**Overview**

This project is composed of two major phases that collectively aim to simplify, automate, and enhance the management of network configurations, firewall rules, virtual switching, and network monitoring on Linux systems. Initially, it was developed to address the complexities associated with establishing, configuring, and maintaining a virtualized multi-segment LAN environment. Building upon these foundations, the second phase broadens the scope by introducing a text-based user interface (TUI) and Python-driven functionalities that streamline interaction with network settings, firewall rules (via nftables), Open vSwitch (OVS) components, and real-time network monitoring data.

The primary objective is to provide administrators and users, even those with limited networking experience, with a command line tool that both reduces manual interventions and fosters a clear understanding of the system’s network state. The resulting tool features menus, submenus, and structured workflows that guide users through configuration steps, ultimately delivering comprehensive network management capabilities.

**Phase 1: Foundational Network Configuration**

The first phase focuses on establishing a robust and efficient network configuration layer in a virtualized environment. Central to this phase is the setup of Ubuntu-based routing, NAT, DHCP, and DNS services, as well as ensuring that clients (Windows Server, Windows 10, and Ubuntu clients) integrate seamlessly into the network. Key achievements in Phase 1 include:

* **Network Topology & LAN Segments**:  
  Creation of multiple LAN segments in a virtual environment (e.g., VMware) to simulate a realistic and isolated network. This architecture allows in-depth testing of routing behavior and address assignments without external interference.
* **Router Configuration**:  
  An Ubuntu router was implemented as a bridging point between internal LAN segments and the external network. This router handles IP forwarding, NAT (Network Address Translation) to facilitate internet-bound traffic, and DHCP services to automate IP address assignment.
* **Static and Dynamic Addressing**:  
  The router and clients can be configured with static IP assignments, ensuring predictable addressing for critical services. Alternatively, DHCP can be employed for automatic and flexible IP assignment. Both temporary and permanent changes are supported, catering to scenarios where transient or persistent configurations are desired.
* **DNS Integration**:  
  A Windows Server instance configured as a DNS server provides hostname-to-IP and IP-to-hostname resolution. Forward and reverse lookup zones, along with DNS forwarders (e.g., using 9.9.9.9), ensure both internal services and external websites can be resolved efficiently.
* **Persistent and Temporary Changes**:  
  All configurations—DNS, hostname, static IP, DHCP activation, and route addition/removal—can be modified either temporarily (persisting until the next reboot or network restart) or permanently (surviving reboots). This duality offers flexibility in testing scenarios before committing changes long-term.

The main challenge of Phase 1 lay in orchestrating these components without human error, ensuring that all modifications (from DNS updates to NAT rules) integrate smoothly. This phase established the bedrock upon which advanced management and visualization tools could be built.

**Phase 2: Advanced Management and Text-based User Interface (TUI)**

Building on the foundational network configuration laid out in Phase 1, Phase 2 introduces a command line TUI written in Python that organizes and controls complex networking tasks into four main areas:

1. **Network Configuration Management**  
   The TUI simplifies the setting of DNS, hostname, static IP, DHCP usage, and routing (both adding and removing routes) through a structured menu system:
   * Users can choose to apply changes temporarily or permanently.
   * The system ensures that the correct set of interfaces is selected for permanent changes, preventing configuration drift or incomplete persistent setups.
2. **Firewall and NAT Management with nftables**  
   Utilizing nftables, this phase enables the definition of firewall rules, NAT configurations, and security policies:
   * Templates for common nftables rules (ct\_state-based, IP-based, ICMP-based, masquerade for source NAT, and DNAT for inbound traffic) are provided.
   * This abstraction reduces complexity for less experienced users, who can select from a menu-driven interface rather than writing raw nftables commands.
   * The TUI ensures the correctness of rules and reduces errors by guiding the user through parameter selection.
3. **Open vSwitch (OVS) Management**  
   Advanced functionalities for managing OVS bridges and ports are integrated:
   * Creating and deleting OVS bridges.
   * Adding and removing ports to those bridges.
   * Controlling port states (up/down), and specifying trunk or access mode.
   * Configuring VLANs and assigning IP addresses to VLAN interfaces.

This approach addresses one of the more complex aspects of modern Linux networking—managing virtual switches and VLAN tagging—through a user-friendly interface.

1. **Network Monitoring Dashboard**  
   This phase includes a real-time network monitoring dashboard:
   * Displays interface states, bandwidth usage, and protocol-level statistics (TCP/UDP).
   * Shows assigned IP addresses per interface.
   * Offers immediate visibility into network health, traffic patterns, and potential issues without requiring deep command-line expertise.

The Phase 2 TUI menus are structured as follows:

**Main Menu:**

1. Network Configuration
2. Manage Firewall (Nftables)
3. Open vSwitch Management
4. Network Monitoring
5. Exit

Each submenu provides intuitive options for operations like changing DNS, setting static IP addresses, manipulating nftables rules (including NAT), configuring OVS, and monitoring network performance.

This phase’s primary challenge lies in ensuring that each function works reliably and transparently. For example, when establishing NAT rules with nftables, the interface’s IP used for NAT must be accurately identified. The TUI and internal logic must thoroughly test these rules to guarantee correct behavior, preserving connectivity and security.

**Project Goals and Challenges**

**Holistic Network Management**:  
The combined phases produce a solution that moves from a manually configured virtual LAN environment (Phase 1) to a fully interactive, TUI-based tool (Phase 2). Users can not only set up their network with stable foundations but also manage, secure, and visualize it in real-time.

**Automation and User-Friendly Approach**:  
Both phases emphasize simplification and automation. Phase 1 reduced complexity in configuring basic network settings, while Phase 2 introduced a structured interface to handle more advanced operations. This design ethos ensures that even users without deep networking backgrounds can carry out complex tasks confidently.

**Flexibility and Adaptability**:  
Throughout both phases, the concept of temporary vs. permanent changes, the use of templates for nftables rules, and the modular approach to OVS configuration underscore a philosophy of adaptability. The system can be tested incrementally, changes can be rolled out cautiously, and failures can be quickly identified and reverted.