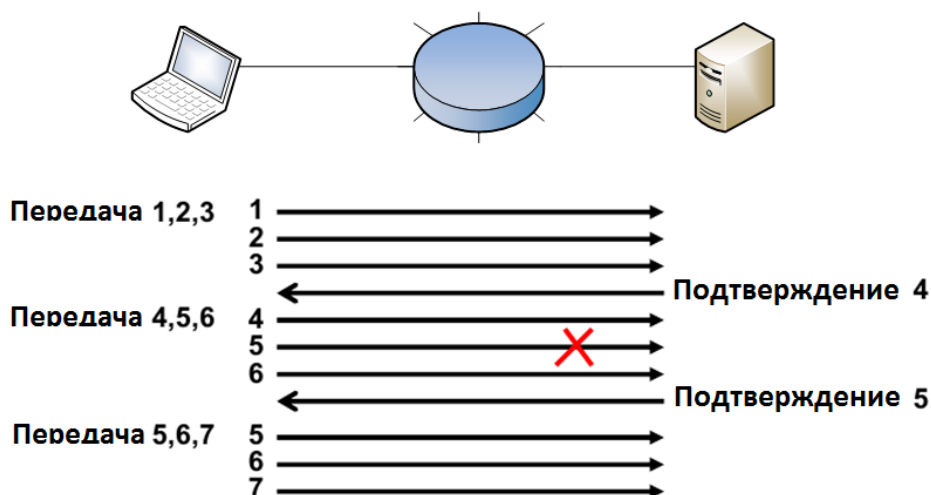
Окно (Window (16 бит) - размер окна в байтах.

Контрольная сумма (Checksum) (16 бит) – поле, состоящее из 16 бит, вычисляемое на основе заголовка сегмента, а также 96 бит псевдозаголовка, который используется перед TCP-заголовком. Псевдозаголовок включает IP-адрес отправителя (4 байта), IP-адрес получателя (4 байта), нулевой байт, байт поля "Протокол", которое совпадает с полем в IP-заголовке и 16 битное поле длины TCP сегмента, измеренное в байтах. Это обеспечивает защиту дейтаграммы TCP от передачи по неправильному маршруту.

Существует много версий протокола TCP, которые по разному обрабатывают сообщения об ошибках и регулируют размер окна.

Протокол TCP не регламентирует правила обработки срочных данных прикладных процессом, поэтому программист должен сам обеспечивать обработку срочного сообщения.

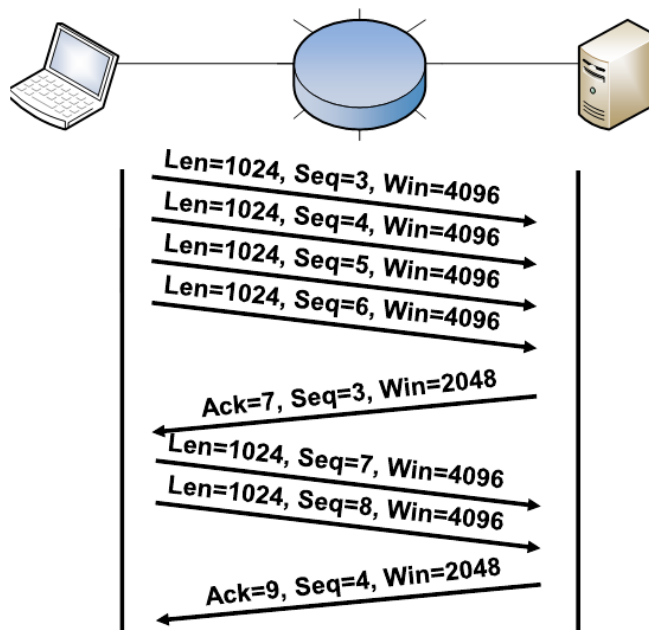
Технология подтверждений



Synchronize (syn) - синхронизировать.

Acknowledge (ack) - подтверждение.

Ниже приведен пример изменения размера окна.



Len= length (длина).

Seq= sequence.

(Номер сообщения в последовательности).

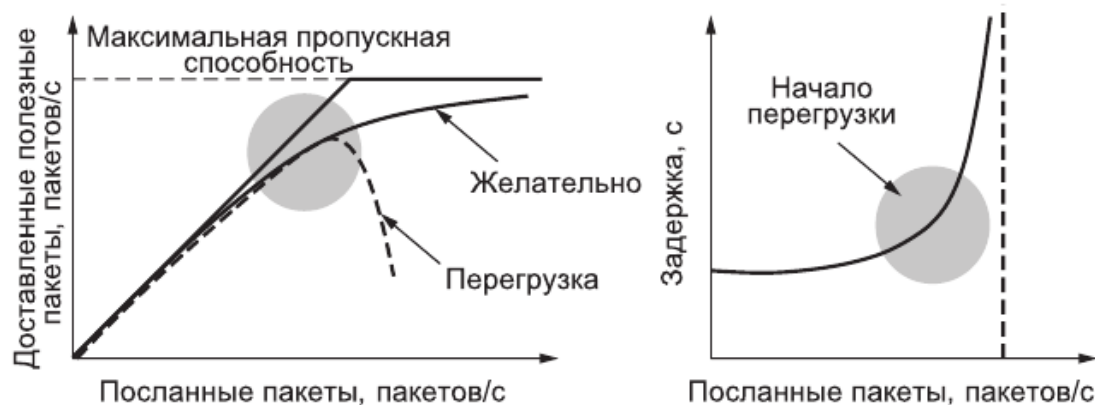
Win=window (размер окна).

Протокол TCP.

- Ориентирован на установление соединения.
- Обеспечивает надежную передачу данных.
- Выполняет управление потоком передачи.

Управление потоком

Протокол выполняет управление потоком данных. В случае детектирования потерь протокол автоматически уменьшает размер окна, тем самым снижая интенсивность передачи. Отправляя сообщения малыми порциями и дожидаясь подтверждения, отправитель обеспечивает более быстрое время реагирования на потери и эффективно использует доступную пропускную способность. К минусам можно отнести эффект холодного старта, когда при установлении соединения протоколу TCP нужно некоторое время для того, чтобы увеличить размер окна до пропускной способности канала. Также такой алгоритм негативно влияет на скорость в каналах с высокой вероятностью потери, в данном случае при каждой потере алгоритм будет снижать размер окна, думая, что в сети возникли перегрузки.



Методы управления потоком

TCP должен хорошо работать как на медленных каналах связи с ошибками, так и на быстрых надежных каналах. Это обеспечивается использованием механизма управления окном.

Изменение размера скользящего окна – основной метод регулирования скорости в TCP.

- Традиционный подход – фиксированный размер окна 8 сегментов TCP.
- Современный подход – динамический размер окна в зависимости от требований приложения и загрузки сети.

Для борьбы с потерями используется буферизация: поток данных буферизируется в ожидание паузы.

Останавливающие сообщения используются для передачи источником сообщений о снижении скорости передачи с помощью ICMP.

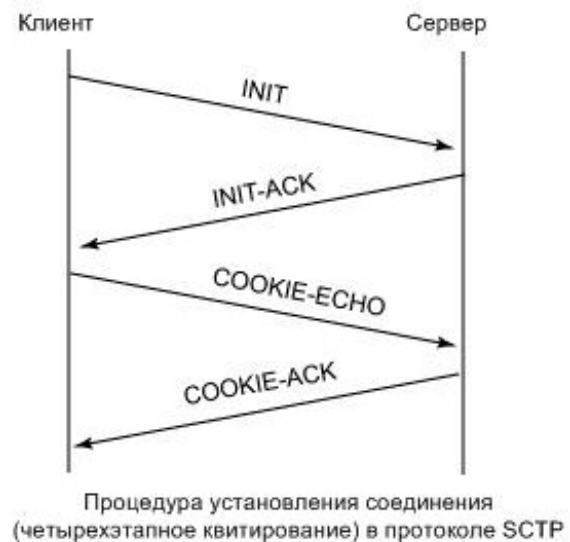
SCTP

Транспортные протоколы развиваются, и встают новые задачи обработки групповых связанных потоков. Например, браузер может запрашивать с сервера несколько страниц, в этом случае создаются отдельные сокеты для каждого соединения.

Управление перегрузкой в данном случае выполняется для каждого потока. Разработан протокол, учитывающий эти особенности и устранивший минусы TCP.

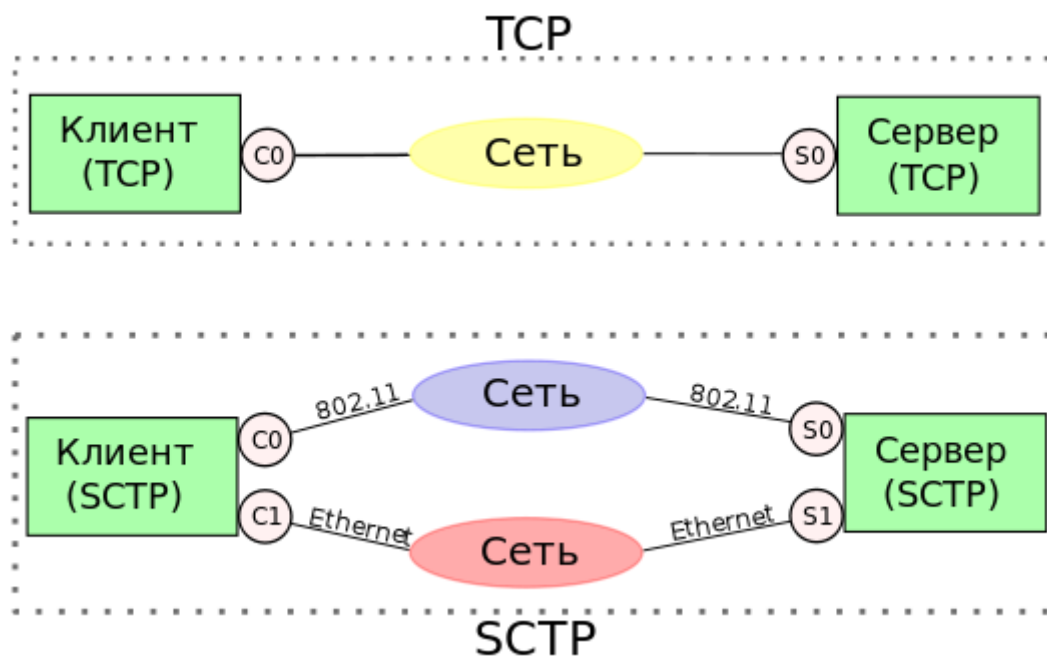
SCTP (Stream Control Transmission Protocol — протокол передачи с управлением потоками), описанный в RFC 4960.

Протокол работает по аналогии с TCP. Из нововведений нужно выделить использование многопоточности и встроенную защиту от DDoS атак в отличие от TCP.



\

Протокол может использовать синхронное соединение хостов по нескольким независимым физическим каналам (multi-homing), например, беспроводную и проводную сеть одновременно.



DCCP

Datagram Congestion Control Protocol — дейтаграммный протокол с управлением перегрузкой, является усовершенствованным протоколом UDP, в который добавили механизм управления перегрузкой. Более подробную информацию по данному протоколу можно посмотреть в RFC 4340.

DCCP отличается следующими пунктами.

- Поток дейтаграмм реализован с механизмом подтверждения получения данных, но без повторной отправки в случае потери данных.
- Ненадежный алгоритм установления и закрытия соединения.
- Согласование параметров передачи данных при установлении соединения.

Протокол предоставляет выбор способов подавления перегрузки канала. Методы управления перегрузкой включают TCP-подобное управление и управление потоком. Протокол позволяет выставлять опции, сообщающие передатчику о том, какие данные были доставлены получателю, какие данные были повреждены при передаче или отброшены в связи с переполнением буфера.

Ключевые особенности протокола.

1. Механизм управления перегрузками и механизм передачи сообщений о перегрузке ECN (Explicit Congestion Notification).
2. Механизм, позволяющий серверу не поддерживать состояния неподтвержденных попыток соединений.
3. Определение узких мест и оптимизирование размера MTU (максимальный размер передаваемого сообщения) пути [RFC 1191].

Место протоколов TCP и UDP в модели OSI

Протоколы TCP и UDP традиционно относятся к транспортному уровню. При этом задачи, решаемые TCP и UDP, не обязательно могут относиться к транспортному уровню модели OSI/ISO. Так протокол UDP транспортного уровня стека TCP/IP является во многом аналогом сетевого протокола IPX-стека IPX/SPX. Не менее важно, что протокол TCP, относясь к транспортному уровню, и в модели OSI/ISO, и в модели TCP/IP частично решает задачи сеансового уровня модели OSI. В частности, это задачи установки и разрыва соединения (это отмечается в учебном пособии [«Анализ трафика мультисервисных сетей»](#), см. п. 1 в доп. материалах). С другой стороны, не всегда используют протокол TCP, создавая протоколы с собственным механизмом управления сеансовым, реализуя сеансовый уровень поверх протокола UDP. Так поступили разработчики протокола QUIC (альтернатива HTTP, решающая задачи сжатия, шифрования и управления сеансом, то есть прикладного, представления и сеансового уровня, работающая поверх протокола UDP. Как несложно догадаться, используются порты 80/UDP и 443/UDP).

Проблема нехватки IPv4 адресов и NAT

NAT

Под термином NAT (Network Address Translation – трансляция сетевых адресов) понимается ряд родственных технологий, относящихся к сетевому, транспортному и даже сеансовому уровню модели OSI.

Традиционно выделяют следующее.

- Static NAT (статический NAT).
- Dynamic NAT (динамический NAT).

- Перегруженный NAT.
- Также существует Destination NAT.

Статический NAT

Предположим, что у нас имеется компьютерная сеть с белыми маршрутизируемыми в Интернет адресами. Например, это может быть компьютерный класс. Любой компьютер, если он в сети, может быть доступен извне, с любого компьютера, подключенного в сети Интернет (такая ситуация наблюдалась в начале 2000-х годов, когда университеты еще обладали сетями класса А и имели возможность щедро назначать их любым собственным ЭВМ). Но такой подход не безопасен. Один из вариантов решения - это использовать вместо белых серые адреса (из диапазонов 10.0.0.0., 192.168.0.0–192.168.255.0 и т.д.). Но тогда возникает вопрос, каким образом осуществить доступ этих машин в Интернет. (Обратную задачу решать не надо, обычные рабочие станции, как правило, не используются, чтобы к ним был доступ извне.) Один из вариантов: выделить одну из машин, которая будет иметь подключение и к внутренней сети, с серыми адресами, и к внешней сети, имея белый IP-адрес, и настроить на данной машине прокси-сервер. Получая запросы от клиентов во внутренней сети, прокси-сервер будет передавать во внешнюю сеть уже от своего имени, таким образом скрывая инфраструктуру локальной сети от внешних наблюдателей. Но это не удобно, так как прокси-сервер (для случая HTTP-сервера) – технология прикладного уровня и позволит получать доступ к WWW, но остальные услуги будут неработоспособны (ни ping, ни smtp, кроме веб-мейл и т.д.).

Технология статического NAT позволяет организовать сокрытие внутренних адресов и доступ к Интернет не на прикладном уровне (HTTP-проху), а на сетевом. Статический NAT подразумевает, что имеется набор серых адресов и набор белых адресов в соотношении один к одному. При прохождении пакета из внутренней сети во внешнюю сеть серый адрес меняется на соответствующий белый, при прохождении в ответ, наоборот, соответствующий белый адрес будет меняться на серый.

Предположим, имеется таблица.

«Частный» IPv4 адрес	«Белый» IPv4 адрес
192.168.32.1	213.18.123.101
192.168.32.2	213.18.123.102
.....	
192.168.32.10	213.18.123.110
192.168.32.11	213.18.123.111
192.168.32.12	213.18.123.112



Компьютер с IPv4-адресом 192.168.32.11 будет виден из внешней сети как 213.18.123.111, когда бы он ни подключился к какому-либо ресурсу. Сам узел 192.168.32.11, в общем-то, ничего не знает об адресе 213.18.123.111, так как шлюз для каждого пакета, адресованного на 213.18.123.111, также будет обратно заменять адрес на 192.168.32.11.

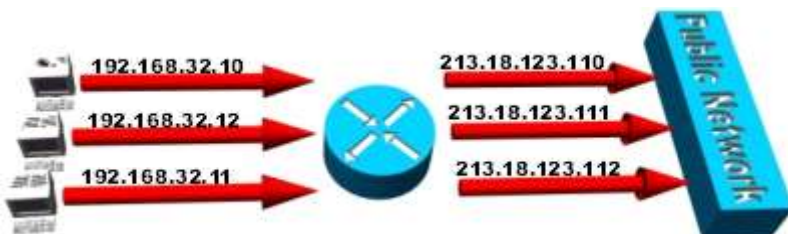
Динамический NAT

Если же количество машин больше, чем белых IP-адресов, и используются они не слишком часто, может использоваться динамический NAT. В этом случае имеется диапазон белых IP-адресов, которые теперь не привязаны к серым IP-адресам. Белые IP-адреса выдаются по мере надобности, и одна машина в разный момент времени может иметь отличные IP-адреса.

NAT-шлюз должен иметь помимо таблицы IPv4 адресов, которые могут быть выделены, также таблицу текущих сессий, связывая серые и выделенные им в соответствии в данный момент времени белые адреса. После того, как машина перестает пользоваться сетевыми услугами, IPv4 белый адрес освобождается, чтобы быть выделенным, например, другой машине.

Доступный диапазон IPv4 адресов: 213.18.123.110 — 213.18.123.115.

Предположим, что три компьютера решили подключиться к каким-либо внешним ресурсам (обратите внимание на порядок подключения).



Динамическая таблица соответствий будет иметь вид.

«Частный» IPv4 адрес	«Белый» IPv4 адрес
192.168.32.10	213.18.123.110
192.168.32.12	213.18.123.111
192.168.32.11	213.18.123.112

Предположим, что узел 192.168.32.11 завершил сессию.

Теперь таблица выглядит вот так.

«Частный» IPv4 адрес	«Белый» IPv4 адрес
192.168.32.10	213.18.123.110
192.168.32.12	213.18.123.111

Предположим, что узел 192.168.32.10 тоже завершил сессию, но к сети подключается узел 192.168.32.15.



«Частный» IPv4 адрес	«Белый» IPv4 адрес
192.168.32.10	213.18.123.110
192.168.32.12	213.18.123.111
192.168.32.15	213.18.123.112

Если узел 192.168.32.11 снова обратится к сети, он уже получит другой адрес, так как предыдущий используемый им адрес занят. Предположим, что 192.168.32.10 завершил сессию. Вполне возможно, что теперь 192.168.32.11 получит адрес 213.18.123.110.



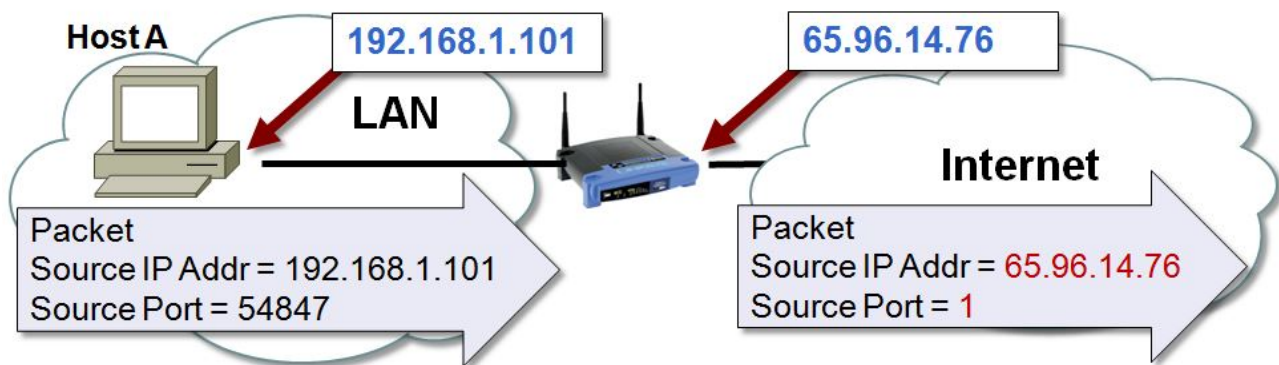
«Частный» IPv4 адрес	«Белый» IPv4 адрес
192.168.32.12	213.18.123.111
192.168.32.15	213.18.123.112
192.168.32.11	213.18.123.110

Проблема нехватки IPv4-адресов

Пространство IPv4-адресов оказалось слишком небольшим. В настоящее время оно почти полностью истрачено. Из-за этого была разработана бесклассовая адресация, блоки адресов стали выделяться меньшими порциями, провайдеры перестали выделять белые IP-адреса (даже динамические). То, что адресов надолго не хватит, стало понятно еще в 1992 году, когда была образована рабочая группа IPng, которая к 1996 году подготовила первую спецификацию IPv6, обладающую гораздо большим пространством адресов. При этом сама технология NAT появилась в 1994 году. Технология статического NAT позволяет обеспечить безопасность, но не сэкономить пространство адресов. Динамический NAT позволяет частично сэкономить пространство адресов, но все равно требует определенного диапазона белых адресов. Дальнейшее развитие идеи: перегруженный NAT, позволяющий обеспечить доступ для нескольких машин с серыми адресами, имея даже всего лишь один белый IP-адрес. Сейчас идет внедрение IPv6 адресов и можно сказать, что для серверной инфраструктуры оно состоялось. (Купите VDS у какого-нибудь провайдера, удостоверьтесь, что IPv6-адрес подключен и с помощью tcpdump отследите любой трафик, например, при попытке с помощью apt-get install установить какое-нибудь новое ПО. Вы будете наблюдать IPv6-трафик.) Но провайдеры не спешат переходить на IPv6, все больше и больше серых адресов скрывая за NAT-шлюзом. Для этих целей даже был выделен еще один блок серых адресов – 100.64.0.0/10. Провайдеры все чаще используют IP-адреса из этого диапазона вместо выделения динамических, но белых IPv4-адресов. Стоит отметить, что NAT-тупиковая, “костыльная” технология. Ряд протоколов не может нормально работать с NAT, для них приходится изобретать заплатки для возможности работы через NAT (такие, как «пассивный режим» FTP).

Перегруженный NAT

Перегруженный NAT (NAT Overload) имеет много синонимов: PAT (Port Address Translation), NAPT (Network Address and Port Translation), Masquerading (Маскарадинг). Если статический NAT может осуществляться устройством «на лету» и замене производятся только L3-заголовки, а динамический NAT также осуществляет замену IP-адресов в L3-заголовках (но устройство уже вынуждено отслеживать активные сессии), то для реализации перегруженного NAT устройство должно анализировать и менять не только L3 но и L4-заголовки. Теперь вместо соответствия серый IPv4-адрес и белый IPv4-адрес используется соответствие IPv4-серый адрес: исходный порт — новый порт на шлюзе с белым IPv4-адресом. Таким образом, шлюз должен отслеживать сессии и очищать таблицы соответствий при завершении клиентских сессий. В частности, в Linux механизм NAT встроен в сетевой фильтр netfilter (и соответственно, управляется правилами iptables), то есть фактически - это сеансовый уровень модели OSI/ISO (при всем при том, что фильтр позволяет анализировать заголовки и канального, и сетевого, и транспортного, и отслеживать прикладные протоколы, смысл в том, что для ушедшего пакета/дейтаграммы необходимо запомнить подстановку, и при обратном ответе суметь сделать подстановку обратно. Таким образом сетевые фильтры отслеживают сессии – такая технология называется stateful filtering. Есть также stateless filtering – без отслеживания состояний, но NAT это как раз stateful filtering).



NAT Translation Table				
	Local IP Address	Source Port #	Internet IP Address	Source Port #
process X, Host A →	192.168.1.101	54,847	= 65.96.14.76	1
Host B →	192.168.1.103	24,123	= 65.96.14.76	2
process Y, Host A →	192.168.1.101	42,156	= 65.96.14.76	3
Host C →	192.168.1.102	33,543	= 65.96.14.76	4

Предположим, компьютер с серым адресом 192.168.1.101 обращается с TCP-порта 54847 к некоторой машине 65.96.14.76 (на какой порт -- не важно, например, на 80). Шлюз, получив сообщение с 192.168.1.101:54847, добавляет эти данные в таблицу, дополнив еще одним полем -- сведениями о выделенном для этого соединения порте, например, 54. После этого в заголовках сетевого и транспортного уровня будет произведена замена: 192.168.1.101:54847 на 65.96.14.76:54 (65.96.14.76 -- белый адрес шлюза).

Сервер, в свою очередь, будет отвечать на 65.96.14.76:54, и шлюз произведет обратную трансляцию, таким образом, клиент получит ответ на 192.168.1.101:54847 и в общем случае не заметит подмены.

Рассмотрим интересную ситуацию. Если следующий клиент (например, машина 192.168.1.104) при этом попытается отправить сообщение по стечению обстоятельств с такого же порта (54847/TCP), работать, как вы догадались, будет. Только в таблице появится еще одна запись 192.168.1.104:54847, 55. Порт на шлюзе будет выделен отличный от предыдущего, следующий по порядку.

Таким образом, в данной модели уже не IP-адрес идентифицирует хост, а пара IP-адрес и порт, хотя, если быть совсем последовательным, то следует сказать, что порт на шлюзе уже идентифицирует не отдельное приложение на данном хосте, а отдельное приложение и хост в локальной сети, на котором оно запущено.

Destination NAT

Как вы поняли, при использовании NAT любая машина в локальной сети может получить доступ к услугам в Интернет, но внешние узлы не смогут инициализировать соединения во внутреннюю сеть.

Предположим обратную ситуацию: у нас имеется белый IPv4-адрес на шлюзе (например, домашнем роутере), но нам необходимо дать доступ на один или два компьютера в локальной сети. В этом помогает Destination NAT. Заходим в настройки роутера (в примере -- настройки для домашнего

WiFi-роутера TP-LINK № TL-MR3420) и прописываем соответствия внешних портов на роутере и внутренних IP-адресов и портов на наших серверах. Трансляция напоминает вышеописанный алгоритм, но таблица статична и работает в обратную сторону.

Виртуальные серверы

ID	Порт сервиса	Внутренний порт	IP-Адрес	Протокол	Состояние	Изменить
1	8080	80	192.168.0.200	Все	Включено	Редактировать Удалить
2	8088	80	192.168.0.201	Все	Включено	Редактировать Удалить
3	8000	80	192.168.0.202	Все	Включено	Редактировать Удалить

[Добавить новую...](#)

[Включить все](#)

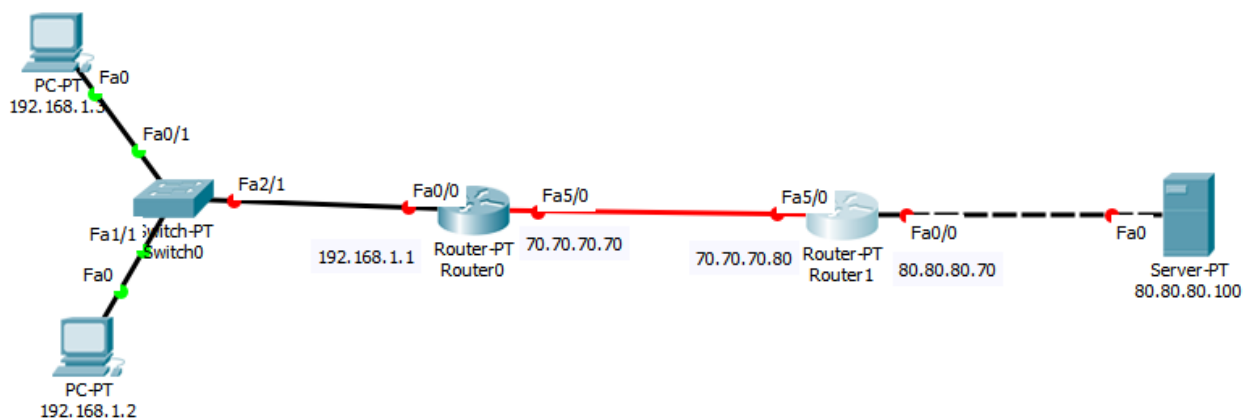
[Отключить все](#)

[Удалить все](#)

Практика

Настройка NAT Overload в Cisco Packet Tracer

Соберем простую схему.



Предположим, нам необходимо настроить сеть таким образом, чтобы сеть 192.168.1.0/24 была скрыта за NAT, и все сообщения в остальные сети шли от IPv4 адреса шлюза 70.70.70.70.

Сначала настроим Router-1.

Настроим сетевые интерфейсы и включим их.

```

Router>
Router>ena
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#int fa0/0
Router(config-if)#ip addr 80.80.80.70 255.255.255.0
Router(config-if)#no shut
Router(config-if)#int fa5/0
Router(config-if)#ip addr 70.70.70.80 255.255.255.0
Router(config-if)#no shut

```

Настроим маршрутизацию по протоколу RIP2. Мы анонсируем обе сети, так как нужно предоставить доступ к сети 80.80.80.0/24, в том числе, к серверу 80.80.80.100, а сервер должен иметь возможность отправлять сообщения в сеть 70.70.70.0/24.

```

Router(config-if)#route rip
Router(config-router)#version 2
Router(config-router)#network 80.80.80.0
Router(config-router)#network 70.70.70.0
Router(config-router)#end

```

Можем проверить таблицу маршрутизации.

```

Router#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

    70.0.0.0/24 is subnetted, 1 subnets
C       70.70.70.0 is directly connected, FastEthernet5/0
    80.0.0.0/24 is subnetted, 1 subnets
C       80.80.80.0 is directly connected, FastEthernet0/0

```

Мы видим только непосредственно доступные сети. Протокол RIP2 настроен, но нужно настроить и другой маршрутизатор.

Теперь настроим Router0.

Настроим сетевые интерфейсы и включим.

```

Router>ena
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#int fa0/0
Router(config-if)#ip addr 192.168.1.1 255.255.255.0
Router(config-if)#no shut

%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

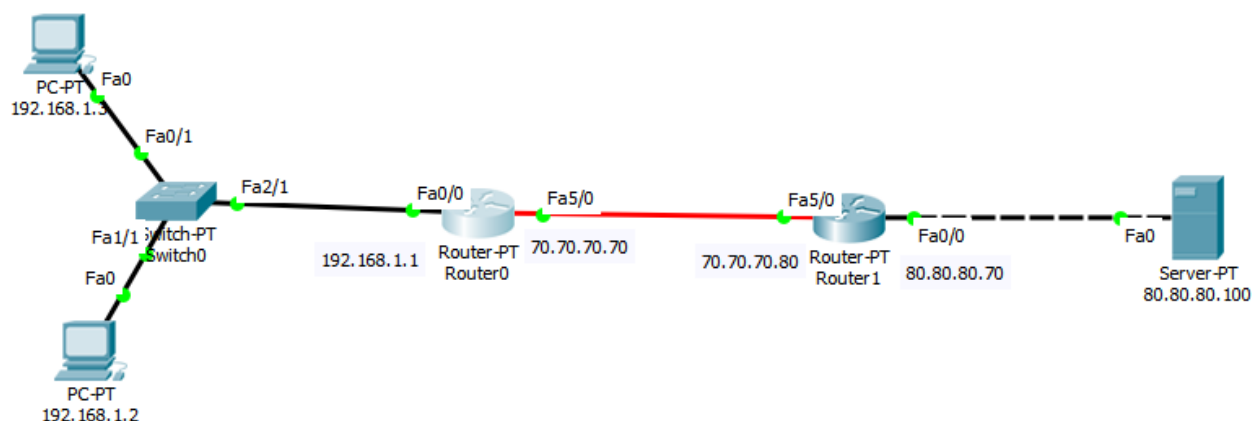
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up

Router(config-if)#int fa5/0
Router(config-if)#ip addr 70.70.70.70 255.255.255.0
Router(config-if)#no shut

%LINK-5-CHANGED: Interface FastEthernet5/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet5/0, changed state to up

```



Настроим динамическую маршрутизацию.

```

Router(config-if)#route rip
Router(config-router)#version 2
Router(config-router)#network 70.70.70.0
Router(config-router)#exit
Router(config)#exit

```

Обратите внимание, мы не будем анонсировать 192.168.1.0 сеть (она скрыта будет за NAT), но нам необходимо анонсировать сеть 70.70.70.0, иначе мы не получим данные о сетях от маршрутизатора Router1.

Проверяем.

```
Router#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

    70.0.0.0/24 is subnetted, 1 subnets
C       70.70.70.0 is directly connected, FastEthernet5/0
R       80.0.0.0/8 [120/1] via 70.70.70.80, 00:00:12, FastEthernet5/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
```

Видим, что сети 70.70.70.0 и 192.168.1.0 доступны напрямую через сетевые интерфейсы, а вот маршрут в сеть 80.0.0.0/8 доступен через шлюз 70.70.70.80 – эта информация получена нами по протоколу RIP2 от второго маршрутизатора.

Теперь настало время настроить NAT.

Сначала создаем список доступа, указывая, какие адреса могут использовать NAT.

```
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#access-list 1 permit 192.168.1.0 0.0.0.255
```

Укажем, что пакеты клиентов с IP-адресов из списка 1 будут подвергаться перегруженной NAT-трансляции при следовании через интерфейс fa5/0.

```
Router(config)#ip nat inside source list 1 int fa5/0 overload
```

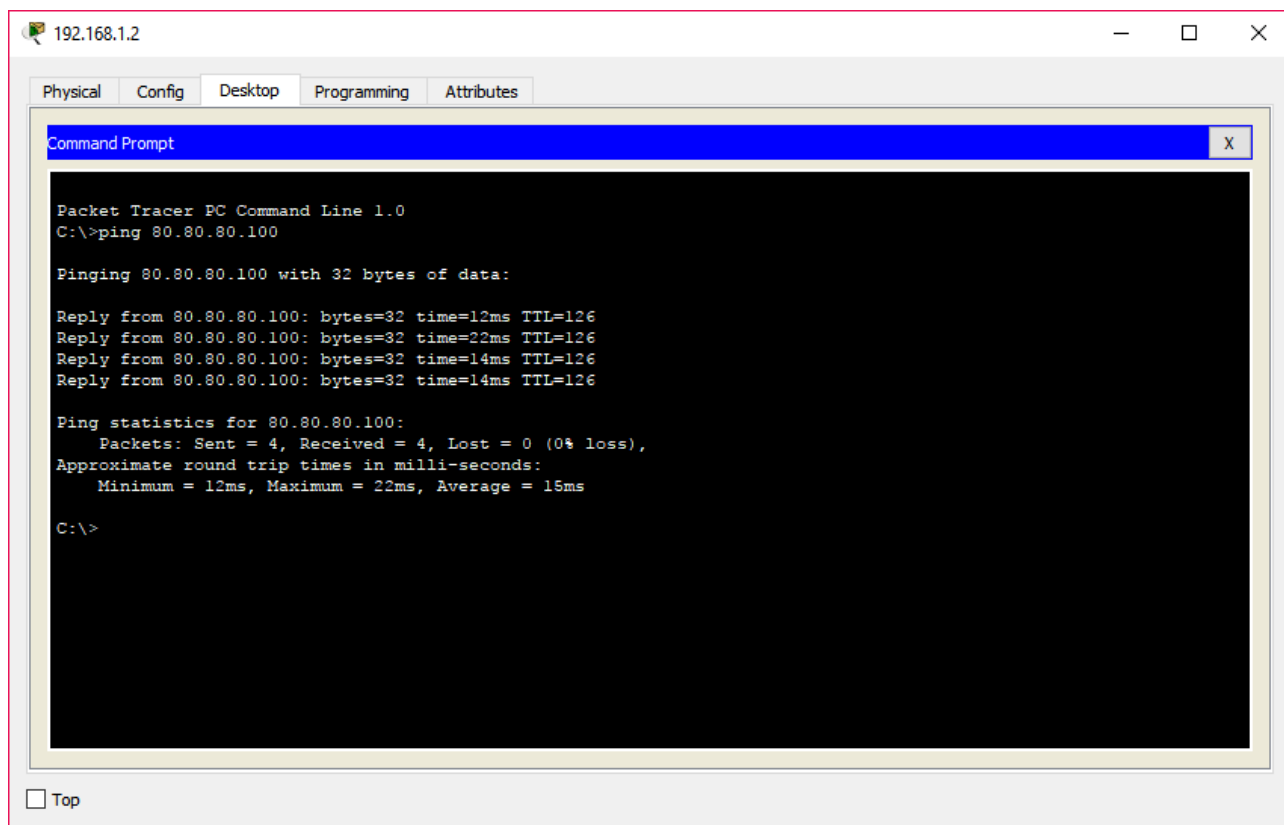
Укажем, что int fa0/0 – внутренний сетевой интерфейс для NAT.

```
Router(config)#int fa0/0
Router(config-if)#ip nat inside
```

Укажем что int fa5/0 – внешний сетевой интерфейс для NAT.

```
Router(config-if)#int fa5/0
Router(config-if)#ip nat outside
Router(config-if)#exit
```

Выполним ping с машины 192.168.1.2.



В режиме симуляции удостоверимся, что IP-адрес действительно подменяется.

Packet Tracer - C:\Users\Cepreij\Desktop\GB\nat.pkt

File View Tools Extensions Help

Back

PDU Information at Device: Router0

OSI Model Inbound PDU Details Outbound PDU Details

At Device: Router0
Source: 192.168.1.2
Destination: 80.80.80.100

In Layers

Layer7
Layer6
Layer5
Layer4

Layer 3: IP Header Src. IP: 192.168.1.2, Dest. IP: 80.80.80.100
ICMP Message Type: 8

Layer 2: Ethernet II Header
0050.0F19.E503 >> 000B.BEA0.2015

Layer 1: Port FastEthernet0/0

Out Layers

Layer7
Layer6
Layer5
Layer4

Layer 3: IP Header Src. IP: 70.70.70.70, Dest. IP: 80.80.80.100
ICMP Message Type: 8

Layer 2: Ethernet II Header
0060.7010.D206 >> 0001.972D.C6D2

Layer 1: Port(s): FastEthernet5/0

1. FastEthernet0/0 receives the frame.

Challenge Me

<< Previous Layer Next Layer >>

Event Viewer

Time(sec) Last Device At Device Type Info

Time(sec)	Last Device	At Device	Type	Info
0.000	--	192.168.1.2	ICMP	
0.004	--	192.168.1.2	ICMP	
0.005	192.168.1.2	Switch0	ICMP	
0.006	Switch0	Router0	ICMP	
0.007	Router0	Router1	ICMP	

Simulation Controls

Back Auto Capture / Play Capture

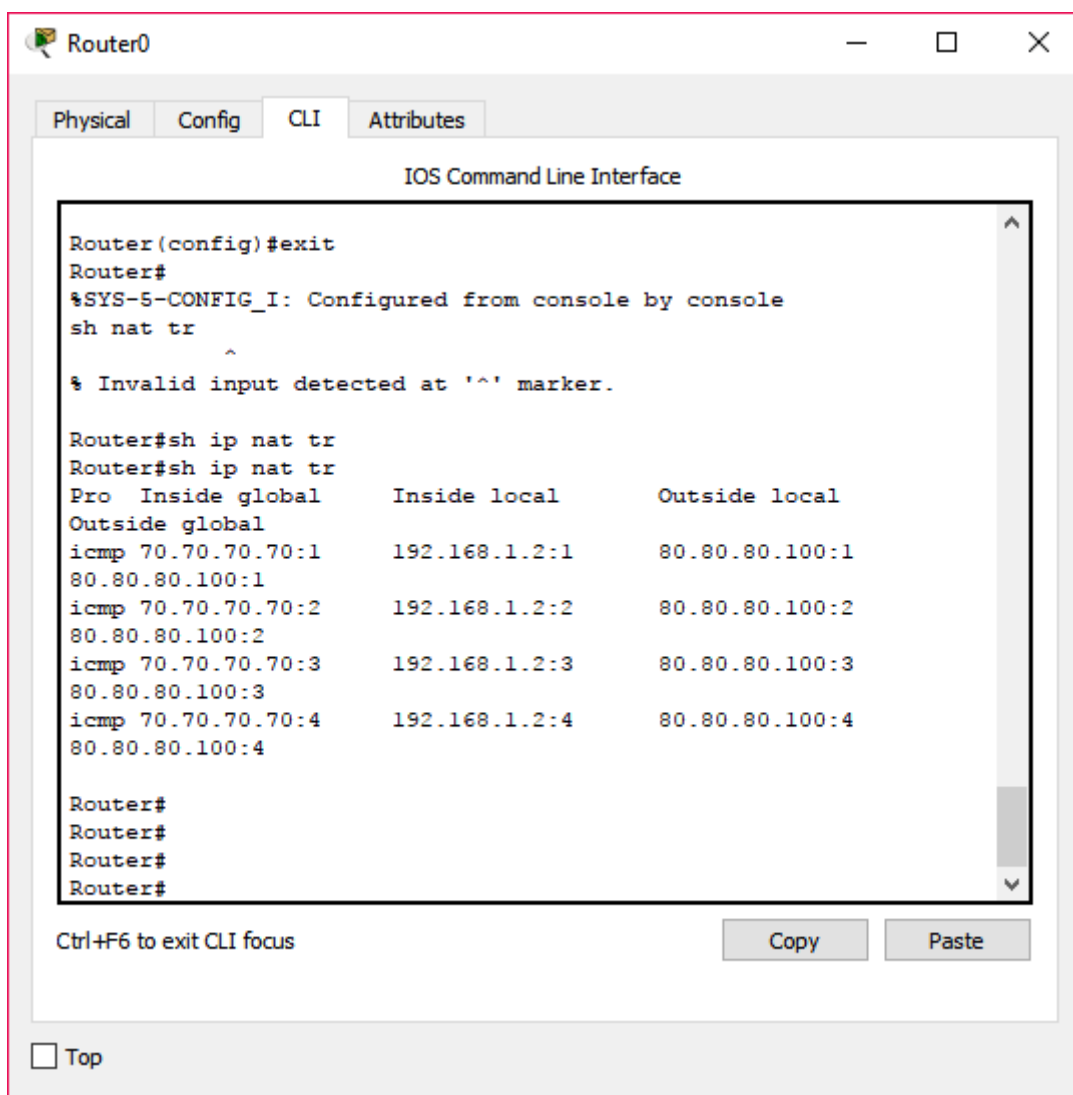
Event Filters - Visible Events

Edit Filters Show All

Может возникнуть интересный вопрос. А каким образом происходит трансляция ICMP, если в ICMP нет портов. Но в ICMP имеется номер последовательности, в трансляции он заменяется аналогично портам.

Мы можем в этом убедиться, если на маршрутизаторе посмотрим таблицу трансляций.

```
Router# sh ip nat tr
```



Самостоятельно настройте NAT-трансляцию (перегруженный NAT) и убедитесь, что подменяются не только IP-адреса, но и порты (зайдите на сервер с помощью браузера по протоколу HTTP, посмотрите пакеты в режиме симуляции, посмотрите на маршрутизаторе таблицу трансляции).

Настройка перегруженного NAT в Linux

В Linux NAT является частью механизма сетевого фильтра. Доступ к настройкам NAT осуществляется с помощью утилиты `iptables`, для которой надо указать, что мы работаем с таблицей `nat` (по умолчанию `iptables` работает с таблицей `filter`).

Предположим, что у нас имеется некая машина с белым адресом 185.195.27.164 и серым адресом 172.16.0.1, смотрящим в локальную сеть (на самом деле это туннельный адрес, но за которым действительно скрывается сеть с серыми IP-адресами).

```
root@tom3: ~  
  
ens3      Link encap:Ethernet  HWaddr 52:54:00:1f:f9:ce  
          inet addr:185.195.27.164  Bcast:185.195.27.255  Mask:255.255.255.0  
          inet6 addr: fe80::5054:ff:felf:f9ce/64 Scope:Link  
          inet6 addr: 2a04:5200:fff3::2b7/48 Scope:Global  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:537600039 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:3063901 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:32816339818 (32.8 GB)  TX bytes:567416562 (567.4 MB)  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:1921 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:1921 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1  
          RX bytes:169055 (169.0 KB)  TX bytes:169055 (169.0 KB)  
  
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  
-00  
          inet addr:172.16.0.1  P-t-P:172.16.0.2  Mask:255.255.255.255  
          inet6 addr: 2002:b9c3:lba4:cafe::1/64 Scope:Global  
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1420  Metric:1  
          RX packets:15556 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:23755 errors:0 dropped:37 overruns:0 carrier:0  
          collisions:0 txqueuelen:100  
          RX bytes:918534 (918.5 KB)  TX bytes:32918075 (32.9 MB)
```

Необходимо настроить машину так, чтобы пакеты, приходящие из сетевого интерфейса tun0, отправлялись в Интернет, но от имени 185.195.27.164.

Во-первых, нам придется включить IP-forwarding, чтобы можно было не только осуществлять маршрутизацию (отправку сообщений и в ens33, и в tun0 в зависимости от масок сетей), но и пересылку сообщений из одних сетей в другие. Посмотрим таблицу маршрутизации и включена ли пересылка в ядре операционной системы (выведем в консоль содержимое файла /proc/sys/net/ipv4/ip_forward из procfs, отображающего содержимое соответствующих данных ядра операционной системы).

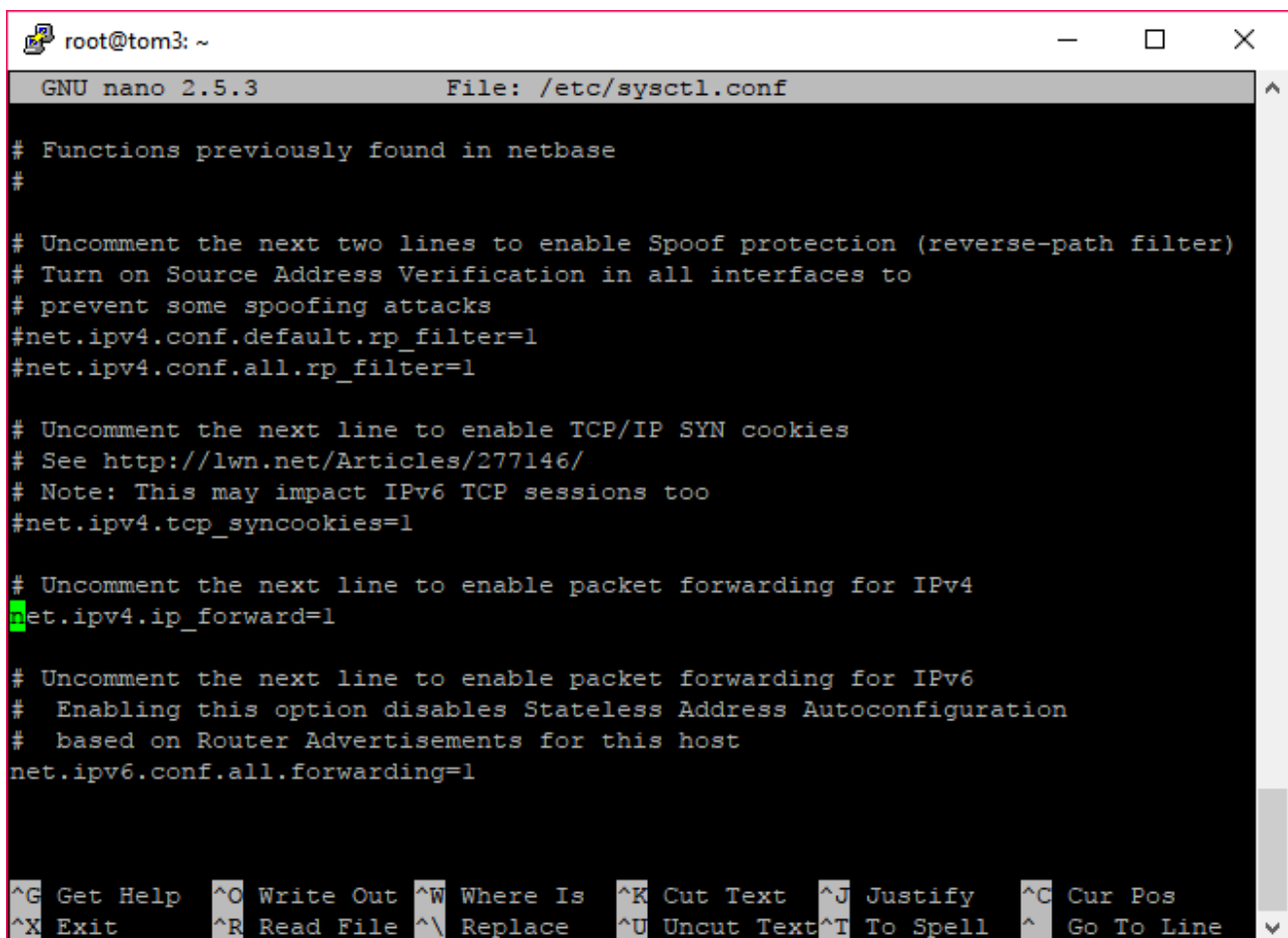

```
root@tom3: ~  
root@tom3:~# route  
Kernel IP routing table  
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface  
default          gw.firstbyte.ru 0.0.0.0         UG    0      0      0 ens3  
10.0.2.0         172.16.0.2     255.255.255.0   UG    0      0      0 tun0  
172.16.0.0       172.16.0.2     255.255.255.0   UG    0      0      0 tun0  
172.16.0.2       *              255.255.255.255 UH    0      0      0 tun0  
172.17.0.0       *              255.255.0.0     U     0      0      0 docker0  
localnet         *              255.255.255.0   U     0      0      0 ens3  
root@tom3:~#  
root@tom3:~# cat /proc/sys/net/ipv4/ip_forward  
0  
root@tom3:~# echo 1 > /proc/sys/net/ipv4/ip_forward  
root@tom3:~#
```

Видим, что пересылка пакетов выключена. Включим ее.

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Чтобы пересылку включить постоянно, нужно отредактировать другой файл (который хранится уже на диске) `/etc/sysctl.conf`.

```
# nano /etc/sysctl.conf
```



```
root@tom3: ~
GNU nano 2.5.3      File: /etc/sysctl.conf

# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
net.ipv6.conf.all.forwarding=1

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Найдем и заменим значение `net.ipv4.ip_forward=0` на `net.ipv4.ip_forward=1`.

Нажмем `Ctrl-O`, чтобы сохранить изменения, и `Ctrl-X`, чтобы выйти.

Выполним команду.

```
# iptables -t nat -A POSTROUTING -o ens3 -j MASQUERADE
```

Мы указываем, что в таблице NAT после выполнения маршрутизации (т.е. в цепочке POSTROUTING) для сообщений проходящих в сетевой интерфейс `ens3` (с белым адресом) мы будем выполнять маскаррад. При получении ответов трансляция будет выполняться обратно, так как Netfilter работает на сеансовом уровне модели OSI/ISO, отслеживая сессии и выполняя замену в обе стороны.

После этого можно проверить работу NAT, прописав в качестве шлюза `172.16.0.1` на других машинах и проверив, что интернет действительно работает, а также воспользовавшись сервисом `2ip.ru`, убедившись, что виден только внешний белый IP-адрес.

Мы выполняем трансляцию для всех сетей. С помощью ключа `-s` можно указать сеть, для которой мы хотим выполнять маскаррад.

Чтобы трансляция работала постоянно, необходимо выполнить команду.

```
# iptables-save >/etc/iptables.rules
```

(Убедитесь, что файл не существует, иначе предыдущее содержимое будет утеряно).

В файле будет строчка.

```
-A POSTROUTING -o ens3 -j MASQUERADE
```

Для Ubuntu 16, чтобы это работало при рестартах системы, в файл /etc/network/interfaces для соответствующего сетевого интерфейса нужно задать правило.

```
iface ens3 inet static
# пропущено несколько строк
pre-up iptables-restore < /etc/iptables.rules
```

Практическое задание

1. Работа в Wireshark. Запустить Wireshark, выбрать любой веб-сайт, определить IP-адрес сервера, отфильтровать в Wireshark трафик по этому IP-адресу. Набрать адрес сервера в строке браузера. Сколько TCP-соединений было открыто и почему. В работе можно использовать источник 1 из списка дополнительных материалов.
2. Настроить перегруженный NAT в предложенной схеме в Cisco Packet Tracer. С помощью режима симуляции удостовериться, что при подключении на веб-сервер происходит подмена IP-адресов и портов. Посмотреть таблицу трансляции NAT на маршрутизаторе.

Дополнительные материалы

1. Лихтциндер Б.Я., Поздняк И.С. Анализ трафика мультисервисных сетей / Учебное пособие для проведения лабораторно-практических занятий / под редакцией Карташевского В.Г., Рецензия Васина Н.Н. Поволжский государственный университет телекоммуникаций и информатики. Кафедра МСИБ. Самара - 2013. - 95 С. - Электронный ресурс [Режим доступа] URL: http://msib.psuti.ru/content/metod/Анализ_трафика_мультисервисных_сетей.pdf
2. Таненбаум Э., Уэзеролл Д. Т18 Компьютерные сети. 5-е изд. — СПб.: Питер, 2012. — 960 с. (Глава 6,7)

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы.

1. https://en.wikipedia.org/wiki/Network_Control_Program
2. http://citforum.ru/operating_systems/linux/ipsysctl/3.shtml
3. <http://computer-is-us.blogspot.ru/2008/08/how-network-address-translation-works.html>
4. <http://simple4ip.com/blog/rus/2012/04/%D0%BD%D0%BE%D0%B2%D0%B0%D1%8F-%D1%81%D0%B5%D1%80%D0%B0%D1%8F-%D1%81%D0%B5%D1%82%D1%8C-100-64-0-010/>

5. <http://www.itkitchen.net/%D0%BD%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0-nat-overload-%D0%BD%D0%B0-cisco/>
6. <https://webhamster.ru/mytetrashare/index/mtb0/28>