**FTP**

Steps

* sudo apt update
* sudo apt install vsftpd
* sudo systemctl status vsftpd
* sudo systemctl start vsftpd
* sudo nano /etc/vsftpd.conf
  + Set anonymous\_enable=NO
  + Set local\_enable=YES
  + Set write\_enable=YES
  + Uncomment chroot\_local\_user=YES
  + pasv\_min\_port=10000
  + pasv\_max\_port=10100
* sudo systemctl restart vsftpd
* sudo ufw allow ftp
  + sudo ufw allow 10000:10100/tcp
* sudo useradd -m testuser
* sudo passwd testuser
* hostname or ifconfig
* sudo ftp your-server-name

After connecting to the FTP server, we can run these commands:

* get
* mget
* put
* mput
* mkdir
* rmdir
* delete
* mdelete

**TELNET**

Steps

* sudo apt update
* sudo apt install telnetd xinetd
* sudo nano /etc/xinetd.d/telnet

service telnet

{

disable = no

flags = REUSE

socket\_type = stream

wait = no

user = root

server = /usr/sbin/in.telnetd

log\_on\_failure += USERID

}

* sudo systemctl restart xinetd
* sudo systemctl status xinetd\
* sudo ufw allow 23/tcp
* telnet server-ip-address
* date
* ls
* uptime
* exit

**Wireshark**

* ip.addr == 192.168.1.1
* http
* tcp
* udp
* tcp.port == 443
* frame contains "youtube.com"
* and, or, not

**NMAP**

Uses

* Real time information of a network
* Detailed information of all the IPs activated on your network
* Number of ports open in a network
* Provide the list of live hosts
* Port, OS and Host scanning

Commands

* **Basic Syntax:** nmap [options] [target]
* **Installing Nmap:** sudo apt-get install nmap
* **Scan using Hostname:** nmap [www.geeksforgeeks.org](http://www.geeksforgeeks.org)
* **Scan using IP Address:** nmap 172.217.27.174
* **To scan using “-v” option:** nmap -v [www.geeksforgeeks.org](http://www.geeksforgeeks.org)
* **To scan multiple hosts:** nmap 103.76.228.244 157.240.198.35 172.217.27.174
* **To scan the whole subnet:** nmap 103.76.228.\*
* **To scan specific range of IP address:** nmap 192.168.29.1-20
* **To scan to detect firewall settings:** sudo nmap -sA 103.76.228.244
* **To identify Hostnames:** sudo nmap -sL  103.76.228.244
* **To scan from a file:** nmap -iL input.txt
* **To get some help:** nmap -h
* **Ping Scan (-sn)**: nmap -sn 192.168.1.0/24
* **Port Scan (-p)**: nmap -p 1-1000 192.168.1.1
* **Service Version Detection (-sV)**: nmap -sV 192.168.1.1
* **OS Detection (-O)**: nmap -O 192.168.1.1
* **Aggressive Scan (-A)**: nmap -A 192.168.1.1
* **UDP Scan (-sU)**: nmap -sU 192.168.1.1
* **SYN Scan (-sS)**: nmap -sS 192.168.1.1
* **FIN, Xmas, and Null Scans (-sF, -sX, -sN)**: nmap -sX 192.168.1.1
* **Decoy Scan (-D)**: nmap -D RND:10 192.168.1.1

**Packet Tracer**

Steps

* Add PC, Wireless Router, Laptop, Cable Modem, Cloud (Internet), Server
* Rename all components
* Wire connection: PC to Wireless Router: FastEthernet0 – Ethernet1
* Wire Connection: Wireless Router to Cable Modem: Internet – Port1
* Wireless Router - Setup:

1. Change subnet mask: 255.255.255.0
2. Maximum number of users: 50
3. Static DNS 1: 208.67.220.220
4. Save Settings

* Wireless Router – Wireless:

1. SSID: HomeNetwork
2. Save Settings

* Laptop – Physical

1. Power OFF
2. Replace with WPC300N
3. Power ON

* Laptop – Desktop – PC Wireless

1. Connect Tab
2. Site Information: connect

* Laptop – Desktop – Command Prompt

1. ipconfig /all

* PC – Desktop – IP Configuration

1. Select DHCP

* PC – Desktop – Command Prompt

1. ipconfig /all

* Blue Wire Connection: Cable Modem to Cloud: Port0 – Coaxial7
* Wire Connection: Cloud to Server: Ethernet6 – FastEthernet0
* Cloud – Config

1. FastEthernet6 – Provider Network: Cable
2. Cable – Click Add

* Server – Services

1. DHCP – Pool Name: DHCPpool
2. DHCP – Default Gateway: 208.67.220.220
3. DHCP – DNS Server: 208.67.220.220
4. DHCP – Start IP Address: 208.67.220.1
5. DHCP – Subnet Mask: 255.255.255.0
6. DHCP – Max number of users: 50
7. DHCP – Service: On
8. DHCP – Click Add
9. DNS – DNS Service: ON
10. DNS – Name: Cisco.com
11. DNS – Address: 208.67.220.220
12. DNS – Click Add

* Server – Config

1. Gateway: 208.67.220.1
2. DNS Server: 208.67.220.220
3. FastEthernet0 – IP Address: 208.67.220.220
4. FastEthernet0 – Subnet Mask: 255.255.255.0

* PC – Desktop – Command Prompt

1. ipconfig /release
2. ipconfig /renew
3. ping cisco.com

* Save As

**Socket Programming**

**TCP Server**

1. **Create a socket** using socket.AF\_INET and socket.SOCK\_STREAM.
2. **Bind** the socket to a specific IP address and port.
3. **Listen** for incoming connections with listen().
4. **Accept** a connection and establish a client-server communication channel.
5. **Send/Receive** data.
6. **Close** the connection.

**Code**

import socket

# Create TCP socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Bind the socket to an IP and port

server\_socket.bind(('localhost', 12345))

# Listen for incoming connections

server\_socket.listen(1)

print("Server is listening on port 12345...")

# Accept a connection from a client

client\_socket, address = server\_socket.accept()

print(f"Connection from {address} has been established.")

# Receive data from the client

data = client\_socket.recv(1024).decode()

print(f"Received from client: {data}")

# Send data to the client

client\_socket.send("Hello from server!".encode())

# Close connections

client\_socket.close()

server\_socket.close()

**TCP Client**

1. **Create a socket** with AF\_INET and SOCK\_STREAM.
2. **Connect** to the server’s IP and port.
3. **Send/Receive** data.
4. **Close** the socket.

**Code**

import socket

# Create TCP socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Connect to the server

client\_socket.connect(('localhost', 12345))

# Send data to the server

client\_socket.send("Hello from client!".encode())

# Receive data from the server

data = client\_socket.recv(1024).decode()

print(f"Received from server: {data}")

# Close the connection

client\_socket.close()

**UDP Server**

1. **Create a socket** using AF\_INET and SOCK\_DGRAM.
2. **Bind** the socket to an IP and port.
3. **Receive** datagrams from clients.
4. **Send** responses to the clients.
5. **Close** the socket when done.

**Code**

import socket

# Create UDP socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# Bind the socket to an IP and port

server\_socket.bind(('localhost', 12345))

print("UDP Server is listening on port 12345...")

# Receive data from a client

data, client\_address = server\_socket.recvfrom(1024)

print(f"Received from client {client\_address}: {data.decode()}")

# Send a response back to the client

server\_socket.sendto("Hello from UDP server!".encode(), client\_address)

# Close the socket

server\_socket.close()

**UDP Client**

1. **Create a socket** with AF\_INET and SOCK\_DGRAM.
2. **Send** a datagram to the server.
3. **Receive** a response from the server.
4. **Close** the socket.

**Code**

import socket

# Create UDP socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# Send data to the server

client\_socket.sendto("Hello from UDP client!".encode(), ('localhost', 12345))

# Receive data from the server

data, server\_address = client\_socket.recvfrom(1024)

print(f"Received from server: {data.decode()}")

# Close the socket

client\_socket.close()

**Summary**

**1. TCP:** Create, bind, listen, accept, send/receive, close.

**2. UDP:** Create, bind, sendto/recvfrom, close.

**Network Commands**

**1. Ping**

* **Purpose**: Tests the reachability of a host on a network and measures the round-trip time.
* **Usage**: Ping a remote host (e.g., google.com).
* **Command**: ping google.com
* **Explanation**: Sends ICMP Echo Request packets to the host, and the host responds with Echo Reply packets. This helps determine whether the host is reachable and measures the time taken for the packets to travel.

**2. Ipconfig**

* **Purpose**: Displays network configuration information such as IP address, subnet mask, and default gateway for all network interfaces.
* **Usage**: Display IP configuration.
* **Command:** ipconfig
* **Explanation**: Shows the IP addresses of all active network interfaces. Useful for troubleshooting network connectivity issues.

**3. NetStat**

* **Purpose**: Displays network connections, routing tables, and network interface statistics.
* **Usage**: Show current active connections and ports in use.
* **Command**: netstat
* **Explanation**: Lists active connections, listening ports, routing tables, and other network-related statistics, which is useful for diagnosing network issues.

**4. NbtStat**

* **Purpose**: Displays protocol statistics and the current NetBIOS over TCP/IP connections.
* **Usage**: Show local NetBIOS names.
* **Command**: nbtstat -n
* **Explanation**: Displays the local NetBIOS names registered on the computer. Can help in troubleshooting NetBIOS issues in a local network.

**5. Hostname**

* **Purpose**: Displays or sets the system's hostname.
* **Usage**: Show the hostname of the system.
* **Command:** hostname
* **Explanation**: Displays the name of the computer on the network, which is useful for identifying systems in a network environment.

**6. Systeminfo**

* **Purpose**: Displays detailed configuration information about the computer, such as OS version, memory, uptime, etc.
* **Usage**: Show system information.
* **Command:** systeminfo
* **Explanation**: Provides detailed information about the operating system, hardware resources, and system uptime, useful for diagnostics.

**7. ARP (Address Resolution Protocol)**

* **Purpose**: Displays and modifies the IP-to-Physical address mapping.
* **Usage**: Show the ARP table.
* **Command**: arp -a
* **Explanation**: Shows the mapping between IP addresses and MAC addresses on the local network, useful for troubleshooting network-related issues.

**8. Nslookup**

* **Purpose**: Queries the DNS to obtain the domain name or IP address mappings.
* **Usage**: Lookup the IP address of a domain.
* **Command**: nslookup google.com
* **Explanation**: Queries the DNS server for the IP address associated with the domain google.com. This is useful for DNS troubleshooting and verification.

**9. Tracert**

* **Purpose**: Traces the route packets take to a destination host.
* **Usage**: Trace the route to a remote host.
* **Command**: tracert google.com
* **Explanation**: Displays each hop a packet takes from the source to the destination. Useful for diagnosing routing issues or network congestion.

**10. PingPath**

* **Purpose**: Similar to tracert, but provides packet loss information at each hop.
* **Usage**: Trace the route with packet loss information.
* **Command**: pingpath google.com
* **Explanation**: **Note**: pingpath might not be available on all systems. It provides similar functionality to tracert but with additional details on packet loss along the path. On Windows, tracert is more commonly used.

**11. GetMac**

* **Purpose**: Displays the MAC addresses of network interfaces.
* **Usage**: Show the MAC addresses of all network interfaces.
* **Command**: getmac
* **Explanation**: Displays the physical MAC addresses for each active network interface. Useful for identifying devices on the local network.

**12. Route**

* **Purpose**: Displays or modifies the routing table of the computer.
* **Usage**: Show the routing table.
* **Command**: route print
* **Explanation**: Shows the routing table, which lists the paths that packets take to different destinations on the network. It helps in troubleshooting routing issues.

**13. NetDiag**

* **Purpose**: Diagnoses network-related problems (this command is more commonly available on Windows systems).
* **Usage**: Run a network diagnostic.
* **Command**: netdiag
* **Explanation**: **Note**: netdiag is primarily available in Windows Server environments and performs diagnostics on the network configuration and connectivity.

**NS2**

sudo apt-get install ns2

sudo apt-get install nam

**Code:**  
set ns [new Simulator]

set nf [open s1.nam w]

$ns namtrace-all $nf

set nf1 [open s1.tr w]

$ns trace-all $nf1

proc finish {} {

global ns nf nf1

$ns flush-trace

close $nf

close $nf1

exec nam s1.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

set n9 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns duplex-link $n1 $n4 1Mb 10ms DropTail

$ns duplex-link $n3 $n4 1Mb 10ms DropTail

$ns duplex-link $n4 $n5 1Mb 10ms DropTail

$ns duplex-link $n5 $n6 1Mb 10ms DropTail

$ns duplex-link $n5 $n7 1Mb 10ms DropTail

$ns duplex-link $n6 $n8 1Mb 10ms DropTail

$ns duplex-link $n7 $n9 1Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient right

$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n1 $n4 orient right-down

$ns duplex-link-op $n3 $n4 orient right-up

$ns duplex-link-op $n4 $n5 orient right

$ns duplex-link-op $n5 $n6 orient right-up

$ns duplex-link-op $n5 $n7 orient right-down

$ns duplex-link-op $n6 $n8 orient right

$ns duplex-link-op $n7 $n9 orient right

set udp0 [new Agent/UDP]

$ns attach-agent $n2 $udp0

$udp0 set fid\_ 1  ;

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

$tcp0 set fid\_ 2  ;

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

set tcpSink0 [new Agent/TCPSink]

$ns attach-agent $n8 $tcpSink0

set null0 [new Agent/Null]

$ns attach-agent $n9 $null0

$ns connect $udp0 $null0

$ns connect $tcp0 $tcpSink0

$ns color 1 Blue

$ns color 2 Red

$ns at 0.5 "$cbr0 start"

$ns at 2.5 "$cbr0 stop"

$ns at 0.5 "$ftp0 start"

$ns at 2.5 "$ftp0 stop"

$ns at 5.0 "finish"

$ns run

