Network Configuration Basics

Hiranya Prasad Bastakoti

Contents

Network Interface configuration

- Network interface is the point of connection between a computer and a network.
- A network interface is the point of interconnection between a computer and a private or public network.
- A network interface is generally a network interface card (NIC), but does not have to have a physical form. Instead, the network interface can be implemented in software.

- Network interface configuration involves setting up and customizing network interfaces on a computer or device.

Steps:

1. Determine network interfaces: Identify the available network interfaces on your computer using commands : ifconfig (Linux/Unix) or ipconfig (Windows) in the terminal or command prompt.

2. Configuration files: Network interface settings are stored in operating system-specific configuration files.

   On Linux/Unix systems, these files are typically found in directories :

   /etc/network or /etc/sysconfig/network-scripts.

3. Static or dynamic IP addressing: Choose between manually assigning a static IP address or using dynamic IP assignment through DHCP.
4. IP address assignment: Set the IP address, subnet mask, and default gateway for the network interface. For static IP addressing, manually enter these values in the configuration file. For dynamic IP addressing, ensure DHCP is enabled.

- 5.DNS configuration: Specify the DNS servers that the network interface should use for domain name resolution. Edit the DNS configuration file or provide DNS server IP addresses in the network interface configuration.
- 6.Network services and protocols: Enable or disable specific network services and protocols based on your requirements, such as IPv4/IPv6, network bridging, VLAN tagging, and firewall settings.

- Restart network services: After modifying the network interface configuration files, restart the network services to apply the changes.

  The specific command or procedure depends on your operating system or distribution, like service network restart, systemctl restart network, or netplan apply.
- Test connectivity: Verify the network interface configuration by testing network connectivity using commands : ping to check connectivity to other devices or hosts on the network.

Diagnosing Network startup issues

- *Network diagnosis refers to the process of identifying the cause of faults in the network.*

Steps:

- Check the status of the network service: Verify the network service status by using the command systemctl status network. Look for any error messages or warnings that can help identify the cause of the network startup problem.
- Examine network configuration files: Review the network configuration files located in the /etc/sysconfig/network-scripts/ directory.
- Typically, the primary configuration file is named ifcfg-<interface_name> (e.g., ifcfg-eth0). Ensure that the configuration settings, such as IP address, subnet mask, gateway, and DNS servers, are accurate and properly configured.

- Verify the status of network interfaces: Use commands like ip link show or ifconfig to check the status of the network interfaces. Make sure the interfaces are marked as "UP" and have valid IP addresses assigned. If an interface is inactive or lacking an IP address, it may indicate a configuration issue or potential hardware problem.
- Restart the network service: Attempt to restart the network service by executing the command systemctl restart network. Pay attention to any error messages that may appear during the restart process.
- Check network logs: Inspect the system logs, typically located at /var/log/messages, for any network-related errors or warnings. Use the grep command to filter the log entries related to networking, such as grep network /var/log/messages.

- Verify connectivity and DNS resolution: Test network connectivity by using the ping command to reach other devices or hosts on the network. Additionally, check DNS resolution by running commands like nslookup or dig to ensure proper resolution of domain names into IP addresses. If there are connectivity or DNS resolution issues, it could indicate problems with network configuration or DNS server settings.
- Review firewall settings: Ensure that the firewall rules are not blocking necessary network traffic. Review the firewall configuration using the firewall-cmd command and make any required adjustments.
- Validate hardware and cables: Verify that the network hardware, including network cards and cables, are properly connected and functioning. Check for any physical damage or loose connections that could impact network startup.

Network troubleshooting commands

- *Troubleshooting is a repetitive, rigorous, and effective process that involves regular analysis and testing of individual network components to ensure smooth operations*.
- Ping: This command is used to test network connectivity by sending ICMP echo request packets to a target host.
- For example, execute ping 192.168.1.1 or ping youtube.com.
- Traceroute or tracepath: These commands display the route that network packets take from your computer to a destination host, showing the IP addresses of intermediate routers along the path.
- For Example: traceroute google.com or tracepath 192.168.1.1.
- ifconfig: This command provides information about network interfaces, including IP addresses, netmasks, and other relevant details.
- For example: ifconfig or ifconfig eth0

- netstat: This command shows active network connections, listening ports, and other network statistics.
- For example, run netstat -tuln to display listening TCP and UDP ports.
- nslookup or dig: These commands query DNS (Domain Name System) servers to resolve domain names into IP addresses.
- For example: nslookup google.com or dig google.com.

- nmap: This powerful network scanning and mapping tool is used to discover open ports, services, and network vulnerabilities.
- Example, execute nmap -p 1-1000 192.168.1.1 to scan ports 1 to 1000 on the specified IP address.
- tcpdump: Use this command to capture and analyze network traffic on a specific network interface. For example, run tcpdump -i eth0 to capture traffic on the eth0 interface.
- iptables: This command-line utility allows for configuring firewall rules in linux, managing network traffic, and filtering packets based on various criteria.
- **arp** — A utility that supports the Address Resolution Protocol (ARP) service of the TCP/IP protocol suite. It lets the network admin view the ARP cache and add or delete cache entries. It can be used to address problems having to do with specific connections between a workstation and a host.
- Example:arp -e

Windows Firewall Configuration

- Access Windows Firewall settings: Press the Windows key, type "Windows Firewall," and select "Windows Defender Firewall" or "Windows Firewall with Advanced Security" from the search results.
- Check firewall status: Ensure that the firewall is enabled for the desired network profile (e.g., domain, private, public) in the Windows Firewall settings.
- Define inbound and outbound rules: Create rules to allow or block specific traffic. You can specify rules based on port numbers, protocols, programs, or specific IP addresses.
- Create a new rule: Use the "New Rule" option in the "Inbound Rules" or "Outbound Rules" section to create a new rule. Follow the wizard to specify the rule criteria, such as ports, protocols, and actions (allow or block).

- Modify existing rules: Right-click on a rule and select "Properties" to modify its settings, including ports, protocols, programs, and actions.
- Customize profiles: Customize the firewall settings for each network profile (e.g., domain, private, public) based on your network environment.
- Access advanced settings: Use the "Advanced settings" option in the Windows Firewall settings to access more advanced configuration options. This allows you to create specific rules, configure connection security rules, and adjust authentication settings.
- Save and apply settings: Save the changes you made to the firewall settings and ensure that they are applied.

Linux firewall configuration

- Select a firewall tool: Choose a suitable firewall tool for your Linux distribution, such as iptables, ufw, or firewalld.
- Install and enable the firewall tool: Install the firewall tool using your distribution's package manager. Enable the firewall and start its service.
- Check firewall status: Use the appropriate command to verify the current firewall status. For example, use iptables -L or iptables -S for iptables, ufw status for ufw, or firewall-cmd --state for firewalld.
- Define firewall rules: Use the specific commands or configuration files for your firewall tool to define rules. For instance, use commands like iptables -A INPUT -p tcp --dport 22 -j ACCEPT to allow SSH connections with iptables, or use ufw allow 22 to enable SSH access with ufw.

- Save firewall rules: Save the rules to make them persistent across reboots. The method varies depending on the firewall tool. For example, use iptables-save > /etc/iptables/rules.v4 for iptables, or use firewall-cmd --runtime-to-permanent for firewalld.
- Modify default policies: Adjust the default policies for incoming and outgoing traffic as needed. This step allows you to control how the firewall handles traffic by default.
- Test the firewall: Verify that the firewall rules are functioning correctly by testing desired connections and ensuring that undesired traffic is blocked.
- Refine rules as needed: Monitor firewall logs and network activity to identify any issues or additional rules that may be necessary. Adjust the rules accordingly to enhance network security.

Mininet Overview

- Mininet is a popular open-source software emulator designed for creating virtual networks within a single machine.
- It is extensively utilized for network programming and testing purposes, primarily in the realm of software-defined networking (SDN).
- Mininet is a *network emulator* which creates a network of virtual hosts, switches, controllers, and links.
- Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.
- Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.
- Mininet provides an easy way to get correct system *behavior* (and, to the extent supported by your hardware, performance) and to experiment with topologies.
- Mininet networks run *real code* including standard Unix/Linux network applications as well as the real Linux kernel and network stack (including any kernel extensions which you may have available, as long as they are compatible with network namespaces.)

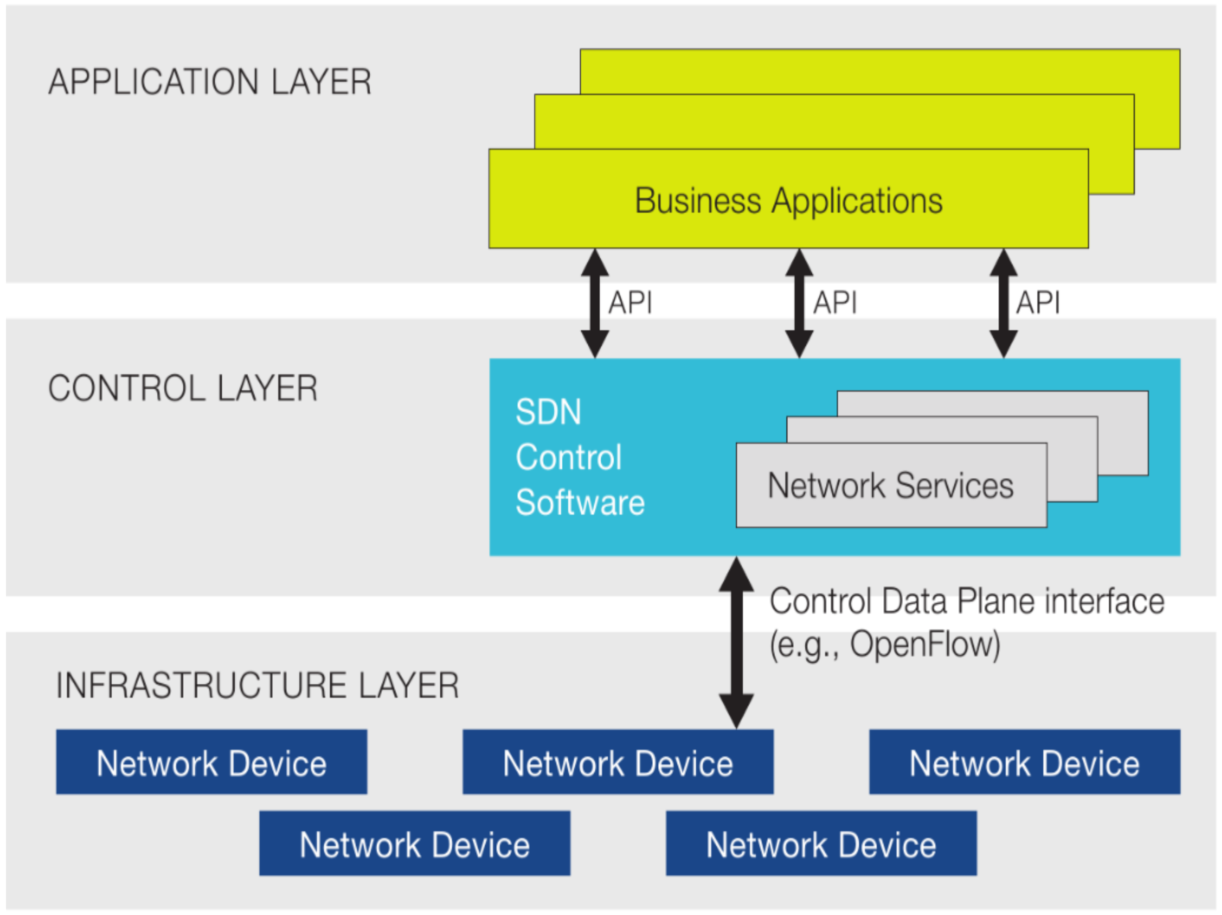
Features

- Provides a simple and inexpensive **network testbed** for developing OpenFlow applications
- Enables **multiple concurrent developers** to work independently on the same topology
- Supports **system-level regression tests**, which are repeatable and easily packaged
- Enables **complex topology testing**, without the need to wire up a physical network
- Includes a **CLI** that is topology-aware and OpenFlow-aware, for debugging or running network-wide tests
- Supports **arbitrary custom topologies**, and includes a basic set of **parametrized topologies**
- It is **usable out of the box** without programming, but
- also Provides a straightforward and extensible **Python API** for network creation and experimentation

SDN

- A software-defined networking (SDN) architecture (or SDN architecture) defines how a networking and computing system can be built using a combination of open, software-based technologies and commodity networking hardware that separate the SDN control plane and the SDN data plane of the networking stack.
- A software-defined network (SDN) architecture (or SDN architecture) defines how a networking and computing system can be built using a combination of open, software-based technologies and commodity networking hardware.
- Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications.
- This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.

SDN


APPLICATION LAYER
Business Applications
API
API
API
CONTROL LAYER
SDN Control Software
Network Services
Control Data Plane interface (e.g., OpenFlow)
INFRASTRUCTURE LAYER
Network Device
Network Device
Network Device
Network Device
Network Device

Architecture of SDN

- SDN Applications:

SDN Applications are programs that communicate behaviors and needed resources with the SDN Controller via application programming interfaces (APIs).

In addition, the applications can build an abstracted view of the network by collecting information from the controller for decision-making purposes.

These applications could include networking management, analytics, or business applications used to run large data centers.

For example, an analytics application might be built to recognize suspicious network activity for security purposes.

- SDN Controller:

The SDN Controller is a logical entity that receives instructions or requirements from the SDN Application layer and relays them to the networking components.

The controller also extracts information about the network from the hardware devices and communicates back to the SDN Applications with an abstract view of the network, including statistics and events about what is happening.

- SDN Networking Devices: The SDN networking devices control the forwarding and data processing capabilities for the network.
- This includes forwarding and processing of the data path.
- *The SDN architecture APIs are often referred to as northbound and southbound interfaces, defining the communication between the applications, controllers, and networking systems.*
- *A Northbound interface is defined as the connection between the controller and applications, whereas the Southbound interface is the connection between the controller and the physical networking hardware.*
- *Because SDN is a virtualized architecture, these elements do not have to be physically located in the same place*.

SDN controller and dataplane communication

- Initialization: When a data plane device connects to the network, it establishes a connection with the SDN controller using TCP/IP.
- Handshake: The data plane device and the SDN controller perform a secure authentication process to establish a trusted communication channel.
- Flow table configuration: The SDN controller pushes flow rules to the data plane devices, specifying how network traffic should be processed and forwarded.
- Event notification: Data plane devices report network events to the SDN controller, such as packet arrival, enabling dynamic network management.
- Packet processing: Upon packet arrival, the data plane device checks the flow rules in its flow table to determine how to handle the packet based on its headers.
- Statistics collection: Data plane devices collect and send network statistics, such as packet counters and bandwidth usage, to the SDN controller for network monitoring and decision-making.

Routing Configuration SDN

- In Software-Defined Networking (SDN), configuring routing involves establishing and managing the paths through which network traffic is directed and forwarded to different destinations.
- The SDN controller plays a crucial role in centrally controlling and configuring the routing behavior within the network.

Steps

- Topology discovery: The SDN controller gathers information about the network topology, including switches, routers, and their connections.
- Routing policy definition: Network administrators define routing policies based on specific requirements and objectives, specifying how traffic should be routed.
- Path computation: The SDN controller computes optimal paths for network traffic based on the network topology and defined routing policies.
- Flow rule installation: The SDN controller translates computed paths into flow rules and installs them in the data plane devices (e.g., switches) to guide traffic forwarding.

- Traffic steering: Data plane devices match incoming packets against flow rules and forward them along the specified paths.
- Dynamic adaptation: The SDN controller monitors network conditions and adjusts the routing configuration as needed, such as computing and installing new paths in case of congestion or failures.
- Policy enforcement: The SDN controller ensures that the defined routing policies are enforced, monitoring traffic and taking actions to maintain policy compliance.

Open source networking monitoring

- Network monitoring tools are software applications or platforms designed to monitor and manage computer networks.
- These tools provide administrators with visibility into the performance, availability, and security of network devices, services, and applications.
- Network monitoring tools typically collect data from network devices, analyze it, and present it in a useful and actionable format.
- They help identify and troubleshoot network issues, optimize network performance, and ensure efficient utilization of network resources.

Functions

- Device Monitoring: Network monitoring tools track the status and performance of devices like routers, switches, servers, and endpoints. They monitor metrics such as CPU usage, memory utilization, bandwidth consumption, and network latency.
- Service Monitoring: These tools ensure that network services and applications are functioning correctly and meeting performance objectives. They monitor protocols like HTTP, SMTP, DNS, and databases, checking for availability, response times, and proper functioning.
- Traffic Analysis: Network monitoring tools capture and analyze network traffic to gain insights into traffic patterns, bandwidth usage, and identify potential bottlenecks. They help administrators optimize network traffic flow and detect abnormal or malicious traffic behavior.

- Alerting and Notifications: These tools generate alerts or notifications when predefined thresholds or conditions are met, allowing administrators to address network issues promptly. Alerts can be delivered via email, SMS, or other communication channels.
- Performance Visualization: Network monitoring tools present network performance metrics and trends through visual representations like graphs, charts, and dashboards. This visual display aids administrators in quickly understanding network conditions and identifying areas that require attention.

- Configuration Management: Advanced network monitoring tools offer configuration management features, enabling administrators to track and manage network device configurations. This ensures consistency and compliance across the network.
- Security Monitoring: Network monitoring tools assist in detecting and mitigating security threats by monitoring network traffic for unauthorized access attempts, malware, or unusual behavior. They provide alerts or integrate with security systems for further investigation.

Examples:

- **Zabbix** The best overall balance between open-source flexibility, support, and out-of-the-box ease of use.
- **Icinga** Great API and documentation.
- **Prometheus** Uses a powerful query language to generate insights and display data.
- **Nagios** Offers both paid and free open source networking monitoring tools.
- **Cacti** Highly customizable, great for operations leveraging big data.

- **Nagios** provides a suite of open-source tools that includes networking, infrastructure, and application monitoring.
- While the platform is open source, the only free version available is Nagios Core.
- Products like Nagios XI provide enterprise-level features, support, and more pre-made dashboards and alerts.

Key Features:

- Nagios Core is free to use
- Extensible with plug-ins
- The leading open-source network monitor
- Multiple support options, including free onboarding assistance
- Alerts and insights are incredibly fast and work in near real-time
- Can monitor virtually anything, uses standard SNMP

Open source networking monitoring (e.g. Nagios)

What can Nagios help you do?
Windows Monitoring
Linux Monitoring
Server Monitoring
Application Monitoring
SNMP Monitoring
Log Monitoring

QA