Intro to Networking & Linux Networking Commands

管培勛 (phkoan)
Credit to: 顏芝佑 (ycyyo), 許仲宇 (hsuchy), 施羿廷 (ytshih)

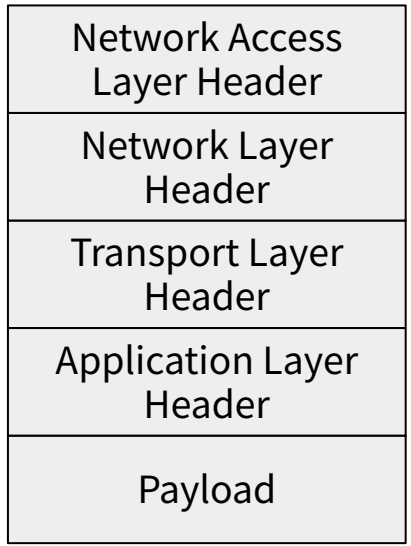
Outline

- Introduction to Networking
- Linux Networking Tools
  - traceroute & mtr
  - tcpdump
  - dig
  - ss
  - SSH
  - iproute2
- Linux Networking Features
  - netfilter
  - systemd-networkd & networkctl
  - systemd-resolved

Introduction to Networking

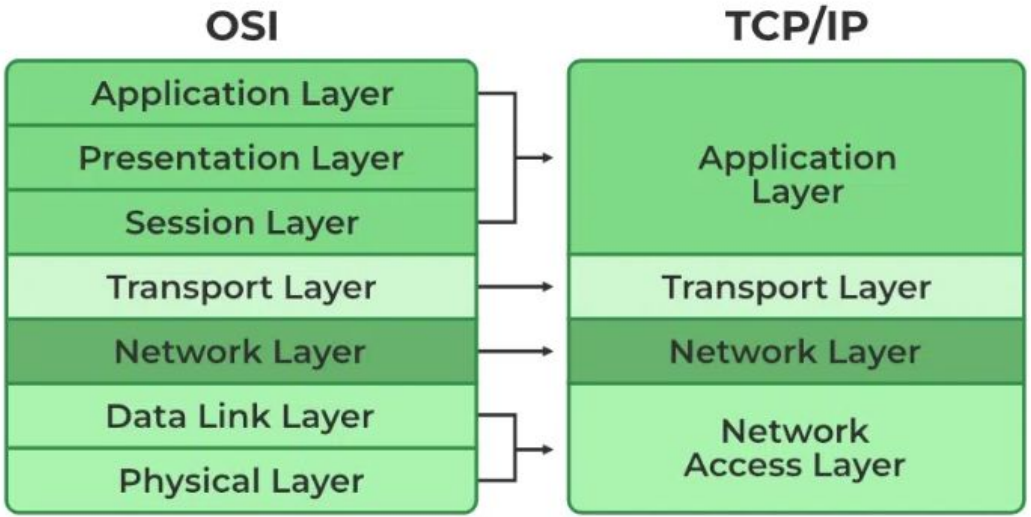
TCP/IP Model

- Network Access Layer (or Link Layer, Layer-2)
  - Handles physical transmission of data frames.
  - Ethernet, IEEE 802.11 (Wi-Fi)
- Internet Layer (or Network Layer, Layer-3)
  - Routes and forwards packets across interconnected networks.
  - IPv4, IPv6
- Transport Layer (Layer-4)
  - TCP, UDP
- Application Layer (Layer-7)
  - HTTP, SMTP



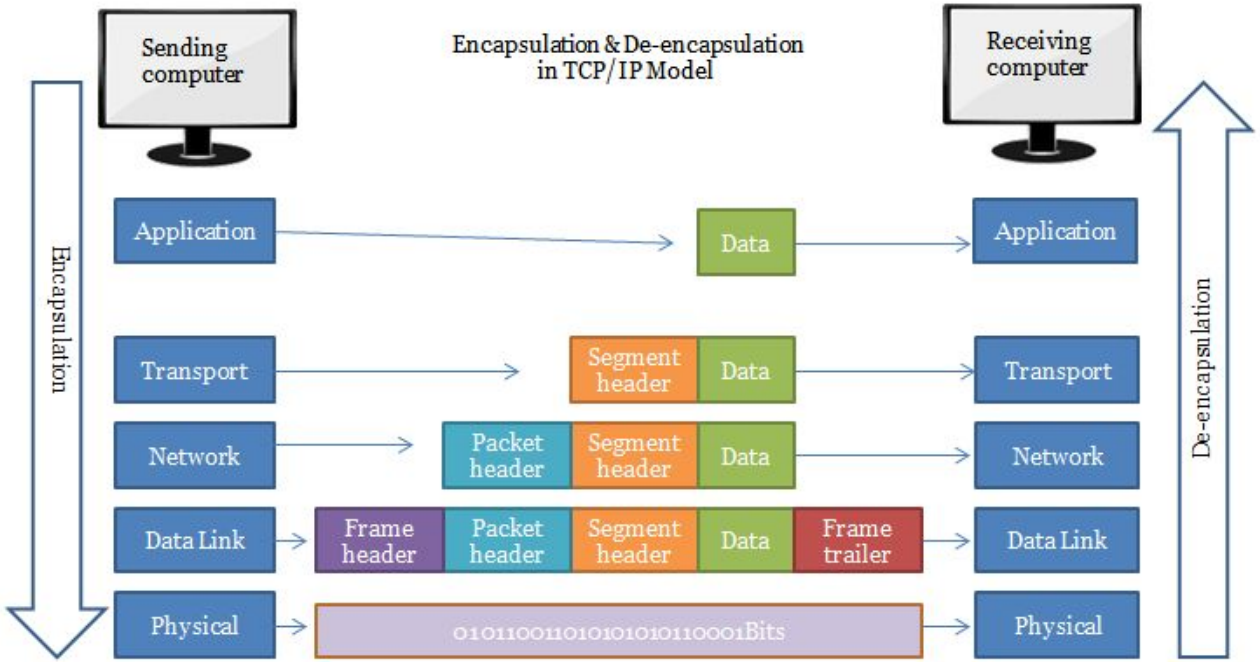
▲ Illustration of encapsulation

FYR: TCP/IP Model vs. OSI Model



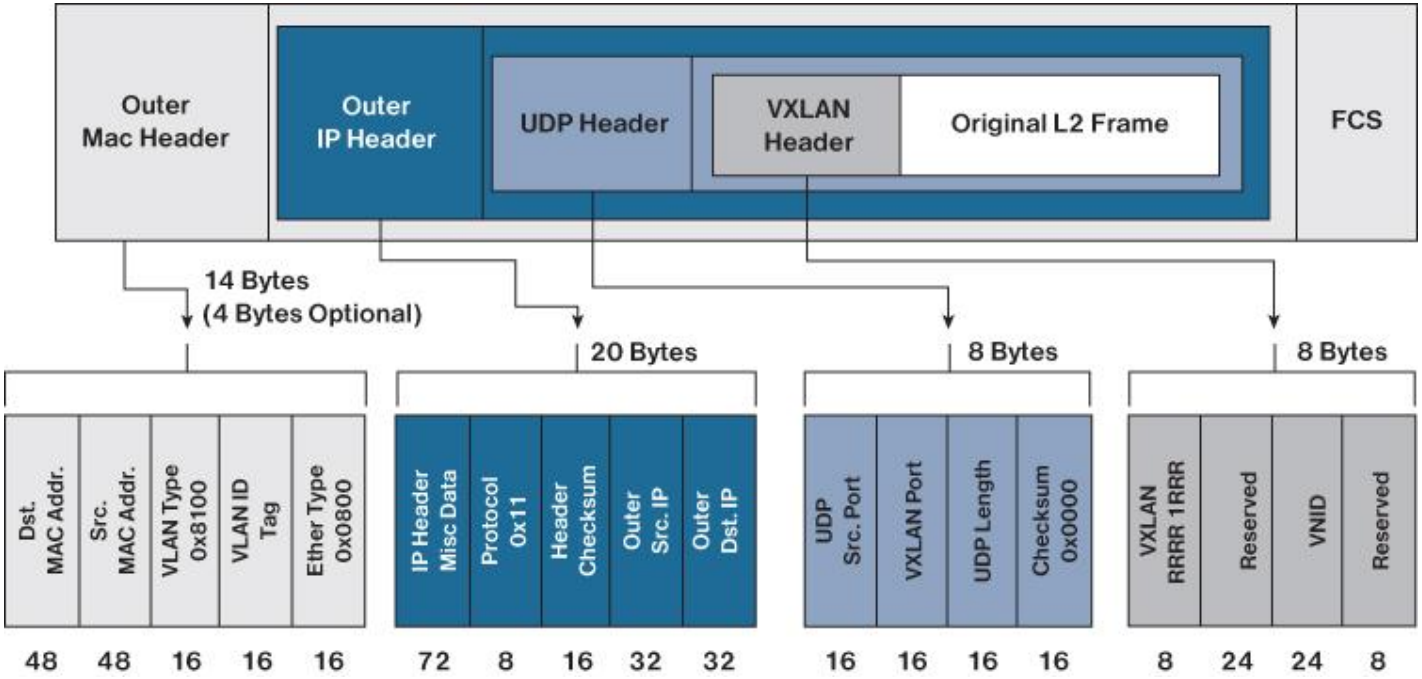
source: TCP/IP Model - GeeksforGeeks

Packet Encapsulation



source: [Data Encapsulation and De-encapsulation Explained]

FYR: Packet Encapsulation in VPN



source: [Virtual Extensible LAN (VXLAN) Overview | Cisco Press]

Layer-2 - Local Area Network (LAN)

- A network that interconnects devices within a limited area of **a single broadcast domain**.
  - Broadcast domain: Every device can receive frames through L2 broadcasting.
- Devices on the same LAN can **physically** communicate with each other with MAC addresses.

Layer-2 - MAC Address & ARP

- MAC (Media Access Control) address is a universally unique identifier for network interface card (NIC).
- ARP (Address Resolution Protocol) helps devices know IP-MAC mapping.

Layer-2 - MAC Learning

- Each **NIC** maintains an ARP table, which stores IP-MAC mapping.
  - When NIC wants to know the MAC address of certain IP address, it will send ARP request to ask who has the IP address.
  - The one having the requested IP address will response with an ARP reply, containing its own MAC address.
- **Switch** maintains MAC address table, which stores port-MAC mapping.
  - Which is "which **MAC address** is reachable through which **port**".
  - When switch sees a packet flows into a port, it records the port-MAC mapping to the table.
  - If table miss, the packet will be flooded to all ports.

source: Cisco Catalyst 2960 Series Switches

Layer-3 - IPv4 address

- 32 bits long, hierarchical addressing
- Consists of two parts
  - High order bits: Network ID (or Subnet ID)
  - Low order bits: Host ID
- Two special IP addresses in each network
  - First IP: Network Number, e.g. 140.113.0.0/24, 140.113.215.128/25
  - Last IP: Broadcast IP, e.g. 140.113.255.255/16, 140.113.215.127/25
  - Conventionally, we use the second-to-last IP address as gateway
- Reserved IP addresses
  - Private network: 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8
  - Loopback addresses: 127.0.0.0/8
  - Link-local addresses, multicast addresses, etc.

Example: 140.113.215.130/25

140.113.215.10000010

Network ID(25 bits) | Host ID (7bits) | prefix length

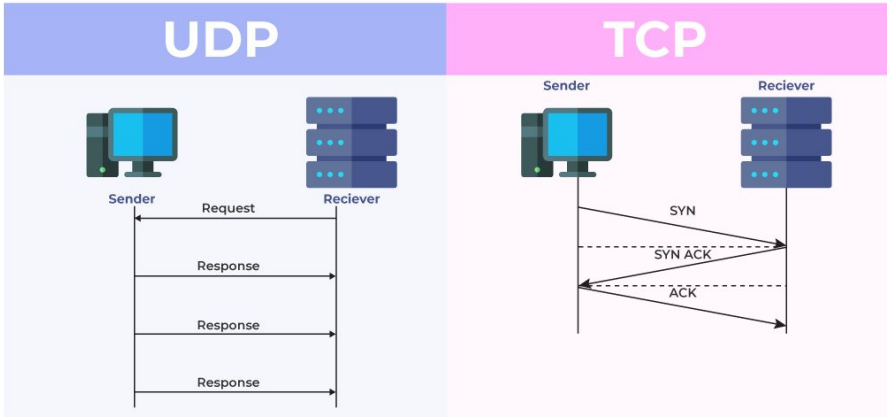
available IP addresses in this network:

140.113.215.128 to 140.113.215.255

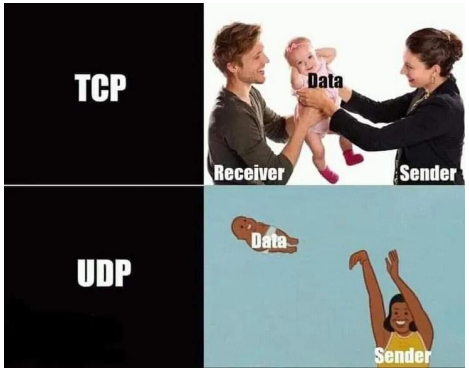
10000000 11111111

Layer-4

- TCP (Transmission Control Protocol) ensures reliable, ordered, and error-checked delivery of data between applications.
- UDP (User Datagram Protocol) sends data without guaranteeing delivery, order, or error correction, making it faster but less reliable.



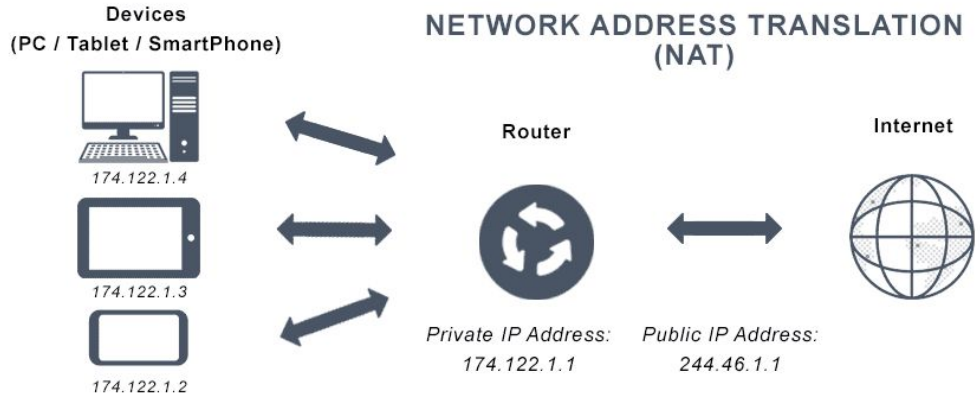
source: [When is UDP preferred to TCP? - GeeksforGeeks]



source: [ProgrammerHumor]

NAT

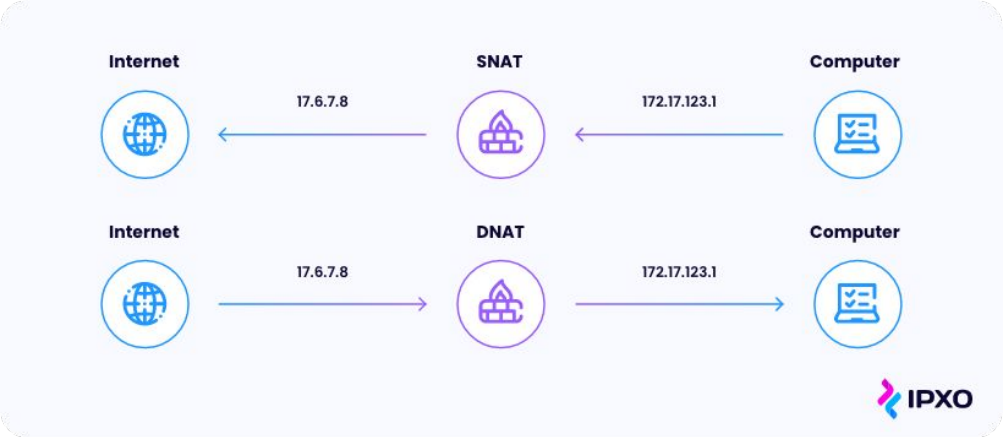
- NAT (Network Address Translation) is a technique that allows multiple devices on a local network to share a single public IP address.
- It is achieved by modifying the IP address information in packet headers as they pass through a router or firewall.



source: [What is Network Address Translation? | VMware]

NAT

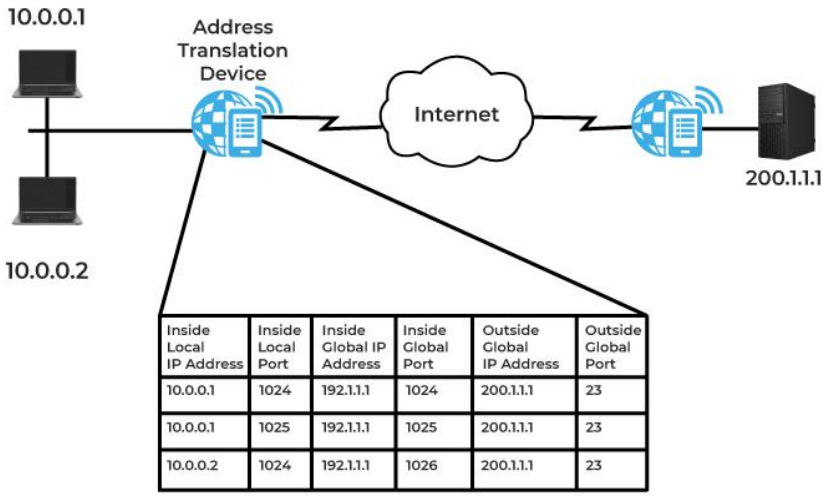
- There are two types: SNAT (Source NAT) and DNAT (Destination NAT).
  - SNAT modifies the source IP address of outgoing packets (usually to a router's public IP).
  - DNAT changes the destination IP of incoming packets to direct them to specific devices within a private network.



source: [What Is Network Address Translation? A Guide to NAT - IPXO]

FYR - NAT Table

- Router or firewall will maintain a mapping of private address & port to public address & port.
- The mapping is called NAT table.
  - Inside Local: Host's private address
  - Inside Global: Host's address after NAT
  - Outside Global: Target host's address
  - Outside Local: Target host seen from the internal network, often the same as Outside global



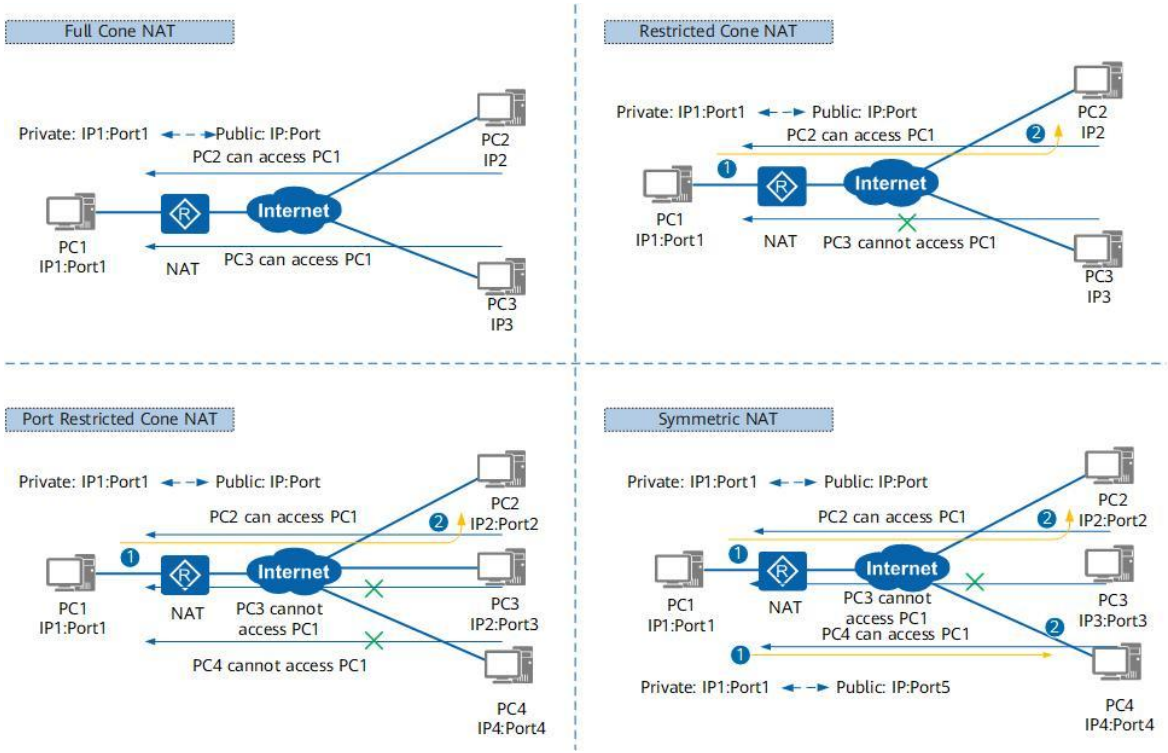
Inside Local IP Address	Inside Local Port	Inside Global IP Address	Inside Global Port	Outside Global IP Address	Outside Global Port
10.0.0.1	1024	192.1.1.1	1024	200.1.1.1	23
10.0.0.1	1025	192.1.1.1	1025	200.1.1.1	23
10.0.0.2	1024	192.1.1.1	1026	200.1.1.1	23

source: What is NAT (Network Address Translation)?

FYR - NAT Cone Type

- For example, in a connection from PC to `1.1.1.1:22`, `ip1:port1` is mapped to `ip2:port2`.
  - **Full Cone**: Any external host can access `ip1:port1` via `ip2:port2`.
  - **Address Restricted Cone**: Only `1.1.1.1` can access `ip1:port1` via `ip2:port2`.
  - **Port Restricted Cone**: Only `1.1.1.1:22` can access `ip1:port1` via `ip2:port2`.
  - **Symmetric**: As strict as Port Restricted Cone, but different destinations will have different public address-port pairs.

FYR - NAT Cone Type



source: [What Is Network Address Translation (NAT)]

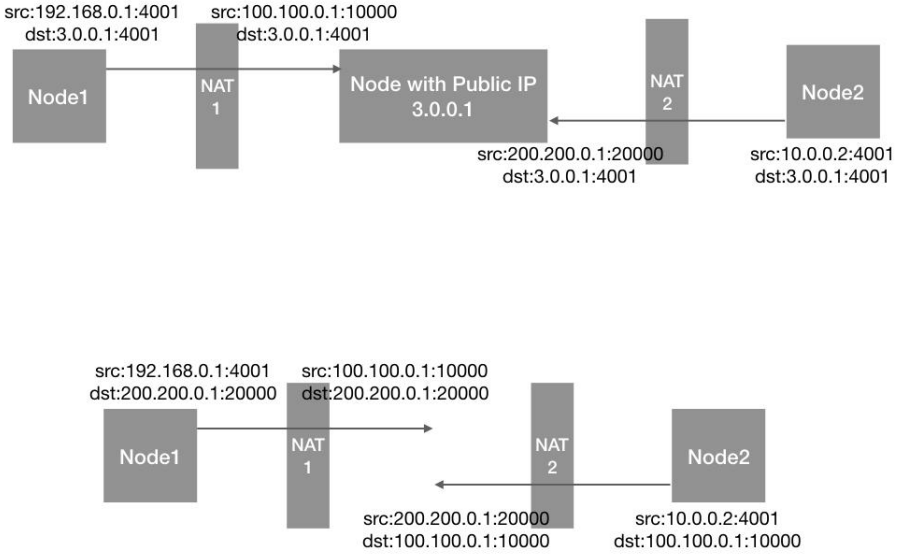
NAT Traversal

- NAT traversal enables devices behind NAT to establish and maintain peer-to-peer connections across the internet by bypassing NAT barriers.
- Methods: UDP hole punching, TURN, etc.
- How NAT traversal works is a great article from Tailscale, you can learn much from it.

NAT Traversal

Example: UDP hole punching.

1. Clients send a request to a public node.
2. The public node knows the NAT address-port pairs of the clients.
3. The public node tells the clients where each other is.
4. The clients directly send messages to the NAT endpoint.



source: Proposal for NAT UDP hole-punching · Issue #433 · libp2p/go-libp2p · GitHub

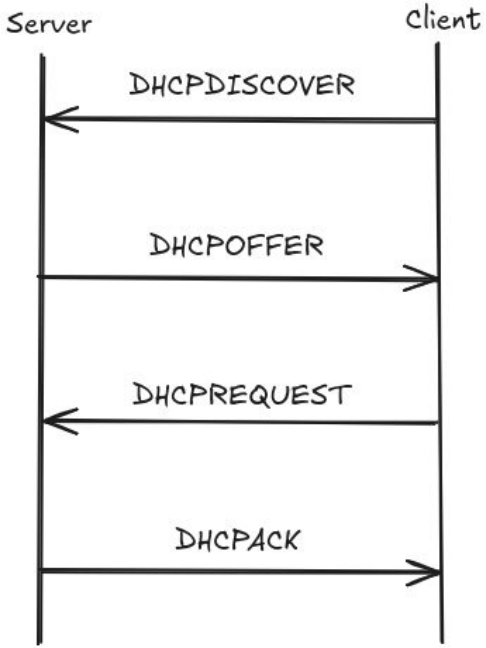
DHCP

- DHCP (Dynamic Host Configuration Protocol) provides a framework for passing configuration information to hosts on a TCP/IP network
  - It can pass IP address, hostname, gateway, DNS server, etc.
- DHCP is based on the Bootstrap Protocol (BOOTP).

DHCP - Client-Server Interaction

1. Client broadcasts DHCPDISCOVER message
2. Server responds with DHCPOFFER message
3. Client sends DHCPREQUEST message to server
4. Server replies with DHCPACK message

Finally, the client gets an IP address (DHCP lease).

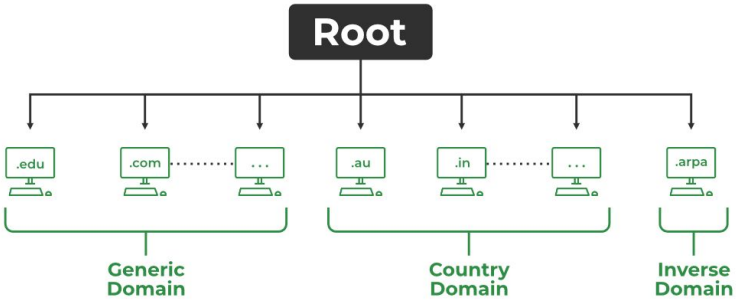


DHCP - Server

- DHCP server has a pool of IP addresses, which can be assigned to clients.
- When DHCP server offers an IP address to client, it remembers the IP-MAC mapping of that client.
- With the records, the DHCP server knows which IP addresses are still available.
    - Note: If you set an IP address in DHCP pool to an interface, the DHCP server will not know about this, and it can lead to IP address conflict.

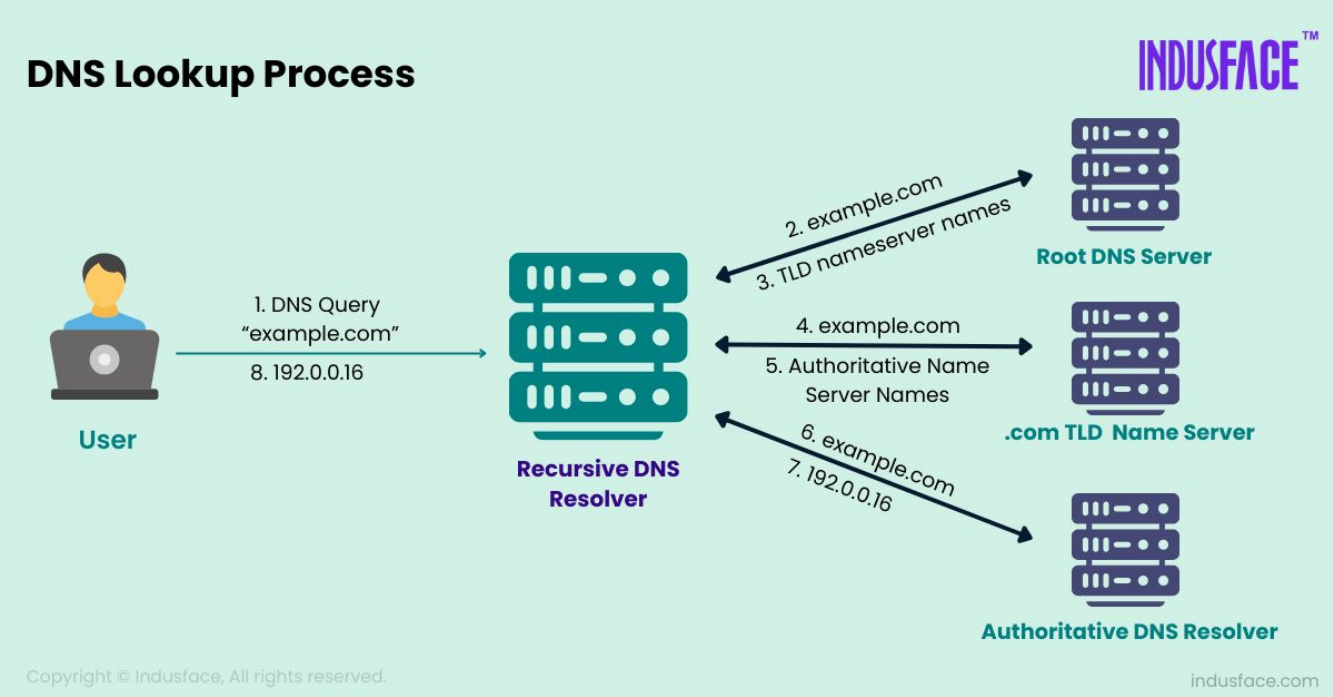
DNS

- DNS (Domain Name System) translates human-friendly domain names (like example.com) into IP addresses, the identifier on Internet.
  - The Internet's phonebook
- A domain name is hierarchical; the rightmost part is the highest level in the hierarchy.
- The resolution starts from the highest level.



source: Domain Name System (DNS) - GeeksforGeeks

DNS



source: [Understanding DNS: What It Is & How It Works | Indusface]

DNS - Roles

- **Client**: Sends DNS queries to server.
- **Resolver**: Accepts your DNS queries and resolve the name for you.
- **Root DNS server**: Directs DNS queries to the appropriate TLD servers based on the extension of a domain name (like .com).
- **TLD Server**: Points to the authoritative DNS servers responsible for specific domain names within its TLD zone.
  - TLD: Top Level Domain
- **Authoritative DNS Server**: Holds the actual DNS records for a domain and provides final, trusted answers to DNS queries.

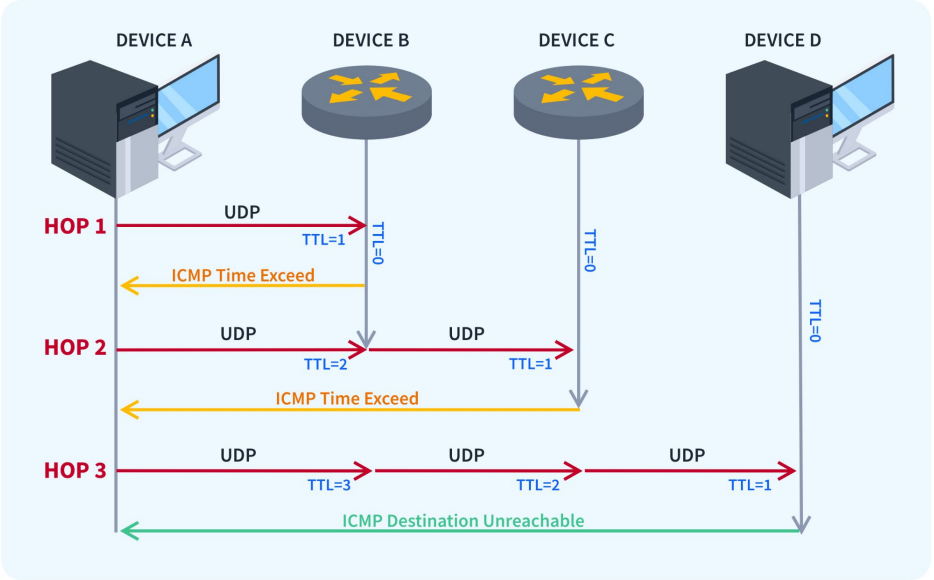
DNS - Record Types

- SOA: The primary authoritative DNS server, administrative information like the domain's serial number, refresh rate, and contact details
- A: The IPv4 address of a domain name
- AAAA: IPv6 version of A record
- NS: A list of the authoritative DNS servers
- MX: The mail server responsible for receiving email on behalf of a domain
- TXT: Arbitrary text data

Linux Networking Tools

traceroute

- It prints every hop packets flows to, from you to a given host.
- It's often used when you want to troubleshoot reachability and routing.



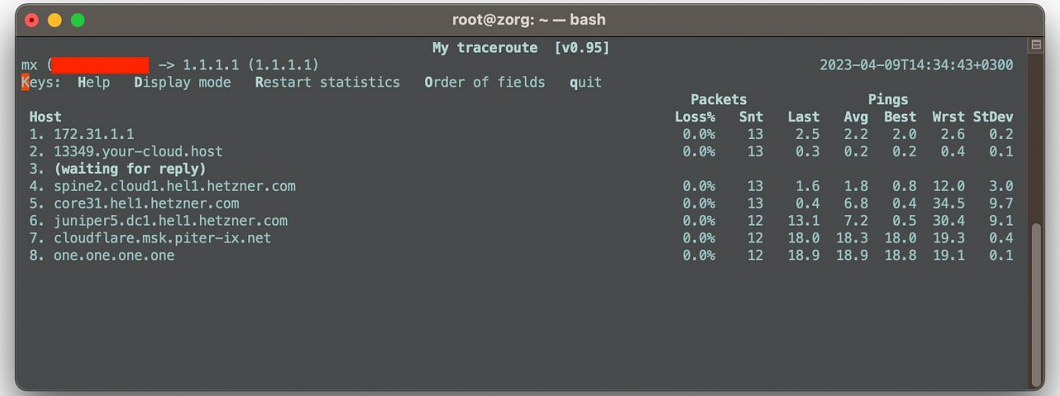
Source: [What is the Traceroute Command in Linux?]

traceroute

- Frequently used options
  - `-n` : Do not try to print hostname for each hop. Only showing numeric IP address.
  - `-I` / `-T` : Using ICMP ECHO/TCP SYN packet for probing route
    - Traceroute use UDP as default. But **Firewall may blocked UDP.**
  - `-z` *`sendwait`* : Waiting *`sendwait`* between each probe packet
    - If *`sendwait`* <= 10, the unit will be second. Otherwise unit will be milisecond.

mtr (My TraceRoute)

- It provides a prettier interface for traceroute.
- It can repeatedly preform traceroute and automatically caculate summarized statistics.



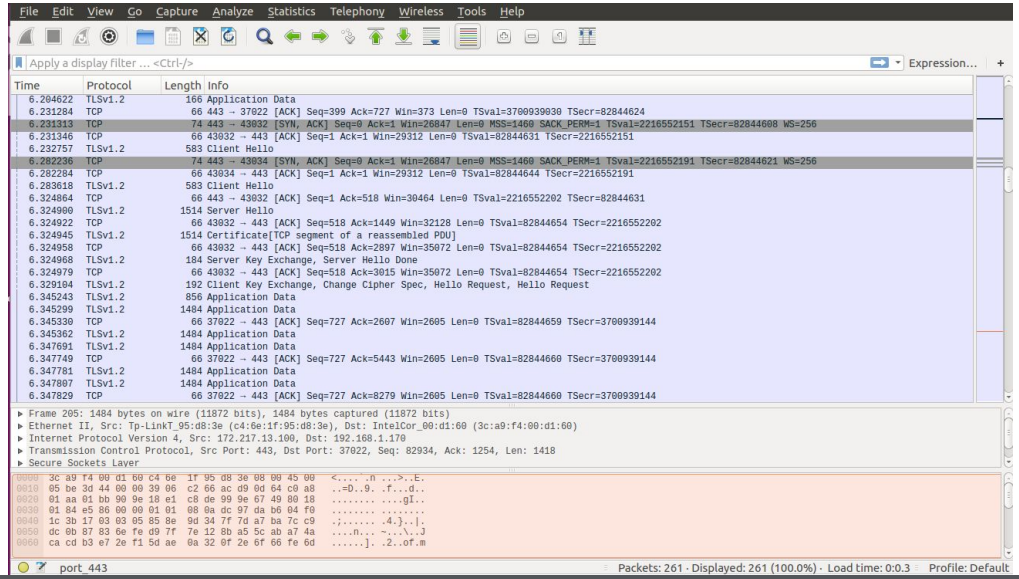
source: [mtr (My TraceRoute) — usage and practice]

tcpdump

- A tool for capturing packets through NIC and dumping captured packets.
- Filter expression allows users to capture packets that satisfy some conditions.
  - Such expression is called **pcap-filter**. See manual.
- Frequently used options
  - `-i` *`interface`*: monitor on *`interface`*. Usually require root permission.
  - `-q/-v/-vv/-vvv`: output quietly / verbosly / more verbosly / much more verbosly
  - `-A/-x/-X`: print captured packet header and content in ASCII / HEX / both.
  - `-w` *`file`*: save captured packet in *`file`* (in **PCAP Capture File Format**, binary)
  - `-r` *`file`*: read packet info from *`file`*.
- For example, `tcpdump -i eth0 dst port 80` captures packets passing through the eth0 interface where the destination port is 80.

FYR: WireShark

- It can be viewed as tcpdump with GUI.
- You can dump the packets with `tcpdump -w` and open the PCAP file with WireShark.



source: [How I use Wireshark]

dig

- A tool to perform **DNS lookups** and display the answers
  - Another useful tool for similar functionality is nslookup.
- Syntax: **`dig`** *`@server name type`*
  - You can also use **`dig`** *`@server`* **`-t`** *`type name`* to eliminate ambiguity
  - The types can be `A`, `AAAA`, `MX`, `TXT`, etc., and the default is `A`.
- Frequently used options
  - **`-x`** *`addr`* : performs reverse DNS lookup for *`addr`*
    - You do not need to provide name and type, if using this options.
  - **`+noall +answer`** : displays only answer section
  - **`+short`** : displays only rightmost column of answer section

SS

- A tool used to dump socket statistics
- Frequently used options
  - `-n`: show numeric port number
  - `-r`: show hostname instead of IP address
  - `-4/-6`: show IPv4/6
  - `-t/-u/-x`: show TCP/UDP/Unix domain socket
  - `-p`: show process which uses socket
  - `-l`: show listening socket

SS

- ss supports advanced filtering.
- The filter can be combined with the options.
- For example,
  - `ss state listening`: filter the sockets in listening state
  - `ss -t dst :22`: filter the TCP sockets connected to 22 port of any host
  - `ss dst :5432 and src 127.0.0.1`: filter the sockets connected to 5432 port of any host and initiate from 127.0.0.1

FYR - netstat

- netstat is an old tool, and ss is its modern alternative.
  - In some old machines, you can only use netstat.
- For example,
  - `netstat -l`: list the listening sockets
  - `netstat -nt | grep ':5432'`: list TCP connections and grep 5432 port

SSH

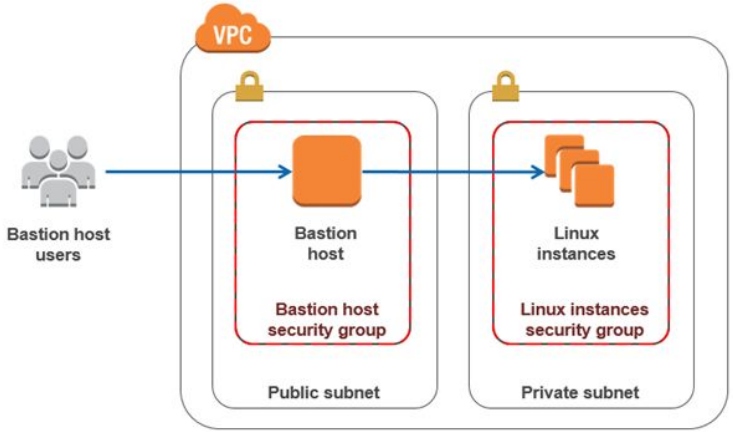
- Perhaps the most useful tool for login into a remote host machine
  - Besides CLI arguments, you can also use config file to manage ssh options, per host basis.
- Frequently used options
  - `-l username`: tried login remote machine as `username`
  - `-p port`: specify `port` on remote machine to connect to.
    - or you can write as: `[user@]dst_hostname[:port]`
  - `-i id_file`: specify `id_file` as private key file for public key authentication
  - `-J [user@]dst[:port]`: first connect to `dst`, then connect to `host` from `dst`.
    - useful when host is public unreachable, and dst is a entrypoint of such intranet.
  - `-L/-R`: SSH port forwarding. Details on following pages.
    - You can also read this great article for more info.

SSH

- You can use SSH key to login without password.
- The steps:
  - Use `ssh-keygen` to generate a key pair.
  - Copy the public key to `~/.ssh/authorized_keys` on remote host.
  - Try ssh again and you should be able to login without password.

SSH

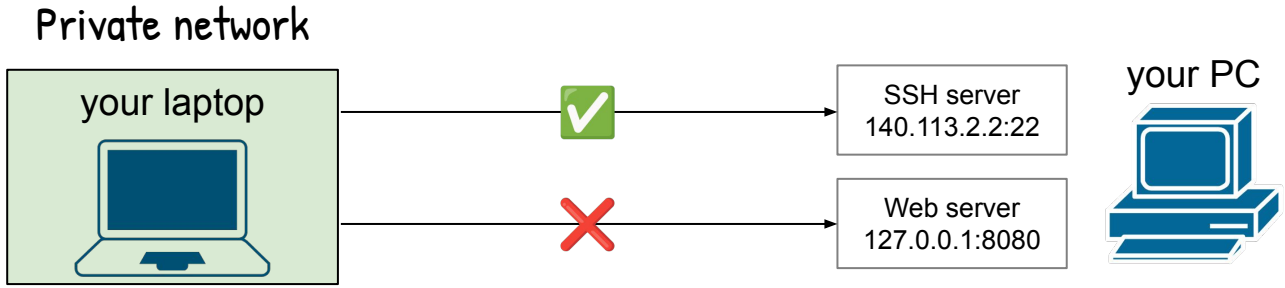
- If you have a SSH bastion, you can use `-J` to access internal hosts via the bastion.
- For example, `ssh -J <user>@<bastion> <user>@<target host>`.



source: Understanding the Purpose and Secure deployment of an SSH Bastion!!

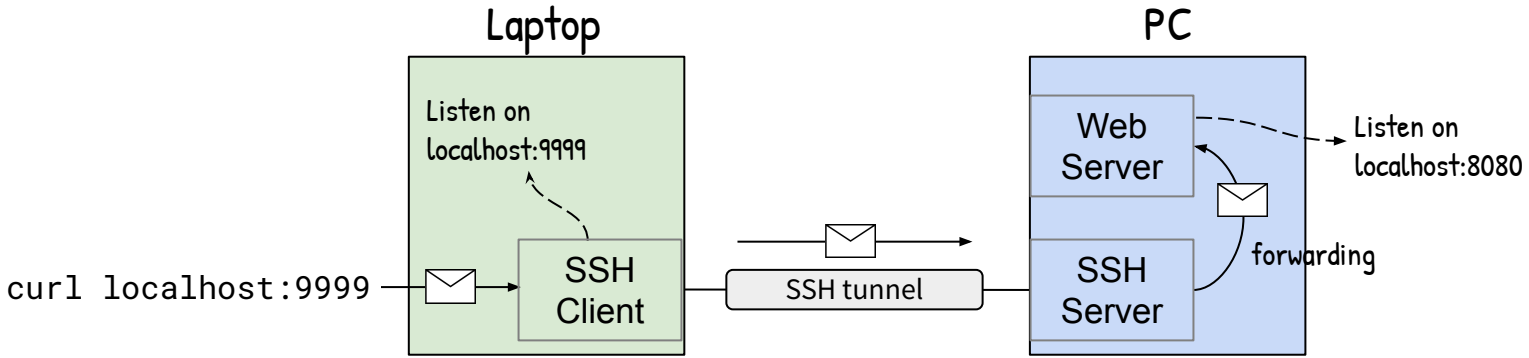
SSH - Local Port Forwarding

- You're doing your homework with an laptop at coffee shop.
  - That is, your laptop is very likely behind NAT, which is **not publicly reachable**.
- Your nginx is running on your PC in your dorm with a **public IP.**
  - The nginx server is listening request on 127.0.0.1:8080.
- You want to access your nginx from your laptop.



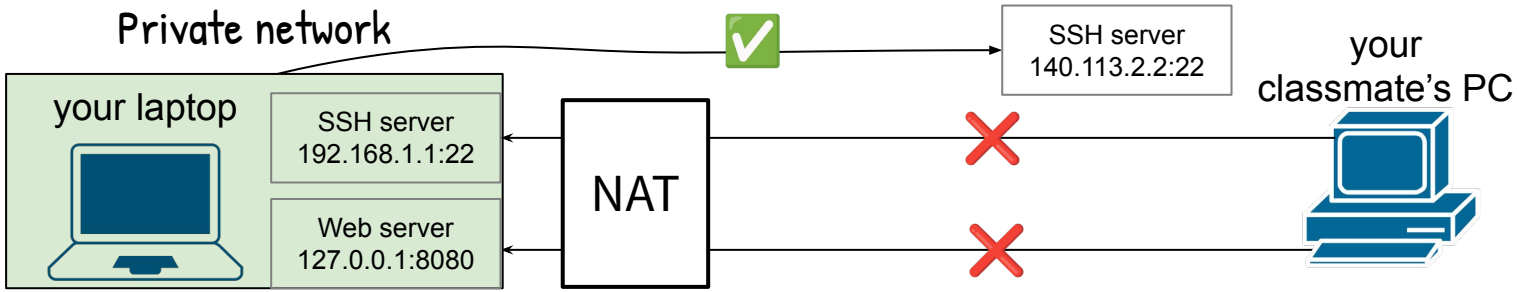
SSH - Local Port Forwarding

- `-L [bind_address:]port:host:hostport`
  - Example: client runs `ssh <server_addr> -L 9999:localhost:5432`
    - `9999`: laptop's port
    - `localhost:5432`: the address remote host (PC) sees
  - SSH client will listen on local address, and forward packet to SSH server.
  - SSH server will transmit forwarded packet to remote address.



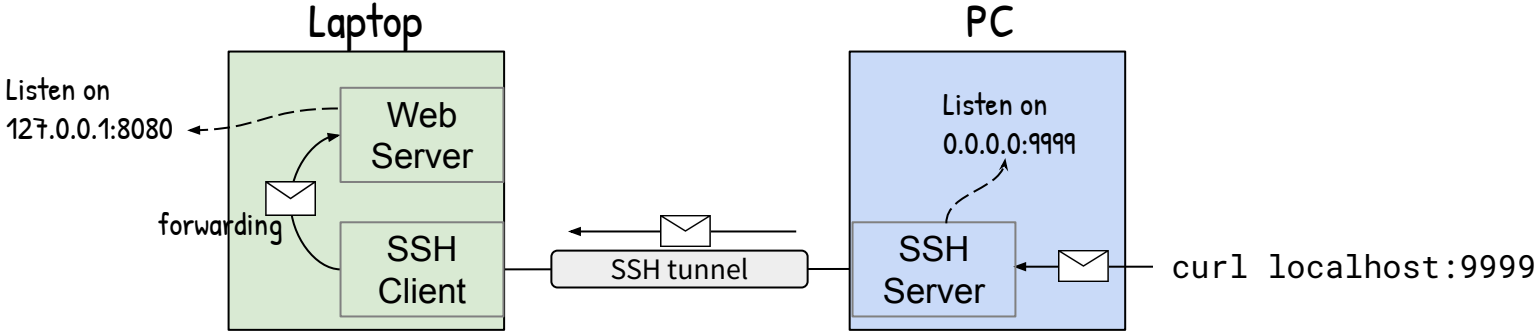
SSH - Remote Port Forwarding

- Your classmate wants to connect to the web server on your laptop, but your laptop is behind NAT.
- Why local port forwarding is impossible in this case?
  - Since your laptop is behind NAT, your classmate cannot SSH onto your laptop.
- Solution: Remote port forwarding



SSH - Remote Port Forwarding

- `-R [bind_address:]port:host:hostport`
  - Example: client runs `ssh server_addr -R 0.0.0.0:9999:127.0.0.1:5432`
    - `0.0.0.0:9999`: listen address of remote PC
    - `127.0.0.1:5432`: service address of the laptop
  - **SSH server** will listen on remote address, and forward packet to SSH client
  - **SSH client** will transmit forwarded packet into local address



iproute2

- A tool can manage various network objects.
- Mostly used:
  - `ip address`: manage address on a device. (man)
  - `ip link`: configure network device, or create virtual device. (man)
  - `ip route`: configure routing table. (man)
  - For reference of other objects, see ip(8)
- Frequently used options:
  - `-c`: print result in color.

iproute2 - ip link

- **`ip link show`**: show all interfaces (`show` can be omitted)
- **`ip link show`** *`bridge`*: show bridge interfaces
- **`ip link add name`** *`dum0`* **`type`** *`dummy`*: add a dummy dev called dum0
- **`ip link add name`** *`veth11`* **`type`** *`veth`* **`peer name`** *`veth22`*
  - Add a veth pair, one end named veth11 and other end name veth22
- **`ip link set`** *`eth0 up/down`*: bring eth0 up/down
- **`ip link set`** *`veth11`* **`master`** *`br0`*: connect ve11 onto bridge br0
- **`ip link del`** *`dum0`*: delete dev dum0

iproute2 - ip address

- **`ip addr show`**: show addresses of all interfaces
- **`ip addr show`** `up`: show addresses of up interfaces
- **`ip a add`** *`192.168.100.1/24`* **`brd`** *`192.168.100.255`* **`dev`** *`eth0`*
  - Add such address to dev eth0
  - `brd` can also be replaced with +, which represents last the IP address in the subnet.
- **`ip a del`** *`192.168.100.1/24`* **`dev`** *`eth0`*: Remove such address
- **`ip a flush dev`** *`eth0`*: remove all address on eth0

iproute2 - ip route

- **`ip route`** `[show]`: print the routing table
- **`ip route add`** *`10.1.100.0/24`* **`via`** *`192.168.2.254`* **`dev`** *`tun0`*
  - Add a route: for packet destination prefix with 10.1.100.0/24, send it from tun0 and use 192.168.2.254 as next hop.
- **`ip route add`** *`default`* **`via`** *`192.168.1.254`* **`dev`** *`eth0`*
  - Set default gateway to 192.168.1.254.
- **`ip route del`** *`10.1.100.0/24`* **`via`** *`192.168.2.254`* **`dev`** *`tun0`*
  - Delete such route.
- **`ip route get to`** *`8.8.8.8`*
  - Show which route will be chosen for 8.8.8.8
  - Useful when debug

iproute2 - ip route

```
default via 172.18.23.254 dev wlo1 proto dhcp src 172.18.16.1 metric 600
172.18.16.0/21 dev wlo1 proto kernel scope link src 172.18.16.1 metric 600
172.18.16.0/24 dev wlo2 proto static scope link src 172.18.16.2
```

- The output of `ip route` is shown above.
- The first field is destination prefix.
  - For most common case, packet route is **decided by its destination**.
  - If the **prefix of packet destination** matchs a route, the route will be a candicate.
  - If there are multiple candidates, the route with **longest prefix** will be chosen.
  - `default` is an alias for 0.0.0.0/0, which covers all IPv4 addresses.
  - For example,
    - 172.18.17.1 matches 1st and 2nd route. The 2nd route will be chosen since it has prefix length 21.
    - 172.18.16.1 matches 1st, 2nd & 3rd route. The 3rd route will be chosen since it has prefix length 24.

iproute2 - ip route

```
default via 172.18.23.254 dev wlo1 proto dhcp src 172.18.16.1 metric 600
172.18.16.0/21 dev wlo1 proto kernel scope link src 172.18.16.1 metric 600
172.18.16.0/24 dev wlo2 proto static scope link src 172.18.16.2
```

- `dev` means the interface the packet will go to.
- `proto` indicates the method to obtain this route
  - `kernel` means it is from auto configuration.
  - `static` means it is manually set.
- `via` means the address of next hop
  - i.e. gateway, i.e. "the next router this packet should go to"
  - Route with no next hop implies destination should resides **in same subnet (LAN)**.

iproute2 - ip route

- If you want to make your Linux machine a router:
  - Besides well configured routing table, you should also **enable packet forwarding**.
  - Perhaps you should also configure netfilter.
- In default, packet route is decided based on destination address
  - If you want to use source address or other info on packet for routing decision, consider **Policy-based routing**
    - For more info, read this article, ip-rule(8), and table option of ip-route(8)

FYR - nmap

- A tool for network discovery and security auditing
- You can use it to quickly found reachable host and their open port.
  - nmap can also infer what service is actually running on this port, see manual.
- **USED WITH CAUTION**. You may get banned. Especially in school.
  - You can use scanme.nmap.org as target, for testing and practicing.

`-sn` : only host discovery	`-n` : do not perform DNS resolve	`-v` : verbose output	`-e` *`interface`* : use which *`interface`*
`-O` : show host OS version	`-sV` : probing for version of service	`-sS` : use TCP SYN for port scanning	`-p` *`portsRange`* : scan only *`portsRange`*

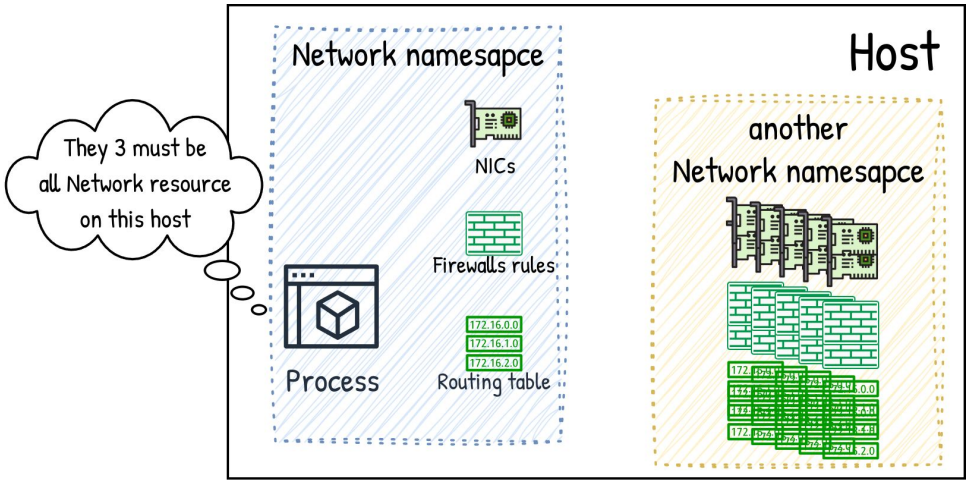
FYR - iperf3

- iperf3 is used to measure network traffic, including throughput and latency.
  - FYR: iperf2 and iperf3 are different projects, maintained by different teams, with different goals.
  - We will only cover iperf3.
- iperf allows you test network between **client** and **server.**
  - Both need to run iperf. server runs **`iperf -s`**, while client runs **`iperf -c`** *`serverAddr`*.
- Frequently used options
  - **`-p`** *`port`* : run iperf on specified port; The default value is 5201.
  - **`-D`** : server only; making iperf3 server run as daemon (at background)
  - **`-b`** *`n[KMGT]`* : client only; set target bitrate to *`n`* bits/sec
  - **`-t`** *`time`* / **`-n`** *`n[KMGT]`* : client only; run for *`time`* seconds / run until send *`n`* bytes

Linux Networking Features

Network Namespace

- Network namespace
  - Fundamental of container network
  - Logically separate network resource
  - Each namespace has its own:
    - Routing table
    - Firewall rules
    - Network devices (NICs)
- Process can only see and manipulate network resource within its namespace.



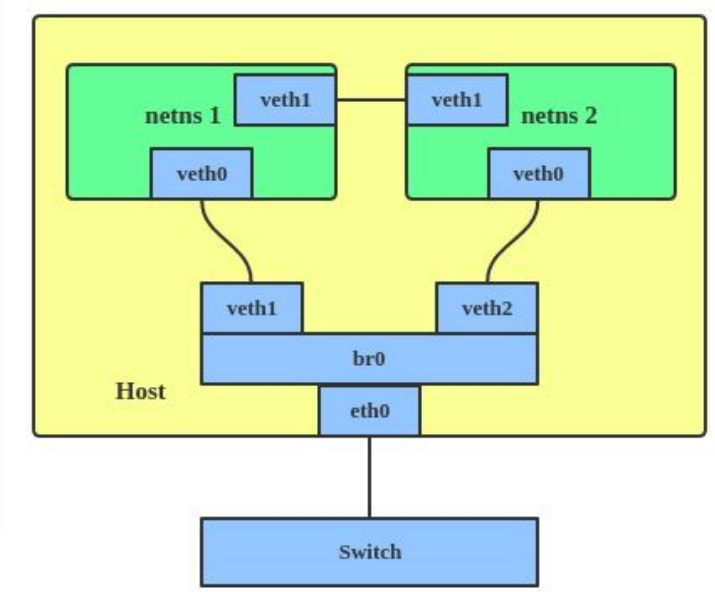
veth

- A special type of virtual interface and always comes in pair.
- Two veth interfaces can be in different namespace.
  - It enables cross-namespace communication.
- Veth transmits packet from/to peer veth
  - You can think the link between veth pair as an ethernet cable.

source: Introduction to Linux interfaces for virtual networking

Bridge

- A special type of virtual interfaces
- It acts like a lightweight virtual switch.
  - Physical/Virtual network interface can connected to it.
- Bridge can forward packets to/from connected interface.



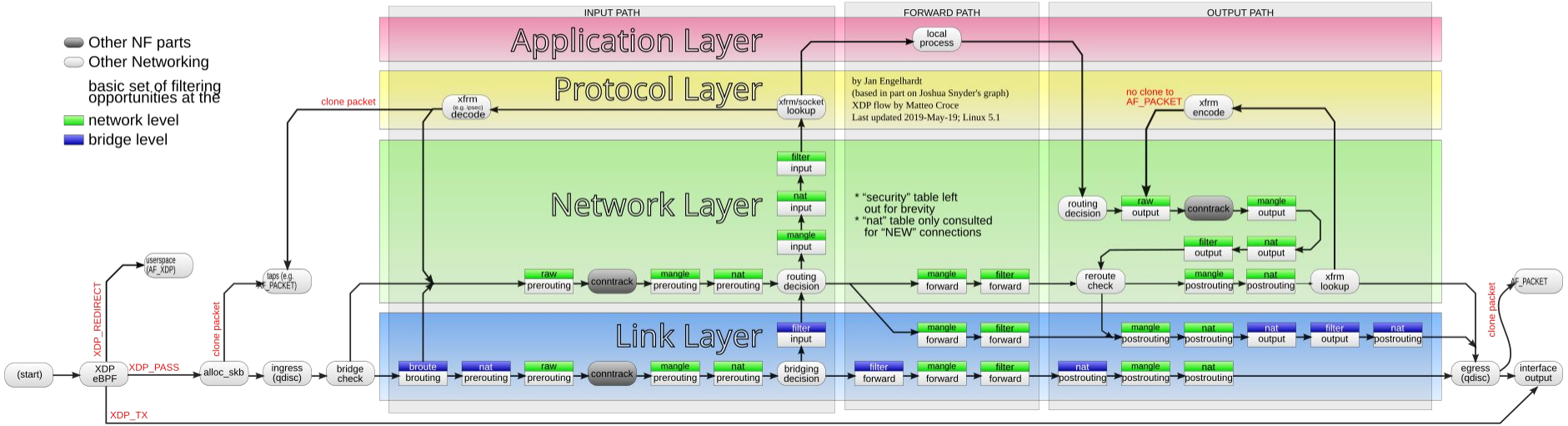
source: [Introduction to Linux interfaces for virtual networking | Red Hat]

netfilter

- Netfilter is a community-driven collaborative FOSS project.
- It provides packet filtering software for the Linux 2.4.x and later kernel.
  - **Packet filtering**
  - **Network address (and port) translation**
  - Userspace packet queueing
  - Packet mangling

netfilter

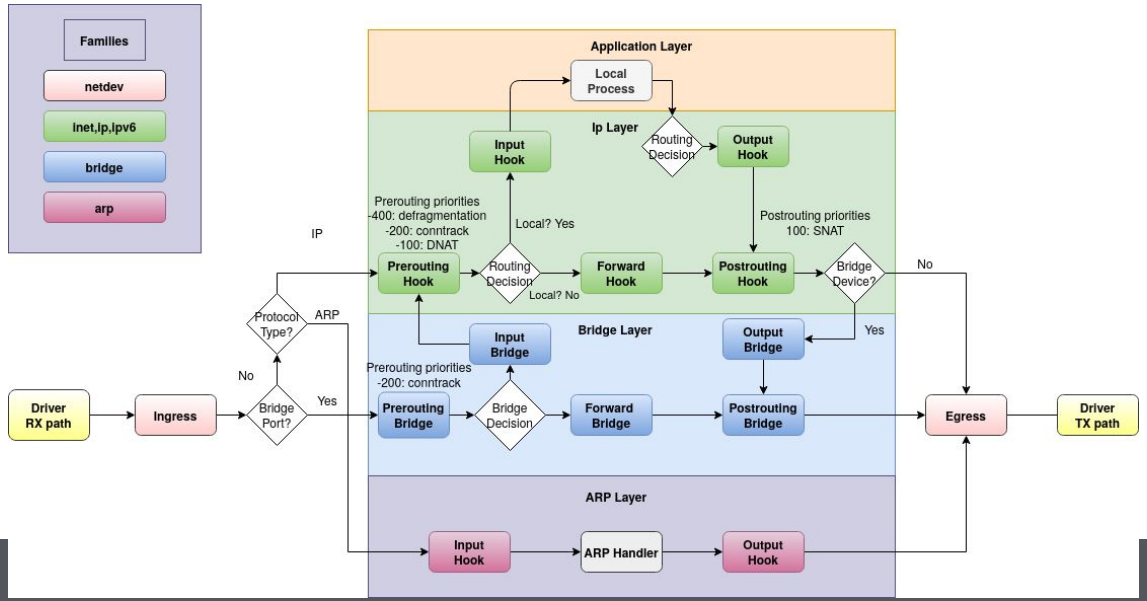
- xtables (legacy)
  - iptables, arptables, ebtables
  - Allowing kernel modules to register callback functions



source: Netfilter | Wikipedia

netfilter

- nftables
  - Replacing the legacy xtables with **nftables families**
  - No predefined tables in nftables framework



source: [Netfilter hooks | nftables wiki]

systemd-networkd

- System daemon can manage network configurations.
  - But it's not the only option in Debian.
- It detects and configures network devices that matched some rules.
  - It's useful for container or VM network, as they may create virtual device dynamically.
  - It can also create virtual device.
- Use three kinds of network configuration
  - `.network` : configures network device
  - `.netdev` : creates virtual network device
  - `.link`: modifies attributes of physical device (used by udev)
- `networkctl` is a CLI tool to interact with daemon.

systemd-networkd - Syntax

- Each configuration file contains multiple sections.
  - `[<Section name>]` indicates the start of a section
- Each section contains one or more key-value pairs
  - syntax: `key=value`
  - One line for one key-value pair.
- Configuration file should be placed in `/etc/systemd/network/` to be recongnized by systemd-networkd.

systemd-networkd - .network

- The `.network` configuration file will only applied on matched interface.
  - We need to specify rules for matching interface.
  - Such rules should reside in `[Match]` section.
  - Interfaces that satisfy **all** rules in `[Match]` section will then be configured by that config file.

```
[Match]
Name=eth0 eth1
```

```
[Match]
Name=eth*
```

```
[Match]
Name=!br0
MACAddress=12:34:56:78:90:ab
Kind=Bridge
```

systemd-networkd - .network

- systemd-networkd loads configuration files in filename lexicographic orders.
  - That is, `bar.network` will be processed before `foo.network`.
- In convention, we prefix filename with 2 digits.
  - e.g. `40-foo.network`
  - This way, we can easily tell the priority of each configuration file
- Note that interface can be configured with only **first matched** configuration file.
  - When an interface is matched by multiple configuration files, the interface will **only** configured by the file with higher priority

systemd-networkd - .network

- Every section except `[Match]` is used for configuring interface.
- Sections you should know:
  - `[Link]` - corresponding with hardware properties of interface
  - `[Address]` - setting address of interface
  - `[Route]` - setting route of interface
  - `[Network]` - miscellaneous
    - Contains key about DNS, DHCP, LLDP, NTP, NAT, Address, Route, Gateway, Master device (e.g. Master bridge), etc.

systemd-networkd - .network

```
[Link]
MACAddress=12:34:56:78:9A:BC
MTUBytes=1500
ActivationPolicy=always-up
```

- It will configure interface:
  - Sets its MAC address to 12:34:56:78:9A:BC
  - Sets its MTU to 1500B
  - Makes it "always-up".

systemd-networkd - .network

```
[Address]
Address=192.168.1.1/24
[Address]
Address=192.168.2.1/24
Broadcast=true
AddPrefixRoute=true
```

- Multiple `[Address]` sections = multiple addresses are configured.
  - The first will be 192.168.1.1/24, and the second will be 192.168.2.1/24
  - `Broadcast=true`: broadcast address is auto derived from address. Default is true.
    - 192.168.1.255, 192.168.2.255 in this case.
  - `AddPrefixRoute=true`: it will add a route based on address. Default is true.
    - Routes with destination prefixes 192.168.1.0/24 , 192.168.2.0/24 in this case.

systemd-networkd - .network

```
[Route]
Destination=192.168.3.0/24
Gateway=192.168.1.254
[Route]
Destination=0.0.0.0/0
Gateway=10.2.2.2
```

- Multiple `[Route]` sections mean that multiple routes are configured.
  - Routes configured by this way will go out via corresponding interface.
- For example, if `eth0` is matched, this config will generate 2 routes:
  - `default via 10.2.2.2 dev eth0`
  - `192.168.3.0/24 via 192.168.1.254 dev eth0`

systemd-networkd - .network

```
[Network]
Address=192.168.1.1/24
Address=192.168.2.1/24
Gateway=192.168.1.254/24
DNS=8.8.8.8
IPv4Forwarding=yes
```

- `Address`: shorthand of `[Address]` section with only `Address` key is specified
  - Specified multiple times = multiple addresses
- `Gateway`: shorthand of `[Route]` section with only Gateway key is specified
  - It will be default route, since destination is not specified
- `DNS`: address of DNS server
- `IPv4Forwarding`: enable packet forwarding on this interface

systemd-networkd - .netdev

- The `.netdev` files can be used to create virtual devices.
- We create a simple bridge interface as example.

systemd-networkd - .netdev

- First, we need a `.netdev` file to create the bridge interface.
  - Follow the convention, name it `20-br0.netdev`.
- After the creation, you can see the interface via `ip a`, but it does not have any address.

```
[NetDev]
Name=br0
Kind=bridge
```

systemd-networkd - .netdev

- We next bind `eth0` to `br0`.
  - Name it `20-br0-eth0.network`.
  - Note: the `.network` file for `eth0` should be removed.
- Finally, we set configure `.network` file for `br0`.
  - Name it `20-br0.network`.

```
[Match]
Name=eth0

[Network]
Bridge=br0
```

```
[Match]
Name=br0

[Network]
DHCP=yes
```

networkctl

- A CLI tool for interacting with systemd-networkd
- `networkctl` or `networkctl list:` list interface and their status
- `networkctl status <interface>:` print detailed status of a interface
- `networkctl reload:` reload configuration

networkctl

- In the output of `networkctl`, you can see operational and setup status.

```
IDX LINK TYPE     OPERATIONAL SETUP
  1 lo   loopback carrier     unmanaged
  2 eth0 ether    degraded    configuring
  3 eth1 ether    routable    configured
  5 dum1 ether    off         unmanaged
  6 tap0 ether    no-carrier  unmanaged
```

networkctl

- `off`: powered off
- `no-carrier`: powered on but has no carrier
  - E.g. for a ethernet interface, no-carrier means unplugged
- `degraded`: has carrier but is not routable yet
  - Definitely cannot go beyond the local network
  - E.g. has address but no gateway
- `routable`
  - Possibly can reach external network

```
IDX LINK TYPE     OPERATIONAL SETUP
  1 lo   loopback carrier     unmanaged
  2 eth0 ether    degraded    configuring
  3 eth1 ether    routable    configured
  5 dum1 ether    off         unmanaged
  6 tap0 ether    no-carrier  unmanaged
```

systemd-networkd - Troubleshooting

- If a network configuration seems does not applied:
  - Check output of `networkctl` for overview of interfaces.
  - Check output of `networkctl status <interface>` for more detailed information.
    - Check which configuration file is applied on such interface.
  - Use `journalctl -u systemd-networkd` to check log of daemon.
  - Use net tools we discussed in previous sections to diagnose network.

systemd-resolved

- A systemd service managing name resolution
- The main configuration file is `/etc/systemd/resolved.conf`, and all files under `/etc/systemd/resolved.conf.d/` will be included.

systemd-resolved

- For DNS, most of the configuration can be done under `[Resolve]` section.
- Frequently used attributes
  - `DNS`: the main DNS servers, listed in space-separated way
  - `FallbackDNS`: DNS servers used when all main DNS servers fail
  - `Domain`: the search domain (the domain suffix)
  - `DNSSEC`: enable DNSSEC or not

```
[Resolve]
DNS=1.1.1.1 8.8.8.8
FallbackDNS=8.8.4.4
Domains=cs.nycu.edu.tw
DNSSEC=yes
```

systemd-resolved - Troubleshooting

- Check the status via `resolvectl status`.
- Check query results via `resolvectl query <domain name>`.