Linux 网络管理基础

红岩网校工作站运维安全部 黄凯升

本节内容

- 1. 一些基础知识
- 2. 网络配置基础
- 3. 路由的基本原理与配置
- 4. Netfilter 子系统的构成与 iptables 的使用
- 5. 常用的网络调试工具

基础知识

网关/路由器

- 网关是网络层上实现网络互连的设备，可将数据包跨子网传输。
- 狭义上的路由器就是有路由功能的设备。
- 相同子网下的主机可以通过主机号直接互连，不需要经过网关。跨子网的主机则需要网关才能进行互连。

IP 与 CIDR

- IP：OSI 模型第三层网络层进行通讯的协议，目前有 IPv4 和 IPv6 两个版本。
- CIDR：“无类别域间路由”是一个用于给用户分配 IP 地址以及在互联网上有效地路由 IP 数据包的对 IP 地址进行归类的方法。

  传统的基于 A、B、C 类地址进行的子网划分已经不能满足需求，

  人们提出了可变长子网掩码来进行任意长度的前缀划分，

  从而可以对地址块进行精细的划分与聚合。

IP 与 CIDR

格式：
CIDR 描述的地址块使用 IP/PREFIXLEN 的格式描述。

例如，一个传统的 C 类地址 192.168.1.123，子网掩码为 255.255.255.0，使用 CIDR 可以写成 192.168.1.123/24。

其中，24 指前缀长度，即网络号在地址中的长度。

IP 与 CIDR

“前缀长度”？
我们知道，对于 192.168.1.123 这个 IPv4 地址和 255.255.255.0 这个子网掩码。
二进制表示如下：

```
11000000 10101000 00000001 01111011          (192.168.1.123)

11111111 11111111 11111111 00000000          (255.255.255.0)
```

子网掩码只能由高位的 1 和低位的 0 组成，因此前缀长度唯一对应一个子网掩码。
可以发现该子网掩码中，1 的个数为 24 个，即前缀长度。

IP 与 CIDR

根据以上的 IP 和前缀长度，可以匹配出网络号为：

```
 (11000000 10101000 00000001 01111011) &
 (11111111 11111111 11111111 00000000) =
 (11000000 10101000 00000001 00000000) = 192.168.1.0
```

主机号为：

```
 (11000000 10101000 00000001 01111011) &
~(11111111 11111111 11111111 00000000) =
 (00000000 00000000 00000000 01111011) = 0.0.0.123
```

IP 与 CIDR

通过前缀长度或子网掩码，我们就可以从一个任意给定的 CIDR 块中匹配出网络号和主机号，接下来无论是进行路由还是其它操作都非常简单。

在 IPv4 中前缀长度的取值范围是 $[0, 32]$，这是因为 IPv4 只有 32 位。

例：重庆邮电大学所拥有的公网 IP 地址范围为 202.202.0.0 – 202.202.47.255，可以用两个 CIDR 块描述：

202.202.0.0/19	202.202.0.0 -	202.202.31.255
202.202.32.0/20	202.202.32.0 -	202.202.47.255

IPv6 —— 下一代 IP 地址

- IPv6 是下一代的 IP 地址，长度为 128 位。
- 比起 32 位的 IPv4，IPv6 可以表示的地址数量达到了 $2^{128} \approx 3.40 \times 10^{34}$，而 IPv4 仅能表示约 42 亿个地址。
- IPv4 使用点分十进制表示，IPv6 使用冒号分割十六进制的形式，分为八组，每组 16 位长度。连续的 0 可以使用 :: 进行压缩（一个地址中只能压缩一次）。

  例：重庆邮电大学官网的 IPv6 地址 `2001:da8:c807:20:202:202:32:60`
- IPv4 现在已经枯竭，最后一块 IPv4 地址于 2019 年 11 月 26 日被分配。

网络地址转换 (NAT)

- 网络地址转换是一种部署在网关上的伪装技术。
- 我们可以从一个方面把它分为以下类型：
  - 基本网络地址转换：
    - 仅支持地址转换，端口不变。
    - 一对一映射，因此需要维护一个外部地址池。
  - 网络地址端口转换 (NAPT)：
    - 目前应用最广泛的 NAT 类型，同时转换地址和端口号。
    - 由于依赖于传输层端口号的变化，支持的传输层协议有限 (通常为 TCP 和 UDP)。
    - 可以多对一映射，外部需要的地址可以较少。
    - 可分为锥形 NAT 和对称 NAT。

网络地址转换 (NAT)

- 我们还可以从另一个方面把它分为以下类型：
  - 源地址转换 (SNAT)：
    - 将经过 NAT 的数据包源地址以及可能的源端口进行修改，伪装成网关的地址和对应的端口。同时保持目标地址与端口不变。
  - 目标地址转换 (DNAT)：
    - 将发往网关的数据包目标地址以及可能的目标端口进行修改后，发往根据规则指定的主机。同时保持源地址与端口不变。
- 通常同时使用 SNAT 和 DNAT 配合来进行网关上的端口映射。

一些特殊的路径

- /etc/resolv.conf　控制系统使用的 DNS 服务器、搜索域
- /etc/hostname　控制系统的主机名
- /etc/hosts　本地域名映射表，在域名解析过程中优先级最高
- /etc/networks　定义一些网络的符号名称
  (如 169.254.0.0 代表链路本地)
- /etc/sysctl.conf　控制 Linux 内核的一些参数

Linux 网络接口

- 本地环回接口
- 以太网接口
- 点对点接口

以上几个是常见的几种类型的网络接口。

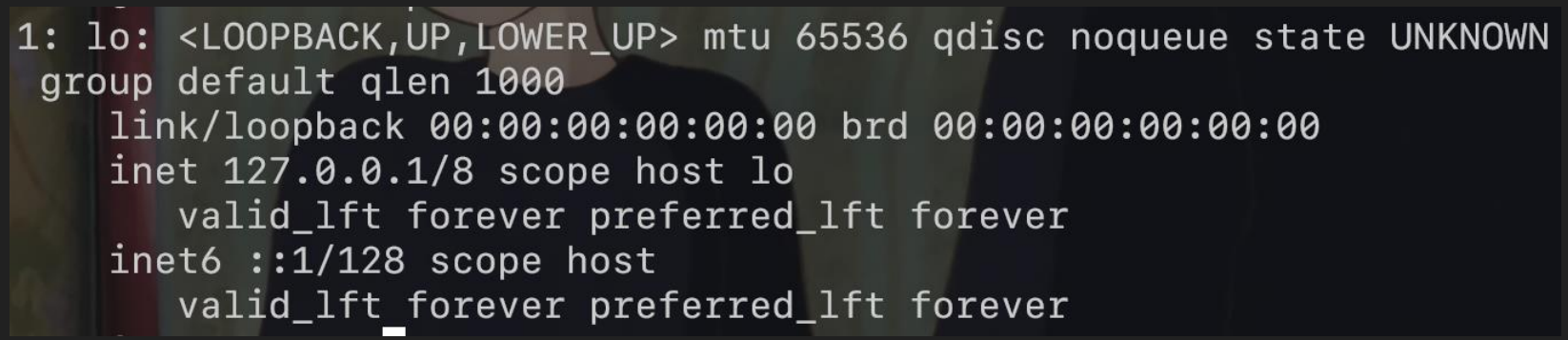
Linux 网络接口

共同点：

- 都在系统中有一个设备名
- 都有 UP 或 DOWN 两种状态，即活跃与不活跃状态
- 都有 MTU 属性，即最大传输单元，长度大于 MTU 的数据帧会被分段发送
- 都有 flags 属性，在 flags 属性中定义该接口的类型与所支持的协议

Linux 网络接口

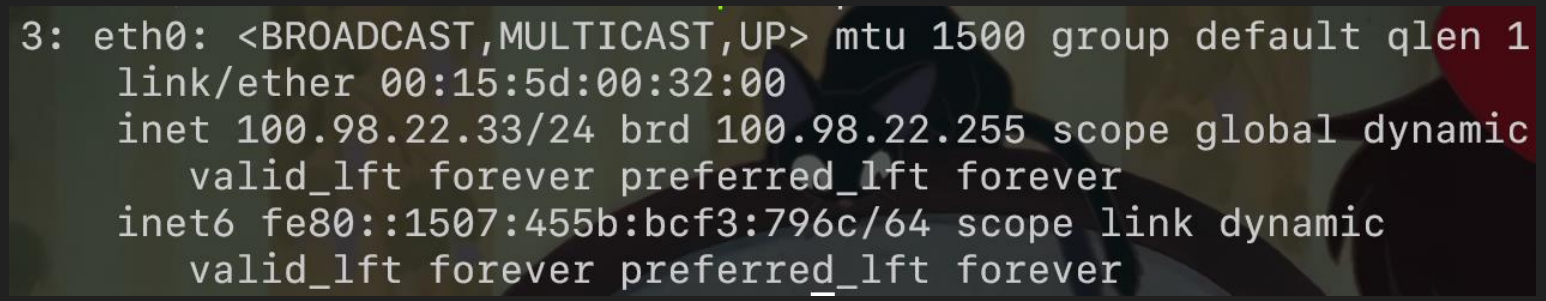
本地环回接口



- 通常接口名为 lo
- MAC 地址为 00:00:00:00:00:00
- IPv4 通常为 127.0.0.1/8，IPv6 通常为 ::1/128
- 特别的，所有目标地址为 127.0.0.0/8 内的包均会被 lo 接口捕获
- 适用于本机程序之间的网络访问，流量不会经过任何网卡

Linux 网络接口

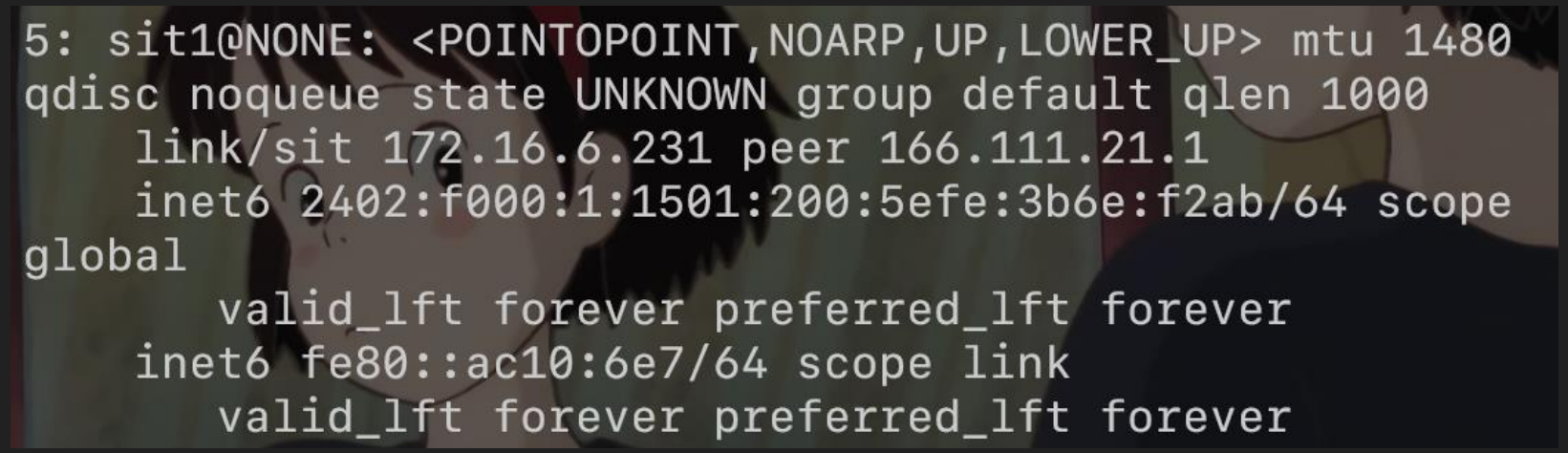
以太网接口



- 接口名没有特定的规则
- 数据链路层确保使用 MAC 地址
- 支持 IPv4/IPv6 网络层协议，通常有一个 fe80:: 开头的 IPv6 链路本地地址
- 路由实现通常都为
- 以太网接口的实例：物理网卡、TAP 设备、Bridge、VLAN 设备、Bond、VETH 等

Linux 网络接口

点对点接口



- 接口名没有特定的规则
- 数据链路层实现不定
- 网络层协议支持不定
- 点对点接口的实例：许多 VPN (如 PPTP、L2TP、WireGuard)、TUN 设备以及一些隧道 (如 IPIP、ISATAP/SIT)

网络配置基础

以 Ubuntu 20.04 LTS 为例

网络配置工具概述

- 我把网络配置工具分为两种：
  - 可对当前网络状态进行即时修改、调试的工具。例如传统的 net-tools，现代的 iproute2 等。
  - 配置可持久化，系统启动时可自动应用保存的配置，或以服务形式常驻运行。可作为一个统一管理网络配置的平台的。例如 Ubuntu 20.04 中常见的 NetworkManager 和 netplan。

iproute2 的使用

- 现代 Linux 使用 iproute2 取代了在 Unix-like 操作系统中广泛使用的 net-tools 系列工具。
- 我们主要使用 iproute2 包中的 `ip` 命令来操作网络。

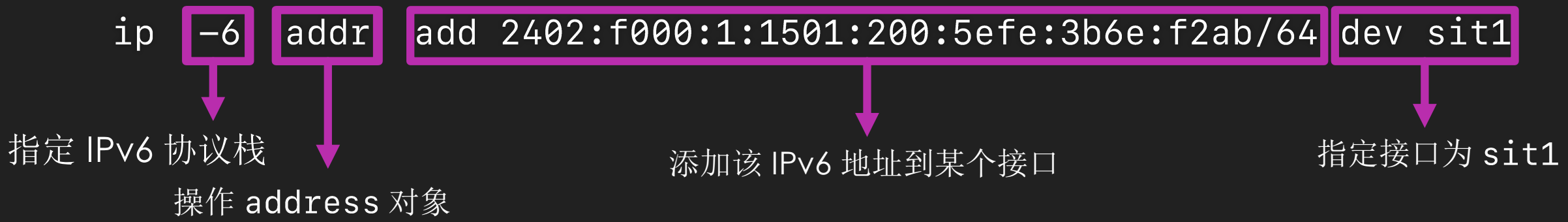
iproute2 的使用

- `ip` 命令非常强大，本节课可能会涉及到的操作对象有：
  - `address`：对网络接口上的 IP 地址进行操作
  - `route`：对系统路由表进行操作
  - `link`：对网卡接口链路进行操作
  - `rule`：对系统处理数据包的策略进行操作
- `ip` 命令可以通过 `-4` 和 `-6` 参数指定使用的是 IPv4 还是 IPv6 协议栈。
- `ip` 命令支持操作对象的前缀缩写，出现歧义时字典序靠前的对象被执行：
  - 例：`ip r`。由于 route 比 rule 字典序靠前，因此等价于执行 `ip route`。
- `ip` 命令的操作对象不带任何命令执行时即表示为输出。

iproute2 的使用

例子：

```
ip -6 addr add 2402:f000:1:1501:200:5efe:3b6e:f2ab/64 dev sit1
```



可以看出，一条 ip 命令的构成主要有 ip 命令本身的参数、操作对象和应用于操作对象的数条子命令构成。而这些子命令可能需要参数，如 add 和 dev 命令就需要传入参数。

其中，有些子命令类似“谓语”，表示动作，如 add。有些则类似“定语”，起限定、修饰的作用，如 dev。

iproute2 的使用

- 授人以鱼不如授人以渔，我们来看看如何自学 `ip` 命令的使用。
- `ip` 命令本身和任何一个操作对象都支持 `help` 子命令，这个子命令将会输出一个帮助信息。
- 还可以通过 `man ip` 来查看更为详细的帮助。

- 以下是执行 ip 或 ip help 取得的输出。

方括号代表可选，在这里给出了名为 OPTIONS 的指令集合

必需参数，同时也给出了一个指令集合

花括号代表一个指令集合，里面包含了可以使用的指令。指令集合可以嵌套。

多条 Usage 表示有多种使用方法。

```
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | address | addrlabel | route | rule | neigh | ntable |
                   tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm |
                   netns | l2tp | fou | macsec | tcp_metrics | token | netconf | ila |
                   vrf | sr | nexthop }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -h[uman-readable] | -iec | -j[son] | -p[retty] |
                    -f[amily] { inet | inet6 | mpls | bridge | link } |
                    -4 | -6 | -I | -D | -M | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
                    -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename] |
                    -rc[vbuf] [size] | -n[etns] name | -N[umeric] | -a[ll] |
                    -c[olor]}
```

where 部分给出了各个“指令集合”中包含的内容。
这里我把 OBJECT 或 OPTIONS 等称之为“指令集合”。

net-tools 概述

- 在早期 Linux 发行版和其它 Unix-like 操作系统如 FreeBSD、macOS 中被广泛使用。
- 常用组件有 `ifconfig`、`route` 等。
- 尽管已经不再推荐在现代 Linux 中使用，但 net-tools 中有些组件因为某些优点或历史原因在现在仍被使用。例如 `ifconfig` 输出接口的信息更加详尽。
- 本节课不过多讲解 net-tools 的使用。

NetworkManager 的使用

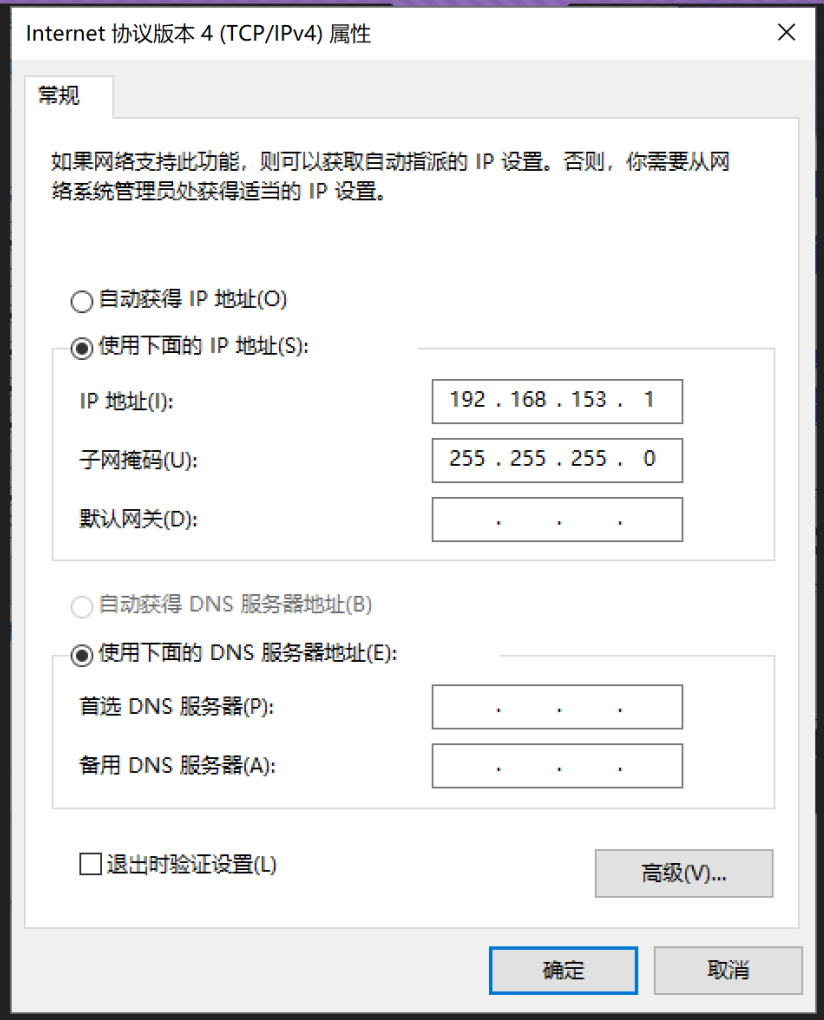
NetworkManager 是一个十分强大的网络管理器。它功能强大，上手较为简单。但面对复杂的网络配置时显得较为繁琐。在命令行下提供了 `nmtui` 这一人类友好的文本 UI 界面。

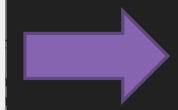
NetworkManager 是基于连接的。这意味着它也可以比较方便地去创建 VLAN 设备、桥接、VPN 等等。同时，它也是 Linux 下最常用的无线网络管理器。

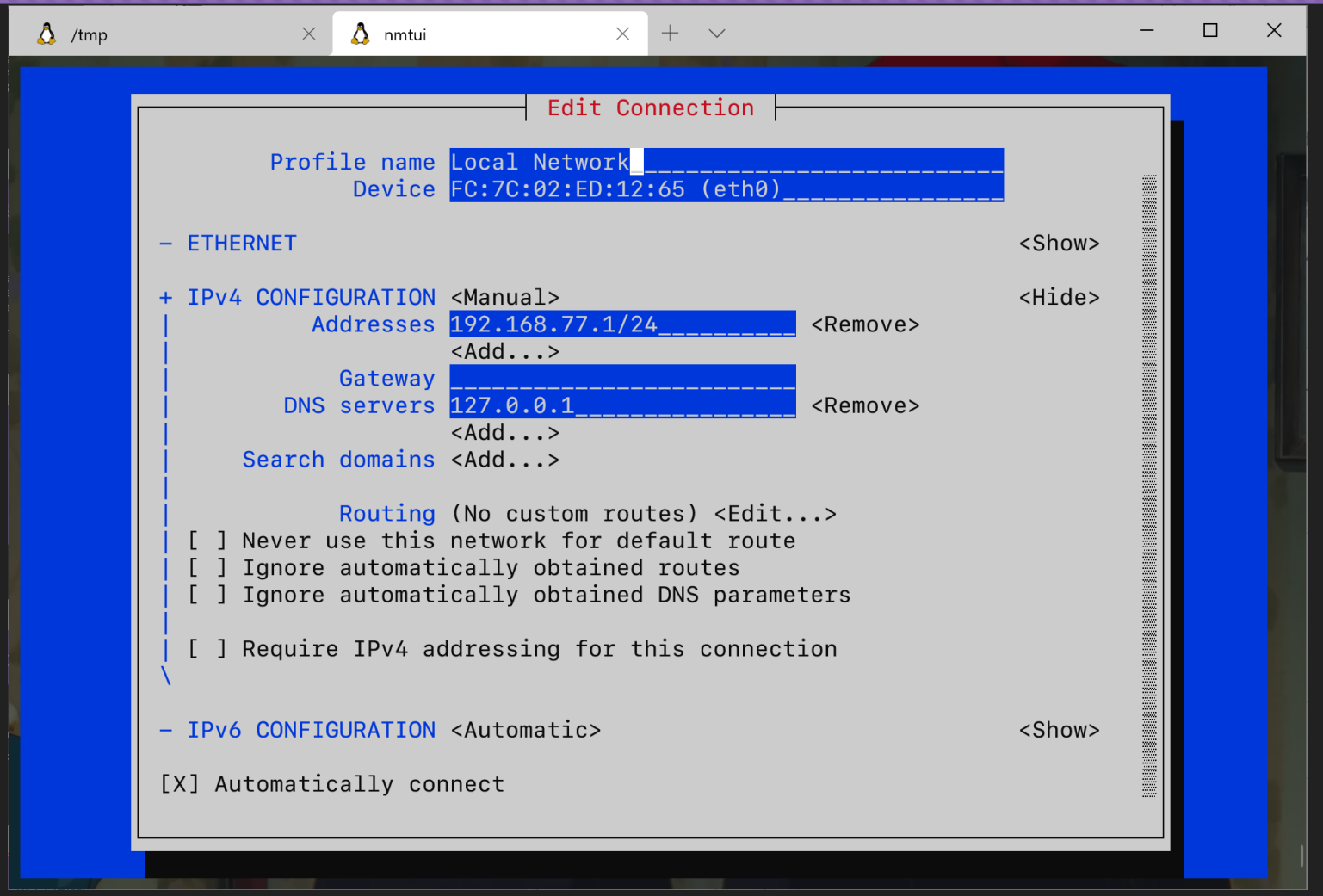
在 Ubuntu 桌面版中，NetworkManager 是默认的网络管理器。

`nmtui` 操作起来比较简单，下面将它与 Windows 网络连接界面做了一下对比，可以看到它很容易上手。

NetworkManager 的使用







Netplan 的使用

NetworkManager 固然很好，但是它本身较重。而且基于连接的配置方法在很多场景配置起来并不优雅。

在服务器版的 Ubuntu 20.04 中，netplan 是默认的配置工具。

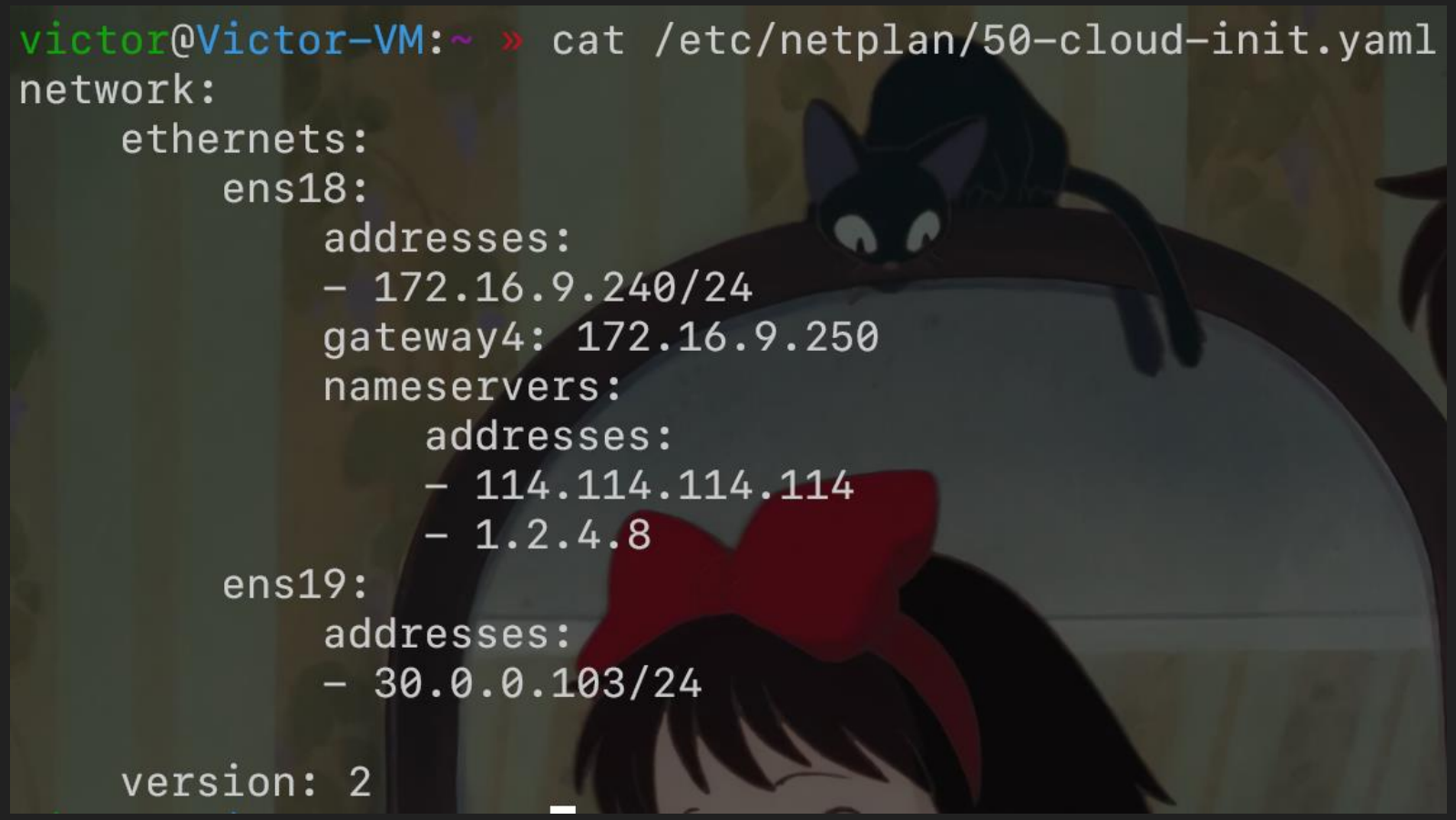
它的配置文件编写使用 YAML 格式。对人类较为友好。

实际上 netplan 只是一个 renderer，
它的后端可以是 system-networkd 或者 NetworkManager。

Netplan 的使用

实例：

- 将 ens18 接口配置为：
  - IPv4 地址 172.16.9.240/24
  - IPv4 默认网关 172.16.9.250
  - DNS 为 114.114.114.114 和 1.2.4.8
- 将 ens19 接口配置为：
  - IPv4 地址 30.0.0.103/24



Netplan 的使用

- 使用 netplan 命令来管理配置，这里讲两个常用的子命令：
  - apply：直接执行配置。
  - try：先尝试配置，一定时间内没有收到反馈就将之前的配置恢复回去。
    有效避免远程调网调炸了直接失联的情况（
- 完整的使用方法可以 netplan help 或者 man netplan 查看。

其它网络管理器的使用

- 在传统的 Ubuntu / Debian 中，使用 ifupdown 管理网络，配置文件位于 `/etc/network/` 。
- 在传统的 CentOS/RHEL 系统中，网络管理由 network 服务管理，配置文件位于 `/etc/sysconfig/network-script/` 或 `/etc/sysconfig/network`。

路由的基本原理与配置

路由

- 路由是对相同子网或不同子网间传输的数据包进行路径选择的过程。
- 路由决定了数据包接下来将被发送到哪里。
- 计算机中将数据包进行路由所依据的是路由表。

路由的过程

- 对于一个特定的数据包，系统会首先提取出它的目的地址(destination)
- 对于这个 destination，系统查询路由表，以一定的规则命中最合适的路由规则，然后按照这个路由规则转发数据包，转发时 TTL -1。
  - TTL 是 Time-To-Live 的缩写，它决定了一个包最多被转发多少次。
- 如果无法命中任何可用的路由，则抛出错误 (Linux 中为网络不可达)

路由表

- 路由表是一系列路由组成的集合
- 一条路由的基本要素有：目的地址、下一跳地址、源地址、接口、Scope (作用域)、类型、Metric (代价)
  - 这些要素未必需要全部满足，例如在链路上的路由可能就没有下一跳地址，通常的路由也不指定源地址
- Linux 下的路由表还有一些其它属性，我们会在演示时介绍

Scope

- Scope 决定了路由表在什么情况下生效，Linux 下有三种 scope
  - global：在任意情况下都生效
  - link：只在链路上生效，即在同一数据链路层时才生效
  - host：只在本机内部转发时生效

类型

- 路由表有几种的类型：
  - unicast/multicast/broadcast/anycast：单播/多播/广播/任播路由
  - local：本机回环路由
  - blackhole：路由黑洞，丢弃所有发往黑洞路由包
  - unreachable：不可达路由，返回目标主机不可达消息

以上仅为几种常见的路由表类型。

路由选择的原则

- 在 Linux 中路由选择的原则是：
  - 优先命中目标网络范围更小的路由。
  - 有 metric 的情况下，以 metric 更小的优先。(不指定 metric 的路由，metric 为 0)
  - default 路由仅在其它路由都命中失败时才会进行命中。

一些特殊的路由

- 并非所有的情况下都有一个网关 IP，例如对于一些点对点接口，它的路由下一跳可能是“在链路上”的。
  - 例如 PPP 连接的网关 IP 写为本机，实际上在 PPP 连接上发送的包是直接通过隧道传递到另一方的 peer 的。这个过程与标准的路由过程就有着较大的差异。
- 默认（default）路由，等于是使用 0.0.0.0/0 作为目的地址。
- 系统会自动为每一个接口创建一个基于该接口本机 IP 与子网掩码的在链路上路由。

路由的配置

- 使用 iproute2 配置路由十分方便，如果我们要让 10.0.0.0/8 都通过下一跳地址 172.22.161.1 路由出去，我们只需一条指令：
  - ip route add 10.0.0.0/8 via 172.22.161.1
- 这里的要求是 172.22.161.1 必须与本机某个网卡上的 IP 处于同一子网。

Netfilter 概述

- Netfilter 是内置于 Linux 内核中的一个子系统，它实现了对网络相关的多种自定义操作，并且允许内核模块注册为回调函数实现插件的编写。
- Netfilter 子系统在用户态的操作工具有 iptables/ip6tables、ebtables、arptables、ipset 和 nftables。上述工具作用的各不相同，本节课主要讲述 iptables 的使用。
- Netfilter 子系统提供了一些钩子，iptables 等包过滤框架通过将函数注册到这些钩子上工作。

iptables 概述

- iptables 是目前被广泛使用的包过滤框架，由内核模块 ip_tables 实现。
- iptables 由四个表 (后来多了一个 security 表)和五条链构成。
- iptables 是有状态的防火墙。通过连接跟踪机制，使得包作为连接的一部分而不是孤立的存在被处理。
- iptables 规则除了所属的链和表分为两部分：
  - 匹配 (match)：匹配系统可以支持多种方式进行匹配，满足匹配条件的包才会应用本条规则
  - 目标 (target)：对包执行的操作，可以是某一种操作，也可以是把包送入某个链

iptables 概述

- 在相同表、相同链下，规则执行的顺序是它们编号的顺序。
- 目标分为终止目标和非终止目标，终止目标，终止目标会阻止之后的规则被执行。

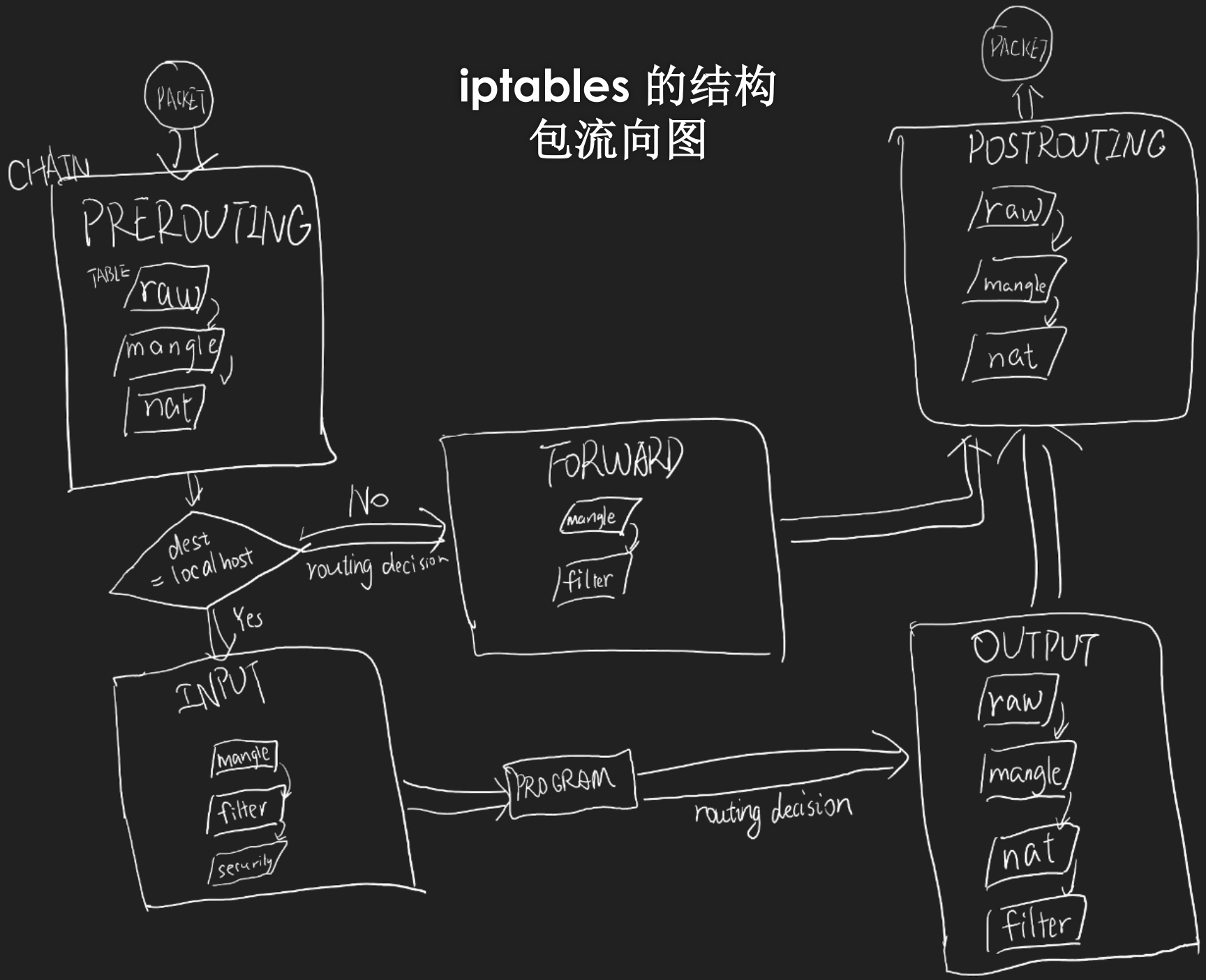
iptables 的结构

- 链：
  - PREROUTING：所有进入本机的包在路由选择前进入的链，
  - INPUT：进入本机且目的地址为本机的包将进入的链
  - FORWARD：进入本机且目的地址不为本机的包将进入的链
  - OUTPUT：仅本机发出的包将进入的链
  - POSTROUTING：所有从本机出去（包括本机发出与经过本机）的包将进入的链

iptables 的结构

- 表：
  - raw：优先级最高，在连接跟踪前处理，可以阻止包进入连接跟踪阶段和其它表。可用于 PREROUTING 和 OUTPUT 链。
  - mangle：用于对包 IP 头信息进行修改的表。可用于所有链。
  - filter：默认表，用于包过滤，决定一个包是否能通过。可用于 INPUT、FORWARD、OUTPUT 链。
  - nat：用于进行网络地址转换，可用于除 FORWARD 的所有链，但用于不同链时的意义可能不同。
  - security：用于对包设置强制访问控制标志处理等等（与 SELinux 有关本次课程不提及）

iptables 的结构
包流向图



iptables 的结构

- 以下介绍一部分常用的匹配：
  - `[!] -i/-o dev`：数据包是否来自某个网络适配器/从某个网络适配器出去
  - `[!] -s/-d ip/cidr`：数据包是否来自某个子网/去往某个子网
  - `[!] -p proto`：数据包是否属于某个传输层协议 (如 tcp、udp)
  - `-m extension ...`：通过扩展的匹配插件进行数据包匹配

iptables 的结构

- 以下介绍一部分常用的目标：
  - RETURN：仅在被调用的链中使用，返回到上一层链 (类似函数 return)。
  - ACCEPT：准许数据包通过。
  - DROP：丢弃数据包。

  } 只有这两个目标可以作为策略使用

  - REJECT：拒绝数据包，发送连接被拒绝信息。
  - REDIRECT：将数据包直接重定向到某个主机:端口，但不改变包。
  - SNAT/DNAT：对包进行源地址转换/目标地址转换。
  - MASQUERADE：伪装，根据出口地址自动进行网络地址转换。
  - MARK：在 mangle 链使用，对包打防火墙标记，该标记仅在本机范围内有效。
  - NOTTRACK：在 raw 链使用，阻止包进入连接跟踪。

iptables 的结构

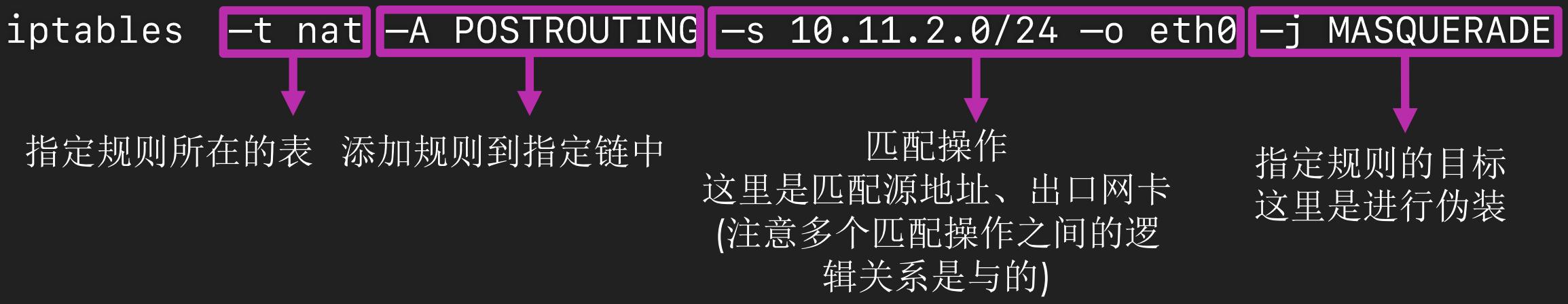
- 包状态：
  - NEW：如果到达的包关连不到任何已有的连接，但包是合法的，就为这个包创建一个新连接。对 面向连接的的协议例如 TCP 以及非面向连接的的协议例如 UDP 都适用
  - ESTABLISHED：当一个连接收到应答方向的合法包时，状态从 NEW 变为该状态。
  - RELATED：包不属于已有的连接，但是和已有的连接有一定关系。
  - INVALID：包不属于已有连接，并且因为某些原因不能用来创建一个新连接，例如无法识别、无法路由等等
  - UNTRACKED：如果在 raw 表中标记为 UNTRACKED，包将不会进入连接跟踪
  - SNAT：包的源地址被 NAT 修改之后会进入的虚拟状态。连接跟踪系统据此在收到 反向包时对地址做反向转换
  - DNAT：包的目的地址被 NAT 修改之后会进入的虚拟状态。连接跟踪系统据此在收到反向包时对地址做反向转换

iptables 命令格式

- 先介绍一部分常用的操作：
  - -A：添加规则　　将规则添加到某个表的某个链中
  - -C：检查规则　　检查该规则是否存在于某个表的某个链
  - -D：删除规则　　从某个表的某个链中删除规则 (可指定规则序号)
  - -I：插入规则　　将规则插入到某个表的某个链中的某个位置 (默认为最靠前位置)
  - -R：替换规则　　替换某个表的某个链中某个位置的规则
  - -L：列出规则　　列出某个表的某个链中的规则 (可选列出序号)
  - -F：清空规则　　清空某个表的某个链中的所有规则
  - -P：配置策略　　设置某个表某个链的默认动作 (只能是 ACCEPT/DROP)
  - -N/-X/-E：创建/删除/更名自定义链

iptables 命令格式

例子：



这条规则的意思是，对于来自 10.11.2.0/24 的包，在路由决策后，允许自动将数据包的源地址伪装成网关在 eth0 接口的地址，也就是进行了 NAT。这个操作在路由器中非常常见。

辅助命令

- `iptables-save`：输出当前应用的 iptables 规则到标准输出或文件
- `iptables-restore`：从文件或标准输入恢复 iptables 规则。注意它会将恢复的规则完全覆盖当前的规则。

实战：制作一个 IPv4 NAT 网关

- 这里我们进行实战，亲手制作一个 NAT 网关以供其它虚拟机上网。同时我们希望使用手机热点的网络访问 Internet，使用本地访问重邮内网。
- 温馨提示：在进行网关配置的时候，请一定注意每一步会引发哪些结果。否则在生产环境中，可能会导致未预期的访问。

实战：制作一个 IPv4 NAT 网关

- 首先，我们需要打开 Linux 的 IPv4 包转发，在 `/etc/sysctl.conf` 中，修改 IPv4 的包转发属性：
- `net.ipv4.ip_forward=1`
- 使用 `sysctl -p` 应用更改。
- 安全提醒：打开包转发意味着在 iptables 允许的情况下，任意目标地址不为本机的包都可以根据路由表被转发到对应的接口。

实战：制作一个 IPv4 NAT 网关

- 接下来我们要做的事情有：
  - 通过 netplan 配置网卡 IP 地址。
  - 配置自定义路由表以访问重邮内网。
  - 配置 iptables 规则进行 NAT。
- 接下来，我们就进入实战。

常用的网络调试工具

本节将进行较少的幻灯片展示，而是以演示为主

Ping

- Ping 是最常用的网络调试工具，它通过发送 ICMP Echo 消息探测目标主机是否可以连通。
- Ping 在各种系统中都有实现。其基本语法为 `ping destination`。
- Windows 中的 ping 默认发送 4 个包，Linux 中的 ping 会持续发包直到手动中断。使用 `-c` 参数可以指定发包数量。
- Ping 的结果并不能保证有意义。Ping 通不意味着目标存活，Ping 不通也不意味着目标不存活。
  - 例如：开启 Windows 防火墙的主机不响应 ICMP 消息，因此表现为 ping 不通。某些特殊的网络环境中 ICMP 消息可能被伪造，因此即使表现为 ping 通也不意味着存活。

Traceroute / MTR

- Traceroute 是进行路由跟踪的工具，它能够探测出本机和对方主机之间的路由链路上经过的路由节点。
- 原理：数据包头部有一个 TTL，决定数据包可以被最多转发多少次，Traceroute 发送 TTL 从 1 递增的包，则在路由链路上总会出现在某个节点处 TTL 减到 0，该节点会返回一个 ICMP 错误 Time to Live Exceeded 信息，在这个信息中就包含了该节点的 IP 地址。由此便可得知各个节点的地址。
- 如果路由链路上某些路由器不响应 ICMP 消息或不遵守规范进行 TTL -1，则无法探测到对应节点。
- MTR 也可以进行路由跟踪，但它同时还可以计算丢包率和延时均值。

Tcpdump

- Tcpdump 是一个抓包工具。
- Tcpdump 不同于其名，它不是只能抓 TCP 包，而是可以抓取某个网卡上经过的所有数据包。甚至数据链路层的包它也可以抓取。
- Tcpdump 使用 -i 参数指定待抓取的网卡，默认以日志形式在终端中输出。我们也可以使用 –w 参数让它将抓到的包一并输出为 pcap 格式。
- 使用 pcap 格式我们即可在 Wireshark 等软件中方便地分析数据包。

Netstat

- Netstat 是一个十分强大的网络调试工具。它可以查看系统路由表、网络分析日志、socket 状态等等。
- 尽管 iproute2 在一些功能如查看路由表等比起 netstat 要方便得多，但 netstat 的部分功能仍是不可替代的。

lsof

- lsof 其实并不是一个网络调试工具。它的作用是列出系统中打开的文件 (list opened files)。
- 但实际上，通过 `-i` 参数，lsof 可以用于查看系统中监听的 TCP/UDP 端口。以及监听此端口的程序。

扩展阅读

- https://man7.org/linux/man-pages/man8/ip.8.html
- https://developer.ibm.com/zh/technologies/linux/tutorials/l-lpic1-109-3/
- https://man7.org/linux/man-pages/man8/iptables.8.html
- https://man7.org/linux/man-pages/man8/iptables-extensions.8.html
- https://zhuanlan.zhihu.com/p/223038075
- https://www.cnblogs.com/ggjucheng/archive/2012/08/19/2646466.html
- https://arthurchiao.art/blog/deep-dive-into-iptables-and-netfilter-arch-zh/
- https://arthurchiao.art/blog/nat-zh/
- https://arthurchiao.art/blog/tcpdump-practice-zh/

福利

- iproute2 好好用啊！如果在 Windows 或者 macOS 上也有多好啊！
- 福利来了：
  - 在 Windows 上，WSL1 发行版是可以管理 Windows 的路由表和 IP 地址的。只要安装一个 WSL1 发行版，以管理员权限打开它就可以了！
  - 在 macOS 上，有一个第三方实现的语法上基本兼容的 `ip` 命令，在安装了 Homebrew 的情况下，只需一条命令即可安装：`brew install iproute2mac`。

作业

提交到 victor@redrock.team
格式：学号-Linux网管-LvN，N 为最高做到的等级

做出进阶作业和之后的就不需要提交基础作业了。

- Lv0：写出对于子网 222.177.140.0 掩码为 255.255.255.128，下一跳地址为 172.22.161.1 的 iproute2 添加路由命令。(必做)
- Lv1：完成基本的网关搭建，实现 Ubuntu 20.04 LTS 的 NAT 网关让其它虚拟机上网，不需要实现 DMZ 主机。截图提交 traceroute 任意地址前两跳。(VMware 网卡虚拟化因素无法 traceroute 到第三跳之后的地址) (必做)

进阶作业：

- Lv2：在 Lv1 的网关上实现端口映射。(提示，使用 SNAT 和 DNAT 配合，以及 iptables 匹配的 tcp/udp 扩展中的 --sport 和 --dport)
- Lv3：查阅资料或可选的购置设备实现一个全功能 Linux 路由器，在寝室里用它上网。(不要求所有功能都在同一个系统上实现。)

新世界的大门：

- Lv4：查阅资料在实现的全功能路由器上，配置自启动的校园网自动登录和佛跳墙。

(自动登录脚本 https://github.com/qwqVictor/CQUPT-Internet-AutoLogin)

EOF

- 吃饭啦吃饭啦！