LINUX NETWORK ADMINISTRATION

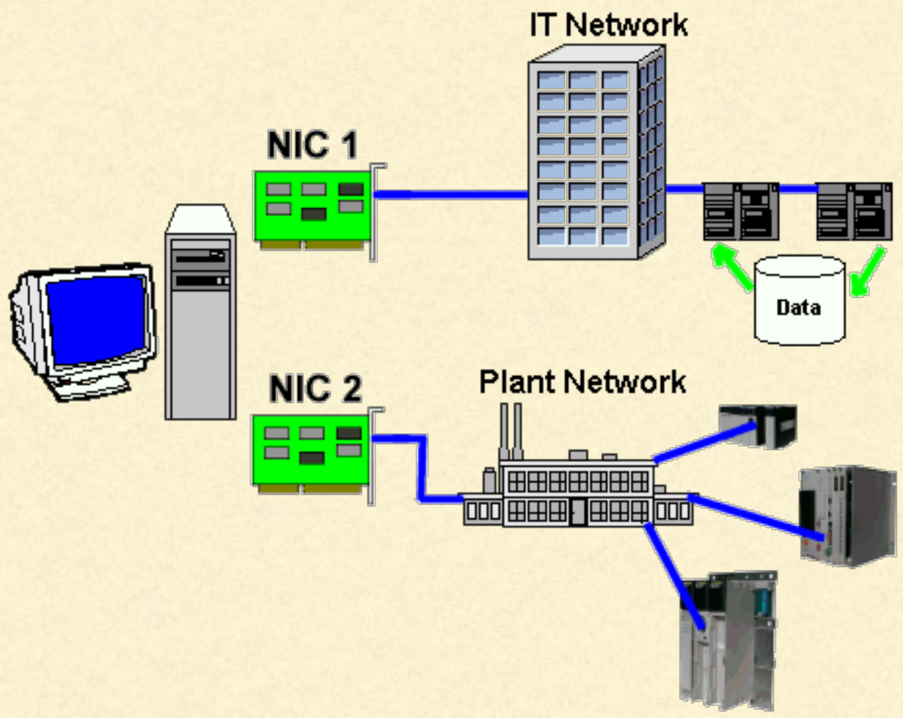
Network Configuration

COMP1071 - Summer 2020

INTERFACES

- Interfaces provide a way to connect a computer to a network for the purpose of communication between processes which may be on separate computers

- Interfaces may be physical (e.g. ethernet or wireless) or virtual (bridge, tunnel, virtual network, loopback)



PHYSICAL INTERFACES

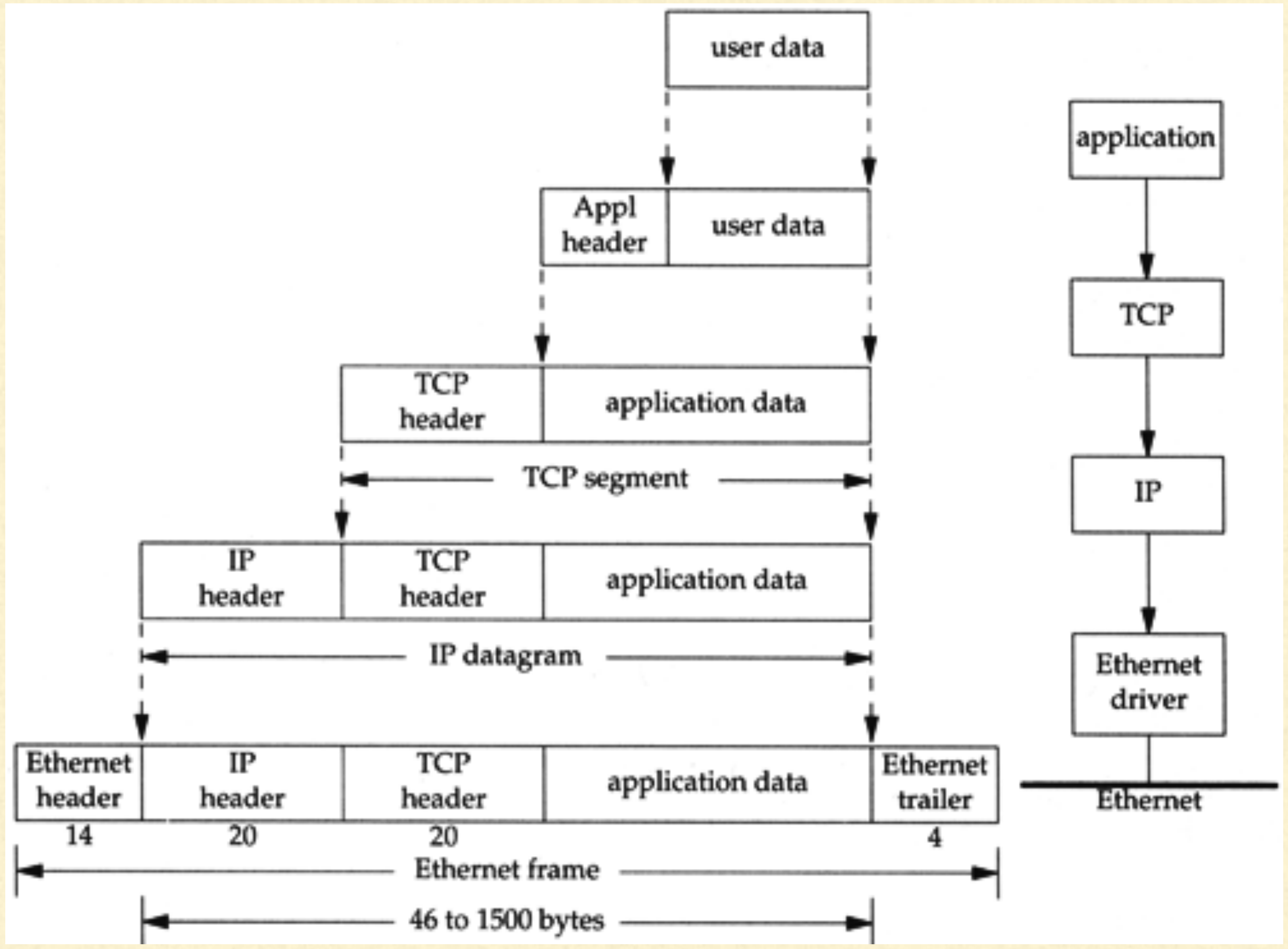
- Ethernet is the most common at speeds of 100Mb, 1Gb, 10Gb and is an implementation of the IEEE 802.3 communications standard
- Wi-fi is the next most common in many different speeds and implements various IEEE 802.11 standards (a/b/g/n/ac/ad) and while it may support multiple radios, each wi-fi interface is a single network interface for our purposes
- Other physical connection technologies include fibre, bluetooth, coaxial cable, DSL, etc.
- Physical interface names are composed of a short name for a driver and an instance number, e.g. eth0, ens33, eno1, enp0s3, wlan0

VIRTUAL INTERFACES

- Can be added onto existing physical interfaces and then used for VPNs, overlaid networks, vlans, and other purposes
- An overlaid interface is created when assigning additional IP addresses and masks to an existing interface
- Overlaid interfaces are traditionally named intf#:#, where the number after the colon is the virtual interface number from 1 to 255
- A vlan interface is created when using IEEE 802.1q tagging to create multiple virtual networks carried on a single ethernet connection with security added by the network infrastructure devices
- VLAN interfaces are traditionally named intf#.#, where the number after the period is the VLAN number
- netplan-based systems no longer use separate interface names for interfaces with overlaid or vlan addresses

DATA ENCAPSULATION

- TCP/IP is a suite of protocols and data formats that enable communications
- It makes use of data encapsulation to allow separation of functionality into layers, abstracting the details of each layer from the others so that the most suitable technologies can be used for each part of the communications task



VIEWING INTERFACES

- The traditional tool to view interfaces is the ifconfig command
- ifconfig can be used to change an interface's configuration or to view it
- ifconfig is deprecated in distros such as Ubuntu, use the ip command instead

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr b8:27:eb:9d:a5:5c
          inet addr:10.26.189.170  Bcast:10.26.189.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:99233 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7041 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41764065 (39.8 MiB)  TX bytes:689826 (673.6 KiB)
```

USING THE IP COMMAND

- The ip command can be used to view and change the running network interface configurations
- The ip command does not modify configuration files, so changes made with ip do not persist through reboots
- The ip command is a swiss army knife, it uses an object name on the command line to tell it what to work on, then a command to tell it what to do with that object
- Common objects include link, address, route
- Object commands depend on the object, but viewing current configuration is always done with the show or list command, and there is usually add and delete
- Objects and object commands can be abbreviated (e.g a for address, s for show, etc.) and list or help is the default object command if you don't give one

IP COMMAND EXAMPLES

```
# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:c5:43:41 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:6a:c1:d8 brd ff:ff:ff:ff:ff:ff
# ip l
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:c5:43:41 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:6a:c1:d8 brd ff:ff:ff:ff:ff:ff
# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:c5:43:41 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 82471sec preferred_lft 82471sec
    inet6 fe80::a00:27ff:fec5:4341/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:6a:c1:d8 brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.2/24 brd 172.16.3.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet 172.16.4.2/24 brd 172.16.4.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet 172.16.5.2/24 brd 172.16.5.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet 192.168.57.4/24 brd 192.168.57.255 scope global dynamic enp0s8
       valid_lft 415sec preferred_lft 415sec
    inet6 fe80::a00:27ff:fe6a:c1d8/64 scope link
       valid_lft forever preferred_lft forever
```

INTERFACE IDENTIFICATION

- lshw is a command that shows the hardware which Linux has recognized on our system
- It can be limited to the network hardware devices, making it easier to find the device names to use for networking commands

```
# lshw -class network
  *-network
       description: Ethernet interface
       product: 82579LM Gigabit Network Connection
       vendor: Intel Corporation
       physical id: 19
       bus info: pci@0000:00:19.0
       logical name: eno1
       version: 04
       serial: ec:a8:6b:8d:ab:d7
       size: 1Gbit/s
       capacity: 1Gbit/s
       width: 32 bits
       clock: 33MHz
       capabilities: pm msi bus_master cap_list ethernet physical tp 10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotiation
       configuration: autonegotiation=on broadcast=yes driver=e1000e driverversion=3.2.6-k duplex=full firmware=0.13-4 latency=0 link=yes multicast=yes port=twisted pair speed=1Gbit/s
       resources: irq:26 memory:f7c00000-f7c1ffff memory:f7c39000-f7c39fff ioport:f080(size=32)
  *-network DISABLED
       description: Ethernet interface
       physical id: 1
       logical name: virbr0-nic
       serial: 52:54:00:c6:09:b4
       size: 10Mbit/s
       capabilities: ethernet physical
       configuration: autonegotiation=off broadcast=yes driver=tun driverversion=1.6 duplex=full link=no multicast=yes port=twisted pair speed=10Mbit/s
```

INTERFACE HARDWARE CONFIGURATION

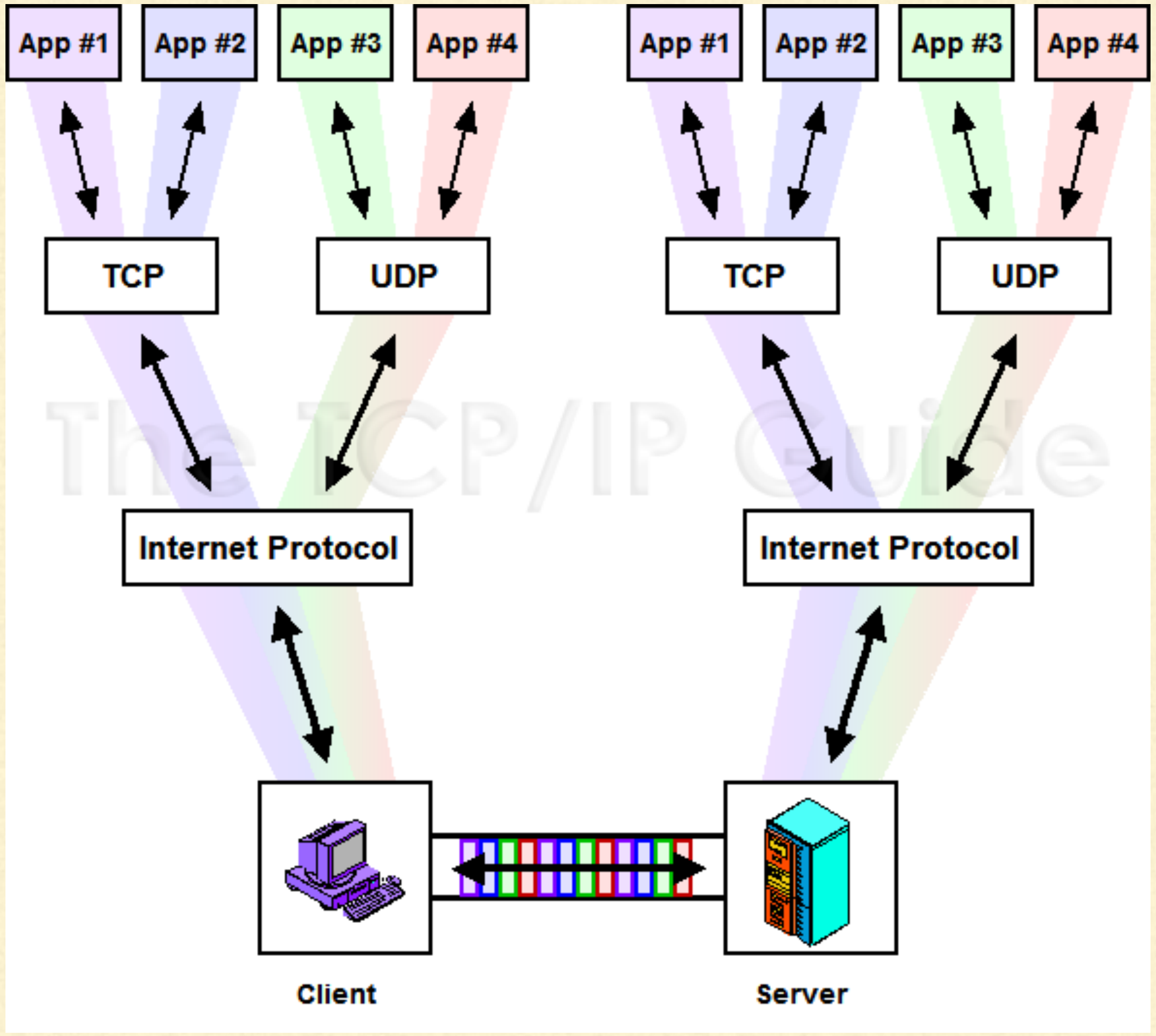
- ethtool and mii-tool are tools you can use to show detailed hardware configuration information for ethernet interfaces
- both may give you strange output for virtual interfaces such as those found in virtual machines

```
# ethtool eno1
Settings for eno1:
        Supported ports: [ TP ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Speed: 1000Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 1
        Transceiver: internal
        Auto-negotiation: on
        MDI-X: on (auto)
        Supports Wake-on: pumbg
        Wake-on: g
        Current message level: 0x00000007 (7)
                               drv probe link
        Link detected: yes

# mii-tool eno1
eno1: negotiated 1000baseT-FD flow-control, link ok
```

ADDRESSING

- An IP address uniquely identifies a communications end point host machine
- You need to know your destination's IP address if you want to communicate with them
- TCP/IP uses a combination of IP address (V4 or V6), protocol (TCP, UDP, etc.), and port (0-65535) to fully identify a communications end point (sometimes called a socket, used by a process)



HOST NAMES VS. ADDRESSES

- Addresses for TCP/IP are either 4 or 16 bytes (IPV4 vs. IPV6)

192.168.1.1
127.0.0.1
2607:f8b0:400b:80a::200e

- We use names instead of addresses for convenience and flexibility

router
localhost
google.com

- Our host's primary name is kept in /etc/hostname and is used to set the name of the machine at boot

myhostname

- Name to address conversion is automated using /etc/hosts

```
192.168.1.1                         router
2607:f8b0:400b:80a::200e   google.com
127.0.0.1                        localhost
```

ADDRESSING EXERCISE

- Use ip addr show or ifconfig to show the IP addresses your computer will respond to and use
- Use netstat with no options, with the -tp options, and with the -lp options to review current TCP/IP connection endpoints and processes
- Use netstat -s to review TCP/IP statistics
- Identify and try using the ss command to get the same information
- Use nmap to see what addresses are in use on the lan

INTERFACE CONTROL

- ifup and ifdown are traditionally used to turn interfaces on and off with their default configuration, normally at boot and shutdown
- /etc/network/interfaces traditionally contains the default configurations for non-GUI managed interfaces
- GUIs change the rules and make up their own rules
- ifup -a runs at boot in systems using the interfaces-style files
- netplan is a newer method in some Linux distros

/ETC/NETWORK/INTERFACES

- Two main lines control each interface
- auto *interfacename*
- iface *interfacename* inet **static**|**dhcp**|manual|**loopback**
- Additional lines must be present for static, minimum is address and netmask

```
auto lo eth0 eth1 eth1:1
iface lo inet loopback

iface eth0 inet dhcp

iface eth1 inet static
    address 192.168.1.2/24
    gateway 192.168.1.1

iface eth1:1 inet dhcp
```

NETPLAN

- The netplan tools use YAML files to specify network configurations in the /etc/netplan directory
- The netplan configuration files are named with a .yaml suffix and have the configuration created during system install
- They use mappings and tokens to specify cumulatively applied parameters
- An interface can appear in multiple YAML files and parameters will be sequentially and cumulatively applied
- Use netplan apply after making changes to any YAML config file

DEFAULT NETPLAN FILE

- A mapping contains all the mappings indented below it
- A mapping can contain more mappings or tokens for a mapping
- A token can be a single value such as true, or 192.168.1.1
- A token can be a list of values such as [1, 2, 3]
- Refer to the man page for details on what can be put into a mapping

```
network:
    ethernets:
        enp0s3:
            addresses: []
            dhcp4: true
        enp0s8:
            addresses:  []
            dhcp4: true
    version: 2
```

SERVER NETPLAN FILE

- Server addresses are statically assigned
- May have private routes
- Nameservers and search domains are statically specified

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s8:
      addresses:
        - 192.168.2.61/24
        - 172.16.4.2/24
     routes:
        - to: 172.16.6.0/24
          via: 172.16.4.1
        - to: 172.16.9.0/24
          via: 172.16.4.1
      nameservers: [192.168.2.1]
      search: [bigcorp.com]
```

NETPLAN AND VLANS

- Netplan sets configuration parameters for matching devices in mappings
- VLANs are a special version of an ethernet device with a VLAN id added to it (see IEEE 802.1q tagging)
- To configure a netplan vlan, create a vlans mapping in a /etc/netplan yaml file and add the VLAN name, id, link, and configuration details for that virtual interface
- The name of the device to create for the vlan virtual interface can be set by simply putting one in the vlans mapping in netplan

```
vlans:
  vlan-name:
    id: 10
    link: ens38
    addresses:
      - 172.16.3.2/24
```

NETWORK CONNECTION VERIFICATION

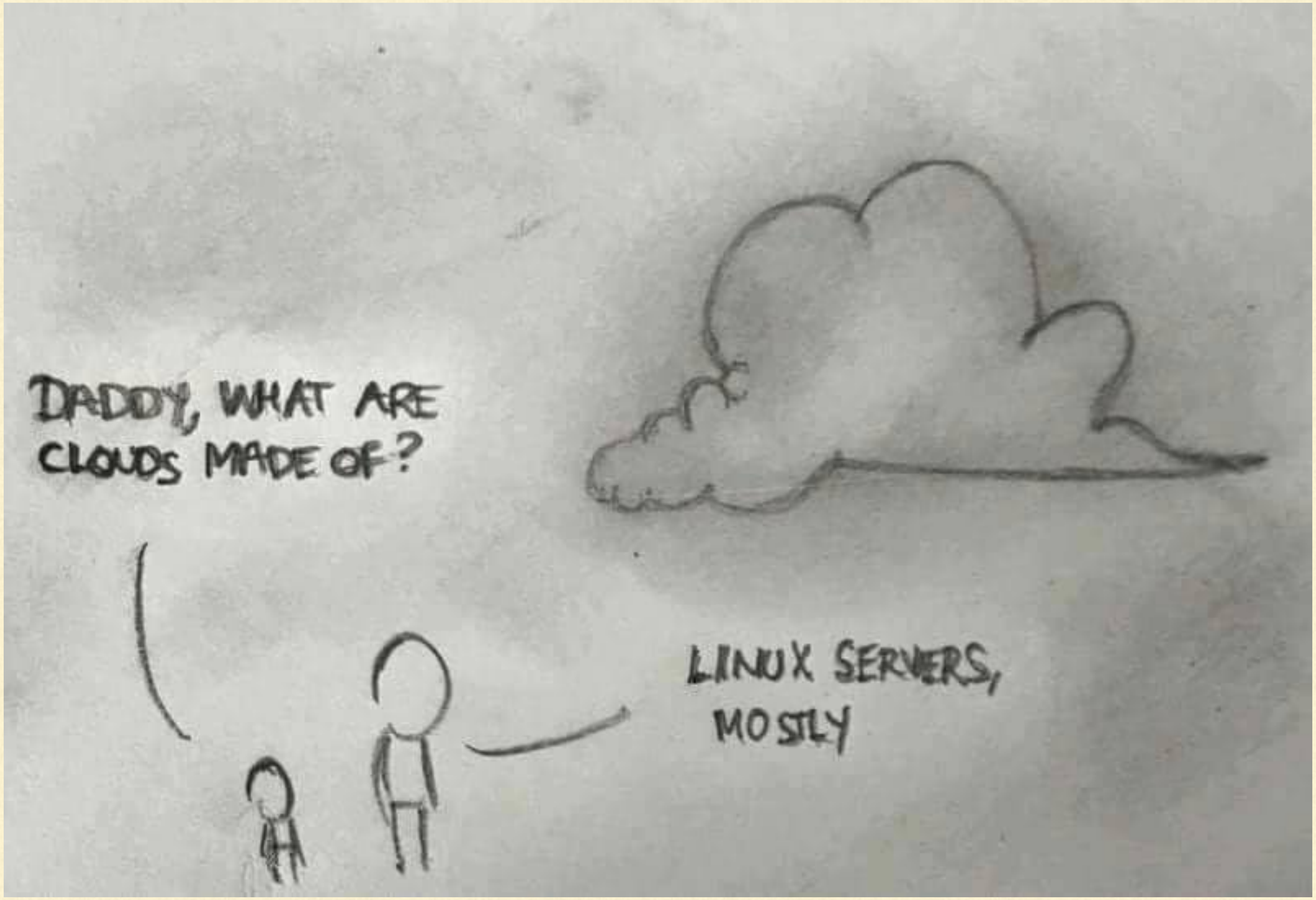
- Once an interface has an IP address assigned and is turned on, it can be used to send and receive data
- Some basic tools for checking network connections include ping, nmap, ss, and netstat
- ping can bounce a diagnostic (ICMP echo) packet off a host
- nmap can be used to generate many kinds of network traffic
- ss and netstat can be used to examine interface statistics and current connections

INTERFACES EXERCISE

- Use lshw -C network to review your available network hardware
- Use ethtool on ethernet interfaces to review your ethernet configuration
- Use ip link show to review interfaces which are configured
- Use netstat -i and ip -s link to review your network statistics

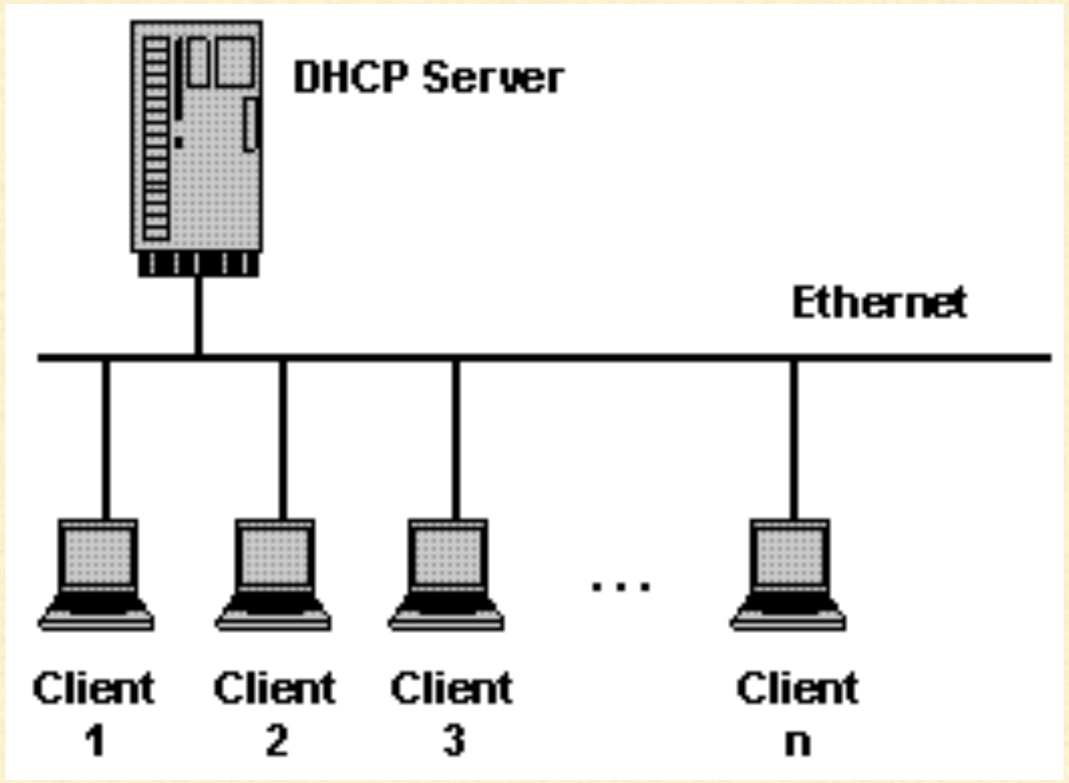
WIRELESS INTERFACES

- iwlist and iwconfig are your primary tools to scan and configure wi-fi interfaces
- wlan0 is a typical device name for a wi-fi connection
- Automatic connection establishment can be done by configuring connections in /etc/wpa_supplicant/wpa_supplicant.conf
- Wireless interfaces are not typically employed on servers for reliability reasons


DADDY, WHAT ARE CLOUDS MADE OF?
LINUX SERVERS, MOSTLY

DHCP CONCEPTS

- DHCP, the Dynamic Host Configuration Protocol, is used to provide automated interface configuration to hosts on a network
- A dhcp client retrieves information from a dhcp server
- Clients can choose what to use from the server without telling the server what they used



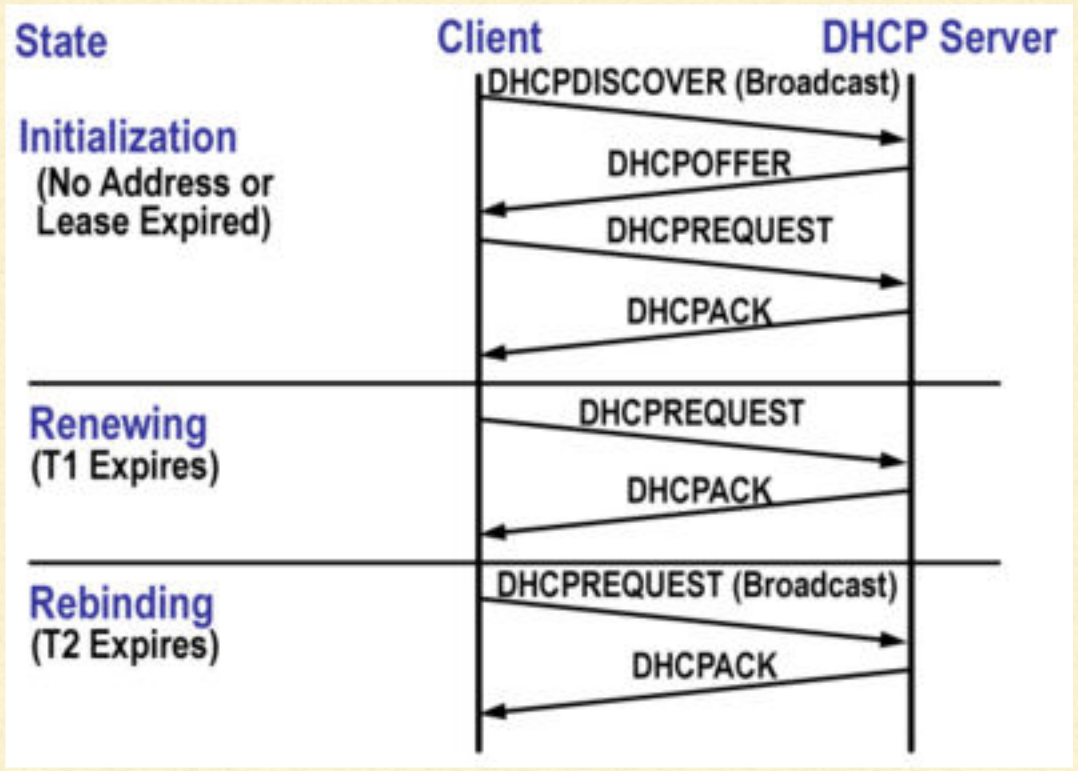
DHCP CLIENT

- Default configuration invoked by using the dhcp keyword on the iface line in /etc/network/interfaces or by setting the dhcp4 attribute to true in your netplan file

- Retrieves and by default uses ip address and netmask, DNS server(s), default gateway

- Expects server to be directly connected on LAN, DHCP uses broadcast

- systemd-resolve --status command can show what has been received from DHCP servers on newer distros



DHCP SERVER

- Server provides configuration in an advisory role
- Can supply many things, usually just ip address and netmask from a pre-defined range, default gateway, and DNS server(s)
- Can provide local data, or perform proxy service for a remote dhcp server
- Configurations are provided to clients on a lease basis, they expire if not renewed by the client on a regular basis

ISC DHCP SERVER

- Simple, easy to configure server, open source, free
- Package name is isc-dhcp-server
- Configuration in /etc/dhcp/dhcpd.conf, many examples in the default file
- Provides individual configurations per interface

ISC DHCP CONFIGURATION

- /etc/dhcp/dhcpd.conf for ISC server
- /etc/dhcp/dhclient.conf, /etc/dhcp/dhclient-{enter,exit}-hooks.d for client
- The man page and example files are the documentation

DHCP LOGS

- /var/log/messages or /var/log/syslog by default
- Automatically aged by logrotate

ROUTE TABLE

- The route table is a table in memory used by the kernel whenever it is asked to send data using TCP/IP
- It provides a list of destinations and identifies where to send data so that it can reach those destinations
- Traffic for directly connected hosts is sent to the interface for that network
- Traffic for remote destinations is sent to a router that knows how to reach the destination network
- A default route is usually set up so that data packets not destined for local networks can be sent to remote hosts such as those found on the internet

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.3.1     0.0.0.0         UG    0      0        0 br0
10.23.134.0     0.0.0.0         255.255.255.0   U     0      0        0 lxdbr0
192.168.3.0     0.0.0.0         255.255.255.0   U     0      0        0 br0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

IMPLICIT ROUTES

- Setting an address on an interface creates an entry in the route table using the address and netmask assigned to that interface
- Packets sent to directly connected hosts use the ARP protocol (a broadcast protocol) to find hardware (MAC) addresses for the destination IP addresses
- To send data to networks not directly connected, we need a host which can reach those networks to pass traffic on to the destination for us
- That forwarding of traffic is called routing and is accomplished by setting the router's MAC address as the destination MAC address in packets even though the destination IP is still the address of the remote host

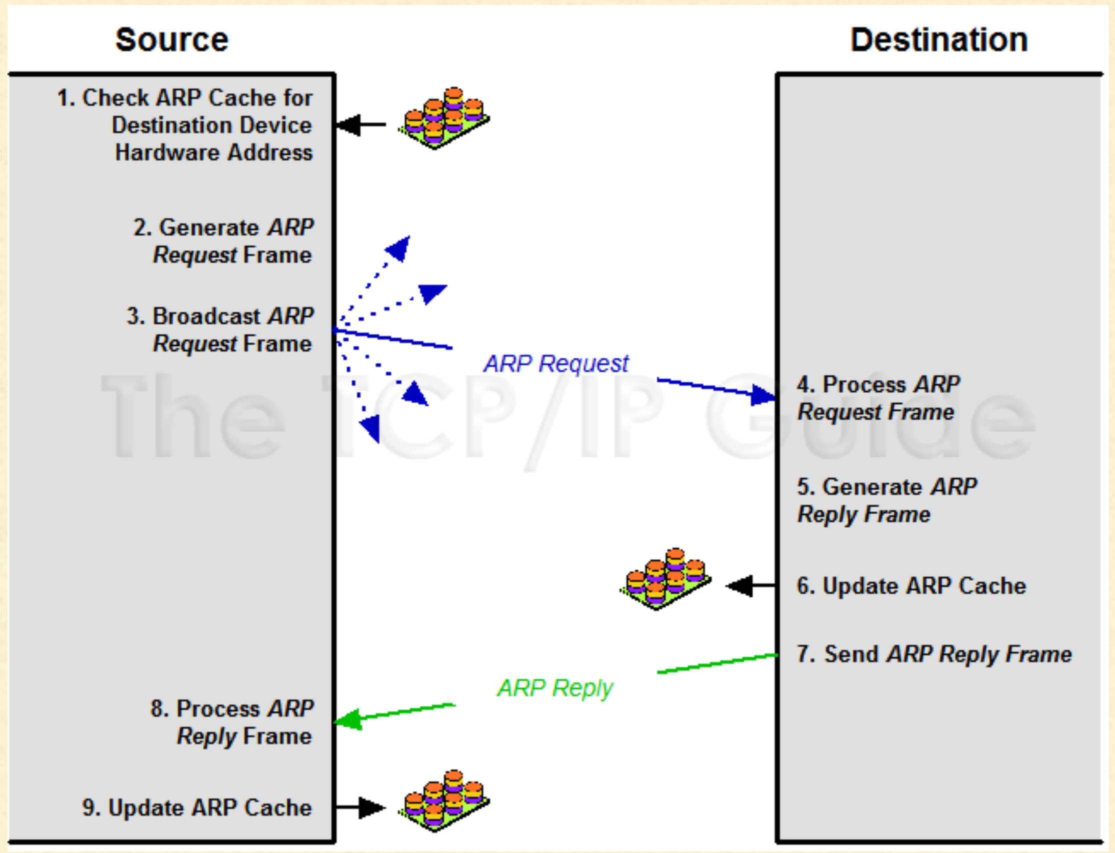

Source
Destination
1. Check ARP Cache for Destination Device Hardware Address
2. Generate ARP Request Frame
3. Broadcast ARP Request Frame
ARP Request
4. Process ARP Request Frame
5. Generate ARP Reply Frame
6. Update ARP Cache
7. Send ARP Reply Frame
ARP Reply
8. Process ARP Reply Frame
9. Update ARP Cache
The TCP/IP Guide

Image courtesy of http://www.tcpipguide.com/free/index.htm

STATIC ROUTING

- A static route is one which is configured explicitly using the route or ip route add command, and remains in place until removed by the route or ip route delete command, or the system shuts down
- It is primarily used to provide private routes or handle routes through devices which do not support dynamic routing protocols
- Zebra (part of Quagga) can be used to automate static route configuration using a Cisco-like interface for systems that do not use netplan
- netplan systems simply add routing entries to their YAML files
- IP forwarding must be enabled to provide routing service

DYNAMIC ROUTING

- Linux supports all major dynamic routing protocols
- The Quagga package provides dynamic routing services using a configuration file syntax similar to Cisco IOS
- IP forwarding can be enabled in /etc/sysctl.conf
- vtysh can be used to send Cisco IOS-like commands to quagga once it is running

WORKING WITH THE ROUTE TABLE

- netstat -r shows the route table
- route is used to add routes (route add), delete routes (route delete), and show the route table
- traceroute sends packets to a destination with increasing TTL values, causing them to fail at the first router, then the next and so on
- The diagnostic packets coming back from each router allows traceroute to display a list of the routers that handled the packets on the way
- The -n option can be used on these commands to show addresses instead of hostnames

FIREWALL

- A firewall is a piece of software that can examine packets and allow, deny, log, or modify them, primarily implemented to prevent unwanted traffic
- The filter software commonly used in Linux distributions is called netfilter, and is primarily configured and examined with the iptables command
- The iptables command can apply rules (tests with consequences) to the packets
- The default firewall state in Ubuntu and many others is that the firewall is disabled, and the default ruleset allows all traffic

UFW

- UFW simplifies firewall management
- It uses profiles stored in /etc/ufw/applications.d/ to name groups of ports and protocols
- It has built-in iptables rules for common firewall tasks stored in /etc/ufw/before and /etc/ufw/after directories

```
# ufw allow 22/tcp

# ufw allow profile

# ufw app list

# ufw app info

# ufw enable

# ufw disable
```

FIREWALL EXERCISE

- Add a rule to your firewall using ufw allowing ssh connections for your command line access
- Check your ufw status
- Enable ufw
- Recheck your ufw status
- Review the output of iptables -L to see what got added to the kernel's firewall rules tables
- Examine the contents of the /etc/ufw directory to see what was put there by the ufw package installation

INTERESTING COMMANDS

ifconfig
iwconfig
route
ufw
ip
ifup
ifdown

lshw
iwlist
netstat
ethtool
mii-tool
ping

nmap
traceroute
iptables
ufw

INTERESTING FILES

/etc/hostname

/etc/hosts

/etc/network/interfaces, /etc/wpa_supplicant/wpa_supplicant.conf

~/.ssh/authorized_keys, ~/.ssh/known_hosts

/etc/ufw/applications.d, /etc/ufw/before, /etc/ufw/after

/etc/sysctl.conf