

Uncertainty-Aware Conformalized Quantile Regression for ML-based Latency Prediction

For Interface Selection in Unlicensed Spectrum

Anders Peter Bundgaard Kristensen,
Nicolai Dalsgaard Lyholm,

Electronics Systems, Aalborg University, 2025





Electronic Systems
Aalborg University
<http://www.aau.dk>

AALBORG UNIVERSITY

MASTER'S THESIS

Title:

Uncertainty-Aware Conformalized Quantile Regression for ML-based Latency Prediction

Theme:

Communication Systems & Machine Learning

Project Period:

1. Sep, 2024 - 4. Jun 2025

Project Group:

ES920

Participants:

Anders Peter Bundgaard Kristensen

✉ bundgard.dybvad@gmail.com

in [Anders Bundgaard](#)

Nicolai Dalsgaard Lyholm

✉ nlyholm@hotmail.com

in [Nicolai Lyholm](#)

Supervisors:

Petar Popovski

Page Number: 92

Date of Completion:

June 4, 2025

Abstract:

Ensuring reliable, low-latency communication in the congested and unlicensed 2.4 GHz ISM band using interfaces based on the IEEE 802.11 and 802.15 standard is a significant problem. This thesis addresses the problem by developing a method to predict the 95th latency quantile (Q95) with a statistically sound measure of uncertainty. We propose an "Uncertainty-Aware Conformalized Quantile Regression" approach. A comprehensive simulation framework was created to model WiFi and Bluetooth Low Energy (BLE) link-level behaviors, generating data to train five distinct latency predictor models: two parametric (MV-Param, GMM-Param) and three quantile regression (MV-QR, GMM-QR MSE, GMM-QR Pinball). These models were then calibrated using split conformal prediction to achieve a 90% confidence level for the predicted Q95 interval. Key findings show that quantile regression models, particularly the GMM-QR Pinball, offer superior predictive accuracy. This model exceeded the target coverage with a 90.37% coverage for both WiFi and BLE, whilst yielding tighter median uncertainty intervals (24.27 ms for WiFi, 11.59 ms for BLE) than parametric alternatives. This work demonstrates a validated methodology for uncertainty-aware latency prediction, enabling more intelligent and reliable interface selection in unpredictable wireless environments.

Preface

This thesis was written by Anders Peter Bundgaard Kristensen and Nicolai Dalsgaard Lyholm from the Department of Electronic Systems at Aalborg University, June, 2025.

We would like to acknowledge Professor Petar Popovski for his inspiring and visionary supervision throughout this project. He consistently encouraged us to explore ambitious ideas and strive for innovative solutions. We are also grateful for his role as Head of the Connectivity Department, which allowed him to incorporate us into a world-leading research team.

The close collaboration with the Connectivity Department at the Institute of Electronic Systems has resulted in many valuable scientific discussions and participation in multiple workshops, all of which have broadened our horizons regarding cutting-edge communication research. This inclusion has also fostered valuable personal connections, providing encouragement and helping us maintain motivation throughout the thesis process.

Aalborg University, June 4, 2025


Contents

Preface	ii
Contents	iii
1 Introduction	1
2 Basic of Wireless Latency, MAC Protocols and Interface Diversity	3
2.1 Deconstructing Communication Latency	3
2.1.1 Propagation Delay	3
2.1.2 Transmission Delay	4
2.1.3 Processing Delay	4
2.1.4 Queuing Delay	4
2.1.5 Medium Access Delay	4
2.2 Wireless Medium Access Protocol	4
2.2.1 WiFi (IEEE 802.11) and CSMA/CA	5
2.2.2 Bluetooth Low Energy (BLE) and FHSS	7
2.2.3 Duty cycle	9
2.2.4 Effect on latency	10
2.3 Interface Diversity Module	10
2.3.1 Cloning	10
2.3.2 Splitting	10
3 System Description	12
3.1 Overview of the Video Transmission System	12
3.1.1 Video Frame Data Flow	12
3.1.2 State Data Flow	13
3.2 Actor Module	13
3.3 Delimitation: Uncertainty-Aware Latency Predictor	15
3.3.1 Available RF Environment State Information for DNN Training	16
4 Data Generation and Simulation Framework	18
4.1 Modularization of Data Generation	18
4.2 Environment Generator	19
4.3 WiFi Simulator	21
4.3.1 Propagation Model	21
4.3.2 Interference Model	24
4.3.3 SINR & Throughput Modeling	26
4.3.4 Visualization of Simulator Outputs	28
4.4 BLE Simulator	29

4.4.1	Number of Retransmission due to Interference ($N_{\text{int}}(p_{\text{int}})$)	30
4.4.2	Number of Retransmissions due to Decoding Errors ($N_{\text{dec}}(p_{\text{dec}})$)	30
4.4.3	Runtime Optimization	32
4.4.4	Visualizing the Simulation Output	33
4.5	Generated Datasets	34
5	Latency Predictor Models	36
5.1	Model Families Overview	36
5.1.1	Parametric Models	36
5.1.2	Quantile Regression Models	37
5.1.3	Inputs/Outputs	38
5.2	Preprocessing	39
5.2.1	Batching Strategy	39
5.2.2	Batching Implementation	39
5.2.3	Prediction Frequency	41
5.3	Parametric Models: Training & Evaluation	42
5.3.1	Loss Function: Kullback Leibler Divergence	42
5.3.2	Hyperparameter Selection and Training Setup	43
5.3.3	Model Evaluation	45
5.4	QR Models: Training and Evaluation	46
5.4.1	Loss Functions: MSE and Pinball Loss	46
5.4.2	Hyperparameter Selection and Training Setup	47
5.4.3	Model Evaluation	48
5.5	Model Comparison	50
6	Conformal Prediction for Uncertainty Calibration	52
6.1	Theoretical Background	53
6.1.1	Symmetric and Asymmetric Uncertainty Intervals	53
6.2	Implementation Details	55
6.2.1	Calibration Procedure for GMM-QR	55
6.2.2	Coverage Validation	57
6.2.3	Strategy for Asymmetric Calibration	58
6.3	Interval Width Comparison	59
7	Discussion and Future Work	61
7.1	Insights from Model Prediction and Calibration	61
7.2	Alternative Framework for Parametric Models	61
7.3	Data Generation	62
7.3.1	Environment Specific Data Generation using Sionna	63
7.3.2	Continual Learning Framework	64
7.4	Dynamic Symmetric and Asymmetric Conformal Calibration Strategy	64
7.5	Actor Implementation	65
7.5.1	Naive Actor	65
7.5.2	Reinforcement Learning-Based Actor	66
8	Conclusion	68
	Bibliography	70

Appendix	73
A Modeling of Utility Metrics	74
B Available Data in Real World Systems	82
C Latency Predictor Results	88
D Conformal Prediction Coverage Analysis	91

List of Figures

1.1	(a) The AAU Space Robotics drone tethered via Ethernet at the European Rover Challenge due to severe Wi-Fi interference. (b) A Ukrainian FPV drone adapted with a fiber-optic cable to counter electronic warfare in a contested environment, photo by [1] 	1
2.1	Example of the backoff procedure in an environment with four stations.	5
2.2	Example of exponential evolution of contention window CW. Inspired by [10]. . .	6
2.3	(a) Hidden Terminal problem. (b) Exposed Terminal problem. Dashed lines indicate interference from a terminal in another network.	7
2.4	Message Sequence Diagram of the ARQ. Scenario 1: No collisions, scenario 2: Two collisions, Dashed lines are acknowledgments [12].	7
2.5	Flowchart of CSA 1 [11].	8
2.6	Simplification of the flowchart of CSA 2 [11].	9
2.7	Pseudo-random number generator	9
3.1	A block diagram of the envisioned video transmission system	12
3.2	In-/output model of the latency predictor.	15
3.3	In-/output model of the latency predictor with a confidence interval around the 95th quantile.	15
3.4	Preamble of 802.11n PHY frame [15].	16
3.5	Two examples of message sequence diagrams, a) showcasing WiFi as the active interface with frequent sampling and periodic sampling on the inactive BLE interface, and b) showcasing the opposite scenario.	17
4.1	Data generation subsystem overview	18
4.2	Generated simulation environment with random placement of devices.	20
4.3	Plot of LOS/NLOS path loss over distance, with highlighted break-point distance, marking the switch in LOS path loss models.	23
4.4	Histogram showcasing the pdf of LOS and NLOS shadowing distribution.	24
4.5	Example of exponential evolution of contention window CW. Inspired by [10]. . .	26
4.6	Histogram of SINR (left) and latency (right) for WiFi transmissions in a single generated simulation environment.	28
4.7	BER and PER for different SNR and BLE transmission modes.	31
4.8	Empirical validation of simplification.	33
4.9	Heat map over observations of required amount of transmissions	33
4.10	BLE latency distribution in an example simulation environment.	34
5.1	Visualization of parametric model, predicting a full probabilistic distribution. . .	37
5.2	Visualization of quantile regression model, directly predicting the $(1 - \alpha)$ -th quantile for the interface i	38
5.3	Flowchart of the batching procedure.	40

5.4	Average KL divergences between fitted model on batch data compared to fitted model on field data at different latency thresholds and distributions	41
5.5	Visualization of the KL divergence between two Gaussian distributions $p(x)$ and $q(x)$ (left plot). Plots the pointwise divergence with the total divergence being the shaded area (right plot).	42
5.6	Parallel coordinates plot of hyperparameter sweep for the MV-Param model. Visualizes parameter combinations and the resulting KL divergence loss.	43
5.7	Visualizes the parameter correlation with respect to validation KL divergence loss for the MV-Param model.	44
5.8	Visualization of randomly selected prediction from the MV-Param model. Showcases the empirical, target and predicted distribution with highlighted 95th quantiles.	45
5.9	95th quantile prediction error distribution for the MV-Param model.	46
5.10	Visualization of WiFi latency distribution with pinball loss function overlay and highlighted minimum loss for a randomly selected field.	47
5.11	Deviation between pinball loss minimum ($\tau = 0.95$) and empirical 95th quantile for WiFi (left) and BLE (right).	47
5.12	Visualization of randomly selected prediction from the GMM-QR Pinball model. Showcases the empirical latency distribution with the predicted and empirical 95th quantile highlighted for WiFi and BLE.	49
5.13	95th quantile error distribution for the GMM-QR Pinball model	49
5.14	Cumulative density function of absolute prediction error of the 95th latency quantile for all trained models, includes a naive baseline estimator as reference. Note, GMM-QR MSE has been dubbed GMM-MSE.	51
6.1	Illustration of conformal prediction, converting point prediction to calibrated uncertainty interval.	52
6.2	Relationship between minimum calibration set size n_{\min} and miscoverage level α	56
6.3	WiFi calibration score distribution.	56
6.4	BLE calibration score distributions.	57
6.5	Empirical and theoretical coverage as a function of calibration time.	58
6.6	Per field coverage distribution for WiFi.	58
6.7	Cumulative density function of uncertainty interval width for all trained models, with a calibration set collected over 500s. Note that GMM-QR MSE has been dubbed GMM-MSE.	60
7.1	Illustration of the proposed framework with a distribution classifier orchestrating a bank of trained parametric models each specialized in its own distribution.	62
7.2	(a) Imported 3D environment in Sionna with highlighted ray-tracing. (b) Corresponding power heatmap with highlighted transmitter position, takes into account device placement constraints based on the environment. The example environment is taken from AAU campus.	63
7.3	Visualizes two scenarios of interface selection. For equal upper bounds, the selection is obvious. However it becomes not straightforward, when upper bounds and interval sizes differ.	66


List of Tables

4.1	Simulation parameters, input for the data generation system.	19
4.2	Data structure for output of data generation system.	19
4.3	Commonly used WiFi channels and their center frequency.	22
4.4	Shadow fading standard deviations for outdoor path losses, as per <i>IEEE 802.11-2014</i> [17].	23
4.5	Mapping of SINR to Modulation and Coding Scheme (MCS) and corresponding PHY throughput [10].	27
4.6	Overview of BLE packet transmission time across PHY modes [11].	29
4.7	Simulation parameters, of the ML dataset and the CP dataset.	35
5.1	Model input and output specifications. Subscript $i \in \{\text{WiFi}, \text{BLE}\}$ denotes the interface. Bold symbols indicate vector-valued GMM parameters.	38
5.2	Supplementary metadata describing the RF environment.	38
5.3	Hyperparameters of the best-performing MV-Param model.	44
5.4	Hyperparameters of the best-performing GMM-Param model.	44
5.5	Hyperparameters of the best-performing MV-QR model.	48
5.6	Hyperparameters of the best-performing GMM-QR MSE model.	48
5.7	Hyperparameters of the best-performing GMM-QR Pinball model.	48
6.1	Uncertainty interval coverage and width for all models, grouped by interface. Width statistics are reported as both mean and median, computed across all fields.	60

1 | Introduction

In modern robotics and telecommunications, the reliability of wireless links is paramount. A failure in communication can lead to mission failure, or worse, the loss of valuable assets. This challenge was starkly illustrated during the 2024 European Rover Challenge (ERC), where the AAU Space Robotics team encountered severe communication issues. Operating in the crowded 2.4 GHz unlicensed spectrum, the team's rover and drone were subjected to a wireless environment saturated by the transmissions of numerous other teams. The result was not a complete connection loss, but a highly unpredictable and variable latency, with spikes that effectively mimicked a total drop in connection. These disruptions to critical control and video feeds ultimately crippled the aerial mission, forcing the team to tether their drone via an Ethernet cable, as depicted in Figure 1.1a, fundamentally compromising its primary advantage of mobility.



Figure 1.1: (a) The AAU Space Robotics drone tethered via Ethernet at the European Rover Challenge due to severe Wi-Fi interference. (b) A Ukrainian FPV drone adapted with a fiber-optic cable to counter electronic warfare in a contested environment, photo by [1] .

This experience at the ERC is a microcosm of a broader and increasingly critical problem inherent to unlicensed spectrum. Protocols such as Wi-Fi, which rely on Carrier-Sense Multiple Access (CSMA), are fundamentally cooperative and perform poorly under heavy contention. As the number of devices increases, the medium becomes a chaotic environment where predicting communication latency becomes infeasible [2].

The implications of this lack of reliability extend far beyond academic competitions and into the realm of mission-critical operations. In modern conflict scenarios, reliance on traditional licensed spectrums presents a significant vulnerability, as an adversary unconstrained by international law can easily target these frequencies with jamming or spoofing attacks. The operational consequence is a degraded or denied communication link, forcing military operators into drastic, suboptimal solutions. In Ukraine, for instance, drone pilots have resorted to tethering their aircraft with fiber-optic cables to bypass pervasive jamming, as seen in Figure 1.1b [3]. This reversion to a physical link, much like the Ethernet tether at the ERC, ensures control but fundamentally compromises the aircraft's mobility. This reality underlines the necessity for systems that can intelligently leverage alternative, less-contested bands. However, these bands

require a new paradigm of adaptable communication to overcome the inherent unpredictability demonstrated so clearly in the congested ERC environment.

For a remote actor, whether a human operator or an autonomous agent controlling a vehicle, the average latency of a communication link is often a poor indicator of its true performance. A link that is fast on average but suffers from occasional, severe latency spikes is unsuitable for time-sensitive applications like control signals [4]. The crucial piece of information is not the expected latency, but a reliable upper bound on the latency. This thesis addresses this challenge directly by asking:

How can we provide a robust, statistically sound measure of uncertainty for latency predictions to enable an intelligent and reliable interface selection to improve latency performance?

To answer this question, this thesis moves beyond traditional point predictions in machine learning. We propose a novel approach using Uncertainty-Aware Conformalized Quantile Regression. Instead of predicting a single latency value, our model predicts an interval for a high quantile of the latency, specifically the 95th latency quantile (Q95). This method, grounded in the rigorous framework of conformal prediction, provides a statistically valid guarantee that the true latency quantile will fall within the predicted interval with a predefined confidence level. This empowers a controlling actor with a quantifiable measure of risk, allowing it to make informed decisions about which combination of communication interfaces, in our case, WiFi and BLE, offers the highest probability of meeting the mission’s real-time requirements at any given moment. This work, focuses on developing and validating this method to add a critical layer of certainty to latency-aware interface selection, transforming unpredictable unlicensed wireless channels into more dependable communication links.

2 | Basic of Wireless Latency, MAC Protocols and Interface Diversity

The Industrial, Scientific, and Medical (ISM) bands has, due to their unlicensed nature, been widely adopted for communication technologies like Wi-Fi, BLE, and LoRa. This lack of a licensing cost makes the spectrum ideal for private consumer networks and low-cost IoT devices. However, the open nature of the ISM bands leads to a crowded and unpredictable environment, similarly to the case experienced by the AAU Space robotics team at the ERC competition which motivated this work. This unpredictable environment is the foundational problem this thesis addresses by developing a method to predict the 95th latency quantile while also providing a measure of the prediction's certainty. To understand the foundation of this problem, we first examine what factors contributes to the overall communication latency. This is followed by a detailed description about the medium access protocols for the two key technologies in this work, Wi-Fi and BLE. Finally, it will discuss high-level strategies for using multiple interfaces to achieve robust communication.

2.1 Deconstructing Communication Latency

The total time it takes for a data packet to travel from a source to a destination, known as end-to-end latency (E2E), is not a single, monolithic value. Instead, it's a composite of several distinct delays encountered along the transmission path. The total end-to-end latency is comprised of the following delays:

- Propagation delay (L_{prop}).
- Transmission delay (L_{trans}).
- Processing delay (L_{proc}).
- Queuing delay (L_{queue}).
- Medium Access delay (L_{ma}).

The total end-to-end latency is then given as the sum of the individual delay components as in Equation (2.1):

$$L_{\text{total}} = L_{\text{prop}} + L_{\text{trans}} + L_{\text{proc}} + L_{\text{queue}} + L_{\text{ma}}, \quad (2.1)$$

Understanding these individual components is crucial for analyzing and predicting network performance, particularly in wireless environments where variability can be significant. Throughout the individual delay components will be shortly introduced.

2.1.1 Propagation Delay

The propagation delay is the time required for the first bit of a signal to travel from the sender to the receiver. It depends on the physical distance between the two points and the propagation speed of the signal through the transmission medium [5]. For a given communication path and medium, the propagation delay is generally fixed.

2.1.2 Transmission Delay

The transmission delay is the time taken to push all of a packet's bits over communication link. The transmission delay is determined by the size of the packet, the transmission rate of the interface and the packet error probability. The transmission rate can change based on the quality of the channel, hereby making the transmission delay a variable delay [5].

2.1.3 Processing Delay

The processing delay arises when processing a packet. This includes tasks like examining the packet header and checking for bit errors. Processing delay is typically small on modern hardware but can vary slightly depending on the device's load and the complexity of the operations. It is often modeled as a fixed negligible value [6].

2.1.4 Queuing Delay

Packets often have to wait in queues at network devices if the device is busy handling other packets. This delay is highly dependent on the level of congestion in the network. If a link is heavily utilized or many packets arrive simultaneously, queuing delays can become substantial and are highly variable.

2.1.5 Medium Access Delay

In shared channels, such as the wireless ISM bands, there are protocols to prevent or manage collisions. The time a station spends waiting for an opportunity to transmit is the medium access delay. The delay is heavily influenced by the specific medium access control (MAC) protocol in use. In licensed spectrum the MAC can be centrally coordinated to ensure a stable latency, using different scheduling methods. However, in unlicensed spectrum, the medium access control is performed uncoordinated requiring the use of MAC protocols such as CSMA/CA, FHSS or duty cycling, resulting in a variable latency delay. Medium access delay is a significant contributor to overall latency in wireless systems, it is inherently random unlicensed spectrum, depending on the number of contending devices, and the level of interference.

In the following section we dive deeper into the medium access protocols that significantly influence the medium access delay, particularly for WiFi and BLE technologies.

2.2 Wireless Medium Access Protocol

To enable multiple networks to operate simultaneously in the same unlicensed spectrum, a framework of regulations and technical standards must be followed. In Europe, short-range devices are allowed to communicate in the ISM bands. Short-range devices are defined by the regulations ECC Recommendation 70-03, as devices with a low risk of interfering with other radio services [7]. This regulatory requirement sets the stage for technical standards organizations, like IEEE, to develop the specific protocols that ensure complaint behavior. The IEEE 802.11 (Wi-Fi) and 802.15 (BLE) standards are prime examples of such standardization, providing engineers with solutions that allow the spectrum to be shared by multiple uncoordinated networks. This section will investigate the two primary medium access strategies central to this thesis:

- Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) used by Wi-Fi.
- Frequency-Hopping Spread Spectrum (FHSS) used by BLE.

Finally, we will briefly examine duty cycling as another common method for ensuring fair spectrum access.

2.2.1 WiFi (IEEE 802.11) and CSMA/CA

WiFi, standardized under IEEE 802.11, is the predominant technology for Wireless Local Area Networks (WLANs) [8]. It enables devices to connect to a network or peer-to-peer wirelessly, mostly operating in the unlicensed and uncoordinated 2.4 GHz, 5 GHz, ISM bands. The core principle of WiFi's medium access control is a "listen before talk" mechanism, called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA is paramount for WiFi's ability to operate in the uncoordinated unlicensed spectrum [9].

Carrier Sensing(CS)

Before attempting to transmit, a station listens to the wireless medium, to assess if the channel is busy. The channel can be busy due to other system actively transmitting. This assessment is referred to as Clear Channel Assessment (CCA). The CCA is based on an energy detection threshold, IEEE 802.11 defines the threshold for the CCA as -62 dBm [10]. If the result of the CCA is negative due to power levels above -62 dBm, the WiFi station ends up in contention. The station will be in contention until the medium is sensed idle for a specific period called the Distributed Inter-Frame Space (DIFS) [9]. However, to reduce the probability of multiple stations transmitting simultaneously as the medium becomes idle a Collision Avoidance (CA) mechanism is used.

Collision Avoidance (CA)

The cornerstone of CA mechanism is a random backoff procedure. An example of the procedure is depicted in Figure 2.1. As the channel is assessed idle for a period of DIFS a backoff is time drawn from a uniform distribution with a range defined by the Contention Window (CW),

$$B \sim U(\min = 0, \max = CW),$$

where B is the number of backoff slots, each with a duration of $20 \mu s$ [9]. The station decrements the backoff timer as long as the channel remains idle. If the channel becomes busy while counting down, the timer is paused, resuming once the channel becomes idle again for a DIFS period. When the backoff timer reaches zero, The station transmits its packet.

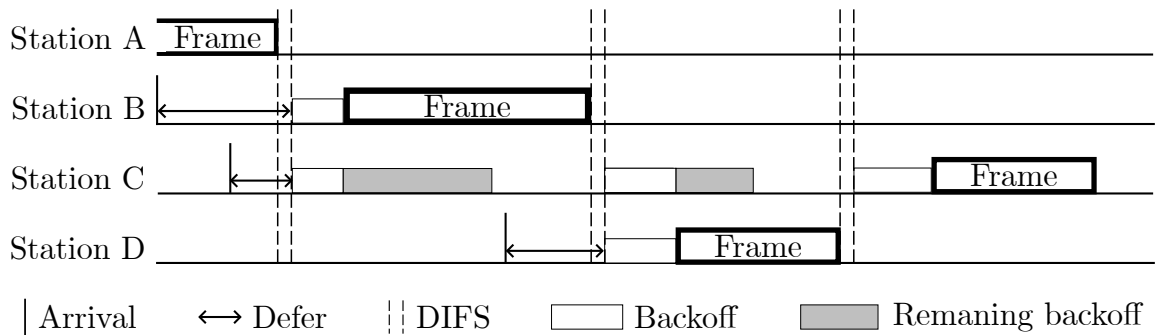


Figure 2.1: Example of the backoff procedure in an environment with four stations.

After successfully receiving a packet, the receiving station sends back an acknowledgement (ACK) before a DIFS time have passed. This ensures, the channel is idle while transmitting the ACK, essentially giving it the highest priority. In case the sending station does not receive an ACK within a certain timeout period, it assumes the packet was lost and will attempt to retransmit. If the packet is lost the CW size will be doubled, up to a maximum limit CW_{max} . This behavior is known as binary exponential backoff and reduces the probability of subsequent collisions. The exponential growth of the contention window is visualized by Figure 2.2.

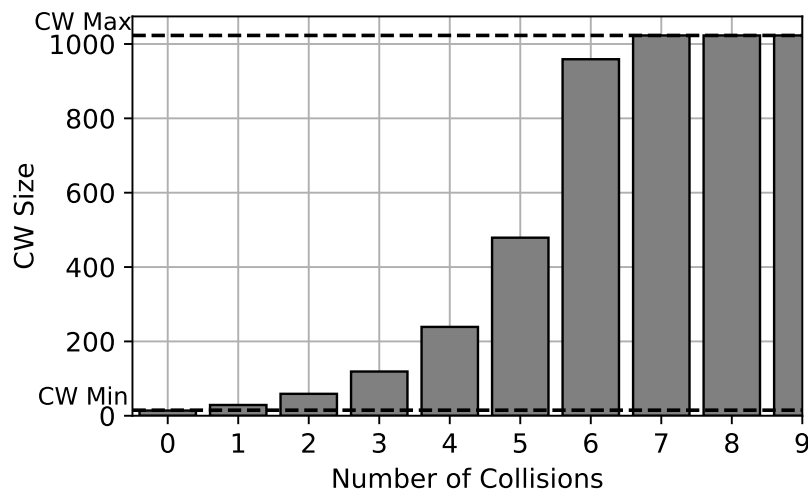


Figure 2.2: Example of exponential evolution of contention window CW. Inspired by [10].

Hidden and Exposed Terminal Problem

While fundamental to medium access, Carrier Sense (CS) is notably affected by the hidden and exposed terminal problems. The hidden terminal problem, depicted in Figure 2.3a, materializes when an interfering node Xia is hidden, out of range or obstructed, from a transmitting node Yoshi. Consequently, Yoshi CCA incorrectly indicates an idle channel, prompting a transmission which then collides at Zoya, who is within range of the interfering hidden terminal Xia.

The exposed terminal problem presents a different challenge depicted in Figure 2.3b. A transmitter Zoya correctly detects an ongoing transmission from another nearby terminal Xia and, as a result, its CCA indicates a busy channel, causing Zoya to defer. This becomes problematic if Xia's transmission poses no actual interference threat to Zoya's intended recipient Yoshi. In such a scenario, Zoya is exposed to a transmission that needlessly prevents it from communicating with Yoshi, thereby reducing overall network efficiency.

To solve these problems, IEEE 802.11 allows the use of a virtual sensing mechanism Request-To-Send/Clear-To-Send(RTS/CTS), to mitigate this problem. Before transmitting a data packet when the medium is idle, the station transmits an RTS frame, with a duration field which signals the time which the medium is occupied for the following data transmission. The receiving station transmits an ACK in the form of a CTS frame.

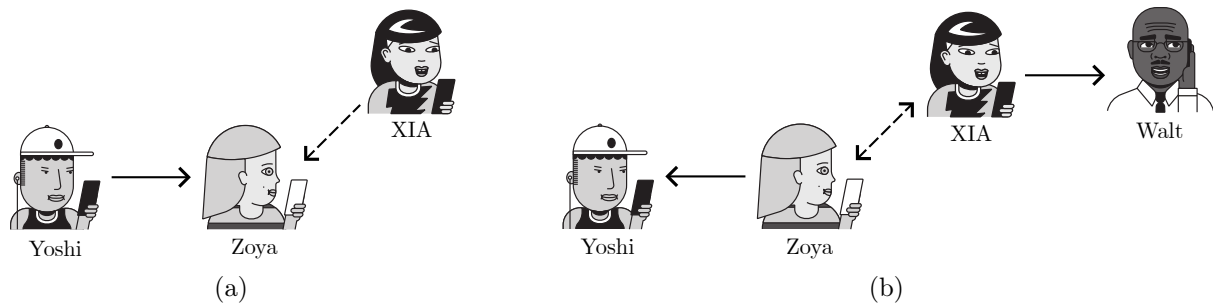


Figure 2.3: (a) Hidden Terminal problem. (b) Exposed Terminal problem. Dashed lines indicate interference from a terminal in another network.

2.2.2 Bluetooth Low Energy (BLE) and FHSS

To operate within the crowded 2.4 GHz ISM band, Bluetooth Low Energy (BLE) employs a different strategy from WiFi's "listen-before-talk" protocol. BLE is more aggressive and agile using a "Try things fast, learn, and adapt" protocol for spectrum sharing. Specifically, the cornerstone of BLE's interference mitigation strategy is called Frequency-Hopping Spread Spectrum (FHSS). The FHSS ensures coexistence between BLE devices and other ISM band devices by changing frequency channel at every connection event. The duration of a connection event in BLE ranges from 7.5 ms to 4 seconds [11]. The frequency hopping allows BLE devices to essentially operate over 37 different data channels lowering the chance of collisions.

As collisions will happen, BLE ensures reliability through an acknowledgment and retransmission scheme at the link layer. When a device sends a data packet, it expects an ACK from the receiving device. If a packet is corrupted or lost, indicated by a failed Cyclic Redundancy Check (CRC) or a timeout, the sender will retransmit the packet, using a Automatic Repeat Request (ARQ). Retransmissions are managed using sequence numbers (SN) and next expected sequence numbers (NESN) in the packet headers to ensure orderly delivery and identify lost or duplicate packets. These retransmissions, while vital for ensuring data eventually gets through, inherently introduce latency variability. Each retransmission attempt adds to the total time taken for a packet to be successfully delivered, making the latency profile of BLE connections dependent on the current interference conditions and the number of retransmissions required. Figure 2.4 showcases different scenarios, visualizing the impact of retransmissions on the latency.

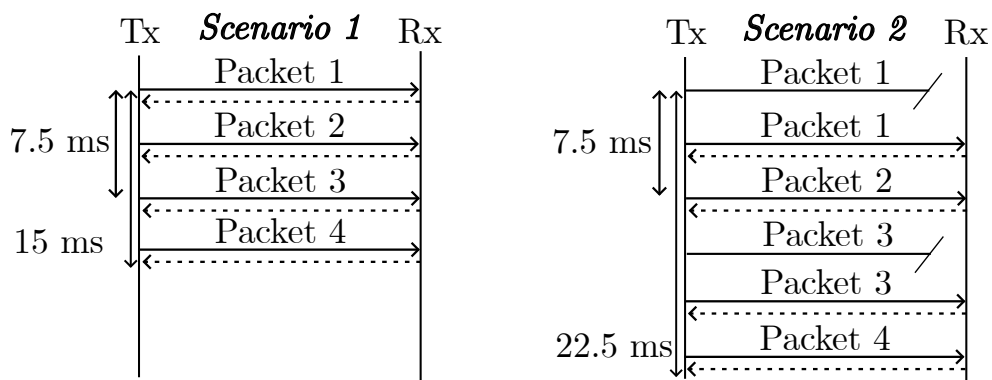


Figure 2.4: Message Sequence Diagram of the ARQ. Scenario 1: No collisions, scenario 2: Two collisions, Dashed lines are acknowledgments [12].

To lower the amount of collisions, BLE uses a variant of FHSS called Adaptive Frequency Hopping (AFH). Instead of using a fixed set of channels or hopping across all available channels blindly, AFH allows a BLE connection to dynamically assess channel quality and restrict its operation to "good" channels [11]. The link layer maintains an AFH Channel Map (CHMAP) which indicates the set of used and unused channels. This map is updated based on feedback regarding channel quality, allowing the system to avoid channels that are experiencing persistent interference, from sources such as nearby WiFi networks. The targeted hopping significantly reduce the probability of collisions with interfering signals, thereby enhancing link reliability. BLE uses two Channel Selection Algorithms (CSA) to implement AFH.

Channel Selection Algorithm 1

CSA 1 is a two step process. First step is calculating the Unmapped Channel Index (UCI) using Equation (2.2),

$$UCI_i = \begin{cases} (UCI_{i-1} + \Delta h_i) \bmod 37, & \text{if } i > 0 \\ \Delta h_i \bmod 37, & \text{if } i = 0 \end{cases} \quad (2.2)$$

where UCI_i is the current UCI and UCI_{i-1} is the last UCI and Δh_i is the hopIncrement. The hopIncrement is a random variable in the range 5 to 16, set when the connection is established [11].

In the second step, the UCI is mapped to a data channel (DC) from the set of used channels. If the UCI maps to an unused channel, the algorithm in Equation (2.3) is used,

$$DC_i = \begin{cases} (UCI_i, & \text{if } UCI_i \in CHMAP[] \\ CHMAP[RMI], & \text{otherwise} \end{cases} \quad (2.3)$$

where RMI is the remappingIndex, $RMI = UCI_i \bmod NUC$ and NUC is the number of used channels.

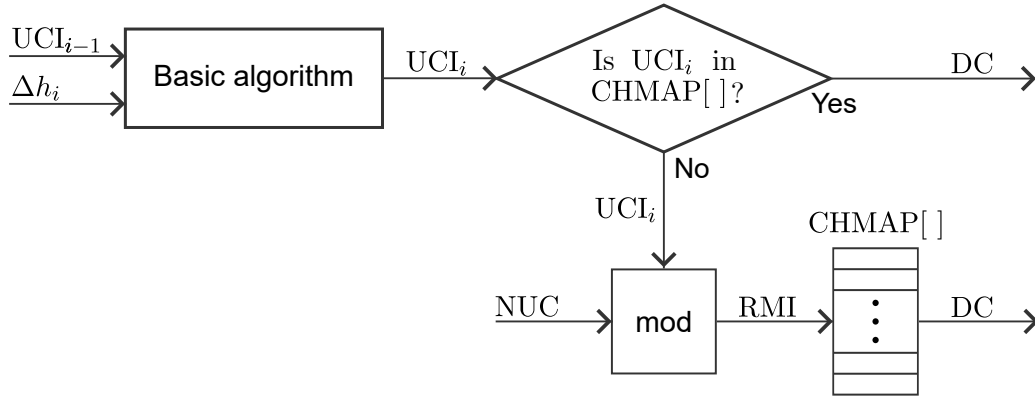


Figure 2.5: Flowchart of CSA 1 [11].

Channel Selection Algorithm 2

CSA 2 is a more complex algorithm that in turns ensures a more random channel selection. CSA 2 supports both channel events and subevents. However, this section will only cover how channel events are handled. The benefit of CSA 2 besides the support for subevents are its add randomness, through the use of a pseudo-random number generator, as illustrated by Figure 2.6. To make a channel selection using CSA 2 the following inputs are used:

- a 6-bit number of channels classified as used channels (NUC).
- A 16-bit fixed channel identifier (CI).
- A 16-bit connection event counter (CEC).

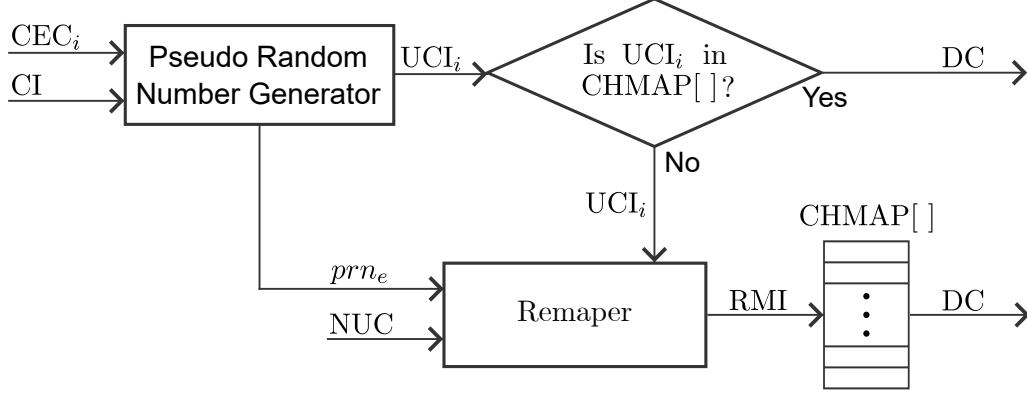


Figure 2.6: Simplification of the flowchart of CSA 2 [11].

The pseudo random generator function is depicted in Figure 2.7. It generates the random numbers through three permutation (PERM) and Multiply, Add, and Modulo (MAM) blocks. The PERM consist of separately bit-reversing the upper 8 input bits and lower 8 input bits. The MAM block is computed as $OUT = (17 \cdot a + CI) \bmod 2^{16}$.

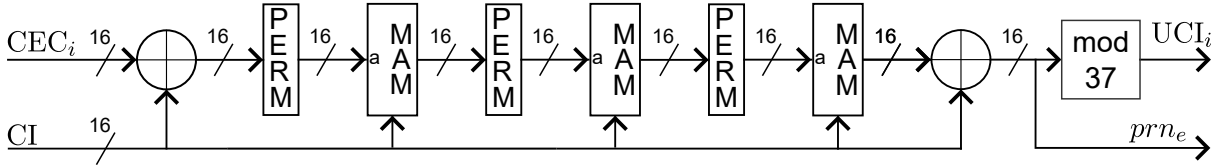


Figure 2.7: Pseudo-random number generator

If UCI is in the CHMAP it is used as the DC, otherwise it is remapped by the RMI as described by equation in Equation (2.4),

$$RMI = \left\lfloor \frac{NUC \cdot prn_e}{2^{16}} \right\rfloor, \quad (2.4)$$

where prn_e is the output of the random generator, in Figure 2.7 before the taking modulo 37.

2.2.3 Duty cycle

Beyond contention-based mechanisms like CSMA/CA and frequency hopping strategies like FHSS, duty cycling is another common method employed to manage spectrum access, particularly in unlicensed bands [7]. Duty cycling refers to the practice of limiting the proportion of time a device is actively transmitting relative to a defined observation period. The primary motivation for implementing duty cycle restrictions are to provide fair access to the shared spectrum by preventing any single device or network from monopolizing a channel, thereby mitigating interference potential. For instance, specific short-range device sub-bands have stringent limits, such as 1% or 0.1% duty cycle [7]. By ensuring devices transmit only for short bursts and remain silent for the rest of the time, duty cycling creates more opportunities for other co-located networks and devices to access the medium.

However, the imposition of duty cycles also has direct consequences on network performance. The most direct impact is a cap on the maximum achievable data throughput. If a device can only transmit for 1% of the time, its average data rate is inherently limited, regardless of the raw transmission speed of its radio. If a device has data to send but has already exhausted its duty cycle budget for the current observation window, it must wait until the next permissible transmission slot. This enforced waiting period can introduce significant and potentially variable latency, which can be detrimental for time-sensitive applications.

2.2.4 Effect on latency

It is clear that the different methods of accessing the medium, affect the latency in different ways when the medium is congested. In WiFi the added latency in a congested network scenario, is caused by the transmitter being in contention waiting for the channel to be idle. The time in congestion depends on number of stations in the area, how often they transmit and the length of their packets. For BLE the latency depends on the number of devices in the area and the number of used channels in the channel map. The MAC access delay for BLE is caused by retransmission of failed packets. As the maximum transmission unit in BLE is smaller than WiFi larger data must be divided and transmitted over multiple packets. A side-effect of dividing the data into smaller packets, is that each packet may experience collisions, cumulatively adding to the MAC delay, whereas WiFi simply transmits one large packet, thereby only accumulating a single MAC delay. As the MAC protocol have different behaviors it may be beneficial to combine interfaces with different protocols. Different methods of combining interfaces will be discussed in the following section.

2.3 Interface Diversity Module

There are multiple strategies which can be used in a multi interface communication system using either one or multiple interfaces simultaneously. In this section three transmission strategies, *Cloning*, *k out of N* and *Weighted* are presented [13].

2.3.1 Cloning

In cloning full copies of the data are transmitted over each of the N used interfaces. As only one message is required to decode the data, cloning is the most robust transmission strategy. It comes however with the cost of N fold redundancy, meaning it is the least energy efficient. Under the assumption of interface independence, the latency-reliability function of cloning is,

$$R(x, \gamma, B) = 1 - \prod_{j=1}^N (1 - R_j(x, \gamma_j \cdot B)), \quad (2.5)$$

where $R(x, \gamma, B)$ is the latency-reliability function, x the latency requirement, B is the packet size, γ the N dimensional interface data allocation vector with index γ_j for a specific interface, for the cloning strategy $\gamma = [1 \ \dots \ 1 \ \dots \ 1]^T$, Cloning is the most reliable of the techniques.

2.3.2 Splitting

Both *k out of N* and *Weighted* are strategies where instead of transmitting a full copy of the data over each interface, only a fraction of the data is transmitted on each. Splitting the data into fraction allows for a trade off between reliability and latency through the selected fraction size. To be able to utilize the splitting strategies the transmissions must be encoded using

error correction codes that uses fragmentation such as rateless or Reed Solomon codes. The receiver is able to decode message, if it receives coded fragments corresponding to approximately $100(1 + \epsilon)\%$.

Strategy: *Weighted*

The *Weighted* strategy differentiates itself by allowing different sizes of coded fragments to be transmitted over different interfaces. The latency reliability function of the *Weighted* strategy is the probability that the sum of successfully received fractions of data is above the threshold γ_d . This can mathematically be described as:

$$R(x, \gamma, B) = \sum_{h=1}^{2^N} d_h \prod_{j=1}^N G_j(x, \gamma_j \cdot B), \quad (2.6)$$

where

$$d_h = \begin{cases} 1, & \text{if } \sum_{j=1}^N c_{h,j} \cdot \gamma_j \geq \gamma_d \\ 0, & \text{otherwise,} \end{cases} \quad (2.7)$$

d_h ensures that only successful scenarios are counted towards the latency reliability function. $c_{h,j}$ is the element in the h -th row and j -th column in the $2^N \times N$ matrix \mathbf{C} , which contains vectors describing all the possible combinations of data reception on each interface. The element can be 0 or 1 depending on whether the interface j 's transmission is successful in scenario h .

$$G_j(x, \gamma_j \cdot B) = \begin{cases} R_j(x, \gamma_j \cdot B), & \text{if } c_{h,j} = 1 \\ 1 - R_j(x, \gamma_j \cdot B), & \text{if } c_{h,j} = 0, \end{cases} \quad (2.8)$$

describes the latency reliability probability that the RAN j experiences the outcome in scenario h .

Strategy: *k out of N*

In *k out of N* the data is split into N equally sized coded fragments where the receiver must receive at least k of the to decode the data. The strategy allows for a tradeoff between reliability and latency as larger redundancy, gives higher reliability at the cost of transmission time.

The latency reliability function of the strategy *K out of N* is the same as the weighted but with $\gamma_j = \frac{1}{k}$ for $j = 1, \dots, N$. It should be noted that *K out of N* is only optimal if the three RAN are the same.

3 | System Description

This chapter gives an overview of a multi-interface system, selecting interfaces suitable for video transmission, while taking into account the interface respective spectrum policies and communication link conditions. Furthermore, the chapter proposes the outline of a uncertainty aware latency predictor, which is the main contribution of this thesis. As stated in the Chapter 1 and Chapter 2, each interface comes with its own spectrum policy CSMA and FHSS, posing both advantages and disadvantages to using either interface.

3.1 Overview of the Video Transmission System

The purpose of the system is to select the combination of interfaces for transmission, optimizing for low energy consumption, end-to-end latency and high video quality.

Figure 3.1 showcases a system diagram illustrating the data flow of the video transmission system, which the latency predictor is designed to be a part of. The actor module, positioned in the top right corner of Figure 3.1, is responsible for choosing the optimal combination of interfaces. The actor module has two parameters to control the system, its choice of compression rate and interface selection. Figure 3.1 is divided into two data flows, the video frames illustrated in black and the state data used to inform the actor, illustrated in gray.

3.1.1 Video Frame Data Flow

At the top left is a leaky bucket feeding raw video frames (r), into a frame queue. This can be moodle as a constant arrival rate for the frame queue. A raw video frame is taken out of the queue, following a First In First Out (FiFo) principle, into the compression module. The compression module compresses the video using a compression rate (κ) chosen by the actor module. In the interface diversity module, compressed frames (c) are encoded and packaged according to the diversity technique and interfaces selected by the actor module (a). The actor chooses among the diversity techniques described in Section 2.3 which dictates the packaging of the transmissions. The selected interfaces, add a time stamp to the data (d_w, d_b) after which, the packets (b, w) are transmitted to a receiver. Throughout this thesis WiFi is used as the CSMA interface and BLE as the FHSS interface.

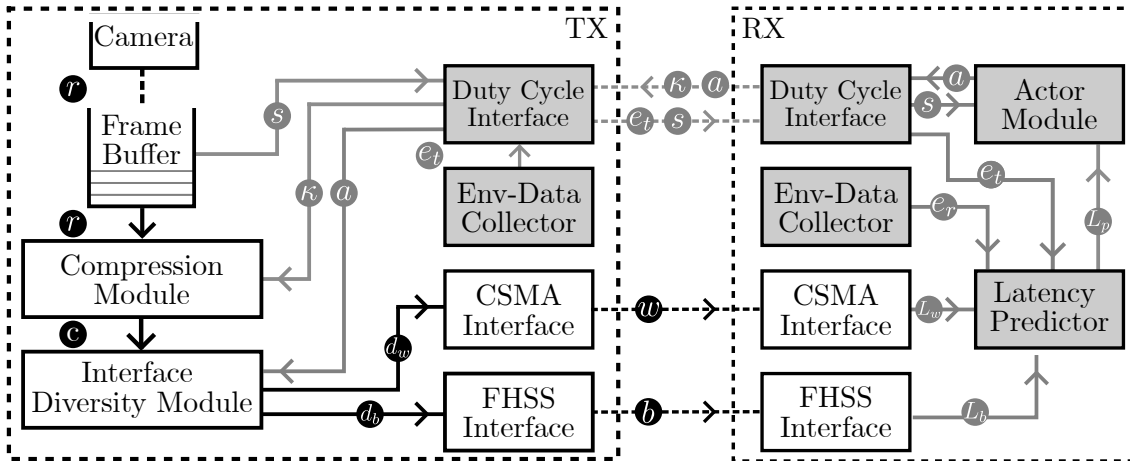


Figure 3.1: A block diagram of the envisioned video transmission system

3.1.2 State Data Flow

For the actor to make a decision on an interface selection and compression rate for optimizing energy consumption, end-to-end latency and video quality, it must have information about the state of the wireless environment. As discussed in Chapter 2 the different spectrum policies latency performance, is impacted by differently by the wireless environment. The gray modules and information flow, describes the state of the wireless environment, that will affect the interface latencies. As such, the state information is primarily used by the latency predictor to estimate the 95th latency quantile. The state information is gathered through WiFi and BLE's protocols or in the RF receivers. From the CSMA and FHSS interfaces channel state information and latency measurements are collected (L_w , L_b). The ENV-Data Collectors, collecting information about the wireless environment state, from both the RX (e_r) and TX (e_t) devices. Information is exchanged periodically between the TX and RX devices through a separate duty cycling interface.

The Latency predictors task is to give a 95 latency quantile guarantee, from which the actor module can select diversity techniques and interfaces, taking into account the amount of frames in the frame buffer (s) to optimize the end-to-end latency of the frames, energy consumption and video quality.

As mentioned, the uncertainty-aware latency predictor is the main contribution of this thesis. The following section outlines the operation of the actor module, which specifies requirements and motivates the implementation of the latency predictor.

3.2 Actor Module

The objective of the actor module is to facilitate energy-efficient, reliable, low latency communication, over a heterogeneous network, consisting of multiple radio access networks (RANs) with different characteristics. As such, the problem is formulated as a network-selection problem along with a resource allocation problem of multiple devices using shared communication resources.

The actor module's task is defined as a multi-objective optimization problem, aiming to optimize the communication network in terms of energy efficiency and video distortion, which occurs as a result of video compression. Intuitively, compressing video reduces the total amount of bits to transmit, thereby improving network performance metrics like energy efficiency and latency, however at a cost of added distortion or potentially lost information. The optimization problem is subject to latency and reliability constraints, as well as device/RAN specific resource limitations. The network selection shall take into account the dynamic propagation environment, in which, the system operate. The actors task is summarized to allocate the available RANs in a way such that:

- Energy consumption is minimized - Essential for maximizing the battery life and operational efficiency.
- Signal distortion is minimized - Ensures minimal information loss due to compression. Introduces a trade-off between smaller data loads and signal distortion.

The above optimization problem is subject to network resource constraints as well as the interface-specific mac protocols, discussed in Section 2.2. All of the optimization objectives are encapsulated in a utility function U , which will serve as the objective function of the overarching optimization problem.

The utility function is formulated as Equation (3.1):

$$U = \sum_{j=1}^R (\alpha E_j) + \beta D, \quad (3.1)$$

where j denote the RAN, E is the energy consumption, D is distortion as a result of video compression and α, β , are weighting coefficients to represent the relative importance. In Appendix A.2 an empirical model of the compression-distortion relationship is made, as well as a model of the energy consumption for transmission.

This thesis considers a scenario, in which the following constraints are imposed on the device:

- The reliability of communicating on a RAN j must exceed R_{th} , defined in terms of latency as $P(L \leq L_{max}) \geq R_{th}$.
- The device can access any number $j \in \{1, \dots, R\}$ RANs, each with different network characteristics.
- The device may split its data transmission among multiple RANs. The fraction of data the device wishes to transmit on a given RAN j is denoted P_j with $P_j \leq 1$ and $\sum_{j=1}^R P_j \geq 1$
- The device must consider relevant spectrum policies when selecting the RANs.
- The data may be compressed by a ratio of κ before transmission, thereby improving the energy efficiency and latency but at a cost of adding distortion.

The optimization problem, with its constraints, can be mathematically presented as Equation (3.2a).

$$\min_{P_j, \kappa} \sum_{j=1}^R \alpha E_j + \beta D \quad (3.2a)$$

s.t.

$$P(L \leq L_{max}) \geq R_{th}, \quad (3.2b)$$

$$0 \leq P_j \leq 1, \quad (3.2c)$$

$$\sum_{j=1}^R P_j \geq 1, \quad (3.2d)$$

$$0 \leq \kappa \leq 1, \quad (3.2e)$$

$$P_j \kappa B \rho \leq r_j \quad (3.2f)$$

where L, L_{max} is the observed latency and maximum allowable latency respectively, R_{th} is the reliability threshold, defined from the probability of satisfying the latency requirement. As such, the optimization problem is defined as minimizing the energy consumption and signal distortion, while satisfying latency, reliability, compression and data bandwidth constraints. For the actor to adhere to the constraint 3.2b, it must have reliable information about the expected worst-case latency of each available RAN. A discussion about the practical implementation of such an actor module, is further discussed in Chapter 7.

3.3 Delimitation: Uncertainty-Aware Latency Predictor

The total latency of a packet transmission is in Equation (2.1) expressed as the sum of multiple components in the system.

$$L_{\text{total}} = L_{\text{prop}} + L_{\text{trans}} + L_{\text{proc}} + L_{\text{queue}} + L_{\text{ma}}, \quad (2.1)$$

where, L_{proc} is the latency incurred due to video compression, which is assumed to be a function of the compression rate κ ; L_{prop} is the propagation delay, which negligible in this system, due to the short distances involved; L_{trans} is the transmission (or serialization) delay, defined as the packet size divided by the link bit rate, and is influenced by both the communication interface and the current channel state; L_{ma} represents the medium access delay, highly dependent on the RF environment and the contention level at the time of transmission; L_{queue} denotes the queuing delay, which is also a potentially dominant factor and depends on the frame arrival rate, the transmission delay (L_{trans}), and the medium access delay (L_{ma}).

In this thesis, we focus on the transmission and Medium access delay as these are directly influenced by the choice of interface. As the medium access delay is random and modeled by a stochastic process, the latency follows a distribution. The purpose of the latency predictor is to give the actor an estimate of the tail of the latency distribution, in this case defined as the 95th quantile. The latency predictor shall be able to predict this quantile, based on the state of the RF environment (e_t, e_r), as well as channel state information (L_w, L_b) from the interfaces, as illustrated by Figure 3.2.



Figure 3.2: In-/output model of the latency predictor.

As the prediction from the latency predictor may fail to capture the underlying distributions 95th quantile precisely, the prediction will come with some uncertainty. This thesis aims to capture the prediction uncertainty, by giving an interval where we guarantee the 95th latency quantile of the underlying distribution lies within with 90% confidence. This interval is what makes the latency predictor Uncertainty-Aware. The size of the interval can be used as a direct measure of the certainty of the latency predictor in a given environment. This uncertainty can then be used as a metric by the actor when deciding the interface diversity technique and interface.



Figure 3.3: In-/output model of the latency predictor with a confidence interval around the 95th quantile.

The mapping between the channel link conditions and the 95th latency quantile is far from trivial, partly due to the medium access protocols. We therefore opt to use a Deep Neural Network (DNN) to learn the mapping function.

3.3.1 Available RF Environment State Information for DNN Training

To be able to train a DNN to predict the latency, a dataset must be constructed. The input features of the DNN will be information about the state of the RF environment that gives insight into the transmission and medium access delays. The labels will either be the corresponding observed 95th latency quantiles or a parametric description of the latency distribution. The idea behind having a latency predictor is to predict the latency of an interface, regardless if the interface has been used for data transmission. As such the RF environment information must be periodically collected and accessible for an interface, which has not been selected for data transmission

The transmission delay are impacted by the bit rate offered by the interface, which is impacted by the SINR of the channel. The SINR affect the receivers ability to decode the received signals resulting in the loss of packets. Both BLE and WiFi, have the ability to change their modulation scheme hereby changing the required SINR and the achievable throughput. BLE support four radio modes with varying data rates Normal (1M PHY), transmitting at 1 Mbps, high-speed (2M PHY) at 2 Mbps and two types of long-range (Coded PHY S2 and S8), supporting rates of 0.5 Mbps and 0.125 Mbps [11]. The throughput of WiFi is dictated by the MCS table, seen in Section 4.3.3.

An investigation of the available information, describing the RF environment is conducted in Appendix B.1. In this investigation two measures of SINR are easily accessible by off-the-shelf hardware: the Received Signal Strength Indicator (RSSI) is collected by an ADC measuring the voltage just before the demodulator [14]. For some WiFi chipsets the CSI is available. The CSI is calculated on the pilot signals L-LTF and HT-LTF in the WiFi preamble, illustrated by Figure 3.4 [15].

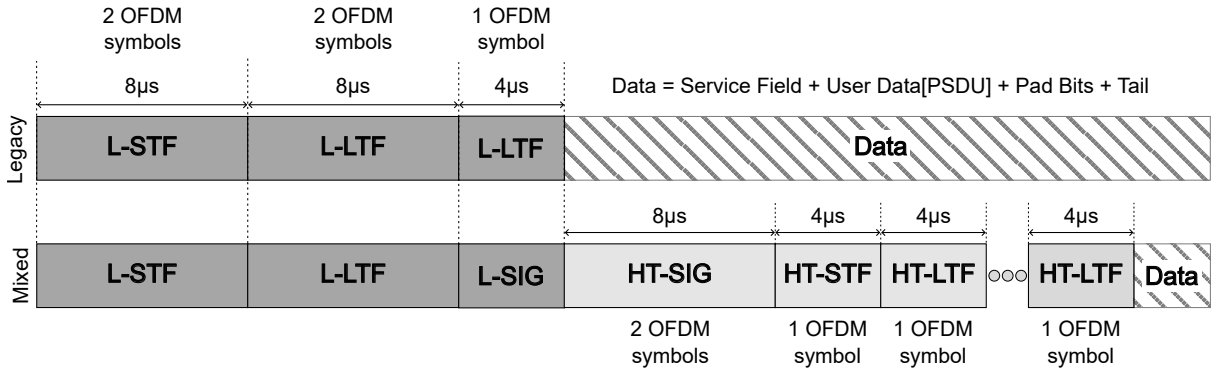


Figure 3.4: Preamble of 802.11n PHY frame [15].

The CSI contains the channel gain, obtained from the channel impulse response, represented by two bytes: a real and imaginary part of the channel gain for each WiFi subcarrier. The input space can be shrunk by calculating the magnitude of each subcarriers channel gain and even further by calculating an SNR using Equation (3.3):

$$\text{SNR} = \frac{\sum_{i=0}^{63} 10 \cdot \log_{10} (P_{\text{tx},i} \cdot |h_i|)}{\text{Noise floor}}, \quad (3.3)$$

where $P_{\text{tx},i}$ is the power of the signal in the i -th subcarrier, and $|h_i|$ is the channel gain of the i -th subcarrier.

To ensure channel measurements are available for an interface not in use, sounding packets are to be sent at fixed intervals t_s . In Figure 3.5, this behavior is captured in a message sequence diagrams, taking into account specific protocols transmission behavior. The interval t_p indicates the rate, at which state data is shared between the interfaces, allowing for a new latency prediction. The impact of the interval t_p , is further investigated in Section 5.2.

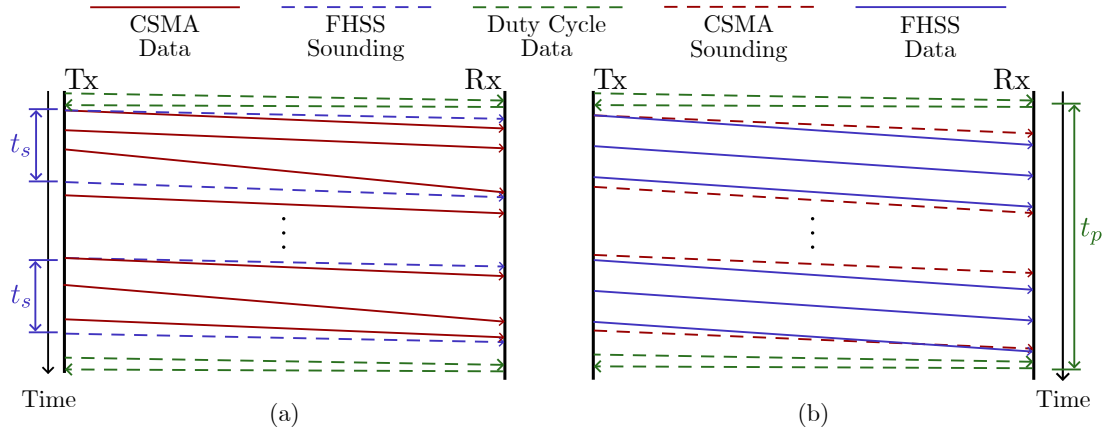


Figure 3.5: Two examples of message sequence diagrams, a) showcasing WiFi as the active interface with frequent sampling and periodic sampling on the inactive BLE interface, and b) showcasing the opposite scenario.

The medium access delay, is dependent on the available resources, and how many devices are trying to access the resource. For WiFi there are a fixed amount of resources available, however the number of devices trying to use these resources vary significantly. More active devices in an area, result in a higher probability of experiencing contention and for that contention to last for a longer time. The 802.11 standard have been studied to find data that could give an indication of the number of WiFi devices in the area. Here we found the two management frames, more specifically the beacon and probe frame, used in the device discovery phase of the IEEE 802.11 standard [9]. As documented in Appendix B.1 these frames are not encrypted and information such as the SSID and operational channel numbers are accessible. Meaning the number of WiFi devices can be used as an input to the latency predictor module.

For the BLE interface neither the number of devices nor the available resources are fixed. As documented in Appendix B.1, BLE also makes use of a discovery algorithm, which gives an indicator of the number of BLE devices on the network. The amount resources in BLE vary with the number of used channels in the channel map due to adaptive frequency hopping.

The training dataset for the DNN consists of latency measurements from WiFi and BLE. The data is gathered by simultaneously transmitting video frames on both interfaces. The latency is then calculated by comparing the timestamp of the received packets (w, b).

To be more in speed up development, the latency predictor models are trained on simulated data. As mentioned, an investigation into available data has been conducted, and simulated inputs are chosen carefully, to not include unrealistic information about the environment.

4 | Data Generation and Simulation Framework

This chapter describes the simulation framework developed to generate training data for latency estimation in WiFi and BLE communication systems. The simulator models link-level behavior, employing appropriate models for signal propagation, interference and MAC-level access control. The generated datasets are subsequently used to train and calibrate deep learning models which predict the resulting latency given information about the channel condition. The chapter is organized into sections, providing a high-level overview of the modules comprising the link-level simulator followed by the specific implementation of each module, and finally a section summarizing the generated datasets for subsequent training of the DNN models and uncertainty calibration.

4.1 Modularization of Data Generation

To facilitate parallel development in the design and implementation process the simulator is modularized into four independent sub-systems.

- Environment Generator: Creates randomized wireless environments and device placements.
- WiFi Simulator: Simulates WiFi link behavior under varying interference and contention.
- BLE Simulator: Simulates BLE link behavior with adaptive frequency hopping and re-transmissions.
- Data Generator: Orchestrates the pipeline, loading simulation parameters from Table 4.1, invokes each subsystem and saves the results as in Table 4.2.

Figure 4.1 visualizes the data flow and inter-dependencies between sub-systems. This modular design facilitates highly reusable and easily tunable subsystems.

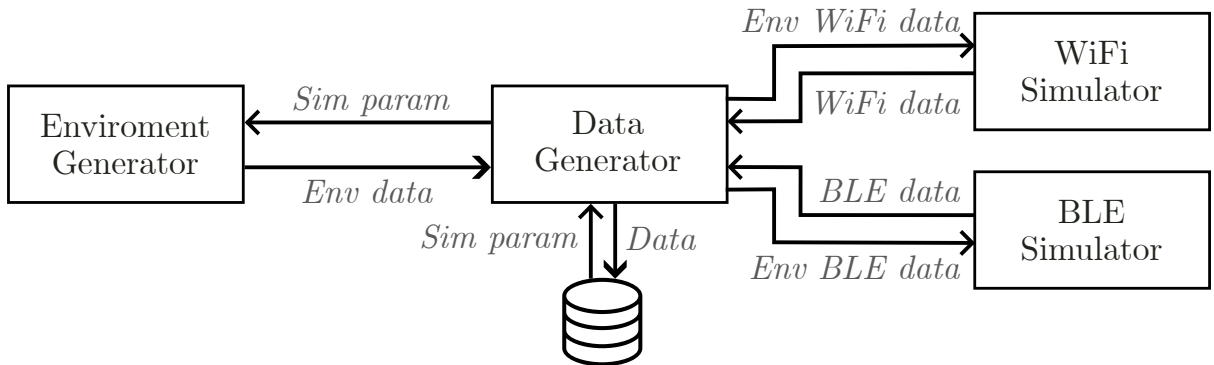


Figure 4.1: Data generation subsystem overview

The *Environment Generator* is responsible for generating the wireless environment and randomly placing transmitters, receivers and interfering devices. It is also responsible for defining the BLE channel map, setting up the WiFi contention mechanism and determining the number of WiFi/BLE interferers and hidden terminals.

The *WiFi simulator* simulates the transmission of a video frame over WiFi with a fixed frame size of 400 kB. It outputs the SINR and resulting latency per transmission. Similarly the *BLE Simulator* models transmissions of 20 kB frames and provides SNR and latency outputs. The simulation parameters are summarized by Table 4.1 and the generated data storage is summarized by Table 4.2.

Simulation Parameters		
Description	Key	Type
Number of environment to generate	num_env	int32
Number of simulations per environment	num_iterations	int32
Simulation environment dimensions [m,m]	env_size	tuple(int32, int32)
Transmit power [dBm]	tx_power	int32
WiFi Channel	channel	int32
CSMA Thresholds power [dBm]	csma_threshold	int32
Min & max number of WiFi interferers	num_WiFi_interferers	tuple(int32, int32)
WiFi frame size [B]	wifi_frame_size	int32
Min & max activation probability of WiFi interferers	random_activation_prob	tuple(float32, float32)
Size of BLE frame size in bytes	ble_frame_size	int32
BLE transmission mode	ble_transmission_mode	str.
Min & max number of BLE interferers	num_ble_interference	tuple(int32, int32)

Table 4.1: Simulation parameters, input for the data generation system.

Data		
Description	Key	Type
Index of simulated environment	env_num	int32
BLE SNR values for each environment	BLE_snr	List(Array(float32))
BLE latency values for each environment	BLE_latency	List(Array(float32))
Number of available BLE channels in each enviroment	CH_Map	List(int32)
Number of BLE interferers in each environment	BLE_interferers	List(int32)
WiFi SINR values for each environment	WiFi_snr	List(Array(float32))
WiFi latency values for each environment	WiFi_latency	List(Array(float32))
Postions of interferers causing contention at Tx for each environment	WiFi_beacons_CSMA	List(Array(tuple(float32, float32)))
Postions of interferers causing contention only at Rx for each environment	WiFi_hidden_terminals	List(Array(tuple(float32, float32)))

Table 4.2: Data structure for output of data generation system.

4.2 Environment Generator

The first step in building the simulator is defining the environment. This means creating a bounded field, where transmitters, receivers and interfering devices are placed within. The field generation can be mathematically described using point processes, defining how the devices are spatially distributed.

The simulator employs the simplest field generation approach, in which devices are uniformly and independently distributed over the simulation area, with the only constraint being the position must be unique. An example of a generated field is showcased by Figure 4.2.

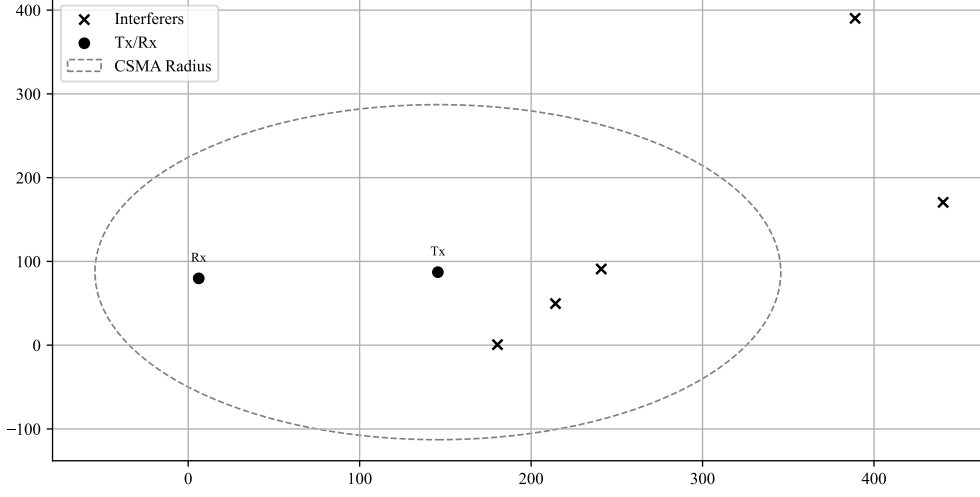


Figure 4.2: Generated simulation environment with random placement of devices.

This type of point process is known as a Binomial Point Process (BPP). The simulation area is defined as a finite 2D region A in the Euclidian plane.

$$A = [0, W] \times [0, H] \subset \mathbb{R}^2 \quad (4.1)$$

where W and H denote the width and height of the simulation area. Each device in the simulation is assigned a position \mathbf{p} , which is randomly drawn from a BPP over A .

$$\mathbf{p}_i \sim U(A), \quad i = 1, 2, \dots, N \quad (4.2)$$

where N is a fixed number of devices, alternatively determined by a random variable. Once the field is generated and devices are placed, the Euclidian distance between any two devices is computed as:

$$d_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\| \quad (4.3)$$

In WiFi, interference may influence the spectrum policy CSMA. The simulator defines a CSMA radius R_{CSMA} , which determines whether a device is able to cause contention at the transmitter. Devices are classified into CSMA and non-CSMA devices depending if they are within the contention radius R_{CSMA} from the transmitter:

$$\|\mathbf{p}_{\text{int}} - \mathbf{p}_{\text{tx}}\| \leq R_{\text{CSMA}} \quad (4.4)$$

The radius is determined based on a power threshold of -62 dBm and varies with the path loss model.

Extensions for Environment Generator

While the current simulator assumes uniform device placement, real-world networks often exhibit spatial clustering and environmental constraints. This means, patterns often occur in devices placements, where clustering may have implications on the achievable throughput. A more realistic placement model can be incorporated into the simulator by moving from a BPP to a clustered point process. Likewise, placement constraints can be arbitrarily added to the point

process. Another added complexity, which could increase the realism of the simulator is the inclusion of time-varying environments, i.e. mobility models. These additions are, however, not explored in this thesis.

4.3 WiFi Simulator

This section documents the development of a WiFi link-level simulator designed to generate a dataset which maps channel state and interference conditions to an achievable latency. While traditional analytical models have been used historically to derive closed-form approximations for these relationships. These models often rely on simplifying assumptions which fail to capture the full complexity of real-world wireless environments.

The challenges of accurately representing a wireless environment arise from the stochastic nature, where a multitude of factors interact with the communication link:

- A varying amount of interfering devices cause dynamic signal degradation.
- Spectrum access policies CSMA, introduce probabilistic access delays that depend on the network congestion.
- Stochastic channel variation cause a randomly fluctuating link quality.

The problem quickly grows in complexity when trying to capture these real-world variations, and deriving an accurate analytical model for latency becomes infeasible.

In order to generate a WiFi dataset that effectively captures the relationship between channel state and performance metrics, the simulator must employ appropriate models for each distinct aspect of WiFi link-level communication. This includes models for

- Signal propagation: Path loss and channel fading.
- Interference effects: Signal degradation and device behavior.
- CSMA access control: Access delay, network congestion.
- Performance evaluation: Channel statistics, achievable latency and throughput.

The remainder of this section documents the modular design of the link-level simulator, the applied models and design choices.

4.3.1 Propagation Model

After defining the spatial environment and device placement in the environment generator, the next step is to model the propagation of signals between devices. A signal propagating through the wireless medium is subject to attenuation and distortion depending heavily on the propagation distance and the propagation environment. The signal degradation can often be attributed to the following factors [16]:

- Path loss: Deterministic signal attenuation, increasing with distance.
- Shadowing: Stochastic large-scale fading, caused by varying environmental obstructions.
- Small-scale fading: Stochastic fluctuations in signal power due to multipath propagation, reflections, diffraction and scattering.

Since the simulator focuses on link-level performance in static deployments, averaging over many channel realizations, the small-scale fading effects are omitted, but are typically implemented as

a Rayleigh or Rician distributed random variable [16]. The received power at the receiver after path loss and shadowing is calculated as:

$$P_{rx} = P_{tx} - \text{PL}(d) + X_\sigma \quad (4.5)$$

where P_{tx} is the transmission power, $\text{PL}(d)$ is the deterministic path loss over distance and X_σ is the stochastic shadowing term.

Path Loss Model

The path loss model follows a two-region model defined by *IEEE 802.11-2014* [17]. The IEEE specification define models for Urban Micro (UMi) and Urban Macro (UMa) scenarios [17]. The simulator implements the UMi model, which includes both a LOS and NLOS path loss model as described by Equation (4.6) and Equation (4.7) [17].

$$\text{PL}_{\text{LOS}}(d) = \begin{cases} 22.0 \log_{10}(d) + 28 + 20 \log_{10}(f_c), & \text{if } 10 \text{ m} \leq d \leq d_{bp} \\ 40 \log_{10}(d) + 7.8 - 18 \log_{10}(h_{AP} - 1.0) \\ \quad - 18 \log_{10}(h_{STA} - 1.0) + 20 \log_{10}(f_c), & \text{if } d_{bp} \leq d \leq 5000 \text{ m} \end{cases} \quad (4.6)$$

$$\text{PL}_{\text{NLOS}}(d) = 36.7 \log_{10}(d) + 22.7 + 26 \log_{10}(f_c), \quad \text{if } 10 \text{ m} \leq d \leq 2000 \text{ m} \quad (4.7)$$

where d is the distance between the transmitter and receiver, d_{bp} is the break-point distance marking the switch between path loss models, f_c is the center frequency of the used WiFi channel as given by Table 4.3, h_{AP} and h_{STA} are the height of the transmitter and receiver respectively. The break-point distance is given by Equation (4.8) and a LOS probability is defined by Equation (4.9) [17].

$$d_{bp} = \frac{4(h_{AP} - 1.0)(h_{STA} - 1.0)f_c \times 10^9}{c} \quad (4.8)$$

$$P_{\text{LOS}}(d) = \min\left(\frac{18}{d}, 1\right) \times (1 - e^{-d/36}) + e^{-d/36} \quad (4.9)$$

Channel	Center Frequency
1	2.412 GHz
6	2.437 GHz
11	2.462 GHz

Table 4.3: Commonly used WiFi channels and their center frequency.

The path loss Equations 4.6 and 4.7 illustrate the difference in signal attenuation over distance for LOS and NLOS scenarios. The path loss is visualized by Figure 4.3, using a transmitter and receiver height of 10m as defined in the IEEE specification for AP to AP communication [17]. Figure 4.3 showcases a significantly higher signal degradation in NLOS conditions.

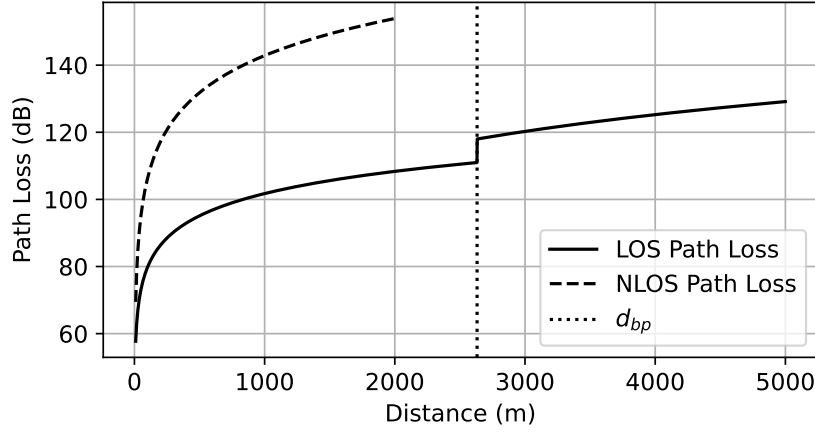


Figure 4.3: Plot of LOS/NLOS path loss over distance, with highlighted break-point distance, marking the switch in LOS path loss models.

Shadow Fading Model

For capturing the effects introduced by physical obstructions in the propagation path, a model for the stochastic shadowing X_σ is necessary. Shadowing is implemented by making random draws from an appropriate distribution, making the signal strength follow a probability distribution centered around the path loss, rather than a fixed value.

Shadowing is typically modeled as a log-normal distribution $X_\sigma \sim \text{Lognormal}(0, \sigma^2)$, with the standard deviation being determined empirically. The IEEE 802.11-2014 specification has documented shadow fading standard deviations as summarized by Table 4.4.

Scenarios	Shadowing Std (dB) (LOS)	Shadowing Std (dB) (NLOS)	Shadowing Std (dB) (Outdoor-to-Indoor in NLOS)
$AP \leftrightarrow STA$	3	4	7
$AP \leftrightarrow AP$	3	4	7
$STA \leftrightarrow STA$	3	4	7

Table 4.4: Shadow fading standard deviations for outdoor path losses, as per *IEEE 802.11-2014* [17].

In the simulator this is implemented by letting the LOS be determined by a Bernoulli r.v. with success probability according to Equation (4.9), that is:

$$\text{LOS} \sim \text{Bernoulli}(P_{\text{LOS}}(d)). \quad (4.10)$$

After which, the shadowing is determined as a draw out of the corresponding log-normal distribution. The probability density function for both the LOS and NLOS shadowing distributions are showcased by Figure 4.4.

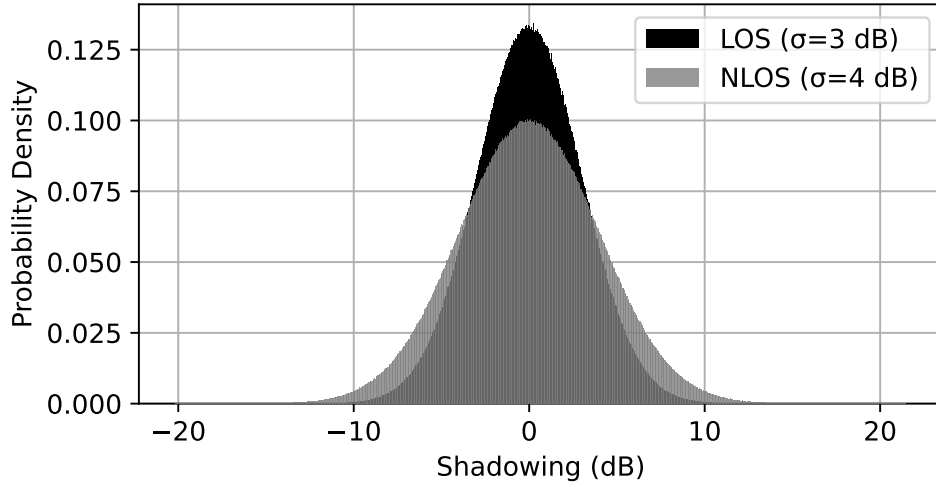


Figure 4.4: Histogram showcasing the pdf of LOS and NLOS shadowing distribution.

With the path loss and shadowing models in place, we can now compute the received power and signal attenuation in the case of just two communicating devices. The next step is to take into account the presence of interfering devices, whose signals are also prone to the effects of this propagation model. As such the propagation model is foundational to accurately computing the SINR and thereby the achievable throughput and latency.

4.3.2 Interference Model

While the propagation model allows for computation of the received power between two communicating devices, interference from other nearby devices may significantly impact the link quality. The simulator models interference, both in the sense of adding noise and also conflicting with the WiFi spectrum policy CSMA/CA. This way, interfering devices may actively cause an increase in the expected latency, or passively degrade the SINR. In order to implement this interference model, appropriate models must be defined for:

- CSMA/CA behavior: Carrier sensing threshold, interferer activation behavior, backoff and contention time.
- Interference power: Aggregation of interference power at receiver, subject to propagation model.
- SINR computation: Extension of SNR from the propagation model to include the aggregate interference power.

Classification of Interferers

As illustrated by Figure 4.2 and described by Equation (4.4), the simulator differentiates between contention-based interference and general uncoordinated interference. This distinction is crucial, since it determines which devices follow medium access rules, and which devices act as uncoordinated interference sources.

Contention-based interferers are defined as devices within R_{CSMA} , these are assumed to adhere to the WiFi medium access protocol, meaning only one device can transmit at a time. The contention radius is defined from the 802.11ax energy detection threshold, which defines the

threshold for being granted a CCA as $P_{ed} = -62$ dBm [17]. Using the propagation model, the contention radius R_{CSMA} is determined as:

$$R_{\text{CSMA}} = \arg \min_d P_{rx}(d) \geq P_{ed} \quad (4.11)$$

where P_{rx} is the received power subject to the propagation model and P_{ed} is the energy detection threshold. The CCA is granted if no other CSMA device is transmitting:

$$\text{CCA} = \begin{cases} 1, & \sum_{i \in \mathcal{N}_{\text{CSMA}}} A_i = 0 \\ 0, & \text{otherwise} \end{cases} \quad (4.12)$$

with $\mathcal{N}_{\text{CSMA}}$ being the set of CSMA devices and A_i being a binary activation variable. The CSMA access rule ensures transmission is only allowed if no other CSMA-enabled device within the contention radius R_{CSMA} is active. This check is performed at the transmitter, however the total interference power affecting the SINR is aggregated at the receiver.

Uncoordinated interferers are defined as devices outside the contention radius, these devices transmit independently and do not cause contention at the transmitter. Examples of these interferers are simply devices operating on the same frequency bands as WiFi, e.g. BLE devices, or even WiFi devices out of range to cause contention. The activity of these devices is modeled by simply adhering to their own activation probability, where active devices contribute to the aggregated interference power at the receiver.

$$I = 10 \log_{10} \left(\sum_{i \notin \mathcal{N}_{\text{CSMA}}} A_i \cdot 10^{P_{\text{int}_i} - \text{PL}_{\text{int}_i} + X_{\sigma, \text{int}_i} / 10} \right) \quad (4.13)$$

Since CSMA-enabled devices defer transmission when another device is active, they do not contribute to the aggregated interference power at the receiver. Instead, interference power is entirely contributed by uncoordinated interferers and devices outside the contention radius.

CSMA Backoff Mechanism

The CSMA backoff mechanism determines, how WiFi devices resolve contention for the channel. If the transmitter finds the channel busy, $\text{CCA} = 0$, it must defer transmission and retry after a random delay. Specific implementation of the backoff mechanism and orchestration between devices in contention is defined in *IEEE 802.11-2020* [10]. Backoff events, as well as the randomized delay are modeled probabilistically as:

$$X \sim \text{Bernoulli}(N_{\text{CSMA}}, \alpha) \quad (4.14)$$

with N_{CSMA} being the number of devices within the contention radius R_{CSMA} and α being the probability of a device actively transmitting. From Equation (4.14), the probability of the channel being clear, i.e. $\text{CCA} = 1$, is given as a binomial pmf evaluated at $X = 0$:

$$P_{\text{clear}} = P(X = 0) = \binom{N_{\text{CSMA}}}{0} \alpha^0 (1 - \alpha)^{N_{\text{CSMA}}} \quad (4.15)$$

If at least one interferer is transmitting at the time of observation, the transmitter enters backoff and retries in a future time slot. This means the number of backoff events before a successful transmission follows a geometric distribution with P_{clear} as the probability of success:

$$N_B \sim \text{Geometric}(P_{\text{clear}}) \quad (4.16)$$

If a device enters backoff, it must select a random backoff counter before trying again. IEEE802.11-2020, defines the backoff counter as a random integer drawn from a uniform distribution over the current contention window [17]:

$$B_i \sim \text{Uniform}(0, CW_i) \quad (4.17)$$

where CW_i is the contention window size at the i -th backoff event. The contention window evolves exponentially after each backoff event as described by Equation (4.18), and illustrated by Figure 4.5.

$$CW_i = \min(CW_{\max}, CW_{\min} \cdot 2^i - 1) \quad (4.18)$$

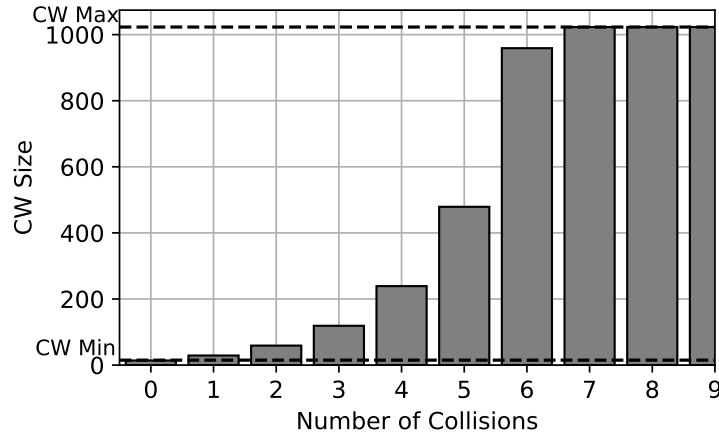


Figure 4.5: Example of exponential evolution of contention window CW . Inspired by [10].

The total contention time, accounting for multiple backoff events and the backoff counter is the sum of all backoff event delays:

$$T_B = \sum_{i=0}^{N_B-1} B_i \cdot T_{slot} = \sum_{i=0}^{N_B-1} \text{Uniform}(0, CW_i) \cdot T_{slot} \quad (4.19)$$

where $T_{slot} = 20 \mu s$ is the duration of a single WiFi time slot. The combined effects of interference power and CSMA contention delay, determine the achievable throughput and latency in the network. While interference directly impacts the SINR and lowers data rate, high contention leads to long channel access delays directly impacting the packet latency. The next section finally establishes the relationship between SINR, achievable latency and throughput, which will serve as the basis for dataset generation and training of a ML model.

4.3.3 SINR & Throughput Modeling

The final objective of the WiFi link-level simulator is to establish a mapping between SINR, congestion and achievable latency or throughput. With the models for environment generation, propagation and interference, the performance of a communication link can be evaluated by:

- Computing SINR at the receiver: Accounting for interference, noise and propagation loss.
- Determine Modulation and Coding Scheme (MCS): IEEE lookup table based on SINR thresholds.
- Calculate achievable throughput/latency: Incorporate MCS-based data rates, a fixed packet size and channel access delays from CSMA.

Computing SINR and MCS

With the interference and propagation models in place, SINR can be computed using the received power P_{rx} , the aggregate interference power I and noise floor N_0 :

$$\text{SINR} = 10 \log_{10} \left(\frac{P_{rx}}{I + N_0} \right) \quad (4.20)$$

where the P_{rx} is given by the received power subject to the propagation model, I is given by Equation (4.13) and $N_0 = B \cdot k \cdot T$. From the SINR, the throughput can be estimated using a given Modulation and Coding Scheme (MCS). IEEE 802.11ax-2020 has defined an MCS table, providing coding schemes, modulations and throughputs for given SINR thresholds [10].

In the simulator, the MCS index is determined as a function of SINR. Each MCS level corresponds to a different modulation type and coding rate, which directly impacts the achievable PHY throughput. The mapping between SINR, MCS, and PHY throughput is defined in Table 4.5.

SINR Threshold (dB)	MCS Index	PHY Throughput (Mbps)
< 2	N/A	0.0
≥ 2	0	6.5
≥ 5	1	13.0
≥ 9	2	19.5
≥ 11	3	26.0
≥ 15	4	39.0
≥ 18	5	52.0
≥ 20	6	58.5
≥ 25	7	65.0
≥ 29	8	78.0

Table 4.5: Mapping of SINR to Modulation and Coding Scheme (MCS) and corresponding PHY throughput [10].

Using Table 4.5, the MCS index is assigned based on the current SINR value. The simulator determines the index by:

$$\text{MCS} = \max \{k \in K_{\text{MCS}} | k \leq \text{SINR}\} \quad (4.21)$$

where K_{MCS} is the set of predefined SINR threshold values corresponding to MCS indices from Table 4.5. Note that the simulator directly uses the PHY throughput, meaning retransmissions and further overhead delays are not considered. Assumptions and model limitations are further discussed at the end of this section.

Computing Latency

As described in Section 4.3.2, a latency penalty is added for each contention event, due to a random backoff time. Successive contention events lead to a growing contention window, further increasing the latency penalty. The total time spent in backoff T_{CSMA} , the latency penalty, is given by Equation (4.19). From the throughput and time spent in CSMA, the total latency for transmission of a single frame is given as:

$$T_{\text{latency}} = T_{\text{CSMA}} + T_{tx} \quad (4.22)$$

where T_{CSMA} is the latency penalty added from contention and T_{tx} is the transmission time, depending on the throughput from the MCS table and the frame size, fixed to 400 kB in the simulator.

Model Assumptions and Limitations

While the throughput model provides an efficient estimate of achievable data rates by incorporating the IEEE 802.11ax MCS table [10]. It introduces several simplifications:

- No retransmissions: The model directly assigns the PHY throughput based on SINR without considering retransmissions. In reality the effective throughput is better estimated using $R_{\text{eff}} = R_{\text{PHY}}(1 - P_{\text{loss}})$, with an appropriate model for SINR induced packet loss.
- No MAC overhead: The model does not consider the delay added by transmission of ACK/NACK, which is neglected due to no retransmissions.
- Perfect MCS adaptation: The simulator operates on an idealized throughput selection, where the modulation and coding scheme perfectly adapt to the SINR.

4.3.4 Visualization of Simulator Outputs

To illustrate the output of the WiFi simulator, Figure 4.6 presents an example distribution of SINR values and corresponding latency for a single simulation environment.

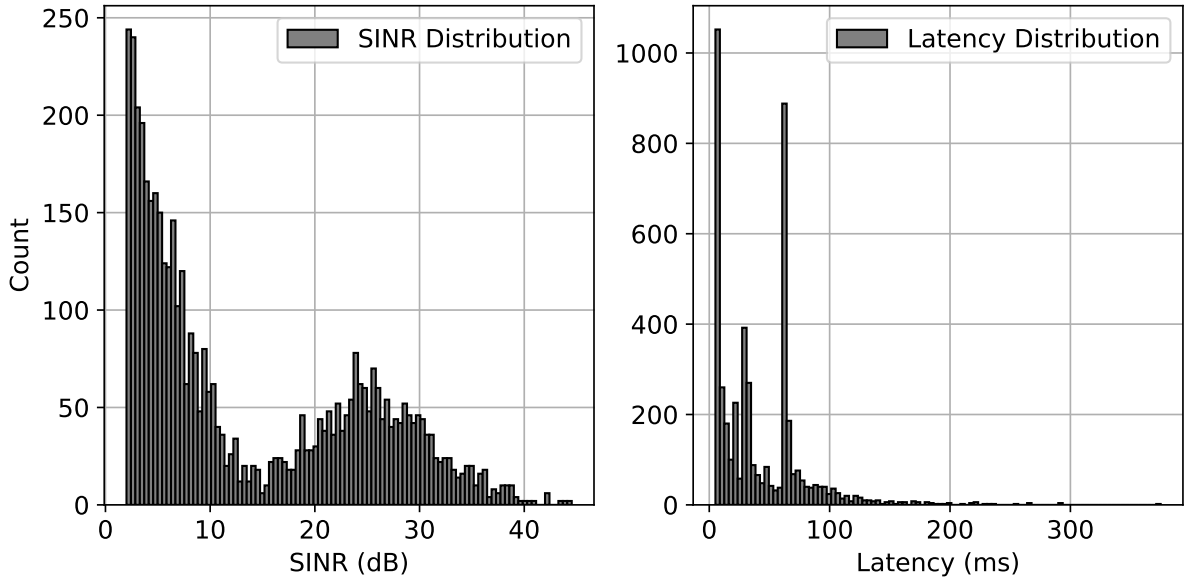


Figure 4.6: Histogram of SINR (left) and latency (right) for WiFi transmissions in a single generated simulation environment.

The SINR distribution captures the stochastic variability introduced by the propagation and interference models, particularly due to shadowing and distance-based path loss.

Both the SINR and latency distribution exhibits very distinct multi-modal behavior resembling a form of mixture model. For the SINR distribution this can be attributed to the stochastic variability introduced by the propagation and interference models, and for the latency distribution, the multi-modality can be attributed to two key phenomena:

- Exponential tail behavior: The CSMA backoff mechanism introduces geometrically distributed contention delays. As described in Section 4.3.2, the number of contention windows grows exponentially with the number of contention events, resulting in the long right-tail in the latency distribution.

- Discrete shifts in the latency: The PHY layer throughput, dictated by the SINR-to-MCS mapping in Table 4.5 introduces discrete changes in transmission time. Resulting in visible clusters in the latency histogram.

These effects combine to produce a latency distribution with potentially overlapping exponential tails and throughput-induced shifts. The multi-modal characteristics of the WiFi distributions motivate the use of a mixture model if the distribution is to be parametrically modeled.

4.4 BLE Simulator

Simulating the BLE link layer involves modeling adaptive frequency hopping, signal propagation, and the effects of noise and interference on packet delivery. The simulator makes the following assumptions:

- Perfect synchronization between central and peripheral devices.
- Advertising and discovery phases are assumed complete.
- Interference is modeled using a simple channel collision model.
- BLE uses uncoded PHY with on-air data rates of 1 Mbps.

Each of the 37 data channels are modeled as a simple collision channel with additive white Gaussian noise (AWGN). Transmission errors can occur either due to a collision with an interferer or due to decoding errors caused by low SNR.

Since BLE's uncoded PHY modes lack forward error correction (FEC), even a single bit error results in complete packet loss. BLE packets include a 3 byte, or 24-bit, CRC. The CRC check result is relayed back to the sender via an automatic repeat request (ARQ) mechanism [11]. If a packet is not acknowledged, it is retransmitted until it is successfully received or the connection's maximum retransmission limit is reached [11].

The total transmission time of a BLE frame accounts for all such retransmissions. Table 4.6 presents the transmission time per packet, including the duration of the payload transmission, and ACK/NACK handling, across different PHY configurations.

Description	1 Mbps		2 Mbps	
	Normal	DPLE	Normal	DPLE
Packet transmission time	0.296 ms	2.088 ms	0.148 ms	1.044 ms
Switch Radio from RX to TX and back	0.29 ms			
ACK/NACK transmission time	0.088 ms		0.044 ms	
Total transmission time per packet	0.674 ms	2.466 ms	0.482 ms	1.378 ms

Table 4.6: Overview of BLE packet transmission time across PHY modes [11].

Based on the retransmission behavior of BLE, the total transmission time for a BLE frame is modeled as:

$$t_{\text{frame}} = \sum_{i=1}^{N_{\text{packets}}} \sum_{j=1}^{N_{\text{dec}}(p_{\text{dec}})} (N_{\text{int}}(p_{\text{int}})) \cdot t_{\text{packet}}, \quad (4.23)$$

where t_{frame} is the total frame transmission time, t_{packet} is the total packet transmission time, $N_{\text{packets}} = \frac{D_{\text{BLE}}}{S_{\text{PDU}}}$ is the total number of packets, D_{BLE} is the Data size in BLE transmission, S_{PDU}

is the Protocol Data Unit (PDU) size., $N_{\text{int}}(p_{\text{int}})$ the number of retransmissions due to interference per BLE packet, $N_{\text{dec}}(p_{\text{dec}})$ the number of retransmissions due to decoding errors per BLE packet

The following section will in further detail expand on how the number of retransmissions are modeled.

4.4.1 Number of Retransmission due to Interference ($N_{\text{int}}(p_{\text{int}})$)

Interference in the BLE channel is modeled by a simple collision channel. Meaning any interference on a BLE channel results in a failed transmission. The probability of success for packet transmission is a function of the available channels in the BLE channel map (CHmap) and the number of available channels that experience interference during the BLE packet transmission. The probability of collision is

$$p_{\text{int}} = \binom{k}{0} p_{\text{ch}}^0 (1 - p_{\text{ch}})^k = (1 - p_{\text{ch}})^k \quad (4.24)$$

The probability of choosing a channel, p_{ch} is uniformly distributed over the available channels. Meaning the probability of choosing any specific channel is:

$$p_{\text{ch}} = \frac{1}{\text{NUC}}, \quad (4.25)$$

where NUC is number of available channels in the BLE channel Map and k is the number of channels with interference.

The number of channels in the BLE CHmap is a random draw of the distribution in Equation (4.26), and is constant throughout all sample in an environment. The distribution essentially models WLAN interference in the channels.

$$\text{NUC} = \text{U}(0, 3) \cdot 10 + 7 \quad (4.26)$$

The number of channels with interference, is drawn from a discrete uniform distribution with its boundaries defined in the simulation parameters.

The probability of a collision is used to model the number of retransmissions due to interference, $N_{\text{int}}(p_{\text{int}})$, using the geometric distribution. The probability of k number of retransmission is given in Equation (4.27).

$$P(X = k) = (1 - p_{\text{int}})^k \cdot p_{\text{int}} \quad (4.27)$$

where k is the number of failures and p_{int} is the probability of success.

4.4.2 Number of Retransmissions due to Decoding Errors ($N_{\text{dec}}(p_{\text{dec}})$)

The number of retransmissions due to decoding errors of a BLE packet is calculated using the geometric distribution, with the probability of decoding the Gaussian Frequency-Shift Keying (GFSK) signal of BLE. The decoding of the GFSK depends on the energy per bit to noise (E_b/N_o) of the signal. The relationship between E_b/N_o and PER are modeled through a simulation. The simulator is based on Matlab's Bluetooth® Toolbox Example, "Bluetooth LE Bit Error Rate

Simulation with AWGN" [18]. Using the `BLEWaveformGenerator` and `BLEIdealReciver`, perfect synchronization is assumed. To compare the two uncoded physical modes of BLE, LE1M and LE2M, E_b/N_o is converted to SNR [19]:

$$\text{SNR}_{\text{dB}} = 10 \cdot \log \left(\frac{E_b}{N_o} \right) + 10 \cdot \log \left(\frac{f_b}{B} \right)$$

where B is the channel bandwidth in hertz, and f_b is the symbol symbols per second rate. Figure 4.7 illustrates the expected PER of the different BLE configurations displayed in Table 4.6.

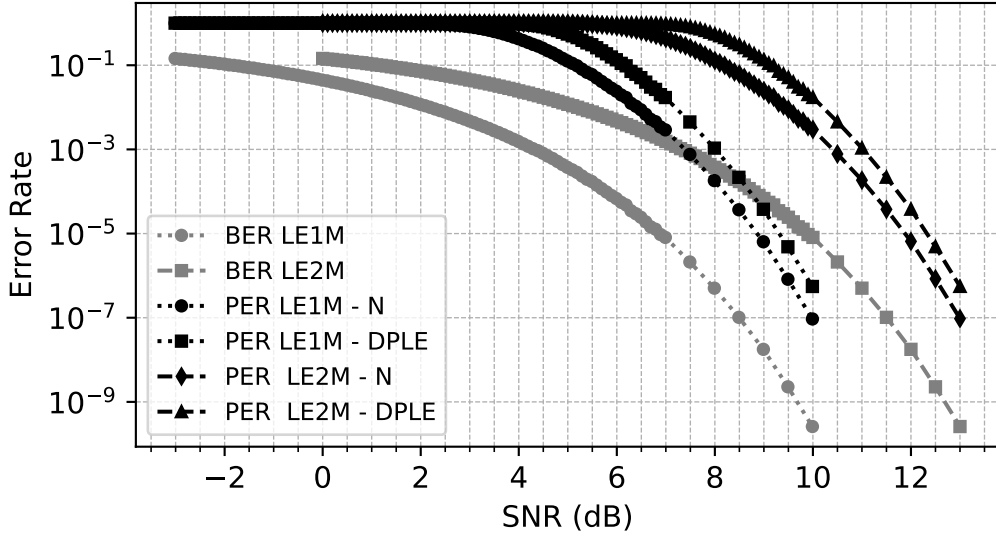


Figure 4.7: BER and PER for different SNR and BLE transmission modes.

The SNR is calculated as

$$\text{SNR}_{\text{dB}} = P_{tx,db} - PL_{db} - P_{\text{Noise},db} \quad (4.28)$$

where $P_{tx,db}$ is the transmitted power in dBm, PL_{db} is the Path loss in dB and $P_{\text{Noise},db}$ is the noise power for 2 MHz in dBm. The noise is modeled as Johnson–Nyquist noise at room temperature and a transmit power of +8 dBm is used, as it is the maximum transmit power of a nRF52840 BLE chips from Nordic semiconductors [14].

$$\text{SNR}_{\text{dB}} = 8 - PL - 10 \cdot \log_{10} \left(\frac{k_b T \Delta f}{1mW} \right) \quad (4.29)$$

The environment is outdoors, the path loss is therefore modeled as a Two-Ray Ground Reflection Model [16].

$$PL_{db} = 40 \log_{10}(d) - 20 \log_{10} (h_t^2 h_r^2) - G \quad (4.30)$$

where, d is the distance between the transmitter and receiver h_t and h_r is height of the transmitter and the receiver both placed at 1.5 m. G is the Antenna gain and is typically between -10 & 10, here we have chosen one of 5 dB [20]. To simplify calculation it is assumed that no stochastic process impact the SNR, as a compensation for this a link margin of 15 dB is used [20].

4.4.3 Runtime Optimization

The processes of calculating the latency of a video frame transmitted over BLE, t_{frame} , described by Equation (4.23) is repeated several times. As such an optimization of algorithm is in order. The basic algorithm is shown in Algorithm 1.

Algorithm 1 Computation of t_{frame}

Require: $D_{\text{BLE}}, S_{\text{PDU}}, p_{\text{int}}, p_{\text{dec}}, t_{\text{packet}}, \text{num_samples}$

Ensure: t_{frame}

```

1:  $t_{\text{frame}} \leftarrow 0$ 
2:  $N_{\text{BLE}} \leftarrow N_{\text{packets}}$ 
3: for  $i = 0$  to  $N_{\text{BLE}}$  do
4:   Sample  $N_{\text{dec}}(p_{\text{dec}}) \sim \text{Geom}(p_{\text{dec}})$ 
5:   for  $j = 0$  to  $N_{\text{dec}}(p_{\text{dec}})$  do
6:     Sample  $N_{\text{int}}(p_{\text{int}}) \sim \text{Geom}(p_{\text{int}})$ 
7:      $t_{\text{frame}} \leftarrow t_{\text{frame}} + (N_{\text{int}}(p_{\text{int}}) \cdot t_{\text{packet}})$ 
8:   end for
9: end for
    
```

The goal is to only sample from one combined distribution, instead of sampling multiple geometric distribution multiple times and taking the sum. We need to determine the distribution of:

$$N_{\text{frame}} = \sum_{i=1}^{N_{\text{dec}}} N_{\text{int}}, \quad (4.31)$$

where N_{frame} is the sum of a random number of geometric variables. A key property of the geometric distributions is that the sum of a number of i.i.d. geometric random variables is, itself a geometric variable with a success probability equal to the product of the two success probabilities. This is due to the memoryless property of the geometric distribution, meaning that the probability of success in the next time period T remains the same regardless of time passed.

This property ensures that repeating geometric trials within a geometric trial preserves the overall geometric structure. Therefore, instead of separately sampling from N_{int} and N_{dec} , we can directly sample from, $N_{\text{frame}} \sim \text{Geom}(p_{\text{int}} \cdot p_{\text{dec}})$. Which can be further reduced by exploiting the negative binomial distribution.

$$\begin{aligned}
 t_{\text{frame}} &= \sum_{i=1}^{N_{\text{packets}}} (N_{\text{frame}}(p_{\text{dec}} \cdot p_{\text{int}})) \cdot t_{\text{packet}} \\
 &= (N_{\text{packets}} + \text{NB}(N_{\text{packets}}, N_{\text{frame}}(p_{\text{dec}} \cdot p_{\text{int}}))) \cdot t_{\text{packet}}
 \end{aligned} \quad (4.32)$$

with:

$$\text{NB}(r, p) = \binom{k+r-1}{k} \cdot (1-p)^k p^r, \quad (4.33)$$

where r is the number of successes until the experiment is stopped, p is probability of success and k is the number of failures.

Figure 4.8 showcases an empirical validation of the optimized algorithm. The validation was performed using a Monte Carlo simulation with 100E6 trials of Algorithm 1.

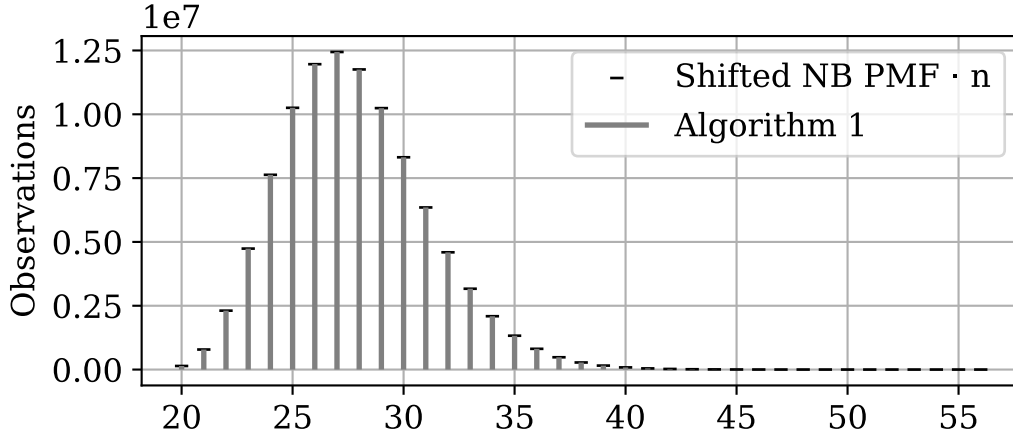


Figure 4.8: Empirical validation of simplification.

A test was conducted showing a improvement of the average draw time from a Negative Binomial distribution is over 10 times faster then doing Algorithm 1 in a JIT optimized python script.

4.4.4 Visualizing the Simulation Output

The BLE simulator takes in a position, a channel map, and number of interferers from the environment. From the position the SNR is calculated using the two-ray model, from which a decoding error probability is computed. Furthermore, the channel map and number of interferers is used to compute the probability of a collision. Multiplying the two probabilities gives us the effective probability p_{eff} of a packet being successfully received and decoded. Figure 4.9 showcases the relationship between the effective success probability and the number of required retransmissions.

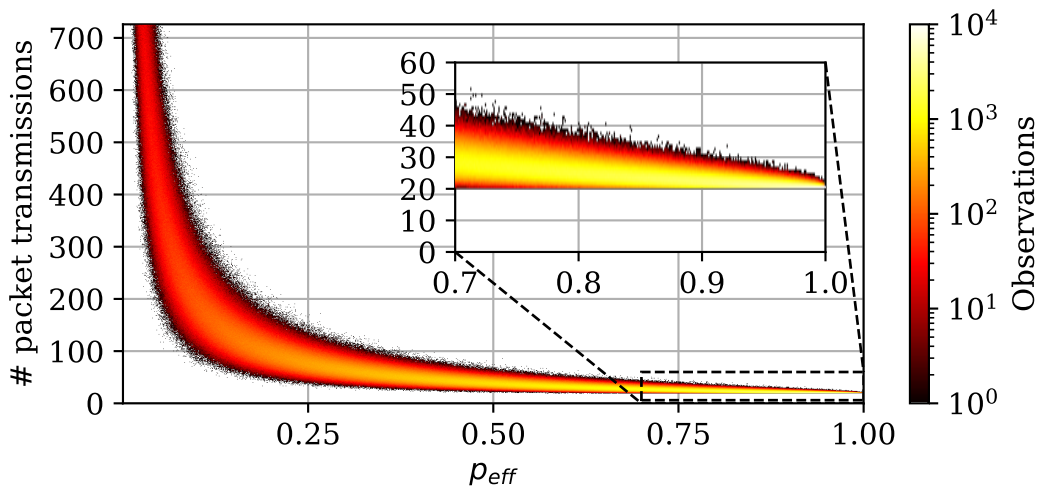


Figure 4.9: Heat map over observations of required amount of transmissions

Using equation Equation (4.32), we can calculate the transmission latency for a frame transmitted over the BLE interface.

$$t_{\text{frame}} = \sum_{i=1}^{N_{\text{packets}}} \sum_{j=1}^{N_{\text{dec}}(p_{\text{dec}})} (N_{\text{int}}(p_{\text{int}})) \cdot t_{\text{packet}} = N_P \cdot 2.466 \text{ ms} \quad (4.34)$$

where N_P the number of packet transmissions. Figure 4.10 showcases the latency distribution output from the BLE simulator from a single simulation environment.

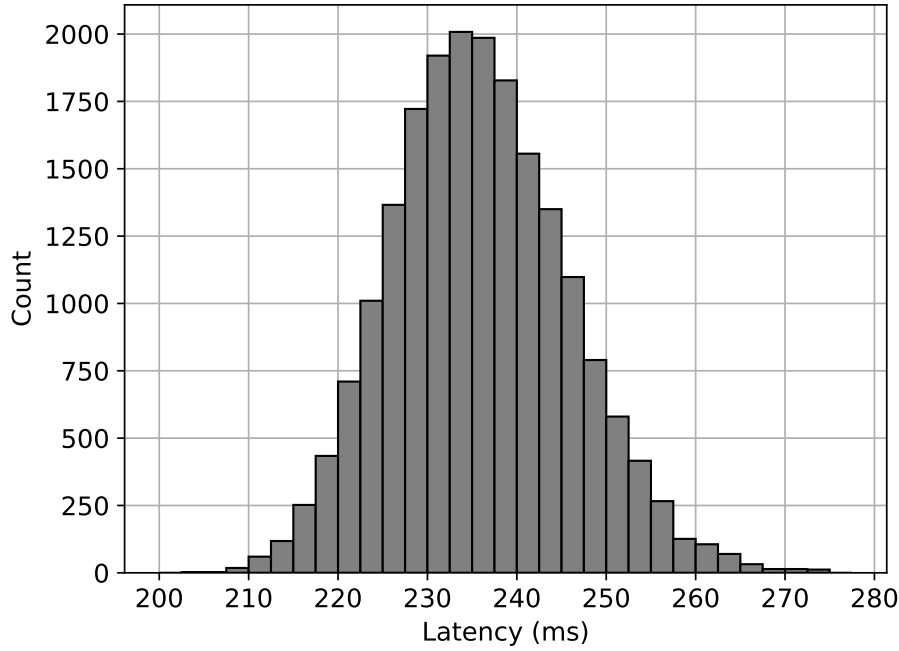


Figure 4.10: BLE latency distribution in an example simulation environment.

4.5 Generated Datasets

Throughout this thesis, a wide range of datasets have been produced. As part of our iterative design process, smaller and less complex datasets have been used to validate concepts and subsystems throughout the development. The final training and results presented in Chapter 5 and 6 are made on large and more complex datasets summarized by Table 4.7. Part of the added complexity stems from making the number of interferers a random variable, further magnified by letting the activation probability vary between interferers. To capture the stochastic nature of the WiFi and BLE simulators, sufficient fields must be generated and simulated for a large number of transmissions. This is especially important for the ML training dataset, since the latency predictor models must be exposed to rare propagation environments. On the other hand, the dataset for conformal prediction may contain fewer simulation environments, but more transmissions per environment, since calibration and evaluation must be performed from the same dataset.

CHAPTER 4. DATA GENERATION AND SIMULATION FRAMEWORK

Description	Key	ML Dataset	CP Dataset
Number of environment to generate	num_env	10E3	1E3
Number of simulations per environment	num_iterations	10E3	1E6
Simulation environment dimensions [m x m]	env_size	400 x 400	
WiFi Transmit power [dBm]	tx_power	20	
WiFi Channel	channel	6	
CSMA Thresholds power [dBm]	csma_threshold	-62	
Min / max number of WiFi interferers	num_WiFi_interferers	3 / 8	
WiFi frame size [B]	wifi_frame_size	400E3	
Min / max activation probability of WiFi interferers	random_activation_prob	0.3 / 0.8	
Size of BLE frame size in bytes	ble_frame_size	20E3	
BLE transmission mode	ble_transmission_mode	LE1M - DPLE	
Min / max number of BLE interferers	num_ble_interference	3 / 10	

Table 4.7: Simulation parameters, of the ML dataset and the CP dataset.

5 | Latency Predictor Models

To enable uncertainty-aware decision making in wireless environments, this chapter presents and compares several neural-network based machine learning models trained to predict the 95th latency quantile of WiFi and BLE. These models serve as the core of the uncertainty-aware latency predictor introduced in Chapter 3, it is trained using the synthetic datasets generated by the link-level simulator from Chapter 4.

We organize our models into two families based on how they treat the latency distribution. The first group, *Parametric distribution models*, aim to capture the entire latency distribution by predicting the parameters of a suitable statistical model allowing for post-hoc analysis. The other group, *quantile regression models*, are trained to directly estimate a specific latency quantile, without assuming a particular latency distribution. This trade-off between flexibility and imposed assumptions forms the basis for the model evaluation in this chapter.

Across both model families, five latency predictors are trained and evaluated:

- Mean-Variance Parametric Regressor (MV-Param): Predicts the full latency distribution as a single Gaussian.
- GMM Parametric Regressor (GMM-Param): Predicts the full latency distribution as a gaussian mixture model.
- Mean-Variance Quantile Regressor (MV-QR): Predicts the $(1 - \alpha)$ -th quantile of the latency distribution from a mean and variance feature input.
- GMM Quantile Regressor (GMM-QR, MSE and Pinball): Predicts the $(1 - \alpha)$ -th quantile of the latency distribution from a Gaussian mixture model feature input.

To contextualize these models, we first formalize the input and output vectors and clarify the data required for each model. We then describe the preprocessing pipeline, transforming the raw simulation outputs into suitable features for training the ML models, this includes batching and statistical fitting. After the preprocessing step, we detail training objectives, loss functions and architectural considerations for each model. Followed by, a comparative evaluation of the predictive performance of each model.

5.1 Model Families Overview

The latency prediction models can be grouped into two distinct families based on how they represent the underlying latency distribution: Parametric distribution models and quantile regression models. This distinction introduces a trade-off between statistical flexibility, distributional assumptions and predictive accuracy.

5.1.1 Parametric Models

Parametric models operate on a foundational assumption of the latency distribution, i.e. Gaussian or Gaussian Mixture Model. These models are trained to predict the parameters of the assumed distribution, from which any $(1 - \alpha)$ quantile can be inferred post-hoc. As such, the underlying distributional assumption dictates the architecture of the neural network, since the size of the parametric representation may vary i.e. mean-variance or Gaussian components.

- The *MV-Param* model assumes the latency follows a Gaussian distribution and directly predicts the distribution's mean and variance ($\hat{\mu}, \hat{\sigma}$). The target quantile is analytically derived from the inverse Gaussian CDF.
- The *GMM-Param* model relaxes the distributional assumption by modeling the latency as a weighted sum of gaussians (GMM), allowing for a multi-modal distribution, similar to what was observed in the WiFi simulator output in Figure 4.6. It predicts the parameters of each gaussian component ($\hat{\mu}, \hat{\sigma}, \hat{w}$)

This family of models provides access to the full latency distribution, allowing for post-hoc statistical analysis such as quantile computation or uncertainty modeling. However, the performance may heavily degrade if the true latency distribution deviates from the underlying assumption. Figure 5.1 illustrates the architecture for parametric models, where the neural network outputs a set of distribution parameters describing the full probabilistic distribution.

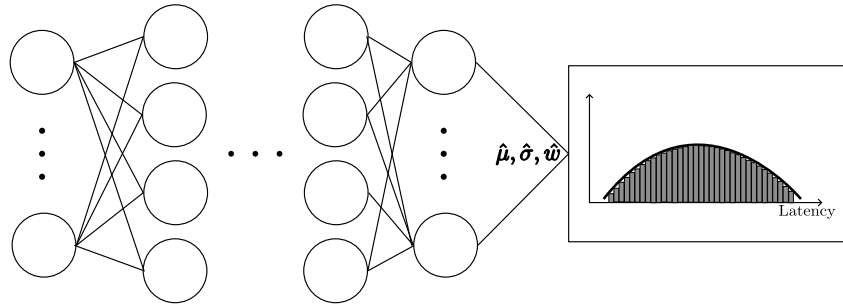


Figure 5.1: Visualization of parametric model, predicting a full probabilistic distribution.

5.1.2 Quantile Regression Models

In contrast to the parametric models, quantile regressors are trained to directly predict a target latency quantile, without modeling the full distribution. These models operate on the raw latency samples and therefore make no assumption on the underlying distribution, making them highly adaptable to complex latency distributions.

- The *MV-QR* model, predicts the $(1 - \alpha)$ -th quantile ($\hat{q}_{1-\alpha}$), using a compact statistical representation of the propagation environment, SINR mean and variance. This model is trained with the traditional quantile loss, also known as the pinball loss function, making it learn the 95th quantile directly.
- The *GMM-QR (MSE)* model, uses a more descriptive statistical representation of the environment (GMM SINR representation) as its input and predicts the $(1 - \alpha)$ -th latency quantile. This model is trained with empirical 95th quantiles as targets using a mean-squared-error loss function rather than the traditional pinball loss.
- The *GMM-QR (Pinball)* model is identical to the MSE variant, but was trained with the traditional pinball loss, rather than precomputed 95th quantile targets.

This model family is particularly well-suited for cases where the latency distribution is skewed or exhibits heavy tails, such as the latency distribution observed in Figure 4.6. This comes at the cost of only providing a point prediction of the target quantile, rather than a full probabilistic distribution., meaning a separate QR model must be trained for alternative quantile targets. Figure 5.2 illustrates the architecture for quantile regression models, where the model directly predicts the target quantile $\hat{q}_{1-\alpha}$.

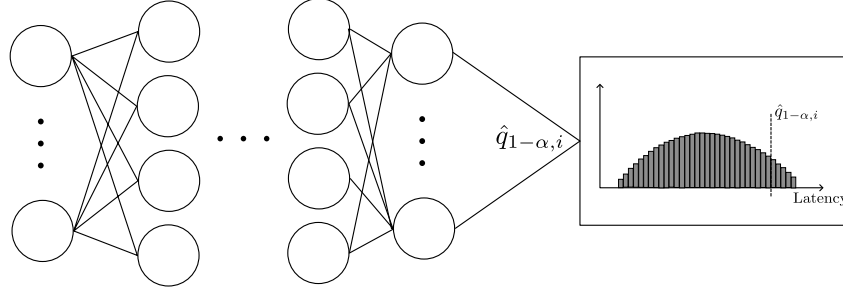


Figure 5.2: Visualization of quantile regression model, directly predicting the $(1 - \alpha)$ -th quantile for the interface i .

5.1.3 Inputs/Outputs

Each model outputs predictions for both WiFi and BLE interfaces. The input representations are defined per interface, with the possibility to use different statistics, i.e. GMM parameters for WiFi SINR and mean-variance statistics for BLE SNR. Similarly for the output latency distribution the following notation is defined:

- Direct latency quantiles $\hat{q}_{0.95,i}$ for QR models
- Parametric representations of latency distributions $(\hat{\mu}_i, \hat{\sigma}_i^2)$ or $(\hat{\mu}_i, \hat{\sigma}_i^2, \mathbf{w}_i)$, for parametric models

where the subscript $i \in \{\text{WiFi}, \text{BLE}\}$ denotes the communication interface and boldface symbols denote vector-valued parameters. Following this notation, input and output specifications for each model is summarized by Table 5.1. Note that all models use metadata describing the RF environment as supplementary inputs to the parametric representation, the metadata is summarized by Table 5.2.

Model	Input Representation	Output	Output Type
MV-Param	μ_i, σ_i^2	$\hat{\mu}_i, \hat{\sigma}_i^2$	Gaussian distribution
GMM-Param	$\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2, \mathbf{w}_i$	$\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\sigma}}_i^2, \hat{\mathbf{w}}_i$	GMM distribution
MV-QR	μ_i, σ_i^2	$\hat{q}_{0.95,i}$	Quantile point prediction
GMM-QR (MSE)	$\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2, \mathbf{w}_i$	$\hat{q}_{0.95,i}$	Quantile point prediction
GMM-QR (Pinball)	$\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2, \mathbf{w}_i$	$\hat{q}_{0.95,i}$	Quantile point prediction

Table 5.1: Model input and output specifications. Subscript $i \in \{\text{WiFi}, \text{BLE}\}$ denotes the interface. Bold symbols indicate vector-valued GMM parameters.

Description	Key
Number of used BLE channels	CH_Map
Number of BLE interferers	BLE_interferers
Number of WiFi interferers in CSMA	WiFi_beacons_CSMA

Table 5.2: Supplementary metadata describing the RF environment.

5.2 Preprocessing

To prepare the data for training, the raw simulation outputs must be transformed into statistical representations as specified by Table 5.1. This involves aggregating individual latency and SINR samples into batches, unto which appropriate Gaussian and GMMs distributions are fitted. These fitted distributions form the input features and targets used by the models throughout this chapter.

5.2.1 Batching Strategy

Batching is required to convert raw field-level measurements into structured statistical features. For this a few batching strategy were considered:

- Sliding window, allowing for overlaps between batches. This effectively reduces the sampling time allowing for more frequent predictions, at the cost of redundant or duplicate data.
- Non-overlapping batches, simplifies the data splitting by forcing every batch to be independent and unique.

We opted for non-overlapping batches to avoid contamination between training and test sets. While sliding windows do shorten the time required to collect sufficient samples for a prediction and may improve the update rate of the actor, it introduces sample overlap which complicates the evaluation.

Batches can be defined in three ways, each with their own benefits and drawbacks; by a fixed number of samples, a fixed memory budget or by a fixed time duration. For example, fixed-memory ensures safe inference on constrained hardware, while fixed-sample ensures uniform batch quality. However, this comes at the cost of a variable inference frequency, which is undesired in feedback-driven systems [21]. Therefore, we opt for a fixed-duration batching strategy, meaning each batch spans a constant time window `Lat_threshold` seconds. This facilitates a fixed inference frequency, consistent with classical control system requirements. It could however, be beneficial to use a combination of the three in a real implementation.

5.2.2 Batching Implementation

The batching logic is shown via a flowchart by Figure 5.3. For each environment, latency and SINR values are split into batches spanning a fixed time window. Batches with fewer than two valid latency observations or with missing values are discarded. In practice, such sparse batches would indicate a significant latency and therefore an unusable interface.

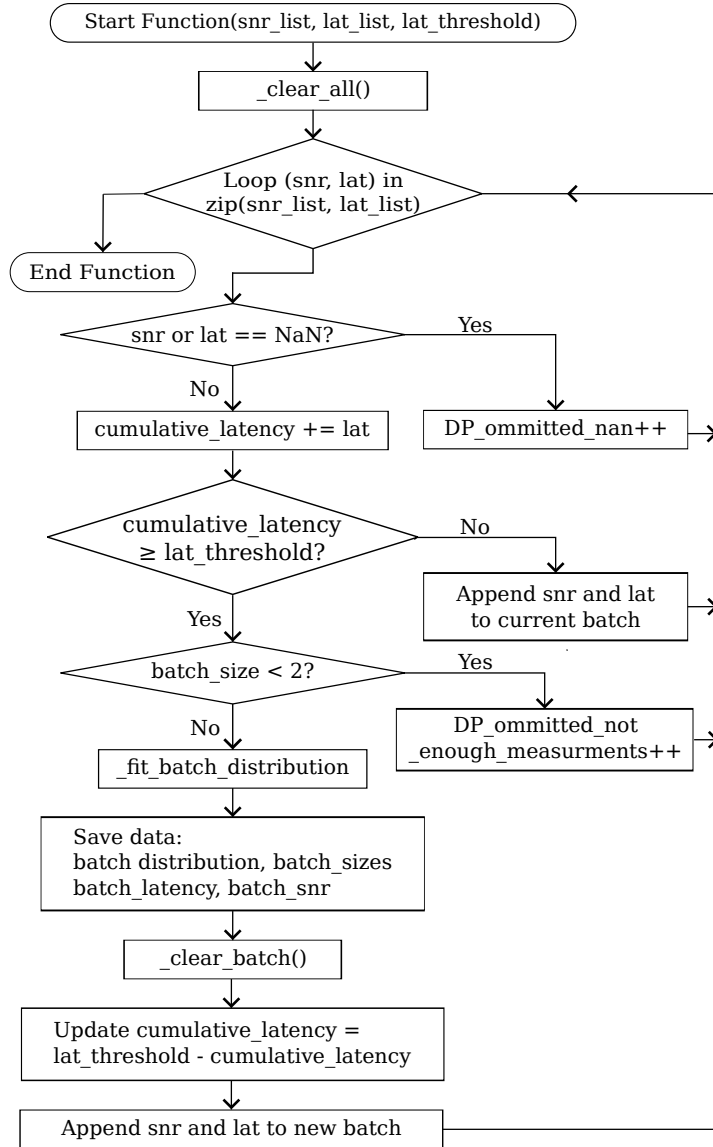


Figure 5.3: Flowchart of the batching procedure.

Each valid batch is processed to extract the mean and variance of both the latency and SINR values, as required by models relying on Gaussian input features. Additionally, for the models relying on a GMM input, we fit an GMMs of up to 8 components the SINR and latency samples using the `GaussianMixture` class from scikit-learn.

The number of Gaussian components is chosen based on the number of interference devices hidden to the transmitter but visible to the receiver. The intuition behind this, is that devices only visible to the receiver, are sufficiently far from the transmitter to not influence its CSMA, but still cause a degradation in SINR at the receiver. This influences the choice of MCS index, which is the primary cause of multi-modality in the latency distribution. In order to keep the dimension of the input to the latency predictor models consistent, we zero-pad the unused components ensure a GMM parameter length of 8. Note that GMMs were only fitted to WiFi distributions, since no multi-modality was observed for BLE. For all purposes, the input to the GMM models can be thought of as an 8-component GMM for WiFi and a single component for BLE.

The fitting process utilizes the Expectation-Maximization (EM) algorithm, iterating up to 100 steps or until a convergence tolerance of 10E-3 is reached. To ensure a valid fit, environments with fewer batches than GMM components are discarded. The EM algorithm alternates between the E-step, where the log-likelihood is computed as:

$$\begin{aligned} Q(\theta; \theta_{\text{old}}) &= \mathbb{E}[l(\theta; \mathcal{X}, \mathcal{Y}) \mid \mathcal{X}, \theta_{\text{old}}] \\ &= \mathbb{E}[\ln(p(\mathcal{X}|\theta)) \mid \mathcal{X}, \theta_{\text{old}}] \\ &= \int l(\theta; \mathcal{X}, y) p(y \mid \mathcal{X}, \theta_{\text{old}}) dy, \end{aligned} \quad (5.1)$$

where, $p(y|\mathcal{X}, \theta_{\text{old}})$ is the conditional density of \mathcal{Y} given the observed data, \mathcal{X} and assuming $\theta = \theta_{\text{old}}$ [22]. The other step, is the M-step, in which the expected value is maximized with respect to the new parameters θ_{new} [22]:

$$\theta_{\text{new}} = \max_{\theta} Q(\theta; \theta_{\text{old}}). \quad (5.2)$$

These batch-level statistics (μ, σ^2) or (μ, σ^2, w) , form the structured dataset for all subsequent model training and evaluation.

5.2.3 Prediction Frequency

The batch duration, `Lat_threshold` directly determines, the rate at which the system can perform a prediction and make a decision on the interface. A short batch duration, may give a more responsive system with frequent predictions and more dynamic switching of the used interface, but the batches may not be sufficiently large to facilitate accurate predictions. To balance the prediction quality and system responsiveness, we evaluate the relationship between batch durations and prediction quality using the Kullback-Leibler (KL) divergence.

For each batch duration `Lat_threshold`, we compute the resulting KL divergence between the full latency distribution, fitted on all latency observations \mathcal{L}_f in an environment, and the batch-level distribution, fitted on the subset of observations $\mathcal{L}_f^{\text{sub}}$. This is done for both of the parametric models meaning both Gaussian (μ, σ^2) and GMM (μ, σ^2, w) representations. The results are shown by Figure 5.4

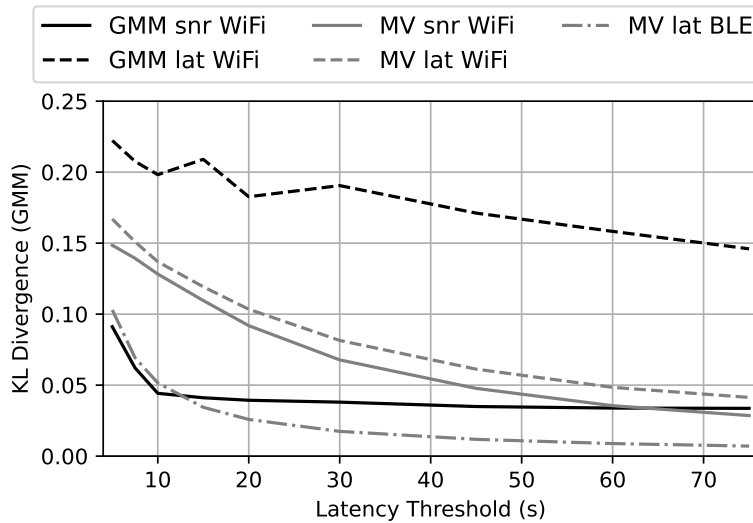


Figure 5.4: Average KL divergences between fitted model on batch data compared to fitted model on field data at different latency thresholds and distributions

Based on Figure 5.4, a batch duration of 20 seconds for both BLE and WiFi has been chosen, balancing the model accuracy and prediction frequency.

5.3 Parametric Models: Training & Evaluation

The parametric models from Section 5.1 are trained to predict the full latency distribution for both WiFi and BLE. These models impose a distributional assumption on the latency, in this case either a Gaussian or a GMM distribution. The models are trained to minimize the divergence between the predicted and empirical latency distributions. By learning the parameters of the assumed distribution, these models provide access to post-hoc statistical analysis, such as, computing the 95th quantile latency $\hat{q}_{0.95,i}$ for each interface $i \in \{\text{WiFi}, \text{BLE}\}$, which is obtained analytically for the Gaussian model and numerically, via. sampling for the GMM.

5.3.1 Loss Function: Kullback Leibler Divergence

To train the parametric models, we minimize the KL divergence between the empirical latency distribution $p(x)$ and the predicted latency distribution $q(x)$ [23]. The KL divergence is a measure of how one probability distribution diverges from a reference distribution, its defined as:

$$\text{KL}(p||q) = \int p(x) \log \left(\frac{p(x)}{q(x)} \right) dx \quad (5.3)$$

This expression measures how much information is lost when the model's prediction $q(x)$ is used to approximate the empirical distribution $p(x)$, in the form of relative entropy [23]. In practice $p(x)$ is estimated from a fitted Gaussian or GMM, as described in Section 5.2.

For the MV-Param model, which assumes a Gaussian latency distribution, the KL divergence has a closed-form solution:

$$\text{KL}(\mathcal{N}_p||\mathcal{N}_q) = \log \left(\frac{\hat{\sigma}}{\sigma} \right) + \frac{\sigma^2 + (\mu - \hat{\sigma})^2}{2\hat{\sigma}^2} - \frac{1}{2}, \quad (5.4)$$

where (μ, σ^2) denote the empirical mean and variance, and $(\hat{\mu}, \hat{\sigma}^2)$ are predicted by the MV-Param model. Figure 5.5 visualizes this divergence, plotting the pointwise KL divergence, the integrand from Equation (5.3), between two Gaussian distributions.

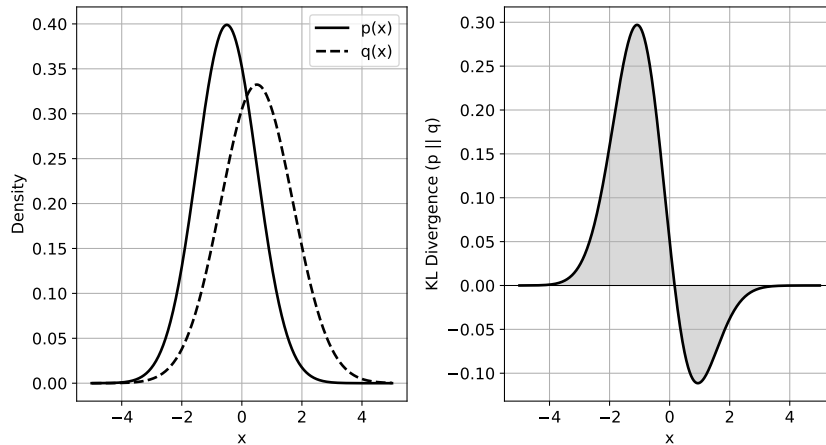


Figure 5.5: Visualization of the KL divergence between two Gaussian distributions $p(x)$ and $q(x)$ (left plot). Plots the pointwise divergence with the total divergence being the shaded area (right plot).

For the GMM-Param model, where both the empirical and predicted distributions are Gaussian mixtures, the KL divergence no longer has a closed-form solution [24]. Instead, various approximation methods are proposed, utilizing the KL divergence between individual Gaussian components, or simply approximating the GMM by a single Gaussian [24]. For training the GMM-Param model, we opt for a variational approximation of the KL divergence:

$$\text{KL}(P||Q) \approx \frac{1}{B} \sum_{b=1}^B \sum_{i=1}^K w_{p,i}^{(b)} \log \left(\frac{\sum_{i'=1}^K w_{p,i'}^{(b)} e^{-\text{KL}(p_i^{(b)}||p_{i'}^{(b)})}}{\sum_{j=1}^K w_{q,j}^{(b)} e^{-\text{KL}(p_i^{(b)}||q_j^{(b)})}} \right), \quad (5.5)$$

where b indexes batches, i, j index Gaussian components, $\text{KL}(p_i^{(b)}||q_j^{(b)})$ is the closed-form KL divergence between Gaussians. The numerator serves as the intra-mixture similarity of $p(x)$, serving as a normalization or reference distribution, and the denominator is the inter-mixture similarity between $p(x)$ and $q(x)$, which quantifies how well the predicted mixture $q(x)$ approximates the true mixture $p(x)$ [24].

5.3.2 Hyperparameter Selection and Training Setup

The hyperparameters for both parametric models are selected through a hyperparameter sweep conducted using the Weights and Biases (WandB) platform. We have configured the sweep to iteratively explore different parameter values for: hidden layers, hidden dimensions, learning rate and batch size. A sweep is conducted for both the MV-Param and GMM-Param model and the best-performing model is selected for evaluation. Figure 5.6 visualizes the parameter combinations and their resulting validation KL divergence loss, while Figure 5.7 shows the importance of each hyperparameter with respect to the validation loss. These figures show the training process for the GMM-Param model, similar sweep results are presented for the MV-Param model in Appendix C.

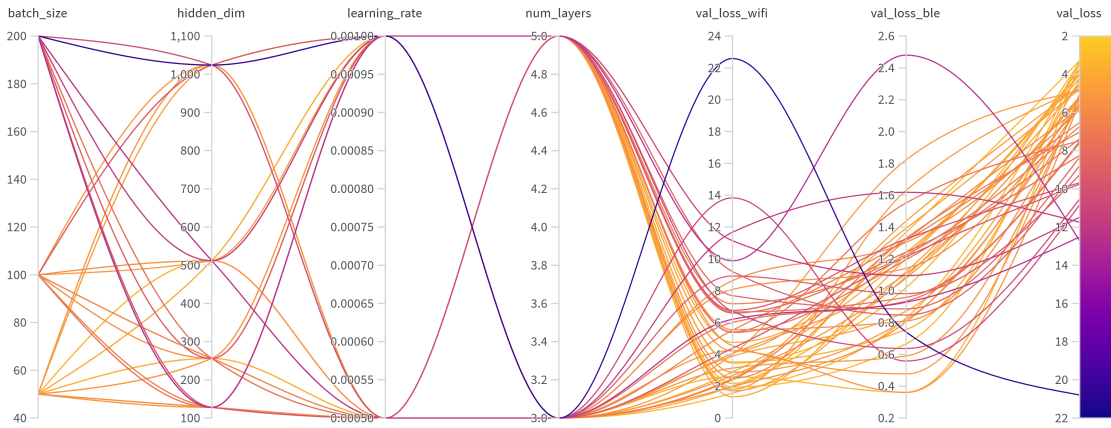


Figure 5.6: Parallel coordinates plot of hyperparameter sweep for the MV-Param model. Visualizes parameter combinations and the resulting KL divergence loss.

CHAPTER 5. LATENCY PREDICTOR MODELS

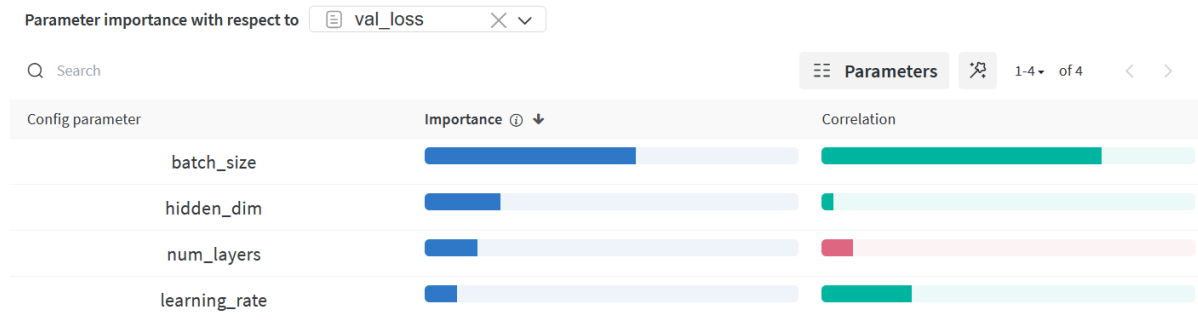


Figure 5.7: Visualizes the parameter correlation with respect to validation KL divergence loss for the MV-Param model.

The final hyperparameters, chosen from the parameter sweeps are summarized by Table 5.3 and 5.4. These tables list the chosen hyperparameters for the best-performing model of each type. The MV-Param model uses a compact input representation and therefore a relatively shallow network, while the GMM-Param model requires a deeper network for its more complex input representation.

Parameter	Details
Input features	μ_i, σ_i^2 for both interfaces (4 total)
Hidden layers	3 densely connected layers
Hidden layer dimension	1024 nodes
Activation function	ReLU
Output layer	$\hat{\mu}_i, \hat{\sigma}_i^2$ (4 total)
Output activation	Softplus for variances
Loss function	Gaussian KL divergence (Equation (5.4))

Table 5.3: Hyperparameters of the best-performing MV-Param model.

Parameter	Details
Input features	$\mu_i, \sigma_i^2, \mathbf{w}_i$ for both interfaces (28 total)
Hidden layers	5 densely connected layers
Hidden layer dimension	1024 nodes
Activation function	ReLU
Output layer	$\hat{\mu}_i, \hat{\sigma}_i^2, \hat{\mathbf{w}}_i$ (28 total)
Output activation	Softplus for variances, softmax for weights
Loss function	Variational KL divergence (Equation (5.5))

Table 5.4: Hyperparameters of the best-performing GMM-Param model.

5.3.3 Model Evaluation

The parametric models are evaluated based on how well the predicted quantiles match the empirical latency quantiles observed in the test data. Since these models predict full latency distributions, any quantile can be extracted post-hoc and evaluated accordingly.

For the MV-Param model, which assumes a Gaussian distribution, the 95th quantile is computed analytically by using the inverse Gaussian CDF:

$$F^{-1}(0.95) = \hat{\mu}_i + \hat{\sigma}_i \cdot \Phi^{-1}(0.95) = \hat{\mu}_i + \hat{\sigma}_i \cdot z_{0.95} \quad (5.6)$$

where $\Phi^{-1}(p)$ is the quantile function of a standard Gaussian distribution and $z_{0.95} \approx 1.65$ is its numerical approximation. Since the quantile function does not have a closed-form solution, numerical approximations are used, typically found from a Z-table [25].

For the GMM-Param model, no closed-form solution exists for the quantile function of a mixture distribution. Although the quantile is theoretically defined from the cumulative density of the mixture, standard tables do not exist. Therefore, we approximate the 95th percentile via Monte Carlo sampling, by drawing 10,000 samples from the predicted GMM and computing the empirical quantile of the sample set.

To visualize how well the MV-Param model captures the latency distribution, Figure 5.8 shows the prediction for a randomly selected test field, Field 5013. The predicted and target distributions are plotted alongside the empirical histogram, with true and predicted 95th quantiles indicated. For BLE, the unimodal Gaussian assumption fits the empirical distribution well. However, for WiFi, the distribution is broader and asymmetric, making it harder to represent with a single Gaussian. Note that the model is trained to predict the target distribution, as defined in Section 5.2, meaning even accurate predictions may not match the empirical quantile if the underlying distributional assumption is a poor fit.

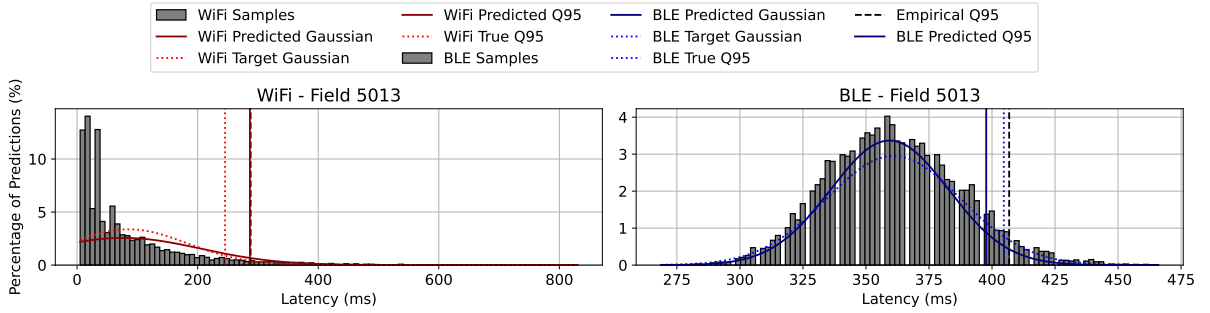


Figure 5.8: Visualization of randomly selected prediction from the MV-Param model. Showcases the empirical, target and predicted distribution with highlighted 95th quantiles.

To quantify the predictive performance across the entire test set, Figure 5.9 shows the distribution of prediction errors for the 95th quantile. BLE predictions are tightly clustered around zero, while WiFi prediction errors are generally larger and more spread due to the mismatch of the Gaussian assumption. Similar evaluation results for the GMM-Param model are presented in Appendix C.

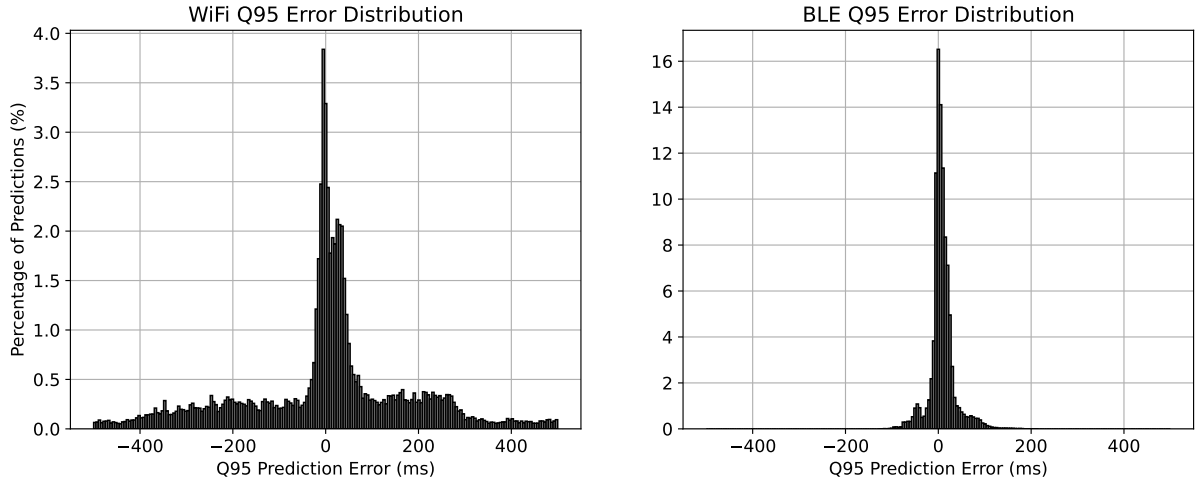


Figure 5.9: 95th quantile prediction error distribution for the MV-Param model.

5.4 QR Models: Training and Evaluation

The quantile regression models from Section 5.1 are trained to directly predict the 95th latency quantile of each interface without assuming any underlying distribution. This makes them more flexible than the parametric models, especially in the presence of skewed, heavy-tailed or multimodal distributions, but limits their output to a single point prediction per interface, rather than the full distribution.

5.4.1 Loss Functions: MSE and Pinball Loss

The QR models are trained using either the pinball loss or MSE.

- MSE-based models, e.g. GMM-QR MSE, use precomputed empirical 95th latency quantiles as the regression target. Meaning the MSE model does not operate on the raw latency samples, but train towards an extracted quantile target from the environment’s latency distribution.
- Pinball loss-based models, MV-QR and GMM-QR Pinball, do not require precomputed quantiles and instead learn to minimize an asymmetric loss function, guiding its predictions towards the 95th quantile.

The pinball loss function is defined as:

$$\mathcal{L}_\tau(y_i, \hat{y}_i) = \begin{cases} \tau(y_i - \hat{y}_i) & \text{if } y_i \geq \hat{y}_i \\ (1 - \tau)(\hat{y}_i - y_i) & \text{otherwise} \end{cases}, \quad (5.7)$$

where y_i and \hat{y}_i are the true and predicted latencies respectively and τ is the target quantile. This means the QR models are making a point prediction on the latency given their respective inputs, and the loss function is guiding the prediction towards the target quantile via asymmetric loss values for under- and overpredicting.

Figure 5.10 visualizes a randomly selected WiFi latency distribution, and the pinball loss function with $\tau = 0.95$. The pinball loss reaches its minimum when the predicted latency \hat{y}_i matches the 95-th quantile as shown by Figure 5.10.

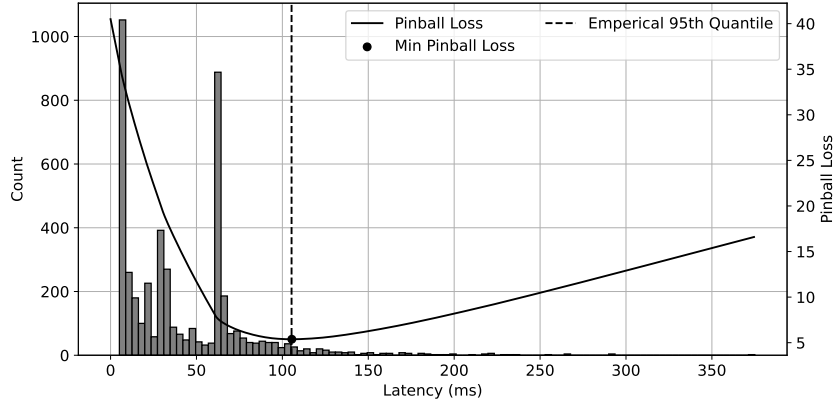


Figure 5.10: Visualization of WiFi latency distribution with pinball loss function overlay and highlighted minimum loss for a randomly selected field.

To further evaluate the pinball loss function, the deviation between loss function minimum and empirical 95th latency quantile, has been computed for the entire test set of 10E3 fields, the deviation is shown by Figure 5.11.

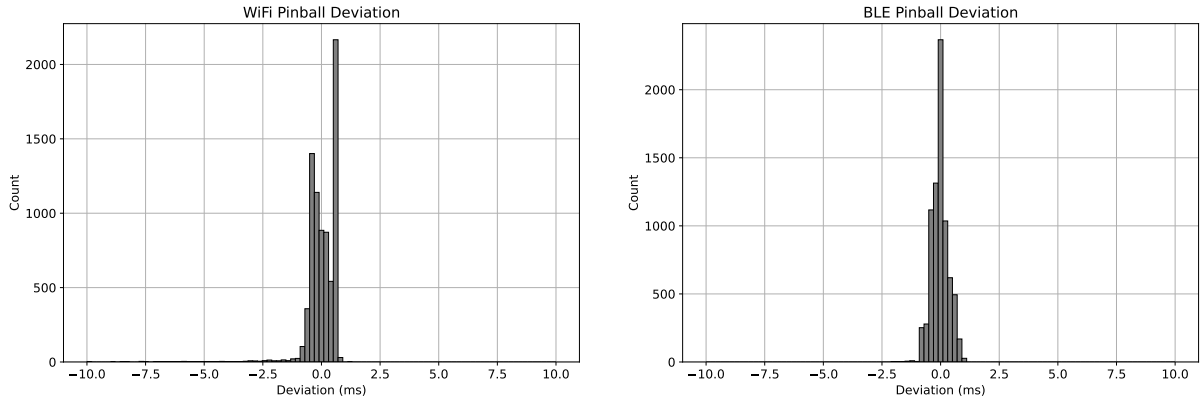


Figure 5.11: Deviation between pinball loss minimum ($\tau = 0.95$) and empirical 95th quantile for WiFi (left) and BLE (right).

5.4.2 Hyperparameter Selection and Training Setup

All QR models use a densely connected feedforward neural network. Similarly to the parametric models, hyperparameter sweeps are conducted for each QR model using WandB. The hyperparameters for the best-performing model of each type is summarized by Table 5.5, 5.6 and 5.7, these models are selected for further evaluation.

Parameter	Details
Input features	μ_i, σ_i^2 for both interfaces (4 total)
Hidden layers	5 densely connected layers
Hidden layer dimension	1024 nodes
Activation function	ReLU
Output layer	$\hat{q}_{0.95,i}$ for both interfaces (2 total)
Output activation	Linear activations
Loss function	Pinball Loss (Equation (5.7))

Table 5.5: Hyperparameters of the best-performing MV-QR model.

Parameter	Details
Input features	$\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2, \mathbf{w}_i$ for both interfaces (28 total)
Hidden layers	5 densely connected layers
Hidden layer dimension	512 nodes
Activation function	ReLU
Output layer	$\hat{q}_{0.95,i}$ for both interfaces (2 total)
Output activation	Linear activations
Loss function	Mean squared error

Table 5.6: Hyperparameters of the best-performing GMM-QR MSE model.

Parameter	Details
Input features	$\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2, \mathbf{w}_i$ for both interfaces (28 total)
Hidden layers	5 densely connected layers
Hidden layer dimension	1024 nodes
Activation function	ReLU
Output layer	$\hat{q}_{0.95,i}$ for both interfaces (2 total)
Output activation	Linear activations
Loss function	Pinball Loss (Equation (5.7))

Table 5.7: Hyperparameters of the best-performing GMM-QR Pinball model.

5.4.3 Model Evaluation

The QR models are evaluated by comparing their predicted 95th latency quantile $\hat{q}_{0.95,i}$ to the empirical quantile computed from the raw latency observations in the test data. Since the QR models directly output a point prediction for the quantile, no further post-processing is required. Figure 5.12 illustrates an example prediction from the GMM-QR Pinball model, showing both WiFi and BLE latency distributions alongside the predicted 95th quantiles. Like in the parametric case, BLE exhibits a unimodal latency distribution, for which the prediction aligns well with the empirical quantile. For WiFi, although the distribution exhibits drastically heavier tails and more skewed distributions, the quantile predictions remain near the targets, demonstrating robustness in the presence of more complex latency distributions as opposed to the parametric models.

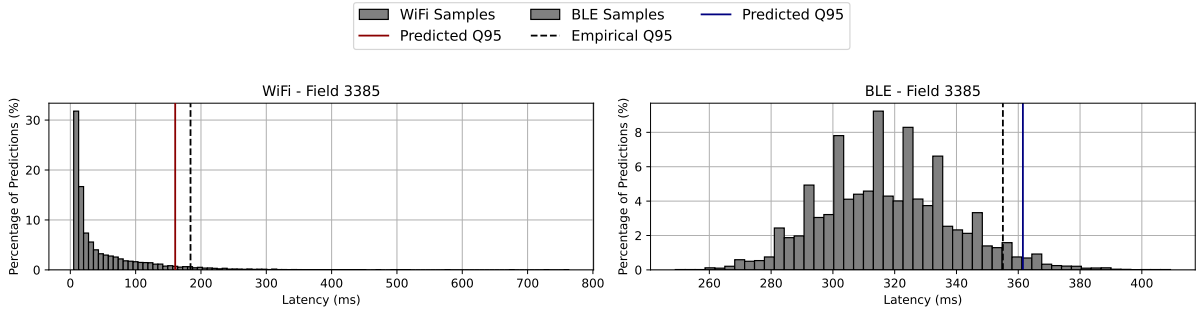


Figure 5.12: Visualization of randomly selected prediction from the GMM-QR Pinball model. Showcases the empirical latency distribution with the predicted and empirical 95th quantile highlighted for WiFi and BLE.

Similarly to the parametric models, the QR models are evaluated across the full test set, Figure 5.13 showcases the error distribution for the GMM-QR Pinball model and corresponding results for the other QR models are deferred to Appendix C. From Figure 5.13, the BLE predictions are tightly located around zero, reflecting consistent accurate quantile predictions, albeit biased towards overpredicting. WiFi predictions also show accurate predictions, but exhibits a broader error distribution, consistent with the more complex latency distribution. Notable for all QR models is a bias towards overpredicting, which may be attributed to the asymmetric loss of the pinball loss function as shown by Figure 5.10.

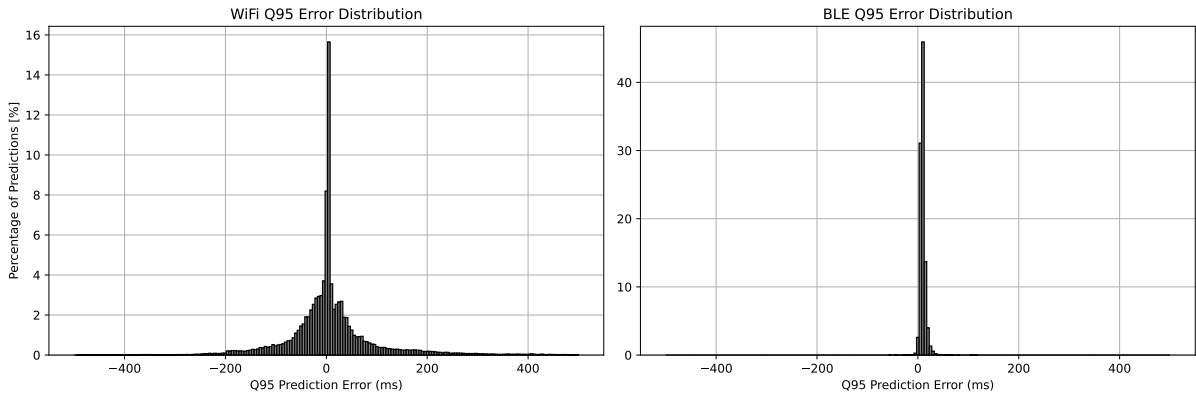


Figure 5.13: 95th quantile error distribution for the GMM-QR Pinball model

The flexibility of QR models, especially those trained with pinball loss, make them particularly effective at capturing the behavior of complex latency distributions, without requiring explicit distributional assumptions. This is particularly reflected by the tightness of the error distribution for the QR models when compared to the parametric ones.

5.5 Model Comparison

The five models introduced throughout this chapter differ fundamentally, either in how they are trained or how they represent the latency prediction: the parametric models predict a full distribution, while QR models directly estimate the 95th latency quantile. To compare their predictive accuracy, we evaluate the absolute error of the predicted 95th quantile against the empirical value, across the entire test set.

In addition to the trained models, we include a naive baseline estimator, which estimates the 95th latency quantile directly from a subset of raw latency observations. For each field f , let $\mathcal{L}_f = \{l_i\}_{i=1}^N$ denote the full set of latency samples. The baseline randomly selects a subset $\mathcal{L}_f^{\text{sub}} \subset \mathcal{L}_f$, where $|\mathcal{L}_f^{\text{sub}}| = n < N$. It then computes the 95th latency quantile as:

$$\hat{q}_{0.95,f}^{\text{baseline}} = Q_{0.95}(\mathcal{L}_f^{\text{sub}}) \quad (5.8)$$

Similarly to the other models, the baseline estimator is compared to the empirical quantile of the corresponding field:

$$q_{0.95,f}^{\text{true}} = Q_{0.95}(\mathcal{L}_f) \quad (5.9)$$

This way the performance of the baseline estimator depends purely on how representative the latency subset is of the entire fields statistics. Naturally as the size n grows, the performance will improve, for this evaluation a single batch was used for the baseline, according to the batching strategy in Section 5.2.

Figure 5.14 presents the CDF of the absolute prediction errors, similar to the individual evaluations conducted in Figure 5.9 and 5.13. Each curve represents a model, and how quickly the CDF rises, reflects the predictive accuracy of the model. Across both interfaces, the QR models, particularly the GMM-QR Pinball, consistently achieve the lowest prediction error. This is consistent with the intuition, that QR models perform well when the underlying distribution is non-Gaussian, skewed or multi-modal. Similarly, we observe a better performance, in predicting WiFi, from GMM-Param than its counterpart parametric model, once again reiterating the consequences of imposing a too simple distributional assumption on the latency distribution. Notably, the performance of GMM-Param suffers for BLE predictions, despite both the MV-Param and GMM-Param models, modeling the BLE distribution identically. This can be explained by an over-prioritization of WiFi predictions, due to each Gaussian component in the GMM contributing with its own loss value, meaning the loss is considerably larger by nature for WiFi predictions.

The baseline estimator, despite requiring no training, shows surprisingly competitive performance, especially for BLE. This is largely due to the low variability and unimodal nature of the BLE latency distribution. Crucially, the BLE simulator does not model stochastic fading, meaning any variance of latency is caused purely by the number of retransmissions. Including stochastic fading would introduce variance on the SNR distribution, likely resulting in more complex latency distributions, degrading the performance of the baseline estimator, as observed for WiFi.

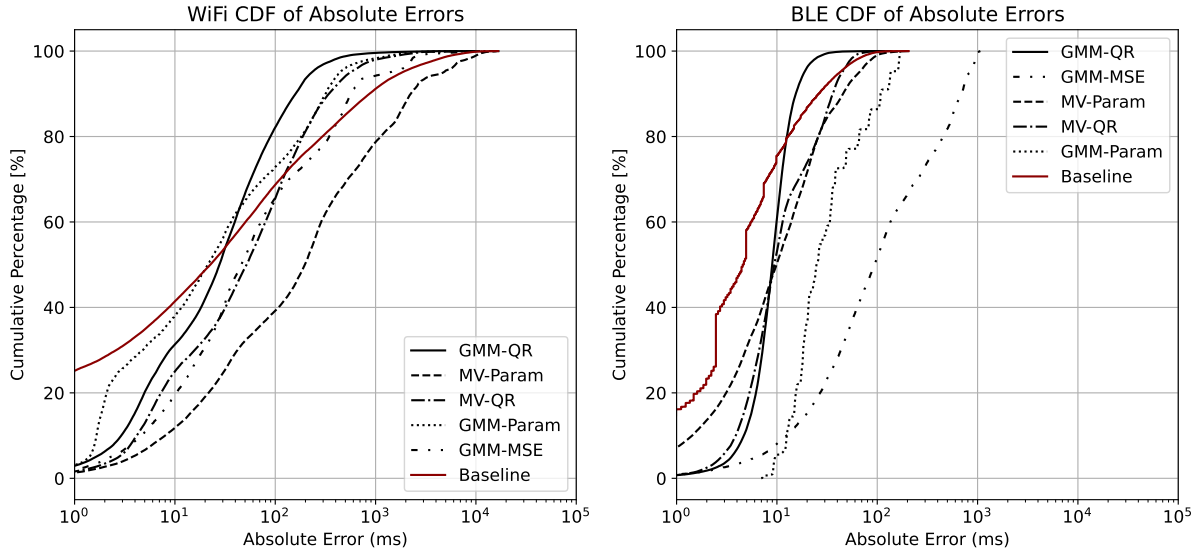


Figure 5.14: Cumulative density function of absolute prediction error of the 95th latency quantile for all trained models, includes a naive baseline estimator as reference. Note, GMM-QR MSE has been dubbed GMM-MSE.

Overall, this comparison highlights the trade-off between:

- Parametric models: providing access to the full latency distribution, allowing for post-hoc analysis, but require distributional assumptions on the underlying data.
- QR models: directly predicting the 95th latency quantile, proving more effective in environments with complex latency distribution, due to their ability to predict without imposing assumptions on the underlying distribution.

This comparison serves as an intermediate result, evaluating the predictive capabilities of each model in isolation. In Chapter 6, we expand on these results to construct uncertainty-aware prediction intervals using conformal prediction.

6 | Conformal Prediction for Uncertainty Calibration

Chapter 5 introduced and evaluated five neural-network based models for latency prediction:

- Mean-Variance Parametric Regressor (MV-Param)
- GMM Parametric Regressor (GMM-Param)
- Mean-Variance Quantile Regressor (MV-QR)
- GMM Quantile Regressor, MSE (GMM-QR MSE)
- GMM Quantile Regressor, Pinball (GMM-QR Pinball)

Each model produces either explicit latency quantile predictions (QR models) or a full parametric distribution from which the quantiles are derived (parametric models). While the predictive accuracy, evaluated in Chapter 5, gives insight into the average performance of the model, it does not fully capture the models uncertainty which is important when used in decision-making tasks such as interface selection.

In practical scenarios, particularly under varying and unpredictable network conditions, model uncertainty plays a central role in selecting a communication interface. A model, that simply produces a point prediction without expressing uncertainty may lead to unstable or unsafe decisions. To address this, point predictors can be paired with tools for constructing well-calibrated uncertainty intervals, enabling decisions to be made under explicit risk bounds [26].

One such tool is Conformal Prediction (CP), a post-hoc calibration method, which wraps any base model and produce uncertainty intervals with guaranteed statistical coverage. An illustration of this idea is shown in Figure 6.1, where a point prediction is post-hoc expanded into a calibrated uncertainty interval using CP. In this chapter, we apply CP to each of the latency predictors and extend the evaluation to account not only for prediction accuracy but also for the width of the resulting uncertainty intervals.

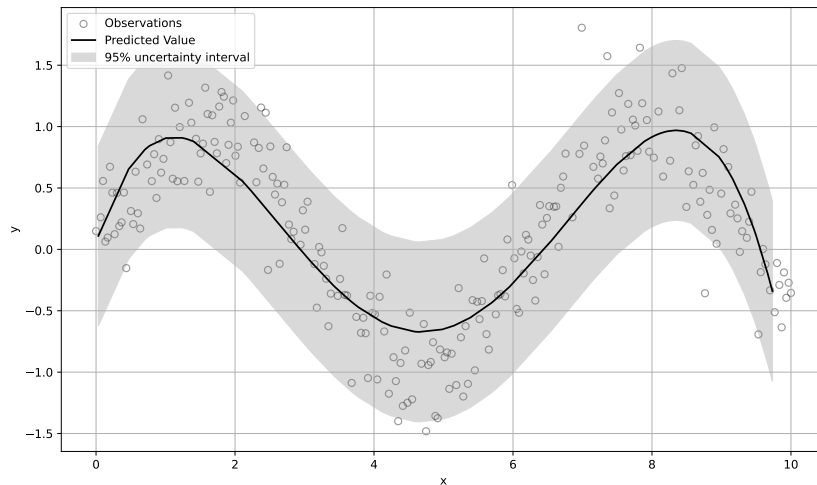


Figure 6.1: Illustration of conformal prediction, converting point prediction to calibrated uncertainty interval.

6.1 Theoretical Background

Conformal Prediction is a framework for constructing uncertainty intervals that are valid under minimal assumptions [26]. The strength of CP lies in its distribution-free nature. CP does not require knowledge about the underlying data distribution nor the architecture of the predictive model [26]. This makes CP well-suited for post-hoc calibration in settings where model architecture differs, such as this case where both quantile regression and parametric models are evaluated under a common uncertainty calibration framework.

While CP exists in many forms, a commonly used approach is Split Conformal Prediction [27]. Split CP leverages a calibration dataset

$$D_C = (\hat{Y}_1, Y_1), \dots, (\hat{Y}_n, Y_n), \quad (6.1)$$

and a test point \hat{Y}_{n+1} , where \hat{Y}_i and Y_i denote the predicted and true values of the i -th sample respectively [27]. Note that the CP calibration data, D_C is separate from the training and evaluation dataset used to train the ML model. The goal of CP is to construct an uncertainty interval $\hat{C}(\hat{Y}_{n+1})$, such that the ground truth Y_{n+1} lies within the interval with a certain probability. Specifically, the method guarantees the following marginal coverage:

$$1 - \alpha \leq \mathbb{P}(Y_{n+1} \in \hat{C}(\hat{Y}_{n+1})) \leq 1 - \alpha + \frac{1}{n+1}, \quad (6.2)$$

where $\alpha \in [0, 1]$ is a user-specified uncertainty level, and $\frac{1}{n+1}$ is a finite-sample guarantee. This finite-sample coverage guarantee holds under the assumption that the calibration set and the test point \hat{Y}_{n+1} are exchangeable, which in practice requires $\hat{Y}_{n+1} \sim P_0$, i.e. that the test point is drawn from the same underlying distribution as the calibration set [26]. The finite-sample guarantee is a central advantage of CP, as it provides valid non-asymptotic uncertainty intervals, regardless of the accuracy or complexity of the underlying predictor model [26].

6.1.1 Symmetric and Asymmetric Uncertainty Intervals

Once a predictive model has been trained and a separate calibration dataset D_C has been prepared, the first step in applying Conformal Prediction is to compute a nonconformity score for each calibration point. This score quantifies the degree to which a model's prediction deviates from the observed ground truth.

While CP allows flexibility in defining the scoring function, the most common choice in regression settings is the residual, i.e. the difference between the predicted and ground truth values. Based on how these residuals are treated, conformal intervals can be constructed in symmetric or asymmetric forms [27].

Symmetric Intervals

The most common setting for CP is using symmetric intervals. In this setting residuals are treated uniformly in both directions. The set of nonconformity scores are defined as the absolute residuals:

$$\mathcal{R} = \left\{ r_i = |Y_i - \hat{Y}_i| \right\}_{i=1}^n. \quad (6.3)$$

The calibration value is then obtained as the $\frac{[(n+1)(1-\alpha)]}{n}$ -quantile of the residuals, this adjusted quantile accounts for the discrete nature of the calibration set and ensures the theoretical coverage guarantee from Equation (6.2) is satisfied:

$$q_{\text{cal}} = Q_{\frac{[(n+1)(1-\alpha)]}{n}}(\mathcal{R}), \quad (6.4)$$

and the uncertainty interval for a test point \hat{Y}_{n+1} becomes:

$$\hat{C}(\hat{Y}_{n+1}) = [\hat{Y}_{n+1} - q_{\text{cal}}, \hat{Y}_{n+1} + q_{\text{cal}}]. \quad (6.5)$$

Symmetric intervals assume the distribution of residuals is approximately symmetric around the prediction. That is, the predictor model is unbiased and uniformly over- and underpredicts. This formulation is simple and effective, but can lead to overly conservative uncertainty intervals in the case of biased models with skewed residual distributions [27].

Asymmetric Intervals

In real-world scenarios, such as wireless communication, the distribution of residuals is often asymmetric. Asymmetries may stem from the loss function used to train the latency predictor. In such cases, an asymmetric form of CP is preferred.

In the asymmetric case, the scoring function is also residuals, but allows for negative residuals:

$$\mathcal{R} = \left\{ r_i = Y_i - \hat{Y}_i \right\}_{i=1}^n. \quad (6.6)$$

the residuals are partitioned into positive and negative components:

$$\mathcal{R}^+ = \{r \in \mathcal{R} | r > 0\}, \quad \mathcal{R}^- = \{r \in \mathcal{R} | r < 0\} \quad (6.7)$$

We then compute two independent calibration scores for either side of the uncertainty interval:

$$q_{\text{cal}}^+ = Q_{\frac{[(n+1)(1-\alpha)]}{n}}(r : r \in \mathcal{R}^+), \quad q_{\text{cal}}^- = Q_{\frac{[(n+1)(1-\alpha)]}{n}}(\{|r| : r \in \mathcal{R}^-\}) \quad (6.8)$$

yielding the final uncertainty interval:

$$\hat{C}(\hat{Y}_{n+1}) = [\hat{Y}_{n+1} - q_{\text{cal}}^-, \hat{Y}_{n+1} + q_{\text{cal}}^+] \quad (6.9)$$

While this formulation allows the interval to expand independently on each side, it does not satisfy the marginal coverage guarantee from Equation (6.2). Instead, the coverage guarantees are split conditionally on whether the model under- or over-predicts the true value:

$$\begin{aligned} 1 - \alpha &\leq \mathbb{P}\left(Y_{n+1} \in \hat{C}(\hat{Y}_{n+1}) \mid Y_{n+1} > \hat{Y}_{n+1}\right) \leq 1 - \alpha + \frac{1}{|\mathcal{R}^-| + 1} \\ 1 - \alpha &\leq \mathbb{P}\left(Y_{n+1} \in \hat{C}(\hat{Y}_{n+1}) \mid Y_{n+1} \leq \hat{Y}_{n+1}\right) \leq 1 - \alpha + \frac{1}{|\mathcal{R}^+| + 1} \end{aligned} \quad (6.10)$$

Here, \mathcal{R}^+ and \mathcal{R}^- refer to the sets of positive and negative residuals defined in Equation (6.7). These conditional coverage guarantees hold separately for the upper and lower bounds, assuming that the calibration set contains sufficient residuals of the appropriate sign. This has practical implications for the implementation of asymmetric CP: the primary motivation for asymmetric intervals is to correct for model bias, yet a strongly biased model may delay calibration, as it requires enough under- and overpredictions to reliably estimate both bounds. A strategy for implementing both symmetric and asymmetric CP will be proposed in Section 6.2.

6.2 Implementation Details

This section outlines how conformal prediction is applied in practice. While five latency models were introduced in Chapter 5, this section focuses on the implementation of CP for the GMM-QR model, with corresponding results for subsequent models being deferred to Appendix D.

Although asymmetric conformal intervals offer greater flexibility and could potentially yield tighter uncertainty bounds, they require sufficient samples of both under- and overpredictions in the calibration set. This requirement poses a practical challenge during deployment, especially for models trained with asymmetric loss functions, e.g, pinball loss, which naturally produce skewed residuals.

Therefore, all evaluations in this chapter are based on symmetric uncertainty intervals, which provides robust coverage without relying on balanced residuals. To address the limitations of this approach and enable asymmetric intervals in future deployment, we propose a calibration strategy in Section 6.2.3, for handling skewed residuals in the asymmetric framework.

6.2.1 Calibration Procedure for GMM-QR

The GMM-QR model produces explicit predictions of the 95th latency quantile, denoted $\hat{y}_{0.95,i}$, for both WiFi and BLE. While prediction provides an estimate of the tail latency, it is not guaranteed to be calibrated. Therefore, we apply symmetric CP to construct uncertainty intervals around the model output, calibrated to a target coverage of 90%, i.e. $\alpha = 0.10$.

When deploying the system in a new environment, calibration data is first collected by transmitting video frames over both interfaces simultaneously. Each transmission is logged along with the channel state, environmental metadata and the observed latency. These observations form the calibration set.

For each calibration sample i , we compute the residual between the quantile of the observed latencies $Y_{0.95,i}$ and the model's predicted quantile $\hat{y}_{0.95,i}$:

$$\mathcal{R} = \{r_i\}_{i=0}^n = \{Y_{0.95} - \hat{y}_{0.95,i}\}_{i=0}^n \quad (6.11)$$

The conformal calibration score is then obtained as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ -quantile of the residuals:

$$q_{\text{cal}} = Q_{\frac{\lceil (n+1)(1-\alpha) \rceil}{n}}(\mathcal{R}) \quad (6.12)$$

The final uncertainty interval is constructed as:

$$\hat{C}(\hat{y}_{0.95,i}) = [\hat{y}_{0.95,i} - q_{\text{cal}}, \hat{y}_{0.95,i} + q_{\text{cal}}] \quad (6.13)$$

Note that the adjusted quantile is only well-defined when the calibration set contains sufficient samples n . Specifically the adjusted quantile must satisfy:

$$\frac{\lceil (n+1)(1-\alpha) \rceil}{n} < 1, \quad (6.14)$$

where a quantile greater than 1 is undefined and a quantile equal to 1 would mean an infinitely large uncertainty interval. The relationship between the minimum required number of samples n_{\min} and the miscoverage level α is showcased by Figure 6.2. For the target coverage of $1 - \alpha = 0.90$, this imposes a minimum calibration size of $n_{\min} = 19$.

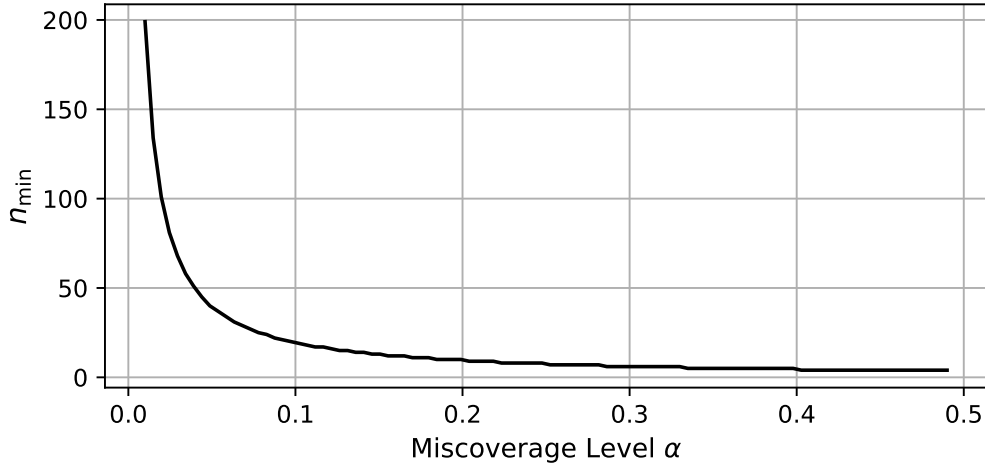


Figure 6.2: Relationship between minimum calibration set size n_{\min} and miscoverage level α .

The calibration procedure was performed independently for WiFi and BLE to account for interface-specific latency dynamics such as spectrum policies. The calibration step was repeated for each field independently, mimicking real-world deployment, where calibration occurs locally per environment.

Figures 6.3 and 6.4 show the distribution of calibration scores across all fields for WiFi and BLE respectively. A majority of fields exhibit a low calibration score below 20 ms. However, a non-negligible number of fields require a larger calibration, highlighting the variability in model accuracy across deployment environments. The following section validates the implementation of CP empirically evaluating the coverage and ensuring the theoretical guarantees are satisfied.

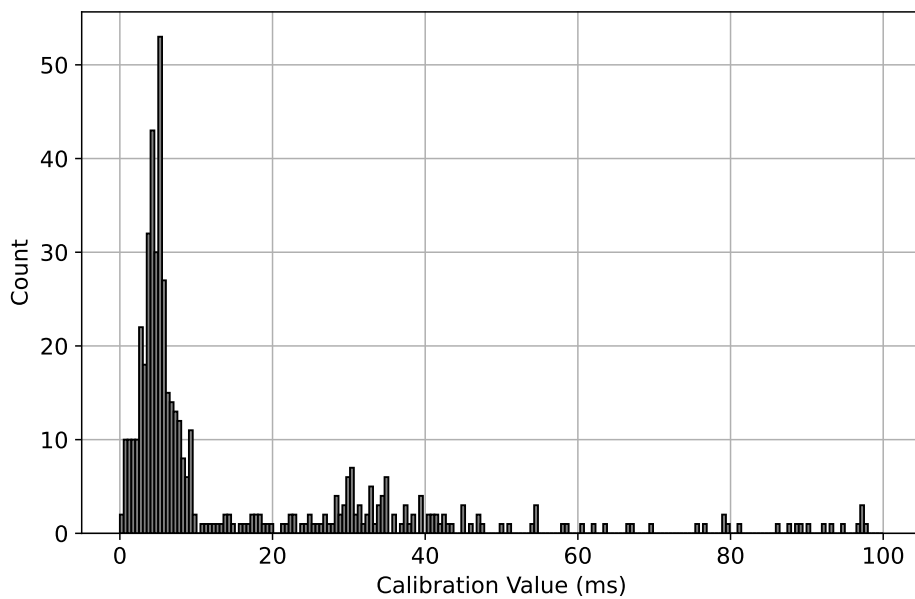


Figure 6.3: WiFi calibration score distribution.

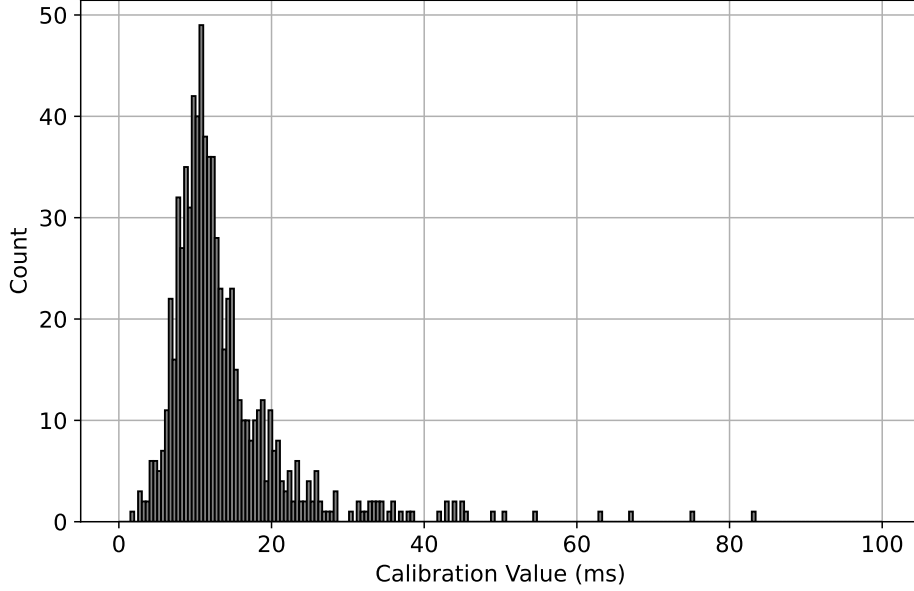


Figure 6.4: BLE calibration score distributions.

6.2.2 Coverage Validation

To validate the implementation of CP, the empirical coverage is measured, i.e. the proportion of true latency quantiles that fall within the calibrated conformal prediction interval. Varying the size of the calibration set, simulates spending more or less time in the calibration phase upon arriving in a new environment. The effect of the calibration set size is expressed by the miscoverage term in Equation (6.2), approaching zero as the calibration set grows.

Each calibration sample corresponds to a batch of latency observations spanning approximately 20 seconds. Therefore, increasing the number of calibration samples reflects longer on-site calibration time. For each configuration of calibration set sizes, the uncertainty intervals are recalibrated following the procedure from Section 6.1, and the empirical coverage is evaluated as:

$$\text{Coverage} = \frac{1}{|D_{\text{test}}|} \sum_{i \in D_{\text{test}}} \mathbf{1} \left\{ Y_i \in \hat{C}(\hat{y}_{0.95,i}) \right\} \quad (6.15)$$

Figure 6.5 shows both the empirical and theoretical coverage bounds as a function of the time spent calibrating. The dark shaded area illustrates the minimum calibration size n_{\min} , required for Equation (6.14) to be satisfied, while the light shaded area denotes the coverage bounds. As expected, the coverage approaches the nominal level of $1 - \alpha = 0.90$, as the calibration set grows and the miscoverage term vanishes as dictated by Equation (6.2). This highlights a trade-off between coverage guarantees and longer on-site calibration time.

In addition to evaluating the average coverage over calibration duration, we also evaluate how the coverage is distributed across individual fields. Figure 6.6 shows the distribution of empirical coverage, computed separately for each field at a fixed calibration duration of $T = 500$ s. We observe that most fields meet or exceed the nominal coverage of 90%, however some fields exhibit slight undercoverage, which is consistent with the finite-sample miscoverage term from Equation (6.2). Note that the results presented here correspond to the Wi-Fi interface. The BLE interface exhibited nearly identical behavior and its results are therefore deferred to Appendix D.

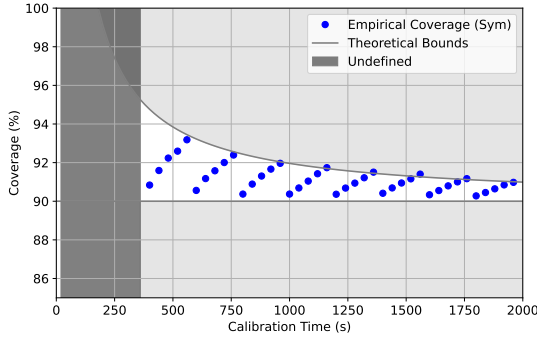


Figure 6.5: Empirical and theoretical coverage as a function of calibration time.

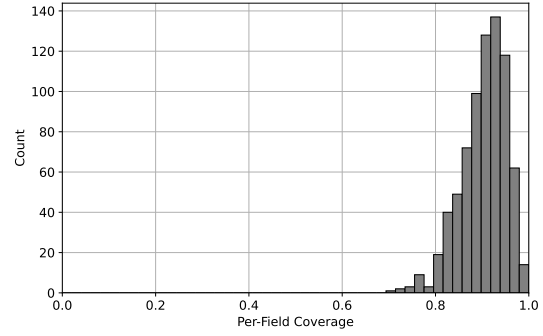


Figure 6.6: Per field coverage distribution for WiFi.

6.2.3 Strategy for Asymmetric Calibration

While this work uses symmetric conformal prediction for evaluation, asymmetric intervals are conceptually attractive in situations where the model exhibits biased residuals. As shown in Section 6.1, asymmetric CP requires separate residual sets for under- and overpredictions. However, this introduces a practical challenge: to compute both calibration scores, the calibration set must contain a sufficient number of each residual. When the model is biased towards consistently overpredicting, it may drastically impact the calibration time, due to insufficient underprediction residuals.

We propose two alternative strategies for enabling asymmetric CP in a practical deployment: a reactive strategy, in which calibration is delayed until sufficient residuals are gathered, here the state of each residual set is periodically monitored, and a proactive method in which a probabilistic model is used to estimate the required number of total calibration samples to feasibly compute the calibration scores.

Reactive Strategy: Observe and Calibrate

In the reactive approach, the system alternates between data collection, inference and calibration. It proceeds as follows:

1. Collect calibration samples: Transmit a batch of packets on both interfaces and log the data.
2. Run inference and compute residuals: Feed the collected samples to the latency predictor and compute the residuals as in Equation (6.7).
3. Check the residual balance: If the condition

$$\min(|\mathcal{R}^+|, |\mathcal{R}^-|) > n_{\min},$$

is satisfied the asymmetric calibration can proceed.

4. Repeat: If the condition is not satisfied, repeat from step 1.

Proactive Strategy: Estimate and Calibrate

The proactive approach models how many samples are needed before asymmetric calibration becomes feasible, given a known or estimated residual bias. This is particularly useful when

minimizing the calibration time is important.

Let $p_u = P(Y_{n+1} \geq \hat{Y}_{n+1})$ and $p_o = P(Y_{n+1} < \hat{Y}_{n+1})$ denote the probability of under- and overprediction respectively. The required number of transmissions to observe at least n_{\min} residuals of a given type can be modeled as a negative binomial distribution:

$$\mathcal{X} \sim \text{NB}(r = n_{\min}, p), \quad (6.16)$$

where r is the required number of samples and $p \in \{p_u, p_o\}$ is the probability of observing an under- or overprediction. The probability of seeing n_{\min} such residuals, within t batches is given by the cumulative density function (CDF):

$$P(\mathcal{X} \leq t) = \sum_{i=n_{\min}}^{t-1} \binom{i+n_{\min}-1}{i} p^{n_{\min}} (1-p)^i \quad (6.17)$$

The CDF allows for estimating the likely calibration time needed to satisfy the requirement $\min(|\mathcal{R}^+|, |\mathcal{R}^-|) > n_{\min}$. The probabilities p_u and p_o can be estimated from prior deployments or a warm-up phase upon system deployment.

6.3 Interval Width Comparison

The empirical coverage ensures the statistical validity of the conformal prediction implementation. It does however, not reflect on the usefulness of the resulting uncertainty intervals. In practical applications such as interface selection, overly conservative intervals may give statistically valid coverage yet offer little value in the decision making. Therefore, we use the interval widths produced by each model after calibration as a performance metric, under the constraint that each model achieves statistically valid coverage. For each test point $i \in D_{\text{test}}$, the interval width is defined as:

$$w_i = q_{\text{cal},i}^+ + q_{\text{cal},i}^- \quad (6.18)$$

where q_{cal}^+ and q_{cal}^- are the upper and lower calibration values, derived from the residuals, as described in Section 6.1. Since the calibration is performed on a per field level. The final evaluation should evaluate the interval width across all fields.

Table 6.1 reports empirical coverage, mean and median interval width for each of the five prediction models. Figure 6.7 visualizes the corresponding cumulative density function of interval widths serving as a comparison of interval tightness between models.

From Table 6.1 it is observed that the median width remains consistently low for all models, indicating that, in many environments, the uncertainty intervals are relatively tight. However the mean width is substantially higher, indicating the presence of some extreme outliers. This shows that the models fails to predict the latency in some fields, resulting in high uncertainty in these fields.

Model	WiFi		BLE	
	Coverage [%]	Width [ms]	Coverage [%]	Width [ms]
GMM-QR Pinball	90.37	Mean: 351.29 Median: 24.27	90.37	Mean: 13.86 Median: 11.59
GMM-QR MSE	90.50	Mean: 505.26 Median: 131.70	90.66	Mean: 261.27 Median: 183.92
MV-QR	90.27	Mean: 430.98 Median: 22.36	90.12	Mean: 16.51 Median: 10.39
GMM-param	90.41	Mean: 1299.34 Median: 1040.07	90.70	Mean: 43.02 Median: 24.91
MV-param	90.51	Mean: 355.53 Median: 42.28	90.27	Mean: 13.40 Median: 6.63

Table 6.1: Uncertainty interval coverage and width for all models, grouped by interface. Width statistics are reported as both mean and median, computed across all fields.

Among the models, GMM-QR Pinball yields the tightest uncertainty intervals across both interfaces, confirming its ability to directly model the 95th latency quantile. The MV-QR model performs nearly as well, particularly on BLE, supporting the assumption that BLE latency is well-approximated by a Gaussian distribution. In contrast, the parametric models, especially GMM-param, exhibit significantly wider intervals, as the conformal calibration compensates for distributional mismatches between predicted and empirical latency. These findings are further illustrated in Figure 6.7, which shows the cumulative density function of interval widths across all fields. The CDF highlights that GMM-QR and MV-QR maintain consistently low-width intervals in most environments, while the parametric models display heavy tails due to rare but extreme prediction errors. This confirms that quantile-based predictors not only improve accuracy, as shown in Chapter 5, but also lead to more informative and compact uncertainty intervals under conformal calibration. This is particularly evident with the chosen parametric distributions, Gaussian and GMM, but more complex distributional assumptions may improve the performance of the parametric models, this is further explored in Chapter 7.

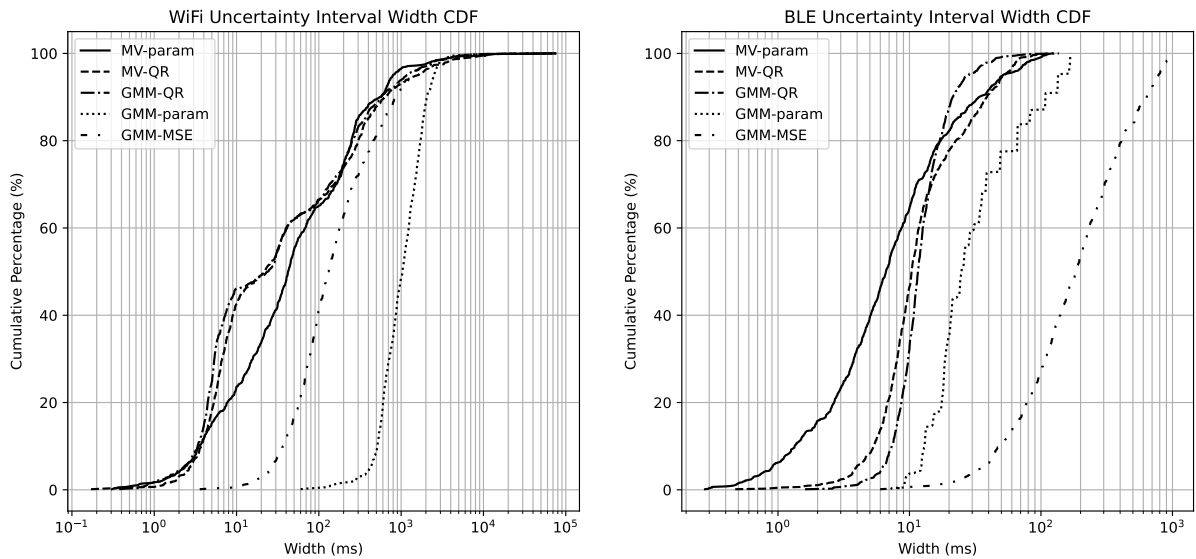


Figure 6.7: Cumulative density function of uncertainty interval width for all trained models, with a calibration set collected over 500 s. Note that GMM-QR MSE has been dubbed GMM-MSE.

7 | Discussion and Future Work

7.1 Insights from Model Prediction and Calibration

The conformal prediction results presented throughout Chapter 6 and summarized by Table 6.1 confirm that the framework consistently provides valid coverage near the target of 90% across both WiFi and BLE for all trained model variants. This both validates conformal prediction as a robust post-hoc calibration framework and serves as a sanity check for its correct implementation.

However, the mean interval widths varied significantly across the different models. As hypothesized in Chapter 6, there is a correlation between predictive accuracy and the tightness of the uncertainty intervals. As expected the GMM-QR model achieved the best predictive accuracy and tightest uncertainty intervals. More broadly, the QR models generally outperformed the parametric ones, particularly in WiFi predictions, where the true latency distribution deviates significantly from the assumed parametric Gaussian and GMM distributions. This highlights the core advantage of QR models, i.e. their ability to adapt to non-Gaussian, heavy-tailed or multi-modal distributions, due to making no assumption on the distribution of the underlying data.

Notably, the BLE predictions exhibited far less variation in interval width across the models, with all models achieving the target coverage with relatively narrow intervals. This is an artifact of the BLE simulator implementation, which lacks stochastic fading and therefore exhibits less channel variability. As discussed in Chapter 5, this likely contributed to the surprisingly strong performance of the baseline predictor for BLE. Similarly, the interval width of the GMM-Param model suffered significantly compared to its MV-Param counterpart, despite both models modeling the BLE SNR and latency with the gaussian distributions. This supports the hypothesis, that BLE predictions are being suppressed by the GMM loss for WiFi predictions, thereby forcing the model to prioritize learning the WiFi distribution.

Another notable pattern was the discrepancy between mean and median interval widths observed for all models and interfaces. As illustrated by Figure 6.7, many fields have narrow uncertainty intervals, but a few outliers contribute disproportionately to the overall mean. This suggests that either the models generalize poorly to rare scenarios in the calibration set, or the conformal prediction dataset contains particularly challenging environments. One plausible explanation is that an abundance of environments in the conformal prediction dataset, include a majority of interference devices outside the CSMA radius, thereby causing a large variability in MCS index, causing multi-modality or skewed-ness beyond what was seen in the model training dataset.

7.2 Alternative Framework for Parametric Models

While the parametric models evaluated in this thesis provide a tractable representation of latency distributions, their performance has consistently fallen behind that of the quantile regressor models. This is especially evident in the presence of a complex, heavy-tailed or skewed latency distribution, like the ones observed for WiFi. In this case, the distributional assumption directly constrains the model's ability to express the latency distribution by imposing an assumption on the underlying distribution.

The parametric models in this thesis rely on either a simple Gaussian assumption, MV-Param, or a Gaussian mixture model assumption, GMM-Param to represent the latency distribution.

Although these distributions are convenient to work with and interpret, they may fail to capture the tail behavior in a complex simulation environment.

Recent work in [28] propose a more flexible solution grounded in Extreme Value Theory (EVT). In their approach, the tail of the latency distribution is explicitly modeled as a Generalized Pareto Distribution (GPD), while the bulk of the distribution is modeled as GMM. This mixture of distinct distributions forms the basis for their Mixture Density Network (MDN), which maps network conditions to the parameters of the composite distributions [28]. This supports the idea from Chapter 5, that the performance of parametric models can indeed be improved, with more informed decisions on the underlying distributional assumptions. Implementing the framework from [28], into the parametric pipeline presented in this thesis, would potentially bridge the gap between parametric and QR models.

Furthermore, this framework opens up for the idea of using a distribution classifier to aid the parametric model in choosing the distributional assumption. With such framework, the system would likely contain a collection of trained parametric models, specialized in their respective distributions from which the classifier serves as an orchestrator of which model to use given the current propagation environment. An overview of this extended framework is showcased by Figure 7.1, in which the system contains a bank of three parametric models, the MV- and GMM-Param models from this thesis and the GMM-GPD model from [28].

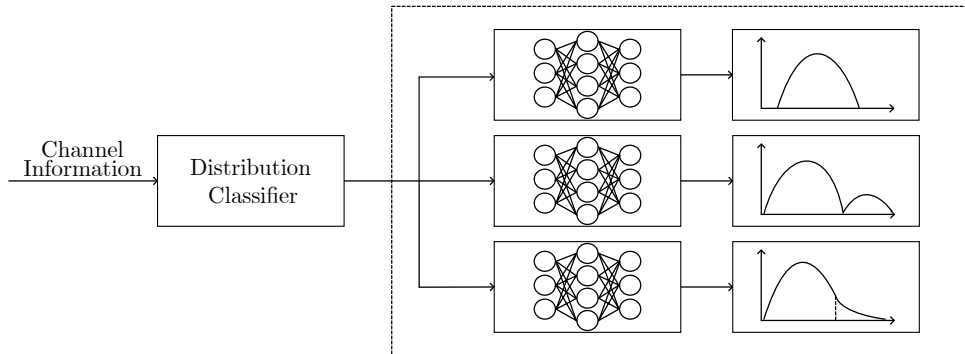


Figure 7.1: Illustration of the proposed framework with a distribution classifier orchestrating a bank of trained parametric models each specialized in its own distribution.

7.3 Data Generation

The machine learning model is never better than the data used to train it. The data generated and used in this thesis was generated primarily with the focus of capturing the behaviors of FHSS and CSMA/CA, to evaluate the ML model ability to predicts these effect on latency and to showcase the use of conformal prediction to quantify the uncertainty. The data in the thesis is based on an outdoor environment with TX, RX and interference devices randomly positioned in a plane of 400 x 400 meters. The path loss was calculated using path loss models such as Two-Ray Ground Reflection Model and the IEEE defined model for Urban Micro environments. As these models are generalized for their respective type of environments, it is not expected that models trained on the simulated data can be successfully deployed in the real world due to the generalizations in the model leading to deviations between simulated and real data.

A different data generation approach which was considered during the thesis, was to generate environment specific data using Sionna RT developed by NVIDIA.

7.3.1 Environment Specific Data Generation using Sionna

Sionna, is an open-source tool for link-level simulations using GPU acceleration. Sionna uses ray tracing (RT) for the simulation of radio wave propagation, incorporating environment parameters, such as transmitter and receiver orientations and position, material properties and antenna patterns [29]. This allows for environment specific simulations with high details. In Figure 7.2 an example simulation is performed in a digital twin environment we have created at AAU campus. The environment is created by importing data from OpenStreetMap into blender, assigning material properties [30]. Tools like `RaySolver` and `RadioMapSolver`, allow for the computation of channel impulse responses (CIR) and SINR heatmaps as shown by Figure 7.2a and 7.2b. As described in Appendix B.1, the CIR is also available from the WiFi devices, and can therefore be used as a more detailed description of the channel compared to the SINR value. The SINR heatmaps can be used for placing interference devices and classifying them as CSMA or non-CSMA devices.

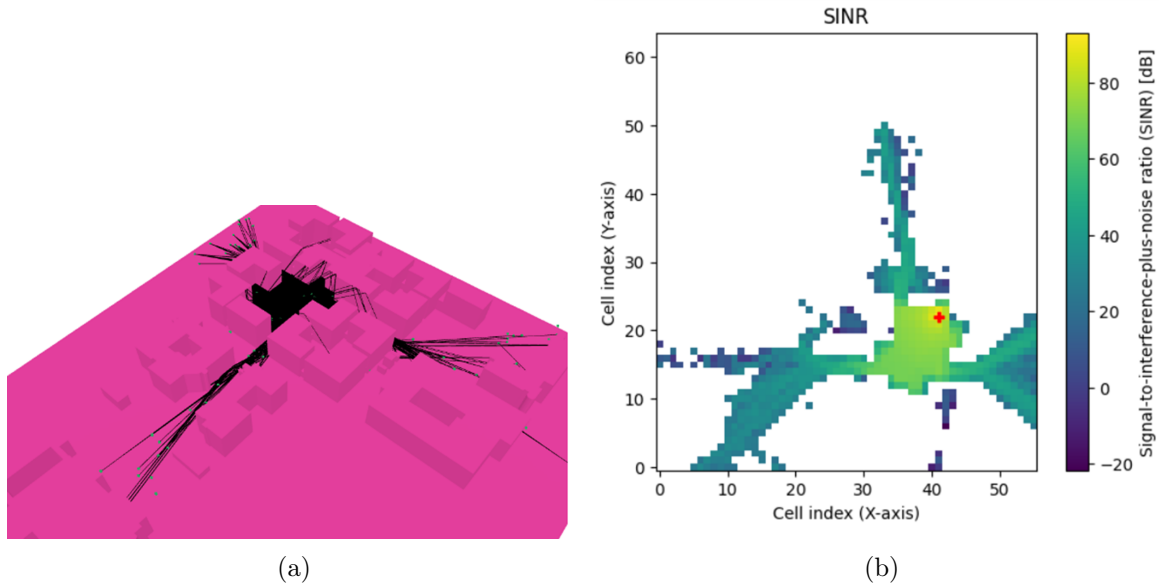


Figure 7.2: (a) Imported 3D environment in Sionna with highlighted ray-tracing. (b) Corresponding power heatmap with highlighted transmitter position, takes into account device placement constraints based on the environment. The example environment is taken from AAU campus.

For the example environment simulation all surfaces are set to concrete. The assignment of material properties is important, when making a digital twin faithful to a real world environment, and is an ongoing research area with some of the proposed methods being [31]:

- Manually assigning material parameters of individual objects in the environment.
- Tuning material parameters using measurements of path loss or delay spread.
- Learning material parameters by differentiable ray tracing

As both of the last methods require, measurements for calibrating the material parameters, it should be discussed what is actuality gained compared to just gathering real world data. One benefit of a digital twin is the flexibility to generate rare scenarios in the environment, ensuring the latency predictor is trained on rare observations and captures the extremes. Another advantage of having a digital is speeding up the training or evaluation of an actor module, since digital twins can operate faster than real time.

7.3.2 Continual Learning Framework

Creating a faithful simulation environment is complex and it may be beneficial to use real-world data for training since the complexities will be inherently captured in the data. However, collecting such data is significantly more time consuming and labor-intensive, compared to synthetic simulation.

Regardless of, whether simulated or real-world data is used for training, deployment in a new environment will always involve collecting an environment-specific dataset for conformal calibration. This calibration dataset, although primarily used to quantify uncertainty, also opens the door for continual learning. Future deployments may leverage these calibration datasets to further train the predictive model, either by fine-tuning or by training a new model on the expanded dataset. Since these datasets are collected as a standard operational routine, a continual learning framework would incur no additional data collection or computational cost.

In this way, the calibration procedure serves not only to provide calibrated uncertainty intervals, but also as a method of facilitating long-term model improvement.

7.4 Dynamic Symmetric and Asymmetric Conformal Calibration Strategy

A key contribution of this thesis is the application of conformal prediction to provide statistically valid uncertainty intervals for latency quantile predictions. While conformal prediction guarantees marginal coverage regardless of, how the interval was formed, Chapter 6 concluded that asymmetric intervals are attractive, in case of skewed or biased data distributions.

In Chapter 6 we discussed the challenge of collecting sufficient samples to construct independent calibration values for either side of the interval. It was concluded that calibration time may drastically increase, if the rate at which either under- or overpredictions are observed is low. Nevertheless, we proposed two strategies for implementing asymmetric conformal prediction:

- Reactive strategy: Repeatedly sample, run inference and observe the state of both residual sets. When the criteria $\min(|\mathcal{R}^+|, |\mathcal{R}^-|) \geq n_{\min}$ is satisfied, calibration can proceed.
- Proactive strategy: Model the required number of samples as a negative binomial distribution, and estimate the required calibration time prior to collecting samples.

A practical extension of these strategies is to treat symmetric intervals as a fallback calibration method. A simple implementation of this would be introducing a calibration time deadline T_{cal}^{\max} , for the reactive strategy this can be formally expressed as:

$$T^* = \arg \min_t (\min(|\mathcal{R}^+(t)|, |\mathcal{R}^-(t)|) \geq n_{\min}), \quad (7.1)$$

where T^* is the earliest time, in which both residual sets have sufficient samples for calibration. The calibration mode is then selected as:

$$\text{Calibration Mode} = \begin{cases} \text{Asymmetric} & \text{if } T^* \leq T_{\text{cal}}^{\max} \\ \text{Symmetric} & \text{if } T^* \geq T_{\text{cal}}^{\max} \end{cases} \quad (7.2)$$

Similarly for the proactive strategy, we simply evaluate the estimated calibration time \hat{T}_{cal} against the deadline T_{cal}^{\max} :

$$\text{Calibration Mode} = \begin{cases} \text{Asymmetric} & \text{if } \hat{T}_{\text{cal}} \leq T_{\text{cal}}^{\max} \\ \text{Symmetric} & \text{if } \hat{T}_{\text{cal}} \geq T_{\text{cal}}^{\max} \end{cases} \quad (7.3)$$

In summary, the choice between symmetric and asymmetric intervals introduce a trade-off between shorter calibration times with symmetric- and tighter intervals with asymmetric uncertainty intervals. To manage this, we propose a strategy, which dynamically assess whether asymmetric calibration is feasible and otherwise falls back to symmetric calibration. This flexibility ensures the system remains deployable in the case of skewed or biased data distribution.

One important limitation of conformal prediction is its reliance on the assumption that the calibration set and future test samples are exchangeable, i.e. they are drawn from the same underlying distribution. In practice, this assumption may be violated under distribution shifts. This happens if the propagation environment changes significantly. Under a distribution shift, the calibrated intervals may no longer provide valid coverage. Indicating a new calibration set should be collected and the calibration redone. As such, a mechanism for detecting distribution shifts would be beneficial. A primitive solution is to simply monitor the empirical coverage and assume a distribution shift if the coverage deviates significantly from the expected $1 - \alpha$ value.

7.5 Actor Implementation

While this thesis focused primarily on the prediction and uncertainty quantification of latency quantiles, a natural extension is to implement and incorporate a decision making module, i.e. the actor module as described in Chapter 3. Such an actor would be responsible for translating the predicted latency bounds into a concrete interface selection, based on predicted latency behavior and associated uncertainty. This section outlines two potential actor implementations: a rule-based naive actor and a more flexible reinforcement learning agent.

7.5.1 Naive Actor

A straightforward strategy for interface selection is to always choose the interface with the lowest predicted upper latency bound. In the context of conformal prediction, this corresponds to:

$$\text{score}_i = q_{0.95,i} + q_{\text{cal},i}^+ \quad (7.4)$$

This rule is well-suited to applications with tight latency budgets, which prioritize worst-case performance, it is also computationally lightweight and easily justifiable from a safety perspective.

However, this upper-bound rule is not always sufficient, as shown by Figure 7.3. In the first case illustrated by Figure 7.3, both interfaces have the same upper bound and this simple rule would therefore indicate no preference, despite WiFi exhibiting a larger interval, and therefore better opportunity for low latencies. As such, interval size encodes potential benefit and not just uncertainty.

This dilemma is further nuanced by the second case illustrated by Figure 7.3. In this case BLE has both a higher upper-bound, but also a lower lower-bound than WiFi. In such a case, the interface selection becomes highly application-specific with a risk trade-off of exceeding a latency threshold for the opportunity of a lower latency, or choosing the safe option at the cost of a more conservative interval. These two scenarios highlight the limitations of the simple decision rule, of choosing solely based on the upper latency bounds.

To better capture this the trade-off, the decision rule can be extended to include a penalty/reward for wide intervals:

$$\text{score}_i = q_{0.95,i} + q_{\text{cal},i}^+ + \lambda(q_{\text{cal},i}^+, q_{\text{cal},i}^-), \quad (7.5)$$

where λ is a tunable parameter, which adjusts the risk aversion. Small values $\lambda \leq 1$ encourage more opportunistic selection, while larger values $\lambda \geq 1$ emphasize more reliable and tighter intervals.

In practice, the interface selection process is rarely governed by the latency bounds alone. Additional application-specific metrics like energy consumption, data distortion due to compression, or interface-specific costs may all influence the optimal choice. These considerations introduce trade-offs which cannot be captured by latency intervals alone. As a result, while the naive actor offers an interpretable and low-complexity baseline, it may fall short in complex scenarios with multi-objective utility functions, motivating the use of more comprehensive interface selection strategies.

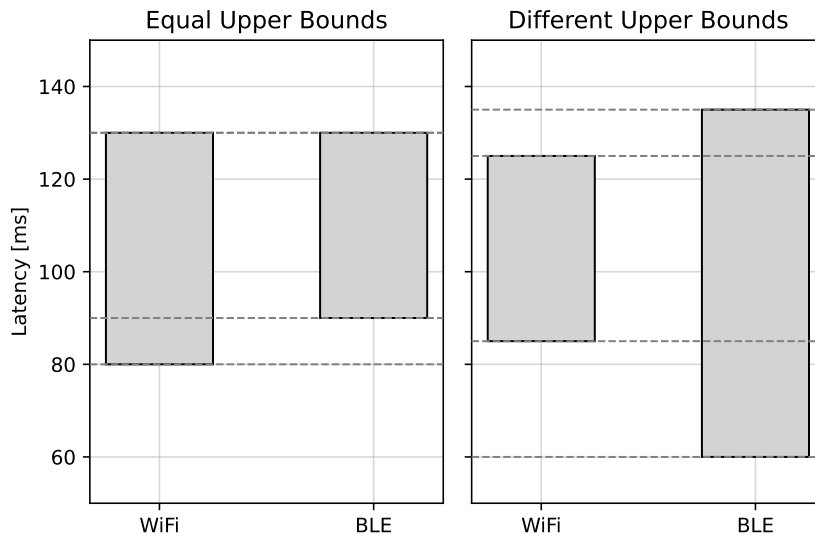


Figure 7.3: Visualizes two scenarios of interface selection. For equal upper bounds, the selection is obvious. However it becomes not straightforward, when upper bounds and interval sizes differ.

7.5.2 Reinforcement Learning-Based Actor

To address the limitations of the naive selection strategy, a more flexible alternative is to implement an actor using reinforcement learning (RL). An RL-based actor can learn a policy, which dynamically selects the optimal interface based on the provided latency intervals and application-specific utility metrics.

The conformal prediction framework provides natural input features for the RL agent. The predicted upper bound $q_{0.95,i} + q_{\text{cal}}^+$, serves as an estimate of the worse-case latency, and can be used to enforce reliability constraints by penalizing latency violations. Meanwhile, the interval width $|q_{\text{cal}}^- + q_{\text{cal}}^+|$ quantifies prediction uncertainty, and can be interpreted as a risk signal. Past outcomes such as observed latencies or task success rates, form the basis of the agents reward, thereby guiding the agent to improve its policy.

The agent's reward function can be designed as in Chapter 3, to reflect various application-specific trade-offs like:

- Minimize latency violations or end-to-end latency, i.e. exceed some reliability threshold.
- Minimize energy consumption, particularly important on battery-constrained devices.

- Avoid compression-induced distortion in lossy transmission.
- Penalize interface switching overhead.

This enables the agent to develop context-aware behaviors. For instance, it may prefer BLE when the predicted performance is similar for both interfaces in order to lower energy consumption. It may learn to utilize interface diversity strategies like cloning or weighted transmissions if the latency budget is tight, or it may learn to defer transmission if both interfaces are predicted to violate latency budgets. This approach aligns closely with the original optimization problem proposed in Chapter 3, where interface selection is posed as a utility-minimization problem under latency, energy and distortion constraints. While the implementation of such a RL-based actor was beyond the scope of this thesis, the foundation provided by the conformalized latency predictor makes it a natural next step.

8 | Conclusion

This thesis addressed the critical challenge of ensuring reliable low-latency communication in the congested unlicensed spectrum, a problem starkly highlighted by real-world scenarios such as the AAU Space Robotics team’s experience at the ERC. The core of this work was to answer: *How can we provide a robust, statistically sound measure of uncertainty for latency predictions to enable an intelligent and reliable interface selection to improve latency performance?*

To this end, we proposed, developed and evaluated a novel approach centered on Uncertainty-Aware Conformalized Quantile Regression. This method moves beyond traditional point predictions by estimating an interval for a high quantile of the latency, specifically the 95th latency quantile (Q95), and providing a statistically valid guarantee that the true latency quantile will fall within this interval with a predefined confidence level of 90%.

A comprehensive simulation framework was developed to generate realistic link-level data for WiFi and BLE interfaces, capturing the complex interplay of propagation, interference, and MAC-level behaviors such as CSMA/CA and FHSS. This data was instrumental in training and evaluating five distinct latency predictor models, categorized into parametric (MV-Param and GMM-Param), and quantile regression (MV-QR, GMM-QR MSE and GMM-QR Pinball), alongside a naive baseline estimator.

The evaluation in Chapter 5 demonstrated that quantile regression models, particularly the GMM-QR Pinball model, achieved superior predictive accuracy for the 95th latency quantile. It consistently showed the lowest absolute prediction error across both WiFi and BLE interfaces, outperforming not only the baseline estimator, which relies on raw subsampled latency observations, but also the parametric models like GMM-Param. This superiority is attributed to the QR models’ ability to directly model the quantile without imposing rigid potentially mismatched distributional assumptions, which proved limiting for parametric models, especially for the complex, multi-modal latency distributions observed in WiFi.

Chapter 6 successfully applied split conformal prediction to calibrate the uncertainty of these latency predictions. The GMM-QR Pinball model, our best-performing variant, achieved the target 90% coverage, specifically 90.37% for both WiFi and BLE. Importantly, it also yielded significantly tighter uncertainty intervals compared to the parametric models while maintaining this coverage. For instance, the GMM-QR Pinball model produced a median interval width of 24.27 ms for WiFi and 11.59 ms for BLE. In contrast, the MV-Param model, the better of the two parametric approaches in this regard, had a wider median interval of 42.28 ms for WiFi and a very competitive 6.63 ms for BLE but with generally higher absolute prediction errors. The GMM-QR Pinball’s ability to produce narrower intervals, e.g., nearly halving the median WiFi width compared to MV-Param, while ensuring statistical coverage underscores its practical advantage. The observed discrepancy between mean, e.g., 351.29 ms for WiFi with GMM-QR Pinball, and median interval widths across all models suggested that while performance was strong in many environments, certain challenging environments led to larger uncertainties, highlighting areas for future model refinement.

The discussion in Chapter 7 explored avenues for future work, including alternative parametric modeling frameworks like those incorporating Extreme Value Theory, advancements in data generation using tools like Sionna RT for environment-specific simulations, the potential for continual learning frameworks leveraging calibration datasets, and dynamic strategies for applying symmetric and asymmetric conformal calibration.

CHAPTER 8. CONCLUSION

Furthermore, the groundwork was laid for implementing a sophisticated actor module, from naive rule-based systems to adaptive reinforcement learning agents, capable of leveraging these uncertainty-aware latency predictions for intelligent interface selection.

In conclusion, this thesis successfully demonstrated that Uncertainty-Aware Conformalized Quantile Regression, particularly with the GMM-QR Pinball model, offers a robust and statistically sound methodology for latency prediction. By providing not just an accurate prediction, surpassing baseline and parametric approaches, but also a calibrated measure of its uncertainty, exemplified by tight median interval widths, 24.27 ms for WiFi, 11.59 ms for BLE, at 90.37% coverage, this work offers a critical tool for developing intelligent interface selection mechanisms. This empowers communication systems to navigate the inherent unpredictability of unlicensed spectrum, transforming potentially unreliable channels into more dependable links for mission-critical applications.

Bibliography

- [1] ArmiaInform, *English: Ukrainian FPV drone with fiber-optic communication channel*, Feb. 2025. [Online]. Available: https://commons.wikimedia.org/wiki/File:UA_fiber-optic_FPV_drone_02.webp (visited on 05/28/2025).
- [2] G. Bianchi, “Performance analysis of the IEEE 802.11 distributed coordination function,” en, *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, Mar. 2000, ISSN: 0733-8716. DOI: [10.1109/49.840210](https://doi.org/10.1109/49.840210). [Online]. Available: <http://ieeexplore.ieee.org/document/840210/> (visited on 06/03/2025).
- [3] D. Sabbagh, “‘They cannot be jammed’: Fibre optic drones pose new threat in Ukraine,” en-GB, *The Guardian*, Apr. 2025, ISSN: 0261-3077. [Online]. Available: <https://www.theguardian.com/world/2025/apr/23/they-cannot-be-jammed-fibre-optic-drones-pose-new-threat-in-ukraine> (visited on 05/27/2025).
- [4] S. B. Kamtam, Q. Lu, F. Bouali, O. C. L. Haas, and S. Birrell, “Network Latency in Teleoperation of Connected and Autonomous Vehicles: A Review of Trends, Challenges, and Mitigation Strategies,” en, *Sensors*, vol. 24, no. 12, p. 3957, Jun. 2024, ISSN: 1424-8220. DOI: [10.3390/s24123957](https://doi.org/10.3390/s24123957). [Online]. Available: <https://www.mdpi.com/1424-8220/24/12/3957> (visited on 05/27/2025).
- [5] P. Popovski, *Wireless Connectivity: An Intuitive and Fundamental Guide*, eng, 1st edition. Newark: Wiley, 2020, ISBN: 978-0-470-68399-6. DOI: [10.1002/9781119114963](https://doi.org/10.1002/9781119114963).
- [6] *Delays in Computer Network*, en-US, Section: Computer Networks. [Online]. Available: <https://www.geeksforgeeks.org/delays-in-computer-network/> (visited on 06/01/2025).
- [7] ERC, *ERC/REC 70-03*, May 2018. [Online]. Available: <https://docdb.cept.org/document/845>.
- [8] *Wi-Fi*, en, Page Version ID: 1292167064, May 2025. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Wi-Fi&oldid=1292167064> (visited on 05/31/2025).
- [9] C. Meraki, *802.11 Association Process Explained*, en, Oct. 2020. [Online]. Available: https://documentation.meraki.com/MR/Wi-Fi_Basics_and_Best_Practices/802.11_Association_Process_Explained (visited on 04/30/2025).
- [10] “IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, Feb. 2021, Conference Name: IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016). DOI: [10.1109/IEEESTD.2021.9363693](https://doi.org/10.1109/IEEESTD.2021.9363693). [Online]. Available: <https://ieeexplore.ieee.org/document/9363693/?arnumber=9363693> (visited on 09/26/2024).
- [11] *Core Specification*, en-US, Aug. 2024. [Online]. Available: <https://www.bluetooth.com/specifications/specs/core-specification-6-0/> (visited on 10/01/2024).
- [12] *Online Power Profiler for Bluetooth LE - opp - Online Power Profiler - Nordic DevZone*, en. [Online]. Available: <https://devzone.nordicsemi.com/power/w/opp/2/online-power-profiler-for-bluetooth-le> (visited on 06/02/2025).
- [13] J. J. Nielsen, R. Liu, and P. Popovski, “Ultra-Reliable Low Latency Communication Using Interface Diversity,” *IEEE Transactions on Communications*, vol. 66, no. 3,

BIBLIOGRAPHY

- pp. 1322–1334, Mar. 2018, Conference Name: IEEE Transactions on Communications, ISSN: 1558-0857. DOI: [10.1109/TCOMM.2017.2771478](https://doi.org/10.1109/TCOMM.2017.2771478). [Online]. Available: <https://ieeexplore.ieee.org/document/8101523> (visited on 09/18/2024).
- [14] *nRF52840 Product Specification*, en. [Online]. Available: https://docs.nordicsemi.com/bundle/ps_nrf52840/page/keyfeatures_html15.html (visited on 11/11/2024).
- [15] *WiFi / ShareTechnote*. [Online]. Available: https://www.sharetechnote.com/html/WLAN_PHY_Frames.html?utm_source=chatgpt.com (visited on 05/06/2025).
- [16] A. Goldsmith, *Wireless Communications*, eng, 1st ed. Cambridge: University Press, 2005, ISBN: 978-0-521-83716-3. DOI: [10.1017/CB09780511841224](https://doi.org/10.1017/CB09780511841224).
- [17] J. Liu, *IEEE 802.11ax Channel Model Document*, English, Sep. 2014. [Online]. Available: <https://mentor.ieee.org/802.11/dcn/14/11-14-0882-04-00ax-tgax-channel-model-document.docx&ved=2ahUKEwjWm7bj5MyLAXylPOHHeyfNVQQFnoECBQQAQ&usg=AOvVaw3o4PLmFdxwCDt3T1LVVD36> (visited on 02/18/2025).
- [18] *Bluetooth LE Blocking, Intermodulation and Carrier-to-Interference Performance Tests*. [Online]. Available: <https://se.mathworks.com/help/bluetooth/ug/bluetooth-le-blocking-intermodulation-and-carrier-to-interference-performance-tests.html> (visited on 12/18/2024).
- [19] *EB/NO Wiki*, en, Page Version ID: 1290091351, May 2025. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Eb/N0&oldid=1290091351> (visited on 06/03/2025).
- [20] Bluetooth SIG, *Understanding Bluetooth Range*, en-US, Apr. 2022. [Online]. Available: <https://www.bluetooth.com/learn-about-bluetooth/key-attributes/range/> (visited on 05/27/2025).
- [21] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback control of dynamic systems*, en, 6. ed. Upper Saddle River, NJ: Pearson, Prentice-Hall, 2010, ISBN: 978-0-13-601969-5.
- [22] M. Haugh, *EM Algorithm*, English, 2015. [Online]. Available: https://www.columbia.edu/~mh2078/MachineLearningORFE/EM_Algorithm.pdf.
- [23] *Information Theory, Inference and Learning Algorithms*, en. [Online]. Available: <https://www.cambridge.org/universitypress/subjects/computer-science/pattern-recognition-and-machine-learning/information-theory-inference-and-learning-algorithms> (visited on 05/27/2025).
- [24] *Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models*. [Online]. Available: https://www.researchgate.net/publication/4249249_Approximating_the_Kullback_Leibler_Divergence_Between_Gaussian_Mixture_Models (visited on 05/02/2025).
- [25] *Standard Normal Table*, 2016. [Online]. Available: <https://math.arizona.edu/~rsims/ma464/standardnormaltable.pdf> (visited on 05/28/2025).
- [26] A. N. Angelopoulos and S. Bates, *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*, en, Jul. 2021. [Online]. Available: <https://arxiv.org/abs/2107.07511v6> (visited on 03/31/2025).
- [27] M. Y. Cheung, T. J. Netherton, L. E. Court, A. Veeraraghavan, and G. Balakrishnan, *Regression Conformal Prediction under Bias*, arXiv:2410.05263 [stat], Oct. 2024. DOI: [10.48550/arXiv.2410.05263](https://doi.org/10.48550/arXiv.2410.05263). [Online]. Available: <http://arxiv.org/abs/2410.05263> (visited on 05/08/2025).
- [28] S. Mostafavi, G. P. Sharma, and J. Gross, *Data-Driven Latency Probability Prediction for Wireless Networks: Focusing on Tail Probabilities*, Version Number: 1, 2023. DOI:

BIBLIOGRAPHY

- 10.48550/ARXIV.2307.10648. [Online]. Available: <https://arxiv.org/abs/2307.10648> (visited on 03/25/2025).
- [29] J. Hoydis, F. A. Aoudia, S. Cammerer, *et al.*, “Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling,” in *2023 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2023, pp. 317–321. DOI: 10.1109/GCWkshps58843.2023.10465179. [Online]. Available: <https://ieeexplore.ieee.org/document/10465179/> (visited on 06/01/2025).
- [30] *OpenStreetMap*, en. [Online]. Available: <https://www.openstreetmap.org/> (visited on 06/01/2025).
- [31] J. Hoydis, F. A. Aoudia, S. Cammerer, *et al.*, “Learning Radio Environments by Differentiable Ray Tracing,” *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 1527–1539, 2024, ISSN: 2831-316X. DOI: 10.1109/TMLCN.2024.3474639. [Online]. Available: <https://ieeexplore.ieee.org/document/10705152/> (visited on 06/01/2025).
- [32] D. Bull and F. Zhang, *David Bull, Fan Zhang - Intelligent Image and Video Compression_ Communicating Pictures (2021, Academic Press)*, English, 2nd. Academic Press, 2021, ISBN: 978-0-12-820353-8.
- [33] *USC Media Communications Lab – MCL-V Database*, en-US. [Online]. Available: <https://mcl.usc.edu/mcl-v-database/> (visited on 12/12/2024).
- [34] C. Courageux-Sudan, A.-C. Orgerie, and M. Quinson, “A Wi-Fi Energy Model for Scalable Simulation,” en, in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Boston, MA, USA: IEEE, Jun. 2023, pp. 232–241, ISBN: 9798350331653. DOI: 10.1109/WoWMoM57956.2023.00038. [Online]. Available: <https://ieeexplore.ieee.org/document/10195804/> (visited on 11/08/2024).
- [35] *Broadband Speed Guide | Federal Communications Commission*. [Online]. Available: <https://www.fcc.gov/consumers/guides/broadband-speed-guide> (visited on 11/07/2024).
- [36] *ESP32-WROOM32 datasheet*. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (visited on 10/10/2024).
- [37] *STM32WLE5JCDatasheet.pdf*. [Online]. Available: <https://files.seeedstudio.com/products/317990687/res/STM32WLE5JC%20Datasheet.pdf> (visited on 11/11/2024).
- [38] *Beacon frames: TIM and DTIM*, en-US. [Online]. Available: <https://academy.nordicsemi.com/courses/wi-fi-fundamentals/lessons/lesson-6-wifi-fundamentals/topic/beacon-frames-tim-and-dtim/> (visited on 04/30/2025).
- [39] *ESP-IDF documentation*. [Online]. Available: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html> (visited on 10/10/2024).

Appendices

A | Modeling of Utility Metrics

A.1 Rate-distortion modeling

Appendix A.1 describes the fundamentals of video compression and rate-distortion modeling. This appendix section is based on the compression scheme Advanced Video Coding or H.264, however the methods used are scheme agnostic. The following section documents how data was collected by applying various levels of compression to the MCL-V dataset, how this process was automated using Python scripts and finally the fitted model along with its residuals and comments about model limitations.

A.1.1 Compression Fundamentals

A simplistic definition of compression is the size ratio between the original video, input, B_i , and the compressed video, output, B_o , which is stored or transmitted.

$$\kappa = \frac{B_i}{B_o} \quad (\text{A.1})$$

This ratio is known as the compression ratio. In typical video compression, typical ratios range from 1:100 to 1:200; however, this can increase to several hundreds or even thousands in modern systems [32]. The compression ratio is an objective measure of how effectively the compression scheme has reduced the size. However, this metric does not account for the resulting quality of the compressed video [32]. Compression is necessary due to the prohibitively high bandwidth requirements of streaming raw video. Even for lower resolution formats like HDTV, the raw bit rate requirement reaches well over 1 Gbps [32] at 30 fps, further increased at higher frame rates and resolution formats.

Video streams consist of a series of still images distributed along a temporal axis. Each still image or frame S is a spatial distribution of sample values referred to as pixels. The pixels of a still image are arranged in a grid structure, lending itself well to matrix notation. Generally an image is described by its pixel matrix with spatial dimensions of $X \times Y$, where each pixel is a vector which varies with the color format. The most commonly used color space is RGB, meaning each pixel is a three-dimensional vector describing intensities of the primary colors (Red, Green, Blue) [32]. As such, a video stream can be mathematically expressed as

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_{0,0,z} & \cdots & \mathbf{s}_{0,X-1,z} \\ \vdots & \ddots & \vdots \\ \mathbf{s}_{Y-1,0,z} & \cdots & \mathbf{s}_{Y-1,X-1,z} \end{bmatrix}, \quad (\text{A.2})$$

where each pixel (for an RGB video)

$$\mathbf{s}[x, y, z] = [s_R, s_G, s_B]$$

An overview of video compression is showcased by Figure A.1. Here the input video S is subject to some encoding function f , which produces a compressed representation of the input data to be reconstructed by the decoder g .

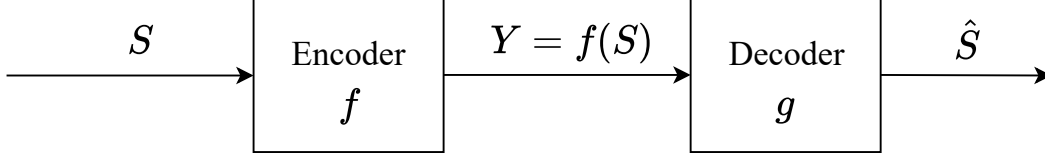


Figure A.1: Overview of video compression flow.

When using lossy compression, the reconstructed video \hat{S} , will differ from the original input due to distortion introduced by the encoding process. Various metrics, such as mean squared error (MSE), signal-to-noise ratio (SNR) or structural-similarity index measure (SSIM) can be used to evaluate the performance of an encoding scheme.

A.1.2 Evaluating a Compression Scheme

Evaluation metrics for compression schemes are typically split into objective and perceptual metrics, with MSE and SNR being objective and SSIM being perceptual [32].

The MSE between two frames is computed as the mean squared reconstruction error between each individual pixel, expressed by Equation (A.3).

$$\text{MSE} = \frac{1}{XY} \left(\sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} (\mathbf{s}_{x,y} - \hat{\mathbf{s}}_{x,y})^2 \right) \quad (\text{A.3})$$

Alternatively, the compression can be evaluated based on the SNR, which compares the power of each original frame, to the power introduced by reconstruction error as noise (the MSE), expressed by Equation (A.4)

$$\text{SNR} = 10 \cdot \log \left(\frac{\frac{1}{XY} \sum_{x,y} (\mathbf{s}_{x,y} - \mu_s)^2}{\frac{1}{XY} \sum_{x,y} (\mathbf{s}_{x,y} - \hat{\mathbf{s}}_{x,y})^2} \right) \quad (\text{A.4})$$

MSE-based metrics are favored when evaluating the objective performance of a compression scheme, due to their ease of calculation. Specifically, the peak SNR (PSNR) in Equation (A.5), has been widely adopted to characterize the image quality of the reconstructed video stream [32].

$$\text{PSNR} = 10 \cdot \log \left(\frac{s_{\max}^2}{\frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} (\mathbf{s}_{x,y} - \hat{\mathbf{s}}_{x,y})^2} \right) \quad (\text{A.5})$$

where $s_{\max} = 2^B - 1$ is the maximum value a pixel intensity can take, defined by the word length, also known as bit depth, B . Although there is no perceptual basis for PSNR, it does align well with subjective assessments. The intuition that a higher PSNR leads to better video quality holds, and generally PSNR values below 20 dB are considered unacceptable, while values in the range 30 to 40 dB are considered very good [32]. Note that the expression for PSNR in Equation (A.5) is for computing the PSNR of a single frame, only considering a single color channel. If the color space has multiple channels, such as RGB, the MSE term becomes:

$$\text{MSE} = \frac{1}{3XY} \left(\sum_{RGB} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} (\mathbf{s}_{x,y} - \hat{\mathbf{s}}_{x,y})^2 \right)$$

However, in luminance-chroma based color spaces such as YUV, it is common to compute the PSNR using only the luminance channel, as this predominantly determines the perceptual quality of the video [32].

A.1.3 Empirical Modeling of H.264 Rate-Distortion Performance

In this section a mathematical model is fitted to empirical rate-distortion data from the MCL-V raw video dataset. The goal is to find a suitable model, which approximates the encoding function f from Figure A.1, used in H.264 compression. Note that rate-distortion and rate-quality are used interchangeably, since we only consider objective quality measures (MSE and PSNR), and distortion simply describes a degradation in quality.

Empirical data is collected from the MCL-V dataset by University of Southern California [33]. The dataset consists of 12 raw videos in the YUV format, showcased by Figure A.2.

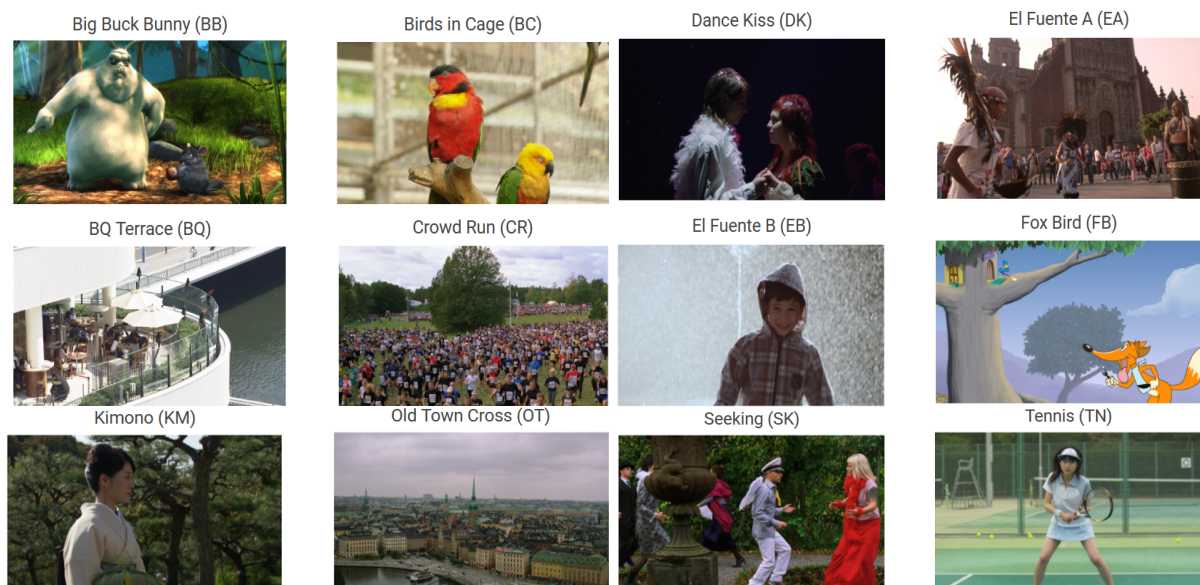


Figure A.2: Sample videos from MCL-V dataset [33].

Each raw video is compressed using FFmpeg, which is an open source multimedia framework, providing encoder and decoder functionalities for a large variety of implemented compression schemes. Each video from the MCL-V dataset is compressed with 11 different target bitrates producing a subset of videos with the following bitrates:

$$\text{target bitrate} = \{10 \cdot 2^{2+i} | i = 0, 1, \dots, 10\} \text{ [kbps]}$$

The process of compressing videos is automated using a Python script, which repeatedly constructs the appropriate bash command to run FFmpeg with appropriate flags as showcased below.

```

1 def compress_video(input_file, output_file, target_bitrate):
2     cmd = [
3         'ffmpeg',
4         '-loglevel', 'error',
5         '-i', input_file,
6         '-c:v', 'libx264',
7         '-b:v', target_bitrate + 'k',

```

APPENDIX A. MODELING OF UTILITY METRICS

```

8         output_file
9     ]
10    print(cmd)
11    subprocess.run(cmd)

```

The generated dataset consists of $12 \cdot 11 = 132$ videos of varying quality. The distortion of each video is assessed using MSE and PSNR, comparing each subset to its respective original raw video. FFmpeg also provides convenience functions for computing the MSE and PSNR between two videos using Equation (A.3) and Equation (A.5). The process of comparing each compressed video to its baseline is automated using the code snippet below, again constructing the FFmpeg bash command with appropriate flags:

```

1 def log_psnr(baselines):
2     for baseline in baselines:
3         #Find all compressed videos for this baseline
4         compressed = glob.glob(f'MCL-V/{os.path.basename(baseline.split(".mp4")
5                                [0])}/*k.mp4', recursive=True)
6         compressed = [file.replace('\\', '/') for file in compressed]
7         for comp in compressed:
8             psnr_log = comp.replace('.mp4', '_psnr.log').replace('\\', '/')
9             #Construct bash command with flags
10            cmd = [
11                'ffmpeg',
12                '-loglevel', 'error',
13                '-i', baseline,
14                '-i', comp.replace('\\', '/'),
15                '-lavfi', f"psnr=stats_file={psnr_log}",
16                '-f', 'null', '-'
17            ]
18            print(f'Comparing {baseline} and {comp}, saving to {psnr_log}')
19            subprocess.run(cmd)

```

An excerpt of the resulting logs is showcased below:

```

1 n:1 mse_avg:9.71 mse_y:12.25 mse_u:5.38 mse_v:3.85 psnr_avg:38.26 psnr_y:37.25
   psnr_u:40.82 psnr_v:42.27
2 n:2 mse_avg:9.72 mse_y:12.24 mse_u:5.51 mse_v:3.84 psnr_avg:38.25 psnr_y:37.25
   psnr_u:40.72 psnr_v:42.28
3 n:3 mse_avg:9.78 mse_y:12.34 mse_u:5.52 mse_v:3.85 psnr_avg:38.23 psnr_y:37.22
   psnr_u:40.71 psnr_v:42.28

```

As previously mentioned, when using a luminance-chroma based color space such as YUV, the PSNR can be evaluated solely from the luma channel. As such, the `psnr_y` column is parsed for each log file, and the result is stored in a dictionary containing the video bitrate and corresponding PSNR values. The PSNR is evaluated as a function of compression ratio and the result is shown by Figure A.3. Note that the compression ratio is given by a modification of Equation (A.1), using bit rates rather than file size and inverting the fraction.

APPENDIX A. MODELING OF UTILITY METRICS

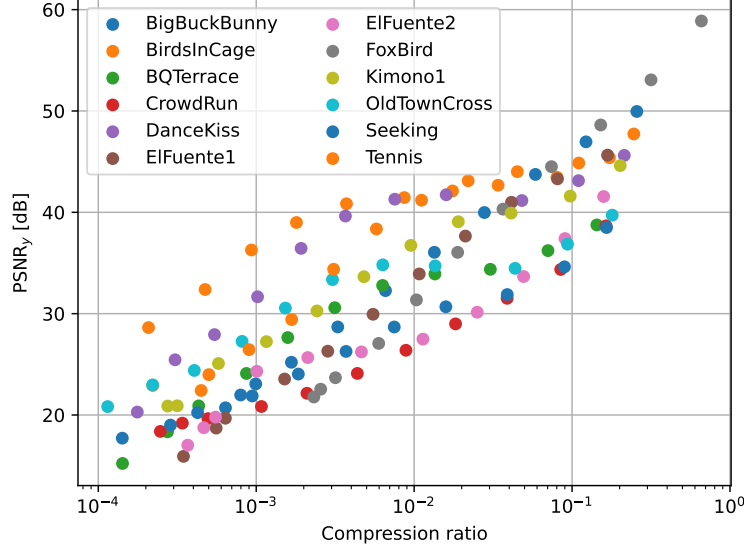


Figure A.3: Plot of luma PSNR as a function of compression ratio.

Key observations from Figure A.3 are, the logarithmic relationship between PSNR and compression ratio, as well as, the difference in PSNR between types of videos. This indicates the compression does not affect the quality of all videos equally. The difference is likely explained by the degree of movement in each video, which is why a large disparity is seen between *BirdsInCage* and *CrowdRun*.

Given the data from Figure A.3, various mathematical functions were fitted and the best fit was observed with a logarithmic function described by Equation (A.6).

$$\text{PSNR}_y(\kappa) = \alpha \ln(\kappa) + \beta, \quad (\text{A.6})$$

where α and β are tunable parameters. The model achieved an R-squared value of 75%, with fitting parameters $\alpha = 3.62$ and $\beta = 50.54$. The fitted model, as well as, residuals are showcased by Figure A.4 and Figure A.5 respectively.

APPENDIX A. MODELING OF UTILITY METRICS

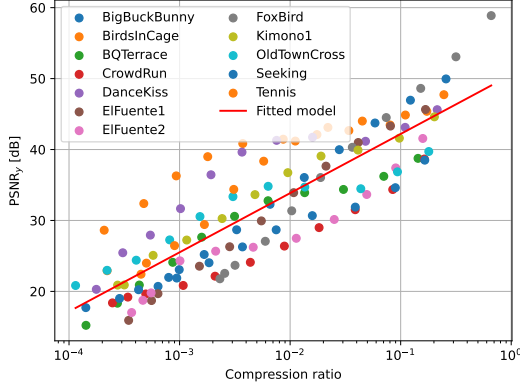


Figure A.4: Plot of fitted model $\alpha \ln(\kappa) + \beta$ with corresponding data points.

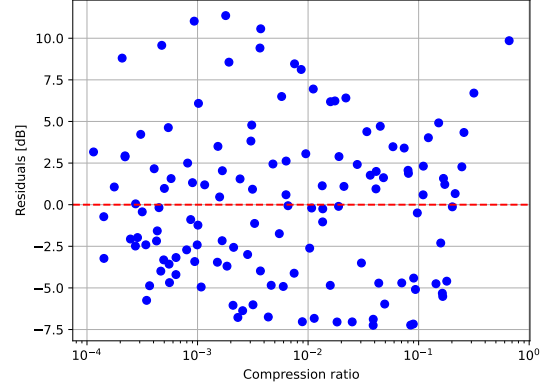


Figure A.5: Plot of residuals for model fit.

From Figure A.4 and Figure A.5, it is clear that the model captures the general relationship between distortion and compression ratio. However the model accuracy seems to vary significantly depending on the input video. This is likely due to the degree of movement or motion in the video. As such, the model would benefit from accounting for the type of video, or being fit to a more homogeneous dataset. Additionally, alternative function approximation methods such as neural network based approaches may be explored given their efficacy, as demonstrated to be effective by the universal approximation theorem. Despite these limitations the model is deemed suitable for the scope of this project and provides the necessary framework for rate-distortion analysis.

A.2 Energy Consumption

Considering the mission-critical application scenario described in Chapter 1, energy efficiency becomes essential, as it directly impacts a device battery life and operational efficiency. Energy efficiency in wireless communication proves challenging, when simultaneously prioritizing a high reliability, as measures to improve reliability often include transmitting over multiple interfaces or simply increasing the transmission power, resulting in higher energy consumption.

When considering a multi-interface device with diverse RANs like WiFi, Bluetooth and LoRa, modeling the energy consumption becomes quite complex. This is largely due to interface-specific setup procedures and varying amounts of overhead, each impacting the energy consumption differently. In this section a general power consumption model is constructed, which is applicable to all the selected interfaces. The model focuses solely on the energy consumed by transmitting raw data bits, omitting the overhead, handshake and other interface specific procedures. Thereby focusing on the cost of basic transmission, without any auxiliary processes.

A common framework for modeling the power usage of a WiFi interface is constructing a state machine and assigning a power cost to each state [34]. Since we are focusing solely on the cost of data transmission, this framework extends to the other RANs. An example of a state machine for an arbitrary wireless communication interface is showcased by Figure A.6.

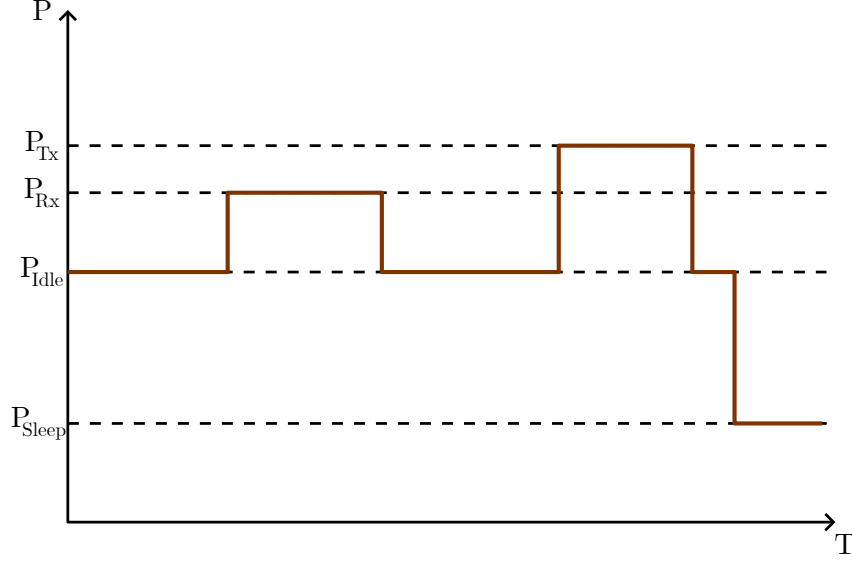


Figure A.6: State machine diagram of arbitrary wireless communication interface.

The state machine in Figure A.6, defines the following four states [34]:

- *Idle* - The interface does not perform any operation, the power consumption is purely from powering the device and keeping it operational.
- *Tx* - The device is actively transmitting data, meaning the the transmitter circuit is powered, thereby increasing the power consumption.
- *Rx* - The device is listening to the wireless channel and receiving data.
- *Sleep* - The device performs no operation, and has deactivated parts of its circuitry in order to reduce power consumption at the cost of having to wake up upon switching state.

Each state has an assigned power consumption $\{P_{Tx}, P_{Rx}, P_{Idle}, P_{Sleep}\}$, which is typically determined empirically, or found in the documentation of a specific interface chip. The energy consumption in a time period T is given by Equation (A.7).

$$E(T) = \int_0^T P(t)dt, \quad (\text{A.7})$$

where T is the time period in seconds and $P(t)$ is the power consumption in watts of any state in time t [34]. This project handles a scenario, in which the devices are only communicating one way to a central receiver, at which power consumption is of no concern. As such, the set of states is simplified to just $\{P_{Tx}, P_{Idle}\}$. In order to establish a flexible model, in which RANs can switch states independently once they have finished their share of the transmission load, the power consumption is evaluated on a 1-second interval. The total power consumption is given by the proportion of time spent in each state and the cost of being in said state as described by Equation (A.8).

$$P_{tot} = \sum_{j=1}^R \bar{P}_j = \sum_{j=1}^R T_{Tx,j} \cdot P_{Tx,j} + T_{Idle,j} \cdot P_{Idle,j}, \quad (\text{A.8})$$

where \bar{P}_j is the time-averaged power of RAN j over the 1-second interval, T_{Tx}, T_{Idle} is the time spend in the *Tx* and *Idle* state respectively and P_{Tx}, P_{Idle} is the power consumption of the corresponding state. Accounting for the fact that a device may distribute its data across multiple

APPENDIX A. MODELING OF UTILITY METRICS

RANs, we compute the time spent in each state based on the amount of bits the device wishes to transmit and the data rate of the chosen RAN.

$$T_{Tx} = \frac{P_j \cdot B_s \cdot \kappa_i}{r_j}$$

$$T_{Idle} = 1 - T_{Tx} = 1 - \frac{P_j \cdot B_s \cdot \kappa_i}{r_j} \quad (\text{A.9})$$

Equation (A.9) essentially describes the fraction of the 1-second interval, in which the RAN is in either its *Tx* or *Idle* state. The amount of data to transmit is determined by the fraction $P_{i,j}$ of the total amount of data B_s after a compression scheme with compression ratio κ_i has been applied. Substituting Equation (A.9) into Equation (A.8) yields the final power consumption model Equation (A.10).

$$P_{tot} = \sum_{j=1}^R \left(\frac{P_j \cdot B_s \cdot \kappa_i}{r_j} \cdot P_{Tx} + \left(1 - \frac{P_j \cdot B_s \cdot \kappa_i}{r_j} \right) \cdot P_{Idle} \right), \quad (\text{A.10})$$

where r_j is the data rate of RAN j , B_s is the bandwidth requirement of the video stream (from Table A.1) and P_{Tx}, P_{Idle} are the power costs of being in the *Tx* and *Idle* states respectively. The power consumption in each state is typically documented in the datasheet of the individual RAN module. For this project an ESP32 will serve as an example of a WiFi module, a NRF52840-Dongle as a Bluetooth module and a WIO-e5-mini as a LoRa module. The power consumption for all of these devices are documented in their datasheet. Generally the datasheet specifies current consumption I in Tx active or idle mode, which can be converted to a power consumption by multiplying by the specified voltage U (typically 3.3 V). The power consumption of the chosen RAN modules is summarized by Table A.2.

Video quality	Bandwidth requirement
4k	25 Mbps
1080p	8 Mbps
720p	5 Mbps
480p	3 Mbps

Table A.1: FCC bandwidth guidelines for video streaming [35].

RAN	Device	Power consumption
WiFi	ESP32	Tx: 627 mW Idle: 66 mW
Bluetooth	NRF52840-DONGLE	Tx: 54.12 mW Idle: 10.89 mW
LoRa	WIO-e5-mini	Tx: 396 mW Idle: 3.47 mW

Table A.2: State power consumption for selected RAN devices [36][14][37].

B | Available Data in Real World Systems

This appendix investigates and documents measures that can be used to describe the congestion of the channel and the the propagation state of the channel. The measures must be available even when a interface is not in use for video transmission. This appendix is divided into the following sections, Channel Congestion and Channel states, describing metrics used to indicate the number of devices sharing the medium and the current channel state information respectively.

B.1 Channel Congestion

In this section we investigate what data is available to describe the channel congestion in the environment for WiFi and BLE. For this, two methods are discussed, a Protocol Agnostic Method and a Protocol Specific Methods.

B.1.1 Protocol Agnostic Method

The protocol agnostic approach to determine the congestion in the environment is to measure the power density in the environment. Figure B.1 shows a spectrogram captured in the CNT lab on a Rohde & Schwarz FPL 1007 spectrum analyzer, observing the power in each of the BLE Sub Channels over time. In practice this can be implemented as a bank of band-pass filters, or a fourier transform. The spectrogram has historically been used in conjunction with image classification models, typically utilizing convolutional neural networks or transformers, a model may be trained to classify the network congestion from a spectrogram of the current channel.

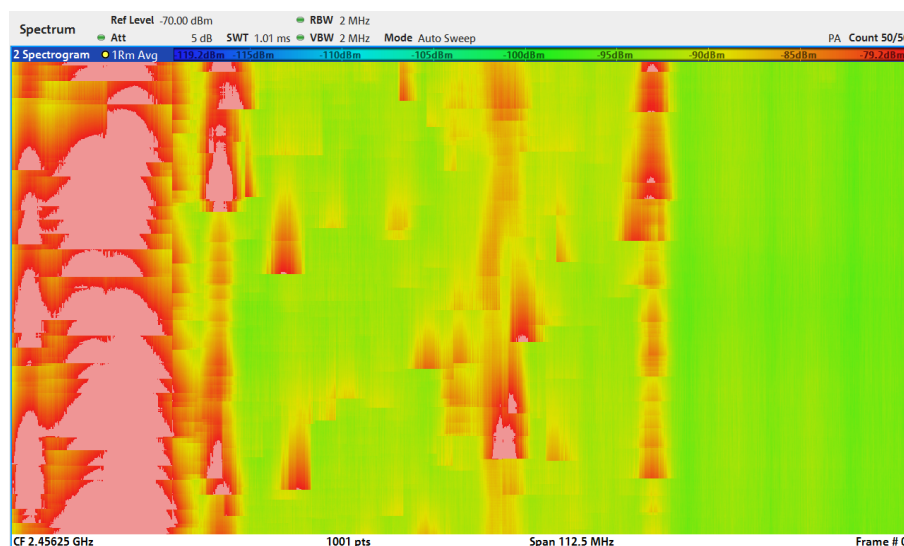


Figure B.1: Waterfall spectrogram over approximately 4 seconds.

B.1.2 Protocol Specific Methods

The protocol specific methods to determine the congestion in the environment are based on data transmitted as part of the protocol. Two types of methods have been considered, one that uses the protocol data from the connection in the device own ongoing transmission and another

APPENDIX B. AVAILABLE DATA IN REAL WORLD SYSTEMS

method where data is collected from other networks operating in the environment through a device in promiscuous mode.

WiFi

In the investigation of WiFi protocol, three frame types, beacon, probe request and response, where found relevant to investigate. To investigate these, Wireshark is used with an ALFA NETWORK AWUS036EW set to promiscuous mode as seen in Listing B.1.

```
1 #!/bin/bash
2 sudo ifconfig wlan1 down
3 sudo iwconfig wlan1 mode monitor
4 sudo ifconfig wlan1 up
5 sudo wireshark -i wlan1
```

Listing B.1: WiFi Bash Script for collecting enviroment data

Inspecting the data accessible in promiscuous, showcased in Table B.1 in Wireshark, we can see the different types of messages: beacon, probe request and probe response frames.

802.11 Frame Capture Analysis					
No.	Source	Destination	Message Type	Power (dBm)	Channel (MHz)
1	TpLinkTechno_a4:dc:90	Beacon frame	-16	2412	
3	Cisco_09:51:26	Broadcast	Beacon frame	-31	2412
5	TpLinkTechno_a4:dc:90	Broadcast	Beacon frame	-16	2412
6	Cisco_09:51:25	Broadcast	Beacon frame	-31	2412
7	TpLinkTechno_a4:dc:90	Broadcast	Beacon frame	-16	2412
8	RaspberryPiT_54:ac:e6	Broadcast	Probe Request	-34	2412
9	RaspberryPiT_54:ac:e6	Broadcast	Probe Request	-33	2412
10	TpLinkTechno_a4:dc:90	Broadcast	Beacon frame	-16	2412
11	Cisco_09:51:27	Broadcast	Beacon frame	-31	2412
12	RaspberryPiT_54:ac:e6	Broadcast	Probe Request	-37	2412
13	RaspberryPiT_54:ac:e6	Broadcast	Probe Request	-37	2412
14	Cisco_09:51:21	Broadcast	Beacon frame	-31	2412
17	TpLinkTechno_a4:dc:90	Broadcast	Beacon frame	-16	2412
21	Cisco_09:51:29	Broadcast	Beacon frame	-31	2412
23	TpLinkTechno_a4:dc:90	Broadcast	Beacon frame	-16	2412
42	TpLinkTechno_a4:dc:90	4e:c7:58:b8:34:63	Probe Response	-16	2412
43	TpLinkTechno_a4:dc:90	f2:51:ff:f5:ce:bd	Probe Response	-16	2412
48	TpLinkTechno_a4:dc:90	Beacon frame	-16	2412	

Table B.1: Analysis of 802.11 protocol information in the area using a ALFA NETWORK AWUS036EW set to promiscuous mode.

The function of the three frames are:

Beacon frame

The beacon frame serve to announce that a WLAN is pressent, and to broadcast key network information to connected devices [38]. The beacon frame is scheduled periodically by a AP, with a interval given by the Beacon interval parameter often set to 100 TU = 102.4 ms. The actual transmission interval varies due to CSMA/CA. Some of the key network information in the beacon frame is:

- SSID
- Supported data rates

APPENDIX B. AVAILABLE DATA IN REAL WORLD SYSTEMS

- Operational channel numbers
- Existence of buffered multicast or unicast messages

The SSID can be used to identify a device herby give an idea of the number of devices in the area.

Probe Request and response frame

Probe request is used as a active method for connection establishments by stations. The probe request is sent to discover a AP, AP receiving it will answer with SSID and other 802.11 capabilities [9]. It's common that station ask for a specific SSID. In other words, stations might be leaking names of network it has been connected to in the past. The probe request can be used to estimate the number of stations and access point in the area.

BLE

Based on a investigation into the BLE protocol, two part of the protocol is fund found to have potential to indite congestion level in the environment. These two parts are the BLE map, and advertisement in channel 37 to 39.

Advertisement Channel

The advertisement channel are used for connection establishment in BLE. There is three primary and 37 secondary advertisement channels. Peripherals can perform many types of advertisement. Using an nRF52840 Dongle programmed as a BLE sniffer the advertisement channels have been sniffed, seen in Table B.2.

BLE Packet Capture Analysis					
No.	Source	Protocol	RSSI (dBm)	Channel	Info
1	26:2b:6a:40:22:7b	LE LL	-56	37	ADV_NONCONN_IND
2	26:2b:6a:40:22:7b	LE LL	-46	38	ADV_NONCONN_IND
3	26:2b:6a:40:22:7b	LE LL	-51	39	ADV_NONCONN_IND
7	6b:c9:f4:c5:da:82	LE LL	-52	37	ADV_IND
11	60:de:86:f7:b9:47	LE LL	-69	38	ADV_IND
13	45:bd:99:a1:23:b7	LE LL	-62	37	ADV_IND
18	41:30:65:9a:74:b2	LE LL	-62	39	SCAN_REQ
19	c5:2c:48:e4:9a:bb	LE LL	-60	37	ADV_IND
23	4f:d1:46:0a:88:03	LE LL	-53	37	ADV_IND
33	5f:9d:16:7e:8f:f1	LE LL	-49	37	ADV_SCAN_IND
43	ca:45:89:96:cd:1c	LE LL	-37	37	ADV_IND
53	1f:f6:1a:aa:88:0a	LE LL	-35	38	ADV_NONCONN_IND
54	1f:f6:1a:aa:88:0a	LE LL	-32	39	ADV_NONCONN_IND
67	45:bd:99:a1:23:b7	LE LL	-63	38	SCAN_RSP
90	1f:f6:1a:aa:88:0a	LE LL	-32	39	ADV_NONCONN_IND
97	5f:9d:16:7e:8f:f1	LE LL	-49	37	ADV_SCAN_IND
1295	58:da:f4:77:bc:46	LE LL	-79	39	ADV_DIRECT_IND

Table B.2: Selected BLE packet from WiFi.

Multiple of the types of advertisement is seen in the Info coulomb of Table B.2 and is summarized in Table B.3. ADV_IND is the most commom. A peripheral uses this type of advertising, to advertise its presence and indicate it will answer a centrals scan request with a scan response,

which is followed by establishing a connection. `ADV_NONCONN_IND`, is used as a beacon as it does not allow central to request a scan or connect. `ADV_SCAN_IND`, will only allow for scan and not connection. `ADV_DIRECT_IND` does not allow for scan but only connections

BLE Frame Type Distribution			
Frame Type	Count	Percentage	Avg. RSSI
<code>ADV_IND</code>	58	39.2%	-65.4 dBm
<code>ADV_NONCONN_IND</code>	52	35.1%	-59.6 dBm
<code>SCAN_REQ</code>	18	12.2%	-69.4 dBm
<code>ADV_SCAN_IND</code>	9	6.1%	-51.8 dBm
<code>SCAN_RSP</code>	2	1.4%	-59.0 dBm
<code>ADV_DIRECT_IND</code>	1	0.7%	-59.0 dBm

Table B.3: BLE Message Type Analysis over 75 ms

Sniffing the BLE channel, the Address and RSSI is also available. The address of BLE devices is either Public, meaning you pay to register it with IEEE, or random address. Random address can both be random static, only allow change of address at boot up or random private changes address periodically.

Channel Map

The Channel map is the part of BLE adaptive frequency hopping specifying which channels are used for transmission. The Channel map is determined by the central and transmitted to the peripheral. The channel map is based on the classification of channels. The core specification define bad channels as [11]:

- A channel may be classified as bad if an ACL or synchronous throughput failure measure associated with it has exceeded a threshold (defined by the particular channel assessment algorithm employed).
- A channel may also be classified as bad if an interference-level measure associated with it, determining the interference level that the link poses upon other systems in the vicinity, has exceeded a threshold (defined by the particular channel assessment algorithm employed).
- A channel may also be classified as bad if an interference-level measure associated with it, determining the interference level that the link poses upon other systems in the vicinity, has exceeded a threshold (defined by the particular channel assessment algorithm employed).

A channel can be classified as bad either from local measurements, Channel classification information from the Host over the Host Controller interface (HCI), or channel classification reports received from peripherals in Link Management Protocol (LMP) CHANNEL [11]. The algorithm used to determined if a Channel is considered bad is implementation specific.

As the channel map directly indicate the number of that can be used, it will also give a indication of what latency can be expected.

B.2 Channel state

In this section we investigate what channel information are available in a established connection. Both WiFi and BLE chip set provide information about the channel state, either as a Channel state Information (CSI) for WiFi or Received Signal Strength Indicator RSSI for BLE. This

investigation is based on developments boards with a ESP32-S3 chip for WiFi and NRF52840 chip for BLE.

B.2.1 Channel state Information - ESP32-S3

The CSI is available for a established WiFi connection. The ESP32-S3 chipset make the CSI available, though the following steps and API calls [39]:

- Select Wi-Fi CSI in menuconfig. Go to Menuconfig > Components config > Wi-Fi > Wi-Fi CSI (Channel State Information).
- Set CSI receiving callback function: `esp_wifi_set_csi_rx_cb()`.
- Configure CSI: `esp_wifi_set_csi_config()`.
- Enable CSI: `esp_wifi_set_csi()`.

The CSI data is stored in a buffer with each item stored as two bytes: imaginary part followed by the real part. In Table B.4 are some of the CSI data collected in a test on the ESP-32-S3.

BLE Packet Capture Analysis						
id.	mac	channel	rssi	noise_floor	local_timestamp	CSI data
58	40:4c:ca:51:52:ec	11	-50	-96	158010101	CSI_data_1
58	40:4c:ca:51:52:ec	11	-49	-96	158106207	CSI_data_2

Table B.4: WiFi Connection data.

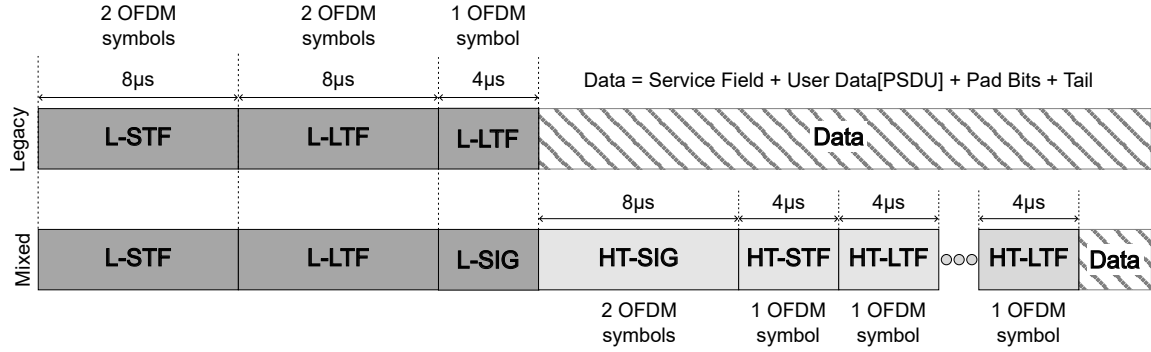
A example CSI data is shown in Table B.5.

CSI_data_1				
Sub carrier	Type	IMG	Real	
0	L-LTF	-85	-80	
1	L-LTF	10	0	
	⋮			
63	L-LTF	-16	-15	
0	HT-LTF	-16	-9	
1	HT-LTF	-12	-25	
	⋮			
63	HT-LTF	-23	-32	

Table B.5: Example of CSI data.

The Legacy Long Training Field (L-LTF) and the High Throughput Long Training Field (HT-LTF) are part of the preamble, illustrated in Figure B.2. The L-LTF is a 8-microsecond field for fine-tuning frequency offset and channel estimation for legacy devices. The HT-LTF are used for fine channel estimation, MIMO channel calibration, and sounding. The number of HT-LTFs sent depends on the number of spatial streams being used.

APPENDIX B. AVAILABLE DATA IN REAL WORLD SYSTEMS



As a input to the Neural network, on can give the all the channels estimations. The input space can be shrunk by calculating the magnitude os each subcarrier. or further shrunk by calculating a SNR using Equation (B.1)

$$SNR = \frac{\sum_{i=0}^{63} 10 \cdot \log_{10} (Tx \cdot |h_i|)}{Noise_floor} \quad (B.1)$$

B.2.2 RSSI - NRF52840

The NRF52840 chipset have a ADC measuring the voltage just before the demodulator. NRF52840 measured the RSSI continuously and filters it using a single-pole IIR filter [14]. After a signal level change, the RSSI will settle after approximately $15 \mu s$. The practical implementation of RSSI measurement is configured by enabling the `RSSISTART` and hereafter the sample can be read from the `RSSISAMPLE` register [14]. The RSSI have a accuracy of ± 2 db.

C | Latency Predictor Results

This appendix provides supplementary figures supporting the evaluation of the latency predictor models evaluated in Chapter 5. Throughout this appendix hyperparameter sweep results and quantile error distribution plots are presented for models which were deferred from the main report chapter.

C.1 Hyperparameter Sweep Results

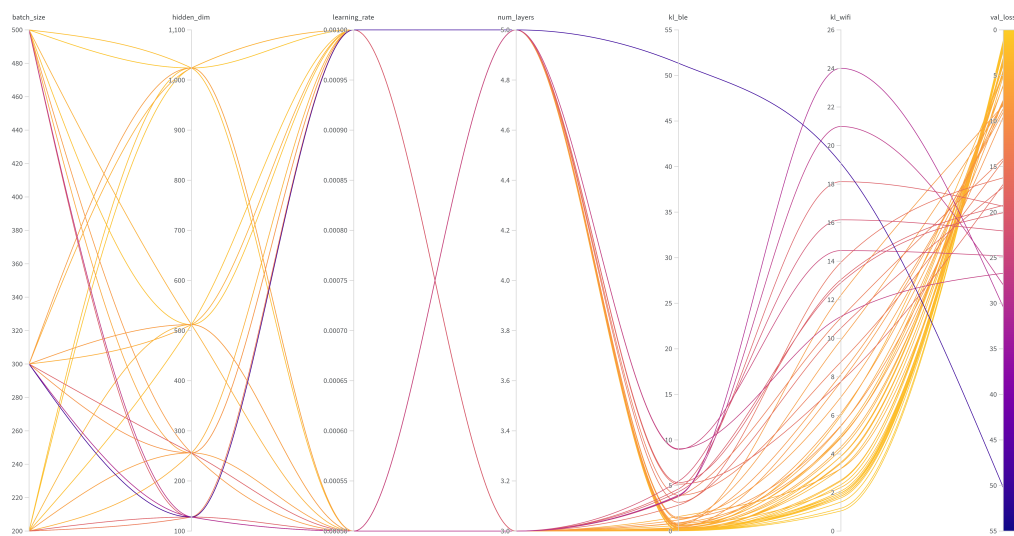


Figure C.1: Hyperparameter sweep of the **GMM-Param** latency predictor model.

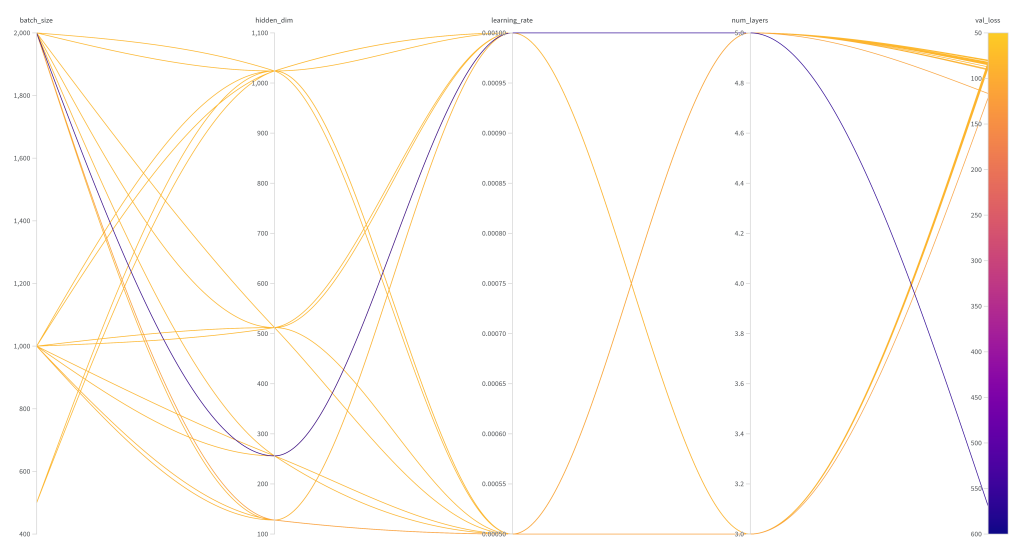


Figure C.2: Hyperparameter sweep of the **MV-QR** latency predictor model.

APPENDIX C. LATENCY PREDICTOR RESULTS

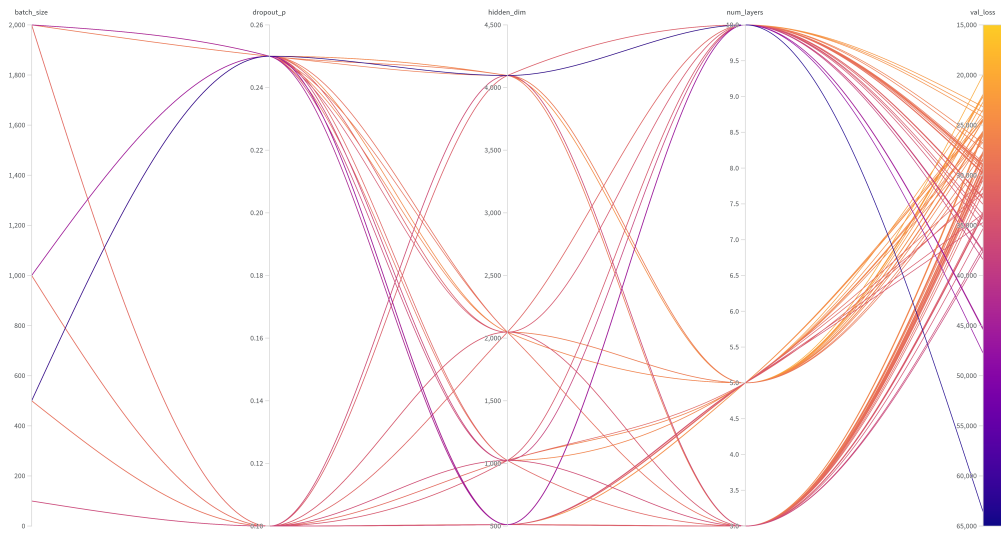


Figure C.3: Hyperparameter sweep of the **GMM-QR MSE** latency predictor model.

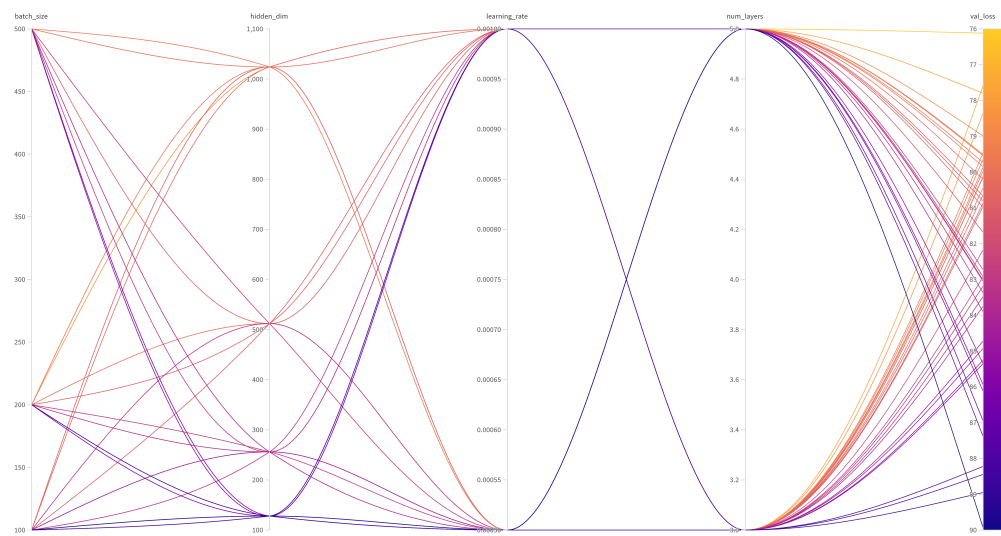


Figure C.4: Hyperparameter sweep of the **GMM-QR Pinball** latency predictor model.

C.2 Model Evaluation

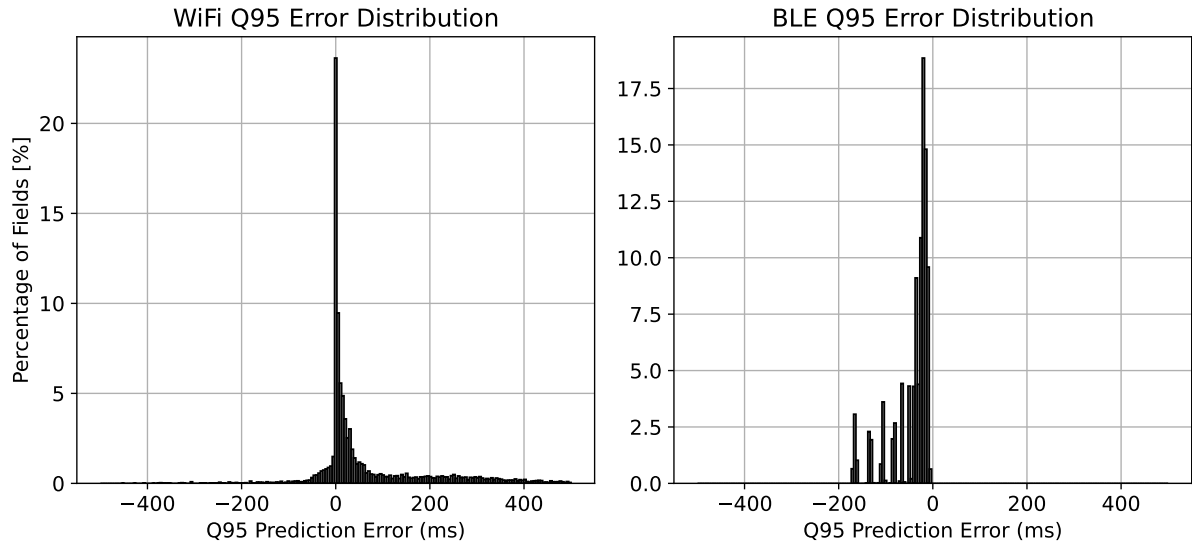


Figure C.5: 95th quantile prediction error distribution for the **GMM-Param** model.

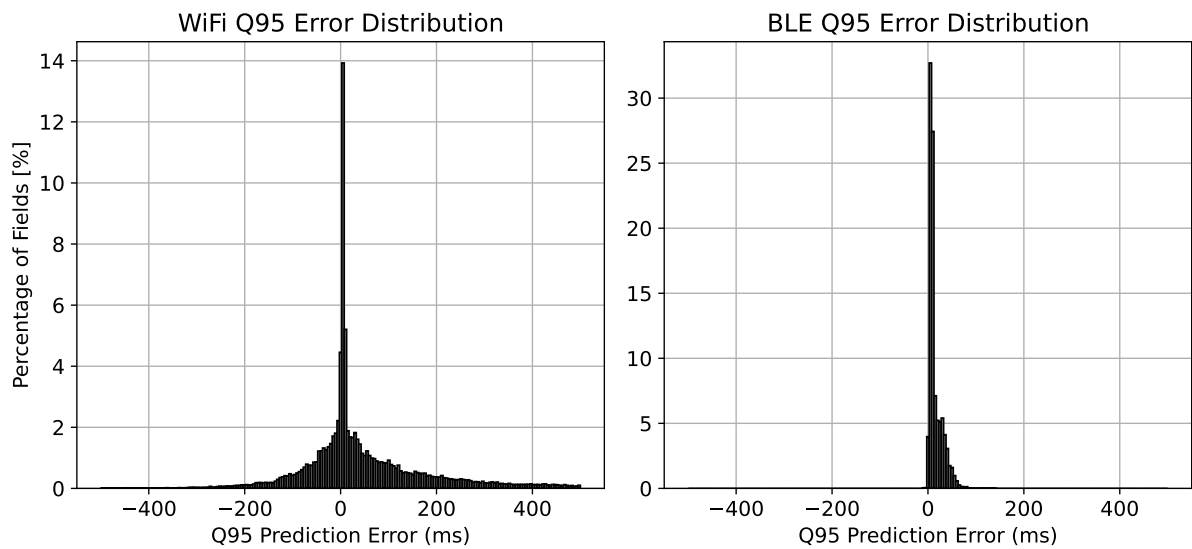


Figure C.6: 95th quantile prediction error distribution for the **MV-QR** model.

D | Conformal Prediction Coverage Analysis

This appendix provides supplementary figures, supporting the coverage analysis of the conformal prediction evaluated in Chapter 6. As mentioned in the main report, the coverage is evaluated for all the trained models and the results are shown through this appendix. Note that the coverage of the GMM-param model exhibit significantly more undercoverage than its counterpart parametric model. This is due to an error in the evaluation, in which the coverage was not evaluated over many realizations of the calibration and evaluation data split. The full evaluation of this model was not conducted due to its excessive computation time.

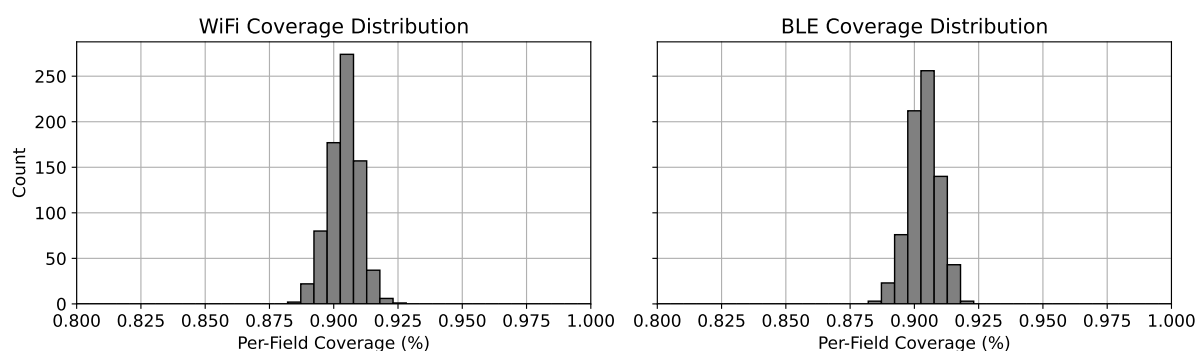


Figure D.1: Coverage distribution for WiFi and BLE for the **GMM-QR** latency predictor model.

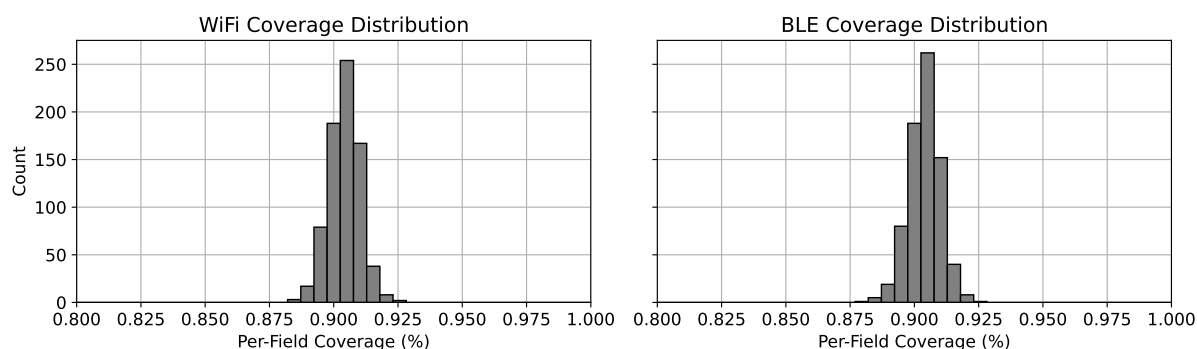


Figure D.2: Coverage distribution for WiFi and BLE for the **GMM-MSE** latency predictor model.

APPENDIX D. CONFORMAL PREDICTION COVERAGE ANALYSIS

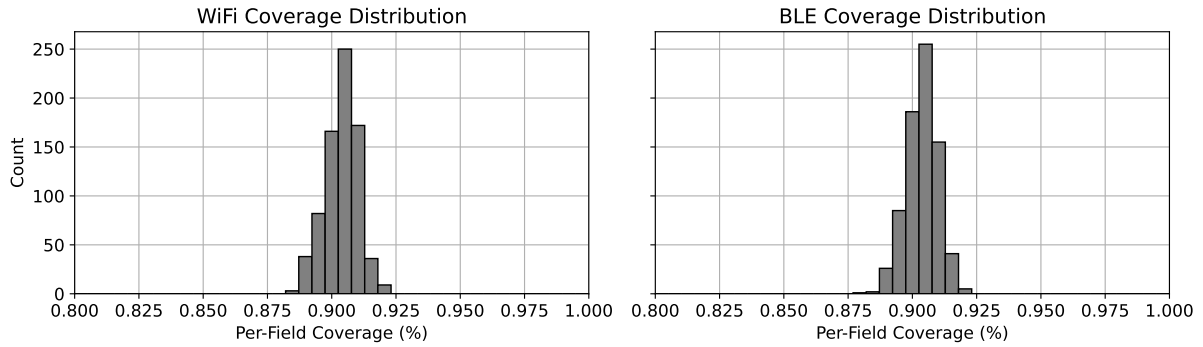


Figure D.3: Coverage distribution for WiFi and BLE for the **MV-Param** latency predictor model.

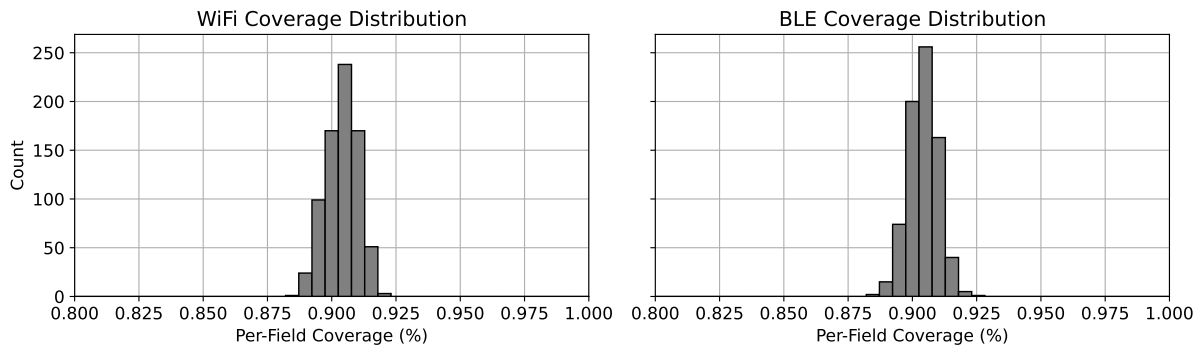


Figure D.4: Coverage distribution for WiFi and BLE for the **MV-QR** latency predictor model.

From the above coverage distributions, it can be concluded that all models converge to the target $1 - \alpha = 0.9$, serving as a sanity check that the conformal prediction is implemented correctly, satisfying its marginal coverage guarantee.