

CN Lab Final Notes

CN Lab 01

◆ Basic Definitions

| Term | Definition |
|-------------------------|--|
| Network | Group/system of interconnected devices. |
| Computer Network | Allows computers (nodes) to share data/resources using cables or wireless links. |
| Host | Any device (PC, printer, server) on a network with a unique IP. |
| Topology | Physical/logical arrangement of network devices and links. |

◆ Common Network Topologies

| Topology | Advantages | Disadvantages |
|------------------------|--|--|
| Fully Connected | High security, alternate paths, simultaneous transfers | Expensive, complex, hard to scale |
| Bus | Cheap, simple, scalable | Fails easily, poor performance, hard troubleshooting |
| Mesh | Fault tolerant, alternate paths | Costly, complex maintenance |
| Star | Easy install/maintenance, scalable | Central failure point, extra cables |
| Ring | Equal access, simple setup | One failure breaks network, limited scalability |

Applications:

- Mesh WiFi, Cloud Providers, ISPs, Corporates.
-

◆ RJ45 & Cable Types

| Type | Used For | Notes |
|------------------------|---|----------------------------|
| Straight Cable | Different devices (PC→Switch, Router→Modem) | Same color order both ends |
| Crossover Cable | Same devices (PC→PC, Switch→Switch) | TX/RX swapped |
| Auto MDI/MDI-X | Automatically adjusts; crossover not needed | |

◆ Key Network Devices

| Device | Function | Example |
|---------------|--|---------------------------|
| Hub | Broadcasts data to all; no filtering | Old small office networks |
| Switch | Sends data to specific MAC; reduces collisions | Modern LANs |
| Router | Forwards packets between networks | Home/ISP router |

Switch vs Hub: Switch = intelligent, Hub = broadcast-all.

◆ Common Terminologies

| Term | Meaning |
|--------------------|--|
| NIC | Hardware that connects host to network |
| MAC Address | Physical device address (e.g. 00:C0:9F:9B:D5:46) |
| IP Address | Logical address for host identification |
| Port | Application identifier (0–65535) |
| Gateway | Router address to reach external networks |
| DNS | Converts domain names ↔ IP addresses |
| DHCP | Automatically assigns IPs dynamically |

◆ OSI Model (7 Layers)

1. **Physical** – Bits, cables
 2. **Data Link** – MAC, switches
 3. **Network** – IP, routers
 4. **Transport** – TCP/UDP, ports
 5. **Session** – Connection control
 6. **Presentation** – Encryption, compression
 7. **Application** – HTTP, FTP, DNS
-

◆ Common Commands

| Linux | Windows | Function |
|------------|----------|-------------------------------|
| ifconfig | ipconfig | View IP configuration |
| hostname | hostname | Show system hostname |
| nslookup | nslookup | DNS info lookup |
| ping | ping | Test connectivity |
| traceroute | tracert | Trace path to destination |
| netstat | netstat | Show active connections/ports |

◆ IP Address Classes

| Class | Range | Default Mask | Network ID | Host ID | Notes |
|-------|---------------------------|---------------------|------------|---------|-----------------|
| A | 1.0.0.0 – 126.0.0.0 | 255.0.0.0 (/8) | 1st octet | Last 3 | Large networks |
| B | 128.0.0.0 – 191.255.0.0 | 255.255.0.0 (/16) | 1st 2 | Last 2 | Medium networks |
| C | 192.0.0.0 – 223.255.255.0 | 255.255.255.0 (/24) | 1st 3 | Last 1 | Small networks |
| D | 224–239 | – | – | – | Multicasting |
| E | 240–255 | – | – | – | Experimental |

Private IP 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
-

◆ Subnet Masks (Default)

| Class | Mask | Example |
|-------|---------------|-------------|
| A | 255.0.0.0 | 10.10.10.10 |
| B | 255.255.0.0 | 150.10.15.0 |
| C | 255.255.255.0 | 192.14.2.0 |

Example:

192.168.1.10 /24 → Network: 192.168.1.0, Host Range: .1–.254

◆ Duplex Communication

| Type | Description | Example |
|-------------|---------------------------|--------------------|
| Half Duplex | Two-way but one at a time | Walkie-talkie, Hub |
| Full Duplex | Simultaneous two-way | Telephone, Switch |

◆ Sample Windows Commands (Lab Tasks)

| Task | Command | Example Output |
|---------------|--------------------------------|--------------------------|
| IP Address | ipconfig | IPv4: 172.16.13.78 |
| Detailed Info | ipconfig /all | Shows DHCP, MAC, Gateway |
| Hostname | hostname | Lab2-u27 |
| Ping | ping 8.8.8.8 / ping google.com | Connectivity test |
| Open Ports | netstat -an | Lists listening ports |
| Path Trace | tracert www.google.com | Shows 11 hops |

◆ Common Troubleshooting

| Problem | Likely Cause |
|------------------------------------|--|
| Ping fails by name but works by IP | DNS resolution issue |
| No connectivity | Bad cable, disabled adapter, wrong gateway |
| DHCP fails | Server off, static IP set, range exhausted |

◆ MAC vs IP

| Aspect | MAC | IP |
|---------|----------------------|----------------------|
| Level | Data Link | Network |
| Type | Physical (permanent) | Logical (assignable) |
| Example | 00:C0:9F:9B:D5:46 | 192.168.0.1 |
| Scope | Local Network | Global Routing |

CN Lab 02 - Cisco Packet Tracer, Switch vs Hub, DHCP, and Basic LAN Setup

◆ 1. Cisco Packet Tracer Overview

Developed by: Cisco Systems (Dennis Frezzo team)

Modes:

- **Realtime Mode:** Normal operation
- **Simulation Mode:** Visualize packet movement

Protocols Supported:

| Layer | Example Protocols |
|---------|-------------------|
| 2 | Ethernet, PPP |
| 3 | IP, ARP, ICMP |
| 4 | TCP, UDP |
| Routing | RIP, OSPF |

◆ 2. Creating a Basic Topology

Step-by-Step

1. **Launch** Packet Tracer → *New File*
 2. **Add Devices:**
 - PCs, Switches, Routers, or Hubs from bottom toolbar
 3. **Connect Devices:**
 - **PC ↔ Switch:** Straight-through
 - **Switch ↔ Switch:** Crossover
 - **PC ↔ PC:** Crossover
 4. **Assign IPs:**
 - PC → Config → FastEthernet
 - Example:
 - IP: 192.168.10.2
 - Mask: 255.255.255.0
 - Gateway: 192.168.10.1
 5. **Test Connectivity:**
 - Desktop → Command Prompt → ping <other IP>
 - Or use **Add Simple PDU Tool (Ping)**
-

◆ 3. Understanding Devices (Functionality)

| Device | Function |
|----------------|---|
| PC0–PC3 | End devices, each with unique IP in 192.168.10.x subnet |
| Server0 | Centralized services; Static IP 192.168.10.100 |
| Switch0 | Connects PC0, PC1, Server0; forwards frames using MAC; uplink to Switch1 |
| Switch1 | Connects PC2 & PC3; extends LAN by linking with Switch0 (same broadcast domain) |

◆ 4. Hub vs Switch

Hub

- Works at **Layer 1 (Physical)**
- Broadcasts all traffic to every port (no filtering)
- Causes **collisions** and unnecessary traffic
- One **collision + broadcast domain** for all devices
- **Example:**
- A sends file to B via Hub
- Hub sends it to B, C, D, E
- Only B uses it; others discard it.

Switch

- Works at **Layer 2 (Data Link)**
 - Forwards frames using **MAC table** (learned dynamically)
 - Each port = **separate collision domain**
 - Reduces congestion and improves efficiency
 - **Example:**
 - A sends file to B via Switch
 - Switch checks MAC table → sends only to B
-

◆ 5. ARP (Address Resolution Protocol)

Used to map **IP** → **MAC**.

Process Example

With a Switch:

1. A wants to send file to B.
2. A knows B's IP but not MAC → broadcasts ARP: "*Who has IP B?*"
3. B replies with its MAC.
4. Switch learns B's MAC and saves it in its **MAC table**.
5. Next time, Switch forwards directly to B → Efficient delivery.

With a Hub:

1. A sends ARP request → Hub broadcasts to all.
2. B replies → Hub sends response to all ports again.
3. Every device receives unnecessary traffic → Inefficient.

✓ In short:

- ARP resolves IP–MAC.
 - Switch = direct delivery.
 - Hub = broadcasts to all.
-

◆ 6. DHCP Configuration (Dynamic Host Configuration Protocol)

Purpose: Automatically assign IPs to clients.

Setup Example

1. Add 1 Server + 3 PCs + 1 Switch
2. Server → Config → FastEthernet
3. IP: 192.168.20.1
4. Mask: 255.255.255.0
5. Server → Services → DHCP
6. Pool Name: LabPool
7. Default Gateway: 192.168.20.1
8. DNS Server: 8.8.8.8
9. Start IP: 192.168.20.10
10. End IP: 192.168.20.50
11. Each PC → Desktop → IP Configuration → Select **DHCP**
12. PC receives automatic IP → verify with ping.

Task Verification:

All PCs get IPs dynamically and can communicate with Server.

◆ 7. Bridge Operation

- Works at **Layer 2 (Data Link)**
- Connects multiple network segments (e.g., Hub0 ↔ Hub1)
- Each hub = single broadcast domain → Bridge separates them
- Learns MACs of connected devices → forwards only where needed
- **Result:** Reduced collisions and better performance

Task Example:

- 8 PCs connected to 2 Hubs via 1 Bridge
 - Bridge filters traffic between domains efficiently.
-

◆ 9. STP (Spanning Tree Protocol)

- Used in **Switches** to avoid loops.
 - Ports show **amber (listening/learning)** → then **green (forwarding)** after ~30 sec.
-

◆ 10. Simulation Mode

Use to visualize packet flow

1. Change to *Simulation Mode*
2. Select **ICMP only** filter
3. Use *Add Simple PDU Tool* → Ping PC0 → PC3
4. Press *Capture/Forward* to observe frames (e.g., ARP → Echo → Reply)

CN Lab 03 – Socket Programming (Theory-Focused Revision)

◆ What Are Sockets?

A **socket** is an endpoint of a bidirectional communication channel between two processes.

Sockets enable **inter-process communication (IPC)** either:

- On the same machine
- On a local network
- Across the internet

Sockets were introduced in **Berkeley UNIX (BSD)** and remain the foundation of network communication APIs today.

◆ Why Do We Use Sockets? (Purpose)

Sockets are used to:

- Allow two programs to communicate 
- Create client-server applications
- Send/receive messages over TCP or UDP
- Provide a standard interface to network protocols

In simple terms:

 *Sockets allow your program to talk to another program over the network.*

❖ Socket Types

| Socket Type | Description |
|-------------|-------------|
|-------------|-------------|

TCP (SOCK_STREAM) Connection-oriented, reliable, ensures ordered delivery.

UDP (SOCK_DGRAM) Connectionless, faster, no delivery guarantee.

Socket Address Families (Domains)

Domain Meaning

AF_INET IPv4 Internet Sockets

AF_INET6 IPv6 Sockets

AF_UNIX Local machine IPC

Socket Creation Syntax (Important Exam Topic)

`socket.socket(socket_family, socket_type, protocol=0)`

Parameters

| Parameter | Meaning |
|----------------------------|---|
| socket_family | e.g., AF_INET (IPv4) |
| socket_type | e.g., SOCK_STREAM (TCP), SOCK_DGRAM (UDP) |
| protocol (optional) | Usually 0 |

Example (most common)

`s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`

Important Socket Functions (Concept Only)

Server-Side Functions

1 bind()

Purpose:

Associates the socket with a specific **IP address and port number**.

Parameters:

```
s.bind((hostname, port))
```

- hostname: IP, hostname, empty string "" (means any local interface), or INADDR_ANY
- port: integer port number (e.g., 8000)

Why bind?

- Because the server must “listen” on a known port
 - Bind tells OS: “Deliver packets on this port to this program.”
-

2 listen()

Starts TCP listener, enabling the server to accept connections.

Purpose:

- Tells OS the socket is passive and waiting for incoming connections.

Parameter:

```
s.listen(backlog)
```

- backlog: maximum queued clients
-

3 accept()

Purpose:

- Waits (blocking) for client connections.
 - Returns a **new socket** for communication + client address.
-

Client-Side Function

connect()

Purpose:

- Actively initiates a TCP connection to the server's IP and port.

Syntax:

```
s.connect((server_ip, port))
```

General Socket Methods

| Method | Purpose |
|---------------|----------------|
|---------------|----------------|

| | |
|---------------|-------------------------|
| send() | Sends TCP data (bytes). |
|---------------|-------------------------|

| | |
|---------------|----------------------------|
| recv() | Receives TCP data (bytes). |
|---------------|----------------------------|

| | |
|-----------------|---------------------|
| sendto() | Sends UDP datagram. |
|-----------------|---------------------|

| | |
|-------------------|------------------------|
| recvfrom() | Receives UDP datagram. |
|-------------------|------------------------|

| | |
|----------------|--------------------|
| close() | Closes the socket. |
|----------------|--------------------|

Name & Address-Related Functions (from manual)

| Function | Purpose |
|---|--------------------------------------|
| <code>socket.gethostname()</code> | Returns the local machine name. |
| <code>socket.gethostbyname(hostname)</code> | Converts hostname → IP address. |
| <code>socket.gethostbyaddr(ip)</code> | Converts IP → hostname. |
| <code>socket.getservbyport(port, protocol)</code> | Gets service name running on a port. |

Steps in Establishing a TCP Connection (Concept Only)

(From the lab manual — very exam-relevant)

Server Side

1. Create socket
2. Bind to port
3. Listen for clients
4. Accept client connection

Client Side

1. Create socket
2. Connect to server (IP, port)

After connection → both sides can send/receive.

CN Lab 04 - HTTP/HTTPS, DNS, Routers

◆ 1. HTTP vs HTTPS

| Feature | HTTP | HTTPS |
|--------------|--------------|----------------------------------|
| URL | http:// | https:// |
| Port | 80 | 443 |
| Security | Unsecured | Secured (SSL/TLS) |
| OSI Layer | Application | Transport |
| Encryption | None | Present |
| Certificates | Not required | Required (signed or self-signed) |

Key Points:

- HTTP: Client requests → server responds → page rendered.
 - HTTPS: Secure communication; encrypts data; uses certificates.
 - Used in banking, payments, emails, corporate websites.
-

◆ 3. DNS (Domain Name System)

Definition:

Hierarchical naming system mapping human-readable domains → IP addresses. Essential for internet functionality.

Common Record Types:

| Record | Purpose |
|---------------------|--|
| A Record | Domain → IPv4 address |
| CNAME Record | Alias → canonical name (multiple systems share IP) |
| NS Record | Authoritative DNS server for a domain |
| SOA Record | Start of Authority; zone server/admin/version info |

DNS Functions:

- Resolves domain names to IPs
 - Delegates authority for subdomains
 - Provides fault tolerance
-

◆ 4. HTTP/HTTPS Lab Implementation (Packet Tracer)

HTTP Steps:

1. Assign IPs to PCs and server.
2. Enable HTTP on server (port 80).
3. PC → Web Browser → Server IP/domain → capture HTTP in simulation mode.
4. Outbound PDU shows HTTP details.

HTTPS Steps:

1. Enable HTTPS on server (port 443).
2. PC → Web Browser → Server IP → capture HTTPS packets.
3. Outbound PDU shows encrypted HTTPS details.

Tip: HTTPS encrypts data; HTTP is plain text.

◆ 6. Routers

Definition: Layer 3 devices connecting multiple networks and forwarding packets based on routing tables.

Features:

- Operates at **Network Layer**
- Contains CPU, memory, I/O interfaces, OS (Cisco IOS/DD-WRT)
- Routing table → Destination, Next hop, Interface

Basic CLI IP Assignment Example:

```
Router> enable
```

```
Router# configure terminal
```

```
Router(config)# interface fastEthernet 0/0
```

```
Router(config-if)# ip address 192.168.1.1 255.255.255.0
```

```
Router(config-if)# no shutdown
```

◆ 8. HTTP Headers & Caching

- **Header Groups:** General, Request, Response, Entity
 - **Caching Headers:**
 - Age → time since response generated
 - Expires → when resource becomes stale
-

CN Lab 05 – SMTP & FTP

1. SMTP (Simple Mail Transfer Protocol)

- **Purpose:** Sending emails between servers & clients.
- **Ports:**
 - Server-to-server: **25**
 - Client-to-server submission: **587** (465 deprecated, sometimes still used)
- **Transport:** TCP
- **Retrieval protocols:** POP3 or IMAP
- **Cisco Packet Tracer Steps:**
 1. Enable **SMTP & POP3** on Mail Server.
 2. Set domain (e.g., fast.com).
 3. Add users:

| Username | Password |
|----------|----------|
| CS | 123 |
| EE | 456 |
| BBA | 789 |

4. Configure email on PCs (Desktop → Email) → Save → Send/Receive.
 5. Use **Simulation Mode** → Filters → SMTP & POP3 → Capture/Forward to observe headers.
-

2. FTP (File Transfer Protocol)

- **Purpose:** Transfer files between client and server.
- **Port:** 21
- **Transport:** TCP
- **Modes:**
 - **Active:** Server connects to client on client's specified port (PORT M)
 - **Passive:** Client connects to server's provided port (PASV) (useful behind firewall)
- **Security:** FTPS (SSL/TLS), SFTP (SSH)
- **Packet Tracer Steps:**
 1. Enable FTP service on server → Add user accounts:

| Username | Password | Permissions |
|----------|----------|-------------------|
| Fast | 1234 | Read, Write, List |
 2. From PC command prompt:
 - Connect: ftp <server_ip> → Login
 - Upload: put test.bin
 - List files: dir
 3. Simulation Mode → Filters → FTP → Capture/Forward to view FTP headers.

DNS

- **Purpose:** Translate hostnames ↔ IP addresses
- **Tools:**
 - nslookup <hostname> → resolve IP
 - nslookup -type=NS <domain> → find authoritative DNS servers
 - ipconfig /displaydns → show cached records
 - ipconfig /flushdns → clear cache

TCP

- **Key concepts:**
 - **Three-way handshake:** SYN → SYN/ACK → ACK
 - **Sequence & Ack numbers** → reliability
 - **Congestion control:** Slow start, congestion avoidance
 - **Flow control:** Receiver-advertised window
-

Quick Commands

- **FTP on PC:** ftp <server_ip> → user <username> → put <file> → dir
- **nslookup:** nslookup www.example.com
- **ipconfig:** ipconfig /all, ipconfig /displaydns, ipconfig /flushdns

CN Lab 06(A) – Telnet & SSH

1. Telnet Configuration

Switch Configuration Example

```
Switch> enable
```

```
Switch# configure terminal
```

```
# Assign IP to VLAN1
```

```
Switch(config)# interface vlan 1
```

```
Switch(config-if)# ip address 06.66.1.1 255.0.0.0
```

```
Switch(config-if)# no shutdown
```

```
Switch(config-if)# exit
```

```
# Set hostname
```

```
Switch(config)# hostname Switch0-Telnet
```

```
# Set enable password
```

```
Switch0-Telnet(config)# enable password cisco
```

```
# Set username/password for login
```

```
Switch0-Telnet(config)# username admin password cisco
```

```
# Configure VTY lines for Telnet  
  
Switch0-Telnet(config)# line vty 0 4  
  
Switch0-Telnet(config-line)# login local  
  
Switch0-Telnet(config-line)# transport input telnet  
  
Switch0-Telnet(config-line)# exit
```

```
# Save configuration
```

```
Switch0-Telnet# write
```

PC Telnet Test

```
C:\> telnet 06.66.3.1
```

```
Username: admin
```

```
Password: cisco
```

```
Switch2-Telnet> enable
```

```
Password: cisco
```

```
Switch2-Telnet#
```

2. SSH Configuration

Router Configuration Example

```
Router> enable
```

```
Router# configure terminal
```

```
# Set hostname
```

```
Router(config)# hostname Router1-SSH
```

```
# Configure interfaces
```

```
Router1-SSH(config)# interface fastEthernet0/0
```

```
Router1-SSH(config-if)# ip address 06.66.1.1 255.255.255.0
```

```
Router1-SSH(config-if)# no shutdown
```

```
Router1-SSH(config-if)# exit
```

```
Router1-SSH(config)# interface fastEthernet0/1
```

```
Router1-SSH(config-if)# ip address 06.67.2.1 255.255.255.0
```

```
Router1-SSH(config-if)# no shutdown
```

```
Router1-SSH(config-if)# exit
```

```
# Set domain & generate RSA keys
```

```
Router1-SSH(config)# ip domain-name mynet.com
```

```
Router1-SSH(config)# crypto key generate rsa
```

```
# choose 1024 bits
```

```
# Set SSH username & secret  
Router1-SSH(config)# username admin secret admin123
```

```
# Enable SSH on VTY lines  
Router1-SSH(config)# line vty 0 4  
Router1-SSH(config-line)# login local  
Router1-SSH(config-line)# transport input ssh  
Router1-SSH(config-line)# exit
```

```
# Set SSH version  
Router1-SSH(config)# ip ssh version 2
```

```
# Set enable password  
Router1-SSH(config)# enable password 123
```

```
# Save configuration  
Router1-SSH# write
```

PC SSH Test

```
C:\> ssh -l admin 06.67.2.1
```

```
Password: admin123
```

```
Router1-SSH> enable
```

```
Password: 123
```

```
Router1-SSH#
```

3. Switch IP Update via Telnet

```
Switch2-Telnet> enable
```

```
Password: cisco
```

```
Switch2-Telnet# configure terminal
```

```
Switch2-Telnet(config)# interface vlan 1
```

```
Switch2-Telnet(config-if)# ip address 06.66.3.5 255.0.0.0
```

```
Switch2-Telnet(config-if)# exit
```

```
Switch2-Telnet(config)# write
```

4. Useful Show Commands

```
# Show running configuration
```

```
Switch# show running-config
```

```
# Show IP addresses
```

```
Switch# show ip interface brief
```

```
Router# show ip route
```

5. Key Notes

- **Telnet:** Port 23, unencrypted, login using username/password.
- **SSH:** Port 22, encrypted, login using username/password or key-based authentication.
- **VTY Lines:** Determines max simultaneous remote sessions (0-4 or 0-15).
- Always write or do wr to save configuration.
- Use no shutdown on all interfaces for connectivity.
- Ping between devices to verify IP configuration.

CN Lab 06(B) - (ACLs)

1. Standard ACLs (Source-based filtering)

- Use numbers **1–99 / 1300–1999**
- Can only match SOURCE IP
- Applied closest to **destination** (Outbound)

Basic Template

```
access-list <number> deny <source>  
access-list <number> permit <source / wildcard>  
  
interface <int>  
  
ip access-group <number> {in | out}
```

Example: Block one PC (192.168.1.10)

```
access-list 10 deny 192.168.1.10  
  
access-list 10 permit 192.168.1.0 0.0.0.255  
  
interface s2/0  
  
ip access-group 10 out
```

2. Extended ACLs (Source + Destination + Protocol + Port)

- Use numbers **100–199 / 2000–2699**
- Applied closest to **source** (Inbound)

General Syntax

```
access-list <num> <permit|deny> <protocol> <source> <wildcard> <dest> <wildcard> [eq  
<port>]
```

3. Unidirectional Blocking Between Networks (ICMP / IP)

Used for "Students cannot ping Teachers" type scenarios.

Block STUDENTS → TEACHERS (ICMP echo), allow reverse

```
access-list 120 deny icmp 192.168.1.0 0.0.0.255 any echo  
access-list 120 permit icmp any 192.168.1.0 0.0.0.255 echo-reply  
access-list 120 permit ip any any  
interface s2/0  
ip access-group 120 out
```

4. Allow Only One Host to Communicate (Selective Access)

Used for "Only CR PC allowed".

Allow one PC → Teachers, block rest

```
access-list 120 permit icmp host 192.168.1.11 192.168.2.0 0.0.0.255 echo  
access-list 120 permit icmp 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255 echo-reply  
access-list 120 deny icmp 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255 echo  
access-list 120 permit ip any any  
interface s2/0  
ip access-group 120 out
```

5. Server Access Restriction (Port-based Rules)

Used for allowing only HTTP, blocking other TCP traffic.

Example: Allow only port 80 → Server

```
access-list 100 permit tcp 192.168.1.0 0.0.0.255 host 192.168.2.6 eq 80  
access-list 100 deny  tcp 192.168.1.0 0.0.0.255 host 192.168.2.6  
access-list 100 permit ip any any  
interface f0/0  
ip access-group 100 in
```

6. Secure Remote Management (SSH allowed, FTP/Telnet blocked)

(Works only if router is in path.)

Allow SSH from 1 host, block Telnet + FTP

```
access-list 110 permit tcp host 192.168.1.8 host 192.168.2.15 eq 22  
access-list 110 deny  tcp 192.168.1.0 0.0.0.255 host 192.168.2.15 eq 21  
access-list 110 deny  tcp 192.168.1.0 0.0.0.255 host 192.168.2.15 eq 23  
access-list 110 permit ip any any  
interface s2/0  
ip access-group 110 out
```

7. Blocking All ICMP Between Networks (Two-way)

Used for full ping-block.

```
access-list 120 deny icmp 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
```

```
access-list 120 deny icmp 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
```

```
access-list 120 permit ip any any
```

```
interface s2/0
```

```
ip access-group 120 out
```

8. Department-based Restrictions (Deny single service only)

Example: Block FTP to a specific employee PC.

```
access-list 130 deny tcp 192.168.2.0 0.0.0.255 host 192.168.1.14 eq 21
```

```
access-list 130 permit ip any any
```

```
interface s2/0
```

```
ip access-group 130 in
```

9. Useful Show Commands (Very Important for Exam)

```
show access-lists
```

```
show ip interface <interface>
```

```
show run | section access-list
```

```
show ip protocols
```

CN Lab 07 – NS-3 (TCP Congestion Control & Simulation)

2. NS-3 Overview

- **Discrete-event network simulator** for networking research.
 - Supports **TCP, UDP, routing, multicast** over wired/wireless networks.
 - Scripts: **OTcl** (preferred) or **C++/Python**.
 - **Core abstractions:**
 - Node → endpoint/router
 - Application → traffic source/sink
 - Channel → connects nodes, sets bandwidth/delay
 - Net Device → interface attached to node
 - Topology Helpers → simplify network setup
 - Linux recommended; install build tools, GUI libraries, NetAnim, optional Python bindings.
 - Example execution:
 - `./waf --run scratch/file # C++`
 - `./waf --pyrun scratch/file.py # Python`
-

3. Simple Network Simulation

- **Topology:** 4 nodes (n0–n3), duplex links with bandwidth & delay.
 - **Traffic Agents:**
 - TCP + FTP (n0 → n3)
 - UDP + CBR (n1 → n3)
 - PacketSink/Null agents handle reception.
 - **Scheduling:** \$ns at <time> "<agent> start/stop"
 - **NAM Visualization:**
 - \$ns namtrace-all out.nam
 - \$ns color <fid> <color>
 - \$ns run executes simulation.
-

4. TCP Congestion Control (Lab Task Observations)

- **Purpose:** Avoid network congestion & packet loss.
- **Setup (Task 1):**
 - Sender: Socket::CreateSocket on n0
 - Receiver: PacketSinkHelper on n1
 - Payload = 1040 bytes, Packets = 1000, Rate = 1 Mbps
 - Duration: Sender 1s–20s, Sink 0s–20s
 - .pcap files for packet inspection; Gnuplot for graphs.
- **Behavior Observed:**
 - **cwnd saw-tooth pattern** → Slow Start + Multiplicative Decrease
 - Sudden drops → congestion detected (TCP Tahoe)
 - Graph: **Packet Byte Count vs Time** shows transmitted bytes and congestion events.

- **Commands/Files:**

- seventh-packet-byte-count.plt → plot congestion graph
 - Output graph: seventh-packet-byte-count.png
-

5. Key Notes

- NS-3 supports **point-to-point**, **CSMA**, and **hybrid topologies**.
- TCP congestion control has 3 phases:
 1. **Slow Start** → exponential increase
 2. **Congestion Avoidance** → additive increase
 3. **Multiplicative Decrease** → on packet loss
- OTcl scripts are lightweight; C++ gives more control.
- Visualization and Gnuplot help understand traffic dynamics and congestion patterns.

CN Lab 08 – Wireless Networks & NAT

2. Wireless Networks

Overview

- Wireless networks allow devices to stay connected without cables.
- **Access Points (APs)** transmit signals; devices need **wireless adapters**.
- WLAN connects like a wired LAN but uses **RF signals**.

Configuration Summary

- Topology: PCs connected to wireless router (WRT/Linksys).
 - DHCP IP pool: 192.168.0.100–192.168.0.150
 - Router IP: 192.168.0.1
 - Tasks performed:
 - Changed **SSID** to student-specific name.
 - Configured **static IPs** for PCs and router.
 - Secured network using **WEP key**.
 - Connected PCs using security key.
 - Device IPs after config (example):
 - PC1: 192.168.0.2, PC2: 192.168.0.4 ... Gateway: 192.168.0.1
 - Verified connectivity using **ping** between devices.
 - **Enhancements:** Added router, configured **SMTP and FTP servers** for email and file sharing within network.
-

3. NAT (Network Address Translation)

Purpose

- Maps **private IP addresses** to **public IP addresses**.
- Conserves IPv4 addresses and provides limited internal network protection.
- **Not a firewall**; used for connectivity.

Types

1. **Static NAT** – 1:1 mapping, typically for servers.
2. **Dynamic NAT** – Pool of public IPs assigned dynamically.
3. **PAT/NAT Overload** – Multiple private IPs share a single public IP via ports.

Configuration Performed

Static NAT Example (Server Access from Public Network):

```
Router(config)# ip nat inside source static 10.10.10.2 20.20.20.100
```

```
Router(config)# interface fastEthernet 0/0
```

```
Router(config-if)# ip nat inside
```

```
Router(config)# interface serial 2/0
```

```
Router(config-if)# ip nat outside
```

```
Router# show ip nat translations
```

Static NAT Example (Other Network):

```
Router(config)# ip nat inside source static 192.168.1.5 200.1.1.5
```

```
Router(config)# interface fastEthernet 0/1
```

```
Router(config-if)# ip nat inside
```

```
Router(config)# interface fastEthernet 0/0
```

```
Router(config-if)# ip nat outside
```

```
Router# show ip nat translations
```

Routing Configuration

```
Router(config)# ip route 10.10.10.0 255.255.255.0 30.30.30.2
```

```
Router(config)# ip route 20.20.20.0 255.255.255.0 30.30.30.1
```

Verification

- **Ping tests:**
 - Internal IPs: success (0% packet loss)
 - Public IP via NAT: partial success or unreachable depending on NAT config
- **Show commands:** show ip nat translations, show ip nat statistics

Note: Dynamic NAT not performed, as it was not covered yet.

4. Switch Configuration

- Enabled VLAN1 and assigned IP 10.10.1.253/24.
 - Enabled all ports: interface range fastEthernet 0/1 - 24 no shutdown.
 - Used a single domain for all users (@lan.edu) due to server limitations.
-

5. Key Notes

- Wireless networks provide **mobility, flexibility, and cost-effective deployment**.
 - NAT is essential for **IPv4 conservation** and **internal network protection**.
 - **Static NAT** is for fixed mapping; **Dynamic NAT** is pool-based; **PAT** allows multiple hosts per IP.
 - Proper **IP addressing, NAT interface config, VLANs, and security keys** are crucial for successful connectivity.
 - Use **ping and show ip nat translations** to verify NAT functionality.
-

CN Lab 09 – Final Notes

Important Commands Recap

Router Basic Commands

```
enable          // Enter privileged EXEC mode  
configure terminal // Enter global configuration mode  
exit            // Exit configuration mode  
copy running-config startup-config // Save configuration  
show ip route    // Display current routing table  
show ip interface brief // Display interfaces, IPs, and status  
show ip protocols // Show routing protocol info  
ping <IP>        // Test connectivity  
tracert <IP>      // Trace route to destination
```

Task 1: Viewing Routing Tables

- Verify connectivity and routing updates in a RIP-enabled network.
- Example outputs of show ip route:

C 10.0.0.0/30 is directly connected, Serial2/0

R 11.0.0.0/8 [120/1] via 10.0.0.2, 00:00:17, Serial2/0

C 192.168.1.0/24 is directly connected, FastEthernet0/0

R 192.168.2.0/24 [120/1] via 10.0.0.2, 00:00:17, Serial2/0

R 192.168.3.0/24 [120/2] via 10.0.0.2, 00:00:17, Serial2/0

- **Codes:**

- C – Connected
- R – RIP
- S – Static
- D – EIGRP
- O – OSPF
- L – Local

(Full code table available in manual.)

- **Gateway of last resort:** Not set in this lab scenario.
-

Task 2: Configuring RIP Version 2

1. Enable RIP v2:

```
router rip  
version 2  
no auto-summary  
network <network_address>  
exit
```

2. Example for Router 1:

```
router rip  
version 2  
no auto-summary  
network 192.168.1.0  
network 10.0.0.0  
exit
```

3. Save configuration:

```
copy running-config startup-config
```

4. **Verification:**

```
show ip route
```

```
show ip protocols
```

- Routes learned via RIP appear as R in routing table.
 - Ensure **automatic network summarization** is disabled for variable-length subnet mask (VLSM) configurations.
-

Task 3: IP Addressing and Subnetting

Interface Configuration

- Assign IP addresses to router interfaces:

```
interface FastEthernet0/0
ip address 195.168.10.1 255.255.255.128
no shutdown
```

```
interface Serial0/2/0
ip address 195.168.10.229 255.255.255.252
no shutdown
clock rate 64000
```

- Repeat for all routers with proper subnetting:
 - /25, /26, /27, /30 masks used for hosts and serial links.
- Configure RIP on all routers for all subnets:

```
router rip
version 2
no auto-summary
network 195.168.10.0
```

```
exit
• Verify interfaces:
show ip interface brief
• Verify routing table:
show ip route
```

Ping and Traceroute Testing

- **Ping** to test connectivity:

```
ping 195.168.10.130
```

```
ping 195.168.10.10
```

- **Traceroute** to check hop-by-hop path:

```
tracert 195.168.10.130
```

```
tracert 195.168.10.10
```

- Example traceroute:

```
1 0 ms 0 ms 0 ms 195.168.10.1
```

```
2 0 ms 0 ms 0 ms 195.168.10.226
```

```
3 0 ms 0 ms 0 ms 195.168.10.130
```

- Confirms routing and hop-by-hop connectivity.
-

Step-by-Step Subnetting Example: Block A (136 Hosts)

Step 1: Determine Required Host Bits (n)

- Required addresses = 136 hosts + 2 (Network & Broadcast) = 138
- Find smallest power of 2 ≥ 138 :
 - $2^7 = 128 \rightarrow$ Too small
 - $2^8 = 256 \rightarrow$ Fits requirement
- **Result:** 8 host bits needed ($n = 8$)

Step 2: Calculate Subnet Mask (CIDR Notation)

- Network bits = 32 - host bits = 32 - 8 = 24
- **CIDR notation:** /24

Step 3: Subnet Mask in Decimal

- /24 \rightarrow first 24 bits are '1's: 255.255.255.0

Step 4: Allocate IP Address Range

| Parameter | Value | Explanation |
|--------------------|---------------|-----------------------------------|
| Network Address | 192.32.16.0 | First address of subnet |
| First Host Address | 192.32.16.1 | First usable host |
| Last Host Address | 192.32.16.254 | Last usable host |
| Broadcast Address | 192.32.16.255 | Used to reach all hosts in subnet |

Step 5: Determine Next Available Subnet

- Last address of Block A: 192.32.16.255
- Next subnet (Block B) starts at **192.32.17.0**

CN Lab 10 – OSPF & BGP

2. Default and Static Routing

2.1 Default Route

- Used to send packets to a next-hop router when the destination network is not in the routing table.
- Recommended for **stub networks** (networks with only one exit path).

Command to configure default route:

```
Router(config)#ip route 0.0.0.0 0.0.0.0 <next-hop-IP>
```

2.2 Static Routing

- Used when routers are connected to multiple networks.
- Manually define the path for specific networks.

Command to configure static routes:

```
Router(config)#ip route <destination-network> <subnet-mask> <next-hop-IP>
```

Example (from lab tasks):

```
Router(config)#ip route 192.168.1.64 255.255.255.192 192.168.1.214
```

```
Router(config)#ip route 192.168.1.160 255.255.255.224 192.168.1.209
```

```
Router(config)#ip route 192.168.1.192 255.255.255.240 192.168.1.209
```

```
Router(config)#ip route 192.168.1.224 255.255.255.252 192.168.1.222
```

```
Router(config)#ip route 0.0.0.0 0.0.0.0 192.168.1.214
```

Verify routing table:

```
Router#show ip route
```

Connectivity Testing (Ping examples from lab):

```
C:\>ping 192.168.1.66
```

```
C:\>ping 192.168.1.194
```

```
C:\>ping 192.168.1.162
```

```
C:\>ping 192.168.1.226
```

```
C:\>ping 192.168.1.130
```

All pings showed **0% packet loss**, confirming correct static routing.

3. Dynamic Routing

3.1 Open Shortest Path First (OSPF)

- **Type:** Link-state routing protocol using Dijkstra algorithm.
- **Terminology:** Router ID, Neighbor, Adjacency, Hello protocol, LSA, OSPF areas.
- **Features:** Fast convergence, scalable, supports VLSM/CIDR, multi-vendor deployment.

OSPF Configuration Example (Lab Task 2):

```
Router(config)#router ospf 1
```

```
Router(config-router)#network 192.168.1.0 0.0.0.255 area 0
```

```
Router(config-router)#network 192.168.2.0 0.0.0.255 area 0
```

```
Router(config-router)#network 192.168.4.0 0.0.0.255 area 0
```

```
Router(config-router)#end
```

- **Verification:** Observe OSPF adjacency and routing table.

```
Router#show ip ospf neighbor
```

```
Router#show ip route ospf
```

3.2 Border Gateway Protocol (BGP)

- **Type:** Path-vector protocol for inter-AS routing.
- **Routing decisions:** Based on AS path, next-hop, local preference, and policies.
- **Application:** Routing between autonomous systems on the Internet.

BGP Configuration Examples (Lab Tasks 3 & 4):

Router 0

```
Router(config)#router bgp 1  
Router(config-router)#neighbor 172.16.0.2 remote-as 71  
Router(config-router)#network 10.0.0.0  
Router(config-router)#exit  
Router(config)#do wr
```

Router 1

```
Router(config)#router bgp 71  
Router(config-router)#neighbor 172.16.0.1 remote-as 1  
Router(config-router)#neighbor 172.14.0.2 remote-as 79  
Router(config-router)#exit  
Router(config)#do wr
```

Router 2

```
Router(config)#router bgp 79  
Router(config-router)#neighbor 172.14.0.1 remote-as 71  
Router(config-router)#network 40.0.0.0  
Router(config-router)#exit  
Router(config)#do wr
```

Verification of BGP:

```
Router#show ip bgp summary
```

```
Router#show ip bgp
```

- Confirms neighbor relationships, advertised networks, and path selection.
- Ping tests between networks validated BGP connectivity.

BGP Between Custom AS Numbers (Task 4 Example):

```
Router(config)#router bgp 65001
```

```
Router(config-router)#neighbor 172.16.0.2 remote-as 65002
```

```
Router(config-router)#network 10.1.0.0 mask 255.255.255.0
```

```
Router(config)#do wr
```

```
Router(config)#router bgp 65002
```

```
Router(config-router)#neighbor 172.16.0.1 remote-as 65001
```

```
Router(config-router)#network 10.2.0.0 mask 255.255.255.0
```

```
Router(config)#do wr
```

Verification:

```
Router#show ip bgp summary
```

5. Key Commands

| Task | Command |
|----------------------|---|
| Enable Router | enable |
| Enter Configuration | configure terminal |
| Interface Config | interface <type> <number> |
| Assign IP | ip address <IP> <mask> |
| Activate Interface | no shutdown |
| Static Route | ip route <dest> <mask> <next-hop> |
| Default Route | ip route 0.0.0.0 0.0.0.0 <next-hop> |
| Configure OSPF | router ospf <process-id> network <IP> <wildcard> area <area-id> |
| Configure BGP | router bgp <AS> neighbor <IP> remote-as <AS> network <IP> mask <mask> |
| Verify Routing Table | show ip route |
| Verify OSPF | show ip ospf neighbor |
| Verify BGP | show ip bgp summary show ip bgp |

CN Lab 11: VLAN (Final Notes)

1. Introduction to VLAN

- VLAN = Virtual Local Area Network, a **logical segmentation** of a LAN into separate broadcast domains.
- Devices in the same VLAN can communicate regardless of physical location.
- Reduces broadcast traffic, improves performance, enhances security, and simplifies management.
- Default VLAN on Cisco switches = **VLAN 1**.

When to use VLANs:

1. Large LANs with **>200 devices**.
 2. High broadcast traffic or congestion.
 3. Users require **security segmentation** or logical grouping for applications (e.g., VoIP phones).
-

2. VLAN Types and Connections

1. **Access Link** – Connects VLAN-unaware devices (e.g., PCs). Frames are untagged.
2. **Trunk Link** – Connects VLAN-aware devices (e.g., switches, routers). Frames are **tagged** with VLAN ID.

VLAN Communication:

- Hosts in the same VLAN communicate normally.
- Inter-VLAN communication requires a router or Layer-3 switch.

Trunking Protocols:

- **802.1Q** (standard) – uses VLAN tagging with a native VLAN for backward compatibility.
- **ISL** (Cisco proprietary) – less common, only for Cisco devices.

3. Switch VLAN Configuration (Post Lab Task Examples)

Task 1: Two VLANs (Dept10, Dept20)

```
Switch(config)# vlan 10
```

```
Switch(config-vlan)# name Dept10
```

```
Switch(config)# vlan 20
```

```
Switch(config-vlan)# name Dept20
```

```
# Assign Access Ports
```

```
Switch(config)# interface range fa0/1-2
```

```
Switch(config-if-range)# switchport mode access
```

```
Switch(config-if-range)# switchport access vlan 10
```

```
Switch(config)# interface range fa0/3-4
```

```
Switch(config-if-range)# switchport mode access
```

```
Switch(config-if-range)# switchport access vlan 20
```

```
# Configure Trunk Port
```

```
Switch(config)# interface fa0/5
```

```
Switch(config-if)# switchport mode trunk
```

```
Switch(config-if)# switchport trunk native vlan 99
```

```
Switch(config)# do write
```

Verification Commands:

```
Switch# show vlan brief
```

```
Switch# show interfaces trunk
```

Task 2: Three VLANs (HR, FINANCE, IT)

```
Switch(config)# vlan 10
```

```
Switch(config-vlan)# name HR
```

```
Switch(config)# vlan 20
```

```
Switch(config-vlan)# name FINANCE
```

```
Switch(config)# vlan 30
```

```
Switch(config-vlan)# name IT
```

```
# Assign Access Ports
```

```
Switch(config)# interface fa0/1-2
```

```
Switch(config-if-range)# switchport mode access
```

```
Switch(config-if-range)# switchport access vlan 10
```

```
Switch(config)# interface fa0/3-4
```

```
Switch(config-if-range)# switchport mode access
```

```
Switch(config-if-range)# switchport access vlan 20
```

```
Switch(config)# interface fa0/5-6
```

```
Switch(config-if-range)# switchport mode access
```

```
Switch(config-if-range)# switchport access vlan 30
```

```
# Configure Trunk Port

Switch(config)# interface fa0/24

Switch(config-if)# switchport mode trunk

Switch(config-if)# switchport trunk allowed vlan 10,20,30

Switch(config)# write memory
```

Task 3: Three VLANs (STUDENT, FACULTY, GUEST)

```
Switch(config)# vlan 10

Switch(config-vlan)# name STUDENT

Switch(config)# vlan 20

Switch(config-vlan)# name FACULTY

Switch(config)# vlan 30

Switch(config-vlan)# name GUEST
```

```
# Assign Access Ports (example)

Switch(config)# interface fa0/1-3

Switch(config-if-range)# switchport mode access

Switch(config-if-range)# switchport access vlan 10

# Configure Trunk Ports

Switch(config)# interface fa0/4

Switch(config-if)# switchport mode trunk

Switch(config-if)# switchport trunk allowed vlan 10,20,30
```

```
# Repeat for all access ports to assign VLANs 10,20,30 as needed
```

```
Switch(config)# write memory
```

Notes:

- Access ports connect PCs; trunk ports connect switches/routers.
 - Trunk ports allow multiple VLANs between switches or to the router.
-

4. Router-on-a-Stick (Inter-VLAN Routing)

- A single router interface can be divided into **subinterfaces** for each VLAN.
- Each subinterface acts as a **default gateway** for its VLAN.
- Encapsulation type = **dot1Q**.

Example: Two VLANs (Dept10 & Dept20)

```
Router(config)# interface fa0/0
```

```
Router(config-if)# no shutdown
```

```
Router(config-if)# exit
```

```
Router(config)# interface fa0/0.10
```

```
Router(config-subif)# encapsulation dot1Q 10
```

```
Router(config-subif)# ip address 192.168.10.1 255.255.255.0
```

```
Router(config)# interface fa0/0.20
```

```
Router(config-subif)# encapsulation dot1Q 20
```

```
Router(config-subif)# ip address 192.168.20.1 255.255.255.0
```

```
Router(config)# write memory
```

Example: Three VLANs (STUDENT, FACULTY, GUEST)

```
Router(config)# interface fa0/0.10
```

```
Router(config-subif)# encapsulation dot1Q 10
```

```
Router(config-subif)# ip address 192.168.10.1 255.255.255.0
```

```
Router(config)# interface fa0/0.20
```

```
Router(config-subif)# encapsulation dot1Q 20
```

```
Router(config-subif)# ip address 192.168.20.1 255.255.255.0
```

```
Router(config)# interface fa0/0.30
```

```
Router(config-subif)# encapsulation dot1Q 30
```

```
Router(config-subif)# ip address 192.168.30.1 255.255.255.0
```

Verification Commands:

```
Router# show ip interface brief
```

```
Router# show running-config
```

5. Testing VLAN Connectivity

1. Within VLAN:

- Ping between PCs in the same VLAN (should succeed).

2. Between VLANs (before inter-VLAN routing):

- Ping between PCs in different VLANs (will fail).

3. After inter-VLAN routing:

- Ping between PCs in different VLANs (will succeed via router subinterfaces).