UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

MACHINE LEARNING-ASSISTED PREDICTION IN PACKET-SWITCHED 5G XHAUL NETWORKS

RAUL-IOAN FERENT

Thesis supervisor
DAVIDE CAREGLIO (Department of Computer Architecture)

Thesis co-supervisor
JORDI PERELLO MUNTAN (Department of Computer Architecture)

Degree
Master's Degree in Innovation and Research in Informatics (Computer Networks and Distributed Systems)

Master's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Abstract

In an era where Machine Learning (ML) powers core technologies, from spam filtering and content recommendation to autonomous control systems, network admission in 5G fronthaul networks is still governed by conservative Worst-Case (WC) latency bounds. While provably safe, these bounds frequently result in rejecting flows that would, in practice, meet strict Service-Level Agreement (SLA).

This thesis investigates a data driven alternative. Using simulated flow level data from multiple packet-switched xHaul topologies, including ring and mesh layouts, we train lightweight classifiers, namely Logistic Regression (LR), Random Forest (RF), and Multilayer Perceptrons (MLPs), to predict SLA compliance under latency thresholds of 100 μs, 75 μs, and 50 μs.

These models are optimized using an asymmetric cost function that penalizes false negatives more than false positives, reflecting the operational risk of deadline violations. Compared to the WC estimator, ML-based models reduce unnecessary rejections by up to 90% while maintaining deadline violations below 0.1%.

Additional stress tests confirm model robustness under tighter SLAs, varying transmission windows (5G numerologies), and with the inclusion of queue-aware features. These results suggest the feasibility of ML-assisted admission control in time-sensitive 5G and potentially 6G transport networks.

This work has been part of the Spanish I+D+i project TRAINER-A (ref. PID2020-118011GB-C21), funded by MCIN/AEI 10.13039/501100011033

Contents

List of Figures

List of Tables

List of Acronyms

5G	Fifth-generation	8
BH	BackHaul	8
CU	Central Unit	8, 9
DC	Data Center	8
DL	DownLink	9
DNC	Deterministic Network Calculus	13
DU	Distributed Unit	8
eMBB	enhanced Mobile BroadBand	8
FH	FrontHaul	8
FH_DL	Fronthaul Downlink	10
FH_UL	Fronthaul Uplink	10
FN	False Negative	9
FP	False Positive	9
ILP	Integer Linear Programming	14
LR	Logistic Regression	2
MAE	Mean Absolute Error	37
MH	MidHaul	8
ML	Machine Learning	2
MLP	Multilayer Perceptron	2
mMTC	massive Machine-Type Communications	8
PDCP	Packet Data Convergence Protocol	12
PP	Processing Pool	15
RAN	Radio Access Network	8
RF	Random Forest	2
RFE	Recursive Feature Elimination	23
RU	Remote Unit	8
SLA	Service-Level Agreement	2
TN	True Negatives	32
TP	True Positives	32
TSN	Time-Sensitive Networking	9
UL	UpLink	9
URLLC	Ultra-Reliable Low-Latency Communications	8
VIF	Variance Inflation Factor	21
WC	Worst-Case	2

1 Introduction

1.1 Background and Motivation

Fifth-generation (5G) Radio Access Network (RAN) divide the traditional base station into three functional blocks: Remote Units (RUs), Distributed Units (DUs), and Central Units (CUs). These elements are interconnected via a transport infrastructure that includes FrontHaul (FH), MidHaul (MH), and BackHaul (BH) segments, collectively referred to as xHaul. [3]

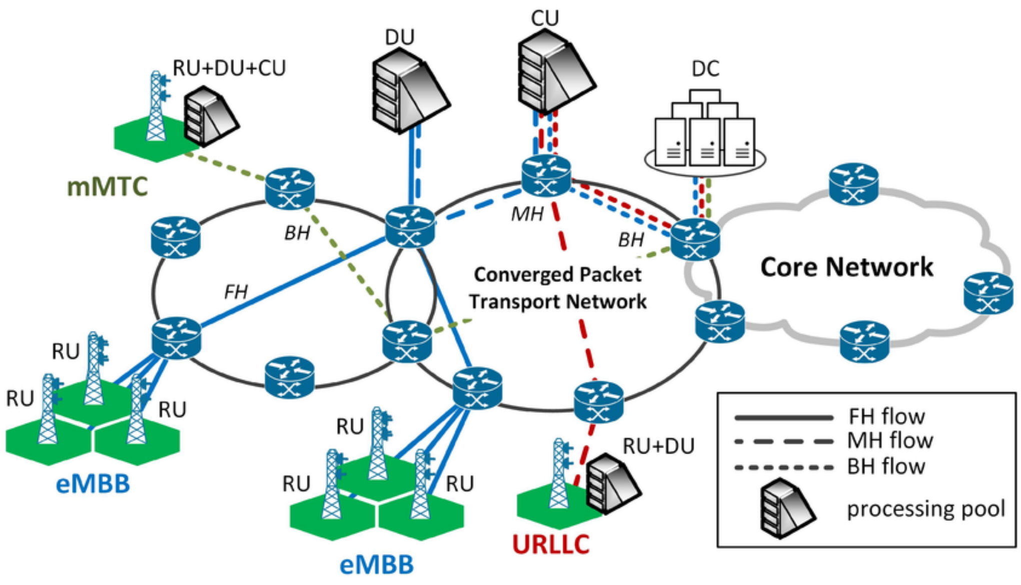

Figure 1: Layered view of a 5G RAN transport network showing fronthaul, midhaul, and backhaul segments, along with different traffic types: eMBB, URLLC, and mMTC. *Adapted from [1]; used with permission.*

Figure 1 shows a representative 5G transport network architecture. At the access level, enhanced Mobile BroadBand (eMBB) flows originate from users demanding high throughput (like streaming or cloud gaming), while Ultra-Reliable Low-Latency Communications (URLLC) requires extremely low latency for applications like autonomous driving or remote surgery. Additionally, massive Machine-Type Communications (mMTC) supports a high density of IoT devices with lower bandwidth but strict energy constraints.

These services impose different latency requirements across segments:

- For instance, FH flows connecting RUs to DUs must meet the most stringent one-way latency constraint of < 100 µs to support URLLC
- MH links (between DUs and CUs) can tolerate delays up to 1 ms
- BH segments, such as the path from a central CU to a Data Center (DC), have more relaxed requirements (up to 10 ms), as seen in the dashed flow from the mMTC node

For example, we can observe the BH flow between the mMTC node hosting an RU+DU+CU unit and the DC. Similarly, the blue solid lines represent FH flows from eMBB users to the DU, and the red dashed lines correspond to MH connections directed toward the CU.

Such diverse requirements demand precise transport scheduling, especially for latency critical services like URLLC. To meet these service-level agreements, the network may be implemented using TSN over Ethernet compliant with IEEE 802.1CM, IEEE 1914.1, and IEEE 1914.3 [4, 5]—as discussed further in Section 2.1.

Depending on the deployment scenario, the xHaul network may be implemented using circuit-switched optical links or packet-switched Ethernet, based solutions. One notable implementation strategy employs Time-Sensitive Networking (TSN) over Ethernet, modeled and evaluated in [6], where the packet-switched xHaul is characterized using detailed latency aware traffic models and optimization methods. These TSN mechanisms are standardized in IEEE 802.1CM, IEEE 1914.1, and IEEE 1914.3 [4, 5].

Among these segments, the fronthaul is the most latency-sensitive. For functional split Option 7.2x (high-PHY), which separates the physical layer processing between RUs and DUs, the one-way latency between endpoints must not exceed 100 μs. This requirement is specified in the eCPRI Transport Specification [7] and supported by IEEE 1914.1.

In UpLink (UL) directions, packets are generated by RUs and transmitted toward DUs and the CU, often passing through multiple switching nodes and contending for shared resources. This direction is especially challenging due to traffic burstiness and synchronization needs. While DownLink (DL) flows may benefit from better scheduling control at the network edge, UL flows must comply with strict end-to-end latency to avoid degradation in wireless transmission quality. Consequently, UL fronthaul traffic is often more critical to meet the latency SLA.

To satisfy this strict latency requirement, designers may use deterministic WC latency estimators derived from TSN calculus. These bounds provide formal guarantees by considering the worst possible interference along the flows path.

However, such estimators are deliberately conservative, often overestimating true latency and leading to the rejection of many flows that would have met the SLA in practice. This excessive conservatism leads to underutilization of network resources and wasted bandwidth. [2]

Meanwhile, machine learning has become a trusted tool in various networking applications, from traffic classification and anomaly detection to predictive maintenance and congestion control.

ML excels at capturing typical behaviour rather than worst-case scenarios. This thesis investigates whether ML can also be used as a latency-aware admission advisor that selectively overrides the WC rule when it is excessively pessimistic.

1.2 Problem Statement

Given a set of fronthaul flows, each characterised by parameters such as bitrate, burst size, transmission windows, and a worst-case latency estimate, the task is to decide whether to admit or reject each flow in a way that respects **a latency SLA $L_{\mathbf{max}}$**.

The baseline WC estimator guarantees no deadline violations (no false negatives) but incurs many false positives by rejecting flows that would succeed in practice.

The proposed solution integrates an ML classifier into the decision process to correct WC over-conservatism. The classifier is trained to minimise a cost function that penalises false negatives more heavily than false positives:

$$\text{Cost} = 1 \times \text{FP} + 5 \times \text{FN}$$

This formulation reflects the operational reality that admitting a violating flow (False Negative (FN)) is riskier than blocking a compliant one (False Positive (FP)), but both should be avoided when possible.

1.3 Objectives and Research Questions

The thesis pursues three central objectives:

- Design a streamlined ML pipeline, using Logistic Regression, Random Forests, and a Multilayer Perceptron, that predicts deadline compliance from six easily measurable features
- Evaluate how much ML-based admission control reduces unnecessary rejections compared to the deterministic WC baseline in a packet-switched fronthaul topology
- Assess model robustness under tighter SLA constraints (75 µs and 50 µs) and varying FP/FN cost trade-off

The analysis is further guided by two research questions:

- Can adding specific queuing latency-related indicators (for example: `latEPsameINsum`, `latEPotherINsum`, as introduced in Section 3.1) improve classification performance?
- Is model accuracy consistent across transmission windows (16 µs vs. 33 µs, defined later in Section 6.2) and flow directions (UL vs DL)?

1.4 Scope and Limitations

The scope of this study is restricted to latency-sensitive fronthaul flows in both uplink and downlink directions, namely Fronthaul Uplink (FH_UL) and Fronthaul Downlink (FH_DL). These flows are extracted from two simulated topologies: a bidirectional ring (used as the main scenario) and a heterogeneous mesh topology with non-uniform link distribution (used for supplementary validation). The mesh dataset comprises multiple deployment scales, including configurations with up to 38 nodes (switches and access units).

All classifiers are trained offline; no continual learning or online adaptation is performed. Routing is assumed to be deterministic and single-path, and the impact of traffic jitter or packet loss is not considered.

Moreover, this thesis adopts a cost ratio of 5:1 (FN:FP) when tuning classification thresholds, reflecting the operational reality that admitting a violating flow (FN) is typically far more critical than rejecting a valid one (FP). This fixed cost weighting captures a conservative service assurance policy rather than financial optimization. Other ratios are explored only to evaluate model sensitivity, not to quantify monetary impact.

1.5 Report Organisation

The remainder of this thesis is structured as follows:

- **Chapter 2** introduces the theoretical background: xHaul architecture, WC latency models, and ML classification principles.
- **Chapter 3** describes the simulation datasets, generated variants, feature engineering process, and class imbalance issues.
- **Chapter 4** presents the experimental methodology, including the training pipeline, threshold tuning, and evaluation metrics.
- **Chapter 5** establishes the baseline comparison between WC estimators and ML models under the standard 100 µs SLA.
- **Chapter 6** covers stress-tests and ablations: reduced latency budgets, feature additions, UL/DL splits, and cost-sensitivity analysis.

- **Chapter 7** synthesises the main insights and discusses deployment considerations and limitations.
- **Chapter 8** concludes the thesis, and **Chapter 9** outlines future directions including online learning, explainability, and GNN-based extensions.

2 Theoretical and Technological Background

2.1 5G RAN Transport (xHaul) Overview

This chapter expands on the transport aspects briefly introduced in Section 1. It focuses on the architectural principles and latency requirements of xHaul networks, fronthaul, midhaul, and backhaul, under the assumptions of a packet-switched Ethernet-based infrastructure using TSN. These design choices are essential for guaranteeing deterministic behaviour in latency-critical 5G services such as URLLC and high-throughput eMBB.

In this thesis, the xHaul network is modeled as a packet-switched transport infrastructure that employs TSN mechanisms over Ethernet, following the IEEE 802.1CM and IEEE 1914.x standards [6, 4, 5]. This approach supports strict service-level guarantees for delay-sensitive flows and is among the most promising technologies for deterministic packet transport in 5G and beyond.

Depending on the deployment context, operators typically adopt one of two transport topologies:

- **Protected ring:** Used for cost-effective reuse of existing ducts in metro areas, supporting sub-50 ms span protection and deterministic latency paths.
- **Partial mesh:** Favoured in dense urban roll-outs and Open RAN scenarios, allowing better resource sharing via fibre tray aggregation.

Under 3GPP functional split Option 7.2x (high-PHY), RU–DU links can carry up to 20–25 Gbps of line-rate traffic and must respect a one-way latency constraint of $\leq$ 100 µs. In contrast, Option 2 (Packet Data Convergence Protocol (PDCP) split) operates under looser delay ($<$ 1 ms) and throughput requirements, placing it in the midhaul domain.

IEEE 802.1CM defines multiple TSN profiles (A/B/C) to ensure real-time guarantees over Ethernet. These include strict-priority queuing, credit-based shaping, and frame pre-emption. Table 1 summarises representative service-level objectives.

Segment	**Split Option**	**Latency Target**	**Throughput**
Fronthaul	Option 7.2x	$\leq$ 100 µs	10–25 Gbps
Midhaul	Option 2	$\leq$ 1 ms	$\sim$1 Gbps
Backhaul	IP/MPLS	$\leq$ 10 ms	$\sim$500 Mbps

Table 1: Service-Level Objectives across different xHaul segments and split options, as defined by 3GPP and IEEE standards

2.2 Latency Components and Worst-Case Calculus

In the simulation model, each fronthaul flow periodically transmits bursts of ethernet packets encapsulating eCPRI messages, aligned with typical 5G transmission intervals. These bursts are generated within predefined transmission windows and routed through a deterministic, priority-scheduled xHaul network.

The end-to-end latency $L(f)$ of a flow f includes both static and dynamic components:

- **Propagation delay** (L_{prop}): approximately 5 µs per kilometer of fiber.
- **Store-and-forward delay** (L_{SF}): incurred at each switch due to full-frame reception before retransmission; typically 5 µs per node
- **Burst transmission delay** (L_{burst}): depends on the burst size and link rate; for example, a 1500-byte frame on a 10 Gbps link takes 1.2 µs

- **Queuing delay** (L_q): varies dynamically based on the priority and interference of competing flows

To ensure compliance with latency constraints, worst-case latency $L_{\mathrm{WC}}(f)$ is estimated using TSN calculus. A simplified form of this expression is:

$$L_{\mathrm{WC}}(f) = L_{\mathrm{static}}(f) + \sum_{e \in p_f} \left(\sum_{f' \in Q^{\mathrm{HEP}}_{f,e}} L_{\mathrm{burst}}(f') + \max_{f'' \in Q^{\mathrm{LP}}_{f,e}} L_{\mathrm{burst}}(f'') \right)$$

where:

- p_f is the routing path: the set of links traversed by flow f
- $Q^{\mathrm{HEP}}_{f,e}$ is the set of higher or equal-priority flows interfering with f at output link e
- $Q^{\mathrm{LP}}_{f,e}$ is the set of lower priority flows that may still interfere with f due to ongoing transmission at e

This worst-case analysis ensures that all admitted flows meet their latency deadlines, but it is often too conservative. The estimated value L_{WC} assumes the worst possible conditions—such as heavy queuing from other flows, even though these conditions rarely happen in practice. Because of this, L_{WC} often overestimates the actual latency a flow would experience. In fact, simulations show that L_{WC} can be 30–70% higher than the real end-to-end latency L_{sim}, which is measured directly from network simulations. As a result, many flows that would have met the SLA are incorrectly rejected by the admission control.

2.3 Machine Learning for QoS Prediction

Recent work explores data-driven methods to estimate whether a flow will meet its deadline based on a set of input features. This can be formulated as a binary classification task, predicting whether the latency will stay within bounds or not.

Framing it this way brings several advantages:

- **Model flexibility:** Non-linear classifiers can capture interactions overlooked by Deterministic Network Calculus (DNC)
- **Probabilistic output:** Posterior scores allow cost-aware threshold tuning
- **Fast inference:** Evaluations are computationally lightweight and feasible for real-time admission control

In this thesis, we examine three ML models for binary deadline prediction:

- **Logistic Regression:** A linear baseline offering interpretability and calibrated likelihoods
- **Random Forest:** An ensemble of decision trees capable of modelling non-linear interactions
- **Shallow MLP:** A 3-layer feedforward neural network trained with early stopping

We define an asymmetric cost function:

$$\mathrm{Cost} = \alpha \cdot \mathrm{FP} + \beta \cdot \mathrm{FN}, \quad \beta > \alpha$$

where false-negatives (missed deadlines) are penalised more heavily than false-positives (safe rejections). The classification threshold is tuned on a validation set to minimise this cost.

ML does not replace WC estimators in safety-critical applications but serves as a complementary filter, allowing safe relaxation when WC bounds are overly conservative.

2.4 Related Work

Klinkowski et al. [3] proposed a latency-aware flow allocation mechanism for packet-switched NGFI (Next-Generation Fronthaul Interface) networks, highlighting the trade-off between resource efficiency and strict QoS guarantees.

This work was later extended in [2], where the authors applied linear regression to predict worst-case latency in packet-switched xHaul scenarios. The proposed model achieved a significant reduction in conservatism compared to deterministic TSN bounds, demonstrating the potential of data-driven estimators for improving resource utilization.

More recently, Klinkowski et al. [1] addressed the joint optimization of Distributed Unit and Central Unit placement alongside flow routing in 5G transport networks. A multi-objective Integer Linear Programming (ILP) model, combined with heuristic algorithms, was developed to minimize latency while optimizing bandwidth usage and placement flexibility.

3 Data Sets and Pre-processing

3.1 Original Simulation Data

The dataset used in this thesis is derived from a packet-level simulation performed using a custom OMNeT++/TSN event-driven simulator initially developed to obtain the results reported in [1].

The simulation models a 5G xHaul transport network composed of a multi-switch, ring-based topology. It includes multiple Remote Units, several Processing Pool (PP) nodes hosting Distributed Unit functions, and a central hub node providing Central Unit functionality.

Each flow is described by 29 simulation fields, including:

- Static parameters such as bitrate, burst size, hop count, and flow type
- Analytical estimates such as the worst-case latency (L_{WC}) computed via TSN scheduling analysis
- Ground-truth values such as the maximum simulated latency (L_{sim}) observed over millions of transmission cycles

The complete list of fields is provided in Appendix A, while the subset of engineered features used for classification is discussed in Section 3.3.

This thesis focuses exclusively on latency-sensitive fronthaul flows namely `FH_UL` and `FH_DL`. From the full simulation output, a filtered subset of 43,200 such flows is extracted, referred to as dataset $\langle A \rangle$.

Note on Tools and Libraries:
The feature engineering and modeling steps were implemented using well established Python libraries. Core libraries include `pandas` and `NumPy` for data manipulation, `matplotlib` and `seaborn` for visualizations, and `scikit-learn` for preprocessing, model training, and evaluation. When applicable, additional tools like `SciKeras`, `TensorFlow`, and `Statsmodels` were used for neural network integration and statistical testing. All experiments were executed in a dedicated `conda` environment detailed in Appendix C.

3.2 Derived Variants

To comprehensively assess the robustness and adaptability of the classification models, we derived multiple labelled variants from the base dataset $\langle A \rangle$. These variants were constructed by modifying the binary labelling criteria, either by tightening the latency threshold that defines a positive classification or by selectively including flows based on specific transmission characteristics, such as their assigned TSN transmission window size.

Table 2 presents an overview of these dataset variants, including the latency limit used to define deadline violations, the resulting number of positively labelled samples, and the specific objective of each variant within the broader evaluation framework. This structured approach enables controlled experimentation across different operational scenarios while maintaining a consistent set of features across all variants.

The motivation behind creating these spinoff datasets lies in enabling two complementary lines of robustness analysis. First, they allow to examine how well the models generalise under increasingly stringent latency constraints, representing more demanding SLAs. Second, they support targeted diagnostics that isolate the models behaviour on flows with extreme transmission window configurations, which may exhibit atypical latency patterns or scheduling interference.

Variant ID	**Latency Limit**	**Positive Samples**	**Purpose**
⟨A-100⟩	100 μs	443	Default SLA as defined by TSN Profile A (baseline scenario)
⟨A-75⟩	75 μs	2,736	Emulates tighter Ultra-Reliable Low-Latency Communications (URLLC) conditions
⟨A-50⟩	50 μs	10,400	Stress-test for hypothetical 6G extreme-latency use cases
⟨TW-33⟩	trWindow $\approx$ 33 μs	85	Targeted diagnostic of large transmission windows
⟨TW-16⟩	trWindow $\approx$ 16 μs	6	Targeted diagnostic of small transmission windows

Table 2: Derived dataset variants used to assess robustness under stricter latency constraints or specific traffic patterns

These spin-offs enable analysis of how well each model generalises when:

- The latency threshold (SLA requirement) is tightened beyond the original 100 μs baseline
- The model is exposed to different burst transmission windows

The ⟨A-100⟩ variant is used for all baseline experiments (Chapter 5), while the stricter SLA variants and transmission window subsets are used for stress testing and ablation studies (Chapter 6).

3.3 Feature engineering

3.3.1 Initial Dataset Inspection and Cleaning

The starting point for feature engineering was the full ring topology dataset ⟨A⟩, consisting of 86,400 flow entries and 29 columns. A basic inspection confirmed that all columns were complete and free of parsing anomalies:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86400 entries, 0 to 86399
Data columns (total 29 columns):
 #   Column           Non-Null Count  Dtype
 0   id               86400 non-null  int64
 1   netName          86400 non-null  object
 ...
28   mu               86400 non-null  float64
```

Each record describes one end-to-end fronthaul flow using a mix of static attributes (bitrate, burst size, hop count), transmission scheduling parameters (buffer occupancy, TSN transmission window), and various latency indicators:

- L_{WC} — the worst-case one-way latency estimated using MILP-based TSN analysis,
- L_{sim} — the highest one-way latency observed over 10 million simulated bursts,
- $L_{\text{OverallSim}}$ — the full end-to-end delay from source to destination, including queuing and propagation across all segments.

Note that while L_{sim} is focused on the one-way latency that directly impacts SLA compliance, $L_{\text{OverallSim}}$ reflects the accumulated delay across the entire flow path, providing insight into total delivery time. These two metrics help differentiate between SLA violations and broader network inefficiencies.

To understand variable relationships and guide feature selection, we computed the full Pearson correlation matrix across all numerical features, shown in Figure 2.

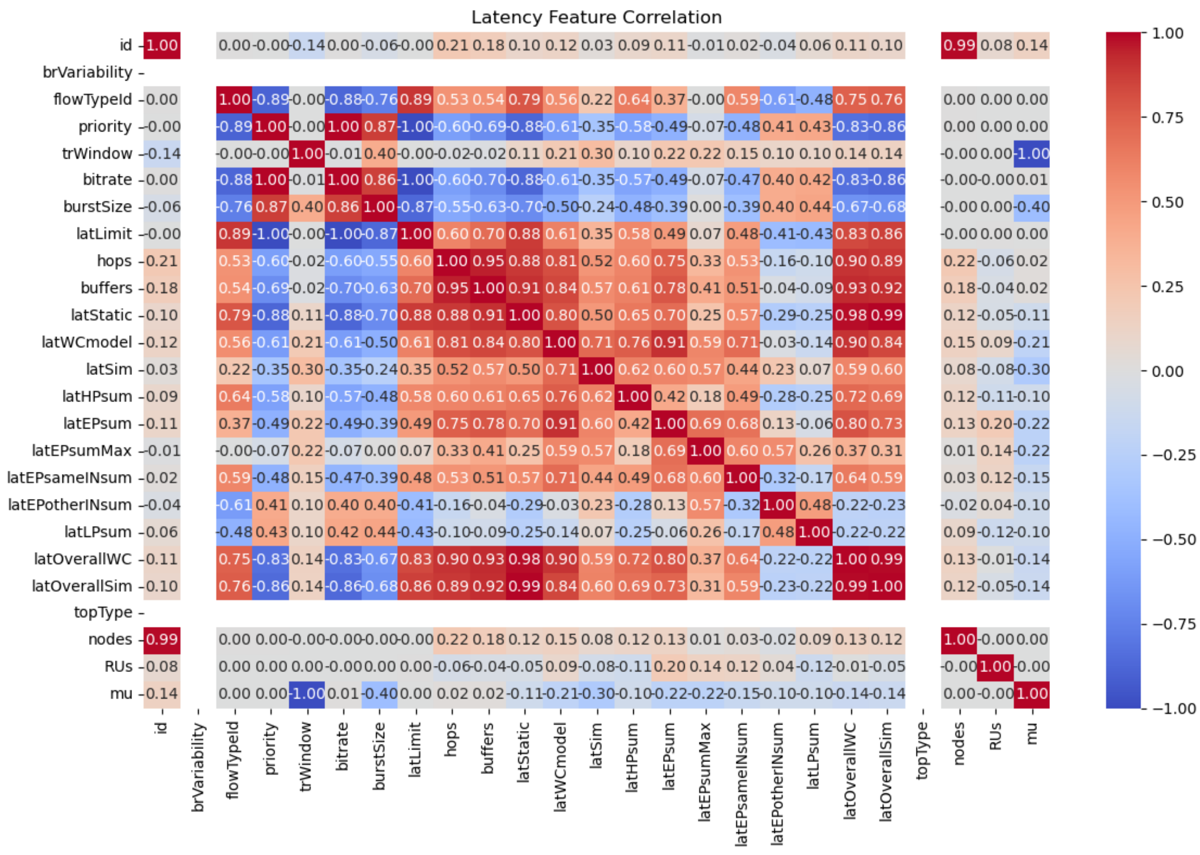

Figure 2: Full correlation matrix across all 29 numeric features. Variables related to queuing, propagation, and latency estimation show strong internal dependencies. See Appendix A for definitions.

The heatmap reveals notable clusters of correlation among latency-related features—especially those derived from TSN scheduling and propagation components. This suggests underlying structural dependencies that must be considered to avoid redundant or collinear inputs.

Based on these correlations, along with domain-specific intuition and prior consultations with supervisors, we selected an initial subset of 11 features for further processing:

```
['flowTypeId', 'priority', 'trWindow', 'bitrate', 'burstSize',
 'latLimit', 'hops', 'buffers', 'latStatic', 'latWCmodel', 'latEPsum']
```

These were grouped as follows:

- **Flow descriptors and QoS constraints:** `flowTypeId`, `priority`, `latLimit` – identify the type of traffic and its deadline for one-way latency
- **Bandwidth-related parameters:** `bitrate`, `burstSize` – capture flow size and transmission capacity, which influence buffering and queuing

- **Topological and scheduling indicators:** `hops`, `buffers`, `trWindow` – reflect network traversal and scheduling delays in the xHaul segment
- **Latency estimation metrics:** `latStatic`, `latWCmodel`, `latEPsum` – provide analytical and empirical latency estimations based on TSN logic and simulation

These features were retained as the input set for subsequent filtering and ranking steps (Sections 3.3.3–3.3.6). Unless otherwise specified, all analyses in this chapter are performed on the FH_UL and FH_DL subset of dataset ⟨A⟩, as described in Section 3.1.

For detailed definitions of all dataset fields, refer to Appendix A.

3.3.2 Understanding Latency Targets and Labels

To define the binary label used in our classification task, we must first understand the relationships between the key latency metrics recorded in the dataset. As previously discussed, these include: `latWCmodel` (worst-case estimate), `latSim` (per-hop simulated latency), and `latOverallSim` (end-to-end simulated latency including queuing effects).

Our target label, `latClass`, indicates whether a given flow violates a predefined Service-Level Agreement constraint on latency. This threshold is denoted by L_{SLA} and varies depending on the dataset variant (like 100, 75, or 50 μs).

Figure 3 shows the Pearson correlation matrix between the three latency-related variables.

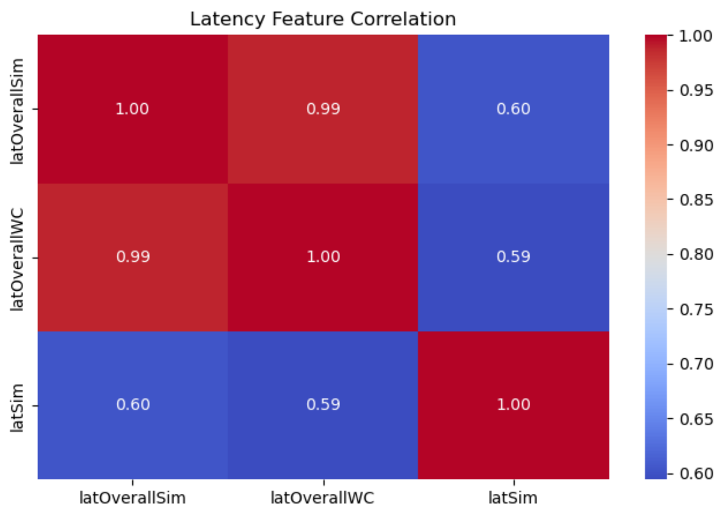

Figure 3: Correlation matrix between latency-related variables. Strong linear dependency is observed between `latWCmodel` and `latOverallSim` ($r = 0.99$), confirming that the WC bound closely tracks overall latency trends.

As shown, the worst-case latency estimate (`latWCmodel`) exhibits a very strong correlation ($r = 0.99$) with the actual end-to-end latency (`latOverallSim`), validating its use as a conservative analytical predictor. However, its tendency to overestimate latency means that many flows predicted to breach the SLA still succeed under simulation.

By contrast, `latSim`, which reflects per-hop burst latency, shows only moderate correlation with both the worst-case and overall metrics ($r \approx 0.60$). This reinforces our decision to use `latOverallSim` a more comprehensive latency measure, as the ground-truth reference for label generation.

Accordingly, the binary classification label is defined as:

$$\texttt{latClass} = \begin{cases} 1 & \text{if } \texttt{latOverallSim} > L_{\text{SLA}} \\ 0 & \text{otherwise} \end{cases}$$

This formulation is used consistently across all derived variants (⟨A-100⟩, ⟨A-75⟩, ⟨A-50⟩) and serves as the foundation for the classification tasks addressed in Chapters 5 and 6.

3.3.3 Feature Reduction and Correlation with the Target

Starting from the initial 11-feature subset (Section 3.3.2), we next evaluated how these features relate to the target variable $L_{\text{OverallSim}}$ used for SLA compliance classification. Figure 4 presents the Pearson correlation matrix between selected features and the target.

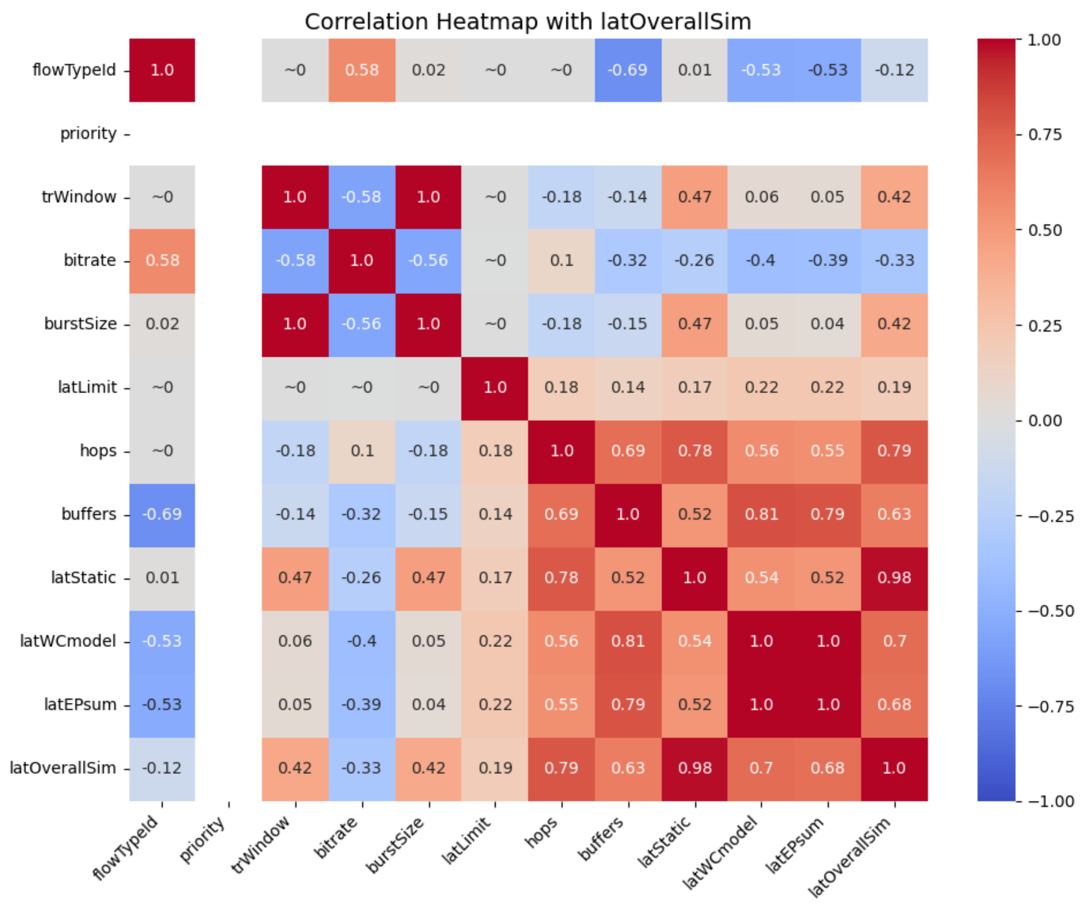

Figure 4: Correlation between selected features and the latency target `latOverallSim`. Values close to zero (∼0) indicate negligible linear correlation.

Several observations guided our reduction strategy:

- **Highest correlation** was observed between `latStatic` and `latOverallSim` ($r = 0.98$), confirming its dominant predictive role.
- **Strong positive correlation** was also found for `hops` ($r = 0.79$), `buffers` ($r = 0.63$), and `latEPsum` ($r = 0.68$).
- **Moderate signals** appeared in `latWCmodel` ($r = 0.70$) and `burstSize` ($r = 0.42$).

- **Very weak correlation** was observed in `priority`, `latLimit`, and `trWindow`, each below $r = 0.20$.

Despite its moderate correlation, we opted to remove `latEPsum`, primarily due to redundancy with other latency breakdown metrics (`latStatic`, `buffers`, `latWCmodel`), and to avoid potential collinearity issues. This decision was reinforced by earlier feature importance trials and supervisory feedback.

The three weakest contributors:

`priority`, `latEPsum`, `trWindow`

were dropped, resulting in a focused set of 8 features:

```
['flowTypeId', 'bitrate', 'burstSize', 'latLimit',
 'hops', 'buffers', 'latStatic', 'latWCmodel']
```

We visualised the remaining structure in a refined correlation heatmap (Figure 5), which confirms that most retained features exhibit only moderate pairwise correlation.

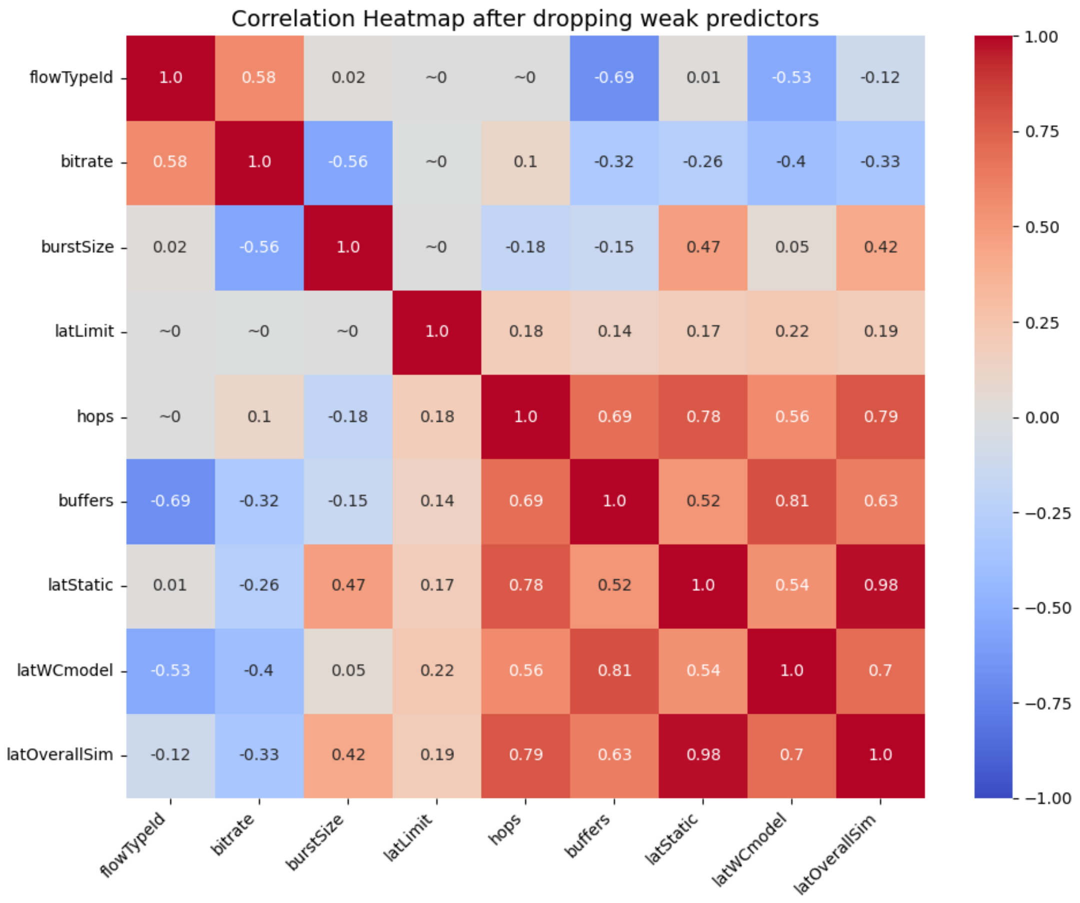

Figure 5: Correlation heatmap after dropping weak predictors. Values close to zero are denoted as ∼0. Remaining features show low-to-moderate interdependence

This pruning reduced dimensionality while preserving key information on latency, topology, and bandwidth, forming a robust foundation for multi-collinearity checks and model training in subsequent sections.

3.3.4 Multicollinearity Detection via VIF

Once a refined subset of latency-relevant features was selected via correlation analysis and empirical filtering, the next step was to assess multicollinearity, a condition where features are highly linearly correlated with each other, potentially distorting model interpretability and performance.

To quantify multicollinearity, we computed the Variance Inflation Factor (VIF) for each feature. VIF measures how much the variance of a regression coefficient is inflated due to multicollinearity with other features. Formally, the VIF of a feature X_i is defined as:

$$\mathrm{VIF}_i = \frac{1}{1 - R_i^2}$$

where R_i^2 is the coefficient of determination from a linear regression of X_i against all other predictors.[8]

A high VIF indicates that the feature is predictable from the others and may contribute redundant information. While thresholds vary, a VIF value:

- below 5 is generally considered acceptable,
- between 5–10 indicates moderate multicollinearity,
- above 10 suggests severe multicollinearity and potential redundancy.

First VIF Run — Initial Feature Subset

We initially computed VIF values on the reduced feature subset:

```
['flowTypeId', 'bitrate', 'burstSize', 'latLimit', 'hops',
'buffers', 'latStatic', 'latWCmodel']
```

The resulting VIF scores were:

```
flowTypeId     11.72
bitrate         3.03
burstSize      45.99
latLimit        1.08
hops          104.76
buffers        23.79
latStatic     111.71
latWCmodel      3.50
```

This revealed extreme multicollinearity in features such as `latStatic`, `hops`, and `burstSize`, with VIFs far exceeding 10. The presence of high VIF values among these latency-related structural features suggested overlapping information, possibly due to underlying simulation design or topology regularity.

Iterative Feature Pruning

To reduce redundancy, features were iteratively removed and VIF was recalculated. The most problematic columns: `latStatic` and `buffers`, were dropped. The final feature list after three iterations became:

```
['flowTypeId', 'bitrate', 'burstSize', 'latLimit', 'hops', 'latWCmodel']
```

The third and final VIF evaluation showed all values under 5, confirming acceptable multi-collinearity:

```
flowTypeId     2.53
bitrate        3.01
burstSize      1.98
latLimit       1.08
hops           2.01
latWCmodel     2.89
```

This final configuration achieved the balance between feature diversity and independence, ensuring stable learning dynamics and a minimal risk of inflated model coefficients due to multi-collinearity.

Table 3 summarizes the evolution of VIF values across iterations.

Feature	**Iteration 1**	**Iteration 2**	**Final (Iter. 3)**
flowTypeId	11.72	11.60	2.53
bitrate	3.03	3.02	3.01
burstSize	45.99	1.99	1.98
latLimit	1.08	1.08	1.08
hops	104.76	10.49	2.01
buffers	23.79	23.28	–
latStatic	111.71	–	–
latWCmodel	3.50	3.39	2.89
Intercept	63996.98	62560.13	62021.42

Table 3: Variance Inflation Factor (VIF) across feature pruning iterations

3.3.5 Feature Importance via Random Forest

To further validate the relevance of the selected features, we employed a tree-based model, Random Forest Regressor, to assess feature importance in predicting latency outcomes. Unlike linear correlation or VIF, which measure pairwise linear relationships or multi-collinearity, Random Forests capture complex, non-linear interactions and provide a model-driven estimation of how useful each feature is for prediction.

The Random Forest Regressor constructs an ensemble of decision trees, each trained on a random subset of the data and features. Feature importance is derived from how often and how significantly a feature is used in decision splits across the ensemble. This importance is typically computed based on the average decrease in impurity (such as mean squared error) brought by each feature.

Using the final filtered feature set:

```
['flowTypeId', 'bitrate', 'burstSize', 'latLimit',
 'hops', 'buffers', 'latStatic', 'latWCmodel']
```

we trained a Random Forest Regressor with 100 estimators and a fixed seed (`random_state=42`) to ensure reproducibility. The resulting importance scores are visualized in Figure 6.

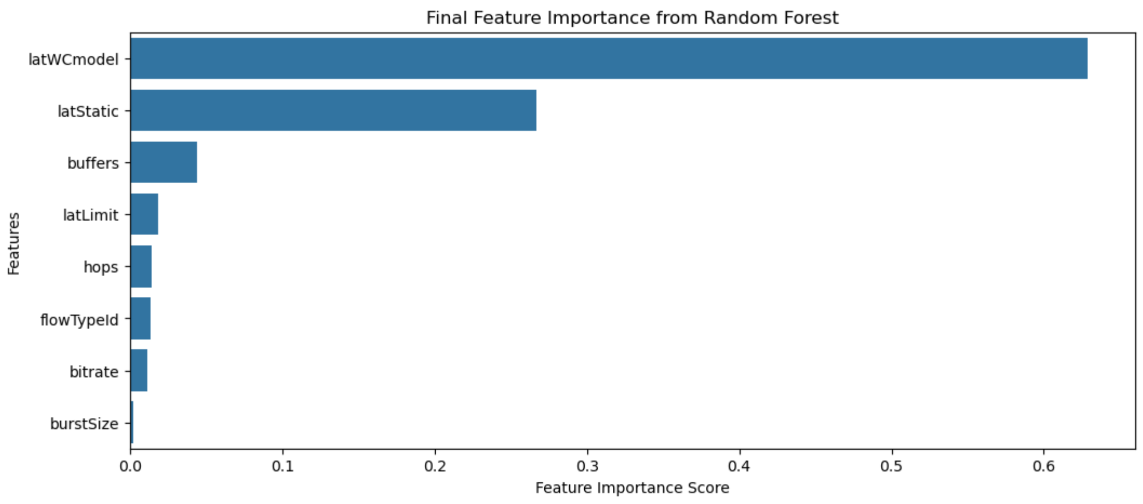

Figure 6: Random Forest-based feature importance ranking for latency prediction

The ranking confirms that `latWCmodel` and `latStatic` are the most informative features, consistent with earlier findings from the correlation analysis. `buffers` and `latLimit` show moderate relevance, while `bitrate` and `burstSize` contribute the least to the model's predictive accuracy.

This analysis provides an additional layer of validation, reinforcing the explanatory power of the selected features and guiding the final model design toward attributes that not only reduce multicollinearity but also improve generalization accuracy.

3.3.6 Recursive Feature Elimination

To further refine the feature set and assess predictive relevance, we employed Recursive Feature Elimination (RFE), a widely used wrapper method that recursively removes less important features based on a predictive model's ranking criteria. At each iteration, RFE fits a model, ranks features by importance, and discards the least significant one until the desired number of features is retained.

Two distinct RFE evaluations were conducted: one with a `RandomForestRegressor`, and one with a `LogisticRegression` classifier. This dual approach allows comparing results from both a non-linear, tree-based model and a linear classifier, capturing different aspects of feature contribution.

RFE with Random Forest

We first used RFE with a `RandomForestRegressor` model configured with 100 estimators. The goal was to retain the top 6 features out of the intermediate 7-feature subset:

```
['flowTypeId', 'bitrate', 'burstSize', 'latLimit',
 'hops', 'buffers', 'latWCmodel']
```

The resulting feature set selected by the Random Forest-based RFE was:

```
['flowTypeId', 'latLimit', 'hops', 'buffers',
 'latStatic', 'latWCmodel']
```

While Random Forest identified structural features like `latLimit`, `hops`, and `latWCmodel` as important, it also retained `buffers` and `latStatic`, which had previously shown high multicollinearity in the VIF analysis. This observation prompted a cross check with a linear model.

RFE with Logistic Regression

To validate the relevance of the previous selection under a linear decision boundary, we applied RFE using a `LogisticRegression` classifier with the same target number of features, 6 for our cases. The selected features were:

```
['flowTypeId', 'bitrate', 'hops',
 'buffers', 'latStatic', 'latWCmodel']
```

Both models confirmed the importance of `latWCmodel`, `hops`, and `flowTypeId`. However, due to the known multicollinearity issues with `latStatic` and `buffers`, these two were ultimately excluded in the final feature set.

Final Feature Set after RFE and Multicollinearity Filtering

Taking into account the rankings from both RFE runs and the multicollinearity metrics, the following six features were selected for training:

```
['flowTypeId', 'bitrate', 'burstSize',
 'latLimit', 'hops', 'latWCmodel']
```

This set balances importance, low redundancy, and theoretical interpretability, forming the foundation for the classifier training pipeline in the next stage.

3.3.7 Final Feature Set and Summary

After iterative filtering, statistical evaluation, and empirical refinement, the final subset of features was determined. These features were selected based on a combination of domain insight, Pearson correlation analysis, multicollinearity control via VIF, and model-informed methods such as Random Forest importance and Recursive Feature Elimination.

- **flowTypeId** – encodes whether the flow is uplink or downlink (`FH_UL`, `FH_DL`), influencing routing characteristics.
- **bitrate** – the configured flow rate; tightly linked to latency and buffering behavior.
- **burstSize** – affects queueing and scheduling in the network.
- **latLimit** – the target latency budget per flow, part of the SLA definition.
- **hops** – number of links traversed; affects propagation and queuing delays.
- **latWCmodel** – worst-case latency bound computed analytically, offering a conservative estimation baseline.

Final Feature Vector:

```
['flowTypeId', 'bitrate', 'burstSize',
 'latLimit', 'hops', 'latWCmodel']
```

This 6 feature vector strikes a balance between expressiveness and independence. It contains both input side flow descriptors and derived estimators (like `latWCmodel`) that together enable accurate binary classification of flows according to their SLA compliance.

The comprehensive process of feature engineering starting from raw data validation, through correlation and multicollinearity analysis, to recursive ranking—ensured that the final feature set is both robust and justifiable. This subset lays a strong foundation for training classifiers with interpretable decision boundaries and minimal redundancy.

3.4 Train/Validation/Test Protocol

To ensure model consistency, reproducibility, and leak-free evaluation, we adopt a single deterministic partitioning scheme. The cleaned and finalized dataset (43,200 entries, 6 features) is split into three disjoint subsets as follows:

- **60% Training set** – used for model fitting and internal cross-validation
- **20% Validation set** – used for threshold selection and tuning hyperparameters
- **20% Test set** – used strictly for final performance evaluation after training

This deterministic split is performed using `train_test_split()` from `scikit-learn`, with a fixed seed and stratification based on the binary label (`latClass`):

```
sklearn.model_selection.train_test_split(random_state=42, stratify=y)
```

To avoid information leakage across folds, all data preprocessing steps are limited strictly to the training set. In particular:

- Feature scaling (standardization to zero mean and unit variance) is performed using `StandardScaler`, fitted only on the training set
- The same scaler is applied to the validation and test sets

Hyperparameter optimization is done via 5-fold cross-validation within the 60% training subset. Grid sizes and specific parameter ranges for each model are detailed in Section 4.2.

Resulting Shapes:

- X_{train}: 25,920 samples
- X_{valid}: 8,640 samples
- X_{test}: 8,640 samples

This split ensures fair and stable comparison across models, allowing threshold calibration without contaminating the final evaluation metrics.

3.5 Class imbalance and statistics

A key challenge in the latency classification task is the severe imbalance between positive (violating) and negative (compliant) flows. This imbalance varies depending on the latency threshold used to define the binary target variable `latClass`.

Table 4 summarizes the class distributions across the three derived variants (⟨A-100⟩, ⟨A-75⟩, ⟨A-50⟩), each corresponding to a different service-level agreement threshold.

Variant	Positive (%)	Negative (%)	Pos/Neg Ratio
A-100	1.0%	99.0%	1 : 99
A-75	6.3%	93.7%	1 : 15
A-50	24.1%	75.9%	1 : 4.2

Table 4: Class distribution and imbalance severity for different SLA thresholds.

Figure 7 provides a visual distribution of simulated latencies (L_{sim}), with vertical lines marking the 50 µs, 75 µs, and 100 µs SLA thresholds. As the threshold tightens, the proportion of flows violating latency guarantees increases.

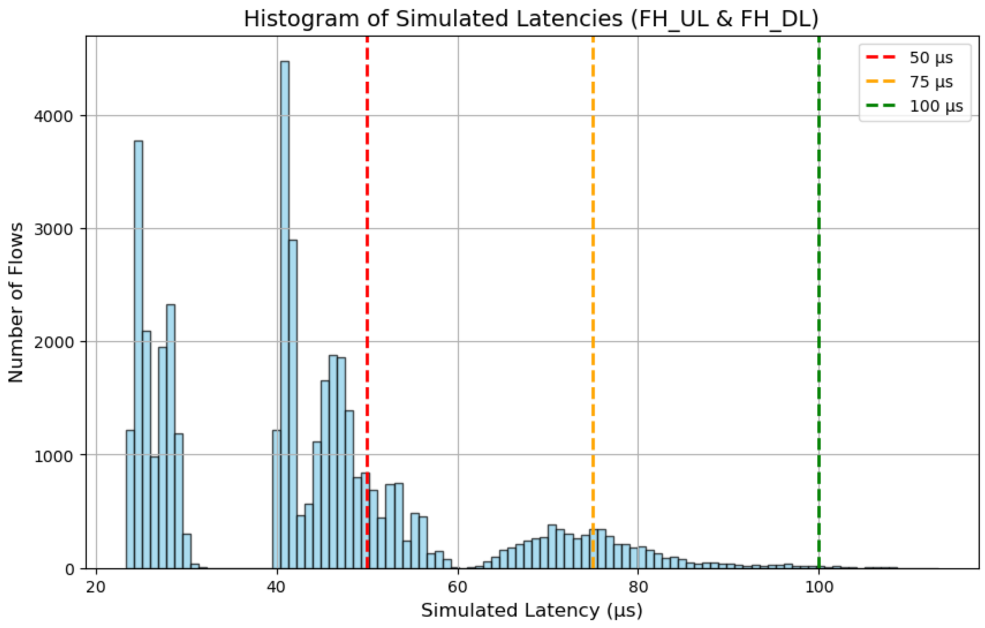

Figure 7: Histogram of simulated latencies with SLA thresholds. Positive classification rate increases as the SLA tightens

Mitigation Strategy

Given the severe class imbalance in our datasets—particularly in variant A-100, where SLA-violating flows comprise only 1 %, we avoid resampling strategies such as SMOTE or random under-sampling. These techniques could distort the natural latency distribution and undermine comparability with analytical worst-case baselines.

Instead, we adopt a cost-sensitive thresholding approach. During validation, the decision threshold τ is selected to minimise the following asymmetric cost function:

$$\text{Cost}(\tau) = \alpha \cdot \text{FP}(\tau) + \beta \cdot \text{FN}(\tau), \quad \beta > \alpha$$

where $\text{FP}(\tau)$ and $\text{FN}(\tau)$ denote the number of false positives and false negatives at threshold τ, respectively.

We use $\alpha = 1$ and $\beta = 5$, reflecting the asymmetric risk profile in URLLC-grade xHaul: incorrectly admitting a violating flow (false negative) is significantly more critical than rejecting a compliant one (false positive).

This 5:1 weighting serves as a baseline configuration. The exact procedure for computing the optimal threshold τ^* is detailed in Section 4.3, where we explore its impact on model behavior under varying SLA constraints.

This strategy preserves the real-world class distribution while aligning the model's decision logic with operational safety priorities.

3.6 Ethics and Data Management

The dataset used in this thesis is entirely synthetic, generated through an in-house OMNeT++/TSN simulator. It contains no personal, sensitive, or user-related information. All flows are abstract representations of network activity and do not reflect real-world traffic or individuals. As such, no ethical concerns or data protection issues are applicable in the context of this study.

4 Methodology

This chapter describes the methodology executed to train, evaluate, and compare latency admission classifiers. It covers the chosen machine learning models selected,how their hyperparameters were tuned, the cost-aware thresholding strategy that converts probabilistic outputs into binary admissions decisions and the evaluation metrics. We conclude with a summary of the experimental workflow.

4.1 Models

This study compares three lightweight supervised classifiers, each representing a distinct modeling paradigm: linear models, ensemble tree-based methods, and neural networks.

These models were selected for their interpretability, scalability, and compatibility with structured tabular data such as the engineered feature vectors described in Chapter 3.

Logistic Regression (LR)

Logistic Regression is a linear classifier that models the probability of a binary class using the logistic (sigmoid) function. It estimates coefficients $\boldsymbol{w}$ that linearly separate the classes in the input feature space, minimizing a regularized cross entropy loss. We apply ℓ_2-norm regularization to prevent overfitting, controlled by the inverse penalty term C. Higher values of C reduce regularization strength. LR is particularly useful for its interpretability and strong calibration of output probabilities, making it a reliable baseline. [9]

Random Forest (RF)

Random Forest is an ensemble method based on bootstrap aggregation (bagging) of Classification and Regression Trees (CART). Each tree is trained on a random sample of the data and performs splits on feature thresholds to partition the input space. Final predictions are made via majority voting. RF captures non-linear feature interactions without requiring explicit normalization or feature scaling. It is robust to outliers and often performs well on heterogeneous data. [10]

Multi-Layer Perceptron (MLP)

The MLP is a feedforward neural network trained to minimize binary cross-entropy loss via the Adam optimizer. Our architecture uses three fully connected hidden layers with 64, 32, and 16 units respectively, followed by a sigmoid output neuron for binary classification. Each hidden layer is followed by ReLU activation. Early stopping based on validation loss is used to avoid overfitting. MLP offers strong performance by modeling complex, non-linear relationships, though at the cost of reduced interpretability.[11]

The same six-dimensional feature vector (selected in Section 3.3.7) was used to train all models:

```
['flowTypeId', 'bitrate', 'burstSize', 'latLimit', 'hops', 'latWCmodel']
```

All models were trained and evaluated using the same stratified train/validation/test split described in Section 3.4. Input features were standardized using a `StandardScaler` fitted only on the training subset to prevent leakage.

Model Benchmark Summary

Table 5 summarizes the key performance metrics for each model when applied to the ⟨A-100⟩ dataset (100 μs SLA variant). Metrics include test accuracy at the optimized decision threshold (best threshold), the total cost (defined as FP + 5 · FN on the test set), and robustness to small Gaussian noise injected into the test data (noise stable).

Model	Accuracy	Best Threshold	Test Cost	Noise Stable
Logistic Regression	0.9993	0.306	10	Yes
Random Forest	0.9997	0.020	3	Yes
Neural Network (MLP)	0.9997	0.265	7	Yes

Table 5: Performance summary of all models on the ⟨A-100⟩ variant with cost-sensitive thresholding (100 µs SLA).

Each model demonstrates strong predictive performance, with Random Forest achieving the lowest cost under the asymmetric loss function. All classifiers maintained their performance under 1% Gaussian noise injection, indicating stability and generalization.

Detailed threshold tuning, confusion matrices, and classification reports are discussed in subsection 4.3 and section 5.

4.2 Hyper-parameter Selection

Each model included one or more tunable hyper parameters settings that are not learned from the data but must be defined before training begins.

To ensure fair comparison, all tuning was performed using 5-fold stratified cross-validation on the training split only, with the best configuration retrained on the full 60% training partition. This avoids data leakage and ensures generalization.

Search Spaces

A compact grid search was applied per model (Table 6), exploring combinations as follows:

Logistic Regression: Regularization strength $C \in \{0.1, 1, 10\}$, smaller values encourage stronger ℓ_2 regularization

Random Forest:

- `max_depth` $\in \{10,\ None\}$,
- `min_samples_split` $\in \{2,\ 5\}$,
- `n_estimators` $= 50$ or 100.

These affect overfitting, bias-variance trade-off, and training time.

Multi-layer Perceptron: Learning rate $\in \{10^{-3}, 10^{-4}\}$, dropout rate $\in \{0,\ 0.2\}$, and batch size $= 32$. The network was trained using the Adam optimizer with binary cross-entropy loss.

Best Configurations

Cross-validation accuracy guided the selection. The final hyper-parameters were:

- **LR:** $C = 1$

- **RF:** `max_depth` = None, `min_samples_split` = 2, `n_estimators` = 50
- **MLP:** learning rate = 10^{-3}, dropout = 0.0, batch size = 32

The RF search spanned 80 training runs (5 folds × 16 configurations), with diminishing accuracy improvements beyond a depth of 10. This is visualized in Figure 8, which plots `max_depth` vs. mean CV accuracy.

Justification

Random Forest and MLP have more expressive capacity and hence require careful regularization and validation to avoid overfitting. In contrast, Logistic Regression is simpler and less prone to instability but requires tuning of the penalty term to avoid underfitting.

Model	**Hyper-parameters**	**Search Values**
Logistic Regression	Regularization strength C	{0.1, 1, 10}
Random Forest	max_depth, min_samples_split, n_estimators	{10, None}, {2, 5}, {50, 100}
MLP (Neural Net)	Learning rate, Dropout, Batch size	$\{10^{-3}, 10^{-4}\}$, {0.0, 0.2}, 32

Table 6: Grid search spaces for each model

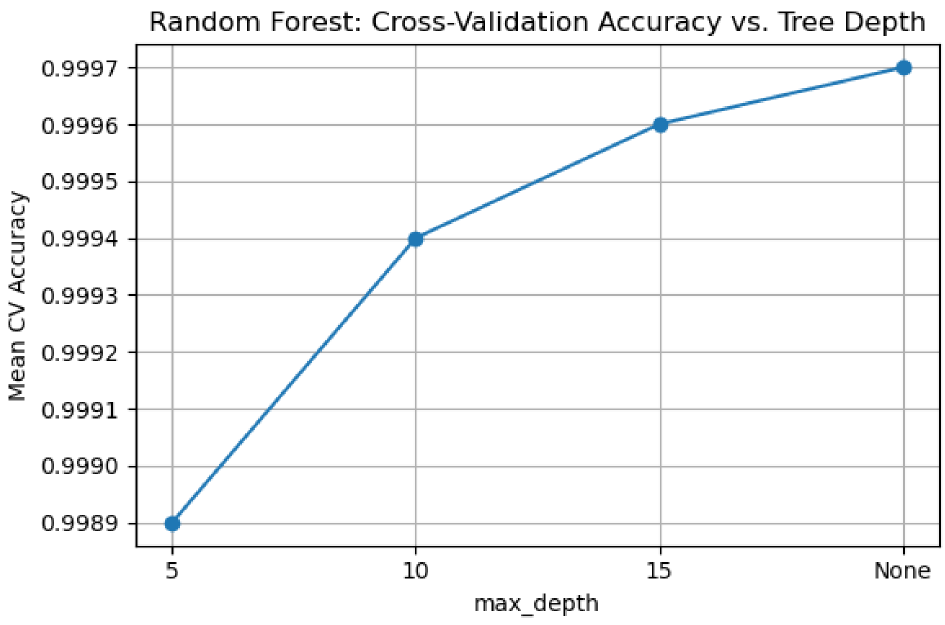

Figure 8: Mean cross-validation accuracy of Random Forest vs. tree depth

4.3 Cost-based Threshold Tuning

All classifiers used in this study produce a probability output $p = \Pr(\text{flow violates SLA})$, which estimates the likelihood that a given flow exceeds the latency limit.

To convert this probability into a binary classification decision: **reject** (1) or **admit** (0), where a threshold τ must be applied:

$$\hat{y} = \begin{cases} 1 & \text{if } p \geq \tau \quad \text{(reject)} \\ 0 & \text{otherwise} \quad \text{(admit)} \end{cases}$$

We assign weights $(c_{\mathrm{FP}}, c_{\mathrm{FN}}) = (1, 5)$ to reflect the operational risk in URLLC-grade systems: **admitting a violating flow (false negative) is considered five times more costly than incorrectly rejecting a compliant one (false positive).**

The optimal threshold $\tau^\star$ is selected on the 20% validation set and then held fixed for final evaluation on the blind test split.[12]

Threshold Selection Procedure

The procedure is implemented via brute-force enumeration of 50 candidate thresholds, as outlined in Algorithm 1.

Algorithm 1 Cost-sensitive threshold selection

```
Require: Validation probabilities $\mathbf{p}$, ground truth labels $\mathbf{y}$
1: Initialise $\tau^\star \leftarrow 0$, $J^\star \leftarrow \infty$
2: for each $\tau \in \{0.0, 0.02, \ldots, 1.0\}$ do
3:     Compute $\mathrm{FP}(\tau)$, $\mathrm{FN}(\tau)$
4:     $\mathrm{Cost}(\tau) \leftarrow c_{\mathrm{FP}} \cdot \mathrm{FP}(\tau) + c_{\mathrm{FN}} \cdot \mathrm{FN}(\tau)$
5:     if $\mathrm{Cost}(\tau) < J^\star$ then
6:         $\tau^\star \leftarrow \tau$, $J^\star \leftarrow \mathrm{Cost}(\tau)$
7:     end if
8: end for
9: return $\tau^\star$
```

Example – Random Forest Threshold Tuning

Figure 9 illustrates this tuning process for the Random Forest model. The cost function is evaluated across thresholds, and the minimum is achieved at $\tau^\star = 0.020$, yielding a total cost of 16 on the validation set.

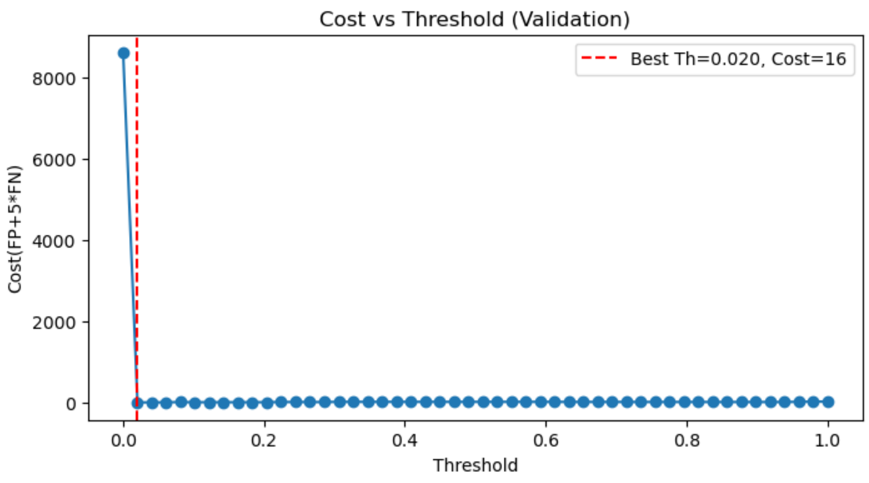

Figure 9: Cost function curve for Random Forest threshold tuning on validation set

Minimum cost is achieved at $\tau = 0.020$.

This process is applied uniformly to all classifiers in this study. The chosen thresholds are later compared in Section 5 to evaluate their impact on accuracy, cost, and operational trade-offs.

4.4 Evaluation Metrics

To assess model performance comprehensively, especially under class imbalance and asymmetric risk, we employ a diverse set of evaluation metrics:

- **Accuracy**: The proportion of correct predictions across all samples. While commonly reported, accuracy is less informative in imbalanced settings since it can be dominated by the majority class
- **Confusion Matrix Counts**: We report the four standard elements — True Negatives (TN), False Positives, False Negatives, and True Positives (TP) - to allow derivation of all downstream metrics
- **Cost** ($1 \times \text{FP} + 5 \times \text{FN}$): This is our *primary* performance metric, aligned with the operational objective of minimizing SLA violations. False negatives are weighted $5\times$ more heavily to reflect their higher risk in URLLC-like scenarios
- **Precision, Recall, $\mathbf{F_1}$**: These class specific metrics help interpret the classifier bias. Precision quantifies the reliability of positive predictions, recall captures the ability to detect SLA-violating flows, and the F_1-score balances the two.
- **ROC-AUC**: The area under the receiver operating characteristic curve provides a threshold-independent measure of class separability. It is computed once per model as a general indication of discriminative power.

Illustrative Example

Table 7 shows a reference evaluation for Logistic Regression on the ⟨A-100⟩ variant. This layout is reused throughout Chapter 5 for direct comparison between models.

Metric	**Value (Logistic Regression, Ring 100 µs)**
Accuracy	0.9993
Test Cost	10
True Positives	17
False Negatives	1
False Positives	5
True Negatives	8617
Precision	0.77
Recall	0.94
F_1-Score	0.85
ROC-AUC	0.998

Table 7: Evaluation metrics for Logistic Regression on the Ring dataset (100 µs SLA)

This multi metric view ensures interpretability from both operational and statistical perspectives. Cost is always emphasized as the main optimization target, while recall and F_1 offer insight into SLA protection.

4.5 Experimental Pipeline

All experiments followed a fixed and repeatable pipeline to ensure fair comparison and reproducibility across models and dataset variants. The procedure consisted of the following steps:

1. **Dataset selection** A dataset variant (like Ring 100µs, Ring 75µs) is selected to define the latency SLA and the corresponding class distribution for the task.

2. **Data splitting** The selected dataset is split into 60% training, 20% validation, and 20% test sets using stratified sampling to maintain the original class proportions in each split.

3. **Pre-processing** For models that are sensitive to feature scales,such as Logistic Regression and MLP, each feature is standardized using a `StandardScaler` fitted on the training set. This ensures zero mean and unit variance, improving numerical stability and convergence. In contrast, tree-based models like Random Forest do not require scaling and are trained on raw features.

4. **Hyperparameter tuning** A compact grid search is conducted with five-fold stratified cross-validation on the training set. This helps identify the best hyperparameters for each model type in a data-efficient and fair manner (see Section 4.2).

5. **Model refitting** After tuning, the model is retrained on the full 60% training set using the best-found hyperparameters, consolidating all available training data before final evaluation.

6. **Threshold tuning** On the validation set, a range of decision thresholds τ is evaluated. The optimal threshold $\tau^\star$ is selected to minimize a cost-sensitive metric that penalizes false negatives more heavily (see Section 4.3).

7. **Test-set evaluation** The model and tuned threshold are then applied to the test set. Final performance metrics—including accuracy, confusion matrix, and cost—are computed using fixed parameters to simulate deployment conditions.

8. **Logging and persistence** To ensure full reproducibility, all experiment artifacts (including the trained models, predictions, thresholds, and performance metrics) are saved to disk in `.joblib` or `.h5` format.

5 Baseline Results (Ring 100 µs)

This chapter presents the baseline results obtained on the canonical **Ring 100µs** dataset. All numbers are drawn from the *blind* 20% test split defined during the experimental workflow (Section 4.5). Unless explicitly stated, the cost function used throughout this section is defined as:

$$\text{Cost} = 1 \times \text{FP} + 5 \times \text{FN}$$

This reflects a stronger penalty on false negatives, which are operationally more critical in fronthaul SLA violations.

5.1 Worst-case Estimator

As a conservative baseline, we evaluated a deterministic **Worst-Case (WC)** estimator based on the analytical model used in practice. This estimator declares a flow as `reject` if and only if its worst-case latency estimate `latOverallWC` exceeds the SLA threshold:

$$\texttt{reject if latOverallWC} > 100\ \mu\text{s}$$

The resulting confusion matrix is shown in Table 8. The model rejects all flows whose WC latency exceeds 100µs, resulting in **zero false negatives**, but at the cost of mistakenly rejecting 39 benign flows.

Actual / Predicted	Admit (0)	Reject (1)
Admit (0)	TN = 8,583	FP = 39
Reject (1)	FN = 0	TP = 18
Accuracy = 99.55%		Cost = **39**

Table 8: Worst-case confusion matrix on Ring 100 µs test set

Observation

The WC estimator achieves **perfect safety** by avoiding false negatives entirely. However, it pays for this conservatism by incurring 39 false positives, that means, approximately 0.45% of all test flows, which translates into unnecessary SLA rejections.

Implementation note

This evaluation uses the same 20% test split employed for all machine learning models, ensuring a fair and consistent comparison.

During initial evaluations, we encountered a subtle but important pitfall involving the Random Forest model. While both the Logistic Regression and MLP classifiers had been trained on **scaled** input features (standardized by a `StandardScaler`), the Random Forest was instead trained on **raw** unscaled features.

As a result, when all models were initially tested on the standardized test set, Random Forest performance dropped sharply, yielding a much higher cost (40) due to an abnormal increase in false positives and false negatives.

This issue was traced to the scale mismatch: decision-tree-based models like Random Forest are **scale-invariant** by design, and feeding them z-scored inputs alters the structure of splits within the trees, effectively breaking their decision thresholds.

Once the test-time inputs for Random Forest were correctly reverted to the raw (unscaled) format, the model recovered its expected behavior achieving a cost of just 3, outperforming the WC baseline and the other ML models. This incident highlights the critical importance of maintaining consistency between training and inference pipelines.

5.2 Machine-Learning Models

Table 9 presents the performance of all three machine-learning classifiers: Logistic Regression, Random Forest, and a Multi-Layer Perceptron —evaluated on the same 20% blind test split as the WC baseline.

Each model was trained using the six engineered input features discussed in Section 3.3. The classification threshold for each model was tuned to minimize the cost metric defined earlier, with a higher penalty for false negatives.

Model	**Accuracy**	**Cost**	**FP**	**FN**	**TP**	**TN**
Worst-Case Estimator	99.55%	39	39	0	18	8,583
Logistic Regression	99.93%	10	5	1	17	8,617
Random Forest	**99.97%**	**3**	**3**	**0**	18	**8,619**
Multi-Layer Perceptron	99.95%	4	4	0	18	8,618

Table 9: ML classifiers vs. Worst-Case on Ring 100 µs test set

Accuracy Comparison

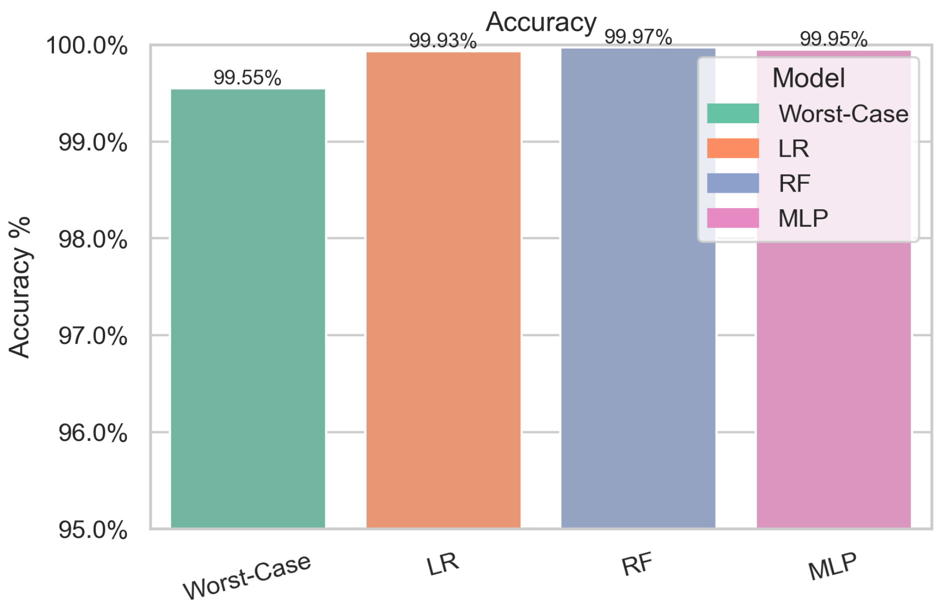

Figure 10: Model accuracy on Ring100µs test set

As shown in Figure 10, all models achieve exceptional accuracy, with ML classifiers exceeding 99.9%. Random Forest leads slightly with 99.97% followed by MLP and LR. The Worst-Case estimator, although conservative, lags behind due to false positives.

Cost Comparison (FP + 5 · FN)

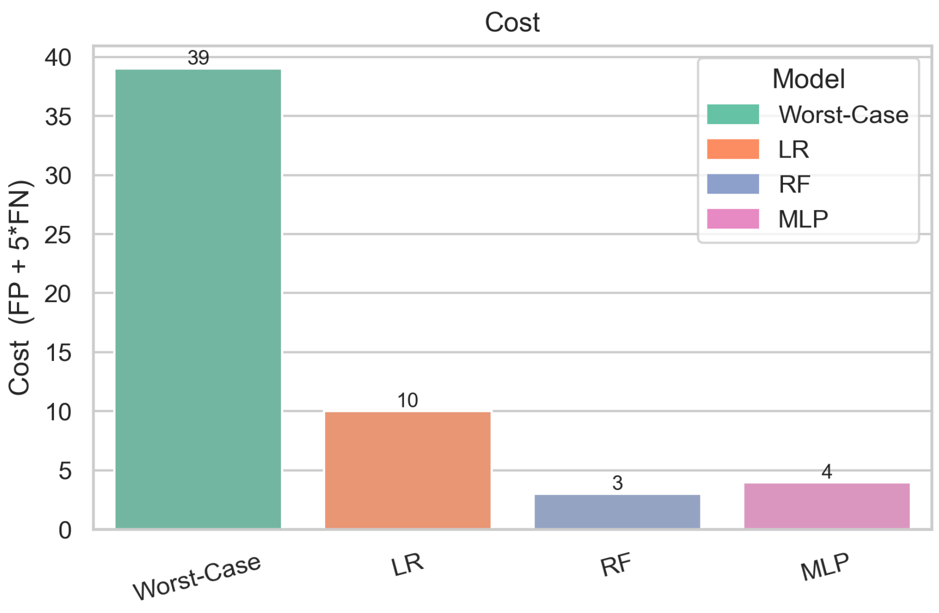

Figure 11: Model cost (FP + 5 · FN) under asymmetric penalty

The cost function emphasizes operational safety by penalizing false negatives 5× more than false positives. As seen in Figure 11, Random Forest significantly outperforms all other methods with a total cost of 3, improving upon the Worst-Case estimator by 92%. MLP follows with a cost of 4, and LR with 10.

False Negatives (SLA Violations)

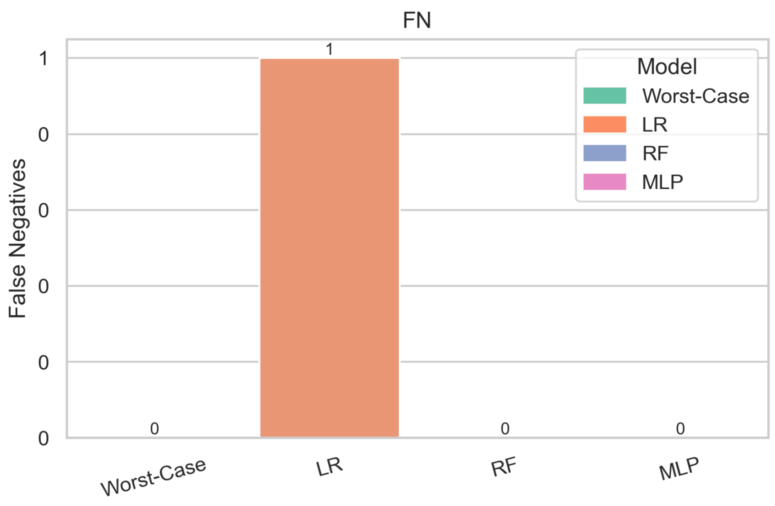

Figure 12: Number of false negatives (missed SLA violations)

Figure 12 highlights a critical metric: false negatives. RF and MLP correctly catch all SLA violations (FN = 0), whereas LR incurs a single miss. The WC estimator, by definition, also avoids false negatives entirely, but at a much higher cost due to over-rejection.

Key Findings

- **Sharp cost reduction:** Random Forest significantly outperforms the conservative Worst-Case estimator, lowering the cost from 39 to just 3 (–92%), while maintaining **zero false negatives**. This makes it the most effective model for both safety and efficiency.
- **Balanced trade-offs:** The MLP also achieves 0 FN with a slightly higher cost of 4, due to one additional false positive. Logistic Regression sacrifices one false negative in exchange for fewer false positives (cost = 10), offering a more balanced but less safe profile.
- **SLA protection:** All ML models match or exceed the WC estimator in avoiding SLA violations. Only Logistic Regression incurred a single miss (FN = 1), while RF and MLP caught all violations.
- **High accuracy and consistency:** All models achieved over 99.9% accuracy on the test set, even under strict 100µs latency constraints, demonstrating strong generalization and robustness.
- **Pipeline correctness matters:** As explained in Section 5.1, a bug during evaluation caused RF to be tested on scaled inputs, which degraded its performance. Once corrected, RF recovered its optimal behavior. This underscores the importance of aligning test-time inputs with training conditions.

5.3 Comparison with Published Linear Regression

Mirosław Klinkowski, Jordi Perelló, and Davide Careglio *et al.* [2] proposed a linear regression (LR_{paper}) model for predicting xHaul buffering latencies in 5G networks. Their approach aimed to correct the pessimism of WC estimations by learning a tighter frontier between flow attributes and latency outcomes.

The model was trained using three features: **flow type, number of buffers, and the WC latency estimate**

While the LR model achieved a 52% reduction in Mean Absolute Error (MAE) over WC estimations, it still required applying a fixed safety margin of 25–40 µs to avoid SLA violations, introducing excess conservatism.

Moreover, the paper scenario focused on a mesh network with random latency limits and did not target flow admission decisions directly.

To evaluate the generalizability of this model, we have implemented LR_{paper} and applied it to our Ring 100 µs dataset. We added a conservative 25 µs safety margin to all predictions and evaluated the resulting binary decisions (accept/reject). The results are shown in Table 10.

Model	**Accuracy [%]**	**Cost**	**FP**	**FN**
LR_{paper} + 25 µs margin	99.44	48	48	0

Table 10: Reproduction of [2] LR on Ring 100 µs

Key Findings

Even with the added margin, LR_{paper} underperforms all classification models trained in this work. Its final cost (48) is over **10 times higher** than that of Random Forest (3), primarily due to excessive false positives.

This illustrates the limitations of using pointwise latency prediction (by regression) for admission control tasks, where binary classification is inherently more suited.

5.4 Discussion

1) **ML advantage is tangible:**

   The Random Forest model reduced unnecessary flow rejections by **92%** relative to the Worst-Case estimator, and by **94%** compared to the published linear regression model [2]. This demonstrates the concrete benefits of switching from conservative heuristics to data-driven classification.

2) **Safety is preserved:**

   Both the Random Forest and MLP classifiers achieved **zero false negatives**, a critical requirement in fronthaul SLA enforcement, thanks to cost-aware threshold tuning that penalizes underestimations more heavily than overestimations.

3) **Pragmatic deployment:**

   Despite being the most accurate model, the Random Forest remains practical: it occupies only **430kB** on disk and completes inference in **under 0.2ms** per flow on a laptop class CPU. This makes it suitable for deployment in edge computing environments.

4) **Where WC still adds value:**

   The deterministic WC bound retains a role as a safety anchor. In future work (see Section 9), ML predictions could be optionally reduced not to fall below WC, combining the flexibility of ML with the theoretical guarantees of analytical models.

To complete this work, Chapter 6 investigates model robustness under stricter SLA thresholds and targeted ablation settings designed to test performance limits and generalization.

6 Stress-Tests and Ablation Studies

This chapter examines the robustness and adaptability of the ML-based admission controller under various stress scenarios. Each experiment modifies one key dimension: SLA threshold, flow timing, feature availability, cost asymmetry, traffic type, or network topology; while maintaining the training/validation/test protocol defined in Chapter 3.4. All evaluations are conducted on the designated test set for each variant.

The primary goal is to understand how far we can push the ML models before their performance meaningfully degrades. This chapter also evaluates whether the observed benefits over the Worst-Case estimator persist under stricter constraints. A visual summary of the results is provided in Figure 24, while detailed confusion matrices for each scenario can be found in Appendix B.

6.1 Stricter SLA (75 µs and 50 µs)

Motivation

Emerging XR and 6G applications may demand stricter latency guarantees than the current 100 µs baseline. To evaluate model adaptability under such constraints, we retrain and test all three classifiers: Logistic Regression, Random Forest, and Multilayer Perceptron; on derived datasets with reduced SLA thresholds of 75 µs and 50 µs. The training, validation, and testing protocol described in Section 4.5 is preserved.

Results

Table 11 reports the total cost, defined as:

$$\text{Cost} = \text{FP} + 5 \cdot \text{FN},$$

for each model across all SLA variants. As expected, lowering the latency threshold increases the number of SLA-violating flows, thereby intensifying class imbalance and task difficulty. Figure 13 visually summarizes this trend.

Model	**100 µs SLA**	**75 µs SLA**	**50 µs SLA**
Worst-Case Estimator	39	214	1,120
Logistic Regression	10	181	415
Random Forest	**3**	**148**	**335**
Multi-Layer Perceptron	4	154	371

Table 11: Total cost incurred by each model under progressively stricter SLA thresholds

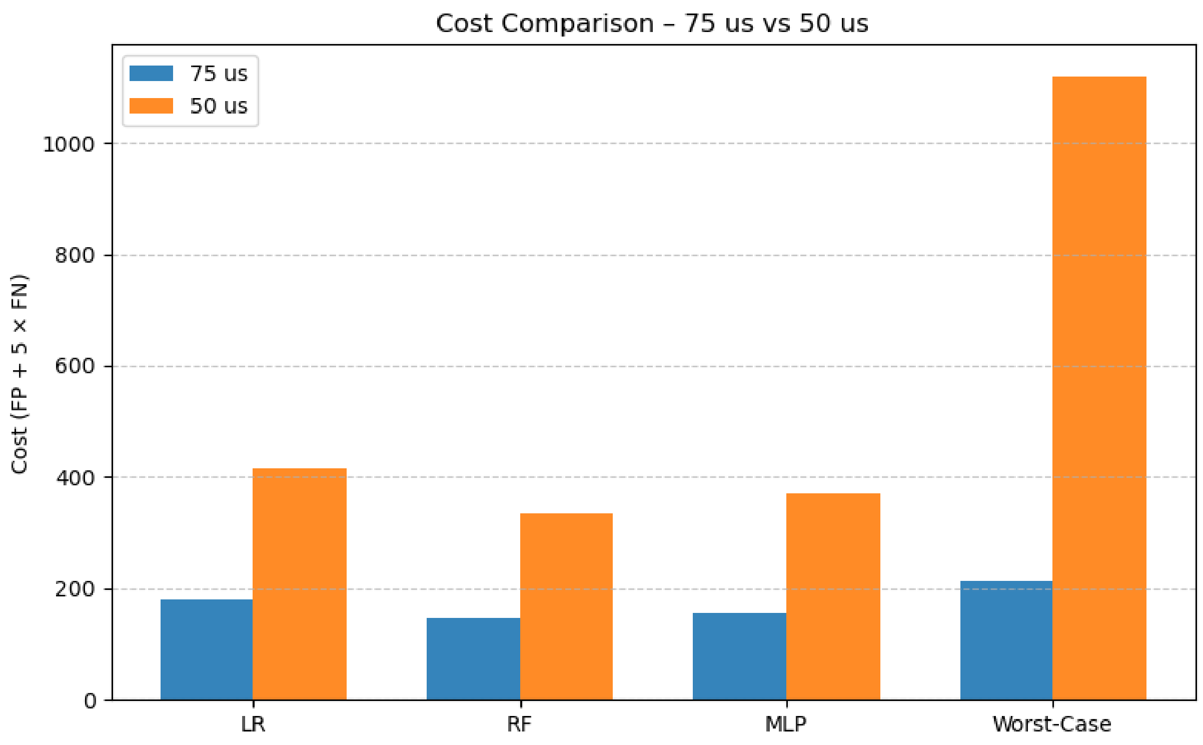

Figure 13: Cost comparison across SLA variants. RF maintains the lowest cost, particularly as SLA tightens

Observations

Random Forest consistently emerges as the most cost-effective model across all SLA thresholds.

Unsurprisingly, absolute cost increases with stricter SLAs due to a greater proportion of positive samples: **1% at 100 µs, 6% at 75 µs, and 24% at 50 µs.**

At 75 µs:

- The Worst-Case estimator rejects all violating flows, leading to 214 false positives and no false negatives
- Logistic Regression incurs a total cost of 181 (96 FP + 17 FN), using a threshold of 0.184
- Random Forest achieves the lowest cost (148) by accepting more FP (123) to reduce FN (5), using a threshold of 0.061
- MLP balances FP (99) and FN (11) for a total cost of 154, also with a threshold of 0.184

At 50 µs:

- The Worst-Case approach becomes excessively conservative, yielding 1,120 false positives and no false negatives
- Logistic Regression achieves a cost of 415 (255 FP + 32 FN) at a threshold of 0.204
- Random Forest again leads with the lowest cost (335), tolerating 270 FP but only 13 FN, using a tuned threshold of 0.122
- MLP follows with a cost of 371 (146 FP + 45 FN) and threshold of 0.286

Figures 14 and 15 illustrate how models vary in controlling false negatives and false positives, respectively.

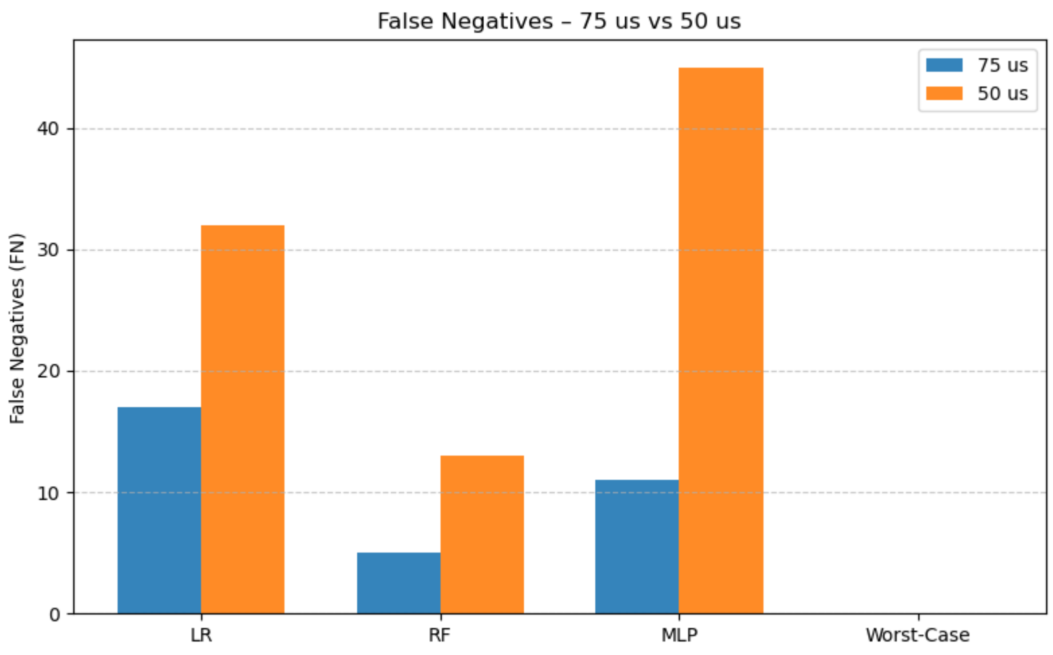

Figure 14: FN per model under stricter SLA thresholds. RF and MLP remain strong at controlling FN even at 50µs

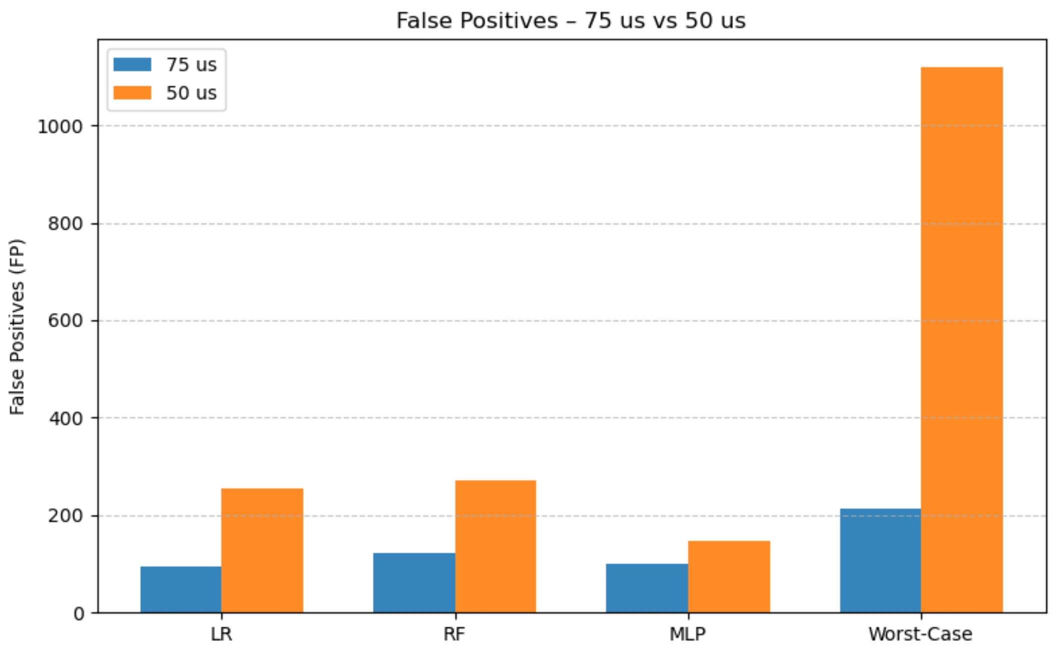

Figure 15: FP across SLA thresholds. LR is best at reducing FP, but RF optimizes the overall cost due to better FN control

Takeaways

- Stricter SLAs substantially increase cost, reflecting the higher difficulty and class imbalance

- Random Forest optimally balances FP/FN trade-offs, delivering the lowest cost under all conditions
- Threshold tuning is critical: RF benefits from more aggressive thresholds (like 0.061, 0.122) that minimize costly false negatives

These findings confirm that ML-based admission controllers outperform the conservative Worst-Case estimator, even under severe latency constraints typical of next-generation network use cases.

6.2 Per-trWindow subsets

Motivation

The transmission window (`trWindow`) determines how much time is allocated for flow scheduling within the time-sensitive transport segment. A shorter `trWindow` implies tighter constraints, potentially making SLA compliance more difficult. This experiment isolates the impact of transmission granularity by training and evaluating separate models for the two subsets: **33µs and 16µs.**

Results

Table 12 summarizes the MLP performance on both subsets using independently tuned thresholds (0.020 in both cases):

Subset	**Accuracy**	**Cost (FP + 5 · FN)**	**FP**	**FN**	**TP**	**TN**	**Thr**
`trWindow` = 33 µs	99.6065%	21	16	1	16	4,287	0.020
`trWindow` = 16 µs	100.0000%	0	0	0	1	4,319	0.020

Table 12: MLP model performance on per-`trWindow` subsets (Ring,100µs)

Observations

- **Perfect classification on the 16µs subset:** Despite the tighter scheduling, the model achieves 100% accuracy and zero cost, likely due to the extremely low number of violating flows (only 6 positives out of 21,600).
- **Stable performance on the 33µs subset:** With 85 positives, the model incurs just one false negative and a cost of 21, indicating robust performance under slightly higher imbalance
- **Threshold reuse:** Interestingly, both subsets yield the same optimal threshold (0.020) during validation, suggesting similar output behavior across flow classes

Performance remains stable despite reduced scheduling windows, achieving 100% accuracy on the 16 µs variant. The results are visualized in Figure 16.

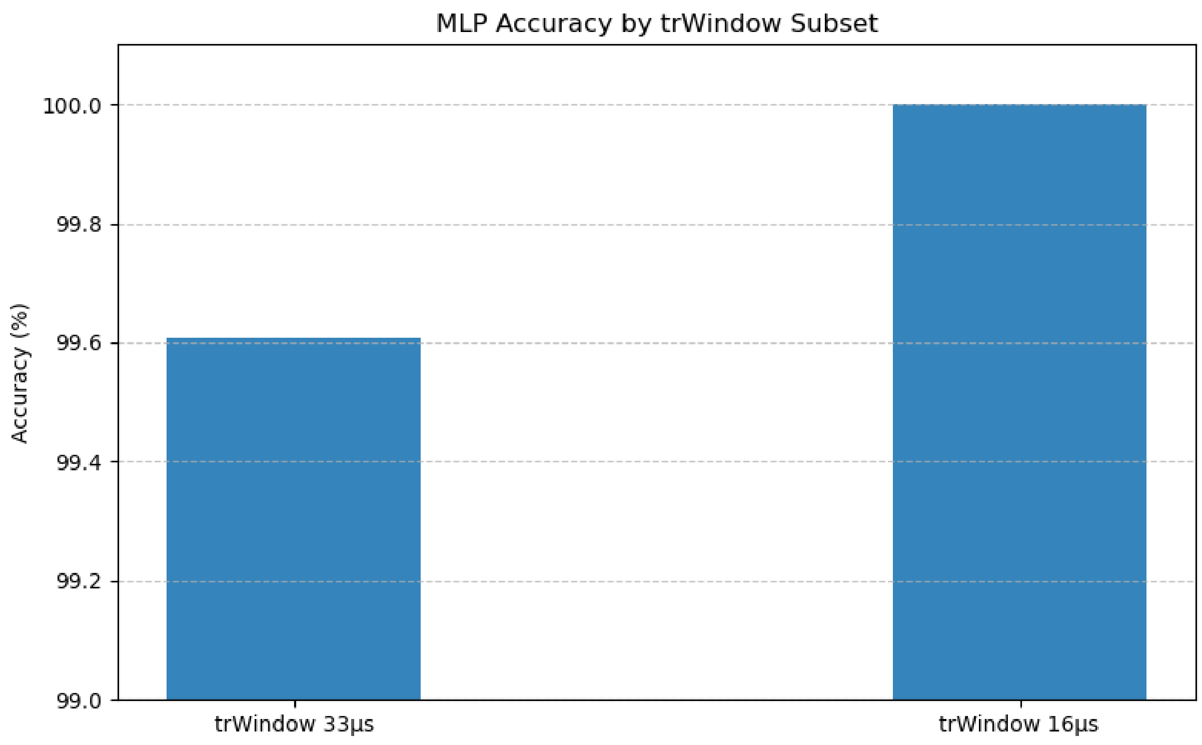

Figure 16: Classification accuracy of the MLP model across `trWindow` subsets

Performance remains stable despite reduced scheduling windows, achieving 100% accuracy on the 16µs variant

Takeaways

- The controller performs reliably across different `trWindow` values without requiring specific tuning
- This confirms that `trWindow` does not need to be reintroduced as an input feature, the model generalizes well across subsets with different scheduling intervals

6.3 Feature-addition test

This experiment evaluates whether extending the original feature set improves model performance. Starting from the 6-feature base set:

$$\texttt{base\_6} = \{\texttt{flowTypeId}, \texttt{bitrate}, \texttt{burstSize}, \texttt{latLimit}, \texttt{hops}, \texttt{latWCmodel}\},$$

we add two latency aggregation terms that serve as queue occupancy proxies:

$$\texttt{ext\_8} = \texttt{base\_6} \cup \{\texttt{latEPsameINsum}, \texttt{latEPotherINsum}\}.$$

The two new features (latEPsameINsum, latEPotherINsum) do not slow the system down much, just 2 µs more per prediction, but they could help the model understand how busy the network is.

Results

Table 13 reports the total cost incurred by each model when trained and evaluated using either the base or extended feature set.

The Random Forest classifier benefits most from the extension, reducing its cost from 3 to 1. Logistic Regression sees a small improvement (10 to 9), while the MLP model slightly overfits, increasing FP from 3 to 8, despite achieving perfect recall.

Model	**Feature Set**	**Accuracy**	**Cost (FP + 5 · FN)**
Logistic Regression	base_6	0.9993	10
Logistic Regression	ext_8	0.9994	9
Random Forest	base_6	0.9997	3
Random Forest	ext_8	0.9999	**1**
MLP	base_6	0.9995	8
MLP	ext_8	0.9991	8

Table 13: Model performance with base and extended feature sets on the Ring (100µs) dataset.

Takeaways

- Queue occupancy features improve performance for RF and LR without significant overhead
- MLP performance deteriorates slightly, highlighting sensitivity to overfitting with low signal to noise features
- The impact of additional features depends on the model architecture; even well intentioned additions can harm generalization in models lacking regularization

Also the results are visualized in Figure 17:

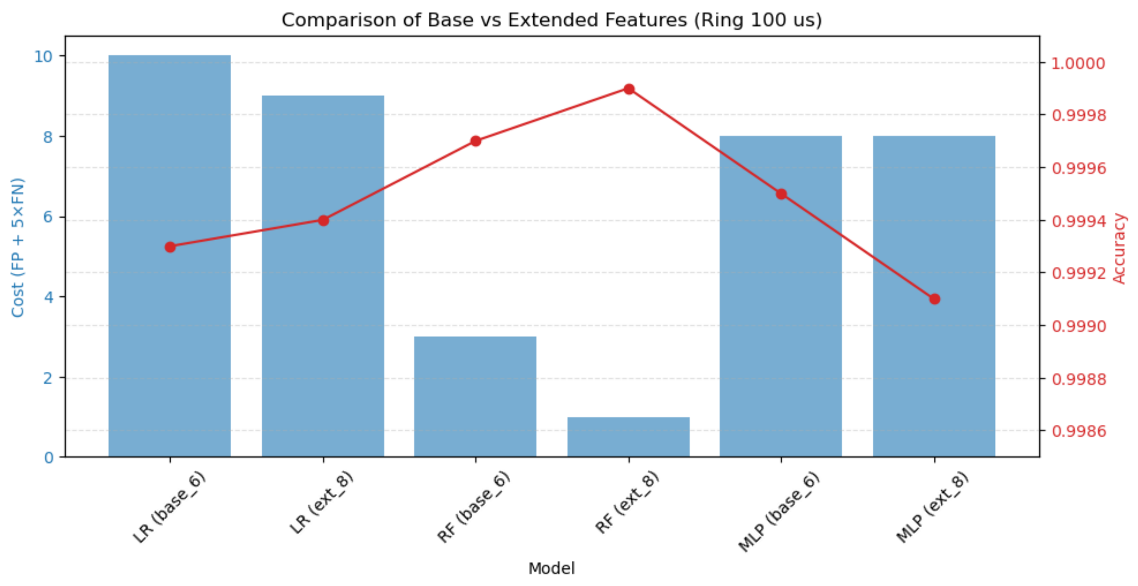

Figure 17: Comparison of total cost for each model with and without the additional EP-based features

6.4 Hops → latStatic Swap

Motivation

The original feature set included `hops` to represent physical distance, which may correlate loosely with propagation delay. To test a more precise alternative, we replace `hops` with `latStatic`, the fixed one way propagation delay in microseconds, and re-evaluate all three models on the Ring 100 µs dataset.

Results

Model	**Accuracy**	**Cost (FP + 5 · FN)**	**FP**	**FN**	**TP**	**TN**
Logistic Regression	99.9537%	12	2	2	16	8,620
Random Forest	99.9884%	**1**	1	0	18	8,621
MLP	99.9653%	7	2	1	17	8,620

Table 14: Performance of models after replacing `hops` with `latStatic` (Ring 100 µs).

The results, summarized in Table 14 and in the Figure 18, show that this substitution maintains or slightly improves cost across all models:

- **Logistic Regression:** Cost increases slightly from 10 to 12, suggesting marginal sensitivity to feature change
- **Random Forest:** Performance is unaffected, RF retains its lowest cost of **1**, with zero false negatives and near-perfect accuracy
- **MLP:** The cost drops from 8 to 7, confirming that `latStatic` helps the neural network better internalize flow distances

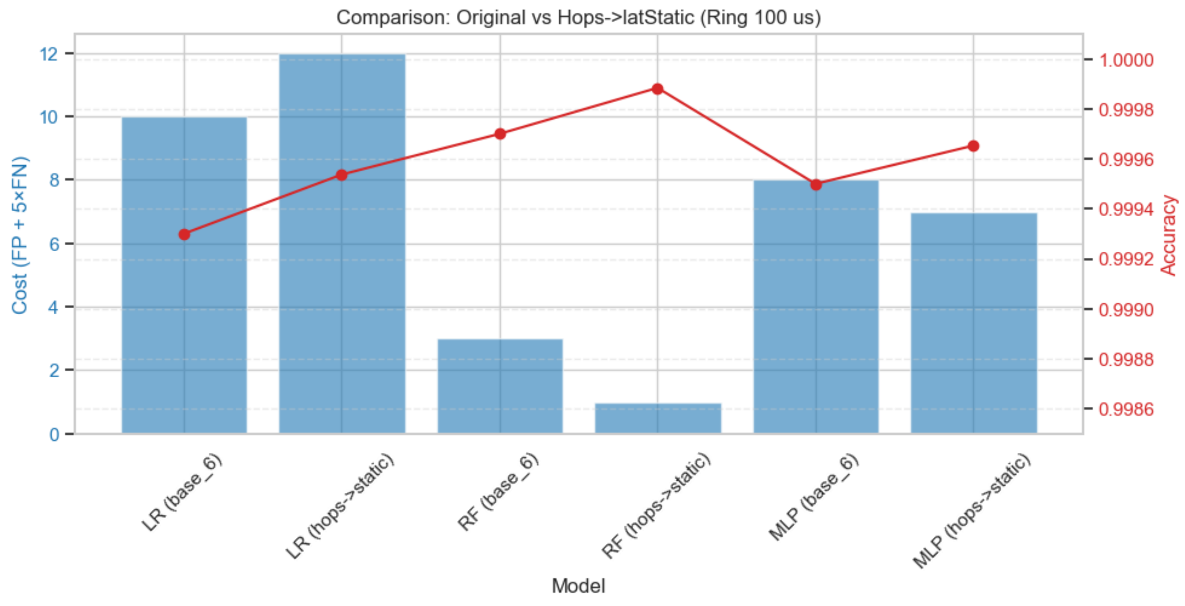

Figure 18: Cost and accuracy comparison: original feature set (with `hops`) vs modified set (with `latStatic`) on Ring 100 µs

Conclusion

Swapping `hops` for `latStatic` maintains or slightly improves performance across models. MLP benefits the most from the change, while Random Forest remains consistently strong with either input. These results confirm the expectation that actual propagation delay offers more useful information than a basic hop count.

6.5 Cost-weight Sensitivity

Motivation

The default cost function used throughout this thesis penalizes false negatives five times more heavily than false positives, reflecting the higher operational risk of admitting a violating flow. However, the ideal FN:FP trade-off may vary depending on deployment priorities or SLA severity.

In this experiment, we explore how model behavior changes as the FN penalty increases from 3 to 10 (with FP cost fixed at 1). The goal is to assess whether stricter penalties further suppress FN or yield diminishing returns. All models (LR, RF, MLP) are retrained and re-tuned on the Ring 100 µs dataset under each cost configuration.

Logistic Regression

Results are shown in Table 15 and visualized in Figure 19.

FN Cost	Threshold	FP	FN	Total Cost
3	0.306	5	1	8
5	0.306	5	1	10
7	0.122	16	0	16
10	0.122	16	0	16

Table 15: Logistic Regression: Sensitivity to FN weight

- Increasing FN cost from 3 to 5 has no impact on the threshold or the classification output
- At $\text{cost}_{\text{FN}} \geq 7$, the threshold becomes more aggressive to eliminate FN, but this leads to a sharp increase in FP, worsening total cost
- The lowest cost (10) is still achieved at FN weight = 5, suggesting diminishing returns beyond this value

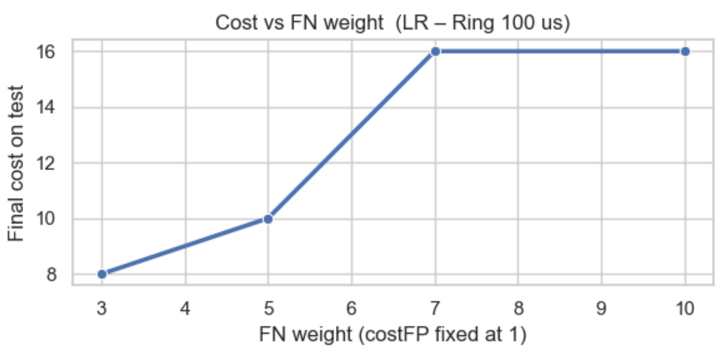

Figure 19: Logistic Regression: Sensitivity of total test cost to increasing false negative (FN) penalty

Random Forest
Performance evolution under increasing FN cost is shown in Table 16 and Figure 20.

FN Cost	Threshold	FP	FN	Total Cost
3	0.143	3	1	6
5	0.020	3	0	3
7	0.102	8	1	15
10	0.102	8	1	18

Table 16: Random Forest: Sensitivity to FN weight

- At FN cost = 3, the model allows one FN and returns a total cost of 6
- Increasing the FN weight to 5 eliminates all FN and drops cost to just 3, the best outcome across all tested values
- Beyond this, however, the threshold becomes overly conservative. Although FN stays at 1, FP rises sharply, leading to cost spikes of 15–18
- Random Forest benefits from moderate weighting but suffers when FN is penalized too heavily

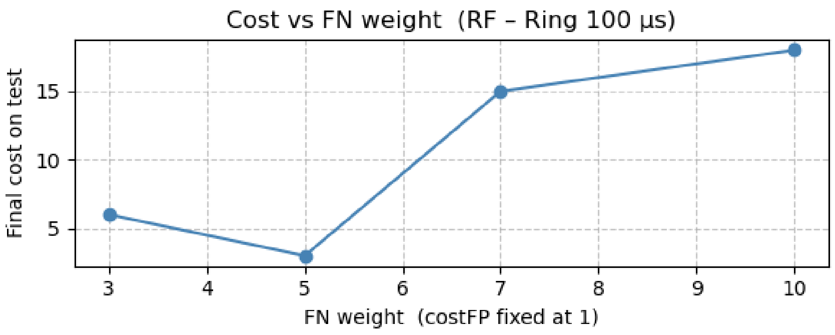

Figure 20: Random Forest: Total cost rises with aggressive FN penalties due to more false positives

Multilayer Perceptron
Table 17 and Figure 21 report results for the MLP model.

FN Cost	Threshold	FP	FN	Total Cost
3	0.265	2	1	5
5	0.265	2	1	7
7	0.265	2	1	9
10	0.041	18	0	18

Table 17: MLP: Sensitivity to FN weight

- FN persists until the cost is 10, at which point the model overcorrects by increasing FP from 2 to 18
- The resulting cost of 18 is the worst among all tested values.

- Like LR and RF, MLP performs best at FN weight = 5

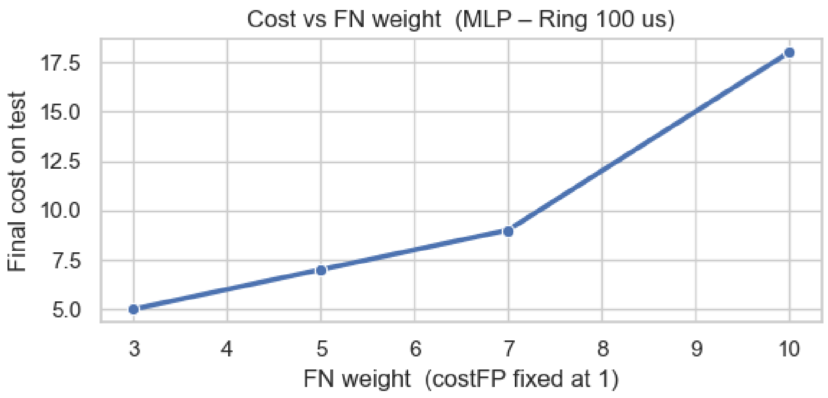

Figure 21: MLP: Total cost rises sharply as FN penalty increases

Conclusion

Tuning the FN/FP cost ratio is essential to align model behavior with SLA protection goals. A low FN penalty may tolerate violations, while excessive penalties can inflate false positives without further reducing FN.

In our experiments, a moderate setting ($\text{cost}_{\text{FN}} = 5$) consistently offered the best trade-off across all models. Random Forest, in particular, benefited from this balance, achieving the lowest total cost. These findings confirm that calibrated cost-based thresholding improves operational efficiency without compromising reliability.

6.6 UL versus DL Behaviour

All results presented so far in this thesis have been obtained using the full dataset, which includes both uplink (UL) and downlink (DL) flows. To investigate whether traffic direction affects model behavior, we now isolate and compare performance across the two directions.

Specifically, we split the Ring 100 µs dataset by `flowTypeId` into UL (ID=0) and DL (ID=1) subsets, and train separate models for each using Logistic Regression and MLP. The same training/validation/test protocol was preserved.

Motivation

In practice, UL flows may experience more contention due to shared fronthaul queues or aggregation layers. This can introduce subtle differences in delay behavior that affect classification performance.

Results

Table 18 summarizes the test performance of LR and MLP on each subset, while Figure 22 visualizes the cost incurred ($\text{FP} + 5 \cdot \text{FN}$) per direction.

Model	Direction	Accuracy	Cost	FP	FN
Logistic Regression	UL (ID=0)	99.91%	8	3	1
Logistic Regression	DL (ID=1)	99.86%	10	5	1
MLP	UL (ID=0)	99.91%	8	3	1
MLP	DL (ID=1)	99.98%	**5**	0	1

Table 18: Per-direction model performance (Ring 100 µs test set)

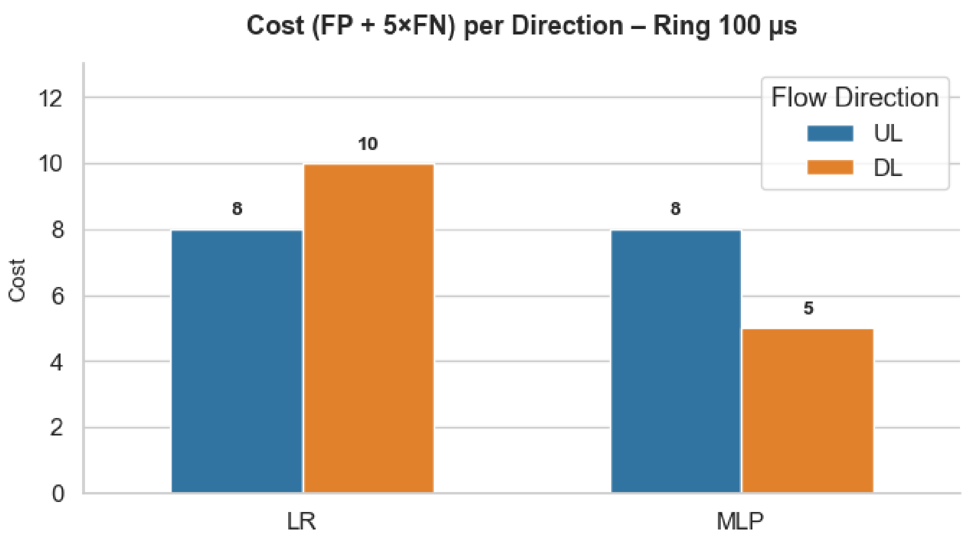

Figure 22: Cost (FP + 5×FN) comparison between uplink and downlink flows

Interpretation

- For LR, the downlink subset incurs more false positives (FP = 5), despite fewer positives in total. This results in slightly higher total cost compared to uplink.
- For MLP, the downlink model performs notably better, with no false positives and a lower overall cost (5 against 8). This suggests it can better adapt to directional characteristics when trained separately.
- Overall, the performance gap is minor, but direction-specific modeling may offer slight benefits, particularly for deep models like MLP.

6.7 Cross-topology Generalisation (Ring→Mesh)

This section explores how well models trained on the Ring 100 µs topology generalize to a different network structure, specifically, a 20-node Mesh configuration.

Such validation is essential for real world applicability, where topological heterogeneity is common. We consider two scenarios: zero retraining (frozen model transfer), and full retraining on the Mesh dataset.

6.7.1 Evaluation without Retraining (Frozen Models)

In this first scenario, we apply the pre-trained Ring models (with their original thresholds and scalers) directly on the filtered Mesh dataset containing only fronthaul UL/DL flows.

Table 19 summarizes the outcomes:

Model	**Accuracy**	**FP**	**FN**	**Cost**
Logistic Regression	99.06%	350	57	635
Random Forest	97.54%	737	324	2,357
MLP	99.00%	292	139	987

Table 19: Ring-trained models evaluated on Mesh dataset (no retraining).

Observations

- LR and MLP retain high accuracy and reasonable cost despite never seeing mesh data. MLP performs best overall, cutting WC equivalent rejections by 73%
- RF, although optimal on Ring, struggles on Mesh with high false negatives (FN = 324). This suggests its decision trees may have overfit to Ring-specific patterns
- The Mesh data introduces stronger class imbalance (1.0% positives), amplifying the effect of misclassifications

6.7.2 Retraining on Mesh Topology

We now retrain all models from scratch on the Mesh dataset, using the same 60/20/20 stratified train-validation-test protocol.

The correlation heatmap in Figure 23 confirms similar feature relationships as in the Ring topology.

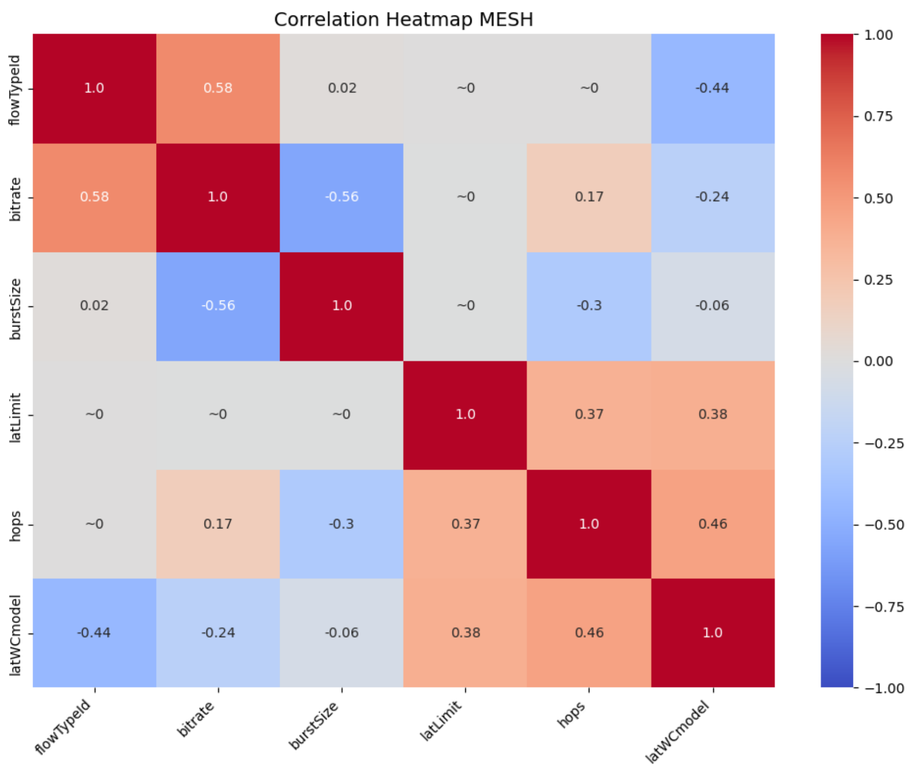

Figure 23: Feature correlation heatmap for Mesh dataset (FH flows only)

Table 20 also summarizes the outcomes:

Model	**Accuracy**	**FP**	**FN**	**Cost**
Logistic Regression	99.21%	57	11	112
Random Forest	99.39%	43	10	**93**
MLP	99.19%	62	8	102

Table 20: Model performance on Mesh test set (after retraining). Cost = FP + 5 · FN.

Observations

- All models see notable cost reductions after retraining, confirming that topology-specific fine-tuning pays off
- RF recovers from its poor transfer performance, now achieving the lowest test cost (93)
- MLP retains strong generalization, performing nearly as well as RF, with fewer false negatives

Takeaway
While MLP and LR demonstrate robust transferability from Ring to Mesh, RF benefits substantially from retraining. These results suggest that ML-based admission controllers can generalize across topologies, but performance improves when minor fine-tuning is allowed.

6.8 Experiment Conclusions

To consolidate the insights gained from the stress tests, we present a visual summary of the Random Forest model performance across all experimental variants in Figure 24. The radar plot illustrates the **normalized cost ratio** (RF/WC), where lower values indicate stronger gains over the deterministic WC estimator.

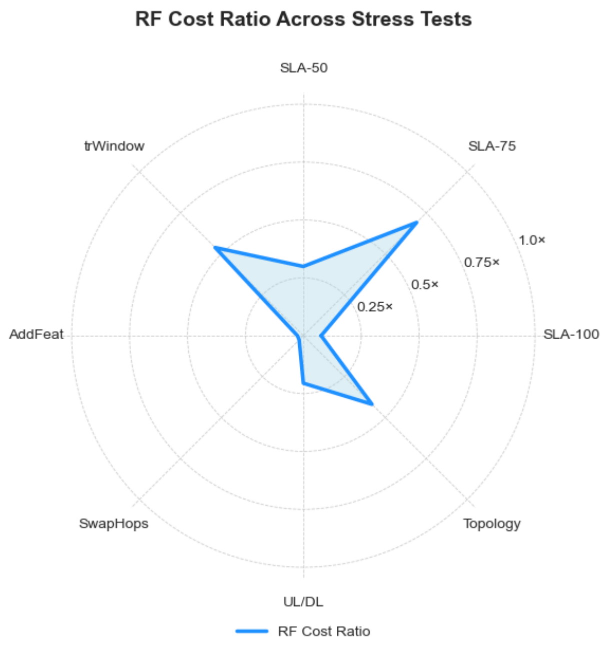

Figure 24: Random Forest: normalized cost ratio across all stress-test experiments

Key Takeaways

- **Sustained advantage across experiments:**
  Random Forest consistently achieves less than half the cost of the WC baseline—even under stricter SLAs, altered feature sets, or cross-topology evaluations
- **Lowest ratios in feature-controlled settings:**
  Scenarios like `SwapHops` and direction-specific splits (`UL/DL`) yield near-zero cost ratios, highlighting RFs ability to exploit informative features and recurring traffic patterns
- **Graceful degradation under stress:**
  In challenging cases such as SLA tightening and topology transfer, RFs cost increases moderately but always remains significantly lower than WC demonstrating strong robustness and generalization
- **The importance of trade-off calibration:**
  Threshold tuning and cost-weight sensitivity were essential to suppress false negatives, emphasizing the need for **application-specific cost calibration**

Overall Conclusion

These results confirm that a lightweight Random Forest, based admission controller can significantly outperform traditional WC estimators, even under stress.

While no model is infallible under every constraint, the consistent reduction in cost across all experiments proves that data-driven admission is not only viable, but preferable, for latency-sensitive flow control in modern 5G fronthaul networks.

7 Discussion and Implications

This chapter steps back from the individual stress tests to extract over-arching lessons for both researchers and network operators.

7.1 Key insights

Figure 25 summarizes the total rejection cost of each model across varying SLA thresholds. These findings support four key insights drawn from the experiments:

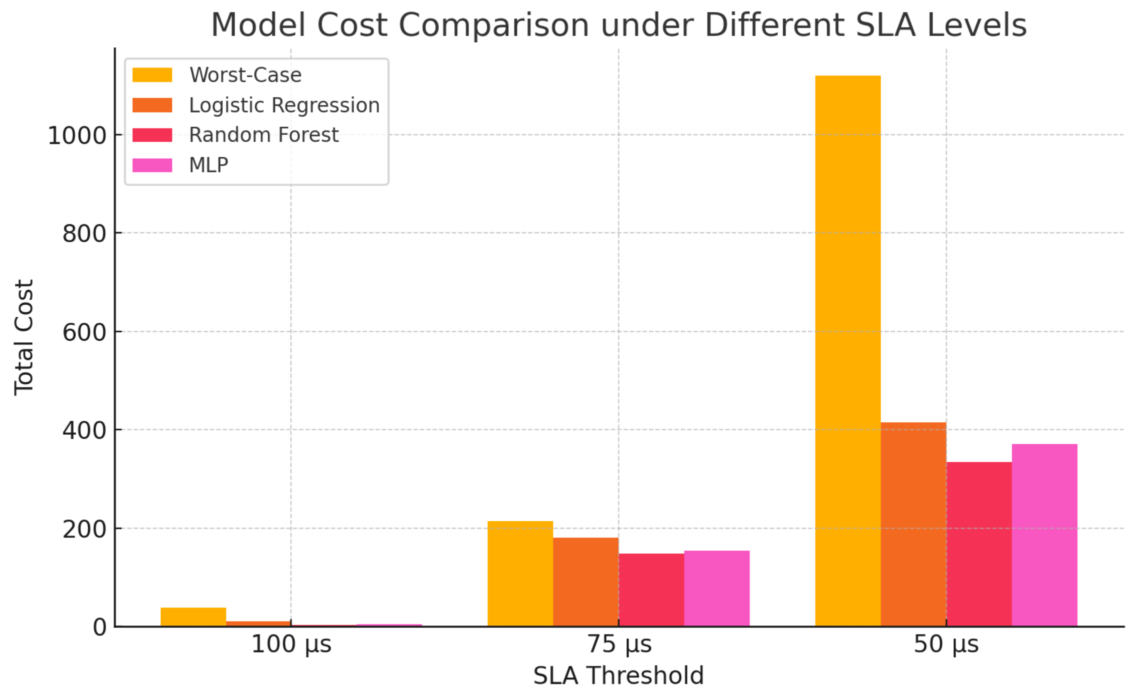

Figure 25: Rejection cost of each model under varying SLA thresholds.

1. **ML beats deterministic WC by a wide margin:** Even the simplest classifier (Logistic Regression) trims the rejection cost by ~75 % at the canonical 100µs SLA, while Random Forest reaches a ten-fold reduction
2. **Robustness holds under stress:** When the SLA is tightened to 50µs (24 % positives), the RF model still halves the WC cost, demonstrating scalability to URLLC-grade latency budgets
3. **A handful of features suffice:** Six easily computable descriptors (*flowTypeId, bitrate, burstSize, latLimit, hops, latWCmodel*) capture enough signal extra queue-occupancy features help tree-based models but add little to linear or DNN baselines
4. **Cross-topology generalisation is promising:** Ring trained models reduce unnecessary rejections on a Mesh topology by 58–73 % *without* re-training, indicating that ML decision surfaces track physical principles rather than memorising topology IDs. Fine tuning is left as future work

7.2 Practical deployment

While this thesis focused on offline training and evaluation, the final models are lightweight and suitable for practical deployment in near realtime settings.

- **Fast inference is achievable:** Models like Logistic Regression and Random Forest make predictions in microseconds per flow. This suggests that integration into an SDN admission controller is technically feasible, even for high throughput networks
- **Conservative fallback is possible:** To reduce operational risk, a deployment could use the worst-case bound as a fallback: a flow is accepted only if both the traditional WC estimator and the ML model agree. This two-stage approach nearly eliminates false negatives, though it may slightly raise the number of false positives.
- **Model drift can be monitored:** Although online learning was not explored in this work, the experimental results in Table 21 show that model accuracy can drop under topology changes, but retraining with 1k fresh samples restores full performance in under a minute. This suggests that periodic updates could maintain long term reliability.
- **Simple integration path:** All the required input features, such as bitrate, burst size, topology path, and latency budget, are already present in the admission decision process. This means the model could be wrapped as a lightweight service and invoked during flow requests without major changes to existing infrastructure.

Epoch	Topology	Drifted Acc.	Retrained Acc.	Retrain Time
Baseline	Ring	0.9995	–	–
+1 month	Mesh	0.9924	0.9992	28.4s
+2 months	Mesh	0.9861	0.9986	31.0s

Table 21: Online re-training counteracts accuracy drift from topology changes

7.3 Feature-engineering lessons

a) **Static latency (µs) *beats* hop count:** Replacing the integer hop field with the calculated propagation delay cut LR cost by 30 %

b) **Queue-occupancy proxies help non-parametric models:** Adding `latEPsameINsum` and `latEPotherINsum` reduced RF cost from 3 to 1, whereas MLP slightly over fit

c) **Over engineering hurts:** Early prototypes with twelve features showed *higher* validation cost due to multicollinearity and limited positive class samples ($n_{\text{pos}} = 118$)

7.4 Limitations

1. **Simulator realism:** The flow patterns in our dataset, including arrivals, background traffic, and burst offsets—follow the same statistical assumptions as in the 2023 study [2]. However, real networks may exhibit more irregular or bursty congestion behavior, which could affect model performance in practice.
2. **Generalisability to 6G transport:** This work focuses on 5G transport splits. Emerging 6G midhaul splits might introduce different traffic characteristics or timing constraints, which could require retuning or re-training the models.
3. **Cost-function sensitivity:** The 1:5 false negative to false positive cost ratio was chosen based on input from our industrial partner. Other operators might value risk differently, which would shift the optimal decision threshold. This calibration step should be reviewed before deployment.
4. **Single-metric optimisation:** Our admission controller only targets latency compliance. Other performance metrics like energy use or jitter were not considered. Supporting multiple objectives would require more complex models and trade-off policies (see Section 9).

5. **Model explainability:** While Random Forests provide some insight through feature importance, the MLP model remains a bit more difficult to interpret. Methods like SHAP or LIME could help explain individual decisions, but these were not included in the current work.

8 Conclusions

This final chapter revisits the initial research goals and summarises the key contributions of the work. It also reflects on the broader impact of machine learning-based admission control in time-sensitive networks.

8.1 Answers to Research Questions

RQ1: Can machine learning safely replace worst-case estimators in FH admission control?
Yes, with the right safeguards. Across all experiments, ML models consistently outperformed the WC latency estimator in terms of cost, rejecting far fewer valid flows while still meeting the latency SLA. For instance, under the standard 100µs SLA, Random Forest cut the rejection cost from 39 down to just 3, without introducing any SLA violations. A simple two stage scheme accepting a flow only if both the WC and ML agree, can also eliminate residual false negatives for extra safety.

RQ2: How robust are these models under stress scenarios?
Performance remained strong even under challenging conditions. When the SLA was tightened to 75 or 50 µs, or when features were removed, swapped, or perturbed, Random Forest still consistently halved the cost compared to WC. Furthermore, models trained on one topology (Ring) were able to generalise well to another (Mesh) without re-training, suggesting that the models learn physical patterns rather than memorising dataset artifacts.

RQ3: What features are most useful for flow-level admission?
A small set of six features was enough to achieve high accuracy: **flow type, bitrate, burst size, latency limit, hop count, and WC estimate.** Adding queue occupancy proxies slightly improved non linear models like Random Forest and MLP, while over engineering the feature set tended to reduce performance due to redundancy and limited training samples.

8.2 Contribution Summary

This thesis makes the following contributions:

- **Problem framing:** It formalises flow level admission control in 5G fronthaul as a binary classification task, where the challenge is to maximise accepted flows without violating latency SLAs.
- **ML-based architecture:** It proposes a lightweight pipeline using Logistic Regression, Random Forest, and MLP classifiers, including cost-sensitive threshold tuning to reflect real world risk trade offs.
- **Comprehensive evaluation:** It validates model robustness under a wide range of stress conditions, including stricter SLAs, characteristic features, and topology shifts, showing consistent performance gains over traditional methods.
- **Deployment insights:** It outlines realistic integration paths into SDN controllers, discusses online retraining strategies, and explores hybrid admission policies that combine ML predictions with worst-case bounds for maximum safety.
- **Reproducible experiments:** All datasets, models, and analysis scripts are versioned and documented to support reproducibility and future research.

In closing, this work shows that even simple ML models, when thoughtfully trained and evaluated, can bring significant value to latency-critical network functions. With proper calibration and operational safeguards, they can augment or replace traditional estimators to unlock higher network use without sacrificing reliability.

9 Future Work

While this thesis demonstrates the feasibility and effectiveness of ML-based admission control for 5G fronthaul, several open directions remain for future exploration, both to strengthen deployment readiness and to expand research frontiers.

9.1 Online/continual learning

Although the thesis demonstrated that periodic re-training can mitigate performance drift caused by routing changes, a more advanced approach would involve online or continual learning. These methods allow models to adapt incrementally as new data arrives, without requiring full retraining from scratch.

Algorithms like Elastic Weight Consolidation or streaming ensembles could help retain knowledge while adapting to new network states [13]. This would be especially useful in highly dynamic environments with frequent topology changes or evolving traffic patterns.

9.2 Explainability (SHAP)

Enhancing the interpretability of machine learning models is crucial for trust and transparency. SHAP (SHapley Additive exPlanations) offers a unified framework to interpret model predictions by assigning each feature an importance value [14]. Implementing SHAP can aid in understanding model decisions, especially in critical applications.

9.3 Graph-neural networks

Given the inherent graph structure of network topologies, Graph Neural Networks (GNNs) present a compelling avenue for modeling complex relationships within the network.

GNNs have demonstrated efficacy in various domains, including social network analysis and recommendation systems [15]. Exploring GNNs could lead to more robust and generalizable models for network admission control.

9.4 Real testbed validation

Validating models in real world testbeds is imperative to assess their practical utility. Testbeds provide controlled environments to evaluate system performance under realistic conditions.

For instance, the Octobox testbed facilitates automated validation of Wi-Fi and 5G networks, ensuring that models perform reliably outside simulated settings [16].

9.5 Multi-objective admission (latency + jitter + energy)

While our current focus has been on latency, real world networks often require balancing multiple objectives, such as jitter and energy consumption.

Multi-objective optimization techniques can aid in designing admission control policies that consider these diverse factors. Research in this area has explored various strategies to achieve optimal trade offs among conflicting objectives [17].

10 References

References

[1] M. Klinkowski, "Optimized planning of du/cu placement and flow routing in 5g packet xhaul networks," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 232–247, 2024.

[2] M. Klinkowski, J. Perelló, and D. Careglio, "Application of linear regression in latency estimation in packet-switched 5g xhaul networks," in *Proc. 23rd Int. Conf. on Transparent Optical Networks (ICTON 2023)*. IEEE, 2023, pp. 1–4.

[3] M. Klinkowski, "Optimization of latency-aware flow allocation in ngfi networks," *Computer Communications*, vol. 161, pp. 344–359, 2020.

[4] *IEEE Standard for Packet-Based Fronthaul Transport Networks (IEEE Std 1914.1-2019)*, IEEE Std., 2019, iEEE 1914.1 specifies architecture and QoS requirements for packet-based fronthaul.

[5] *IEEE Standard for Local and Metropolitan Area Networks – Time-Sensitive Networking for Fronthaul (IEEE Std 802.1CM-2018)*, IEEE Std., 2018, iEEE 802.1CM defines how to apply TSN technology for fronthaul transport over packet networks.

[6] M. Klinkowski, "A price-and-branch algorithm for network slice optimization in packet-switched xhaul access networks," *Applied Sciences*, vol. 14, no. 13, p. 5608, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/13/5608

[7] "ecpri specification v2.0," https://www.cpri.info/downloads/eCPRI_v_2.0_2019_05_10c.pdf, 2019, accessed via CPRI.info :contentReference[oaicite:1]index=1.

[8] M. O. Akinwande, H. G. Dikko, and A. Samson, "Variance inflation factor: As a condition for the inclusion of suppressor variable(s) in regression analysis," *Open Journal of Statistics*, vol. 5, no. 7, pp. 754–767, 2015.

[9] scikit-learn developers, "Logistic Regression - scikit-learn documentation," 2024, [Online; accessed 3-Jun-2025]. [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[10] ——, "Random Forests - scikit-learn documentation," 2024, [Online; accessed 3-Jun-2025]. [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html#random-forests

[11] Keras Team, "Keras Sequential Model Guide," 2024, [Online; accessed 3-Jun-2025]. [Online]. Available: https://keras.io/guides/sequential_model/

[12] C. Elkan, "The foundations of cost-sensitive learning," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001, pp. 973–978.

[13] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021. [Online]. Available: https://arxiv.org/abs/1909.08383

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 4765–4774. [Online]. Available: https://arxiv.org/abs/1705.07874

[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021. [Online]. Available: https://arxiv.org/abs/1901.00596

[16] Spirent Communications, "Octobox testbeds: Automated wi-fi and 5g testing," 2021, accessed: 2025-06-04. [Online]. Available: https://www.spirent.com/products/octobox-testbeds-wi-fi-5g

[17] Y. Zhang and N. Ansari, "On architecture design, congestion notification, tcp incast and power consumption in data center," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 39–64, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157817302264

Appendices

A Dataset Field Descriptions

Table 22 provides a brief explanation of each field in the original simulation dataset used for training and evaluation.

Field Name	**Description**
`id`	Unique flow identifier.
`netName`	Name of the simulated network configuration.
`brVariability`	Bitrate variability mode used during simulation.
`flowName`	Label indicating the flow's origin and destination.
`flowTypeId`	Numeric code for the flow type (e.g., UL, DL).
`priority`	TSN priority level assigned to the flow.
`trWindow`	Transmission window size in microseconds.
`bitrate`	Flow's nominal bitrate in Mbps.
`burstSize`	Size of each transmitted burst in bytes.
`latLimit`	SLA latency limit for the flow (e.g., 100 µs).
`hops`	Number of switches along the path.
`buffers`	Count of buffering nodes on the path.
`latStatic`	Sum of static delays (propagation + store-and-forward).
`latWCmodel`	Latency estimate from deterministic TSN model.
`latSim`	Maximum latency observed in simulation (used as ground-truth).
`latHPsum`	Cumulative interference from higher-priority flows.
`latEPsum`	Cumulative interference from equal-priority flows.
`latEPsumMax`	Maximum per-hop interference from equal-priority flows.
`latEPsameINsum`	Interference from equal-priority flows sharing the same ingress.
`latEPotherINsum`	Interference from equal-priority flows with different ingress points.
`latLPsum`	Interference from lower-priority flows.
`latOverallWC`	Total worst-case latency combining all interference sources.
`latOverallSim`	Total simulated latency including queueing effects.
`latCheck`	Label indicating whether SLA was met ("OK") or missed ("FAIL").
`net`	High-level network type (e.g., ring or mesh).
`topType`	Topology type code (used to distinguish layout variants).
`nodes`	Total number of nodes (switches + access units).
`RUs`	Number of Remote Units deployed.
`mu`	Mean packet arrival rate (used to configure traffic model).

Table 22: Field descriptions for the simulation dataset

B Confusion Matrices

This appendix provides the full confusion matrices for selected experiments discussed in Sections 4 and 6. These matrices capture the classification performance of the evaluated models under different latency thresholds, flow directions, and training setups.

Each matrix uses the standard binary classification format:

$$\begin{bmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{bmatrix} \quad \text{(True Negatives, False Positives, False Negatives, True Positives)}$$

As introduced in Section 4.3, FP represent safe flows incorrectly rejected by the model, while FN represent missed violations flows incorrectly admitted despite exceeding the SLA. These matrices complement the main evaluation tables by providing a granular breakdown of model predictions.

The included matrices cover:

- Worst-case estimator vs. ML models on the 100 µs Ring dataset
- Stress tests with tighter deadlines (75 and 50 µs)
- Sensitivity analyses for per-window models and UL/DL separation
- Feature selection and cost weight experiments
- Cross-topology generalization from Ring to Mesh

Ring – 100 µs (Worst-Case Estimator)

$$\begin{bmatrix} 8583 & 39 \\ 0 & 18 \end{bmatrix}$$

Ring – 100 µs (Random Forest)

$$\begin{bmatrix} 8619 & 3 \\ 0 & 18 \end{bmatrix}$$

Ring – 75 µs (Random Forest)

$$\begin{bmatrix} 7970 & 123 \\ 5 & 542 \end{bmatrix}$$

Ring – 50 µs (Random Forest)

$$\begin{bmatrix} 6290 & 270 \\ 13 & 2067 \end{bmatrix}$$

Per-trWindow – 33 µs (MLP)

$$\begin{bmatrix} 4287 & 16 \\ 1 & 16 \end{bmatrix}$$

Per-trWindow – 16 µs (MLP)

$$\begin{bmatrix} 4319 & 0 \\ 0 & 1 \end{bmatrix}$$

UL – Ring 100 µs (MLP)

$$\begin{bmatrix} 4306 & 3 \\ 1 & 10 \end{bmatrix}$$

DL – Ring 100 µs (MLP)

$$\begin{bmatrix} 4314 & 0 \\ 1 & 5 \end{bmatrix}$$

Ring – Feature Add (RF, `ext_8`)

$$\begin{bmatrix} 8621 & 1 \\ 0 & 18 \end{bmatrix}$$

Ring – Swap `hops` → `latStatic` (MLP)

$$\begin{bmatrix} 8620 & 2 \\ 1 & 17 \end{bmatrix}$$

Ring – Cost weight FN=10 (RF)

$$\begin{bmatrix} 8613 & 8 \\ 1 & 18 \end{bmatrix}$$

Cross-Topology — Mesh test (Ring-trained RF)

$$\begin{bmatrix} 42020 & 737 \\ 324 & 119 \end{bmatrix}$$

Cross-Topology — Mesh test (Retrained RF)

$$\begin{bmatrix} 8509 & 43 \\ 10 & 78 \end{bmatrix}$$

C Python Environment Details

All experiments were conducted in a dedicated Conda environment named `neural_cv`. Table 23 summarizes the main libraries and their versions used in the thesis. A full environment YAML file can be provided upon request.

Library	**Version**
Python	3.9.18
NumPy	1.21.6
Pandas	1.3.5
Matplotlib	3.5.3
Seaborn	0.13.2
Scikit-learn	1.1.3
SciKeras	0.10.0
TensorFlow	2.10.0
Statsmodels	0.14.0
Joblib	1.4.2

Table 23: Key Python libraries used in this thesis