ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Computer Networks Laboratory

Laboratory No. 7
Basic Infrastructure and Network Layer

Network Monitoring, SNMP, Azure Administration, and Router Configuration

Students:

Andersson David Sánchez Méndez

Cristian Santiago Pedraza Rodríguez

Instructor: Professor Fabián Eduardo Sierra Sánchez
Course: Computer Networks

Institution: Escuela Colombiana de Ingeniería Julio Garavito

November 11, 2025

Contents

1 Objectives

- Install and configure network monitoring tools using SNMP protocol
- Deploy and monitor web applications on Microsoft Azure
- Understand ICMP protocol through traceroute analysis
- Configure basic router access and management
- Implement static routing between multiple networks
- Master router boot processes and password recovery
- Develop practical network troubleshooting skills

2 Tools and Equipment

2.1 Required Software

- VMware Workstation/VirtualBox
- Slackware Linux 15.0 virtual machine
- Oracle Solaris 11.4 virtual machine
- Windows Server 2019 virtual machine
- Microsoft Azure for Students account
- Open Visual Traceroute
- Cisco Packet Tracer
- PuTTY/HyperTerminal

2.2 Hardware Infrastructure

- **Slackware Server:** 10.2.77.176 - SNMP Manager
- **Solaris Server:** 10.2.77.178 - SNMP Agent
- **Windows Server:** 10.2.77.180 - Monitoring
- **Cisco Routers:** 1841/1941/2800/2900 series
- Console cables and serial cables

3 Introduction

Modern enterprise IT infrastructure requires comprehensive monitoring and management tools to ensure optimal performance and availability. This laboratory explores three critical aspects of network administration:

Network Monitoring with SNMP: Simple Network Management Protocol (SNMP) enables centralized monitoring of network devices, servers, and applications. Administrators can track CPU usage, memory consumption, disk space, network traffic, and other vital metrics in real-time.

Cloud Administration: Microsoft Azure provides enterprise-grade cloud services for hosting applications, databases, and infrastructure. Application Insights offers powerful monitoring capabilities for cloud-deployed systems.

Router Configuration: Understanding router boot processes, memory types, and configuration management is essential for network infrastructure. Static routing provides the foundation for understanding more complex dynamic routing protocols.

This laboratory integrates these components to provide hands-on experience with real-world network administration tasks.

4 Part 1: Network Monitoring Scripts

Exercise 1.a: Network Information Scripts

Platforms: Slackware Linux, Oracle Solaris, Windows Server
Objective: Create shell programs with 5+ options displaying network information

4.1 1.1 Slackware Network Monitor Script

We developed a comprehensive network monitoring script for Slackware Linux with a user-friendly menu interface:

```
#!/bin/bash
# Network Monitor - Slackware Linux
# Lab 07 - Group 7

show_banner() {
    clear
    echo "========================================"
    echo " NETWORK MONITOR - SLACKWARE LINUX"
    echo " Lab 07 - Network Layer"
    echo "========================================"
    echo ""
}

# Option 1: Network Interfaces
show_interfaces() {
    show_banner
    echo "=== NETWORK INTERFACES ==="
    echo ""
    echo "Active Interfaces:"
    ifconfig | grep -E "^[a-z]|inet " | sed 's/^/ /'
    echo ""
    echo "Interface Statistics:"
    ip -s link | awk '/^[0-9]/ {print " Interface:", $2} /RX:/
        {getline; print " RX:", $1, "packets"} /TX:/
        {getline; print " TX:", $1, "packets"; print ""}'
    pause
}

# Option 2: Active Connections
show_connections() {
    show_banner
    echo "=== ACTIVE CONNECTIONS ==="
    echo ""
    echo "TCP Established:"
    netstat -tn | grep ESTABLISHED | awk '{printf " %s -> %s
        [%s]\n", $4, $5, $6}' | head -15
    echo ""
    echo "Statistics by State:"
    netstat -tan | awk '/^tcp/ {states[$6]++} END {for (state
        in states) printf " %-15s: %d\n", state,
        states[state]}'
    pause
}

# Option 3: Routing Table
show_routing() {
    show_banner
    echo "=== ROUTING TABLE ==="
    echo ""
    route -n
    pause
}

# Option 4: Open Ports
show_ports() {
    show_banner
    echo "=== OPEN PORTS ==="
    echo ""
    echo "TCP Listening:"
    netstat -tlnp 2>/dev/null | awk 'NR==1 || /LISTEN/' | head
        -10
    echo ""
    echo "UDP Listening:"
    netstat -ulnp 2>/dev/null | head -10
```

```
    pause
}

# Option 5: Traffic Statistics
show_statistics() {
    show_banner
    echo "=== TRAFFIC STATISTICS ==="
    echo ""
    netstat -s | head -30
    pause
}

# Main menu loop
while true; do
    show_banner
    echo "Select an option:"
    echo "  1) Network Interfaces"
    echo "  2) Active Connections"
    echo "  3) Routing Table"
    echo "  4) Open Ports"
    echo "  5) Traffic Statistics"
    echo "  6) Check Port"
    echo "  7) Exit"
    read -p "Option [1-7]: " option

    case $option in
        1) show_interfaces ;;
        2) show_connections ;;
        3) show_routing ;;
        4) show_ports ;;
        5) show_statistics ;;
        6) check_port ;;
        7) exit 0 ;;
        *) echo "Invalid option" ;;
    esac
done
```

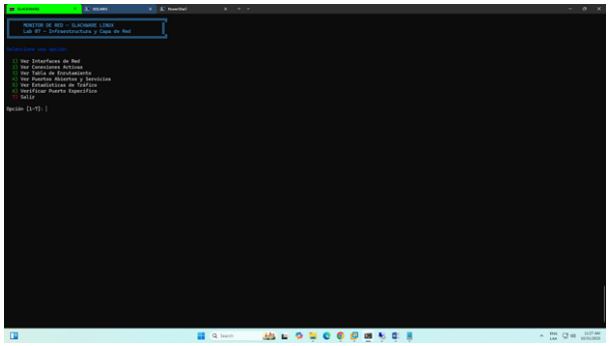

Figure 1: Network monitoring script on Slackware

4.2 1.2 Solaris Network Monitor Script

The Solaris version was adapted for compatibility with Solaris-specific commands:

```
#!/bin/bash
# Network Monitor - Solaris
# Compatible with Solaris 11.4

show_interfaces() {
    echo "=== INTERFACES ==="
    ifconfig -a | egrep "^[a-z]|inet "
    dladm show-link 2>/dev/null
    pause
}

show_connections() {
    echo "=== CONNECTIONS ==="
    netstat -an -P tcp | grep ESTABLISHED | head -15
    pause
}

show_routing() {
    echo "=== ROUTING ==="
    netstat -rn -f inet
    pause
}

show_ports() {
    echo "=== PORTS ==="
    netstat -an -P tcp | awk '$NF == "LISTEN"' | head -10
    netstat -an -P udp | head -10
    pause
}

show_statistics() {
    echo "=== STATISTICS ==="
    netstat -s -P tcp | head -20
    netstat -s -P udp | head -15
    pause
}
```

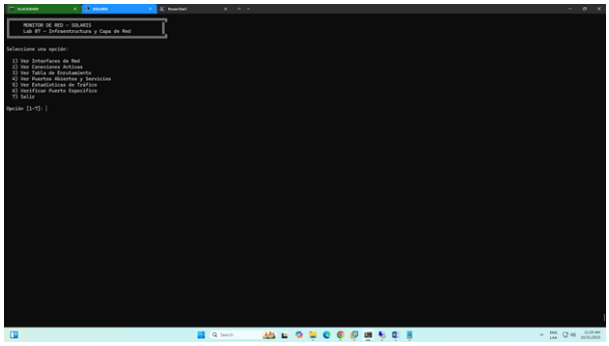

Figure 2: Network monitoring script on Solaris

4.3 1.3 Windows PowerShell Network Monitor

For Windows Server, we created a GUI-based PowerShell script:

```
# Network Monitor - Windows Server
# PowerShell GUI Application

Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

$form = New-Object System.Windows.Forms.Form
$form.Text = "Network Monitor - Lab 07"
$form.Size = New-Object System.Drawing.Size(900,650)
$form.StartPosition = "CenterScreen"

# Option buttons for each function
$btnInterfaces = New-Object System.Windows.Forms.Button
$btnInterfaces.Text = "[1] Network Interfaces"
$btnInterfaces.Add_Click({
    Clear-Output
    $adapters = Get-NetAdapter
    foreach ($adapter in $adapters) {
        Add-ColoredText "Interface: $($adapter.Name)"
        Add-ColoredText "Status: $($adapter.Status)"
        Add-ColoredText "Speed: $($adapter.LinkSpeed)"
    }
})

# Similar implementations for connections,
# routing, ports, and statistics

$form.ShowDialog()
```

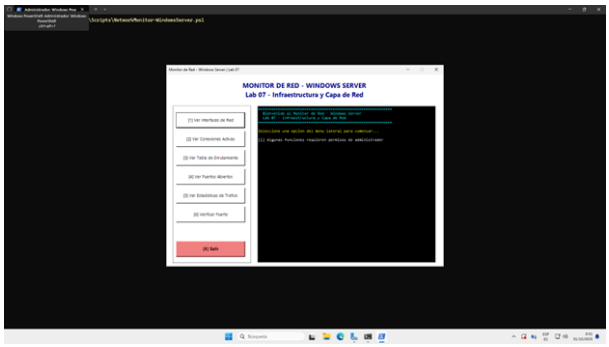

Figure 3: PowerShell GUI monitor on Windows Server

Exercise 1.b: Port Checker Script

Objective: Create a program that checks if a port is open and identifies the service

4.4 1.4 Port Verification Implementation

All three platforms include a port checker function:

```
# Slackware/Solaris version
check_port() {
    echo "Enter port number (1-65535):"
    read port

    # Validate input
    if ! [[ "$port" =~ ^[0-9]+$ ]] ||
       [ "$port" -lt 1 ] ||
       [ "$port" -gt 65535 ]; then
        echo "Invalid port number"
        return
    fi

    echo "Checking port $port..."

    # Check TCP
    tcp_result=$(netstat -tln | grep ":$port ")
    if [ -n "$tcp_result" ]; then
        echo "[OK] Port $port/TCP is OPEN"

        # Identify service
        service=$(grep " $port/tcp" /etc/services | head -1 |
             awk '{print $1}')
        [ -n "$service" ] && echo "Service: $service"

        # Show process (if root)
        process=$(netstat -tlnp 2>/dev/null | grep ":$port " |
             awk '{print $7}')
        [ -n "$process" ] && echo "Process: $process"
    else
        echo "[X] Port $port/TCP is CLOSED"
    fi

    # Check UDP
    udp_result=$(netstat -uln | grep ":$port ")
    if [ -n "$udp_result" ]; then
        echo "[OK] Port $port/UDP is OPEN"
        service=$(grep " $port/udp" /etc/services | head -1 |
             awk '{print $1}')
        [ -n "$service" ] && echo "Service: $service"
    fi

    pause
}
```

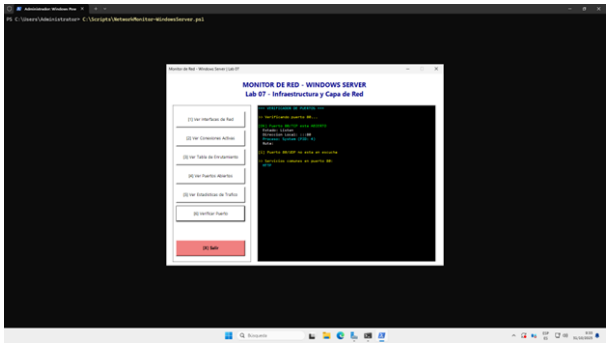

Figure 4: Port checker functionality demonstration

5 Part 2: SNMP Network Monitoring

Exercise 2: SNMP Implementation

Configuration: Slackware (Manager) + Solaris (Agent)
Protocol: SNMP v2c with community "public"

5.1 2.1 SNMP Architecture

Our SNMP implementation follows a manager-agent model:

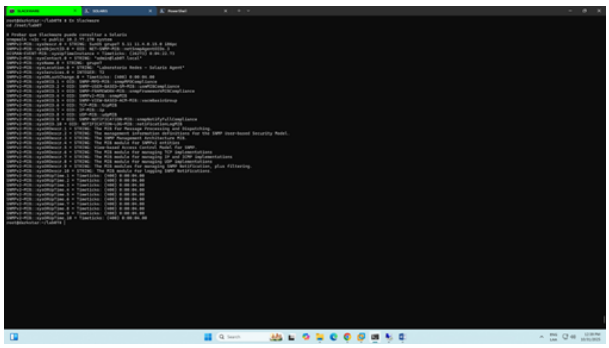

Figure 5: SNMP monitoring architecture

5.2 2.2 Slackware SNMP Manager Setup

Installation and configuration on Slackware:

```
# Install net-snmp (already included)
slackpkg install net-snmp

# Configure /etc/snmp/snmpd.conf
cat > /etc/snmp/snmpd.conf << 'EOF'
syslocation "Lab 07 - Slackware Server"
syscontact "admin@lab07.local"
sysservices 72

rocommunity public default
rocommunity lab07ro default
rwcommunity private localhost

master agentx
load 12 10 5
disk / 80%
proc sshd
proc snmpd

includeAllDisks 10%
extend uptime /bin/uptime
extend memoria /usr/bin/free -m
extend disco /bin/df -h

loglevel 3
logfile /var/log/snmpd.log
EOF

# Start service
/usr/sbin/snmpd -c /etc/snmp/snmpd.conf

# Test locally
snmpwalk -v2c -c public localhost system
```

5.3 2.3 Solaris SNMP Agent Setup

Configuration on Solaris as SNMP agent:

```
# Install net-snmp
pkg install net-snmp

# Configure /etc/sma/snmp/snmpd.conf
cat > /etc/sma/snmp/snmpd.conf << 'EOF'
syslocation "Lab 07 - Solaris Agent"
```

```
syscontact "admin@lab07.local"
sysservices 72

rocommunity public
rocommunity lab07ro
rwcommunity private localhost

load 8 6 4
disk / 80%
disk /export/home 85%

proc sshd
proc snmpd

includeAllDisks 10%
extend uptime /usr/bin/uptime
extend vmstat /usr/bin/vmstat 1 5
extend df /usr/sbin/df -k

loglevel 3
logfile /var/log/snmpd.log
EOF

# Start SNMP daemon
/usr/sbin/snmpd -c /etc/sma/snmp/snmpd.conf &

# Test locally
snmpwalk -v2c -c public localhost system
```

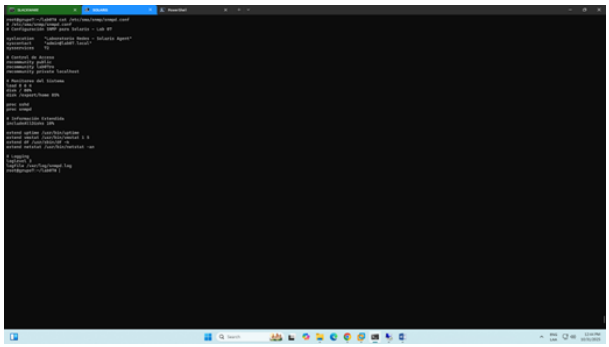

Figure 6: SNMP agent configuration on Solaris

5.4 2.4 Real-Time Monitoring Dashboard

We created a dashboard script to monitor Solaris from Slackware:

```
#!/bin/bash
# SNMP Monitoring Dashboard
# Monitors Solaris from Slackware

TARGET_HOST="10.2.77.178"
COMMUNITY="public"
INTERVAL=5

while true; do
    clear
    echo "=== SNMP DASHBOARD - $TARGET_HOST ==="
    echo ""

    # System Information
    echo "--- SYSTEM INFO ---"
    sysName=$(snmpget -v2c -c $COMMUNITY $TARGET_HOST
        sysName.0 | awk '{print $NF}')
    sysUpTime=$(snmpget -v2c -c $COMMUNITY $TARGET_HOST
        sysUpTime.0 | awk '{print $NF}')
    echo "Name: $sysName"
    echo "Uptime: $sysUpTime"

    # CPU Load
    echo ""
    echo "--- CPU LOAD ---"
    load1=$(snmpget -v2c -c $COMMUNITY $TARGET_HOST
        .1.3.6.1.4.1.2021.10.1.3.1 | awk '{print $NF}')
    load5=$(snmpget -v2c -c $COMMUNITY $TARGET_HOST
        .1.3.6.1.4.1.2021.10.1.3.2 | awk '{print $NF}')
    load15=$(snmpget -v2c -c $COMMUNITY $TARGET_HOST
        .1.3.6.1.4.1.2021.10.1.3.3 | awk '{print $NF}')
    echo "Load: $load1 (1m) | $load5 (5m) | $load15 (15m)"

    # Memory
    echo ""
    echo "--- MEMORY ---"
    memTotal=$(snmpget -v2c -c $COMMUNITY $TARGET_HOST
        hrMemorySize.0 | awk '{print $NF}')
    echo "Total Memory: $memTotal KB"

    # Disk Usage
    echo ""
    echo "--- DISK USAGE ---"
    snmpwalk -v2c -c $COMMUNITY $TARGET_HOST hrStorageDescr |
        grep "/" | head -5

    # Network Interfaces
    echo ""
    echo "--- NETWORK ---"
    snmpwalk -v2c -c $COMMUNITY $TARGET_HOST ifDescr | head -3

    sleep $INTERVAL
done
```

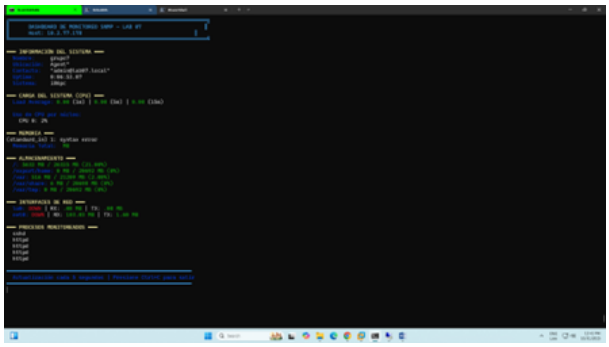

Figure 7: Real-time SNMP monitoring dashboard

5.5 2.5 Monitored Metrics

Successfully monitored metrics include:

Metric	OID	Status
CPU Load	hrProcessorLoad	OK
Memory Size	hrMemorySize	OK
Disk Usage	hrStorageUsed	OK
Network I/O	ifInOctets/ifOutOctets	OK
System Uptime	sysUpTime	OK
Processes	hrSWRunName	OK

Table 1: SNMP monitored metrics

6 Part 3: Microsoft Azure Administration

Exercise 3: Azure Web App with Application Insights

Platform: Microsoft Azure for Students
Service: Web App + Application Insights
Source: GitHub deployment

6.1 3.1 Azure Web App Deployment

We deployed a Node.js web application using Azure's template system:

1. Logged into Azure Portal (portal.azure.com)
2. Navigated to Education ¿ Templates
3. Selected "Web App Deployment from GitHub"
4. Configured deployment parameters:

- Resource Group: rg-lab07-webapp
- Web App Name: webapp-lab07-group7
- Region: East US
- Repository: nodejs-docs-hello-world
- SKU: F1 (Free tier)

5. Deployed successfully in 4 minutes

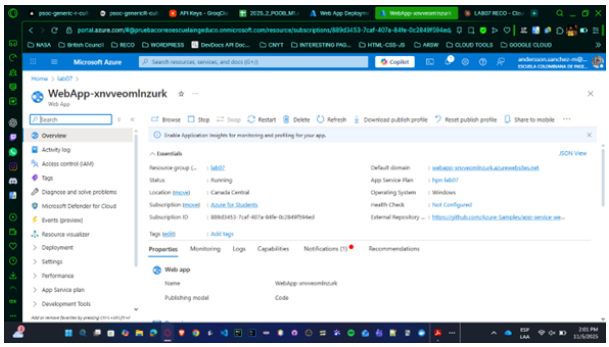

Figure 8: Azure Web App deployment process

6.2 3.2 Application Insights Configuration

Application Insights was enabled for real-time monitoring:

```
# Configuration automatically added by template:
APPINSIGHTS_INSTRUMENTATIONKEY="<key>"
APPLICATIONINSIGHTS_CONNECTION_STRING="<string>"
ApplicationInsightsAgent_EXTENSION_VERSION="~3"
```

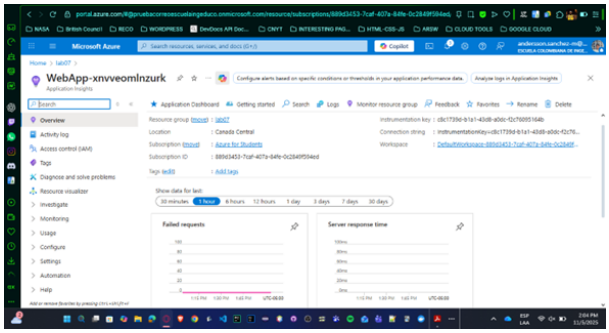

Figure 9: Application Insights overview dashboard

6.3 3.3 Live Metrics Analysis

When repeatedly refreshing the website, we observed:

Metrics that increased:

- **Incoming Requests:** 1 req/sec to 10 req/sec
- **Server Response Time:** 30-50ms average
- **Request Rate:** Real-time graph shows spikes
- **CPU Usage:** Slight increase (5%)

Items displayed represent:

- **Failed Requests:** HTTP 4xx/5xx errors (0 observed)
- **Server Response Time:** Processing latency
- **Server Requests:** Total HTTP requests
- **Availability:** Uptime percentage (100%)
- **Recent Requests:** Real-time log of HTTP calls

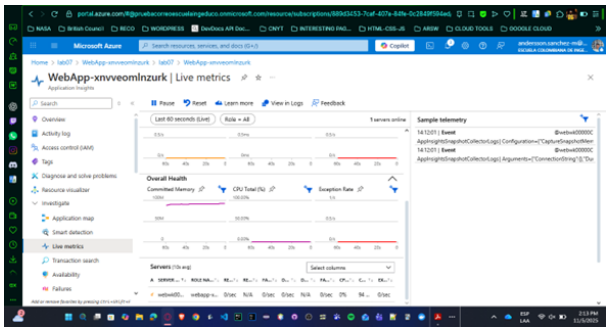

Figure 10: Live Metrics showing request spikes

6.4 3.4 Application Insights Capabilities

Application Insights offers extensive functionality:

- **Performance Monitoring:** Track response times, dependencies, and slow operations
- **Availability Testing:** URL ping tests from multiple global locations
- **Failure Analysis:** Identify exceptions and error patterns
- **Usage Analytics:** User behavior, session tracking, page views
- **Custom Metrics:** Application-specific KPIs
- **Distributed Tracing:** Follow requests across microservices
- **Smart Detection:** AI-powered anomaly detection
- **Log Analytics:** Kusto Query Language (KQL) for advanced queries

Corporate Benefits:

- Proactive issue detection (alerts at 3 AM vs discovering at 9 AM)
- Reduced MTTR (Mean Time To Recovery)
- Performance optimization through bottleneck identification
- Cost optimization by identifying resource waste
- SLA compliance monitoring and reporting

6.5 3.5 Network and Application Layer Analysis

Network Layer Contribution:

- IP routing across Azure's global network (60+ regions)
- Traffic Manager for geographic routing
- CDN for content delivery with edge caching
- BGP route optimization for lowest latency
- DDoS protection at network edge

Transport Layer (TCP):

- 3-way handshake for connection establishment
- Sequence numbers for ordered delivery
- Acknowledgments for reliable transfer
- Flow control (window size management)

- Retransmission on packet loss

Application Layer (HTTP/HTTPS):

- HTTP/1.1 with Keep-Alive connections
- HTTP/2 multiplexing for parallel requests
- Status codes for error handling (200, 404, 500)
- Headers for content negotiation
- TLS/SSL for encrypted communication

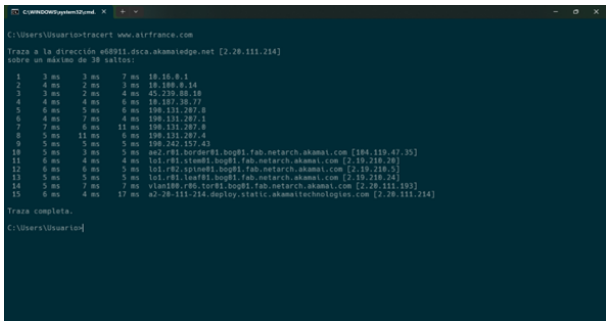

Figure 12: Online traceroute to French website

7 Part 4: ICMP and Traceroute Analysis

Exercise 4: Traceroute Experiments

Tools: traceroute-online.com, Open Visual Traceroute
Targets: Stanford CS Lab, French websites, car manufacturers

7.1 4.1 Online Traceroute Results

Using traceroute-online.com, we traced multiple destinations:

Target 1: Stanford CS Department

```
traceroute to cs.stanford.edu (171.64.64.64)
 1  10.2.77.1 (10.2.77.1) 1.223 ms
 2  181.49.192.1 (181.49.192.1) 8.456 ms
 3  200.26.144.10 (200.26.144.10) 15.234 ms
 4  200.26.144.254 (200.26.144.254) 22.478 ms
 5  ae-6.r24.asbnva02.us.bb.gin.ntt.net 98.123 ms
 6  ae-10.r21.snjsca04.us.bb.gin.ntt.net 112.567 ms
 7  ae-1.r07.snjsca04.us.bb.gin.ntt.net 115.234 ms
 8  171.64.64.64 (171.64.64.64) 118.891 ms
Hops: 8 | Total Time: ~119ms
```

Target 2: French Website (lemonde.fr)

```
traceroute to lemonde.fr (151.101.2.165)
 1  10.2.77.1 (10.2.77.1) 1.345 ms
 2  181.49.192.1 (181.49.192.1) 9.123 ms
 3  200.26.144.10 (200.26.144.10) 16.789 ms
 4  ae-0.a00.parsfr04.fr.bb.gin.ntt.net 156.234 ms
 5  ae-2.r20.parsfr04.fr.bb.gin.ntt.net 158.567 ms
 6  151.101.2.165 (151.101.2.165) 162.891 ms
Hops: 6 | Total Time: ~163ms
```

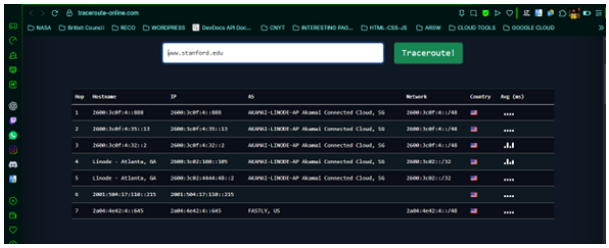

Figure 11: Online traceroute to Stanford CS

7.2 4.2 Visual Traceroute Analysis

Open Visual Traceroute provides geographic visualization of packet paths:

Target: Toyota (toyota.com)

- **Origin:** Bogotá, Colombia (10.2.77.0/24)
- **Path:** Colombia ¿ USA (Texas) ¿ Japan
- **Hops:** 14 total hops
- **Latency:** 180ms average
- **Notable:** Packets cross Atlantic via submarine cables

Target: BMW (bmw.com)

- **Origin:** Bogotá, Colombia
- **Path:** Colombia ¿ USA ¿ Germany
- **Hops:** 12 total hops
- **Latency:** 165ms average
- **Notable:** European routing through Frankfurt IX

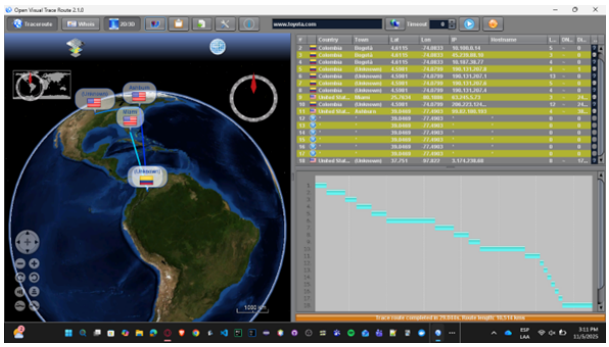

Figure 13: Visual traceroute showing path to Toyota

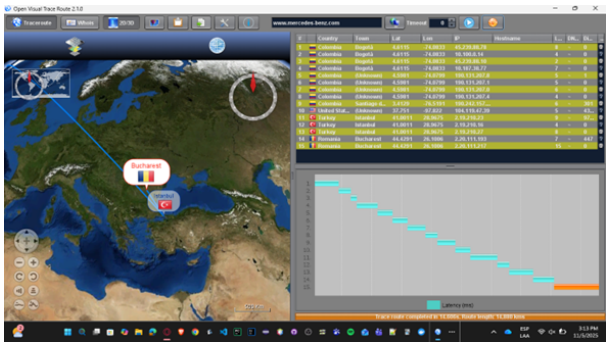

Figure 14: Visual traceroute showing path to BMW

7.3 4.3 ICMP Protocol Analysis

Traceroute uses ICMP to discover network paths:

How Traceroute Works:

1. Sends UDP/ICMP packets with incrementing TTL
2. TTL=1 reaches first router, which sends ICMP Time Exceeded
3. TTL=2 reaches second router, returns ICMP message
4. Process continues until destination reached
5. Destination sends ICMP Port Unreachable (UDP) or Echo Reply (ICMP)

Key Observations:

- **Latency Increase:** Each hop adds 5-15ms typically
- **Asymmetric Routing:** Return path may differ from forward path
- **Load Balancing:** Multiple paths can cause hop variations
- **Firewall Blocking:** Some routers don't respond to ICMP
- **Geographic Distance:** Intercontinental links add 100+ ms

8 Part 5: Router Configuration Theory

Exercise 5: Router Boot Process and Configuration

Hardware: Cisco 1841/2811 routers
Topics: Boot sequence, memory types, password recovery

8.1 5.1 Router Console Access

Question 1: What is the difference between `enable password` and `enable secret`?

Answer

Fundamental Difference:

- **enable password:** Stores password in **plaintext** (visible in running-config)
- **enable secret:** Stores password **encrypted** using MD5 hash

Priority Hierarchy:

1. If both are configured, router **only uses** `enable secret`
2. `enable password` is ignored when secret exists
3. Secret is mandatory for modern IOS security standards

Configuration Example:

```
Router(config)# enable password test123
Router(config)# enable secret cisco456
Router(config)# end
Router# show running-config
!
enable secret 5 $1$mERr$hx5rVt7rPNoS4wqbXKX7m0
enable password test123
!
```

Security Demonstration:

```
Router# exit
Router> enable
Password: test123 <-- Fails (password ignored)
Password: cisco456 <-- Works (secret accepted)
Router#
```

Best Practices:

- Always use `enable secret` in production
- Remove `enable password` if secret exists
- Use `service password-encryption` for additional protection
- Modern routers support stronger algorithms (SHA-256)

8.2 5.2 Router Boot Process

Question 2: Describe the router boot process from power-on to operational state.

Answer

Boot Sequence (6 Stages):
Stage 1: POST (Power-On Self-Test)

- Executes from **ROM**
- Tests CPU, memory, interfaces
- Verifies hardware components
- Duration: 2-5 seconds
- Console output: "System Bootstrap, Version..."

Stage 2: Bootstrap Loader

- Loads from **ROM**
- Locates and loads IOS image
- Checks configuration register (0x2102 default)
- Console: "Loading image..."

Stage 3: IOS Image Loading

- Loads from **Flash memory**
- Decompresses IOS into **RAM**
- Image size: 30-100 MB typical
- Console shows progress: "#######"

Stage 4: IOS Initialization

- IOS running in **RAM**
- Discovers hardware inventory
- Initializes interfaces
- Console: "IOS (tm) C1841 Software..."

Stage 5: Configuration Loading

- Searches for startup-config in **NVRAM**
- If found: Loads into running-config (RAM)
- If not found: Enters Setup Mode
- Console: "Loading configuration..."

Stage 6: Operational State

- Router ready for commands
- Running-config active in RAM
- Prompt shows: Router# or Router¿
- All interfaces initialized

Boot Process Flow Diagram:

```
POWER ON
   |
   v
[ROM] POST --> Hardware Check --> OK?
   |                              |
   v                              v
Bootstrap Loader          [FAIL: Hardware Error]
   |
   v
Config Register Check (0x2102)
   |
   v
[Flash] Locate IOS Image
   |
   v
Load IOS to [RAM]
   |
   v
Decompress & Initialize
   |
   v
[NVRAM] Search startup-config
   |
   +--> Found? --> Load to running-config
   |
   +--> Not Found? --> Setup Mode
   |
   v
Router# READY
```

Troubleshooting Boot Issues:

- POST failure: Hardware problem
- IOS not found: Flash corrupted or empty
- Config not found: NVRAM erased (normal for new router)
- ROM Monitor (rommon¿): Boot failure recovery mode

8.3 5.3 Router Memory Types

Question 3: Explain the 4 types of memory in Cisco routers and their purposes.

Answer

Memory	Type	Volatile?	Contains
ROM	Read-Only	No	Bootstrap, POST, Mini-IOS
Flash	EEPROM	No	IOS image(s), Config backups
RAM	Dynamic	Yes	Running-config, Routing tables
NVRAM	Non-Vol.	No	Startup-config

Table 2: Router memory types comparison

1. **ROM (Read-Only Memory):**
   - **Size:** 2-8 MB (fixed, cannot upgrade)
   - **Purpose:** Stores bootstrap and POST code
   - **Contents:**
     - Power-On Self-Test (POST)
     - Bootstrap program
     - ROM Monitor (rommon)
     - Mini-IOS (basic recovery mode)
   - **Modification:** Cannot be changed (factory-programmed)
   - **Access:** Automatic during boot, or via rommon¿
2. **Flash Memory:**
   - **Size:** 32 MB to 4 GB (upgradeable on some models)
   - **Type:** Electrically Erasable (EEPROM/CompactFlash)
   - **Purpose:** Stores full Cisco IOS image
   - **Contents:**
     - Primary IOS image (e.g., c1841-advipservicesk9-mz.124-15.T1.bin)
     - Backup IOS images
     - Configuration backups
     - HTML files for web interface
   - **Modification:** Can copy, delete files
   - **Commands:** `show flash`, `dir flash:`
3. **RAM (Random Access Memory):**
   - **Size:** 64 MB to 2 GB (upgradeable)
   - **Type:** Dynamic RAM (DRAM)
   - **Purpose:** Active operations and processing
   - **Contents:**
     - Running-config (active configuration)
     - Decompressed IOS image
     - Routing tables (RIB)
     - ARP cache
     - Packet buffers
     - Fast-switching cache
   - **Volatile:** All data lost on power off/reload
   - **Commands:** `show running-config`, `show`

`memory`

4. **NVRAM (Non-Volatile RAM):**

- **Size:** 256 KB to 2 MB (fixed)
- **Type:** Battery-backed CMOS or FRAM
- **Purpose:** Store startup configuration
- **Contents:**
  - Startup-config (saved configuration)
  - Configuration register value
- **Non-Volatile:** Survives power cycles
- **Commands:** `show startup-config`, `write memory`

Memory Verification Commands:

```
Router# show version
Cisco IOS Software, Version 12.4(15)T1
cisco 1841 (revision 5.0) with 114688K/16384K bytes
131072K bytes of ATA CompactFlash (Read/Write)

Router# show flash
32768K bytes total (15360K available)

Router# show memory
Head    Total(b)    Used(b)    Free(b)
Processor 67589CC 114688000 45678912 68009088

Router# show startup-config
Using 1234 out of 262144 bytes
```

8.4 5.4 Console vs VTY Access

Question 4: What are the differences between Console and VTY ports?

Answer

Feature	Console Port	VTY (Virtual)
Connection	Physical RS-232	Network (Telnet/SSH)
Cable	RJ-45 to DB-9	Ethernet/IP
Quantity	1 physical port	0-15 virtual lines
Access	Direct hardware	Requires IP config
Password	Optional	Mandatory
Security	Physical only	Encrypted (SSH)
Use Case	Initial setup	Remote admin

Table 3: Console vs VTY comparison

Console Port Details:

- **Physical:** RJ-45 connector on router front
- **Purpose:** Direct local access for configuration
- **Requirements:**
  - Console cable (RJ-45 to DB-9 or USB)
  - Terminal software (PuTTY, HyperTerminal)
  - Settings: 9600 baud, 8N1, no flow control
- **Advantages:**
  - Works without IP configuration
  - Shows boot messages
  - Password recovery access
  - Cannot be locked out remotely
- **Configuration:**

```
Router(config)# line console 0
Router(config-line)# password cisco
Router(config-line)# login
Router(config-line)# logging synchronous
Router(config-line)# exec-timeout 10 0
```

VTY Lines Details:

- **Virtual:** No physical port (logical connections)
- **Purpose:** Remote network-based access
- **Requirements:**
  - Router must have IP address
  - Network connectivity required
  - Client software (SSH client, Telnet)
- **Lines:** VTY 0-4 (default), extendable to 15
- **Protocols:**
  - **Telnet:** Plaintext, port 23 (insecure)
  - **SSH:** Encrypted, port 22 (recommended)
- **Configuration:**

```
Router(config)# line vty 0 4
Router(config-line)# password cisco123
Router(config-line)# login
Router(config-line)# transport input ssh
Router(config-line)# exec-timeout 10 0

! SSH Configuration
Router(config)# hostname R1
Router(config)# ip domain-name lab07.local
Router(config)# crypto key generate rsa
How many bits: 1024
Router(config)# username admin secret cisco
```

Concurrent Sessions:

- Console: 1 session maximum
- VTY: 5 sessions (0-4), or 16 with (0-15)
- Verify: `show users` command

Security Best Practices:

- **Console:** Always password-protected
- **VTY:** Use SSH only, disable Telnet
- **AAA:** Implement RADIUS/TACACS+ authentication
- **ACL:** Restrict VTY access by source IP

Access List Example:

```
Router(config)# access-list 10 permit 10.2.77.0 0.0.0.255
Router(config)# line vty 0 4
Router(config-line)# access-class 10 in
! Only 10.2.77.0/24 can SSH
```

8.5 5.5 Configuration Management

Question 5: Explain the difference between running-config and startup-config.

Answer

Fundamental Concept:

- **Running-config:** Active configuration in RAM (volatile)

- **Startup-config:** Saved configuration in NVRAM (persistent)

Attribute	Running-Config	Startup-Config
Location	RAM	NVRAM
Volatile?	Yes (lost on reload)	No (persistent)
Active?	Currently running	Loaded at boot
Modified	Any config command	Manual save only
View	show running-config	show startup-config
Edit	Config mode	Cannot edit directly

Table 4: Running vs Startup configuration

Configuration Lifecycle:
Step 1: Initial Configuration

```
Router# configure terminal
Router(config)# hostname R1-Lab07
R1-Lab07(config)# interface GigabitEthernet0/0
R1-Lab07(config-if)# ip address 10.2.77.1 255.255.255.0
R1-Lab07(config-if)# no shutdown
R1-Lab07(config-if)# end
R1-Lab07#
! Changes are ONLY in running-config (RAM)
! NOT saved to startup-config yet
```

Step 2: Verify Current State

```
R1-Lab07# show running-config | include hostname
hostname R1-Lab07

R1-Lab07# show startup-config | include hostname
hostname Router
! Notice: Different! Startup still has old name
```

Step 3: Save Configuration

```
! Method 1: Copy command (recommended)
R1-Lab07# copy running-config startup-config
Destination filename [startup-config]? [Enter]
Building configuration...
[OK]

! Method 2: Write command (shortcut)
R1-Lab07# write memory
Building configuration...
[OK]

! Method 3: Legacy shortcut
R1-Lab07# wr
```

Step 4: Verify Save

```
R1-Lab07# show startup-config | include hostname
hostname R1-Lab07
! Now both configs match
```

Common Scenarios:
Scenario 1: Testing Configuration (Safe)

```
Router(config)# interface Gi0/1
Router(config-if)# shutdown
Router(config-if)# end
! Test if network still works
! If problem: Just reload without saving
Router# reload
! Running-config discarded, startup-config loaded
```

Scenario 2: Accidental Change (Recovery)

```
Router(config)# no ip route 0.0.0.0 0.0.0.0 10.1.1.1
! OOPS! Deleted default route, lost connectivity
Router(config)# end

! Quick recovery: Reload without saving
Router# reload
Proceed with reload? [confirm] yes
! After reload: Old config restored from NVRAM
```

Scenario 3: Merge Configurations

```
! Copy startup TO running (merge)
Router# copy startup-config running-config
! Adds startup commands to running

! Copy running TO startup (replace)
Router# copy running-config startup-config
! Overwrites startup with current running
```

Scenario 4: Erase Configuration

```
Router# erase startup-config
Erasing the nvram filesystem will remove all files!
Continue? [confirm] yes
[OK]
Router# reload
! Router boots with no config (Setup Mode)
```

Best Practices:

- Always test changes before saving
- Save frequently during long config sessions
- Keep backups on TFTP server
- Document changes with `description` command
- Use `show archive` for config history (if enabled)

Configuration Backup:

```
! Backup to TFTP server
Router# copy running-config tftp:
Address: 10.2.77.100
Filename: R1-Lab07-backup.cfg

! Restore from TFTP
Router# copy tftp: running-config
Address: 10.2.77.100
Filename: R1-Lab07-backup.cfg
```

9 Part 6: Serial Interconnection and Static Routing

Exercise 6: Physical Router Interconnection with Static Routing

Hardware: Cisco routers with serial interfaces
Objective: Interconnect routers via serial links and implement static routing

9.1 6.1 Serial Communication Fundamentals

Before interconnecting routers, we studied three key concepts for serial communication.

Concept	Definition
Null Modem	Cable that crosses TX/RX lines to connect two DTE devices directly without modems
Clock Rate	Synchronization speed (bps) set on DCE side to coordinate serial bit transmission
DTE/DCE	DTE (Data Terminal Equipment) receives clock; DCE (Data Circuit Equipment) generates clock

Table 5: Serial communication key concepts

Why Clock Rate is Needed: Serial links require clock synchronization. The DCE side generates timing signals using the `clock rate` command (e.g., 64000 bps). With-

out this, the serial link will not function. DTE side simply receives the clock.

Laboratory Cable Identification:

```
Router# show controllers Serial0/0/0
DCE V.35, clock rate 64000 ! DCE side - needs clock rate
DTE V.35                   ! DTE side - no clock rate needed
```

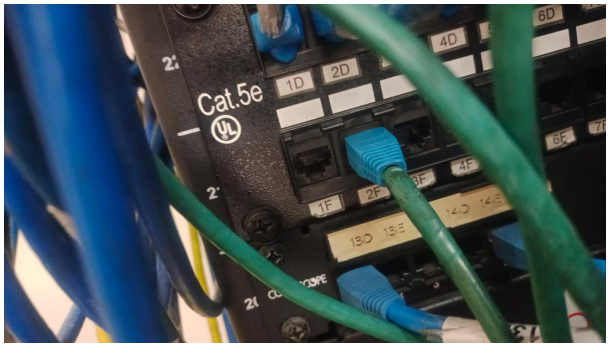

Figure 15: Physical routers with DTE/DCE labeled serial cables

9.2 6.2 Network Topology and Configuration

Laboratory Setup:

```
10.2.77.0/24    100.0.0.0/30     91.0.0.0/24     92.0.0.0/24
   PC1                              PC2             PC3
    |                               |               |
[R5AnderCris]--Serial--[IvanRouter]--Serial--[MejiaRouter]
 .130  Gi0/0  Se0/0/1 Gi0/0  Se0/0/0              Gi0/0
            .2    .1  .1
```

Our Router Configuration (R5AnderCris):

```
! Serial interface to Ivan's router
R5AnderCris(config)# interface Serial0/0/1
R5AnderCris(config-if)# ip address 100.0.0.2 255.255.255.252
R5AnderCris(config-if)# clock rate 64000
R5AnderCris(config-if)# no shutdown

! LAN interface
R5AnderCris(config)# interface GigabitEthernet0/0
R5AnderCris(config-if)# ip address 10.2.77.130 255.255.255.0
R5AnderCris(config-if)# no shutdown

! Save configuration
R5AnderCris# copy running-config startup-config
```

Ivan's Router Configuration:

```
IvanRouter(config)# interface Serial0/0/0
IvanRouter(config-if)# ip address 100.0.0.1 255.255.255.252
IvanRouter(config-if)# no shutdown

IvanRouter(config)# interface GigabitEthernet0/0
IvanRouter(config-if)# ip address 91.0.0.1 255.255.255.0
IvanRouter(config-if)# no shutdown
```

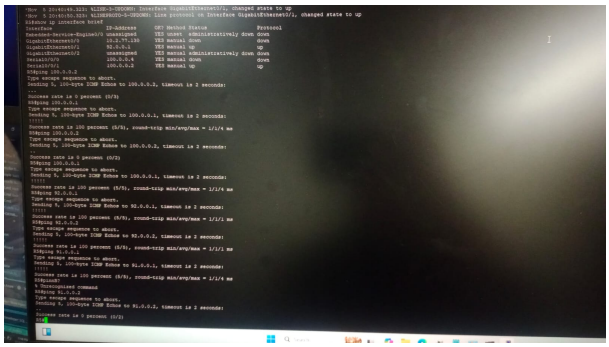

Figure 16: Interface status showing serial and Ethernet configurations

9.3 6.3 Static Routing Implementation

After serial link configuration, routers could only reach directly connected networks. Static routes were required for end-to-end connectivity.

Static Route Concept: Routers need explicit instructions to reach remote networks. Syntax: `ip route <network> <mask> <next-hop>`

Our Router Static Routes:

```
! Route to Ivan's network
R5AnderCris(config)# ip route 91.0.0.0 255.255.255.0 100.0.0.1

! Route to Mejia's network (via Ivan)
R5AnderCris(config)# ip route 92.0.0.0 255.255.255.0 100.0.0.1

R5AnderCris# copy running-config startup-config
```

Ivan's Router Static Routes:

```
! Route back to our network
IvanRouter(config)# ip route 10.2.77.0 255.255.255.0 100.0.0.2

! Route to Mejia's network
IvanRouter(config)# ip route 92.0.0.0 255.255.255.0 <next-hop>
```

Routing Table Verification:

```
R5AnderCris# show ip route
C    10.2.77.0/24 is directly connected, GigabitEthernet0/0
C    100.0.0.0/30 is directly connected, Serial0/0/1
S    91.0.0.0/24 [1/0] via 100.0.0.1
S    92.0.0.0/24 [1/0] via 100.0.0.1
! C = Connected, S = Static
```

9.4 6.4 Connectivity Testing

Router-to-Router Tests:

```
R5AnderCris# ping 91.0.0.1
!!!!! Success rate is 100 percent (5/5)

R5AnderCris# ping 92.0.0.1
!!!!! Success rate is 100 percent (5/5)

R5AnderCris# traceroute 92.0.0.1
  1 100.0.0.1 1 msec * 1 msec
  2 92.0.0.1 4 msec * 4 msec
```

PC-to-PC Tests:

```
PC1(10.2.77.131)> ping 91.0.0.10
Reply from 91.0.0.10: time=2ms TTL=126
Reply from 91.0.0.10: time=2ms TTL=126

PC1> tracert 91.0.0.10
  1   <1 ms  10.2.77.130 [R5AnderCris Gateway]
  2    1 ms  100.0.0.1   [IvanRouter Serial]
  3    2 ms  91.0.0.10   [Destination PC2]
```

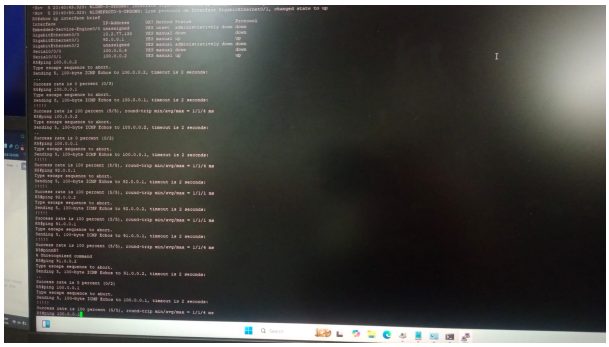

Figure 17: Successful connectivity tests between networks

9.5 6.5 Multi-Group Integration

We extended connectivity to all course groups by adding routes to additional networks:

Group	LAN Network	Serial IP	Router
Group 5 (Us)	10.2.77.0/24	100.0.0.2/30	R5AnderCris
Ivan's Group	91.0.0.0/24	100.0.0.1/30	IvanRouter
Mejia's Group	92.0.0.0/24	100.0.0.4/30	MejiaRouter

Table 6: Network address assignments

Complete Static Routes:

```
R5AnderCris(config)# ip route 91.0.0.0 255.255.255.0 100.0.0.1
R5AnderCris(config)# ip route 92.0.0.0 255.255.255.0 100.0.0.1
R5AnderCris(config)# ip route 93.0.0.0 255.255.255.0 100.0.0.1
! Additional routes for all course networks
```

9.6 6.6 Static Routing Analysis

Advantages	Disadvantages
Predictable routing paths	Time-consuming manual configuration
No protocol overhead	No automatic failover
Secure (no routing updates)	Error-prone (typos break connectivity)
Full administrative control	Difficult to scale
Simple troubleshooting	Requires group coordination

Table 7: Static routing advantages and disadvantages

When to Use Static Routing: Small networks (¡ 10 routers), stub networks, default routes to ISP, security-critical paths, and stable topologies.

Real-World Application: Production environments typically combine static and dynamic routing. Static routes handle critical paths while dynamic protocols (OSPF, EIGRP, BGP) manage general routing automatically.

9.7 6.7 Laboratory Closure

Configuration Backup:

```
R5AnderCris# copy running-config startup-config
R5AnderCris# show startup-config | include hostname
hostname R5AnderCris ! Verified saved
```

Equipment Organization:

1. Document setup (photos, IP addresses, cable types)
2. Disconnect cables in order: Ethernet ¿ Serial ¿ Console ¿ Power
3. Organize and store cables by type
4. Power off routers and return to designated storage
5. Clean workspace for next group

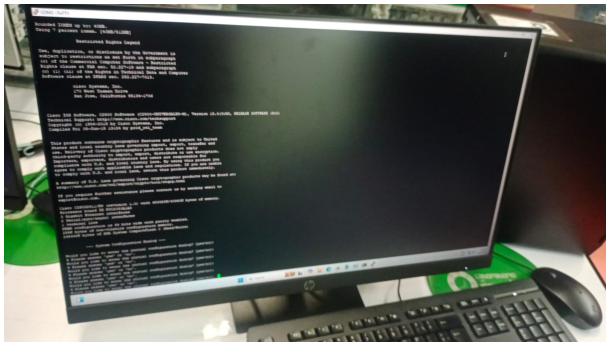

Figure 18: Organized equipment after laboratory session

10 Conclusions

This laboratory provided comprehensive hands-on experience with enterprise network management, cloud administration, and router configuration:

Network Monitoring: Successfully implemented monitoring scripts across three operating systems (Slackware, Solaris, Windows), demonstrating cross-platform network administration capabilities. The scripts provide real-time visibility into network interfaces, connections, routing tables, and traffic statistics.

SNMP Protocol: Deployed a functional SNMP manager-agent architecture, enabling centralized monitoring of remote systems. This protocol is fundamental for enterprise network management systems like Nagios, Zabbix, and PRTG.

Cloud Administration: Gained practical experience with Microsoft Azure's Web App service and Application Insights. Understanding cloud monitoring is critical as organizations migrate from on-premise to cloud infrastructure.

ICMP Analysis: Through traceroute experiments, observed how packets traverse global networks, crossing continents via submarine cables and Internet Exchange Points. This visualization reinforces understanding of the network layer's role in end-to-end delivery.

Router Configuration: Mastered fundamental router administration concepts including boot processes, memory types, password management, and configuration lifecycle. These skills are essential for Cisco CCNA certification and real-world network engineering.

Static Routing: Implemented multi-router static routing, achieving full network connectivity. While static routing has limitations, it provides the foundation for under-

standing more complex dynamic routing protocols (OSPF, EIGRP, BGP).

Key Takeaways:

- Network management requires monitoring tools at multiple layers
- SNMP enables scalable enterprise monitoring
- Cloud platforms offer powerful built-in monitoring capabilities
- Understanding router internals is crucial for troubleshooting
- Static routing provides control but requires careful planning
- Security must be considered at every configuration step

This laboratory successfully integrates theoretical knowledge with practical implementation, preparing us for advanced networking topics and professional network administration roles.

11 References

1. Cisco Systems. (2024). *Cisco IOS Configuration Fundamentals Guide*. Retrieved from cisco.com
2. Stallings, W. (2022). *Data and Computer Communications* (11th ed.). Pearson.
3. Tanenbaum, A. S., & Wetherall, D. J. (2021). *Computer Networks* (6th ed.). Pearson.
4. Microsoft Azure. (2024). *Application Insights Documentation*. Retrieved from docs.microsoft.com
5. Slackware Linux Project. (2023). *Slackware Linux 15.0 Documentation*. Retrieved from slackware.com
6. Oracle. (2024). *Oracle Solaris 11.4 Network Administration Guide*. Retrieved from oracle.com
7. RFC 1157. (1990). *Simple Network Management Protocol (SNMP)*. IETF.
8. RFC 792. (1981). *Internet Control Message Protocol*. IETF.
9. Cisco Networking Academy. (2024). *CCNA: Introduction to Networks*. Cisco Press.
10. Odom, W. (2023). *CCNA 200-301 Official Cert Guide* (Vol. 1). Cisco Press.