

Chapter 1

ABSTRACT

Virtual network configuration refers to the process of creating and managing virtualized network resources within a cloud or data center environment. This approach enables the abstraction of physical network infrastructure, allowing for greater flexibility, scalability, and efficiency in managing network services. Key components include virtual switches, routers, and firewalls, which facilitate communication between virtual machines and external networks.

The configuration process involves defining network topologies, assigning IP addresses, and implementing security protocols. Advanced features such as network slicing, dynamic resource allocation, and automated provisioning enhance performance and reduce operational costs. By leveraging software-defined networking (SDN) and network function virtualization (NFV), organizations can optimize their network resources to meet evolving demands and improve overall reliability.

At the core of virtual network configuration are technologies such as software-defined networking (SDN) and network function virtualization (NFV). These technologies decouple network functions from hardware, allowing for the dynamic allocation of resources and simplified management. Key components include virtual switches, routers, and firewalls, which facilitate communication among virtual machines (VMs) and between VMs and external networks.

Chapter 2

INTRODUCTION

About the Project

Virtual Network Configuration is the process of setting up and managing virtualized networking environments that mimic traditional physical networks. This involves defining the necessary components such as IP addresses, subnets, routing tables, and security rules to ensure resources within the network can communicate securely and efficiently.

Purpose:

The main purpose of virtual network configuration is to enable seamless communication between virtual resources (like virtual machines, containers, databases) while providing flexibility, scalability, and security. Virtual networks allow organizations to logically separate and manage workloads, reducing the need for physical networking hardware.

Goals:

Isolation: Segregate environments (e.g., development, testing, production) or workloads for better management and security.

Connectivity: Facilitate communication between virtual resources, other networks, and external environments (like on-premises systems or the internet).

Security: Implement firewall rules, access control, and encryption to protect network traffic and prevent unauthorized access.

Cost Efficiency: Reduce the need for physical network devices, lowering infrastructure costs and simplifying network management.

Overall, virtual network configuration helps create a flexible, secure, and cost-effective networking environment that supports modern IT infrastructure needs.

Chapter 3

SCOPE OF THE PROJECT

Problem Statement

CRUD:Network Settings -setup_virtual_networks(network_config).Setup and configure virtual networks.-update_network_settings(settings_id):Update network settings as needed.

Languages/Technologies Used

- **Programming Language:** Python (can also be implemented in C++)
- **Tools:** Any basic code editor (VS Code, PyCharm, Turbo C++, etc.)
- **OS:**Windows/Linux

Chapter 4

IMPLEMENTATION

Imagine an IT department within a medium to large organization that manages multiple virtual networks for various departments, such as finance, human resources, and development. The Virtual Network Configuration system helps the IT team efficiently configure, manage, and update virtual networks, ensuring that network resources are used optimally and securely.

CRUD: Network Settings The system provides capabilities to create, read, update, and delete network settings, allowing the IT team to maintain accurate and current configurations for each virtual network.

Example:

- **Create:** When a new department is formed, the IT team creates network settings for the finance department, specifying the IP address range, subnet mask, and other relevant configurations.
- **Read:** The IT staff can view existing network settings to ensure they align with the department's requirements. For instance, they check the settings for the Development Network to verify the allocated resources.
- **Update:** If the organization undergoes a merger, the IT team can update the network settings to accommodate new departments, changing the IP address range and adding VLAN configurations as necessary.
- **Delete:** When a project is completed, the corresponding virtual network settings are deleted from the system to avoid clutter and confusion.

Chapter 5

ALGORITHM

1. Start
2. Initialize a dictionary `network_settings` to store network data.
1. Create Network
 - If `network_id` exists → print "Already exists"
 - Else → add network and confirm creation
2. Read Network
 - If `network_id` exists → print details
 - Else → print "Does not exist"
3. Update Network
 - If `network_id` exists → update fields and confirm
 - Else → print "Does not exist"
4. Delete Network
 - If `network_id` exists → delete and confirm
 - Else → print "Does not exist"
5. Setup Virtual Network
 - If `network_id` does not exist → call `create_network()`
 - Else → print "Already exists"
6. Update Network Settings
 - If `settings_id` exists → update with `new_config`
 - Else → print "Does not exist"
7. End

Chapter 6

SOURCE CODE

```
network_settings = {  
    1: {'name': 'Network_1', 'ip_range': '192.168.1.0/24', 'status': 'active'},  
    2: {'name': 'Network_2', 'ip_range': '192.168.2.0/24', 'status': 'inactive'}  
}  
  
def create_network():  
    try:  
        network_id = int(input("Enter network ID: "))  
        name = input("Enter network name: ")  
        ip_range = input("Enter IP range (e.g., 192.168.1.0/24): ")  
        status = input("Enter status (active/inactive): ")  
        if network_id in network_settings:  
            print(f"❌ Network with ID {network_id} already exists.")  
        else:  
            network_settings[network_id] = {  
                'name': name,  
                'ip_range': ip_range,  
                'status': status  
            }  
            print(f"✅ Network '{name}' created with ID {network_id}.")  
    except ValueError:  
        print("Invalid input. Network ID must be an integer.")  
  
def read_network():
```

```
try:

    network_id = int(input("Enter network ID to view: "))

    network = network_settings.get(network_id)

    if network:

        print(f"📁 Network {network_id}: {network}")

    else:

        print(f"❌ Network with ID {network_id} does not exist.")

except ValueError:

    print("Invalid input. Please enter a valid network ID.")

def update_network():

    try:

        network_id = int(input("Enter network ID to update: "))

        if network_id in network_settings:

            name = input("Enter new name (press Enter to skip): ")

            ip_range = input("Enter new IP range (press Enter to skip): ")

            status = input("Enter new status (press Enter to skip): ")

            if name:

                network_settings[network_id]['name'] = name

            if ip_range:

                network_settings[network_id]['ip_range'] = ip_range

            if status:

                network_settings[network_id]['status'] = status

            print(f"✅ Network {network_id} updated.")

        else:

            print(f"❌ Network with ID {network_id} does not exist.")
```

```

except ValueError:

    print("Invalid input. Please enter a valid network ID.")

def delete_network():

    try:

        network_id = int(input("Enter network ID to delete: "))

        if network_id in network_settings:

            del network_settings[network_id]

            print(f"🗑 Network {network_id} deleted.")

        else:

            print(f"❌ Network with ID {network_id} does not exist.")

    except ValueError:

        print("Invalid input. Please enter a valid network ID.")

def setup_virtual_network():

    try:

        network_id = int(input("Enter new virtual network ID: "))

        name = input("Enter network name: ")

        ip_range = input("Enter IP range: ")

        status = input("Enter status: ")

        if network_id not in network_settings:

            network_settings[network_id] = {

                'name': name,

                'ip_range': ip_range,

                'status': status

            }

            print(f"✅ Virtual network '{name}' set up successfully.")

```



```

else:

    print(f"⚠ Network with ID {network_id} already exists.")

except ValueError:

    print("Invalid input. Please enter a valid network ID.")

def update_network_settings():

    try:

        settings_id = int(input("Enter network ID to update configuration: "))

        if settings_id in network_settings:

            ip_range = input("Enter new IP range (press Enter to skip): ")

            status = input("Enter new status (press Enter to skip): ")

            if ip_range:

                network_settings[settings_id]['ip_range'] = ip_range

            if status:

                network_settings[settings_id]['status'] = status

            print(f"✅ Network {settings_id} updated with new configuration.")

        else:

            print(f"❌ Network with ID {settings_id} does not exist.")

    except ValueError:

        print("Invalid input. Please enter a valid ID.")

def display_menu():

    print("\n🌐 Virtual Network Configuration")

    print("-----")

    print("1. Create Network")

    print("2. Read Network")

    print("3. Update Network")

```

```
print("4. Delete Network")

print("5. Setup Virtual Network")

print("6. Update Network Settings")

print("7. Exit")

def main():

    while True:

        display_menu()

        choice = input("Select an option (1-7): ")

        if choice == '1':

            create_network()

        elif choice == '2':

            read_network()

        elif choice == '3':

            update_network()

        elif choice == '4':

            delete_network()

        elif choice == '5':

            setup_virtual_network()

        elif choice == '6':

            update_network_settings()

        elif choice == '7':

            print("🏠 Exiting... Goodbye!")

            break

        else:

            print("❌ Invalid choice. Please select from 1 to 7.")
```

```
if __name__ == "__main__":  
    main()
```

Chapter 7

OUTPUT

Virtual Network Configuration

1. Create Network
2. Read Network
3. Update Network
4. Delete Network
5. Setup Virtual Network
6. Update Network Settings
7. Exit

Select an option (1–7): 1

Enter network ID: 3

Enter network name: network_3

Enter IP range (e.g., 192.168.1.0/24): 123

Enter status (active/inactive): active

☒ Network 'network_3' created with ID 3.


Virtual Network Configuration

1. Create Network
2. Read Network
3. Update Network
4. Delete Network

5. Setup Virtual Network
6. Update Network Settings
7. Exit

Select an option (1–7): 2

Enter network ID to view: 2


 Network 2: {'name': 'Network_2', 'ip_range': '192.168.2.0/24', 'status': 'inactive'}

Virtual Network Configuration

1. Create Network
2. Read Network
3. Update Network
4. Delete Network
5. Setup Virtual Network
6. Update Network Settings
7. Exit

Select an option (1–7): 4

Enter network ID to delete: 3

 Network 3 deleted

Chapter 8

CONCLUSION

Providing an organized and structured way to manage network settings across various virtual networks, ensuring that configurations are current and accurate. Enabling seamless setup of virtual networks tailored to specific departmental needs, allowing for flexibility and isolation of resources. Facilitating quick and efficient updates to network settings in response to changing organizational requirements, maintaining security and compliance.

It ensures that the network infrastructure is robust, secure, and adaptable, ultimately supporting business operations and enhancing productivity across

1. AI-Driven Automation: Automate network setup, management, and self-healing using AI.
2. Enhanced Security: Implement context-aware security, zero trust, and AI-driven threat detection.
3. Advanced Monitoring: Real-time interactive maps, predictive analytics, and detailed traffic insights.
4. User-Friendly Interfaces: Drag-and-drop design, pre-built templates, and virtual testing environments.
5. Multi-Cloud Integration: Seamless networking across multiple cloud providers and edge computing support.
6. Scalability: Use NFV, dynamic bandwidth, and latency optimization. and so on....

Chapter 9

REFERENCES

<https://www.sdxcentral.com/security/definitions/data-security-in-the-cloud-best-practices/virtual-network-security-works/>

<https://docs.oracle.com/cd/E19504-01/802-5753/6i9g71m52/index.html>

<https://docstore.mik.ua//manuals/hp-ux/en/T2767-90141/ch08s01.html>