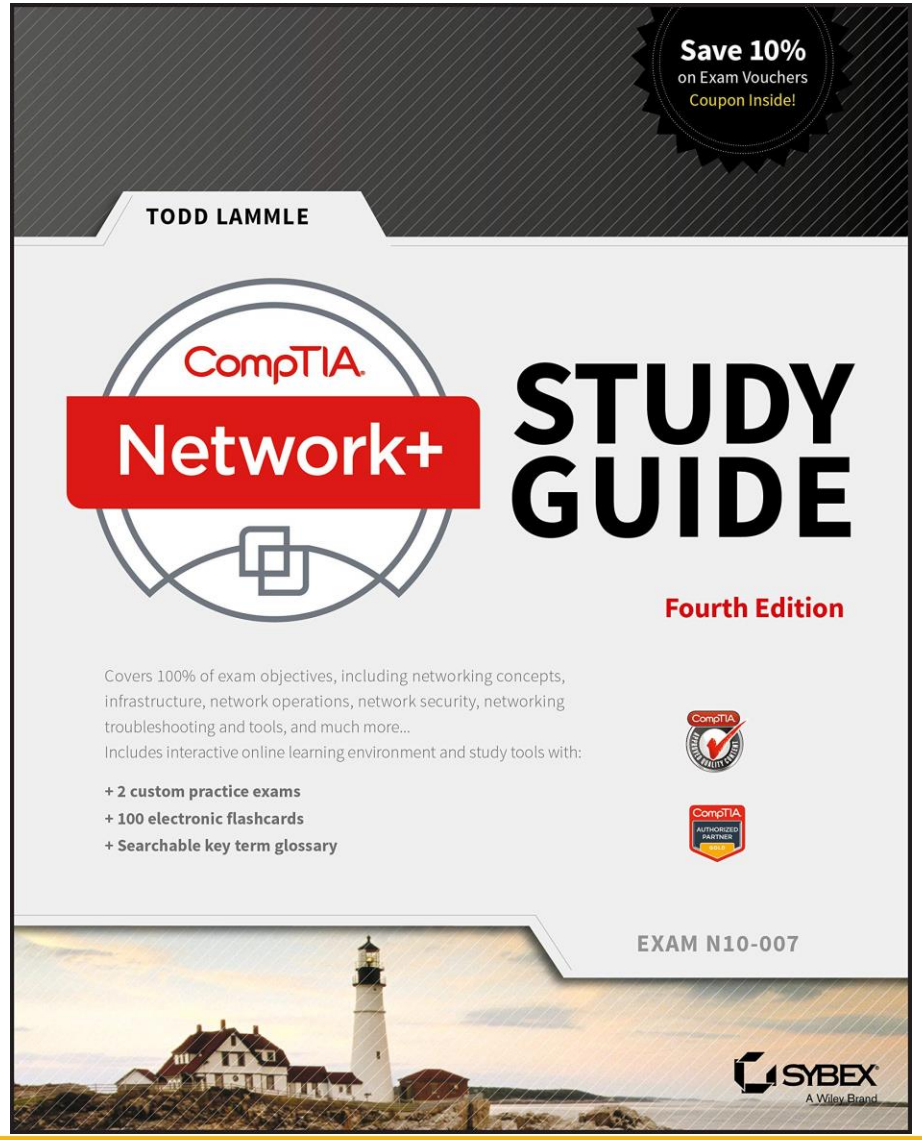
Configuring Basic Networking

Objectives:

- ✓ 109.1 Fundamentals of internet protocols
- ✓ 109.2 Persistent network configuration
- ✓ 109.3 Basic network troubleshooting
- ✓ 109.4 Configure client side DNS

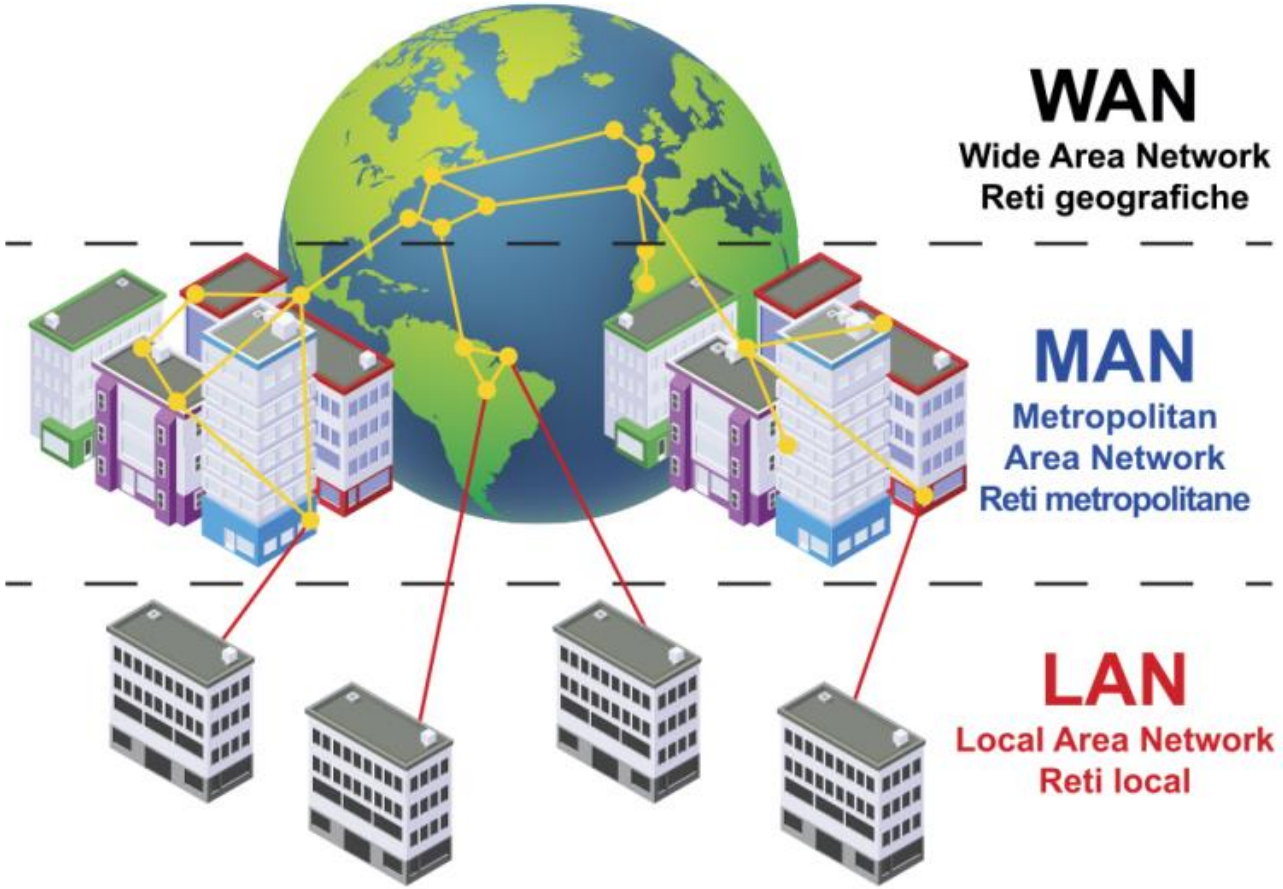
Recommended Book



NETWORKING BASICS

Before we take a look at how Linux handles network connectivity, let’s go through the basics of computer networking. Computer networking is how we get data from one computer system to another.

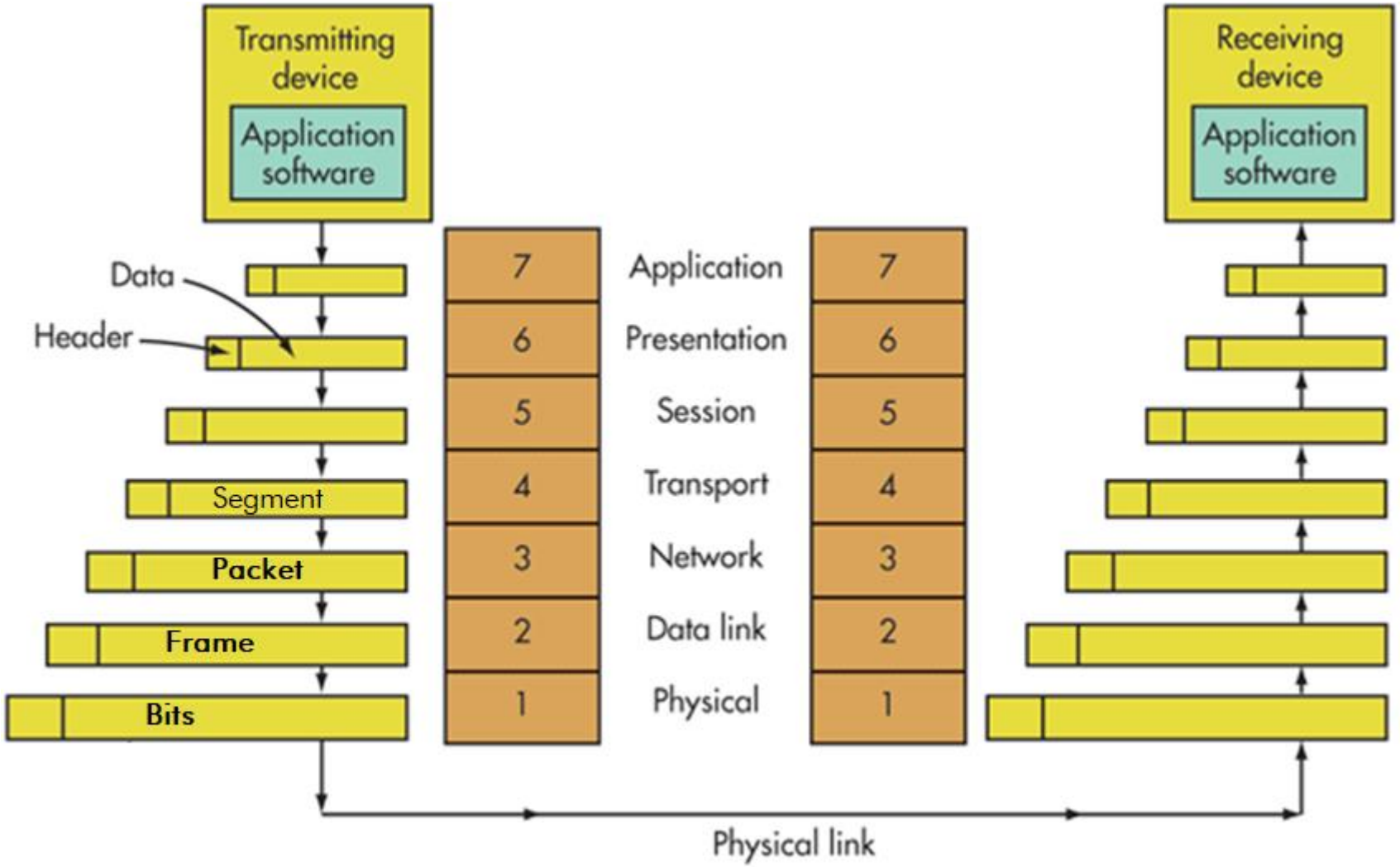
Networking Basics



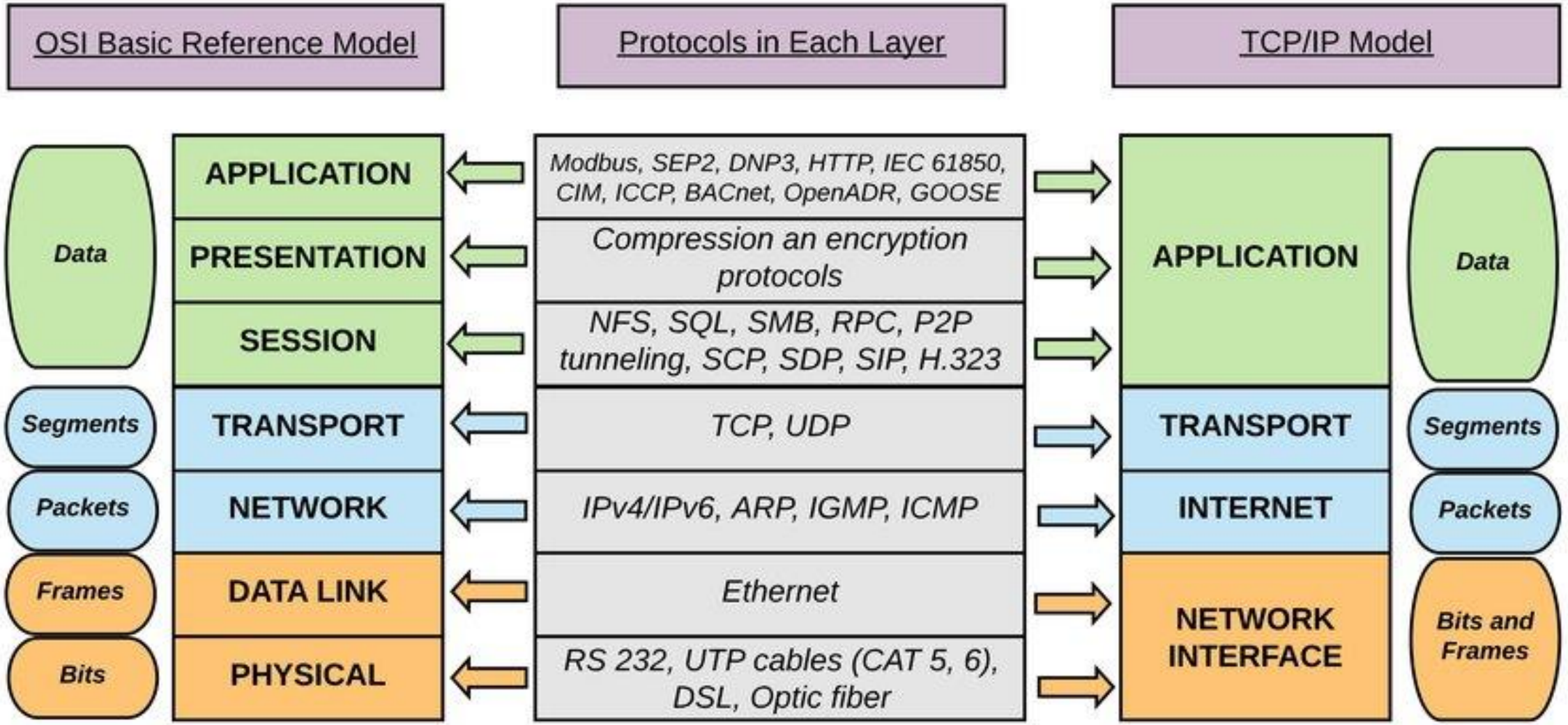
Networking Basics

- To help simplify things, computer networks are often described as layered systems.
- Different layers play different roles in the process of getting the data from one network device to another.
- There's lots of debate, though, on just how best to split up the networking layers.
- While the standard Open Systems Interconnection (OSI) network model uses seven layers, we'll use a simplified four-layer approach to describing the network functions:
  - ✓ The physical layer
  - ✓ The network layer
  - ✓ The transport layer
  - ✓ The application layer

OSI Model



OSI vs TCP/IP



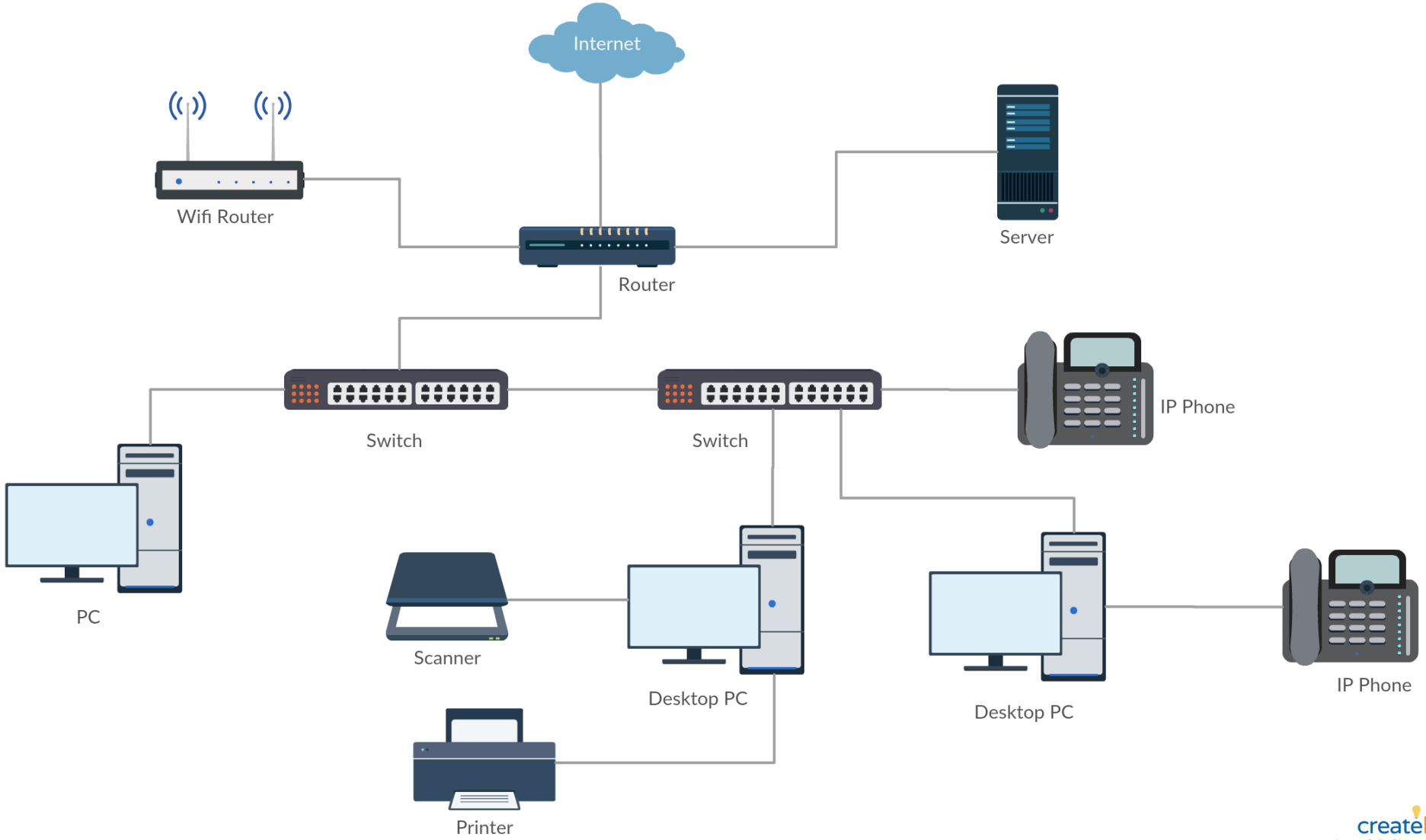
The Physical Layer

- The physical layer consists of the hardware required to connect your Linux system to the network.
- There are two main methods used to connect network devices:
  - ✓ wired
  - ✓ wireless

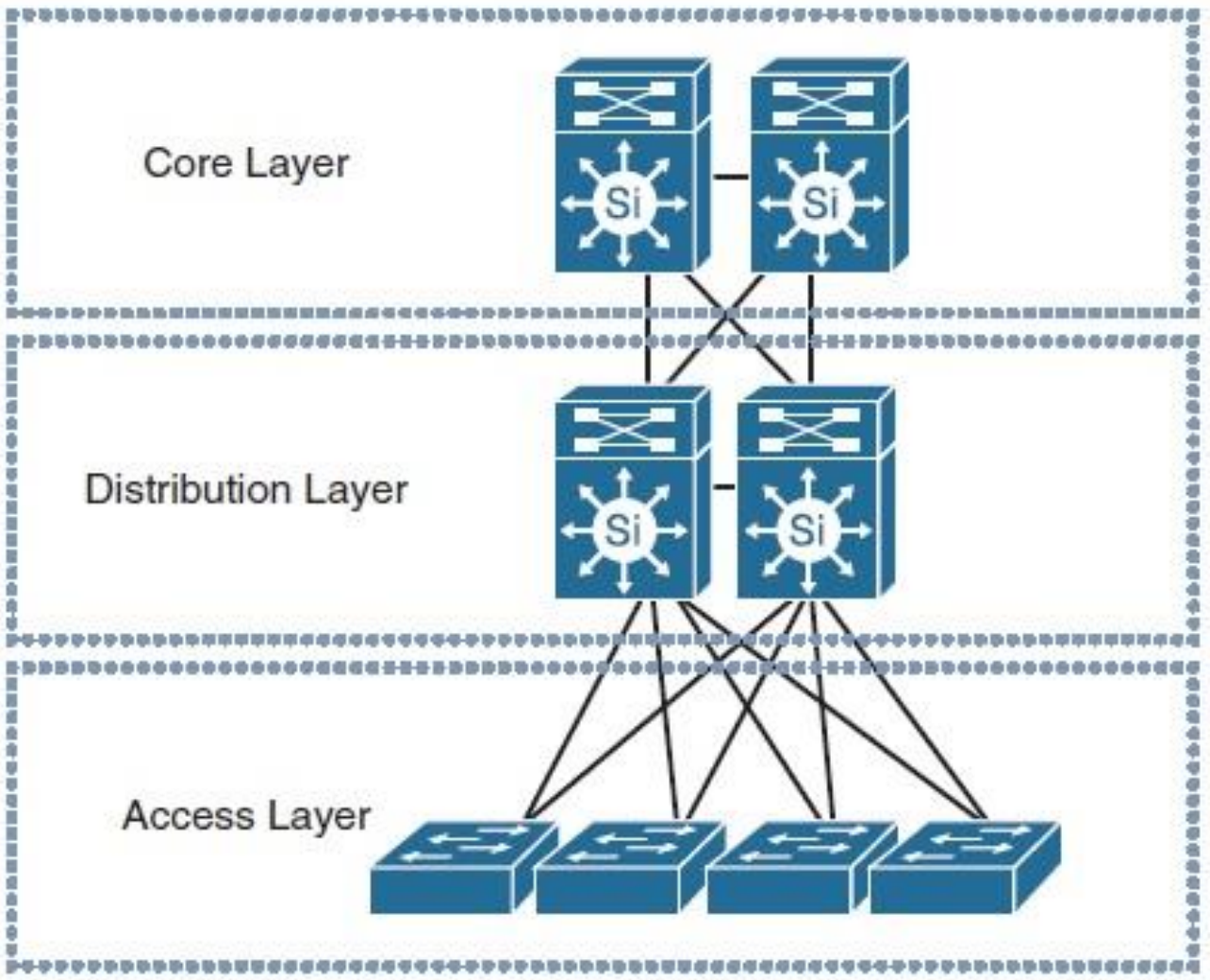
Wired Network

- Wired network connections use a series of network switches to connect network devices using special Ethernet cables.
- The network switch accepts data packets from the network device and then sends the data packets to the correct destination device on the network.
- For large office network installations, switches are usually connected in a cascade design to help reduce traffic load on the network.
  - ✓ Switches can be interconnected with one another to help segment the network traffic into smaller areas.

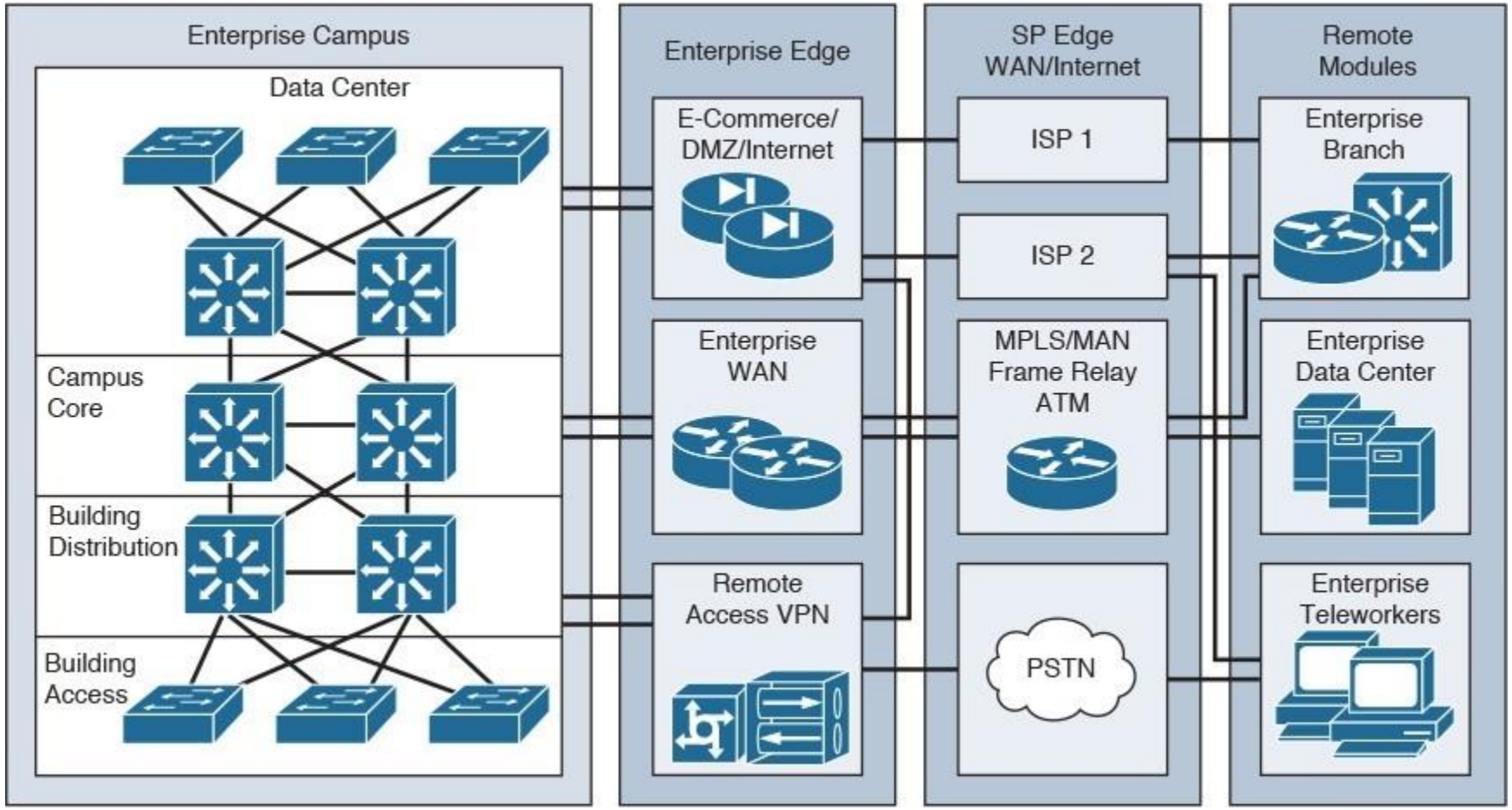
Wired Network



Cisco Three Tier Network Model



Wired Network



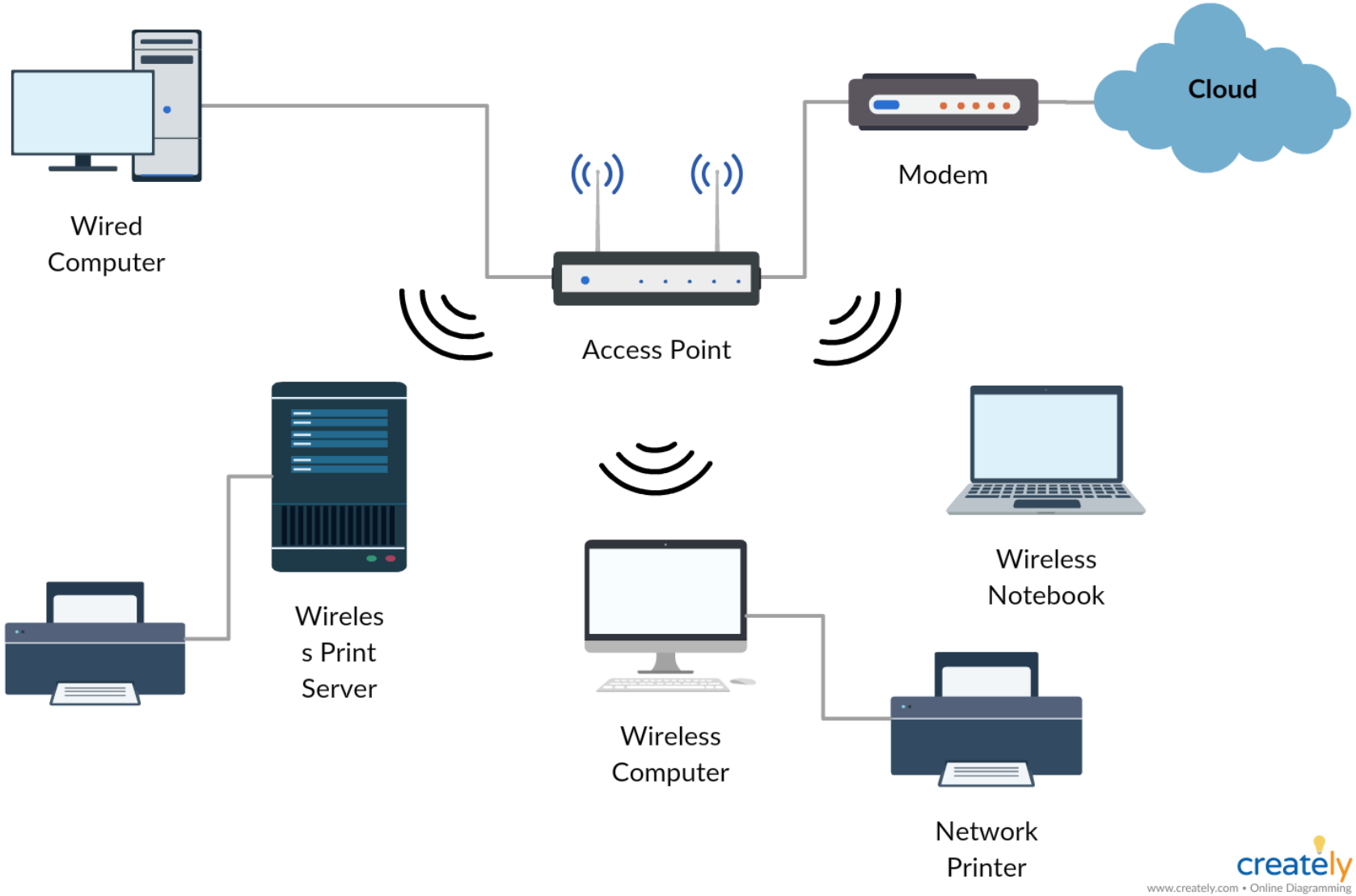
Wired Network

- While the term “wired” may make you think of copper cables, it can also apply to network connections that use fiber-optic cables.
- Fiber-optic cables use light to transmit data down a thin glass strand, achieving faster speeds and covering longer distances than conventional copper connections.
- Although wired networking can be cumbersome, it does provide the fastest network speeds (currently up to 100 gigabits per second).

Wireless Network

- Most small office and home networks utilize wireless networking.
- Instead of using physical wires or fiber cables to connect devices, wireless networking uses radio signals to transmit the data between the network device and a network access point.
- The access point works in a similar way to the switch in that it controls how data is sent to each network device communicating with it.
- Each access point uses a unique service set identifier (SSID) to identify it from other access points, which can be a text name or a number.
- You just tell your Linux system which access point to connect to by specifying the correct SSID value.

Wireless Network



Wireless Network

- **The downside to wireless networking is that you can't control where the radio signals travel.**
  - ✓ It's possible that someone outside of your home will see your access point signals and try to connect to them.
- **Because of that, it's important to implement some type of encryption security on your access point.**
  - ✓ Only devices using the correct encryption key can connect to the wireless access point.
- **Common wireless encryption techniques are**
  - ✓ Wired Equivalent Privacy (WEP),
  - ✓ Wi-Fi Protected Access (WPA),
  - ✓ Wi-Fi Protected Access version 2 (WPA2).

The Network Layer

- The network layer controls how data is sent between connected network devices, both in your local network and across the Internet.
- For data to get to the correct destination device, some type of network addressing scheme must be used to uniquely identify each network device.
  - ✓ The most common method for doing that is the **Internet Protocol (IP)**.
- To connect your Linux system to an IP network you'll need four pieces of information:
  - ✓ An IP address
  - ✓ A netmask value
  - ✓ A default router
  - ✓ A hostname

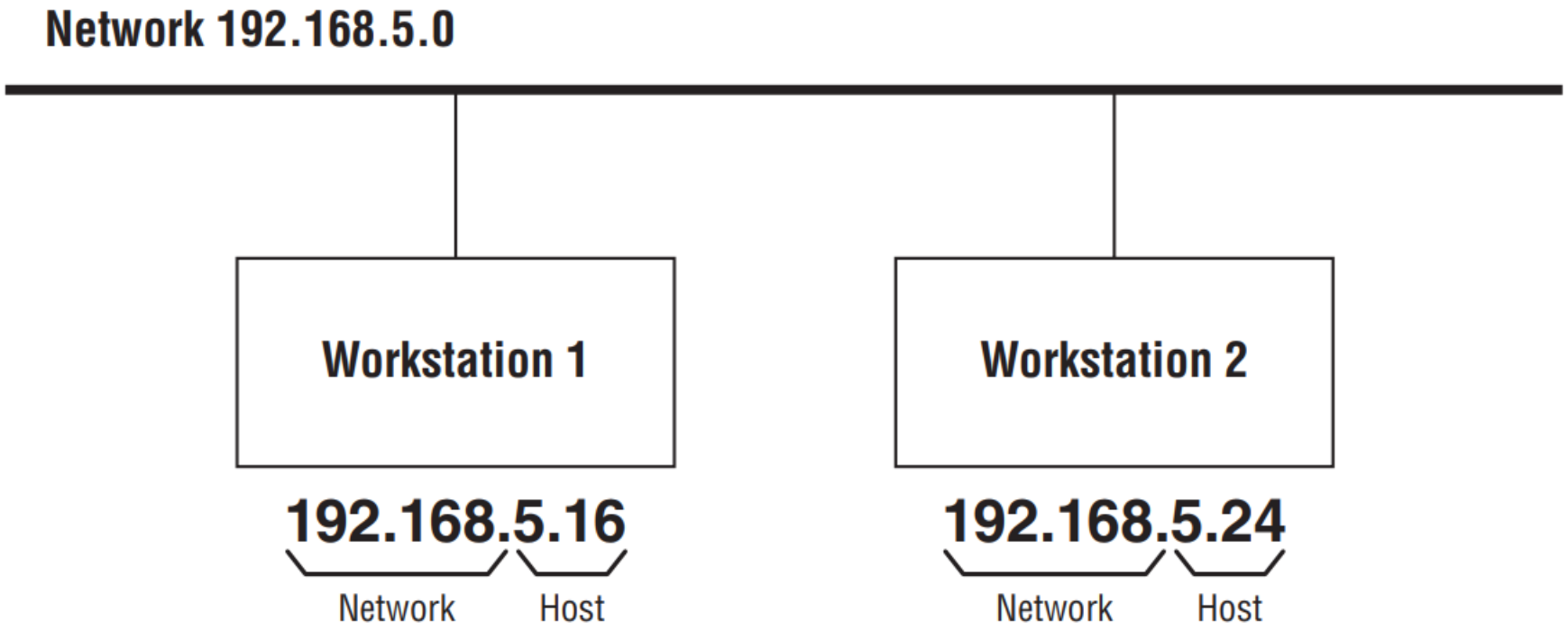
The IP Address

- In an IP network, each network device is assigned a unique 32-bit address.
- Networking layer software embeds the source and destination IP addresses into the data packet so that networking devices know how to handle the data packet and the Linux system knows which packets to read and which to ignore.
- To make it easier for humans to recognize the address, IP addresses are split into four 8-bit values, represented by decimal numbers, with a period between each value.
  - ✓ This format is called dotted-decimal notation.
  - ✓ For example, a standard IP address in dotted decimal notation looks like 192.168.1.10.

The IP Address

- **IP addresses are split into two sections.**
  - ✓ One part of the IP address represents the network address.
    - All devices on the same physical network have the same network address portion of their IP addresses.
    - For example, if your home network is assigned the network address 192.168.1.0, all of the network devices must start with the IP address 192.168.1.
  - ✓ The second part represents the host address.
    - Each device on the same network must have a unique host address.

The IP Address



The IPv6 Address

- To complicate things even further, an updated IP network protocol has been introduced called IP Version 6 (IPv6).
- The IPv6 networking scheme uses 128-bit addresses instead of the 32-bit addresses used by IP, which allows for lots more network devices to be uniquely identified on the Internet.
  - ✓ The IPv6 method uses hexadecimal numbers to identify addresses.
- The 128-bit address is split into eight groups of four hexadecimal digits separated by colons, such as

```
fed1:0000:0000:08d3:1319:8a2e:0370:7334
```

- If one or more groups of four digits is 0000, that group or those groups may be omitted, leaving two colons:

```
fed1::08d3:1319:8a2e:0370:7334
```

- However, only one group of zeroes can be compressed this way.

The IPv6 Address

- **The IPv6 protocol also provides for two different types of host addresses:**
  - ✓ Link local addresses
  - ✓ Global addresses
- **The IPv6 software on a host device automatically assigns the link local address.**
- **The link local address uses a default network address of `fe80::` and then derives the host part of the address from the media access control (MAC) address built into the network card.**
  - ✓ This ensures that any IPv6 device can automatically communicate with any other IPv6 device on a local network without any configuration.
- **The IPv6 global address works similarly to the original IP version:**
  - ✓ each network is assigned a unique network address, and each host on the network must have a unique host address.

Netmask Address

- The netmask address distinguishes between the network and host address portions in the IP address by using **1 bit** to show which bits of the 32-bit IP address are used by the **network** and **0 bits** to show which bits represent the **host address**.
- Since most people don't like working with binary numbers, the netmask address is usually shown in dotted-decimal format.
  - ✓For example, the netmask address 255.255.255.0 indicates the first three decimal numbers in the IP address represent the network address, and the last decimal number represents the host address.

Netmask Address

- When working on the Internet, it's crucial that no two physical Internet connections have the same IP address.
- To accomplish that, the Internet Assigned Numbers Authority (IANA) maintains strict control over the assignment of IP network addresses.
- However, not all networks need to be connected to the Internet, so to differentiate those networks, IANA has made the distinction between public and private IP networks.
- Specific subnetwork ranges are reserved for private IP networks:
  - ✓ 10.0.0.0 to 10.255.255.255
  - ✓ 172.16.0.0 to 172.31.255.255
  - ✓ 192.168.0.0 to 192.168.255.255
- These private IP addresses can't be used for Internet traffic; they work only on local networks.

Classless Inter-Domain Routing (CIDR)

- There is another way to represent netmask addresses called Classless Inter-Domain Routing (CIDR) notation.
- CIDR notation represents the netmask as just the number of masked bits in the IP address.
- CIDR notation is usually shown with a slash between the network address and the CIDR value.
  - ✓ Thus, the network 192.168.1.0 and netmask 255.255.255.0 would have the CIDR notation of 192.168.1.0/24.
- Although CIDR notation is becoming popular in the networking world, Linux configuration files still use the netmask value to define the network.

NAT

- **As you can imagine, with the popularity of the Internet, it didn't take long for IANA to run out of available public IP address networks.**
  - ✓ However, in a brilliant move, the idea of network address translation (NAT) saved the day.
- **A NAT server can take an entire private IP network and assign it a single public IP address on the Internet.**
  - ✓ This is how you can connect your entire home network to a single ISP Internet connection and everything works.

Default Router

- With IP and IPv6, devices can communicate directly only with other devices on the same physical network.
- To connect different physical networks together, you use a router.
  - A router passes data from one private network to another.
- Devices that need to send packets to hosts on remote networks must use the router as a go-between.
- Usually a network will contain a single router to forward packets to an upper-level network.
  - This is called a default router (or sometimes, a default gateway).
- Network devices must know the local default gateway for the network to be able to forward packets to remote hosts.
- Thus, for a device to communicate in an IP network, it must know three separate pieces of information:
  - Its own host address on the network
  - The netmask address for the local physical network
  - The address of a local router used to send packets to remote networks
- Here's an example of what you would need:
  - Host address: 192.168.20.5
  - Netmask address: 255.255.255.0
  - Default gateway: 192.168.20.1

Host Names

- With all of these IP addresses, it can be impossible trying to remember just what servers have what addresses.
- Fortunately for us, yet another network standard is available that can help out.
  - ✓ The **Domain Name System (DNS)** assigns a name to hosts on the network.
- With DNS, each network address is assigned a domain name (such as linux.org) that uniquely identifies the network, and each host in that network is assigned a unique host name, which is added to the domain name to uniquely identify the host on the network.
- Thus, to find the host `anisa` on the domain example.org, you'd use the DNS name `anisa.example.org`.
  - ✓ The DNS system uses servers to map host and domain names to the specific network addresses required to communicate with that server.
    - Servers responsible for defining the network and host names for a local network interoperate with upper level DNS servers to resolve remote host names.
- To use DNS in your network applications, all you need to configure is the address of the DNS server that services your local network.
  - ✓ From there, your local DNS server can find the address of any host name anywhere on the Internet.

Dynamic Host Configuration Protocol

- Trying to keep track of host addresses for all of the devices on a large network can become cumbersome.
- Keeping individual IP address assignments straight can be a challenge, and often you'll run into the situation where two or more devices accidentally are assigned the same IP address.
- The Dynamic Host Configuration Protocol (DHCP) was created to make it easier to configure client workstations, which don't necessarily need to use the same IP address all the time.
  - With DHCP, the client communicates with a DHCP server on the network using a temporary address.
  - The DHCP server then tells the client exactly which **IP address**, **netmask address**, **default gateway**, and even **DNS server** to use.
  - Each time the client reboots, it may receive a different IP address, but that doesn't matter as long as it's unique on the network.
- Although DHCP is great for clients, it's not a good idea to use for servers.
  - Servers need to have a fixed IP address so that clients can always find them.
- While it's possible to configure static IP addresses in DHCP, usually it's safest to manually configure the network information for servers.
  - This is called a static host address.

The Transport Layer

- **The transport layer can often be the most confusing part of the network.**
- **Whereas the network layer helps get data to a specific host on the network, the transport layer helps get the data to the correct application contained on the host.**
  - ✓ It does that by using ports.
- **Ports are sort of like apartment numbers.**
  - ✓ Each application that's running on a network server is assigned its own port number, just like different apartments in the same apartment building are assigned separate apartment numbers.
  - ✓ To send data to a specific application on a server, the client software needs to know both the server IP address (just like the apartment building address) and the transport layer port number (just like the apartment number).
- **Two common transport protocols are used in the IP networking world:**
  - ✓ Transmission Control Protocol (**TCP**)
  - ✓ User Datagram Protocol (**UDP**)

Transmission Control Protocol (TCP)

- The Transmission Control Protocol (TCP) transport protocol sends data using a guaranteed delivery method.
- It ensures that the server receives each portion of data that the client computer sends, and vice versa.
- The downside is that a lot of overhead is required to track and verify all of the data sent, which can slow down the data transfer speed.

User Datagram Protocol (UDP)

- **For data that's sensitive to transfer speed (such as real-time data like voice and video), that can cause unwanted delays.**
  - ✓ The alternative to this is the User Datagram Protocol (UDP) transport protocol.
- **UDP doesn't bother to ensure delivery of each portion of the data, it just sends the data out on the network and hopes it gets to the server!**
- **Though losing data may sound like a bad thing, for some applications (such as voice and video) it's perfectly acceptable.**
  - ✓ Missing audio or video packets just show up as blips and breaks in the final audio or video result.
  - ✓ As long as most of the data packets arrive, the audio and video is understandable.

Internet Control Message Protocol (ICMP)

- Though not used for sending application data, there is one more transport layer protocol that you'll need to know about.
- There's a need for network devices to communicate "behind the scenes" with each other, passing network management information around the network.
- The Internet Control Message Protocol (ICMP) provides a simple way for network devices to pass information such as error messages and network routing information to make it easy for each client to find the required resource on the network.

The Application Layer

- The application layer is where all the action happens.
- This is where the network programs process the data sent across the network and then return a result.
- Most network applications behave using the client/server paradigm.
  - ✓ With the client/server paradigm, one network device acts as the server, offering some type of service to multiple network clients (such as a web server offering content via web pages).
    - The server listens for incoming connections on a specific transport layer port assigned to the application.
    - The clients must know what transport layer port to use to send requests to the server application.
- To simplify that process, both TCP and UDP use well-known ports to represent common applications.
  - ✓ These port numbers are reserved so that network clients know to use

The Application Layer

Port	Protocol	Application
20	TCP	File Transfer Protocol (**FTP**) data
21	TCP	File Transfer Protocol (**FTP**) control messages
22	TCP	Secure Shell (**SSH**)
23	TCP	**Telnet** interactive protocol
25	TCP	Simple Mail Transfer Protocol (**SMTP**)
53	TCP & UDP	Domain Name System (**DNS**)
80	TCP	Hypertext Transfer Protocol (**HTTP**)
110	TCP	Post Office Protocol version 3 (**POP3**)
123	UDP	Network Time Protocol (**NTP**)
139	TCP	**NetBIOS** Session Service
143	TCP	Internet Message Access Protocol (**IMAP**)

Port	Protocol	Application
161	UDP	Simple Network Management Protocol (**SNMP**)
162	UDP	Simple Network Management Protocol trap
389	TCP	Lightweight Directory Access Protocol (**LDAP**)
443	TCP	Hypertext Transfer Protocol (**HTTPS**) over TLS/SSL
465	TCP	Authenticated SMTP (**SMTPS**)
514	TCP & UDP	Remote Shell (TCP) or Syslog (UDP)
636	TCP	Lightweight Directory Access Protocol over TLS/SSL (**LDAPS**)
993	TCP	Internet Message Access Protocol over TLS/SSL (**LDAPS**)
995	TCP	Post Office Protocol 3 over TLS/SSL (**POP3S**)

CONFIGURING NETWORK FEATURES

There are five main pieces of information you need to configure in your Linux system to interact on a network:

- ✓ The host address
- ✓ The network subnet address
- ✓ The default router (sometimes called the gateway)
- ✓ The system host name
- ✓ A DNS server address for resolving host names

Network Configuration Files

- Linux systems that utilize the `systemd` initialization method normally use the `systemd-networkd` daemon to detect network interfaces and automatically create entries for them in the network configuration files.
  - ✓ You can modify those files manually to tweak or change network settings if necessary.
- Unfortunately, though, no single standard configuration file exists that all distributions use.
  - ✓ **Debian-based**: `/etc/network/interfaces` file
  - ✓ **Red Hat–based**: `/etc/sysconfig/network-scripts` directory
  - ✓ **OpenSUSE**: `/etc/sysconfig/network` file

Network Configuration Files

Sample Debian network static configuration settings

```
auto eth0
iface eth0 inet static
address 192.168.1.77
netmask 255.255.255.0
gateway 192.168.1.254
iface eth0 inet6 static
address 2003:aef0::23d1::0a10:00a1
netmask 64
gateway 2003:aef0::23d1::0a10:0001
```

Sample Debian network DHCP configuration settings

```
auto eth0
iface eth0 inet dhcp
iface eth0 inet6 dhcp
```

Network Configuration Files

Sample CentOS network interface configuration settings

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp0s3
UUID=c8752366-3e1e-47e3-8162-c0435ec6d451
DEVICE=enp0s3
ONBOOT=yes
IPV6_PRIVACY=no
```

Sample CentOS network file configuration settings

```
NETWORKING=yes
HOSTNAME=mysystem
GATEWAY=192.168.1.254
IPV6FORWARDING=yes
IPV6_AUTOCONF=no
IPV6_AUTOTUNNEL=no
IPV6_DEFAULTGW=2003:aef0::23d1::0a10:0001
IPV6_DEFAULTDEV=eth0
```

Network Configuration Files

- You will also need to define a DNS server so that the system can use DNS host names.
- For `systemd` systems, the DNS server is generated by the systemd-resolved program.
- For legacy `SysVinit` systems, that's handled in the `/etc/resolv.conf` configuration file:

```
domain mydomain.com
search mytest.com
nameserver 192.168.1.1
```

- domain: defines the domain name assigned to the network.
  - ✓ By default, the system will append this domain name to any host names you specify.
- search: defines any additional domains used to search for host names.
- nameserver: where you specify the DNS server assigned to your network.
- Some networks can have more than one DNS server, just add multiple nameserver entries in the file.

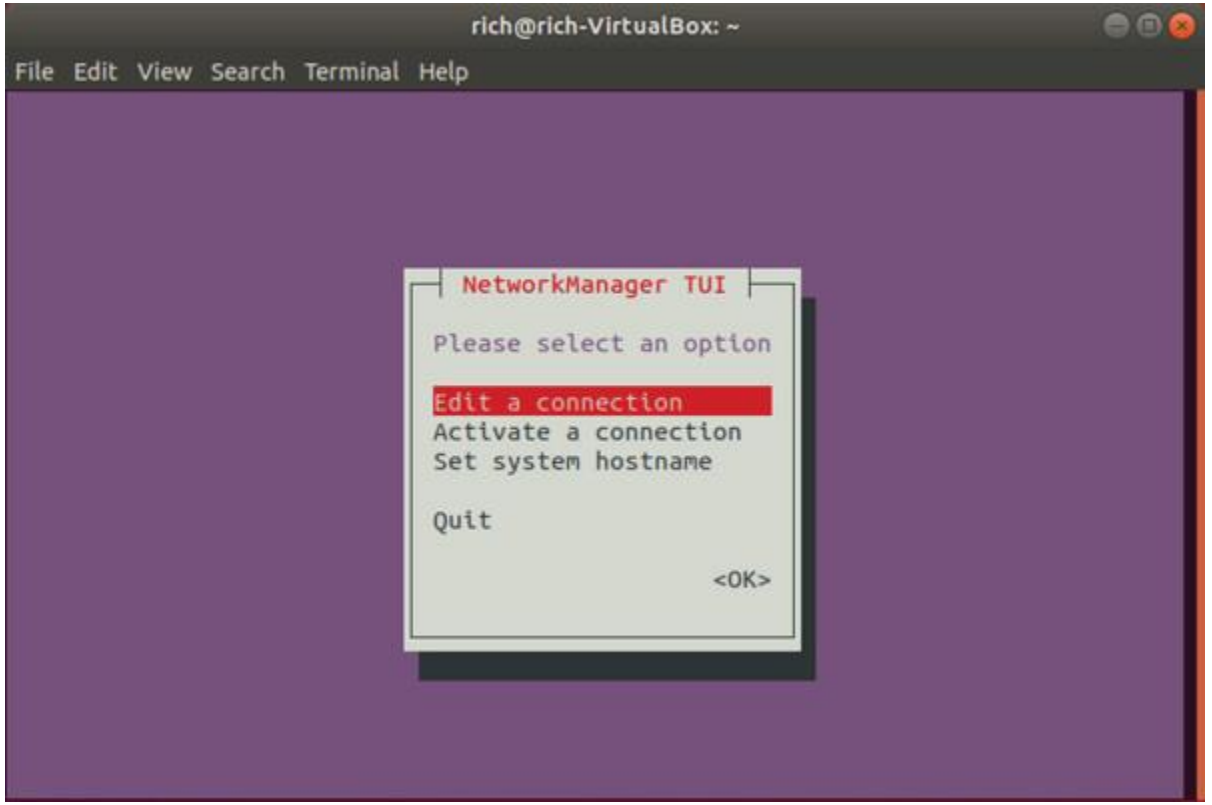
Command-Line Tools

- If you’re not working with a network configuration files, you’ll need to use the Linux command-line tools to set the network configuration information.
- You have quite a few different command-line tools at your disposal.

Network Manager Command-Line Tools

- **The Network Manager tool provides two different types of command-line tools:**
  - ✓ **nmtui** - Provides a simple text-based menu tool
    - The nmtui tool displays a stripped-down version of the graphical tool, where you can select a network interface and assign network properties to it.
  - ✓ **nmcli** - Provides a text-only command-line tool
    - The nmcli tool doesn't attempt to use any type of graphics capabilities, it just provides a command-line interface where you can view and change the network settings.
    - By default, the command displays the current network devices and their settings.

The Network Manager nmtui command-line tool



The default output of the nmcli command

```
$ nmcli
enp0s3: connected to Wired connection 1
"Intel 82540EM Gigabit Ethernet Controller (PRO/1000 MT Desktop Adapter)
ethernet (e1000), 08:00:27:2C:35:D2, hw, mtu 1500
ip4 default, ip6 default
inet4 192.168.1.77/24
route4 0.0.0.0/0
inet6 2600:1702:1ce0:eeb0::6d0/128
inet6 fe80::16d2:b8f:7f78:f3ed/64
route6 2600:1702:1ce0:eeb0::/60
route6 2600:1702:1ce0:eeb0::/64
route6 ::/0
...
```

- The nmcli command uses command-line options to allow you to set the network settings:

```
# nmcli con add type ethernet con-name eth1 ifname enp0s3 ip4 10.0.2.10/24 gw4 192.168.1.254
```

Legacy Tools

- If your Linux distribution doesn't support one of the Network Manager tools, there are usually legacy tools, including utilities from the `net-tools` package, available in most Linux distributions.
- Here are a few of the basic command-line tools that you can use:
  - ✓ **ethtool**: Displays Ethernet settings for a network interface
  - ✓ **ifconfig**: Displays or sets the IP address and netmask values for a network interface
  - ✓ **iwconfig**: Sets the SSID and encryption key for a wireless interface
  - ✓ **route**: Sets the default router address

Legacy Tools

- **ethtool** - query or control network driver and hardware settings

`ethtool devname`

- ✓ The **`ethtool`** command allows you to peek inside the network interface card Ethernet settings and change any properties that you may need to communicate with a network device, such as a switch.
- ✓ By default, the **`ethtool`** command displays the current configuration settings for the network interface

```
$ ethtool enp0s3
```

Legacy Tools

ifconfig - configure a network interface

```
ifconfig [-v] [-a] [-s] [interface]
ifconfig [-v] interface [aftype] options | address ...
```

- ✓ You can change features such as speed, duplex, and whether or not the network interface attempts to auto-negotiate features with the switch.
- ✓ The `ifconfig` command is a legacy command for configuring network device settings.
- ✓ It allows you to set the network address and subnet mask for a network interface:

```
$ sudo ifconfig enp0s3 down 10.0.2.10 netmask 255.255.255.0
```

- ✓ You can list all of the network interface settings on your system using the `ifconfig` command with no command-line options.

Legacy Tools

- **iwconfig** - configure a wireless network interface

```
iwconfig interface [OPTION]...
```

  - ✓ Before you can use the ifconfig command to assign an address to a wireless interface, you must assign the wireless SSID and encryption key values using the `iwconfig` command:

```
# iwconfig wlan0 essid "MyNetwork" key s:mypassword
```

- If you don't know the name of a local wireless connection, you can use the `iwlist` command to display all of the wireless signals your wireless card detects.
- Just specify the name of the wireless device, and use the scan option:

```
$ iwlist wlan0 scan
```

Legacy Tools

- **route** - show / manipulate the IP routing table

```
route [add] [del] target gw gateway
```

  - If your network is connected to multiple networks via multiple routers, you can manually create the routing table in the system by using the `add` or `del` command-line option for the route command.

```
# route add default gw 192.168.1.254
$ route
```

The iproute2 Package

- **ip** - show / manipulate routing, network devices, interfaces and tunnels

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
```

- ✓ **address** Display or set the IPv4 or IPv6 address on the device
- ✓ **addrlabel** Define configuration labels
- ✓ **l2tp** Tunnel Ethernet over IP
- ✓ **link** Define a network device
- ✓ **maddress** Define a multicast address for the system to listen to
- ✓ **monitor** Watch for netlink messages
- ✓ **mroute** Define an entry in the multicast routing cache
- ✓ **mrule** Define a rule in the multicast routing policy database
- ✓ **neighbor** Manage ARP or NDISC cache entries

The iproute2 Package

✓ **netns**	Manage network namespaces
✓ **ntable**	Manage the neighbor cache operation
✓ **route**	Manage the routing table
✓ **rule**	Manage entries in the routing policy database
✓ **tcpmetrics**	Manage TCP metrics on the interface
✓ **token**	Manage tokenized interface identifiers
✓ **tunnel**	Tunnel over IP
✓ **tuntap**	Manage TUN/TAP devices
✓ **xfrm**	Manage IPSec policies for secure connections

The iproute2 Package

```
$ ip address show
```

```
# ip address add 192.168.1.77/24 dev enp0s3
# ip route add default via 192.168.1.254 dev enp0s3
# ip link set enp0s3 up
```

Kernel Network Parameters

- You can fine-tune networking parameters for a network interface using the `/etc/sysctl.conf` configuration file.
- This file defines kernel parameters that the Linux system uses when interacting with the network interface.
- This has become a popular method to use for setting advanced security features, such as to disable responding to ICMP messages by setting the `icmp_echo_ignore_broadcasts` value to `1`, or if your system has multiple network interface cards, to disable packet forwarding by setting the `ip_forward` value to `0`.

Getting Network Settings Automatically

- If your network uses DHCP, you'll need to ensure that a proper DHCP client program is running on your Linux system.
- The DHCP client program communicates with the network DHCP server in the background and assigns the necessary IP address settings as directed by the DHCP server.
- Three common DHCP programs are available for Linux systems:
  - ✓ `dhcpcd`
  - ✓ `dhclient`
  - ✓ `pump`
- The `dhcpcd` program is becoming the most popular of the three, but you'll still see the other two used in some Linux distributions.

Bonding Network Cards

- Bonding allows you to aggregate multiple interfaces into one virtual network device.
- You can then tell the Linux system how to treat the virtual network device using three different basic types:
  - **Load balancing**: Network traffic is shared between two or more network interfaces.
  - **Aggregation**: Two or more network interfaces are combined to create one larger network pipe.
  - **Active/passive**: One network interface is live, and the other is used as a backup for fault tolerance.

Network interface bonding modes

Mode	Name	Description
0	balance-rr	Provides load balancing and fault tolerance using interfaces in a round-robin approach
1	active-backup	Provides fault tolerance using one interface as the primary and the other as a backup
2	balance-xor	Provides load balancing and fault tolerance by transmitting on one interface and receiving on the second
3	broadcast	Transmits all packets on all interfaces
4	802.3ad	Aggregates the interfaces to create one connection combining the interface bandwidths
5	balance-tlb	Provides load balancing and fault tolerance based on the current transmit load on each interface
6	balance-alb	Provides load balancing and fault tolerance based on the current receive load on each interface

Network interface bonding modes

- To initialize network interface bonding, you must first load the bonding module in the Linux kernel:

```
$ sudo modprobe bonding
```

- This creates a bond0 network interface, which you can then define using the ip utility:

```
$ sudo ip link add bond0 type bond mode 4
```

- Once you've defined the bond type, you can add the appropriate network interfaces to the bond using the ip utility:

```
$ sudo ip link set eth0 master bond0
$ sudo ip link set eth1 master bond0
```

- The Linux system will then treat the bond0 device as a single network interface utilizing the load balancing or aggregation method you defined.

BASIC NETWORK TROUBLESHOOTING

Once you have a Linux system running, there are a few things you can do to check that things are operating properly. This section walks through the commands you should know to monitor the network activity, including watching what processes are listening on the network and what connections are active from your system.

Sending Test Packets

- One way to test network connectivity is to send test packets to known hosts.
- Linux provides the `ping` and `ping6` commands to do that.
- The ping and ping6 commands send **Internet Control Message Protocol (ICMP)** packets to remote hosts using either the IP (ping) or IPv6 (ping6) protocols.
  - ✓ ICMP packets work behind the scenes to track connectivity and provide control messages between systems.
- If the remote host supports ICMP, it will send a reply packet back when it receives a ping packet.

Sending Test Packets

- **ping** - send ICMP ECHO_REQUEST to network hosts

```
ping [OPTIONS...]
```

- ✓ The basic format for the ping command is to just specify the IP address of the remote host.
- ✓ `-c count`
  - Stop after sending count ECHO_REQUEST packets.
  - With deadline option, ping waits for count ECHO_REPLY packets, until the timeout expires.

```
$ ping 8.8.8.8
```

```
$ ping -c 4 8.8.8.8
```

Tracing Routes

- Sending ping packets can be a useful tool, but there's not much you learn if the ping test packet doesn't come back.
- You have no way of knowing where in the path between your client and the remote server the network failed.
- That's where the `traceroute` and `tracepath` commands come in.

Tracing Routes

- **traceroute** - print the route packets trace to network host

```
traceroute [options]
```

- ✓ The `traceroute` command, and its IPv6 version `traceroute6`, shows the steps (called hops in network terms) taken to get from your local network to the remote host.
- ✓ It finds each router hop along the path by **sending ICMP packets** with short time-to-live (TTL) values so that each test packet can survive only one hop further than the previous packet.
- ✓ This can map all of the routers the test packets traverse getting to the final destination.

Tracing Routes

- **tracepath** - traces path to a network host discovering MTU along this path

```
tracepath [options]
```

  - ✓ The `tracepath` command, and its IPv6 version `tracepath6`, also show the steps taken to get to a remote host but use **UDP packets** instead of ICMP packets.
  - ✓ These have a better chance of being allowed to pass through routers in the Internet.

Finding Host Information

- **host** - DNS lookup utility

```
host [OPTIONS...]
```

  - ✓ Sometimes the problem isn't with network connectivity but with the DNS host name system.
  - ✓ You can test a host name using the host command.
  - ✓ The host command queries the DNS server to determine the IP addresses assigned to the specified host name.
  - ✓ By default, it returns all IP addresses associated with the host name.
  - ✓ Some hosts are supported by multiple servers in a load balancing configuration.
  - ✓ The host command will show all of the IP addresses associated with those servers.
  - ✓ You can also specify an IP address for the host command and it will attempt to find the host name associated with it.

Finding Host Information

```
$ host www.linux.org
www.linux.org is an alias for linux.org.
linux.org has address 107.170.40.56
linux.org mail is handled by 20 mx.iqemail.net.
$ host www.linux.org
www.linux.org is an alias for linux.org.
linux.org has address 107.170.40.56
linux.org mail is handled by 20 mx.iqemail.net.
$ host 107.170.40.56
56.40.170.107.in-addr.arpa domain name pointer iqdig11.iqnection.com.
```

Finding Host Information

dig - DNS lookup utility

```
dig [global-queryopt...] [query...]
```

- The dig command displays all DNS data records associated with a specific host or network.
- For example, you can look up the information for a specific host name.

```
$ dig www.linux.org
```

- Or you can look up DNS data records associated with a specific network service, such as a mail server:

```
$ dig linux.org MX
```

Finding Host Information

- **nslookup** - query Internet name servers interactively

```
nslookup [-option] [name | -] [server]
```

- If you need to look up DNS information for multiple servers or domains, the nslookup command provides an interactive interface where you can enter commands.

```
$ nslookup
```

Finding Host Information

- The `getent` command is a generic tool used to look for entries in any type of text database on the Linux system.
- When using `getent` to look up a host name, it parses through the host databases as defined in the `/etc/nsswitch.conf` configuration file to include the local `/etc/hosts` file.
- Thus, if you have a lot of local network hosts defined in the `/etc/hosts` file, it'll return those host names quicker than using the standard `host` or `dig` command.
- If you use the command without specifying a host name, it returns all hosts stored in the local `/etc/hosts` file

```
$ getent hosts
$ getent hosts mydatabase
```

ADVANCED NETWORK TROUBLESHOOTING

Besides the simple network tests shown in the previous section, Linux has some more advanced programs that can provide more advanced information about the network environment. Sometimes it helps to be able to see what network connections are active on a Linux system. There are two ways to troubleshoot that issue: the netstat command and the ss command.

The netstat Command

❖ **netstat** - Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships

```
netstat [OPTION...]
```

- ✓ The netstat command can provide a wealth of network information for you.
- ✓ By default, it lists all open network connections on the system.
- ✓ `-l, --listening` display listening server sockets
- ✓ `-t, --tcp` limit the output to just TCP connections
- ✓ `-u, --udp` limit the output to just UDP connections
- ✓ `-s, --statistics` display networking statistics (like SNMP)

Examining Sockets

- **ss** - another utility to investigate sockets

```
ss [options] [ FILTER ]
```

  - ✓ A program connection to a port is called a socket.
  - ✓ The `ss` command can link which system processes are using which network sockets that are active.
  - ✓ **-a, --all**
    - Display both listening and non-listening (for TCP this means established connections) sockets.
  - ✓ **-n, --numeric**
    - Do not try to resolve service names. Show exact bandwidth values, instead of human-readable.
  - ✓ **-p, --processes**
    - Show process using socket.
  - ✓ **-t, --tcp**
    - Display TCP sockets.

The netcat Utility

- **nc** - arbitrary TCP and UDP connections and listens

```
nc [OPTION...]
```

- ✓ You can use netcat to test just about any type of network situation, including building your very own client/server test tool.
- ✓ For example, to have netcat listen for incoming client connections on TCP port 2000, use the -l command option:

```
$ nc -l 2000
```

- ✓ Then, from another Linux system you can use netcat to connect to the listening server:

```
$ nc 192.168.1.77 2000
```

- ✓ When the connection is established, anything you type in either the client or the server side is sent to the other end of the connection and displayed. For example, when you type text at the client, like so:

```
$ nc 192.168.1.77 2000
This is a test
```

- ✓ it appears in the output back on the server:

```
$ nc -l 2000
This is a test
```