

Islamic Azad University

Network Management

&

Monitoring Tool

by

Ghazal Fayaztorshizi

A Bachelor’ s Thesis

Advisor: Dr. Mojtaba Asgari

Islamic Azad University – Mashhad Branch

Winter 2024

Table of Contents

1. [Introduction](#introduction)
2. [Installation and Setup](#installation-and-setup)
   1. [Prerequisites](#prerequisites)
   2. [Project Files](#project-files)
   3. [Running the Tool](#running-the-tool)
3. [Features (Phases)](#features-phases)
   1. [Phase 1: Network Configuration](#phase-1-network-configuration)
   2. [Phase 2: Nftables Management](#phase-2-nftables-management)
   3. [Phase 3: Open vSwitch (OVS) Management](#phase-3-open-vswitch-ovs-management)
   4. [Phase 4: Network Monitoring](#phase-4-network-monitoring)
4. [Challenges and Troubleshooting](#challenges-and-troubleshooting)
   1. [Phase 1 Issues](#phase-1-issues)
   2. [Phase 2 Issues](#phase-2-issues)
   3. [Phase 3 Issues](#phase-3-issues)
   4. [Phase 4 Issues](#phase-4-issues)
5. [Developer Tools and Utilities](#developer-tools-and-utilities)
   1. [nmcli](#nmcli)
   2. [nft](#nft)
   3. [ovs-vsctl](#ovs-vsctl)
   4. [ip / resolvectl / hostnamectl](#ip--resolvectl--hostnamectl)
   5. [curses (Python TUI)](#curses-python-tui)
6. [Recommended Screenshots](#recommended-screenshots)
7. [Summary and Next Steps](#summary-and-next-steps)

[1. Introduction](#introduction)

[1.1 Background and Motivation](#Background)

[1.2 Objectives](#Objectives)

[1.3 Significance of the Study](#Significance)

[2. Phase 1](#Phase1)

[4. Testing and Validation](#TestingandValidation)

[4.1 Connectivity Testing Between Clients](#ConnectivityTesting)

[4.2 DHCP Functionality Verification](#DHCPFunctionalityVerification)

[4.3 DNS Resolution Testing](#DNSResolutionTesting)

[4.4 NAT and Internet Access Testing](#NATandInternetAccessTesting)

[5. Results and Discussion](#ResultsandDiscussion1)

[5.1 Evaluation of DHCP and DNS Services](#EvaluationDHCPandDNS)

[5.2 Challenges and Troubleshooting](#ChallengesandTroubleshooting1)

[5.3 Security Implications](#SecurityImplications1)

[6. Conclusion](#Conclusion1)

[6.1 Summary of Findings](#SummaryFindings1)

[6.2 Achievement of Objectives](#AchievementofObjectives1)

[6.3 Recommendations for Future Work](#RecommendationsforFutureWork1)

**1. Introduction**

**1.1 Background and Motivation**

Simulating real-world networks often requires significant hardware resources, making it challenging to explore complex configurations or troubleshoot advanced scenarios. To address this limitation, virtualization emerges as an ideal solution—allowing us to create local networks entirely within virtual environments such as VMware. By doing so, we can implement routers, clients, and servers without the usual costs and logistical hurdles of physical equipment.

However, understanding the basic structure of a virtual network is only the first step. As networks continue to grow rapidly in both size and complexity, the question arises: **How can we efficiently manage numerous changes and settings without manual, error-prone procedures?** This leads to the concept of **network automation**, which not only saves time but also significantly reduces human error and enhances scalability.

**Note**: This project constitutes **Phase 2** of a broader initiative. In **Phase 1**, a virtual network was implemented, focusing on simulating LAN environments, deploying an Ubuntu-based router, and setting up services like DHCP, DNS, and NAT. **Phase 2** (the focus here) extends this foundation by simplifying the configuration and monitoring of Linux networks through a **Text User Interface (TUI)**. It is designed in four **phases**, each tackling a specific aspect of network administration:

1. **Phase 1** – Network Configuration
2. **Phase 2** – Firewall & NAT (nftables) Management
3. **Phase 3** – Open vSwitch (OVS) Management
4. **Phase 4** – Network Monitoring

Instead of manually editing files like /etc/network/interfaces, resolv.conf, or applying complex nft and ovs-vsctl commands, this TUI-based approach offers an **interactive menu** to handle each operation safely.

**1.2 Objectives**

* **Provide an Intuitive Interface for Network Administrators**
  + Develop a text-based user interface (TUI) to eliminate cumbersome command-line sequences.
  + Simplify tasks like configuring IP addresses, firewall rules, switch ports, and real-time monitoring.
* **Integrate Multiple Network Tools in One Place**
  + Combine nmcli (for permanent network configurations), nftables (for firewall & NAT rules), and Open vSwitch (for virtual switching) under a single TUI umbrella.
  + **Ensure consistent logging and error handling across these tools.**
* **Implement Real-Time Network Monitoring**
  + Provide real-time bandwidth, interface status, and protocol statistics (TCP/UDP).
  + Enhance situational awareness for network operators, allowing quick troubleshooting of performance issues.
* **Demonstrate the Efficiency Gains of Automation**
  + Compare manual network modifications against automated TUI flows.
  + Show how adopting these automated approaches can reduce errors and optimize daily network operations.

**1.3 Significance of the Study**

In modern computing environments, the ability to **simulate** and **automate** network configurations holds immense value. By leveraging **virtualization**, we circumvent resource constraints and gain a risk-free sandbox to test sophisticated setups.

As networks evolve and grow, **automation** takes center stage, enabling faster deployments, streamlined updates, and more reliable performance. This project not only deepens our understanding of foundational networking concepts but also provides hands-on experience with best practices in modern network management. Consequently, the study equips practitioners and learners alike with vital skills to keep pace with the demands of large-scale, dynamically changing networks.

**Installation and Setup**

**Prerequisites**

* **Linux OS with Python 3 installed.**
* **Root (sudo) privileges: Many operations (like changing IPs, setting firewall rules) require root access.**
* **Dependencies:**
  + **nmcli (usually part of NetworkManager)**
  + **nft (from nftables)**
  + **ovs-vsctl (from openvswitch-switch)**
  + **Python 3 library: curses (commonly available as python3-curses)**

**Project Files**

* **All-in-One Script: A single Python file (project\_main.py, for example) containing all four phases.**
* **Phase-Specific Scripts (Optional):**
  + **phase1.py, phase2.py, phase3.py, phase4.py to run each module independently.**

**You can clone or download the repository**

**Install any missing dependencies (e.g., apt-get install -y nftables openvswitch-switch python3-curses).**

**Run the code with sudo:**

**sudo python3 project\_main.py**

2. Features (Phases)

This Phase 2 project internally comprises **four sub-phases**—each sub-phase can be run independently or from a unified “Main Menu” approach:

1. **Phase 1: Network Configuration** (Basic config TUI)
2. **Phase 2: Nftables Management** (Firewall & NAT TUI)
3. **Phase 3: Open vSwitch Management** (Virtual switch TUI)
4. **Phase 4: Network Monitoring** (Real-time stats)

**4. Results and Discussion**

**4.1 Evaluation of DHCP and DNS Services**

* **DHCP**: Observed consistent IP address assignment. No repeated collisions found.
* **DNS**: Both local name resolution and external forwarding succeeded. If logs indicated resolution times, no major latencies were reported.

**4.2 Challenges and Troubleshooting**

(Also detailed in the next **Challenges and Troubleshooting** section, but in summary:

* Running as **root** was essential.
* Service start failures or invalid IP inputs triggered TUI error logs.
* Real-time monitoring needed stable sampling to reflect accurate bandwidth usage.)

**4.3 Security Implications**

* **Firewall**: By implementing nftables rules, the system secures inbound/outbound traffic precisely.
* **OVS**: With trunk and access port controls, VLAN segmentation prevents cross-network traffic leaks.
* **Monitoring**: Real-time stats help detect anomalies early (e.g., spikes in bandwidth usage could mean DDoS or a runaway process).

**5. Challenges and Troubleshooting**

Below is a **detailed** list of issues and solutions encountered across each **internal** sub-phase (Phases 1–4) of this automation tool.

**Phase 1: Network Configuration**

1. **Must Run as Root**
   * **Issue**: Without sudo, nmcli, ip commands fail.
   * **Resolution**: Check os.geteuid() at startup. If not 0, prompt user with an error and exit.
2. **Sub-Process Failures (nmcli, ip)**
   * **Issue**: Invalid interface names or partial command arguments raised subprocess.CalledProcessError.
   * **Resolution**: Wrap calls in try/except, log errors to phase1.log, and show TUI warnings.
3. **Invalid IP Addresses / Subnet Masks**
   * **Issue**: Users might type 999.999.999.999 or a negative mask.
   * **Resolution**: validate\_ip() checks with ipaddress.IPv4Address; ensure mask\_int is 0–32.
4. **Route Not Found After Addition**
   * **Issue**: ip route add might not reflect in ip route show if gateway is wrong.
   * **Resolution**: Re-check with route\_exists(). If missing, log an error and revert.

**Phase 2: Nftables Management**

1. **Nftables Service Fails to Start**
   * **Issue**: systemctl start nftables could return non-zero exit.
   * **Resolution**: Log warnings, attempt reinstall or flush rules, quietly continue if nft works.
2. **Syntax Error with ICMP Rules**
   * **Issue**: Using ip protocol icmp type echo-request accept triggered parse errors.
   * **Resolution**: Updated syntax to ip saddr <src> ip daddr <dst> icmp type echo-request accept.
3. **DNAT / Masquerade Must Go in ip nat**
   * **Issue**: Attempted NAT rules in the filter table caused “invalid argument.”
   * **Resolution**: Place DNAT in prerouting, masquerade in postrouting, appended to /etc/nftables.conf.
4. **User Provided Invalid Inputs**
   * **Issue**: Non-numeric ports or malformed IP addresses break nft.
   * **Resolution**: Validate user input, check port ranges (1–65535), display TUI errors.

**Phase 3: Open vSwitch (OVS) Management**

1. **port\_exists\_in\_bridge Reference Removed**
   * **Issue**: NameError if references remained.
   * **Resolution**: Rely on ovs-vsctl del-port to fail, log the error. No separate function needed.
2. **Trunk to Access (or Vice Versa) Failing**
   * **Issue**: Trying to remove trunks on a non-trunk port returned exit status 1.
   * **Resolution**: Use --if-exists remove port <port\_name> trunks, or a try/except block.
3. **Adding System vs. Internal Ports**
   * **Issue**: System ports must exist, internal ports do not.
   * **Resolution**: TUI question: “System interface or OVS internal?” If system, check interface\_exists(); if internal, skip.
4. **VLAN Interface ‘20’ Not Found**
   * **Issue**: Input like “20” alone is not a valid interface name.
   * **Resolution**: In TUI, instruct user to enter e.g. br0.20. If they only type “20,” show error it doesn’t exist.

**Phase 4: Network Monitoring**

1. **Reading Real-time Stats in 1-second Loops**
   * **Issue**: TUI needed non-blocking input to let user quit with ‘q’.
   * **Resolution**: screen.nodelay(True) plus a small time.sleep(1) loop. On ‘q’, break out.
2. **Inconsistent or Missing /sys/class/net/... Values**
   * **Issue**: Virtual interfaces might not supply link speed or operstate.
   * **Resolution**: Catch exceptions, log warnings, display ‘unknown’ if speed is absent.
3. **Protocol Stats Approach**
   * **Issue**: Parsing /proc/net/snmp or using ss is system-dependent.
   * **Resolution**: We used a quick approach with ss -t -a -n for TCP and ss -u -a -n for UDP. If it fails, log an error but don’t crash.
4. **User Quits TUI Mid-Loop**
   * **Issue**: Potential exceptions if they press ESC or Ctrl-C.
   * **Resolution**: try/finally in real-time loop to restore screen.nodelay(False) and handle partial data gracefully.

**6. Conclusion**

**6.1 Summary of Findings**

This Phase 2 project introduced a robust, TUI-driven approach to automating and monitoring network tasks. Each sub-phase solves a unique challenge:

* **Phase 1**: Provided a streamlined interface for DNS, IP, and route management.
* **Phase 2**: Simplified firewall and NAT rule creation using nftables.
* **Phase 3**: Offered a user-friendly method to build and manage virtual switching topologies with OVS.
* **Phase 4**: Delivered on-demand, real-time monitoring of bandwidth usage and protocol stats.

**6.2 Achievement of Objectives**

1. **Intuitive Interface**: The TUI menus for each sub-phase greatly reduce manual CLI overhead.
2. **Integration of Tools**: nmcli, nft, and ovs-vsctl operate under one solution, consistent logging, and error handling.
3. **Real-Time Monitoring**: The live bandwidth loop and TCP/UDP stats interface effectively highlight network activity.
4. **Automation Efficiency**: Trials show fewer input errors, faster reconfigurations, and clearer logs for debugging.

**6.3 Recommendations for Future Work**

1. **Blockchain-Based Lab Monitoring**:
   * **Possibility**: Integrate blockchain as a transparent, tamper-proof ledger for logging critical network events. For instance, bandwidth usage or firewall rule changes could be registered on a private blockchain for auditing.
   * **Advantage**: Ensures logs are immutable, enabling advanced accountability in multi-operator environments.
2. **Advanced Security Features**:
   * Add intrusion detection or advanced ACL templates in nftables.
3. **CI/CD Testing**:
   * Automate tests for each sub-phase with ephemeral VMs to ensure new changes don’t break existing functionality.
4. **Container-Oriented Approach**:
   * Extend the TUI to manage container networks (e.g., Docker or Kubernetes integration).