# Network Basics

# Topics

- Quick review of network fundamentals
  - The ISO OSI 7-layer model (and why it matters)
  - Network and host part of an IP address
  - Function of the subnet mask
  - Differences between IPv4 and IPv6
- Home/SMB networks
  - How NAT works
  - Firewalls
- Enterprise networks
  - VLANs
  - Cloud
- Linux network interface config and teaming using nmcli
  - nmcli commands
  - ifcfg files
- Practical demos

# OSI 7-Layer Model Basics

Media

- 1. Physical Layer
  - *Bits*—cables, connectors, WiFi, voltages, frequencies
- 2. Data Link Layer (MAC sublayer to L1 & LLC sublayer to L3)
  - *Frames*—Ethernet LAN, Switches, bridges, device addr.
- 3. Network Layer
  - *Packets*—IP addresses, Routers, subnetting, ping (ICMP)

Host

- 4. Transport Layer
  - *Segments*—Ports, TCP/UDP, "connections," sequencing
- 5. Session Layer
  - Inter-host communication, connection flow on ports
- 6. Presentation Layer
  - Formatting, encryption, screen layout
- 7. Application Layer
  - Network services to applications via HTTP, SMTP, etc. (not the applications themselves)

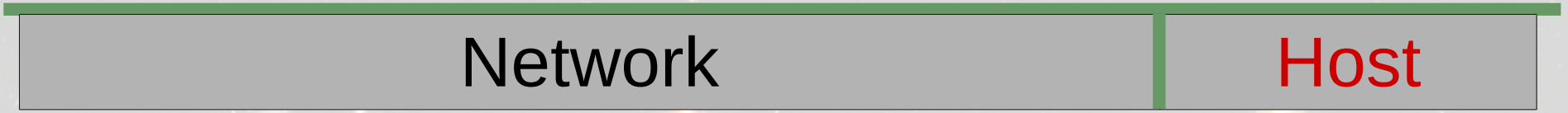
# IPv4—Made it all possible

## (Open Source IPv4 won out over closed IPX)

IP Address is in Binary inside the machine
But humans need to express IPv4 in Decimal

An IPv4 address is 32 bits long,
Which has 4 octets, each of 8 binary bits,
Thus 4 octets x 8 bits each = 32 bit IP address

192.168.1.25/24
Netmask: 255.255.255.0
11000000.10101000.00000001.00011001
11111111.11111111.11111111.00000000



# Anatomy of an IPv4 Address

Network.Host

(**IPv4** "/24" == bits in the **network** part)

192.168.1.25/24 (Class C, 256-2 hosts)

Netmask: 255.255.255.0

172.16.45.203/16 (Class B, 65,536-2 hosts)

Netmask: 255.255.0.0

10.250.145.36/8 (Class A, 16,777,216-2 hosts)

Netmask: 255.0.0.0

What about all the in-between places?

IP Address is in Binary inside the machine
But humans need to express IPv4 in Decimal

## Classless-InterDomain Routing (CIDR)

192.168.1.25/23 (512-2 hosts)

Netmask: 255.255.254.0

11000000.10101000.0000000**1**.00011001

11111111.11111111.1111111**0**.00000000


192.168.2.25/23 One bit changes both network and host address (in this case).

Netmask: 255.255.254.0

11000000.10101000.0000001**0**.00011001

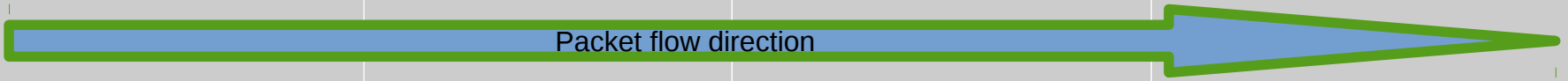
11111111.11111111.1111111**0**.00000000

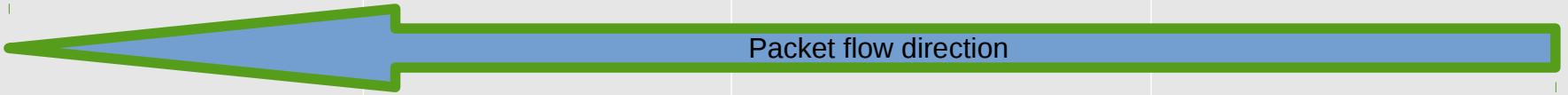
# NAT—There and back again

You can see my public face,
but not my private things
(unless I invite you into my private space).

# IPv4 Network Address Translation (NAT)
# Port Address Translation (PAT)

| Step | Private IP Device<br>192.168.1.3 | Router LAN Side<br>192.168.1.1 | Router WAN Side<br>12.34.56.78 | Internet |
|---|---|---|---|---|
| 1. | Source<br>192.168.1.3:26354<br>↓<br>Destination<br>78.56.34.12:443 | Src & Dst IP:Port info entered into NAT/Port Table<br><br>Forwards to WAN | Source<br>12.34.56.78:46871<br>↓<br>Destination<br>78.56.34.12:443 | Destination<br>78.56.34.12 receives packet on Port 443 with return IP and Port info, along with HTTPS request |
| 2. | Destination<br>192.168.1.3<br>receives packet on Port 26354 with HTTPS response from<br>78.56.34.12 | Src & Dst IP:Port info lookup in NAT/Port Table<br><br>Forwards to LAN IP that requested | Router receives packet from<br>78.56.34.12<br>on Port 46871 | Source<br>78.56.34.12:16927<br>↓<br>Destination<br>12.34.56.78:46871 |



It’s ~~less~~ differently complicated with IPv6

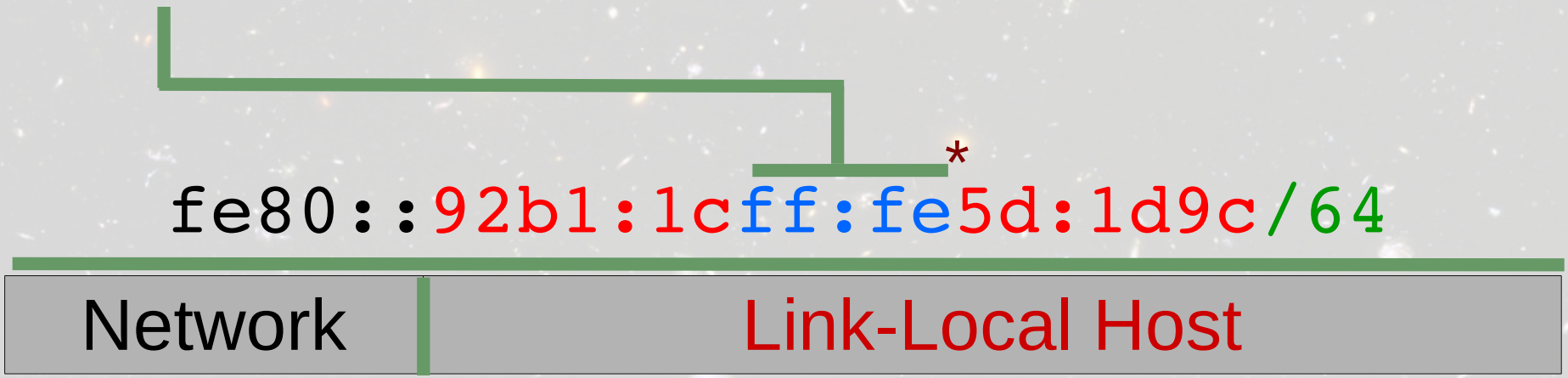
IP Address is in Binary inside the machine
But humans need to express IPv6 in HexaDecimal

An IPv6 address is 128 bits long, (same as MD5 hash)
Which has 8 groups, each of 16 binary bits,
Thus 8 groups x 16 bits each = 128 bit IPv6 address
**No `netmask` in IPv6**, but it does have a "/" to divide the network and host parts of the address
**(IPv6 "/64" == bits in the host part)**

`2001:db8:85a3:5b68:c2:8a2e:370:7334/64`

IP Address is in Binary inside the machine
But humans need to express IPv6 in HexaDecimal

No need for DHCP (or APIPA) in IPv6, because a unique IPv6 link-local address is generated from the 48-bit interface MAC address + 16 bit padding == 64 (to pad MAC to 64 bits for link local)




*IEEE-defined 64-bit Extended Unique Identifier (EUI-64)

IP Address is in Binary inside the machine
But humans need to express IPv6 in HexaDecimal

May need to add interface name after the IPv6 address to ping: e.g. `ping 2001::96:1d5%<int>`

```
$ ping -c1 20a4::9738:9136:1595:83b8
PING 20a4::9738:9136:1595:83b8(20a4::9738:9136:1595:83b8) 56 data bytes
64 bytes from 20a4::9738:9136:1595:83b8%enp1s3:
icmp_seq=1 ttl=64 time=5.13 ms
```

# More than just simple networks
## Enterprise VLANs

- Use tools such as:
  - MPLS (Multiprotocol Label Switching)
  - VPNs
  - Equipment vendor configurations

- May or may not align with subnetting schemas

- May be arranged by (any combination of):
  - Region
  - Organization
  - Function
  - (custom defined)

# More than just simple networks

## Cloud

- Software Defined Networking (SDN)
- Internal Routing Rules
- Increasingly is:
  - Virtual
  - Containerized

# Firewalls and network filtering

- Firewalld
  - Services
  - Zones
  - Dynamic
  - Changes active without restarting

- Iptables
  - Chains
  - Rules
  - Static
  - Must restart to make changes active

# Firewalld

```
# firewall-cmd --state
  Running

# firewall-cmd --list-all
(not as detailed output as from
iptables -L)

# firewall-cmd --add-port \
12345/tcp --permanent

# firewall-cmd –list-ports

# firewall-cmd --reload
```

# Network Manager

# Now for servers too!

`nmcli`—Network Manager Command Line Interface
(some introductory commands)

# Interface Configuration with nmcli

## Find out your interface “connections”

```
$ nmcli con show
<NAME> <UUID> <TYPE> <DEVICE>
```

## Find out your interface “Devices”

```
$ nmcli dev show
GENERAL.DEVICE:         <dev>
GENERAL.TYPE:           ethernet
GENERAL.HWADDR:         <MAC>
GENERAL.CONNECTION:     <con>
IP4.ADDRESS[2]:         x.x.x.x/24
IP4.GATEWAY:            192.168.x.1
IP6.ADDRESS[1]:
     fe80::52e1:1bff:fe2e:2a7e/64
```

# Add an IP Address with nmcli

```
$ nmcli con mod ens3 \
+ipv4.addresses "10.10.10.110/8"

$ nmcli connection down ens3

$ nmcli connection up ens3
```

Then check your ifcfg file:

```
$ vim /etc/sysconfig/network-
scripts/ifcfg-ens3

IPADDR=10.10.10.110
PREFIX=8
```

# “Team” 2 NICs together with nmcli

```
# dnf install NetworkManager-team
```

Get device names to edit correct ifcfg files:

```
$ nmcli con show # for dev & UUID info
```

```
$ nmcli c down <UUID>
→ for each connection to be “teamed”
→ no other connections for each
  device/interface to be “teamed”
```

Then edit each applicable ifcfg file:

```
# vim /etc/sysconfig/network-
scripts/ifcfg-<dev>
```

# 2 NICs with 1 IP: Edit team ifcfg file

```
# File is ifcfg-team0

DEVICE="team0"
DEVICETYPE="Team"
ONBOOT="yes”
BOOTPROTO=none
NETMASK=255.255.255.0
IPADDR=192.168.122.50
TEAM_CONFIG='{"runner": {"name":
"lacp"}}'
```

# 2 NICs with 1 IP: Edit device ifcfg files

```
# Files are ifcfg-eth0 and ifcfg-eth1

DEVICE=eth0
HWADDR=52:54:00:F0:5D:9A
DEVICETYPE=TeamPort
ONBOOT=yes
TEAM_MASTER=team0
TEAM_PORT_CONFIG='{"prio": 100}'
```

# Demos and Practicals!