Виртуальный сетевой интерфейс в linux. TAP vs TUN

12/04/2016

Читатели, не нуждающиеся в теоретическом изложении концепции виртуальных сетевый интерфейсов Linux, могут сразу перейти к настройке по ссылкам:

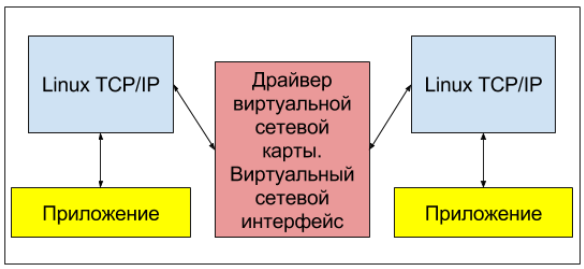
#Создаем виртуальный интерфейс в linux вручную

- #Создаем интерфейс типа tun с помощью ip
- #Создаем интерфейс типа tap с помощью ip
- #Создаем интерфейс типа dummy с помощью ip

#Создаем виртуальный интерфейс в linux с помощью systemd-networkd

- #Создаем интерфейс типа tun с помощью systemd-networkd
- #Создаем интерфейс типа tap с помощью systemd-networkd
- #Создаем интерфейс типа dummy с помощью systemd-networkd

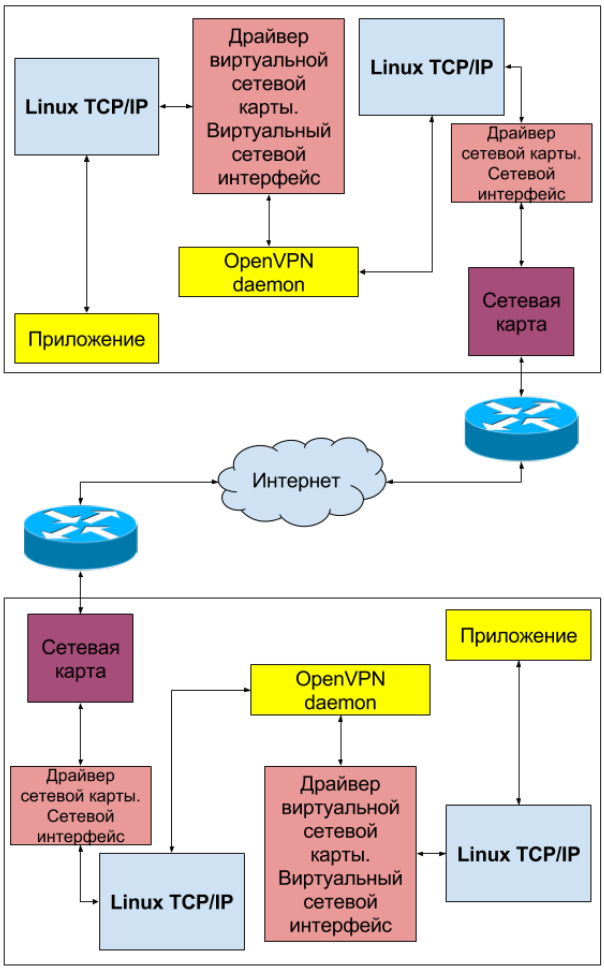
Создавать сетевые интерфейсы в linux нам позволяют различные модули ядра. Но там, где для реальных железных сетевых карт эти модули ядра, или как их еще называют - драйверы, обеспечивают прием данных от стека TCP/IP и их формирование уже в виде электрического сигнала на сетевой карте, драйверы виртуальных сетевых интерфейсов (loopback) могут лишь, приняв эти данные, отдать их какому-нибудь приложению для дальнейшей обработки. Такая функциональность может быть востребована, если на вашем сервере установлены программы, использующие стек TCP/IP для обмена данными и, понятно, не нуждающиеся в выводе этих данных в реальную сеть. Пример: веб-сайт на drupal связывается с базой данных, установленной на этом же сервере:



Другим распростаренным примером использования виртуальных сетевых интерфейсов (loopback) в linux может быть их использование для целей построения виртуальных частных сетей - VPN. Вы наверняка слышали о таких технологиях как OpenVPN, GRE, WireGuard и т.д. Каждый из этих демонов создает виртуальный сетевой интерфейс который служит для прозрачной маршрутизации данных между узлами, находящимися на удалении друг от друга и не имеющих возможности прямого взаимодействия. Рассмотрим общую сетевую топологию на примере OpenVPN:

Темы

- Cloud (1)
- Coreutils (5)
- Data Bases (2)
- DNS (1)
- Docker (2)
- Kubernetes (4)
- Linux (34)
- Network (6)
- SystemD (8)
- VPN (1)
- WEB (2)
- Windows (4)
- Виртуализация (4)



От используемого драйвера зависит тип интерфейса, его скорость, допустимый размер MTU и т. д. Совсем даже не обязательно, что загружать драйвер в ядро вам придется самостоятельно. Скорее всего, создавая интерфейс нужного типа, система сама подберет и загрузит требуемый драйвер. Вам лишь останется сконфигурировать уже работающий loopback интерфейс. В данной статье мы рассмотрим 3 возможных на конец 2016 года типа виртуальных интерфейсов в linux: tun, tap и dummy. Отличие интерфейсов tun и tap заключается в том, что tap старается больше походить на реальный сетевой интерфейс, а именно он позволяет себе принимать и отправлять ARP запросы, обладает MAC адресом и может являться одним из интерфейсов сетевого моста, так как он обладает полной поддержкой ethernet - протокола канального уровня (уровень 2). Интерфейс tun этой поддержки лишен, поэтому он может принимать и отправлять только IP пакеты и никак не ethernet кадры. Он не обладает MAC-адресом и не может быть добавлен в бридж. Зато он более легкий и быстрый за счет отсутствия дополнительной инкапсуляции и прекрасно подходит для тестирования сетевого стека или построения виртуальных частных сетей (VPN). Виртуальный интерфейс типа dummy очень похож на tap, разница лишь в том, что он реализуется другим модулем ядра.

Создаем виртуальный интерфейс в linux вручную

Создавать и удалять интерфейсы, назначать IP и MAC адреса, изменять MTU и многое другое нам помогает утилита ip. Пользоваться ip удобно и легко, но помните, что произведенные изменения будут потеряны после перезагрузки компьютера. Используйте ip в целях тестирования.

Создаем интерфейс типа tun

```
ip tuntap add dev tun0 mode tun
ip address add 192.168.99.1/30 dev tun0
ip address show tun0
2: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN
group default qlen 500
  link/none
  inet 192.168.99.1/30 scope global tun0
    valid_lft forever preferred_lft forever
```

Как видим у нас теперь есть виртуальный интерфейс с именем "tun0", у него есть IP-адрес, и ни слова о MAC-адресе - всё, как мы и рассчитывали. Его уже можно пинговать, и на нем уже можно запускать слушающие сервисы. Но что будет, если мы попытаемся добавить этот интерфейс в бридж?


```
ip link set dev tun0 master br0
RTNETLINK answers: Invalid argument
```

Команда ip логичным образом выдала ошибку - нет никакого смысла добавлять в бридж интерфейс, не обладающий поддержкой ethernet.

Создаем интерфейс типа tap

```
ip tuntap add dev tap0 mode tap
ip address add 192.168.99.5/30 dev tap0
ip address show tap0
3: tap0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
default qlen 1000
  link/ether d6:1c:67:cd:6f:80 brd ff:ff:ff:ff:ff:ff
  inet 192.168.99.5/30 scope global tap0
    valid_lft forever preferred_lft forever
```

У нас теперь появился новый виртуальный интерфейс с именем "tap0", у него есть как IP-адрес, так и MAC-адреса. Его также можно пинговать, и на нем также можно запускать слушающие сервисы. Команда, добавляющая интерфейс в бридж уже не выдаст ошибку, потому что это интерфейс, обладающий поддержкой ethernet:

```
ip link set dev tap0 master br0
```

Создаем интерфейс типа dummy

```
ip link add dev dum0 type dummy
ip address add 192.168.99.9/30 dev dum0
ip address show dum0
4: dum0: <BROADCAST,NOARP> mtu 1500 qdisc noop master br0 state DOWN group
default qlen 1000
  link/ether 1a:37:3b:0f:da:be brd ff:ff:ff:ff:ff:ff
  inet 192.168.99.9/30 scope global dum0
    valid_lft forever preferred_lft forever
```

Вы наверняка заметили, что команда для добавления интерфейса изменилась. Ничего необычного. Так написана утилита "ip". Ну и

конечно, виртуальный интерфейс типа dummy можно легко добавить в бридж:

```
ip link set dev dum0 master br0
```

Создаем виртуальный интерфейс в linux с помощью systemd-networkd

В systemd-networkd за создание интерфейсов отвечают одни конфигурационные файлы, имеющие суффикс ".netdev", а за их настройку другие, имеющие суффикс ".network". Соответственно нам понадобиться в /etc/systemd/network создать по паре конфигурационных файлов для каждого из исследуемых типов интерфейсов

Создаем интерфейс типа tun

Создадим соответственно файлы tun0.netdev с содержимым:

```
[NetDev]
Name=tun0
Kind=tun
```

и tun0.network:

```
[Match]
Name=tun0

[Network]
Address=192.168.98.1/30
```

Создаем интерфейс типа tap

Создадим соответственно файлы tap0.netdev с содержимым:

```
[NetDev]
Name=tap0
Kind=tap
```

и tap0.network:

```
[Match]
Name=tap0

[Network]
Address=192.168.98.5/30
```

Создаем интерфейс типа dummy

Создадим соответственно файлы dum0.netdev с содержимым:

```
[NetDev]
Name=dum0
Kind=dummy
```

и dum0.network:

```
[Match]
Name=dum0
```

```
[Network]
Address=192.168.98.9/30
```

Стоит отметить, что если вы планируете маршрутизировать траффик через виртуальные интерфейсы (а, используя их для цели создания виртуальных частных сетей (VPN), вы точно этого хотите), то в конфигурационный файл в секии "Network" следует добавить диррективу "IPForward=yes".

Темы: Linux Network VPN SystemD

Поделиться: