Department of Electronics & Telecommunication Engineering
University of Moratuwa

EN2150 - Communication Network Engineering

SDN Algorithm Development

Group Assignment

Packet Pioneers

Dilshan N.L.	210129P
Dimagi D.H.P	210131N
Dissanayaka D.M.P.C	210140P
Gunawardana W.N.M	210199D

Date - 2024.08.30

Contents

1 Introduction

The evolution of networking technologies and the growing demand for high-quality, reliable services necessitate the development of sophisticated routing algorithms. Traditional routing protocols focus on the shortest path, often leading to congestion and inefficient use of network resources. In contrast, Software-Defined Networking (SDN) provides a flexible framework for dynamic routing, allowing for real-time adjustments based on current network conditions.

This report presents routing algorithms for Traffic Engineering (TE) and Quality of Service (QoS) in an SDN environment. These algorithms aim to optimize network resource utilization and enhance the user experience by leveraging the centralized control and programmability offered by SDN.

2 Problem Statement

In modern data center networks (DCNs), efficient management of network traffic is crucial to prevent congestion, ensure high performance, and maximize the utilization of network resources. Traditional routing protocols, such as OSPF (Open Shortest Path First) or RIP (Routing Information Protocol), typically route traffic based on the shortest path between a source and a destination. While this approach is straightforward, it often leads to over-utilization of certain links, causing congestion, increased latency, packet loss, and suboptimal performance. This is particularly problematic in environments where large volumes of data traffic are common, such as in cloud data centers, where services like streaming, big data analytics, and virtualized network functions are heavily dependent on reliable network performance.

2.1 Challenges:

1. **Congestion on Shortest Paths**: Traditional routing algorithms prioritize the shortest path, which can lead to excessive traffic on certain links while other network paths remain underutilized. This uneven distribution of traffic results in congestion, increasing the likelihood of packet loss, higher latency, and jitter, especially during peak usage times.

2. **Inefficient Use of Network Resources**: When traffic is not evenly distributed across all available links, some network links may be underutilized, leading to wasted bandwidth and inefficient use of the infrastructure. This is a significant concern for network operators who have invested heavily in the network infrastructure and expect a good return on investment (ROI).

3. **Dynamic and Real-Time Requirements**: Modern network environments are highly dynamic, with traffic patterns that can change rapidly. Static routing decisions made without real-time network data may quickly become suboptimal, necessitating a more dynamic approach to routing that can adapt to changing conditions.

4. **Complexity in Managing Multiple Paths**: Determining the best path among multiple possible routes requires sophisticated algorithms that can account for various factors, such as current traffic load, link capacity, and potential congestion points. This complexity is further increased when the network must handle multiple flows simultaneously, each with different priority levels and QoS requirements.

3 Objective

Develop a routing algorithm for traffic control over a network using Software-Defined Networking (SDN). The algorithm must handle two primary scenarios:

1. **Traffic Engineering**: Efficiently distribute traffic across multiple paths to ensure optimal utilization of all network links.

2. **Quality of Service (QoS)**: Provide the best possible user experience by prioritizing different traffic types according to their specific requirements.

4 Traffic Engineering (TE) Algorithm

4.1 Parameters to Consider

1. **Bandwidth (B)**: Available bandwidth on the link.
2. **Latency (L)**: Time taken for a packet to travel across the link.
3. **Link Utilization (U)**: The current usage level of the link.
4. **Packet Loss Rate (PLR)**: The fraction of packets lost on the link.
5. **Jitter (J)**: Variation in packet arrival times.
6. **Distance (D)**: Physical distance or hop count between nodes.

4.2 Path Cost Calculation

The path cost between two nodes i and j is calculated using the following formula:

$$C_{ij} = w_B \cdot \left(\frac{1}{B_{ij}}\right)^{p_B} + w_L \cdot (L_{ij})^{p_L} + w_U \cdot (U_{ij})^{p_U} + w_{PLR} \cdot (PLR_{ij})^{p_{PLR}} + w_J \cdot (J_{ij})^{p_J} + w_D \cdot (D_{ij})^{p_D}$$

Where:

- $w_B, w_L, w_U, w_{PLR}, w_J, w_D$ are the weights assigned to each parameter.
- $p_B, p_L, p_U, p_{PLR}, p_J, p_D$ are the exponents applied to each parameter to reflect their importance.

Note: The weights and exponents for each parameter vary depending on the application requirements:

- For applications like **video streaming or video editing**, where high bandwidth and efficient link utilization are critical, the weight and exponent for B (bandwidth) and U (link utilization) should be higher.
- For **real-time applications** like **VoIP or video conferencing**, where low latency and low jitter are essential, the weight and exponent for L (latency) and J (jitter) should be prioritized.
- The importance of D (distance) might increase in scenarios where the number of hops directly impacts performance, such as in certain security-sensitive applications.
- PLR (packet loss rate) becomes critical in environments where data integrity is paramount, like financial transactions or critical control systems.

By adjusting the weights $w_B, w_L, w_U, w_{PLR}, w_J, w_D$ and exponents $p_B, p_L, p_U, p_{PLR}, p_J, p_D$, the algorithm can be tailored to meet the specific needs of various applications, ensuring optimal routing decisions.

4.3 Algorithm Steps

4.3.1 Data Collection

The SDN controller continuously gathers data on the parameters listed above from all network routers and switches. Protocols like OpenFlow can be used to request and aggregate this data.

- Event-Driven Updates: In OpenFlow, these updates occur in response to specific events, such as new packets arriving or changes in flow entries. These messages are typically sent with low latency, generally within **1 to 10 milliseconds**.
- Periodic Updates: For ongoing statistics or flow table information, OpenFlow allows for periodic updates. The frequency of these updates can be configured, with typical intervals ranging from **100 milliseconds to 1 second**.

4.3.2 Parameter Calculation

Each router calculates the following parameters based on real-time data:

- **Bandwidth** (B): $B = \text{Link Capacity} - \text{Current Throughput}$
- **Latency** (L): $L = \frac{\text{RTT}}{2}$ (measured using ping or similar methods)
- **Link Utilization** (U): $U = \frac{\text{Current Throughput}}{\text{Link Capacity}}$
- **Packet Loss Rate** (PLR): $PLR = \frac{\text{Number of Lost Packets}}{\text{Total Number of Sent Packets}}$
- **Jitter** (J): $J = \frac{1}{n-1}\sum_{i=1}^{n-1} |(T_{i+1} - T_i) - (T_i - T_{i-1})|$
- **Distance** (D): Measured either as the hop count or physical distance between nodes.

4.3.3 Flow Chart

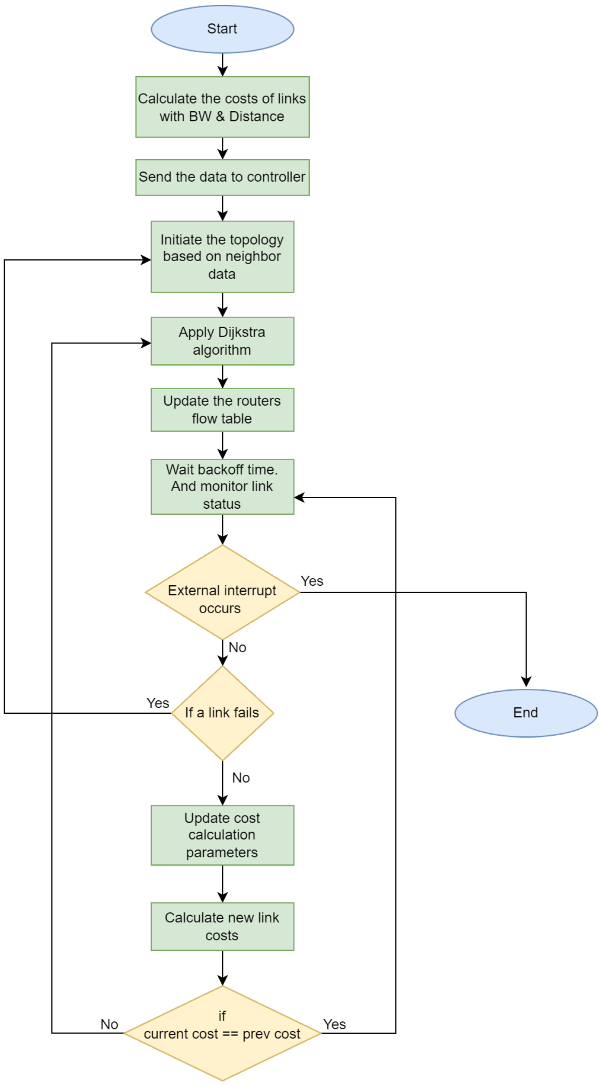

Figure 1: Flow Chart

4.3.4 Justification for Each Step of the Algorithm

1. **Begin:** Marks the start of the algorithm, initializing the process.

Algorithm 1 Link Cost Calculation and Routing Update

```
 1: begin
 2: Calculate link costs with bandwidth and distance
 3: Send data to the controller
 4: Initiate topology based on neighbor data
 5: Apply Dijkstra's algorithm
 6: Update flow tables
 7: Monitor link status
 8: if external interrupt occurs then
 9:    end
10: else
11:    if link failure detected then
12:       Go to line 3
13:    else
14:       Update cost parameters
15:       Calculate new link costs and send to controller
16:       if costs unchanged then
17:          Go to line 7
18:       else
19:          Go to line 5
20:       end if
21:    end if
22: end if
```

2. **Calculate link costs with bandwidth and distance:** Determines the cost of each link by considering bandwidth and distance to reflect network performance. Only distance and bandwidth are considered at the beginning since the network is being initiated.

3. **Send data to the controller:** Communicates the calculated link costs to the SDN controller for centralized decision-making.

4. **Initiate topology based on neighbor data:** Builds the network topology using neighbor data, forming the basis for routing decisions.

5. **Apply Dijkstra's algorithm:** Computes the shortest paths from a source to all other nodes in the network, ensuring optimal routing.

6. **Update flow tables:** Updates the flow tables in each router with the newly computed paths, ensuring accurate data packet routing.

7. **Monitor link status:** Continuously monitors link status to detect changes or failures, maintaining network reliability.

8. **If external interrupt occurs, end:** Allows the algorithm to safely terminate in response to external interrupts.

9. **If link failure detected, go to line 3:** Adapts to link failures by recalculating link new topology, costs and routing paths.

10. **Update cost parameters:** Adjusts cost calculation parameters to reflect changing network conditions and requirements.

11. **Calculate new link costs and send to controller:** Recalculates link costs with updated parameters and sends them to the controller.

12. **If costs unchanged, go to line 7:** Skips unnecessary steps if recalculated costs are unchanged, directly monitoring the network.

13. **Else, go to line 5:** Recalculates routing paths if the new link costs differ, ensuring the network adapts to changes effectively.

4.4 Link Cost Function Adjustments

The composite link cost function C_{ij} is dynamically adjusted to reflect the importance of different QoS metrics for the identified traffic type. Below are examples of how the function is adjusted for different types of traffic, including specific values for the parameters (weights and power factors). In this example, we have demonstrated how to maintain **three different routing tables** for three distinct types of traffic and how the lowest-cost paths change based on the type of traffic. (Please note that the traffic values for each case are not compared, as this is just an example demonstration.)

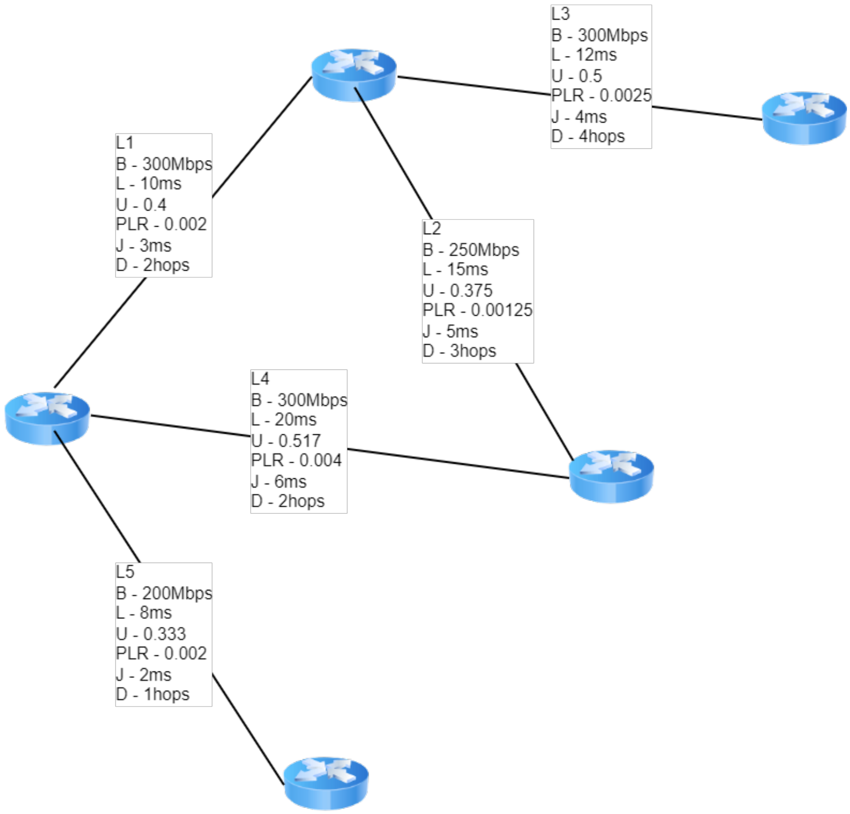

Figure 2: Eaxmple Network Configuration

4.4.1 VoIP (QOS=0)

Prioritize latency and jitter to ensure clear and uninterrupted voice communication.

$$C_{ij} = 1000 \cdot \left(\frac{1}{B_{ij}}\right)^{1.0} + 0.4 \cdot (L_{ij})^{2.0} + 1 \cdot (U_{ij})^{1.0} + 0.2 \cdot (PLR_{ij})^{1.0} + 0.2 \cdot (J_{ij})^{2.0} + 0.5 \cdot (D_{ij})^{1.2}$$

Lowest Cost path calculation for QOS=0 case:

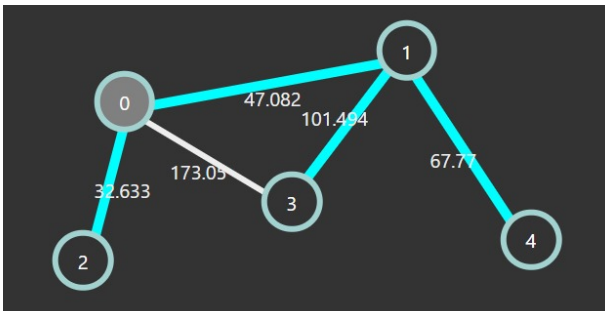

Figure 3: Example Network Topology for node 0

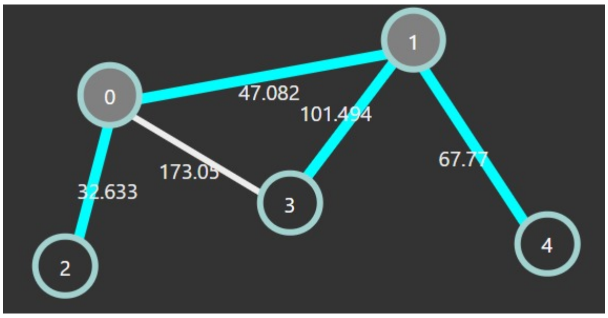

Figure 4: Example Network Topology for node 1

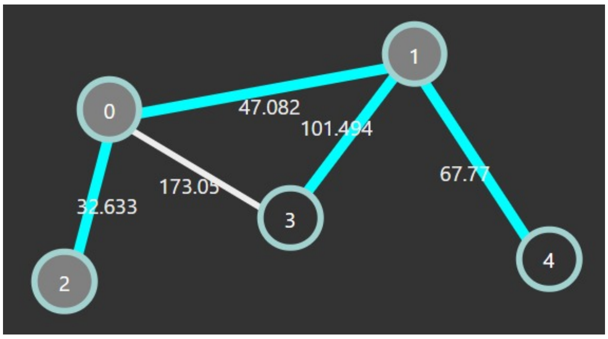

Figure 5: Example Network Topology for node 2

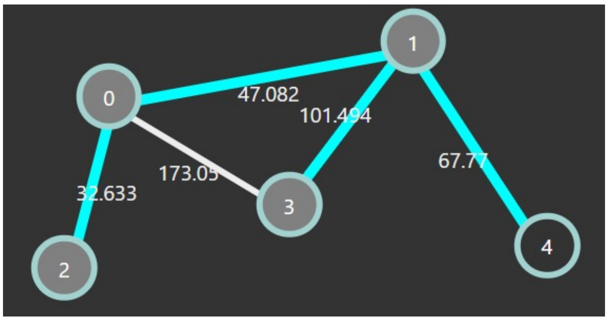

Figure 6: Example Network Topology for node 3

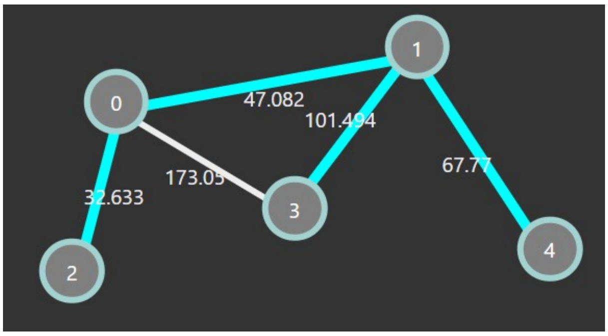

Figure 7: Example Network Topology for node 4

4.4.2 Video Streaming (QOS=1)

Prioritize bandwidth, jitter, and packet loss to ensure smooth video playback with minimal buffering.

$$C_{ij} = 4000 \cdot \left(\frac{1}{B_{ij}}\right)^{1.5} + 0.1 \cdot (L_{ij})^{1.0} + 1 \cdot (U_{ij})^{1.0} + 0.2 \cdot (PLR_{ij})^{1.2} + 0.2 \cdot (J_{ij})^{1.5} + 0.5 \cdot (D_{ij})^{1.2}$$

Lowest Cost path calculation for QOS=1 case:

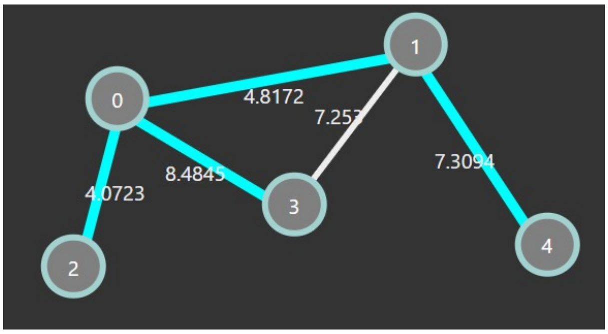

Figure 8: Example Network Topology for node 0

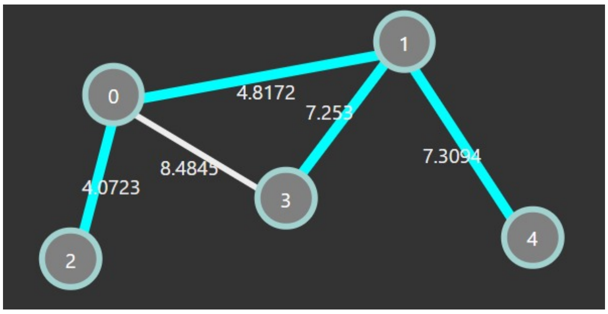

Figure 9: Example Network Topology for node 1

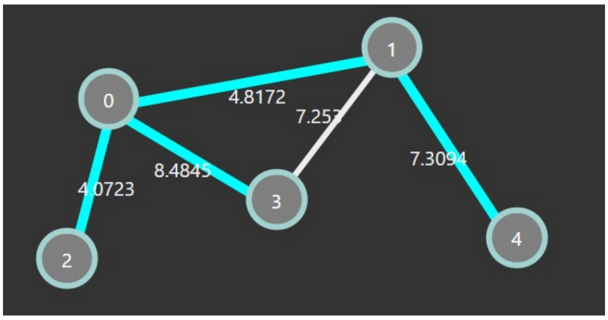

Figure 10: Example Network Topology for node 2

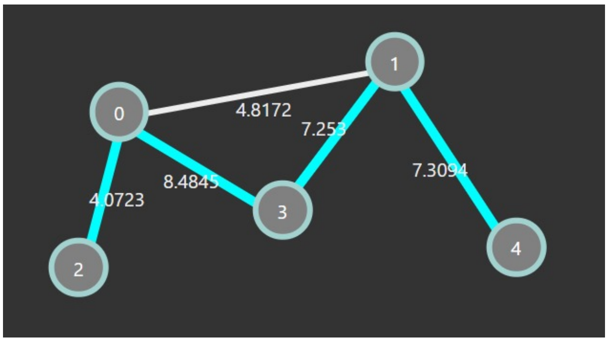

Figure 11: Example Network Topology for node 3

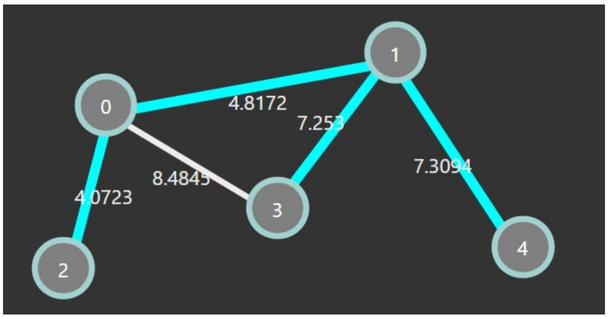

Figure 12: Example Network Topology for node 4

4.4.3 File Downloads (QOS=2)

Maximize bandwidth and minimize link utilization to ensure fast and efficient data transfer.

$$C_{ij} = 5000 \cdot \left(\frac{1}{B_{ij}}\right)^{2.0} + 0.1 \cdot (L_{ij})^{1.0} + 3 \cdot (U_{ij})^{1.5} + 0.05 \cdot (PLR_{ij})^{1.0} + 0.05 \cdot (J_{ij})^{1.0} + 0.5 \cdot (D_{ij})^{1.2}$$

Lowest Cost path calculation for QOS=2 case:

Figure 13: Example Network Topology for node 0

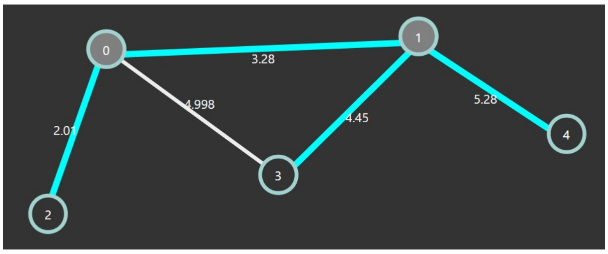

Figure 14: Example Network Topology for node 1

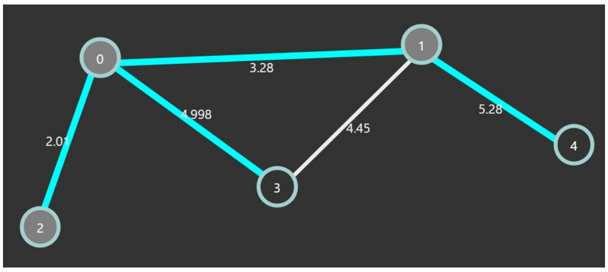

Figure 15: Example Network Topology for node 2

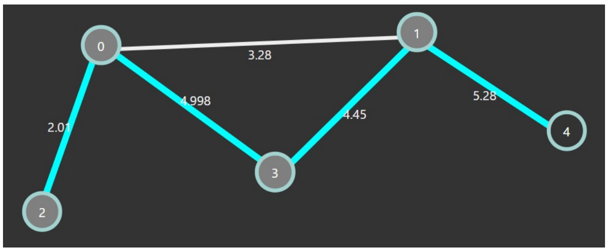

Figure 16: Example Network Topology for node 3

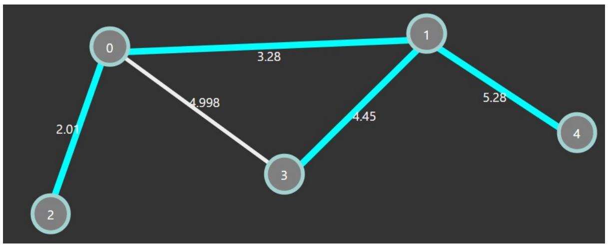

Figure 17: Example Network Topology for node 4

5 Quality of Service (QoS) Algorithm

5.1 Objective

The primary objective of the Quality of Service (QoS) algorithm is to ensure that network traffic is managed in a way that meets the specific requirements of different applications. This involves dynamically selecting routing paths that can satisfy various QoS criteria, such as low latency, high bandwidth, minimal packet loss, stable jitter, and optimal distance. The goal is to provide the best possible user experience, even in varying network conditions.

5.2 Challenges in QoS Routing

QoS routing faces two main challenges:

1. **Traffic Engineering**: Ensuring balanced traffic across the network to improve overall performance and avoid congestion.
2. **End-to-End Dynamics**: Identifying specific routes that meet the QoS requirements of applications like telephony over IP, which demand low delays, high throughput, and minimal packet loss.

Despite the availability of high-bandwidth links, issues like jitter during burst traffic remain, especially under conditions where traditional IP services treat all data flows equally. This leads to inefficiencies such as congestion and poor resource utilization. Real-time applications, in particular, suffer from packet drops and retransmissions, further degrading performance.

5.3 QoS Metrics and Composite Link Cost Function

To achieve the desired QoS, the RDSS algorithm evaluates multiple QoS metrics, including bandwidth, latency, link utilization, packet loss rate, jitter, and distance. These metrics are used in a composite link cost function that assigns a cost to each link in the network. The composite link cost function is central to the RDSS algorithm's ability to select paths that meet the QoS requirements of various applications.

Key Parameters for QoS:

- **Bandwidth (B)**: Available bandwidth on the link.
- **Latency (L)**: Delay experienced in the link.
- **Link Utilization (U)**: Current utilization of the link.
- **Packet Loss Rate (PLR)**: Fraction of packets lost on the link.
- **Jitter (J)**: Variation in packet arrival times.
- **Distance (D)**: Physical distance or hop count between nodes.

Composite Link Cost Function:

The composite cost of a link between nodes i and j is computed using the following formula:

$$C_{ij} = w_B \cdot \left(\frac{1}{B_{ij}}\right)^{p_B} + w_L \cdot (L_{ij})^{p_L} + w_U \cdot (U_{ij})^{p_U} + w_{PLR} \cdot (PLR_{ij})^{p_{PLR}} + w_J \cdot (J_{ij})^{p_J} + w_D \cdot (D_{ij})^{p_D}$$

Where:

- C_{ij} is the composite cost of the link between nodes i and j.
- $w_B, w_L, w_U, w_{PLR}, w_J, w_D$ are the weights assigned to bandwidth, latency, utilization, packet loss rate, jitter, and distance, respectively.
- $p_B, p_L, p_U, p_{PLR}, p_J, p_D$ are the power factors applied to each metric, allowing for non-linear scaling.

5.4 Relationship Between the Link Cost Function and QoS

The composite link cost function is crucial in ensuring Quality of Service (QoS) because it integrates multiple QoS parameters into a single metric that can be used to evaluate and select the optimal path for a given flow. Here's how each component of the link cost function relates to QoS:

Bandwidth (B):

Higher bandwidth is essential for applications that require large data transfers, such as video streaming or file downloads. The inverse relationship $\left(\frac{1}{B_{ij}}\right)$ ensures that links with higher bandwidth (and thus lower cost) are preferred, contributing to better QoS.

Latency (L):

Low latency is critical for real-time applications like VoIP and online gaming. By directly incorporating latency into the cost function, the algorithm ensures that paths with lower delay are prioritized, improving the quality of these time-sensitive applications.

Link Utilization (U):

Lower link utilization indicates less congestion, which can reduce the likelihood of packet delays and losses. Prioritizing paths with lower utilization helps maintain high performance and QoS, especially under heavy network load.

Packet Loss Rate (PLR):

High packet loss can severely degrade the quality of services such as video calls or streaming. Including packet loss in the cost function ensures that paths with minimal packet loss are selected, enhancing the reliability and quality of these services.

Jitter (J):

Jitter, or the variability in packet delay, can disrupt the smooth delivery of data in real-time applications. By factoring jitter into the cost function, the algorithm favors paths that provide more consistent delay, which is crucial for maintaining the quality of real-time communication.

Distance (D):

Shorter physical distance or fewer hops generally lead to lower latency and reduced risk of packet loss. Including distance in the cost function helps in selecting paths that are inherently more efficient and reliable, contributing to overall better QoS.

By combining these metrics into a single composite cost, the algorithm can evaluate all potential paths and select the one that best satisfies the QoS requirements for the specific application. This ensures that network resources are optimally used, and the end-user experience is enhanced.

5.5 How Link cost function dynamically adapts to different QoS scenarios

Packet Arrival

When a packet arrives at an SDN-enabled switch, the switch, instead of making routing decisions on its own, forwards the packet to the SDN controller. This is a standard behavior in SDN, where the control logic is centralized within the controller, which has a global view of the network.

Deep Packet Inspection (DPI)

Deep Packet Inspection (DPI) analyzes the contents of packet payloads to identify specific application signatures, enabling the SDN controller to classify the type of traffic accurately. By recognizing different protocols and signatures, DPI can distinguish between various types of network traffic, such as:

- **Video Streaming:** Detected through video streaming protocols within the packet data.
- **VoIP (Voice over IP):** Identified by locating VoIP-specific communication signatures.
- **File Transfers:** Recognized by detecting protocols and activities associated with file transfers.

DPI provides the SDN controller with a deep understanding of the traffic type, allowing it to make informed routing decisions that optimize the network based on the specific demands of each application.

Once the traffic is classified, the SDN controller applies the appropriate QoS scenario, adjusting the link cost function to suit the specific performance needs of the application. The weights and priorities in the cost function are adapted accordingly:

- **For Video Streaming:** The emphasis is placed on minimizing jitter and packet loss while ensuring adequate bandwidth. The link cost function prioritizes routes that maintain high quality for continuous data streams.
- **For VoIP:** Low latency and minimal jitter are critical. The link cost function favors paths that reduce delay and maintain smooth voice communication.
- **For File Transfers:** Bandwidth and link utilization take precedence, as bulk data transfers are less sensitive to latency and jitter. The cost function focuses on optimizing throughput and efficiency.

By aligning the routing decisions with the specific performance requirements of each type of traffic, the network can deliver superior service quality and efficiency across all applications.

QoS Decision

The composite link cost function, C_{ij}, is dynamically adjusted to reflect the importance of different QoS metrics based on the identified traffic type. Below are examples of how the function is adjusted for various types of traffic, including specific values for the parameters (weights and power factors).Units of those parameters are shown as below:

Bandwidth (Mbps)
Latency (ms)
Link Utilization (dimensionless or percentage)
Packet Loss Rate (**ppt, parts per thousand**)
Jitter (ms)
Distance (no.of hops)

VoIP

Objective: Prioritize latency and jitter to ensure clear and uninterrupted voice communication.

Link Cost Function:

$$C_{ij} = 1000 \cdot \left(\frac{1}{B_{ij}}\right)^{1.0} + 0.4 \cdot (L_{ij})^{2.0} + 1 \cdot (U_{ij})^{1.0} + 0.2 \cdot (PLR_{ij})^{1.0} + 0.2 \cdot (J_{ij})^{2.0} + 0.5 \cdot (D_{ij})^{1.2}$$

- **Bandwidth (1000, $p_B = 1.0$):** High priority with linear scaling to prevent queuing.
- **Latency (0.4, $p_L = 2.0$) & Jitter (0.2, $p_J = 2.0$):** High impact with squared scaling to minimize delays and variability.
- **Utilization (1, $p_U = 1.0$):** Moderate priority with linear scaling to avoid congestion.
- **Packet Loss (0.2, PPLR = 1.0):** Moderate concern with linear scaling for voice quality.
- **Distance (0.5, $p_D = 1.2$):** Prefers shorter paths with slight non-linear scaling to reduce latency.

Video Streaming

Objective: Prioritize bandwidth, jitter, and packet loss to ensure smooth video playback with minimal buffering.

Link Cost Function:

$$C_{ij} = 4000 \cdot \left(\frac{1}{B_{ij}}\right)^{1.5} + 0.1 \cdot (L_{ij})^{1.0} + 1 \cdot (U_{ij})^{1.0} + 0.2 \cdot (PLR_{ij})^{1.2} + 0.2 \cdot (J_{ij})^{1.5} + 0.5 \cdot (D_{ij})^{1.2}$$

- **Bandwidth (4000, $p_B = 1.5$):** Crucial with non-linear scaling to heavily penalize low bandwidth paths and prevent buffering.
- **Latency (0.1, $p_L = 1.0$):** Less critical with linear scaling.
- **Utilization (1, $p_U = 1.0$):** Avoids congestion with linear scaling.
- **Packet Loss (0.2, PPLR = 1.2) & Jitter (0.2, $p_J = 1.5$):** Moderately high priority with non-linear scaling to maintain video quality.
- **Distance (0.5, $p_D = 1.2$):** Prefers shorter, stable paths with slight non-linear scaling.

File Downloads

Objective: Maximize bandwidth and minimize link utilization to ensure fast and efficient data transfer.

Link Cost Function:

$$C_{ij} = 5000 \cdot \left(\frac{1}{B_{ij}}\right)^{2.0} + 0.1 \cdot (L_{ij})^{1.0} + 3 \cdot (U_{ij})^{1.5} + 0.05 \cdot (PLR_{ij})^{1.0} + 0.05 \cdot (J_{ij})^{1.0} + 0.5 \cdot (D_{ij})^{1.2}$$

- **Bandwidth (5000, $p_B = 2.0$):** Extremely high priority with squared scaling to favor high-capacity paths.
- **Latency (0.1, $p_L = 1.0$):** Low priority with linear scaling, tolerating delays.
- **Utilization (3, $p_U = 1.5$):** Highly weighted with non-linear scaling to avoid bottlenecks.
- **Packet Loss (0.05, PPLR = 1.0) & Jitter (0.05, $p_J = 1.0$):** Minimal impact with linear scaling.
- **Distance (0.5, $p_D = 1.2$):** Moderately considered with slight non-linear scaling, favoring shorter paths.

Flow Rule Deployment

The SDN controller deploys flow rules to the switches to enforce the selected QoS policy. These rules determine how packets are forwarded through the network. For instance:

- **Video Packets:** May be marked with a higher DSCP value (e.g., Assured Forwarding) to ensure they receive priority treatment throughout the network.
- **VoIP Packets:** Routed through paths with low latency and jitter, potentially using real-time protocol prioritization mechanisms.
- **File Download Packets:** Routed through paths that have higher available bandwidth, possibly using a best-effort service class.

Link Cost Function in Action

The flow rules reflect the path selected by the adjusted link cost function. For example, the selected path for video traffic will be the one that minimizes the composite cost considering the high weights on bandwidth, jitter, and packet loss.

These rules ensure that traffic is managed according to the QoS requirements identified during the classification and decision-making stages.

The SDN controller continuously monitors network conditions and traffic flows to ensure that the QoS requirements are being met. If there are significant changes in the network (e.g., increased congestion, link failure), the controller may re-evaluate the traffic flows and update the flow rules accordingly.

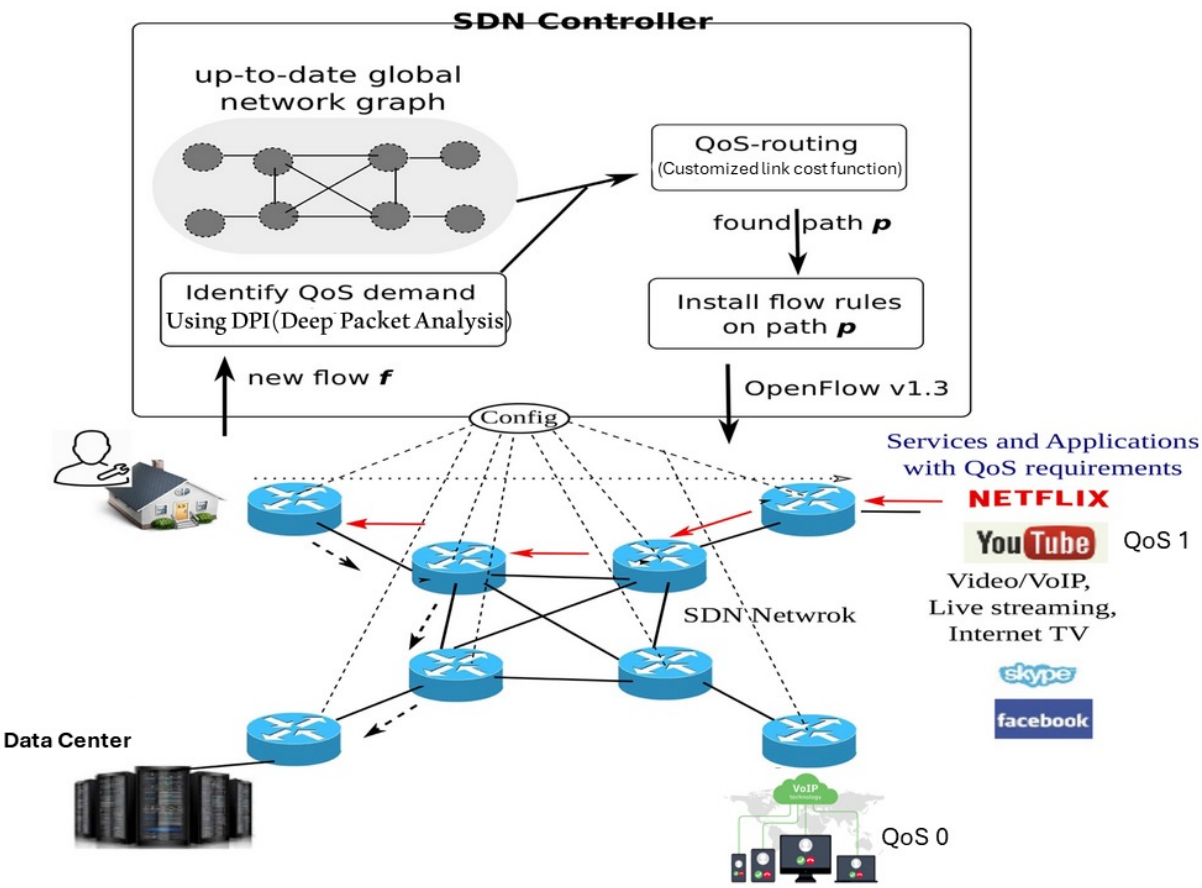

Figure 18: SDN-Based QoS Routing Architecture

6 Performance Evaluation

In this section, we assess the effectiveness of the proposed SDN-based routing algorithm by evaluating key performance metrics. The performance is analyzed based on the algorithm's scalability, its ability to maintain network security and isolation, and its efficiency in recovering from failures. The evaluation aims to highlight the algorithm's robustness and adaptability in dynamic network environments.

6.1 Scalability Considerations

To ensure optimal performance as the network grows, the following scalability strategies are employed:

Distributed Control Plane

Multiple controllers distribute the computational load and avoid single points of failure, allowing horizontal network scaling.

Efficient Data Collection

- **Event-Driven Updates**: Trigger updates only on significant changes.
- **Aggregation**: Minimize data sent to the controller by aggregating metrics.

Optimized Path Computation

- **Hierarchical Routing**: Manageable regions reduce computation complexity.
- **Incremental Updates**: Update only affected paths.

Flow Table Management

- **Flow Aggregation**: Reduces entries in flow tables.
- **Dynamic Expiration**: Intelligent expiration policies free up resources.

6.2 Security and Isolation

Robust security and isolation mechanisms protect the network from unauthorized access and ensure data integrity:

Secure Communication Channels

All controller-device communications are encrypted using TLS to prevent tampering.

Traffic Isolation

- **VLANs**: Logical traffic separation.
- **QoS Policies**: Ensure critical traffic is prioritized.

Intrusion Detection and Prevention

Real-time monitoring with IDS/IPS allows for automatic detection and mitigation of malicious activities.

Policy Enforcement

Centralized and dynamic policy management ensures consistent security across the network.

6.3 Failure Recovery

Failure recovery strategies ensure network resilience and uninterrupted service:

Detection and Redundancy

- **Heartbeat Mechanisms**: Detect unresponsive devices.
- **Precomputed Backup Paths**: Ready alternative routes for quick failover.

Rapid Re-routing

- **Local Re-routing**: Quickly find alternative paths near the failure.
- **Partial Flow Updates**: Update only affected flow tables.

Graceful Degradation

If immediate re-routing is not possible, traffic shaping and service prioritization ensure critical services maintain operation.

Algorithm Integration with Failure Recovery

The pseudocode incorporates failure recovery:

Algorithm 2 Link Cost Calculation and Routing Update

```
1: begin
2: Calculate link costs with bandwidth and distance
3: Send data to the controller
4: Initiate topology based on neighbor data
5: Apply Dijkstra's algorithm
6: Update flow tables
7: Monitor link status
8: if external interrupt occurs then
9:     end
10: else
11:     if link failure detected then
12:         Go to line 3
13:     else
14:         Update cost parameters
15:         Calculate new link costs and send to controller
16:         if costs unchanged then
17:             Go to line 7
18:         else
19:             Go to line 5
20:         end if
21:     end if
22: end if
```

7 Conclusion

This report has explored the development and implementation of a QoS-based routing algorithm designed to optimize network performance for various types of traffic, including VoIP, video streaming, and file downloads, within a Software-Defined Networking (SDN) environment. By leveraging a composite link cost function, the algorithm dynamically adjusts routing decisions based on real-time network conditions and the specific QoS requirements of each traffic type.

The algorithm's effectiveness lies in its ability to automatically classify traffic and apply tailored weights and power factors to different QoS metrics—such as bandwidth, latency, jitter, packet loss, utilization, and distance—ensuring that the selected network paths best meet the demands of the application in question. For VoIP, the emphasis on low latency and jitter ensures clear and uninterrupted voice communication. Video streaming benefits from prioritized bandwidth and minimized packet loss, ensuring smooth playback with minimal buffering. File downloads are optimized for maximum bandwidth and minimized link utilization, enabling fast and efficient data transfer.

The flexibility and adaptability of the SDN controller allow for continuous monitoring and dynamic adjustment of flow rules, ensuring that the network can respond to changes in traffic patterns and maintain high performance across all services. This approach not only enhances user experience but also makes efficient use of network resources, supporting the overall goals of modern network management.

In conclusion, the integration of QoS-based routing with SDN presents a powerful solution for managing diverse and demanding network traffic, offering a scalable and responsive framework that can be adapted to various network environments and evolving application needs.

References

[1] Umme Zakia and Hanene Ben Yedder, *Dynamic Load Balancing in SDN-Based Data Center Networks*, Simon Fraser University, Vancouver, British Columbia, Canada, 2024.
Email: {uzakia, hbenyedd}@sfu.ca
https://www.researchgate.net/profile/Umme-Zakia/publication/321408419_Dynamic_load_balancing_in_SDN-based_data_center_networks/links/5cc3662ba6fdcc1d49b21dca/Dynamic-load-balancing-in-SDN-based-data-center-networks.pdf

[2] *Elbashaer, M. O., Aldeghaishem, A., Lloret, J., Alrajeh, N. (2021). A QoS-based routing algorithm over software-defined networks. Journal of Network and Computer Applications, 194, 103215.*
https://doi.org/10.1016/j.jnca.2021.103215