Department of Electronic & Telecommunication Engineering
University of Moratuwa, Sri Lanka.

EN2150 Communication Network Engineering
SDN Algorithm Development

Group : ByteLink

210387D	Mihiranga N.G.D.
210391J	Morawakgoda M.K.I.G.
210610H	Sirimanna N.T.W.

31st August 2024

Contents

1 Introduction

In modern networking, the efficient and effective routing of traffic is critical to ensuring both optimal resource utilization and high-quality user experiences. Traditional routing protocols, which primarily consider the shortest path based on the destination IP address, often result in traffic congestion on certain network links, leading to inefficiencies and suboptimal use of the network infrastructure. This limitation is particularly concerning for service providers, who invest significantly in network infrastructure and need to ensure that all links are utilized to their fullest potential to achieve a favorable Return on Investment.

Software-Defined Networking offers a paradigm shift by centralizing control in the network's controller, which can dynamically program the network to meet specific requirements, such as traffic engineering and Quality of Service (QoS). Unlike legacy routing protocols, SDN allows for more granular control over how traffic is routed through the network, based on real-time data collected from the data plane.

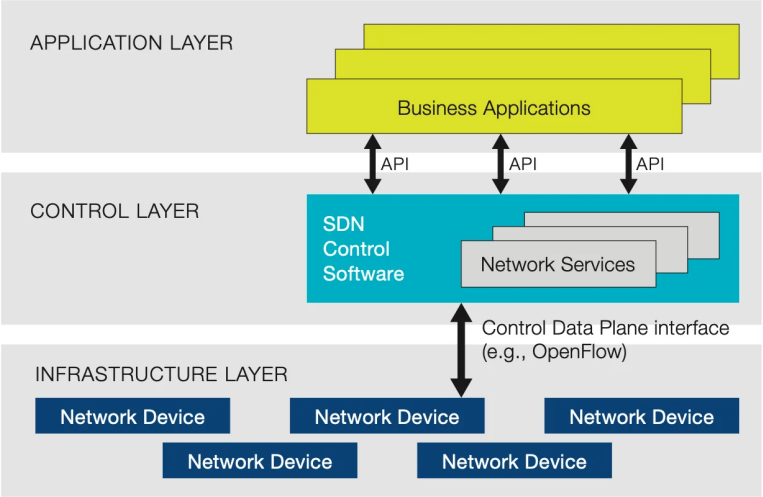


Figure 1: Software Define Network

This report presents the development of routing algorithms tailored to two critical scenarios: Traffic Engineering and Quality of Service. The Traffic Engineering algorithm aims to distribute network traffic across multiple paths, preventing congestion and ensuring that all network links are effectively utilized. Meanwhile, the QoS algorithm is designed to prioritize traffic based on the application type, ensuring that users experience the best possible quality regardless of network conditions.

2 Traffic Engineering Algorithm

Traffic Engineering aims to optimize the use of network resources by efficiently distributing traffic across all available links, rather than relying solely on the shortest path. The objective is to ensure that traffic is distributed across multiple paths to maximize resource utilization and avoid congestion on any single link. This approach is crucial for preventing the over-utilization of certain links, which can lead to congestion, and under-utilization of others, which would result in wasted resources.

2.0.1 Algorithm Overview

This algorithm begins with gathering detailed network data to understand its structure and limitations. Next, it calculates the most efficient routes using algorithms like Dijkstra's or Bellman-Ford, considering factors such as latency and bandwidth. The algorithm then continuously monitors network conditions to dynamically adjust traffic distribution, balancing the load and avoiding congestion. Finally, it updates forwarding rules on network devices to enforce the opti mized traffic paths, ensuring efficient and resilient network operation.

Algorithm 1 Traffic Engineering Algorithm in SDN

```
Initialize network topology with nodes and links
Initialize SDN controller to collect data from the data plane at regular intervals
while true do
    data_plane_metrics ← controller.collect_metrics()
    congestion_detected ← check_congestion(data_plane_metrics)
    if congestion_detected then
        for each flow ∈ active_flows do
            paths ← calculate_alternative_paths(flow, topology, data_plane_metrics)
            selected_path ← select_optimal_path(paths, data_plane_metrics)
            controller.install_flow_entries(flow, selected_path)
        end for
    end if
    monitor_traffic_and_adjust_paths(topology, data_plane_metrics)
end while
```

Figure 2: Traffic Engineering Algorithm with SDN

2.0.2 Network Initialization

Network initialization is the critical first step in Traffic Engineering, where the network controller compiles comprehensive information about the network's current state. This step lays the groundwork for effective traffic management by providing a detailed understanding of the network's structure, capabilities, and initial conditions.

Network Topology Discovery

The first task for the network controller is to map out the entire network topology. This involves identifying all nodes (such as routers and switches) and the links connecting them. The controller utilizes standard protocols, such as the Link Layer Discovery Protocol (LLDP) for discovering the network topology and Simple Network Management Protocol (SNMP) for monitoring the status of network links. Through these protocols, the controller gathers essential details about each node and link.

The essential details about each node and link:

- **Link Bandwidth:** The maximum data transfer rate supported by each link.
- **Link Latency:** The time it takes for data to traverse each link.
- **Processing Power and Memory:** The capabilities of each node in terms of processing speed and available memory.
- **Node Interfaces:** The number and type of interfaces available on each node.

For even a small, simple network, this process enables the controller to understand the bandwidth, latency, and current status (active or down) of each link, along with the connections between nodes.

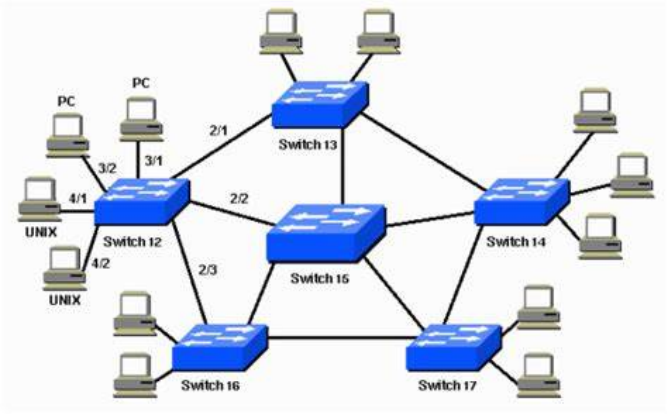


Figure 3: Network Toplolgy

Link Capacities and Constraints

The controller assesses the capacity of each link, which is the maximum amount of data that can be transmitted over it per unit of time. The constraints that could affect the performance of the link.

- **Policy Restrictions:** Organizational policies that limit how certain links can be used.
- **Physical Limitations:** Physical attributes of the links that might limit their capacity.
- **Security Constraints:** Security requirements that might restrict data flow over specific links.

Hence, the controller can establish utilization thresholds for each link to prevent congestion. Historical usage patterns and real-time network data are then analyzed to adjust these thresholds dynamically, ensuring that the network remains adaptable to current conditions.

Building the Traffic Demand Matrix

The controller constructs a traffic demand matrix to effectively manage traffic. This matrix provides an initial estimate of the traffic demands between different nodes in the network, serving as a crucial tool for predicting potential congestion points. The matrix is based on:

- **Historical Data:** Previous traffic patterns that offer insights into typical network behavior.
- **Real-Time Data:** Current traffic conditions that reflect the present state of the network.

The traffic demand matrix is essential for identifying patterns and predicting congestion, allowing the controller to allocate resources more effectively. In networks with stable traffic patterns, this matrix may require only occasional updates. However, in more dynamic environments, continuous updates are necessary to reflect the rapidly changing traffic flows.

Processing the Collected Data

Once all relevant data is collected, it is crucial to process this information into a format that the Traffic Engineering algorithm can utilize efficiently. This involves standardizing the data and assigning weights to the network links based on their characteristics, such as:

- **Latency:** Links with lower latency may be assigned lower weights, making them more favorable for time-sensitive traffic.
- **Bandwidth:** Links with higher bandwidth may also receive lower weights, prioritizing them for data-heavy flows.

2.0.3 Shortest Path Calculation

Establishing Initial Shortest Paths

To ensure efficient network routing, the initial step is to establish the shortest paths for all potential traffic flows. This is done using traditional shortest-path algorithms that determine the most efficient route between two nodes based on criteria such as distance or link cost. Dijkstra's Algorithm is often preferred due to its scalability and efficiency, particularly in complex networks where link weights are based on metrics like latency or bandwidth. However, in cases where the network's cost structure includes negative weights, the Bellman-Ford Algorithm is necessary, despite its higher computational complexity, due to its ability to handle such conditions.

Assigning Link Weights

Regardless of the algorithm used, computing these paths requires assigning specific weights to each link in the network. These weights represent the "cost" of traversing the link and can be based on several factors.

- **Hop Count:** The number of intermediate devices the data passes through. A lower hop count indicates a more direct path, and each hop can be assigned a weight of 1, favoring routes with fewer hops.
- **Latency:** The time required for data to traverse a link. Links with lower latency are assigned lower weights, making them more attractive for time-sensitive traffic.
- **Bandwidth:** Links with greater available bandwidth are often assigned lower weights, encouraging their use for data-heavy traffic.

Updating Routing Tables

Once the shortest paths are calculated for each node pair in the network, these paths are used to update the routing tables of network devices. Each entry in a routing table typically includes:

- **Destination Network:** The final destination that the traffic is intended for.
- **Next-Hop Address:** The address of the next device in the path towards the destination.
- **Interface:** The specific interface through which the data should be sent to reach the next hop.

The SDN controller automatically updates these routing tables to reflect the newly computed shortest paths, ensuring that traffic is routed efficiently across the network.

2.0.4 Monitor Traffic and Network Conditions

The iterative process where the algorithm begins by continuously monitoring the network's links. The controller gathers the following metrics for each link:

- Current Link Utilization (L)
- Available Bandwidth (B)
- Packet Loss Rates (P)

Additionally, the traditional cost (C) calculated during the previous step is also known. Using these metrics, the overall cost (F) for each path is calculated as:

$$F = w_1 \cdot e^{L} + w_2 \cdot e^{\frac{1}{B}} + w_3 \cdot e^{P} + w_4 \cdot C$$

where:

- w_1, w_2, w_3, and w_4 are weights assigned to each parameter.
- These weights are fixed and must be determined before implementation.

.

2.0.5 Load Distribution and Balancing

After calculating the overall cost F for each potential path, the controller selects the path with the lowest cost to route traffic. By incorporating link utilization, available bandwidth, and packet loss rate into the cost calculation, we achieve a balanced distribution of network traffic. This ensures that traffic is intelligently spread across the network, avoiding congestion.

For instance, if a particular path has a high link utilization, the term $w_1 \cdot e^L$ will increase, raising the overall cost F for that path, thereby discouraging its selection. The exponential function is chosen to amplify the impact of any unfavorable conditions on a path, ensuring that even a single adverse factor significantly influences the route's cost. In contrast, a linear approach would not penalize such conditions as effectively.

2.0.6 Implementation of Forwarding Rules

To enforce the chosen traffic distribution, the controller updates the forwarding rules on the network devices. This step involves modifying the forwarding tables based on the selected paths. The controller makes these decisions on a per-flow basis, leveraging the five-tuple (source IP, destination IP, source port, destination port, and protocol) to establish precise routing paths for different types of traffic.

This level of granularity allows the Software-Defined Networking (SDN) controller to tailor the traffic flow according to specific requirements, such as Quality of Service (QoS). Once the optimal paths are identified, the controller propagates these decisions by installing the corresponding flow entries on the network devices. Protocols like OpenFlow facilitate this process by enabling the controller to communicate directly with the devices. During implementation, it is critical to maintain network consistency and ensure that performance benchmarks are met.

2.0.7 Continuous Path Optimization

The network's dynamic nature requires continuous monitoring and adjustment of traffic paths. The controller must adapt to changing conditions, such as fluctuations in traffic or hardware failures. When a link or router fails, the controller quickly recalculates the costs and redirects traffic along alternative paths. This real-time adjustment capability ensures that the network remains resilient and that traffic continues to flow efficiently despite disruptions. The controller's ability to respond swiftly to such events is crucial for maintaining optimal network performance.

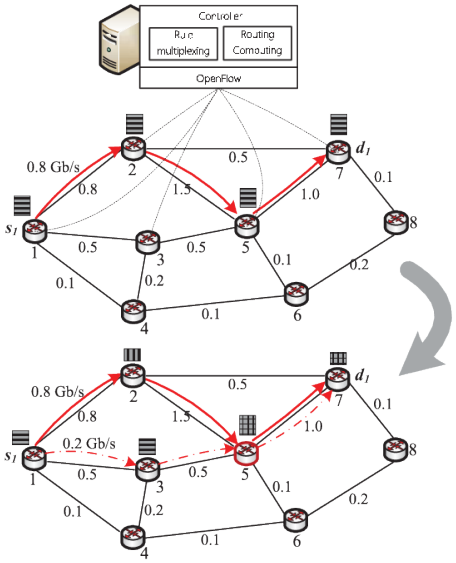


Figure 4: Path recalculation

3 Security

Security in an SDN network is crucial because it centralizes control, making the controller a high-value target for attacks that could disrupt the entire network. Ensuring robust security measures prevents unauthorized access and manipulation of network flows, safeguarding both data and infrastructure.

A key security objective in SDN is to safeguard communication between the control plane and the data plane, as well as between the control plane and the application plane. This can be accomplished by deploying robust authentication and encryption mechanisms.

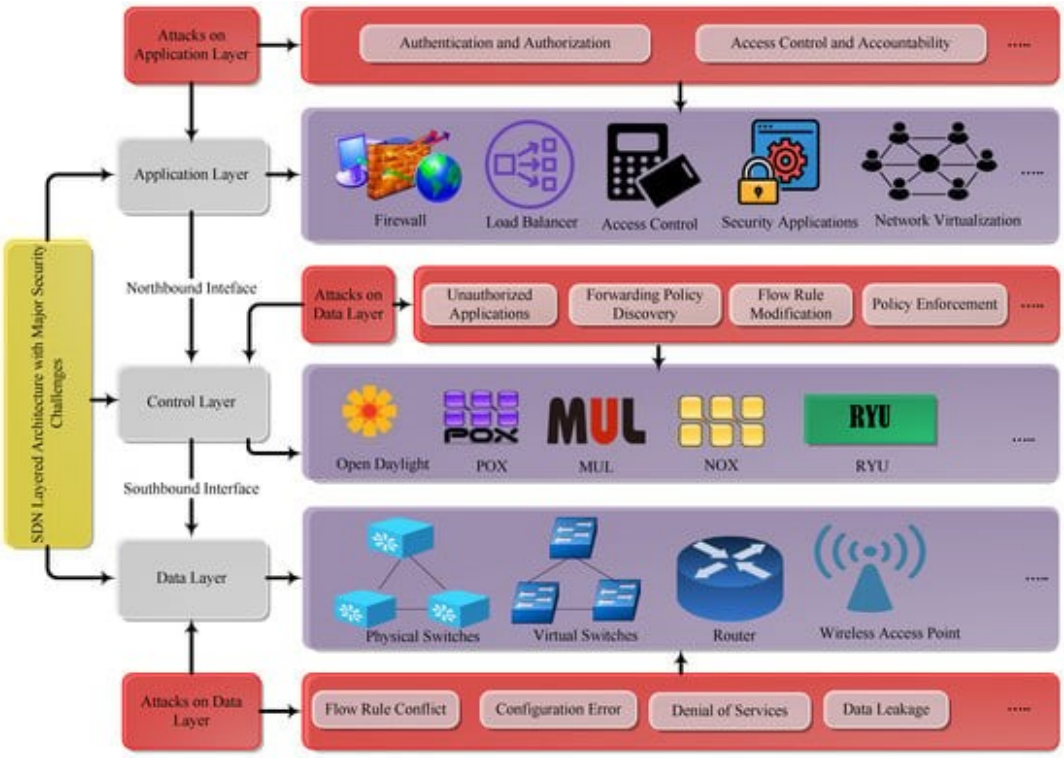


Figure 5: SDN Architecture with major security attacks

Authentication and Encryption are essential to making an SDN network secure.

3.1 Authentication

- **What It Is:** SDN Authentication involves verifying the identities of devices, users, and applications within the SDN network.
- **Purpose:** The goal of authentication is to ensure that only trusted and authorized entities can access and control the SDN network.

3.1.1 Device Authentication

Method:

- Deploy X.509 certificates on all SDN controllers, switches, and other network elements.
- Implement certificate-based mutual authentication during the initialization of communication between controllers and switches.

- Configure 802.1X on SDN switches to enforce port-based access control for devices attempting to connect to the network.

How It Works in Our Network:

- X.509 certificates establish a trusted chain of communication by verifying the identity of devices within the SDN network. When a switch or controller attempts to connect, it presents its certificate, which is validated against a certificate authority (CA) to confirm its legitimacy.
- 802.1X ensures that only authenticated devices can connect to SDN switches. This prevents unauthorized devices from accessing the network, adding a layer of security at the physical access level.

3.1.2 User Authentication

Method:

- Integrate the SDN controller with a centralized authentication system - TACACS+.
- Enable Multi-Factor Authentication (MFA) for all administrative users managing the SDN network.

How It Works in Our Network:

- TACACS+ provides centralized control over user access to the SDN controller and related systems. When a user attempts to log in, their credentials are validated against the centralized system, ensuring that only authorized users can make changes to the network configuration.
- MFA adds an extra security layer by requiring users to authenticate through a secondary method, such as a one-time code sent to a mobile device, reducing the risk of unauthorized access even if passwords are compromised.

3.1.3 API Authentication

Method:

- Generate and assign unique API keys or OAuth tokens for each application or script that interacts with the SDN controller's APIs.
- Configure the SDN controller to validate these keys or tokens during API calls.

How It Works in Our Network:

- API keys or OAuth tokens ensure that only pre-approved applications and scripts can interact with the SDN controller. When an API call is made, the SDN controller checks the token or key, verifying that the request is coming from a legitimate source. This prevents unauthorized applications from gaining control over network resources or data.

3.2 Encryption

What It Is: SDN Encryption is the practice of securing data within the SDN network, both while it's in transit (moving between network elements) and at rest (stored in devices or databases).

Purpose: The goal is to protect data from being intercepted or accessed by unauthorized parties.

3.2.1 Data Encryption In Transit

Method:

- Configure SSL/TLS encryption on all communication links between SDN controllers, switches, and other network components.

- Alternatively, implement IPSec tunnels for securing communication across the network where end-to-end encryption is needed.

How It Works in Our Network:

- SSL/TLS ensures that data exchanged between SDN controllers and switches is encrypted, preventing eavesdropping or tampering during transmission. This is critical for maintaining the integrity of control messages and sensitive data within the network.
- IPSec provides an alternative solution where full encryption of data packets is required across the network, especially for links that connect remote or less-secure segments of the SDN network. IPSec ensures confidentiality, integrity, and authentication of data in transit.

3.2.2 Data Encryption At Rest

Method:

- Encrypt the storage drives of all SDN devices, including controllers and switches, where configuration files, logs, and sensitive data are stored.
- Encrypt databases that store network topology information and user credentials.

How It Works in Our Network:

- Encryption at rest protects sensitive data stored within SDN devices, ensuring that even if a device is compromised, the data remains unreadable without the encryption keys. This secures configuration files, logs, and other critical data against unauthorized access.
- Encrypting databases ensures that vital information such as network topology details and user credentials cannot be accessed by unauthorized parties, even if the database is breached. This adds an additional layer of protection for stored data within the network.

4 Isolation

Isolation refers to the practice of segregating different segments of the network to prevent unauthorized access and ensure that network traffic from one segment does not interfere with or access other segments.

This is crucial for maintaining security and operational integrity, particularly in environments where sensitive data or critical applications are involved.

To effectively ensure network isolation in our SDN environment, **segmenting the network** through VLANs and VRFs will provide robust isolation by logically dividing traffic and creating separate routing tables for different segments.

4.1 Implement VLANs (Virtual Local Area Networks):

- **Create VLANs:** Define and configure VLANs for different types of traffic or departments within the SDN network. For example, create separate VLANs for management traffic, user data, and guest access.
- **Configure SDN Switches:** Set up SDN switches to tag and handle traffic according to VLAN assignments. Ensure that VLAN tags are correctly applied to packets as they traverse the network.
- **Apply VLAN Policies:** Enforce policies that restrict communication between VLANs. For instance, ensure that devices in one VLAN cannot communicate with devices in another VLAN unless explicitly permitted.

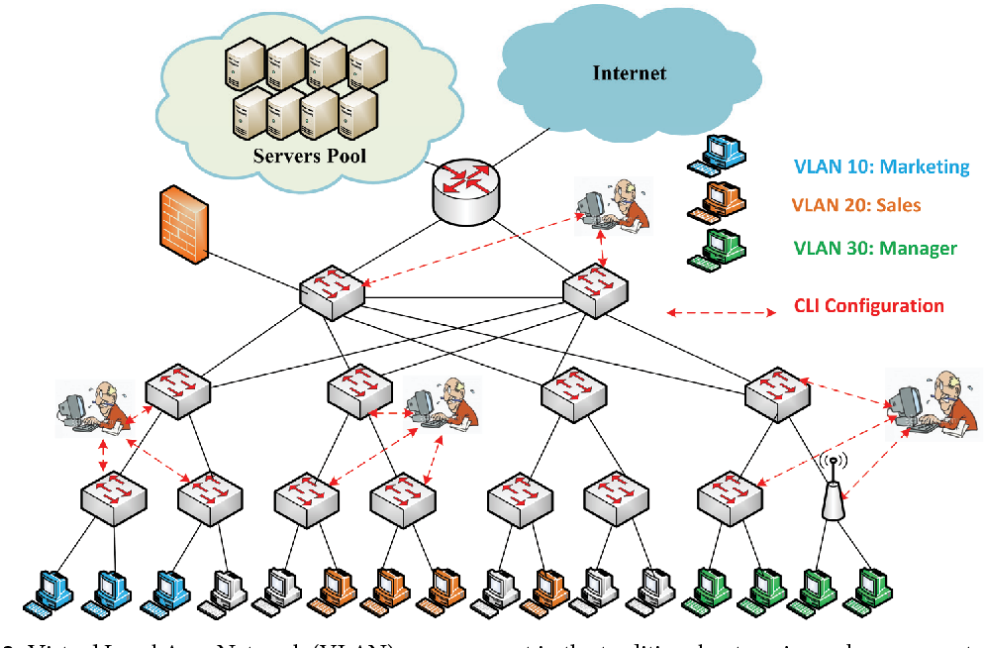


Figure 6: VLANs

4.2 Use VRFs (Virtual Routing and Forwarding):

- **Configure VRFs:** On routers and switches, set up VRFs to create multiple virtual routing tables. This allows for separate routing domains, enhancing isolation between different network segments.
- **Associate VLANs with VRFs:** Link VLANs to specific VRFs to ensure that traffic is routed according to its VLAN context, maintaining isolation between different traffic types or departments.

4.2.1 How It Works in Our Network:

- **VLANs:** VLANs logically divide the network into distinct segments, each isolated from the others. By tagging traffic with VLAN IDs, SDN switches can manage and direct traffic within its VLAN, ensuring that traffic intended for one VLAN does not interfere with or access another VLAN's traffic.
- **VRFs:** VRFs further enhance isolation by creating separate routing tables for different segments of the network. This means that each VRF handles routing independently, ensuring that traffic within one VRF does not impact or mix with traffic in another VRF.

5 Periodic re-evaluation

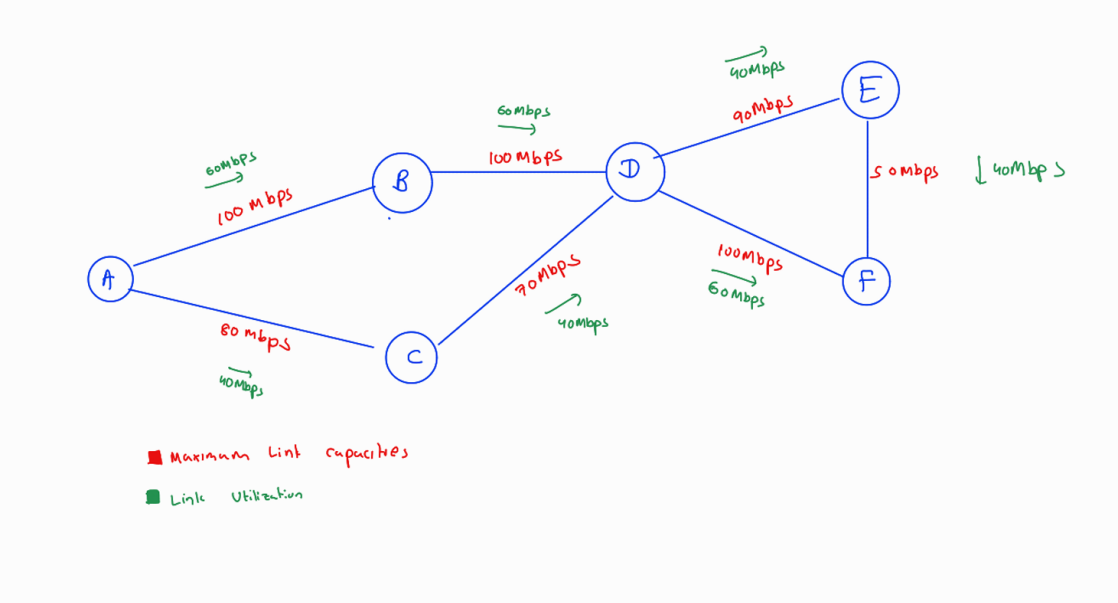
The purpose of periodic re-evaluation in our network is to continuously assess and adjust network performance to adapt to dynamic changes. This process ensures optimal resource utilization, prevents congestion, and maintains quality of service and security by regularly updating routing paths, policies, and configurations based on real-time data and performance metrics.

1. **STEP - Continuous Monitoring:**
   - **Real-time Metrics:** Monitor key metrics such as link utilization, latency, packet loss, jitter, and error rates continuously. These metrics provide immediate insights into network performance and help identify any issues promptly.
   - **Periodic Metrics:** Collect throughput and error rates at less frequent intervals. This data helps in understanding overall network health and performance trends over time.
2. **STEP - Data Analysis:**
   - **Trend Analysis:** Analyze the collected metrics to identify trends and anomalies, such as sudden increases in link utilization or unexpected traffic patterns.
   - **Anomaly Detection:** Implement anomaly detection algorithms to spot unusual traffic patterns that may indicate potential security threats or network issues.
3. **STEP - Routing Adjustment:**
   - **Re-calibration:** Based on the analysis, adjust routing paths, traffic policies, and network configurations to optimize performance and prevent congestion.
   - **Load Balancing:** Re-distribute traffic across multiple paths if necessary to prevent overloading any single link.
4. **STEP - QoS and Bandwidth Management:**
   - **QoS Adjustment:** Re-prioritize flows to ensure critical applications receive adequate bandwidth and low latency paths.
   - **Bandwidth Allocation:** Adjust bandwidth reservations or allocations based on current usage trends to better manage network resources.
5. **STEP - Controller Updates:**
   - **Push Updates:** Push updated forwarding rules, QoS policies, and security configurations to network devices based on re-calibrated decisions.
   - **Feedback Loops:** Implement feedback loops to monitor the impact of changes and ensure they have the desired effect.
6. **STEP - Deployment and Validation:**
   - **Phased Deployment:** Deploy changes in phases to minimize risk, and have a rollback procedure in place for unintended negative consequences.
   - **Impact Monitoring:** Closely monitor key metrics after deployment to validate that the changes have resolved issues and did not introduce new problems or vulnerabilities.

6 Example Scenario

Consider a network with six nodes: A, B, C, D, E, and F. The network has multiple paths available for routing traffic between nodes.

Network Structure



- **Traffic Demand :** There is a traffic demand of 100 Mbps from Node A to Node F.

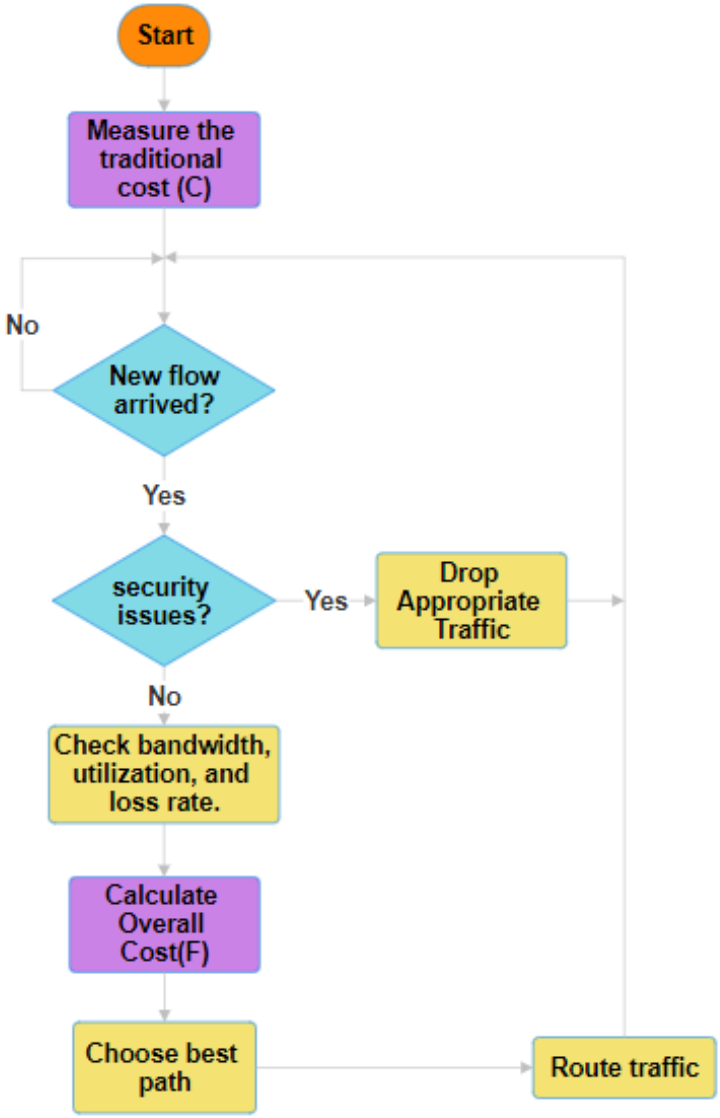
- **Traditional Shortest Path :**The traditional shortest path based on the highest capacity links would be A → B → D → F.

- **Network Utilization Monitoring :** The network controller monitors the traffic and notices the following:
  - The links A → B, B → D, and D → F would reach 80% utilization if the entire 100 Mbps traffic demand were routed through the path A → B → D → F.
- **Dynamic Routing Decision :** To avoid overutilization and potential congestion, the controller decides to split the traffic. Specifically:
  - 60 Mbps continues to flow through the shortest path A → B → D → F.
  - The remaining 40 Mbps is rerouted through an alternative path A → C → D → E → F.

  This decision helps balance the load and prevents any single link from becoming a bottleneck.
- **Outcomes :**
  - **Load Balancing:** By routing part of the traffic through the alternative path A → C → D → E → F, the controller reduces the load on the links A → B, B → D, and D → F.
  - **Optimized Network Performance:** This dynamic adjustment helps ensure that the network operates efficiently, minimizing the risk of congestion and maintaining optimal performance for all traffic moving from Node A to Node F.

- **Key Considerations :**
  - **Real-Time Monitoring:** The network continuously monitors the utilization of each link to detect potential congestion before it becomes an issue.
  - **Adaptive Traffic Management:** By dynamically splitting traffic based on real-time conditions, the network can better handle varying loads and ensure that critical traffic is not delayed.
  - **Resource Utilization:** This strategy optimizes the usage of the network's available capacity, ensuring a smooth flow of data even during high-demand periods.

7 Summary of Algorithm



8 Quality of Service (QoS)

8.1 Need Analysis

To ensure Quality of Service (QoS), the proposed algorithm dynamically selects routing paths based on the type of application traffic. This approach contrasts with traditional routing methods, which generally focus on optimizing overall network performance without considering application-specific requirements. Different applications have distinct QoS needs, and addressing these is crucial for optimizing user experience. Examples of application-specific QoS requirements include:

- **Online Gaming:** Requires low latency and minimal jitter for a responsive gaming experience.
- **Real-Time Voice Communication (e.g., VoIP):** Demands low latency to avoid delays and echo during conversations.
- **File Transfer:** Needs low packet loss and high bandwidth to ensure efficient and reliable data transfer.
- **Video Streaming:** Requires high bandwidth for smooth video delivery; low latency is less critical due to buffering.
- **Video Conferencing:** Requires low latency and low jitter for seamless real-time interaction.
- **Email and Text Communication:** Needs moderate bandwidth and low packet loss for timely and reliable message delivery.

The proposed algorithm modifies existing traffic engineering techniques to adjust routing decisions based on application type, ensuring high QoS for diverse applications.

8.2 Algorithm

The key modification in this QoS routing algorithm is the dynamic adjustment of routing weights based on application type. This approach allows prioritizing traffic according to specific application requirements, optimizing the quality of experience. The following methods are used to identify application types for traffic flows:

8.3 Port-Based Classification

Initially, traffic is classified based on port numbers associated with specific applications. This method uses the 5-tuple flow identifier:

- Source IP
- Source Port
- Destination IP
- Destination Port
- Protocol

Port-based classification provides a straightforward method for identifying application types but may have limitations in distinguishing between applications that use the same ports.

8.4 Behavioral Flow Analysis

Behavioral Flow Analysis monitors and analyzes flow-level characteristics to infer application types. It assumes that different applications exhibit unique traffic patterns, such as burstiness and periodicity.

Process of Behavioral Flow Analysis:

1. **Traffic Flow Monitoring:** Collects metrics like packet size distribution, flow duration, and inter-arrival times.
2. **Pattern Recognition:** Matches observed traffic patterns against predefined profiles of application behaviors.
3. **Adaptive Profiling:** Continuously updates profiles based on real-time data to adapt to new application behaviors.

Advantages:

- **Low Resource Overhead:** Requires less computational power than methods like DPI (Deep Packet Inspection).
- **Effective with Encrypted Traffic:** Does not need access to payloads, making it suitable for encrypted traffic.
- **Scalability:** Handles large volumes of traffic with minimal impact on network performance.

Disadvantages:

- **Initial Calibration Required:** Needs accurate profiling of application behaviors, which may require manual configuration.
- **Latency Sensitivity:** Real-time classification requires efficient processing to avoid delays during high traffic loads.

Implementation Strategy:

1. **Profile Database Creation:** Establish profiles of common application traffic patterns.
2. **Real-Time Flow Monitoring:** Deploy sensors to continuously monitor traffic flows.
3. **Matching Engine:** Compare real-time data with stored profiles to determine application types.
4. **Dynamic Adaptation:** Update profiles dynamically based on observed changes in traffic patterns.

8.5 Machine Learning Approach

Once sufficient data is collected from port-based classification and Behavioral Flow Analysis, a Machine Learning (ML) model will be developed and trained. This model will use traffic characteristics to recognize application patterns and adapt to network changes.

Advantages:

- **Adaptability:** Learns to recognize new applications and adapts to traffic pattern changes.
- **Resource Efficiency:** Less computationally intensive once trained.
- **Encrypted Traffic Handling:** Focuses on traffic behavior rather than payload content.

However this approach may struggle with new or unknown applications not included in the training data. In such cases, Deep Packet Inspection (DPI) can be employed to analyze the contents of the packets more thoroughly, helping to identify and classify the unknown applications.

Development Plan:

1. **Data Collection:** Gather data from port-based classification and Behavioral Flow Analysis.
2. **Model Training:** Develop and train an ML model on the collected data.
3. **Continuous Training:** Regularly update and retrain the model to maintain accuracy and adapt to new traffic patterns.
4. **Anomaly Detection:** Use DPI as a fallback method for detecting anomalies if necessary.

8.6 Implementation Details

- **Weight Management:** Store weights for different application types in a network controller database. These weights represent traffic priorities and resource allocations.
- **Routing Decisions:** Retrieve weights based on identified application types and use them to determine optimal routing paths. This ensures high-priority applications receive necessary resources even during congestion.
- **Controller Integration:** Integrate the QoS routing algorithm with a centralized network controller (e.g., an SDN controller) to manage dynamic routing decisions across the network.

Algorithm 1 QoS Routing Algorithm

```
procedure QoSRouting(packet)
    flow ← Extract_Flow_From_Packet(packet)
    APPLICATION_TYPE ← Classify(packet, flow)
    WEIGHTS ← NetworkController.GET_QOS_WEIGHTS(APPLICATION_TYPE)
    return DetermineRoutingPath(WEIGHTS)
end procedure
procedure Classify(packet, flow)
    if KnownPort(packet) then
        return APPLICATION_TYPE_BY_PORT
    else if DataSufficient() then
        return PredictWithML(flow)
    else
        return BehavioralAnalysis(flow)
    end if
end procedure
```

Figure 7: QoS Algorithm with SDN

9 References

1. **How can you secure your software-defined network (SDN)?** - https://www.linkedin.com/advice/1/how-can-you-secure-your-software-defined-network-iyile.
2. **M. Zhang, W. Li, H. Xu, et al. (2022). A Survey on Network Traffic Engineering in Software Defined Networks: Challenges and Solutions. IEEE.** https://ieeexplore.ieee.org/document/9667584.
3. **I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," Computer Networks, vol. 71, pp. 1-30, Oct. 2014. doi: 10.1016/j.comnet.2014.06.002.** https://www.sciencedirect.com/science/article/abs/pii/S1389128614002254.
4. **Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review** - https://www.mdpi.com/2388762.
5. **Quality of Service (QoS) in Software Defined Networking (SDN): A Survey** - Murat Karakusa and Arjan Durresia, *Computer Networks*, vol. 107, pp. 61–83, 2016. Available: https://www.sciencedirect.com/science/article/am/pii/S1084804516303186.