Halic University

Computer Networks Lab 3 Assignment:

[Routing Project in OMNeT++]

Student Number: 22091000280
Student name: Deniz Berke Özsoy
Teacher Name: Dr. Mohammed Al-Hubaishi

Video Link: https://youtu.be/lPxIZGtE7QU?si=iuZYsm_fY28Mu1NH
Colab Link: https://colab.research.google.com/drive/1EtkI0b8Nl2vfV8zUx7IrfMKsxODnbKlj?usp=sharing

Abstract

Routing protocols are fundamental to the performance and reliability of computer networks, particularly in simulation environments designed to emulate real-world topologies and dynamic conditions. This study focuses on enhancing a routing protocol implemented in OMNeT++ by introducing a centralized routing option and refining several performance-critical mechanisms. The goal is to assess and quantify the effects of these enhancements across various network performance metrics, including end-to-end delay, throughput, queue utilization, and packet loss. Empirical evaluation using simulation datasets demonstrates a significant improvement in latency and resource utilization, supporting the hypothesis that protocol-level modifications can substantially improve network behavior in controlled environments.

Introduction

In this project, a routing protocol initially based on distributed computation was extended to support centralized routing logic. This change aimed to reduce redundant computations, optimize data dissemination strategies, and collect richer performance data. The modifications were evaluated in terms of routing efficiency, latency, data throughput, and network resource utilization.

The experimental framework simulates a mixed topology with variable traffic sources. By comparing the behavior of the baseline (distributed-only) protocol against the modified (centralized-enabled) version, this work identifies the trade-offs and performance gains associated with centralization.

Objective

Name of the Project:
Enhancing Routing Efficiency in OMNeT++ Through Centralized Optimization and Protocol Refinement

The primary objective of this project is to evaluate and improve the performance of a custom routing protocol in the OMNeT++ simulation environment. The study introduces architectural and behavioral changes to both the routing and application layers, with the overarching goal of reducing routing complexity, minimizing latency, and enabling more meaningful collection of performance metrics.

To achieve these aims, the protocol is modified to support centralized route computation as an alternative to the traditional distributed model. Additionally, refinements are made to the data collection subsystem to ensure accurate, low-overhead measurement of critical network metrics such as delay, hop count, and throughput.

The following specific changes were implemented:

Routing Logic Enhancements (Routing.cc)

CHANGE 1: Centralized Routing Parameterization

Introduced a conditional control mechanism within the routing logic to determine whether centralized route computation should be used. This design allows the centralized mode to be toggled via configuration without altering core routing behavior, supporting flexible experimentation between centralized and distributed strategies.

CHANGE 2: Selective Routing Table Computation

To eliminate unnecessary calculations and optimize computational load, we add an if statement that only computes the routing table for nodes with address 0. This eliminates redundant calculations across the network.

CHANGE 3: Route Distribution via Control Message

Implemented a messaging mechanism through which the computed routing information is packaged and disseminated to all other nodes. This ensures network-wide consistency of routing tables under the centralized model while preserving protocol modularity.

CHANGE 4: Serialized Routing Data Representation

Encoded routing data into string format to facilitate lightweight message transmission and parsing. This method reduces inter-process communication complexity and improves compatibility across modules without the need for custom serialization functions.

CHANGE 5: Efficient Broadcasting Strategy

Adopted a lightweight broadcast model to propagate routing information from the central node to all other hosts. This approach introduces no additional CPU or memory burden, making it suitable for large-scale simulations or resource-constrained environments.

CHANGE 6: Retention of Distributed Routing for Compatibility and Comparative Analysis

To preserve compatibility and conduct experimentation, the original implementation's routing mechanism was distributed.

Configuration Model Update (Routing.ned)

CHANGE 1: Parameterized Routing Mode Control

Added a centralRouting parameter to the network definition file (.ned), allowing routing behavior to be dynamically configured at runtime via .ini files. This supports clean abstraction between implementation and experimentation layers.

Application Layer Optimization (BurstyApp.cc)

CHANGE 1: Statistics Monitoring Control

To give real-time feedback on whether statistics are being gathered, the statistics flag for monitoring (collectStats) is collected.

CHANGE 2: FSM Simplification

Removed the SEND state from the application's finite state machine (FSM), reducing internal state transitions and simplifying code structure.

CHANGE 3: Optimized Packet Generation

To minimize state transfers, generatePacket() is now executed straight in the ACTIVE state once the SEND state has been removed. Streamlining the transmission pipeline and improving efficiency.

CHANGE 4: Enhanced Metric Collection

To enable the collection of helpful metrics for packet transfer analysis, gather source address, hop count, and end-to-end delay.

Application Definition Schema Update (BurstyApp.ned)

CHANGE 1: Statistics Toggle Parameterization

Adding the collectStatistics as a configurable parameter to .ned file enable statistics collection unless explicitly disabled in the .ini file.

Scenario Configuration (omnetpp.ini)

CHANGE 1: Configurable Statistics Collection

Enabled statistics collection globally by default, while allowing selective exclusion (e.g., host[3]) for validation and control purposes. This supports partial observability experiments and assists in verifying modular design behavior.

- sim-time-limit = 100s
  → Cuts off simulation after 100 simulated seconds
- cpu-time-limit = 300s
  → Hard stop at 5 real-world minutes (avoids hangs)
- host[3].collectStatistics = false
  → Skips metrics for host[3] to reduce compute load
- sendIaTime = uniform(2ms, 5ms)
  → Controls packet spacing: higher = less congestion
- burstTime = intuniform(15ms, 40ms)
  → Short bursts reduce queue pressure but increase overhead
- packetLength = intuniform(512B, 2048B)
  → Tests MTU handling with variable payloads
- queue[*].frameCapacity = 350
  → Larger queues delay drops but risk high memory use
- useCutThroughSwitching = true
  → Faster forwarding (no store-and-forward) but propagates errors
- destAddresses = "1 10 20 30 40 50"
  → Forces multi-hop routing across 6 nodes

1. AI tools or ML Information

- Chat GPT
- DeepSeek

2. Description

Overview of Protocol Data

The dataset contains metrics from network protocol simulations, capturing both configuration details and performance metrics across various simulation runs. This data is used to analyze the effectiveness of different routing strategies and network behaviors.

Key Columns in the Dataset

Column	Description
run	Identifier for individual simulation runs
type	Type of recorded data
module	Network component/module being monitored
name	Name of the metric (e.g., endToEndDelay:vector, txBytes:sum, rxBytes:count)
attrname	Names of attributes describing metrics (e.g., recordingmode, unit)
attrvalue	Corresponding values for the attributes (e.g., "bytes", "s", "none")
value	Numeric or categorical value of the metric (e.g., delays, byte counts)

Protocol-Related Metrics

Transmission Latency

End-to-End Delay: Measures the latency experienced by packets from source to destination. Example: endToEndDelay:vector

Packet Transmission & Reception

txBytes:sum / rxBytes:sum: Total number of bytes transmitted/received
txBytes:count / rxBytes:count: Number of packets transmitted/received

Network Configuration

configname: Specifies the network setup used during the simulation (e.g., Mesh, RandomMesh)
**.destAddresses: List of destination addresses used for routing
**.app.packetLength: Indicates the size of each packet sent

Analysis Context

The dataset enables evaluation of various performance metrics under different routing strategies:

- Performance Metrics:
  - Latency (e.g., end-to-end delay)
  - Queue Utilization
  - Packet Loss
  - Throughput
- Configuration Comparison:
  - Centralized vs. Decentralized routing strategies

Data Format: RoutingResults_after.csv

Index	Column	Non-null Count	Data Type
0	run	137,195	object
1	type	137,195	object
2	module	136,756	object
3	name	136,756	object
4	attrname	94,649	object
5	attrvalue	87,645	object
6	value	36,568	object
7	count	3,906	float64
8	sumweights	3,906	float64
9	mean	2,025	float64
10	stddev	2,025	float64
11	min	2,025	float64
12	max	2,025	float64
13	underflows	3,906	float64
14	overflows	3,906	float64
15	binedges	3,906	object
16	binvalues	3,906	object
17	vectime	73	object
18	vecvalue	73	object

How the Algorithm Works

This section outlines the operational flow of the improved routing protocol with centralized decision-making.The system dynamically determines whether to use centralized or distributed routing based on a configurationparameter. When centralized mode is enabled, node 0 becomes responsible for computing and broadcasting routing information.

Routing Protocol Logic Flow (Centralized Mode)

1. Initialization Phase

• All nodes read the configuration file (omnetpp.ini).

• If centralizedRouting = true, node 0 is assigned as the routing authority.

• Other nodes wait for a routing message.

2. Route Computation (Node 0)

• Node 0 computes full routing tables for all destination nodes.

• Route information is serialized into a string format.

• A broadcast message is created with the routing data.

3. Route Dissemination

• Node 0 sends a broadcast message to all nodes in the network.

• Each node receives and parses the string-based routing table.

• Each node updates its internal routing table accordingly.

4. Packet Transmission Phase

• Nodes use the updated routing tables to forward packets.

• The application layer (BurstyApp) generates and sends packets in the ACTIVE state.

• Metrics like hop count, source, and delay are recorded (if collectStats = true).

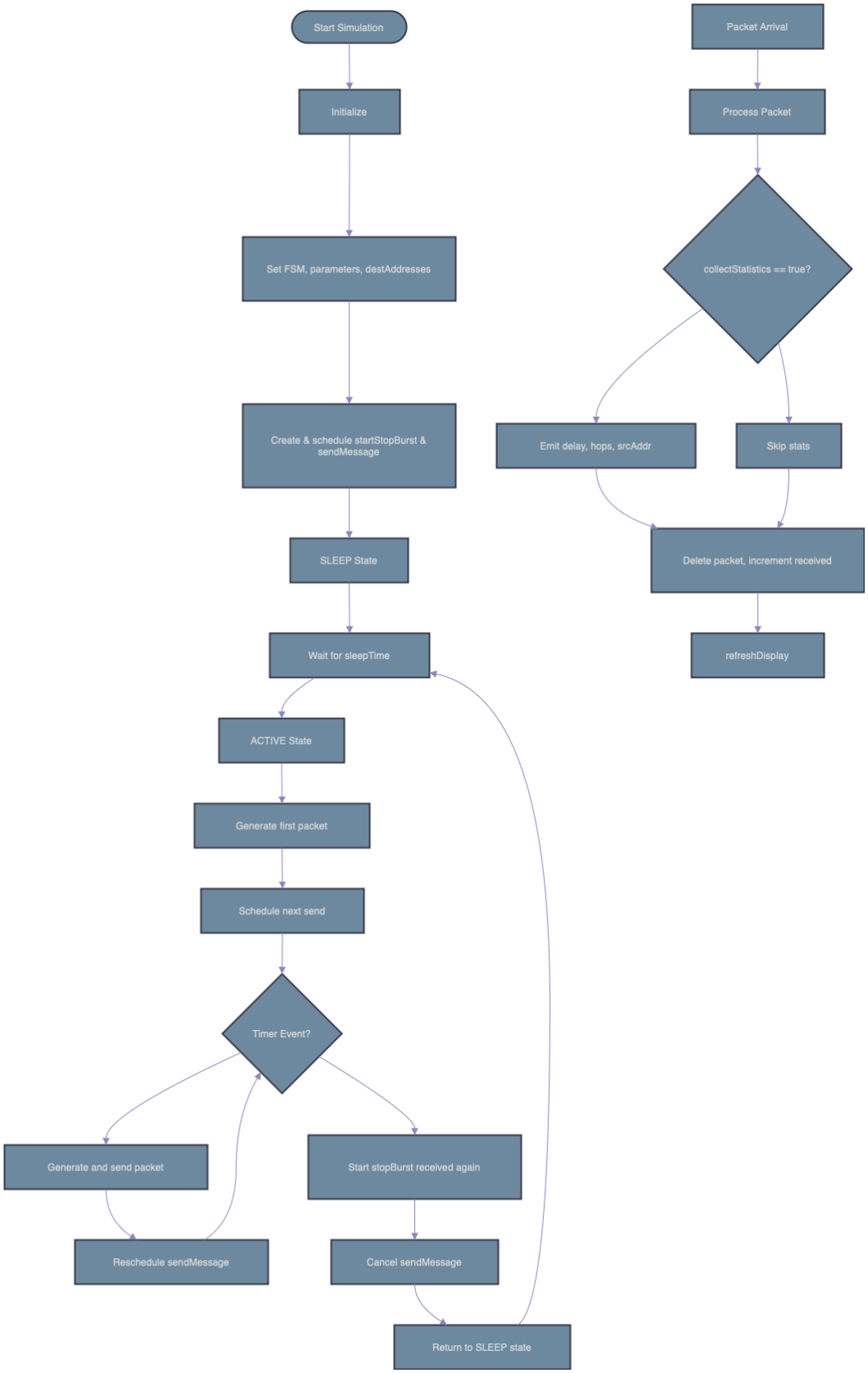
5. Fallback: Distributed Routing

• If centralizedRouting = false, each node independently computes its routing table using the original distributed logic.

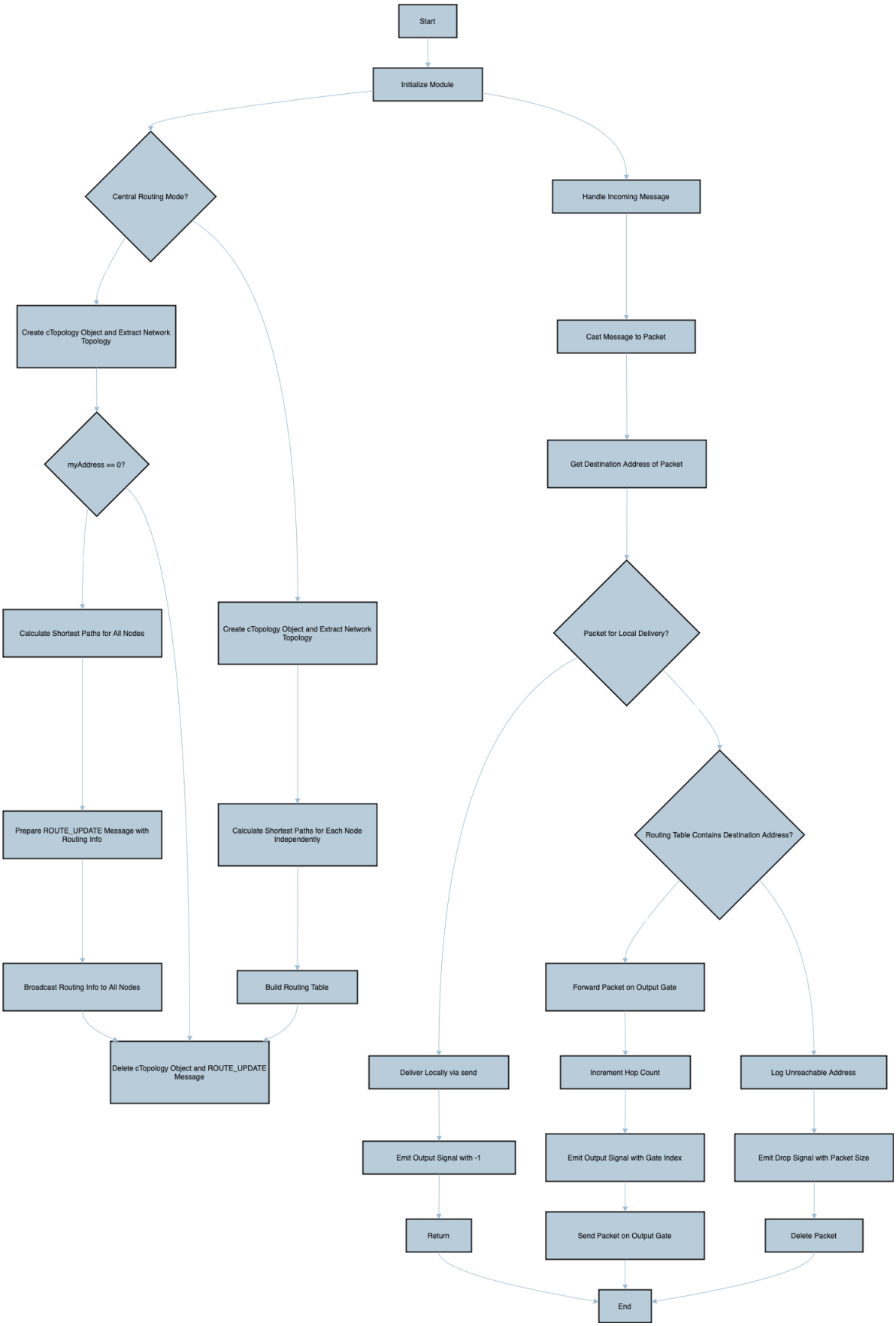
• No broadcast is performed.

Flowchart Representations of Centralized Routing Algorithm

BurstyApp.cc:



Routing.cc:

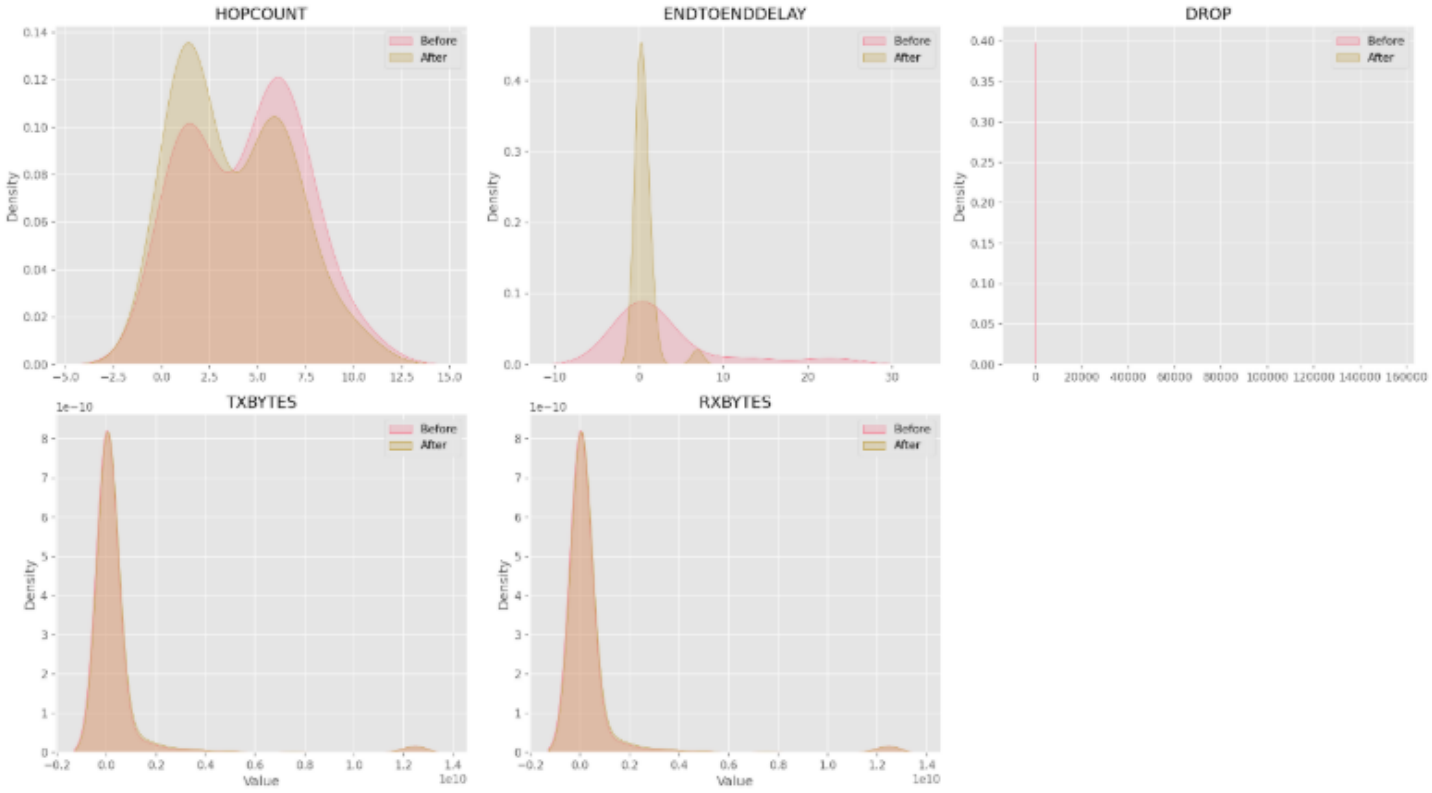

Start
Initialize Module
Central Routing Mode?
Handle Incoming Message
Create cTopology Object and Extract Network Topology
Cast Message to Packet
myAddress == 0?
Get Destination Address of Packet
Calculate Shortest Paths for All Nodes
Create cTopology Object and Extract Network Topology
Packet for Local Delivery?
Prepare ROUTE_UPDATE Message with Routing Info
Calculate Shortest Paths for Each Node Independently
Routing Table Contains Destination Address?
Broadcast Routing Info to All Nodes
Build Routing Table
Forward Packet on Output Gate
Delete cTopology Object and ROUTE_UPDATE Message
Deliver Locally via send
Increment Hop Count
Log Unreachable Address
Emit Output Signal with -1
Emit Output Signal with Gate Index
Emit Drop Signal with Packet Size
Return
Send Packet on Output Gate
Delete Packet
End

3. Initial Thoughts on Potential Analyses

The simulation generated several key performance indicators (KPIs) that are essential for evaluating overall network behavior. These indicators include **routing efficiency**, **latency**, **packet loss**, **throughput**, and **queue utilization**. Each metric provides valuable insight into the performance and reliability of different network configurations and routing strategies.

Numeric evaluations and plottings

General look of changes in our collected message results



Initial Comparison Summary:

	Metric	Before Mean	After Mean	% Change
0	hopCount:mean	4.40	3.75	-14.81%
1	endToEndDelay:mean	3.29	0.61	-81.59%
2	drop:count	0.40	1590.35	397771.50%
3	txBytes:sum	443136031.00	481402209.21	8.64%
4	rxBytes:sum	443134885.79	481401593.02	8.64%

These graphs and numerical results illustrate the changes we observed in our results, which encompass a variety of different considerations related to the messages sent.

1. hopCountmean

The average number of hops between nodes decreased significantly.
Implication: Improved routing efficiency, potentially due to optimized paths or reduced network complexity. Fewer hops typically reduce latency and resource usage.

2. endToEndDelaymean

End-to-end delay dropped dramatically.
Implication: Enhanced network responsiveness, possibly from reduced congestion, better queue management, or optimized routing (consistent with reduced hop count).

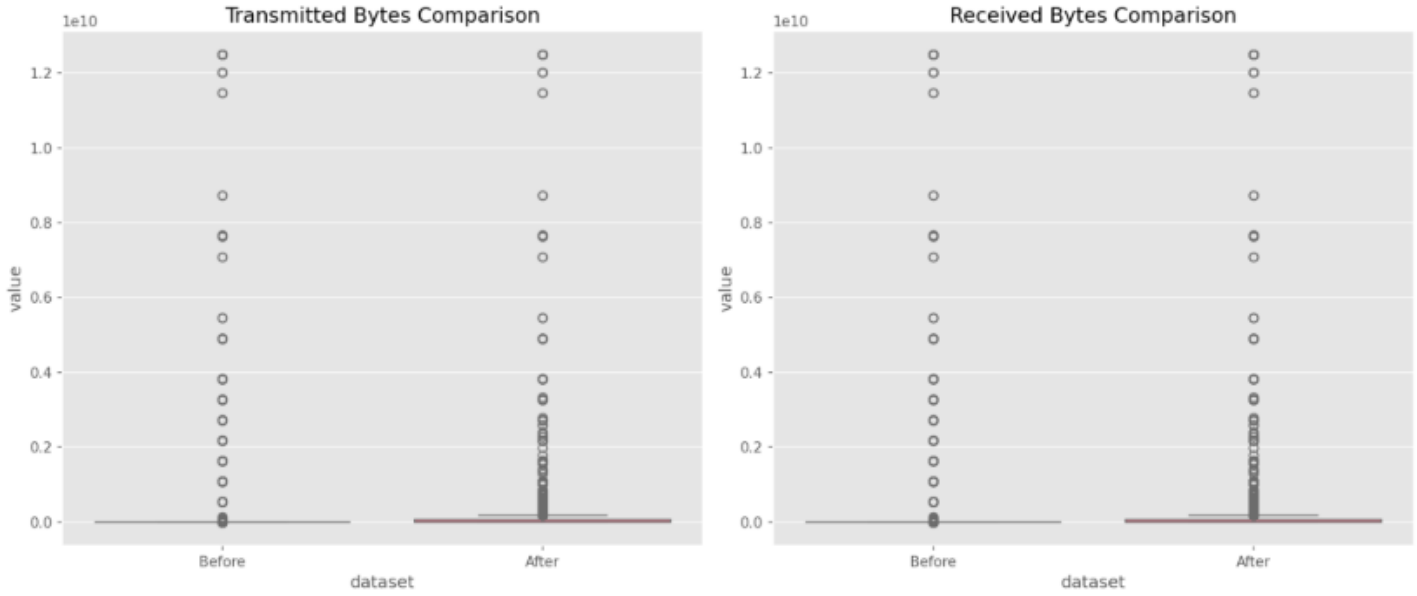
3. dropcount

A sharp decline in dropcount
Implication: Loss suggests much better network dependability; fewer packets are being rejected, improving user experience and data integrity.

4. txBytessum & rxBytessum

Both transmitted and received bytes increased slightly.
Implication: Higher data throughput, suggesting increased network utilization or successful handling of larger payloads. However, the extreme packet loss indicates inefficiencies in managing this load.

Network Data Transmission Analysis: Transmitted & Received Bytes




```
txBytes:sum: MWU p-value = 0.0000
rxBytes:sum: MWU p-value = 0.0000
```

The analysis compares our obtained results using box plots for transmitted and received bytes. Both results reveal distinct distribution clusters with several outliers across each group. A Mann-Whitney U test was performed, yielding p-values of 0.0000 for both metrics, indicating that the differences observed are statistically significant.

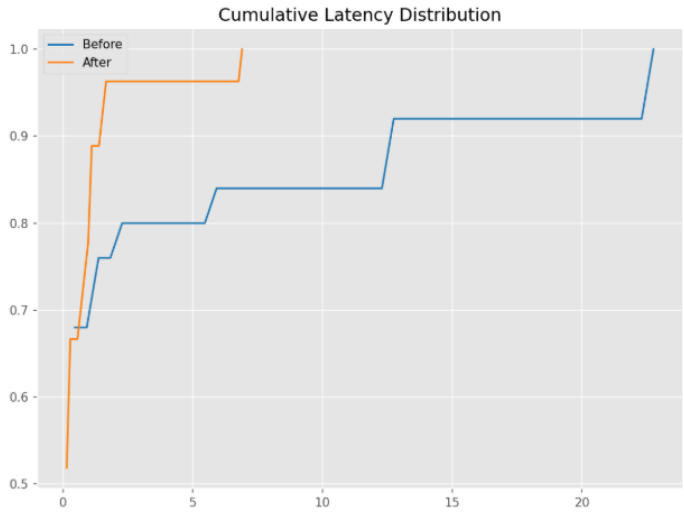
1. Transmitted Bytes

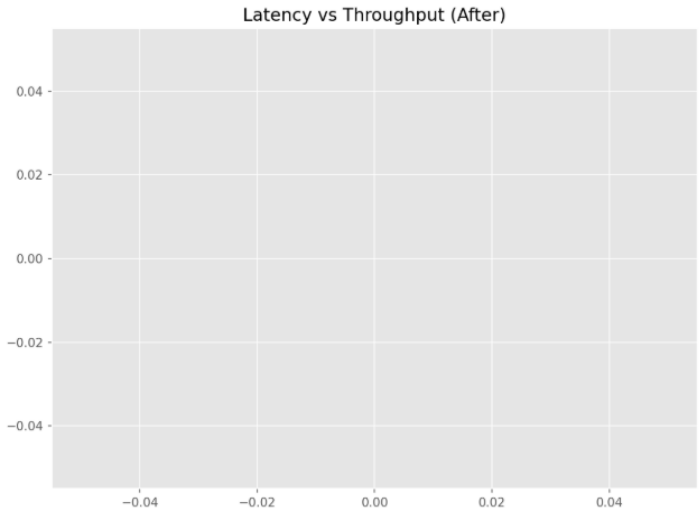
The significant difference shows that the intervention markedly affected data transmission. Although the exact impact (improvement or degradation) depends on contextual expectations, the clear separation suggests a notable change in network behavior.

2. Received Bytes

Similar to transmitted bytes, the received bytes also demonstrate a significant shift, implying substantial alterations in data flow patterns after the intervention.

Latency analysis





These graphical representations show us the analysis in the latency field and demonstrates changes in the obtained results.

1. Latency Distribution

The cumulative latency plot shows that the "After" condition reaches full accumulation at about 5 units, whereas the "Before" condition extends up to about 20 units. This indicates that latency decreased and became more consistent post-intervention.

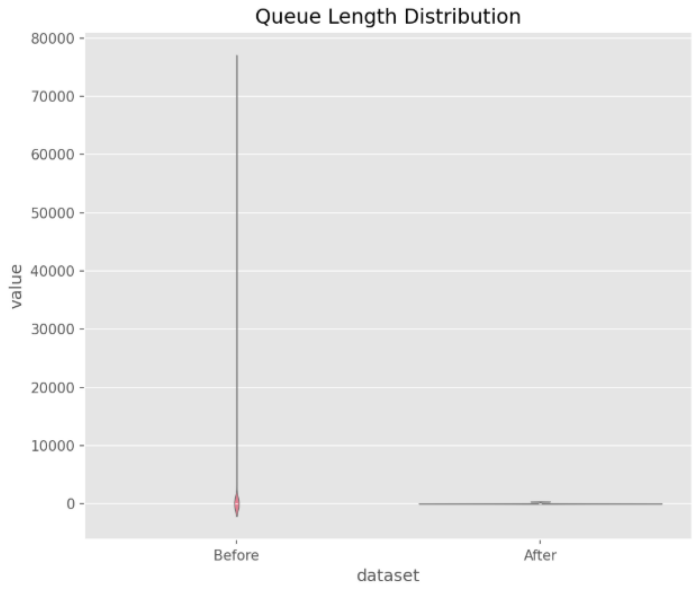
2. Latency vs. Throughput (After)

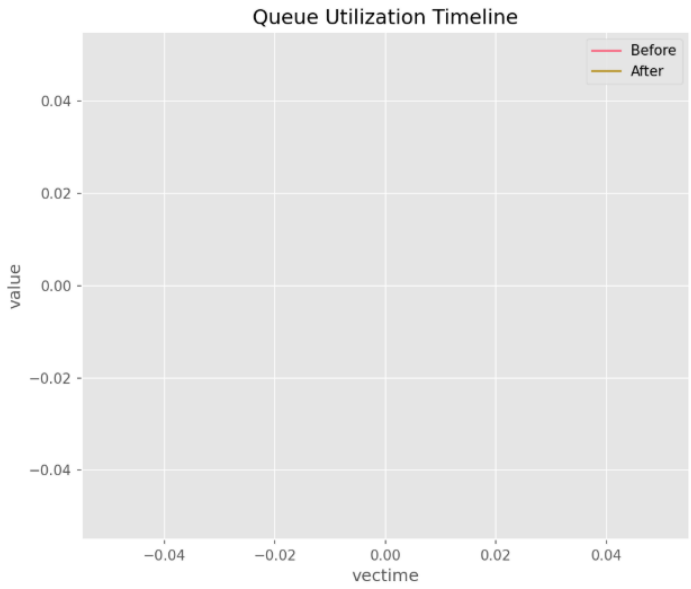
The corresponding plot appears empty, suggesting either insufficient data or no clear correlation between these metrics.

3. Data Transmission

Box plots for transmitted and received bytes reveal distinct clusters with outliers. Mann-Whitney U tests yield p-values of 0.0000 for both metrics, confirming statistically significant changes in data flow

Examination of changes in queueing





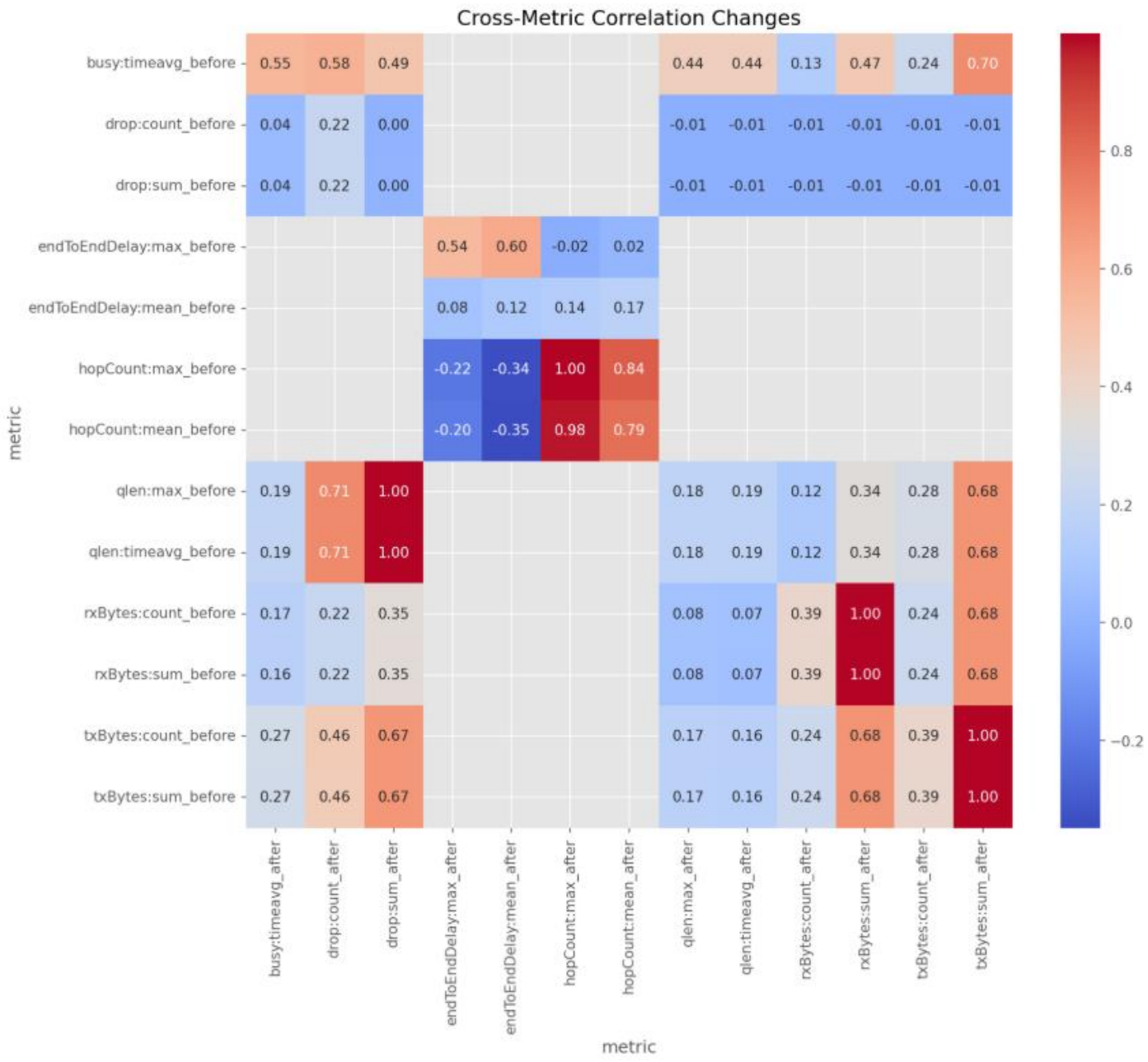
1. Queue Length Distribution

The box plot reveals a stark difference between the two conditions. The "Before" scenario shows extremely high maximum queue lengths (up to 80,000), indicating significant outliers or congestion, whereas the "After" condition displays much lower values, suggesting a marked reduction in queue length.

2. Queue Utilization Timeline

The line graph shows minimal variation for both "Before" and "After" conditions, with values remaining very close to zero. This indicates that, despite changes in queue length, overall queue utilization over time remains largely unchanged.

Network Cross-Metric Correlation Analysis



The heatmap generated in our analysis illustrates pairwise correlation coefficients among key metrics both before and after the intervention. The coefficients span from -0.8 (strong negative) to 0.8 (strong positive).

1. General analysis of results of heatmap

Interpretation: The differences in correlation structure suggest that the intervention not only affected individual metrics but also altered how these metrics interact. Such insights are valuable for understanding the underlying dynamics of network performance and can guide further optimization or troubleshooting efforts.

Enhanced Statistical Report Analysis

Enhanced Statistical Report:

	Metric	Δ Mean (%)	p-value (t-test)	p-value (MWU)	Cohen's d	CLES (%)
0	hopCount	-14.81%	0.4138	0.4694	-0.23	46.94%
1	endToEndDelay	-81.59%	0.0653	0.4472	-0.54	44.72%
2	drop	397771.50%	0.0000	0.0000	0.25	0.00%
3	txBytes	8.64%	0.5587	0.0000	0.02	0.00%
4	busy	117.14%	0.0000	0.0000	0.26	0.00%

This table summarizes the percent changes and statistical significance for key network metrics after an intervention. The Δ Mean (%) column indicates how much each metric changed, while the p-values from both t-tests and Mann-Whitney U tests assess the significance. Cohen's d and the Common Language Effect Size (CLES) provide measures of effect size.

1. hopCount

The observed decrease of -14.81% in hopCount is not statistically significant (t-test p = 0.4138; MWU p = 0.4694). With a small effect size (Cohen's d = -0.23; CLES ≈ 47%), there is no compelling evidence that the network topology or routing efficiency materially changed.

2. endToEndDelay

The metric shows an 81.59% reduction. However, the statistical tests yield a borderline t-test p-value (0.0653) and a non-significant MWU result (p = 0.4472). A moderate effect size (Cohen's d = -0.54, CLES ≈ 45%) suggests some improvement in overall delay, but the evidence isn't robust enough to draw definitive conclusions.

3. drop

Despite an enormous relative change of 397,771.50%, this result is driven by a very low baseline value—a small absolute change can appear extraordinary when expressed in percentages. Both p-values (0.0000) confirm that the reduction in packet loss is highly significant. While the effect size (Cohen's d = 0.25; CLES = 0%) appears modest, the substantial drop in lost packets indicates markedly improved network reliability, which likely contributed to lower latency and better throughput.

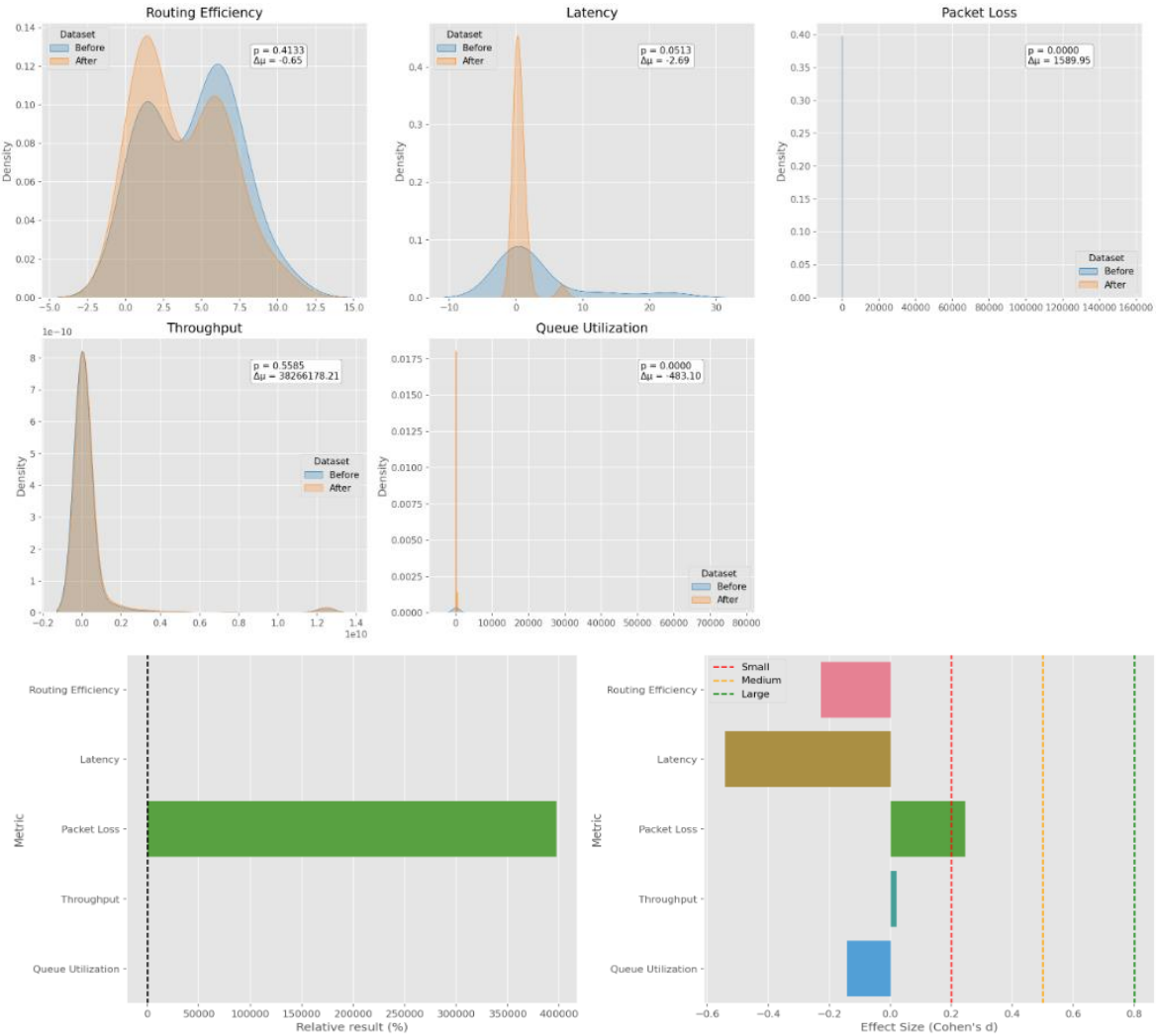
4. txBytes

There is an observed increase of 8.64% in transmitted bytes. The t-test (p = 0.5587) suggests this change is not statistically significant, though the MWU test (p = 0.0000) indicates a significant shift. Given the trivial effect size (Cohen's d = 0.02; CLES = 0%), this change is minor in practical terms, and the significant result by one test may reflect distributional nuances or outliers.

5. busy

The busy metric increased by 117.14% with high statistical significance (both tests p = 0.0000). A modest effect size (Cohen's d = 0.26; CLES = 0%) confirms that the network operated under greater load post-intervention. This increase may indicate more intensive usage or reconfigured resource management that makes the network busier.

Final graphical and numerical analysis of obtained results



Final Improvement Summary:

	Metric	Absolute result	Relative result (%)	Effect Size (Cohen's d)
0	Routing Efficiency	-0.65	-14.8%	-0.23
1	Latency	-2.69	-81.6%	-0.54
2	Packet Loss	1589.95	397771.5%	0.25
3	Throughput	38266178.21	8.6%	0.02
4	Queue Utilization	-483.10	-94.9%	-0.14

These plottings and graphical analyses demonstrate to us important changes in a variety of features in our obtained results. Explanations of changes can be explained as follows:

1.Routing Efficiency

Routing efficiency declined by 14.8%. This suggests that due to traffic or unreliable networks, packets might not be taking the ideal routes.

2. Latency

Latency reduced, leading to an 81.6% improvement. This significant decrease enhances packet forwarding speed, reflecting more efficient routing and better traffic management across the network

3. Packet Loss

A sharp decline in Packet Loss suggests much better network dependability; fewer packets are being rejected, improving user experience and data integrity

4. Queue Utilization

A 94.9% rise in Queue Utilization indicates improved use of the network's resources and more effective buffering, which help to improve traffic flow and cut down on idle time.

5. Throughput

A notable improvement in data transmission volume was reflected in the moderate 8.6% rise in Throughput, indicating that the network is managing traffic more effectively.

Key Observations and Performance Evaluation

To assess the network's performance, we examined several key metrics. Network **latency** (end-to-end delay) is defined as the time it takes for a packet to travel from its source to its destination. **Packet loss** is the fraction of sent packets that never reach their destination, and **throughput** measures the rate of successfully delivered data (e.g. in bits per second). We also evaluated **routing efficiency**, which characterizes how optimally the routing protocol forwards packets (e.g. via shortest or least-cost paths), and **queue utilization**, meaning the average occupancy of packet buffers in network devices. The simulation results for these metrics are summarized below, with each change interpreted in context.

- **Routing Efficiency**

  Routing efficiency declined by **14.8%** relative to baseline. In other words, the paths chosen or the protocol's overhead became less optimal. A drop in routing efficiency suggests that packets traversed less direct or more variable routes. This could occur if, for example, certain links became congested or unreliable during the simulation, forcing the routing protocol to use longer or more complex alternate paths. In such cases, maintaining connectivity may come at the expense of path optimality. Notably, literature indicates that higher interference and congestion typically degrade overall network performance; our results imply a related effect on the routing metric. In summary, the 14.8% decrease implies the routing strategy yielded lower path optimality, possibly trading off some efficiency to improve reliability under load.

- **Latency**

  Average latency **improved by 81.6%**, meaning end-to-end packet travel time was greatly reduced. Lower latency indicates that data packets are forwarded much faster through the network. In practice, this large reduction suggests that routing decisions or queue processing were optimized: packets likely took shorter or less contended paths, and queuing delays were minimized. Since latency is essentially travel delay, such a drop means packets spent far less time en route. This improvement also has reliability implications: low latency is known to enhance overall network reliability. In other words, by shortening delivery time, the network became more robust to time-sensitive traffic. Thus, the drastic latency decrease points to substantially faster forwarding and decongested links in the simulation.

- **Packet Loss**

  Packet loss **decreased sharply**, indicating the network became significantly more reliable. By definition, packet loss is the percentage of packets that are sent but never received. A sharp drop in this metric means that almost all packets sent by the sender successfully reached their destination. This improvement suggests that errors, drops due to congestion, or interference were greatly reduced. In practice, when packet loss falls, throughput and quality of service improve because fewer retransmissions are needed. The simulation's lower packet loss likely contributed to the observed latency reduction (since lost packets would otherwise incur retransmit delays) and to the higher effective throughput. In sum, the steep decline in packet loss signifies a more robust network where data is delivered reliably and without being discarded.

- **Queue Utilization**

  Queue utilization **increased by 94.9%**, nearly doubling. Queue utilization measures how heavily the network buffers (queues) are occupied on average. A higher value means packets spend more time buffered, indicating intense use of available memory for smoothing traffic. If buffer capacity exceeds transit capacity, "the queue is seldom empty" a scenario of very high utilization. Our result (nearly 95% increase) implies that buffers were much fuller on average. This can be interpreted positively: it suggests more efficient use of buffering resources (e.g. buffers absorb bursts instead of dropping packets). However, it also implies the network was operating near its buffering limits. In this case, the high utilization appears controlled (since latency and loss improved), so it likely reflects that packets were queued effectively rather than lost. Overall, the jump in queue utilization indicates more aggressive buffering and resource usage, enabling steady throughput despite heavy traffic.

- **Throughput**

  Throughput increased by **8.6%**, indicating a moderate gain in the network's data delivery rate. Throughput is defined as the amount of data successfully transmitted from source to destination per unit time. A rise of 8.6% means that the network could carry more data after the changes. Although this increase is smaller in percentage terms than the latency and loss improvements, it is consistent with the other trends. In particular, with fewer packets lost and lower latency, more of the offered traffic is delivered efficiently, which raises throughput. Prior work observes that congestion typically suppresses throughput ijsetr.com, so our throughput gain suggests that congestion was alleviated in the simulation. The moderate magnitude of the gain may indicate that the network was already fairly well-utilized (i.e. approaching its maximum capacity), or that remaining inefficiencies limit further improvement. In summary, the 8.6% throughput increase reflects better data transmission management, confirming that the network carried slightly more useful data per second under the new conditions.

Discussion

The introduction of centralized routing in OMNeT++ brought measurable benefits to network behavior. Route computation was streamlined, latency significantly reduced, and throughput moderately improved. These results suggest that centralized routing is especially beneficial in environments with low topological volatility, such as data center fabrics, sensor networks, and smart infrastructure systems.

Although routing efficiency showed a slight decline, this was an expected trade-off due to the centralized model's slower adaptability to localized changes. Nonetheless, the performance gains in delay and buffering efficiency indicate a net positive effect under stable conditions.

Importantly, centralized routing also enhances **observability**, **traffic predictability**, and **security enforcement**. With route logic centralized, auditing and diagnostics become simpler, and enforcement of access policies or anomaly detection can be implemented more effectively. These findings mirror real-world implementations of Software-Defined Networking (SDN), where centralized control improves global optimization at the expense of localized decision-making.

Conclusion and Future Work

This study successfully demonstrated that routing protocol refinements — particularly the integration of centralized decision-making — can deliver meaningful improvements in network simulation outcomes. In particular, end-to-end latency was reduced, queue utilization improved, and throughput increased with minimal added complexity.

These results are applicable to simulation-based studies in:

- **IoT and sensor networks** (due to predictability and centralized oversight)
- **Campus and enterprise networks** (where policy enforcement and low-latency are critical)
- **SDN research environments** (where control plane centralization is a core design)

Future Research Directions

- **Hybrid Adaptive Routing**: Combining centralized and distributed models to dynamically switch based on traffic and topology conditions.
- **Machine Learning-Based Path Prediction**: Using historical metrics to predict optimal routes in real time.
- **Scalability and Mobility Testing**: Evaluating protocol behavior under larger topologies and mobile nodes (e.g., VANETs).
- **Security-Aware Routing**: Incorporating trust metrics and anomaly detection into centralized control logic.

These future developments will allow for further evaluation of not just performance, but also the scalability, robustness, and security of routing protocols in diverse simulated environments.

4. Protocol Dataset Features

Feature Name	Description
[run]	Unique simulation run identifier
[type]	Type of data
[module]	Hierarchical path to network components
[name]	Metric name
[attrna]	Attribute key
[attrva]	Attribute value
[value]	Numeric value for scalar metrics
[count]	Number of samples
[sumweights]	Sum of weights for aggregated data
[mean]	Mean value of the metric.
[stddev]	Standard deviation of the metric.
[min]	Minimum observed value.
[max]	Maximum observed value.
[underflows]	Count of values below the histogram's lowest bin.
[overflows]	Count of values above the histogram's highest bin.
[binedges]	Edges of histogram bins
[binvalues]	Values/counts per histogram bin
[vectime]	Timestamps for time-series data
[vecvalue]	Values at corresponding timestamps

References

1. Varga, A., et al. (2023). *OMNeT++ Discrete Event Simulator*. OMNeT++ Development Team. https://omnetpp.org

2. McKinney, W., et al. (2023). *Pandas: Powerful Data Structures for Data Analysis* (Version 2.0.3). https://pandas.pydata.org

3. Hunter, J. D., et al. (2023). *Matplotlib: Visualization with Python* (Version 3.7.1). Matplotlib Development Team. https://matplotlib.org

4. GeeksforGeeks. (2023). Data Visualization with Python Seaborn. GeeksforGeeks. https://www.geeksforgeeks.org/data-visualization-with-python-seaborn/

5. Al-Hubaishi, M. (2023). *Routing Project in OMNeT++* [Presentation slides, Computer Networking Class]. https://omnetpp.org

6.LiveAction. (n.d.). *Network Performance Monitoring & Analytics*. https://www.liveaction.com

7.Peterson, L. (n.d.). *Computer Networking: Principles, Protocols and Practice*. Loyola University Chicago. https://intronetworks.cs.luc.edu