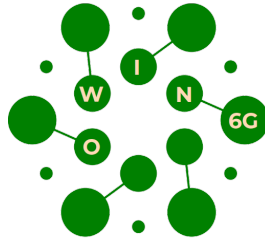


HORIZON-MSCA-2022-DN-01

(Marie Skłodowska-Curie Actions Doctoral Networks)



Project Number: 101119624

Acronym: OWIN6G

Project title: Optical and Wireless sensors Networks for 6G scenarios

Work Package 2: Physical and MAC Layer

Deliverable D2.3 (D8): MAC protocol designs

MaxLinear, Inc - Valencia, Spain

April 2026

Authors

Name	Organization	e-mail
Francisco Rau	MaxLinear, Inc	frau@maxlinear.com
Carlos Herranz	MaxLinear, Inc	cherranz@maxlinear.com
Iñaki Val	MaxLinear, Inc	ival@maxlinear.com

Executive Summary

This deliverable presents the technical outcomes of Task 2.3 within the OWIN6G project, addressing the design of MAC layer scheduling algorithms for IEEE 802.11 wireless sensor networks. Two complementary artificial-intelligence strategies have been investigated and validated through extensive NS-3 simulations. The first strategy employs a machine learning-based traffic flow classifier that dynamically identifies application types and maps them to the appropriate IEEE 802.11 EDCA access categories, achieving significant latency reductions for time-sensitive services under congestion. The second strategy applies reinforcement learning to the OFDMA multi-user scheduling problem introduced in IEEE 802.11ax, optimising resource allocation across multiple stations. Both strategies demonstrate the feasibility and benefit of integrating AI techniques into the IEEE 802.11 MAC layer for next-generation hybrid optical-wireless 6G networks.

Contents

1.	WORK PACKAGE OBJECTIVES	5
2.	MOTIVATION.....	6
3.	CHALLENGES AND REQUIREMENTS.....	7
3.1	HETEROGENEOUS TRAFFIC COEXISTENCE.....	7
3.2	DENSE AND CONGESTED ENVIRONMENTS.....	7
3.3	OFDMA SCHEDULING COMPLEXITY.....	7
4.	AI STRATEGIES FOR THE IEEE 802.11 MAC LAYER.....	8
4.1	AI-BASED TRAFFIC FLOW CLASSIFIER	8
4.1.1	FEATURE EXTRACTION FROM NETWORK FLOWS	9
4.1.2	APPLICATION MODELLING IN NS3	13
4.1.3	MACHINE LEARNING MODEL	14
4.1.4	EDCA-BASED QoS PRIORITISATION	14
4.1.5	TCP-AWARE SCHEDULING	15
4.1.5.1	TCP AND THE CONGESTION WINDOW.....	15
4.1.5.2	THE PROBLEM IN WI-FI NETWORKS	15
4.1.5.3	LIMITATIONS OF DOWNLINK-ONLY ML-BASED CLASSIFICATION	16
4.1.5.4	PROPOSED SOLUTION: FORCED UL BE-QUEUE LOOKUP IN THE AP SCHEDULER	17
4.1.5.5	EXPERIMENTAL VALIDATION	17
4.1.5.5.1	NETWORK SCENARIO	18
4.1.5.5.2	RESULTS	19
4.2	AI-BASED MULTI-USER SCHEDULER.....	20
4.2.1	OFDMA RESOURCE ALLOCATION PROBLEM.....	20
4.2.1.1	ASSUMPTIONS.....	21
4.2.2	REINFORCEMENT LEARNING FORMULATION.....	21
4.2.3	DEEP Q-NETWORK (DQN) ARCHITECTURE	24
4.2.3.1	TRAINING HYPERPARAMETERS	25
4.2.3.2	TRAINING RL AGENTS	25
5.	RESULTS	26
5.1	NS-3 SIMULATION ENVIRONMENT	26
5.1.1	NS-3 SIMULATOR.....	26
5.1.2	NS3-AI MODULE.....	26
5.1.3	INTEGRATION OF AI STRATEGIES WITH NS-3	27
5.2	AI-BASED TRAFFIC FLOW CLASSIFIER SIMULATION RESULTS	28
5.2.1	SIMULATION SCENARIO	28
5.2.1.1	METRICS.....	29
5.2.2	EVALUATION RESULTS ON SIMULATION SCENARIOS.....	30
5.2.2.1	SCENARIO 1: SINGLE AP WITH HEAVY LOCAL CONTENTION.....	30
5.2.2.2	SCENARIO 2: OBSS CONGESTION	32
5.2.2.3	AVERAGE AGGREGATE THROUGHPUT AND DELAY TRADE-OFF.....	34
5.2.2.4	FEATURE ROBUSTNESS UNDER CONGESTED CONDITIONS	34
5.2.3	CONCLUSIONS FOR AI-BASED TRAFFIC FLOW CLASSIFIER.....	35
5.3	AI-BASED MULTI-USER SCHEDULER SIMULATION RESULTS.....	36
5.3.1	SIMULATION SCENARIO	36
5.3.2	USE CASES.....	37
5.3.3	EVALUATION RESULTS ON TRAINED USE CASES.....	37
5.3.3.1	NON-CONGESTED SCENARIO (TRAINED)	38
5.3.3.2	CONGESTED SCENARIO (TRAINED).....	39
5.3.3.3	MODEL CONSISTENCY AND GENERALISATION	40
5.3.4	EVALUATION RESULTS ON NON-TRAINED USE CASE	40



5.3.5	CONCLUSIONS FOR AI-BASED MULTI-USER SCHEDULER	41
6.	SUMMARY.....	42
7.	APPENDIX.....	43
7.1	FEATURE SELECTION PROCESS	43
7.1.1	SUMMARY TABLE	43
7.1.2	EXPLANATION OF EACH ITERATION	44
7.1.3	DATASET FROM NS3 APPLICATIONS	46
7.1.4	CONFUSION MATRIX TRAINING MODEL	48
8.	REFERENCES	49

List of Figures

FIGURE 1:	FLOW-BASED PACKET SEGMENTATION AND TEMPORAL WINDOWING.....	9
FIGURE 2:	ACKs DELAYED BY AC_BE CONTENTION	16
FIGURE 3:	UL-BE SOLICITATION VIA TRIGGER FRAME (PREFERRED AC: STARTING FROM AC_BE)	17
FIGURE 4:	TWO OVERLAPPING BASIC SERVICE SETS (OBSS_1 AND OBSS_2)	18
FIGURE 5:	NS3-AI STRUCTURE	26
FIGURE 6:	SCENARIO 1 - A SINGLE BSS WITH 1 AP (AP1) AND 10 ASSOCIATED STAs.....	30
FIGURE 7:	SCENARIO 2 - AN OBSS TOPOLOGY WITH TWO APs..	32
FIGURE 8:	CONFUSION MATRIX – TRAINING ETHERNET CHANNEL.....	48
FIGURE 9:	CONFUSION MATRIX – TESTING ETHERNET CHANNEL	48

List of Tables

TABLE 1:	SUMMARY OF T2.3.....	5
TABLE 2:	HYPERPARAMETERS EVALUATED IN ML MODEL	14
TABLE 3:	THROUGHPUT AND DELAY METRICS MEASURED FOR THE 50 MBPS TCP FLOW (STA1)	19
TABLE 4:	ACTION SPACE FOR 4 STAs	23
TABLE 5:	DOUBLE DQN NETWORK CONFIGURATION LAYER.....	24
TABLE 6:	DOUBLE DQN NETWORK HYPERPARAMETER	25
TABLE 7:	ML FRAMEWORK USING NS3-AI.....	27
TABLE 8:	SIMULATION PARAMETERS FOR SCENARIOS	28
TABLE 9:	APPLICATION CONFIGURATION FOR SCENARIO 1	30
TABLE 10:	SCENARIO 1: BE BASELINE VS. ML-BASED CLASSIFICATION.	31
TABLE 11:	APPLICATION CONFIGURATION FOR SCENARIO 2.	32
TABLE 12:	SCENARIO 2 - BE BASELINE VS. ML-BASED CLASSIFICATION RESULTS	33
TABLE 13:	SIMULATION SETTINGS AND PARAMETER VALUES	36
TABLE 14:	USE CASES DEFINED FOR THE THREE EVALUATIONS	37
TABLE 15:	NON-CONGESTED TRAINED CASE RESULTS	38
TABLE 16:	CONGESTED TRAINED CASE RESULTS.....	39
TABLE 17:	NON-TRAINED CASE RESULTS	41
TABLE 18:	ITERATIVE FEATURE SELECTION PROCESS	44

1. Work Package Objectives

The work package (WP) objectives are as follows. Highlighted in bold are the items covered in this report.

Objectives: To

- i. optimize the energy usage of the wireless sensor network (WSN) for Internet of Things (IoT) and Internet of Everything (IoE) applications in sensitive medical and industrial environments, which is crucial for battery-powered sensor nodes (SNs);
- ii. analyze the limitations of sensor infrastructure, including powerline communication (PLC) as a cost effective backbone;
- iii. propose relaying and switching solutions in order to reduce power consumption as well as provide the necessary tools to improve the backbone for radio frequency (RF) and optical wireless communication (OWC) in a WSN;
- iv. provide tools for improving hybrid RF-OWC backbones in WSNs;
- v. **design efficient WSN medium access control (MAC) layer protocols that incorporate machine learning and perform a trade-off between data rates and power consumption.**

Work Package Number	WP2	Lead Beneficiary	MaxLinear, Inc
Task	2.3		
Start Month	9	End Month	32
WP Name		Physical and MAC Layer	
Task Name		MAC protocols	

Table 1: Summary of T2.3

Task 2.3: MAC protocols (Leader: MXL, Conts.: ALL) [DC 4]. In this task, we will (i) design efficient MAC layer scheduling algorithms for WSN, based on existing mobile RF standards such as 802.11; (ii) assume different traffic patterns, including sensor communications and media applications; (iii) investigate the application of machine-learning based scheduling algorithms for the hybrid OWIN6G network as well as the incorporation of energy efficiency awareness into the scheduling process; and (iv) determine the balance between high data rates and low energy consumption for different application settings.

2. Motivation

The IEEE 802.11 standard has undergone remarkable evolution over the past decades, progressing from initial data rates of 1 Mbps in 1997 to theoretical peaks approaching 10 Gbps with IEEE 802.11ax (Wi-Fi 6) [1] and beyond. However, despite these substantial increases in physical layer capacity, the Medium Access Control (MAC) layer remains a critical bottleneck in dense wireless environments [2]. As the number of connected devices continues to grow, driven by the proliferation of IoT sensors, video streaming endpoints, real-time collaboration tools, and emerging applications requiring low-latency communication, the contention-based channel access mechanism of IEEE 802.11 faces increasing pressure that raw throughput improvements alone cannot resolve [3].

The fundamental challenge in modern Wi-Fi networks is not the aggregate capacity, but rather how intelligently the available radio resources are scheduled and prioritized across heterogeneous traffic flows with diverse Quality of Service (QoS) requirements [4]. A video conferencing session and a file download may share the same Access Point (AP), yet they have dramatically different latency sensitivities. The IEEE 802.11e Enhanced Distributed Channel Access (EDCA) mechanism was designed precisely for this differentiation, providing four Access Categories (Voice, Video, Best Effort, Background) with distinct contention parameters [5]. In practice, however, the EDCA mechanism is frequently rendered ineffective because the contextual information it relies upon is actively destroyed by the network infrastructure itself. Network boundaries actively strip QoS marks through 'DSCP bleaching', resetting the field to zero and forcing flows into the default Best Effort class [6]. This practice is particularly aggressive at domain exits; for example, when packets leave a service provider network, their DSCP values are overwhelmingly bleached to zero before traversing the remainder of the path [7]. As a result, by the time traffic arrives at the Access Point, the EDCA mechanism is left completely blind and unable to differentiate latency-sensitive flows.

In this context, Artificial Intelligence (AI) with Machine Learning (ML) and Deep Learning (DL) emerge as a compelling approach to transform the static, rule-based MAC layer operations into adaptive, data-driven mechanisms capable of responding to real-time network conditions [8].

The OWIN6G project [9], funded under the EU Horizon MSCA-DN programme, investigates the application of artificial intelligence to MAC protocol design through Task 2.3, which specifically targets the design of efficient MAC layer scheduling algorithms for wireless sensor networks based on existing mobile RF standards such as IEEE 802.11.

Within this task, two complementary AI-based strategies have been developed and evaluated:

The first strategy, the AI-Based Traffic Flow Classifier, addresses the problem of identifying and differentiating application types in real-time at the AP level. This is framed as a supervised learning classification problem. By deploying a ML classifier that inspects flow-level statistical features rather than payload content, the AP can dynamically assign each flow to the appropriate EDCA Access Category (AC), restoring QoS prioritisation without requiring any cooperation from applications or client stations.

The second strategy, the AI-Based Multi-User Scheduler, targets the Orthogonal Frequency Division Multiple Access (OFDMA) resource allocation problem introduced in IEEE 802.11ax. OFDMA enables simultaneous transmission to multiple users by subdividing the channel bandwidth into Resource Units (RUs), but the scheduling algorithm that determines which stations receive which RUs is intentionally left unspecified by the standard. The combinatorial complexity of this allocation problem, combined with the dynamic nature of wireless channel conditions and user traffic demands, makes it a natural candidate for Reinforcement Learning

(RL) approaches that can learn optimal scheduling policies through interaction with the network environment.

Both strategies share a common motivation: leveraging AI to bridge the gap between the theoretical capacity of modern Wi-Fi standards and the actual user experience in congested, heterogeneous network environments. Where the traffic classifier operates at the flow level to ensure that latency-sensitive applications receive MAC-layer priority, the multi-user scheduler operates at the resource level to optimize how bandwidth is distributed among competing stations. Together, they represent a comprehensive, AI framework for next-generation Wi-Fi MAC protocol design that addresses the core objectives of Task 2.3.

3. Challenges and Requirements

The design of efficient MAC layer scheduling algorithms for IEEE 802.11 networks, capable of supporting both wireless sensor communications and media-intensive applications, must address several interconnected technical challenges. These challenges arise from the need to accommodate diverse traffic patterns, integrate AI into real-time scheduling decisions, and validate the proposed solutions across multiple deployment scenarios.

3.1 Heterogeneous Traffic Coexistence

APs must simultaneously manage traffic with vastly different characteristics, such as latency-critical voice and video conferencing, high-throughput video streaming, gaming, and bulk data applications. To optimize performance, the MAC scheduler must dynamically differentiate among these flows and allocate resources in real-time. However, because most applications fail to signal their QoS requirements via standard Differentiated Services Code Point (DSCP) fields, real-time traffic identification becomes a complex challenge that must be resolved directly at the AP.

3.2 Dense and Congested Environments

The proposed solutions must be validated not only in isolated single Basic Service Set (BSS) environments but also under Overlapping BSS (OBSS) conditions. In these scenarios, co-channel interference from neighbouring networks introduces additional contention and unpredictable channel occupancy. The AI-based MAC strategies must demonstrate robustness under these conditions, where interference from neighbouring networks can consume a significant fraction of the available airtime and disrupt the reliability of managed transmissions.

3.3 OFDMA Scheduling Complexity

The OFDMA feature of IEEE 802.11ax allows the AP to subdivide the channel into multiple RUs and assign them to different stations simultaneously. While this dramatically increases spectral flexibility, it also introduces a highly complex scheduling problem: the number of valid RU allocations grows significantly with the number of associated stations and available RU sizes. Given this complexity, and the fact that the scheduling mechanism is intentionally left unstandardised in IEEE 802.11ax, previous work has explored classical optimisation approaches such as linear programming and the Hungarian algorithm¹. However, the difficulty of finding an optimal solution motivates the exploration of ML algorithms, which are capable of

¹ The Hungarian algorithm is a combinatorial optimization method used in WLANs to achieve optimal resource allocation (e.g., subcarriers or time slots) by mapping users to channels to maximise overall network throughput.

learning patterns from large datasets and, when applied correctly, of making effective allocation decisions.

4. AI strategies for the IEEE 802.11 MAC layer

This deliverable investigates two complementary AI strategies that address different aspects of MAC layer intelligence in IEEE 802.11 networks. The first, an AI-Based Traffic Flow Classifier, operates at the boundary between the network/transport layer and the MAC layer, dynamically identifying application types and triggering appropriate QoS treatment through the EDCA mechanism. The second, an AI-Based Multi-User scheduler, learns how to assign RU effectively to stations in order to maximise network performance. The two strategies are not mutually exclusive; they can be combined to address traffic prioritisation and multi-user resource allocation simultaneously.

4.1 AI-Based Traffic Flow Classifier

Traditional classification methods, such as port-based inspection and Deep Packet Inspection (DPI), have become increasingly ineffective due to the widespread adoption of encryption and the tendency of modern applications to tunnel all traffic over a small set of well-known ports [10]. As a result, ML-based approaches have emerged as the dominant paradigm, leveraging statistical features extracted from packet flows, such as packet sizes, inter-arrival times, and byte counts, to classify traffic without inspecting payload content [11].

A wide range of supervised classifiers have been applied to this problem, from k-Nearest Neighbours and Support Vector Machines to ensemble methods like Random Forest and Gradient Boosting, with the latter consistently achieving the highest accuracy across diverse datasets [11], [12]. On the other hand, DL architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and autoencoders have demonstrated the ability to automatically extract discriminative features from raw traffic data, outperforming traditional ML methods even when the traffic is encrypted or captured at different network layers [10].

In the specific context of Wi-Fi, traffic classification has been used to distinguish interactive XR/VR traffic from background flows, enabling latency-aware prioritisation at the access point that can reduce VR traffic delays by up to 4.2x compared to scenarios without prioritisation [11]. Furthermore, the IEEE 802.11 AIML Topic Interest Group has identified traffic-aware QoS optimisation as a key use case for AI/ML-native Wi-Fi, highlighting the growing standardisation interest in embedding classification intelligence directly into the MAC layer [13].

However, most existing approaches evaluate classification performance in isolation, without integrating the classifier into a real-time QoS enforcement pipeline at the AP level. Building on these foundations, the traffic flow classifier presented in this Section addresses this gap by adopting a supervised learning approach based on flow-level statistical features to dynamically assign each detected flow to the appropriate EDCA Access Category at the AP.

4.1.1 Feature Extraction from Network Flows

The traffic flow classifier is designed to operate at the AP, observing packets as they arrive from the Layer 3 and before they are queued for wireless transmission. The classification pipeline begins with a feature extraction stage that transforms raw packet streams into a fixed-dimensional feature vector suitable for ML inference.

The feature extraction operates on a sliding window basis. A 5-second observation window advances in 1-second steps, producing overlapping windows that capture both transient and sustained traffic characteristics per one flow, shown in Figure 1. Within each window, the system maintains per-flow state indexed by the standard 5-tuple (source IP, destination IP, source port, destination port, protocol). For each incoming packet, the system either updates an existing flow or initiates a new one.

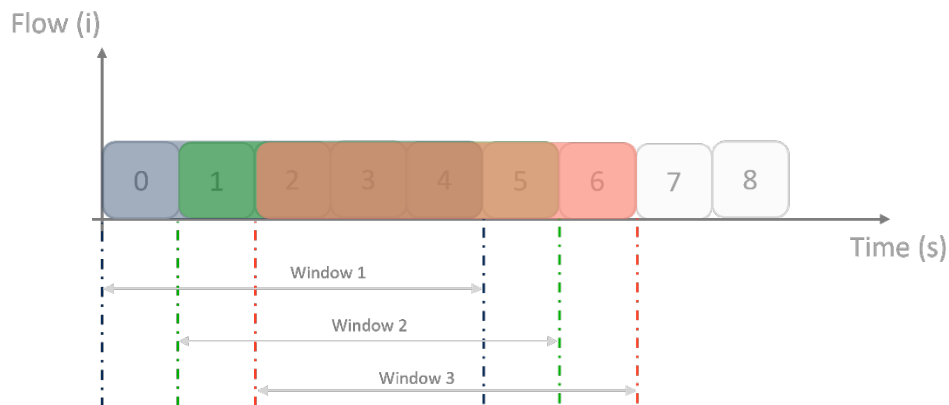


Figure 1: Flow-based packet segmentation and temporal windowing

The 5-second window length is consistent with the traffic burst patterns reported in [14]. Shorter windows would fragment the data and degrade the robustness of the extracted metrics, while longer windows would smooth out temporal dynamics and reduce responsiveness to traffic variations. A sliding-window approach with a step size of one second is employed to preserve temporal resolution without introducing additional latency.

In total, 16 features are extracted per flow and per window. These features were selected through a feature importance analysis that removed redundant and congestion-sensitive variables from an initial set of 29 candidates introduced in [14]. The reduction to 16 features is itself a contribution (see Appendix 7.1), as it improves classifier robustness under Wi-Fi congestion conditions where some network-layer statistics become unreliable. The following categories of features are computed from the packets falling within the current window:

1. Basic Metrics (1 feature):

- Protocol: Identifies the transport layer protocol of the packet, where 0 indicates TCP and 1 indicates UDP.

2. Statistics of Packet Sizes (4 features):

- Minimum Packet Size: The size of the smallest packet (in bytes) within the flow.
- Mean Packet Size: The average size of packets (in bytes) within a flow.
- Standard Deviation of Packet Size: The standard deviation of packet sizes within the flow, indicating variability.
- Maximum Packet Size: The largest packet size (in bytes) observed in the flow, often corresponding to Maximum Transmission Unit (MTU) limits.

3. Temporal (3 features):

Inter-Arrival Time (IAT) is the time intervals (in seconds) between consecutive packets within a flow. Key statistics derived from IAT include:

- Minimum IAT: The minimum IAT between two consecutive packets in the flow.
- Standard Deviation of IAT: This indicates the variability in packet arrival times in the flow, highlighting irregular traffic patterns
- IAT Variance Ratio: The ratio of IAT variance to mean IAT, representing the relative variability in the arrival time of the packets.

4. Flow Dynamics (2 features):

- Burstiness: A measure of irregularity in the packet size distribution, calculated as the normalized difference between the standard deviation and the mean of the packet sizes, as show in Equation 4.1:

$$\text{Burstiness} = \frac{\text{Std Packet Size} - \text{Mean Packet Size}}{\text{Std Packet Size} + \text{Mean Packet Size}} \quad (4.1)$$

if $\text{Std Packet Size} + \text{Mean Packet Size} \neq 0$

Low burstiness (close to -1) indicates steady traffic with packet sizes that are consistent and predictable, and high burstiness (close to 1) indicates sporadic traffic with high variability in packet sizes.

- Directions: Each biflow is canonicalized by lexicographically ordering the endpoint tuples (IP, port). The endpoint with the smaller tuple is designated as the *forward* side. This convention is purely header-based.
Let N_{fwd} and N_{rev} denote the number of packets in the forward and reverse directions within the window and define $N = N_{\text{fwd}} + N_{\text{rev}}$. The feature is defined for $N > 0$ in the following Equation 4.2:

$$\text{Directions} = \frac{N_{\text{fwd}}}{N} \quad (4.2)$$

Values near 0.5 indicate symmetric exchanges, whereas values close to 1 (or 0) indicate strong asymmetry (downlink or uplink-dominated flows).

5. Rate-Based (3 features):

- Packet Rate: The rate of packets per second in the flow, as calculated in Equation 4.3:

$$\text{Packet Rate} = \frac{\text{Total Packets}}{\text{Duration}} \quad (4.3)$$

With:

- Total Packets: The sum of forward and reverse number of packets in a flow.
- Duration: The total duration of the flow, from the first packet to the last, within the 5-second sliding window.



- **Data Transmission Intensity (DT Intensity):** Based on physics, the idea is established through an analogy to Newtonian principles. In this context, the analysis of network traffic adopts concepts similar to those in classical mechanics. In this context, the total amount of data transmitted in a flow (*distance = bytes*) is analogous to the distance travelled in a physical system, representing the volume of data exchanged. Similarly, the total duration of the transmission (*time = duration*) corresponds to the period over which the data transfer occurs.

The mean inter-arrival time ($dt = IAT$), reflecting the average time interval between consecutive packets, serves as a measure of temporal granularity. This relationship forms the basis for defining metrics such as Data Transmission Intensity, which is calculated using Equation 4.4:

$$DT\ Intensity = \frac{Total\ Bytes \times Duration}{Mean\ IAT} \quad (4.4)$$

With:

- Mean IAT: The average interval between packets, reflecting the regularity of packet transmission in the flow.

This metric integrates the volume of data and its temporal distribution, capturing both the intensity of traffic flows.

- **RMS Packet Size:** Root Mean Square of packet sizes within the flow, providing a measure of the magnitude of packet sizes, which is calculated using Equation 4.5:

$$RMS\ Packet\ Size = \sqrt{\frac{1}{n} \sum_{i=1}^n (Packet\ size\ of\ packet\ i)^2} \quad (4.5)$$

6. Aggregation and Smoothing (3 features):

Equation 4.6 applies an Exponential Moving Average (EMA) to three metrics, computing each value by combining the current observation with the previous history, and assigning weight to recent changes depending on the factor α :

$$EMA_t = \alpha \cdot X_t + (1 - \alpha) \cdot EMA_{t-1} \quad (4.6)$$

where:

EMA_t : The smoothed value at the current time window,

X_t : is the observed value in the current window,

α : The smoothing factor (a value between 0 and 1), controlling how much weight recent values have.

- Higher α (e.g., 0.5): Reacts faster to recent changes.

- Lower α (e.g., 0.1): More stable, relies more on past values.

EMA_{t-1} : The smoothed value from the previous window.



If a flow disappears for several windows, the last EMA value of these three metrics is exponentially decayed over that interval before incorporating the new observation when the flow reappears, as calculated using Equation 4.7.

$$EMA_t = (1 - \alpha)^{(t - t_{prev})} \cdot EMA_{t_{prev}} + \alpha \cdot X_t \quad (4.7)$$

where:

EMA_t : is the smoothed value at the current time window,
 t_{prev} : is the last window where the flow was observed,
 X_t : is the new observed value when the flow reappears,
 α : The smoothing factor,
 $(t - t_{prev})$: represents the number of missing windows.

- Average Data Transmission Intensity: Exponentially weighted average data transmission intensity over time windows, providing a smoothed view of flow dynamics.
- Average Maximum Packet Size: This reflects the average of the largest packet sizes observed in each window, useful for identifying changes in maximum data load.
- Average Minimum Packet Size: This tracks the average of the smallest packet sizes, offering insights into the consistency of minimal data transmissions.

4.1.2 Application Modelling in NS3

To train and evaluate the classifier, a diverse set of network applications was modelled in the NS-3 simulator [15]. Each application generates traffic with characteristic statistical patterns that the classifier must learn to distinguish. The modelled applications include:

- **Video Conference:** A unidirectional UDP downlink traffic generator at up to 70 Mbps operating at 30 FPS, with packet sizes between 64 and 1428 bytes based on a real traffic trace. The model reproduces realistic bitrate variations by shuffling and scaling packet sizes with random burst patterns.
- **Gaming (Quake 3):** a bidirectional application producing approximately 0.14 Mbps with 110-byte packets, exhibiting the characteristic low-rate, low-latency profile of interactive gaming. Packet Interval is a random variable with distribution $0.6 \cdot E + 0.4 \cdot C$, where E is exponentially distributed with mean 4.29 ms, and C is the constant 10.75 ms.
- **VoIP:** a downlink application at 0.12 Mbps with 120-byte packets, generating constant-rate, small-packet traffic.
- **Video Streaming (DASH model):** a downlink application at up to 50 Mbps with 576-byte packets, modelling adaptive bitrate streaming over HTTP/TCP using the FDASH algorithm [16], with an initial burst followed by sustained delivery.
- **Iperf TCP:** a downlink TCP bulk transfer at up to 50 Mbps with 588-byte packets, representing generic TCP file transfers.
- **Iperf UDP:** a downlink UDP stream at up to 70 Mbps with 1428-byte packets, representing high-throughput UDP applications.
- **Bursty UDP:** a downlink application at up to 70 Mbps with 1475-byte packets and 100-ms burst periods, capturing the bursty nature of aggregation traffic.
- **Bulk TCP:** a downlink TCP flow at up to 70 Mbps with 1475-byte packets, modelling large file downloads.

The training set was built using a fixed data rate for each traffic application. Specifically, 10 Mbps was used for Bulk TCP, Bursty UDP, Iperf TCP, and Iperf UDP; 3 Mbps for Video Conference; and 20 Mbps for Video Streaming.

The test set, however, included both samples at the nominal training rate and additional samples generated at different data rates (up to 70Mbps), in order to evaluate the model's robustness and generalization under unseen transmission-rate conditions.

4.1.3 Machine Learning Model

The classifier employs eXtreme Gradient Boosting[17] (XGBoost), a gradient-boosted decision tree ensemble widely recognized for its strong predictive performance in classification tasks. To identify the optimal hyperparameter configuration, an exhaustive grid search combined with 5-fold cross-validation [18] (GridSearchCV) was performed exclusively on the training set, thereby avoiding any data leakage from the test set.

Hyperparameter	Values evaluated	Description
<i>n_estimators</i>	20, 50, 100, 200, 300	Number of boosting rounds
<i>max_depth</i>	3, 5, 7, 9	Maximum tree depth
<i>learning_rate</i>	0.01, 0.1, 0.2, 0.3	Step size shrinkage
<i>subsample</i>	0.5, 0.8, 1.0	Row subsampling ratio per tree
<i>colsample_bytree</i>	0.5, 0.8, 1.0	Column subsampling ratio per tree
<i>colsample_bylevel</i>	0.5, 0.8, 1.0	Column subsampling ratio per level
<i>colsample_bynode</i>	0.5, 0.8, 1.0	Column subsampling ratio per node

Table 2: Hyperparameters evaluated in ML model

For each candidate combination, the mean cross-validation accuracy across the five folds was computed, and the configuration achieving the highest score was selected. This systematic tuning procedure helps improve generalization to unseen data and reduces the risk of overfitting.

The model was trained on the 16-feature vectors extracted from Ethernet-channel captures and never retrained on Wi-Fi data. Under these conditions, it achieved a classification accuracy of 99.82% on the held-out test set. This high accuracy on clean (non-congested) data establishes the baseline performance of the classifier. The subsequent validation under Wi-Fi congestion scenarios, reported in Section 5.2, confirms that the model generalises effectively to wireless channel conditions.

4.1.4 EDCA-Based QoS Prioritisation

Once the classifier identifies the application type of each downlink flow, the AP maps it to the corresponding IEEE 802.11 EDCA Access Category. The EDCA mechanism, defined in the IEEE 802.11e amendment [19], differentiates channel access by assigning different contention parameters (Arbitration Inter-Frame Spacing, Contention Window sizes, and Transmit Opportunity (TXOP) limits) to four ACs: Voice (AC_VO), Video (AC_VI), Best Effort (AC_BE), and Background (AC_BK). Delay-sensitive applications, such as VoIP, are assigned to AC_VO. Video Streaming, Gaming, and Video Conferencing are mapped to AC_VI, whereas Iperf UDP, Iperf TCP, Bulk TCP, and Bursty UDP are assigned to AC_BE.

This dynamic, ML-driven mapping replaces the ineffective static DSCP-based approach where applications universally mark their traffic as Best Effort. The framework thus closes the gap between the network/transport layer, where application-discriminating features are observable, and the MAC layer, where QoS enforcement is executed.

4.1.5 TCP-Aware Scheduling

Leveraging the ML-based traffic classifier, which is capable of identifying TCP flows, the AP scheduler can adapt its operation to the specific dynamics of TCP. When serving a STA carrying TCP traffic, the scheduler appends a Trigger-based uplink (UL) phase immediately after the downlink (DL) transmission within the same TXOP. This allows freshly generated TCP acknowledgements (ACKs) to be returned to the sender without contending for the medium, thereby reducing the effective round-trip time (RTT).

This design is based on the following key assumptions:

1. **Fast ACK generation at the STA:** STAs are capable of generating and enqueueing TCP ACKs within a short processing interval between the DL and UL phases, enabling their transmission during the immediately following UL phase.
2. **STA independence from AP-side QoS policies:** STAs are not aware of, nor do they enforce, the DSCP-based classification rules applied at the AP. Consequently, the UL phase is initiated by means of a Trigger Frame (TF) indicating Best Effort (BE) as the preferred starting Access Category (AC), allowing stations to transmit frames from any AC queue, from the lowest priority (AC_BE) up to the highest (AC_VO), in accordance with the TF rules.

4.1.5.1 TCP and the Congestion Window

The TCP protocol relies on acknowledgements (ACKs) to regulate the data transmission rate: the sender injects new data into the network as ACKs arrive from the receiver. The amount of data that can be outstanding at any given time is limited by the congestion window size, which grows as ACKs are successfully received and shrinks upon detecting congestion signals, such as packet loss or excessive delays leading to retransmission timeouts.

Design parameters, including the initial congestion window and the receiver window, further constrain this behaviour. As a result, delayed or lost ACKs directly translate into throughput degradation; in extreme cases, the congestion window may stall entirely, bringing data transmission to a halt. Timely delivery of ACKs is therefore critical to sustaining TCP performance.

4.1.5.2 The Problem in Wi-Fi Networks

In IEEE 802.11 networks employing Enhanced Distributed Channel Access (EDCA), all nodes (both the AP and associated STAs) compete for medium access under the same contention-based rules. When contention levels are high, packets accumulate in the transmit buffers of nodes that fail to gain channel access promptly. As the number of active contenders increases, channel access delays grow, queues build up, and overall network efficiency degrades.

This phenomenon is particularly detrimental for TCP traffic compared to UDP. Unlike UDP, TCP adapts its sending rate based on the congestion it infers from the network, primarily through ACK timing and loss. As a result, increased contention and buffering delays on the wireless channel are interpreted by TCP as congestion, leading to reduced congestion windows and diminished throughput.

4.1.5.3 Limitations of Downlink-Only ML-Based Classification

When ML-based traffic classification is applied at the AP to prioritize downlink flows, for example, promoting video traffic from Best Effort (AC_BE) to Video (AC_VI), the classification inherently applies only to downlink transmissions. While the AP can classify and prioritize outgoing data frames, the corresponding TCP ACKs are generated at the stations and transmitted on the uplink.

Since STAs do not apply any traffic flow re-classification, TCP ACKs remain mapped to the default Best Effort queue (AC_BE). Consequently, the following situation arises: the AP gains channel access, secures a TXOP, and transmits high-priority downlink data using favourable EDCA parameters (e.g. AC_VI or AC_VO). However, the ACKs for those same flows are queued in the STA's BE queue, where they contend with less favourable EDCA parameters, including longer Arbitration Inter-Frame Space (AIFS) durations and larger contention windows, as illustrated in Figure 2.

EDCA allocates smaller contention windows, shorter AIFS, and larger TXOP limits to higher-priority access categories and inversely penalizes lower-priority ones. As a result, ACKs experience excessive delays or losses, triggering TCP congestion control mechanisms that reduce the sending rate. In effect, downlink prioritization undermines its own benefits by starving the uplink ACKs required to sustain TCP throughput.

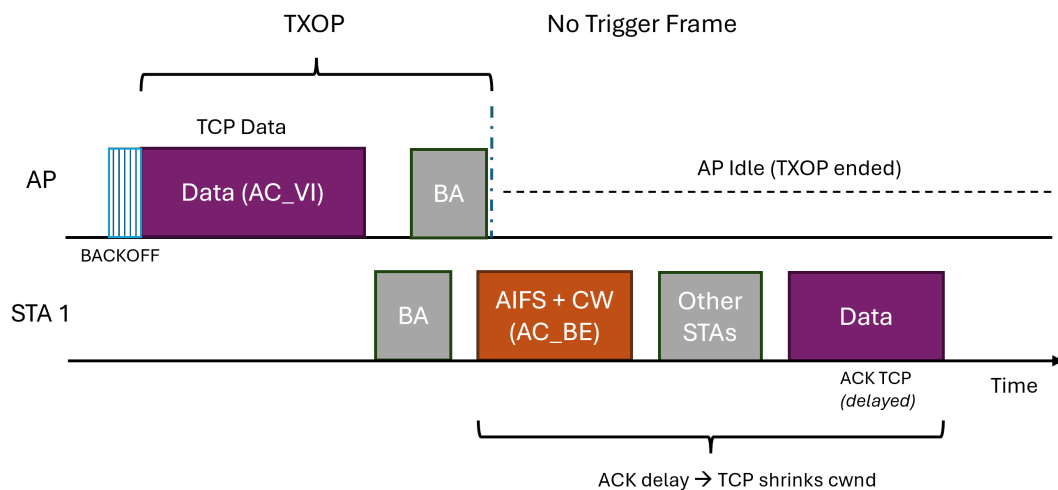


Figure 2: ACKs delayed by AC_BE contention

4.1.5.4 Proposed Solution: Forced UL BE-Queue Lookup in the AP Scheduler

To address this limitation, the proposed solution illustrated in Figure 3, introduces a modification to the AP scheduler behaviour. After winning contention and obtaining a TXOP, the AP explicitly controls the uplink access by transmitting a Trigger Frame (TF) during the UL phase, with the Preferred AC field configured to solicit transmissions starting from the stations' Best Effort queues.

Rather than restricting uplink solicitation to the access category that won contention on the downlink, the AP effectively scans uplink queues starting from AC_BE and moving toward higher-priority ACs. This mechanism prompts stations to transmit queued TCP ACKs within the same TXOP, fully bypassing EDCA contention on the uplink.

By ensuring timely delivery of TCP ACKs, the sender's congestion window remains open, RTT is reduced, and the throughput gains achieved through ML-based downlink traffic classification are preserved. This coordinated DL/UL scheduling approach restores end-to-end TCP efficiency while remaining compatible with standard-triggered uplink operation in IEEE 802.11 networks.

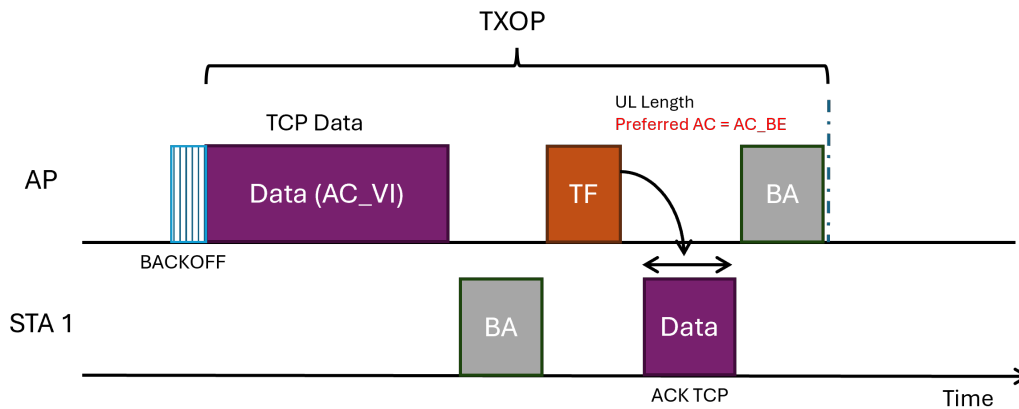


Figure 3: UL-BE solicitation via Trigger Frame (Preferred AC: starting from AC_BE)

4.1.5.5 Experimental Validation

To validate the proposed UL-BE solicitation mechanism, a comparative evaluation was conducted across three scheduling configurations:

- **Baseline (BE only):** All traffic mapped to the default Best Effort queue, no ML classification, no scheduler modification.
- **ML classification only:** The AP's ML classifier elevates the target flow from AC_BE to AC_VI on the downlink, but the uplink scheduler remains unmodified. TCP ACKs still contend through AC_BE with standard EDCA parameters.
- **ML classification + TB-UL (AC_BE):** The ML classifier operates on the downlink as before, and the AP scheduler additionally issues Trigger Frames with Preferred AC², starting from AC_BE during the uplink phase of each TXOP, soliciting TCP ACKs directly from the stations' BE queues.

² Preferred AC is a field in the trigger-dependent user information of a Basic Trigger frame that specifies the lowest recommended access category for MPDU aggregation in the A-MPDU of a trigger-based uplink response.

4.1.5.5.1 Network Scenario

The evaluation topology shown in Figure 4, introduces mutual interference and contention for medium access.

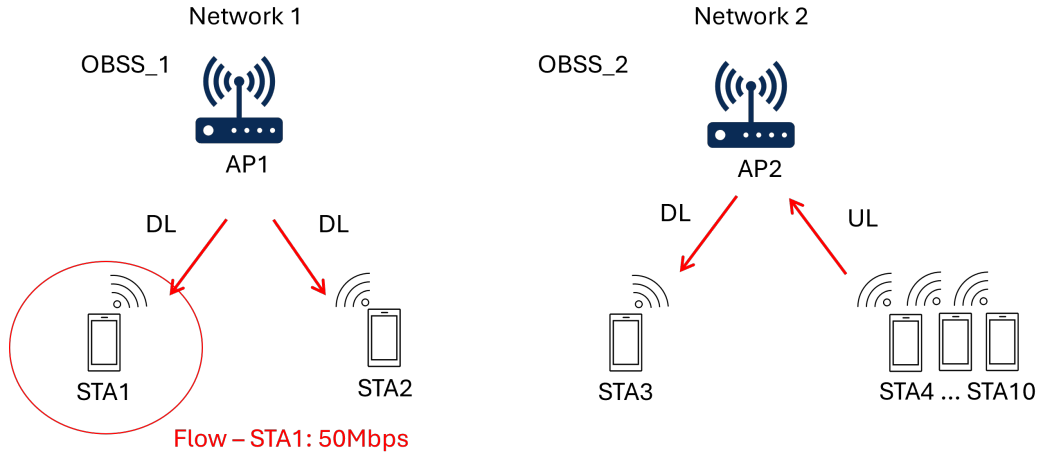


Figure 4: Two overlapping basic service sets (OBSS_1 and OBSS_2) sharing the same physical channel

- Network 1 (OBSS_1) comprises AP1 serving two stations: STA1 receives the primary high-throughput downlink flow (50 Mbps, BULK_TCP), while STA2 receives a secondary flow (5 Mbps, BULK_TCP).
- Network 2 (OBSS_2) is designed to saturate the shared channel: AP2 delivers downlink traffic to STA3, while seven additional stations (STA4–STA10) transmit uplink UDP traffic (70 Mbps aggregate) toward AP2, generating sustained contention and co-channel interference.

This topology emulates a dense deployment with overlapping coverage areas. The network operates in Wi-Fi 7 (EHT) mode on the 5 GHz band with 160 MHz channel bandwidth. At the PHY layer, all transmissions use MCS 11, a single spatial stream (NSS = 1), and an 800 ns guard interval. The OBSS path loss between nodes of different BSSs is set to 50 dB. The AP employs a Trigger-Based EDCA scheduler; the STAs use EDCA.

4.1.5.5.2 Results

Table 3 summarizes the performance of the 50 Mbps flow directed to STA1, measured after 27 seconds of simulation:

TCP Flow 50 Mbps (STA 1) Simulation Time: 27 s	Baseline	ML classification	ML classification + TB-UL
DL Throughput (Mbps)	42.77	44.13	49.02
UL Throughput (ACKs) (Mbps)	1.12	1.15	1.27
DL Delay (ms)	8	3	2
UL Delay UL (ms) ³	0	5	0

Table 3: Throughput and delay metrics measured for the 50 Mbps TCP flow (STA1) with different classification and scheduling configurations.

Results confirm that ML classification alone yields a modest throughput improvement (+3.2%) and reduces downlink delay from 8 ms to 3 ms due to a smaller contention window of AC_VI. However, the uplink delay increases from 0 to 5 ms. A direct consequence of TCP ACKs remaining in the station's BE queue, where they face unfavourable EDCA parameters and heavy contention from OBSS_2's uplink traffic. This ACK bottleneck limits the TCP congestion window and limits the throughput gain.

When the proposed TB-UL mechanism is activated alongside ML classification, the AP solicits ACKs via Trigger Frames with Preferred AC (starting from AC_BE), bypassing EDCA contention entirely. The effect is immediate: uplink delay drops back to 0 ms, ACK throughput rises to 1.27 Mbps, and downlink throughput reaches 49.02 Mbps. A 14.6% improvement over the baseline and 11.1% over ML classification alone. Downlink delay further decreases to 2 ms.

These results demonstrate that traffic prioritisation, in the particular case of DL TCP flow needs to accelerate the uplink return path for TCP ACKs in Wi-Fi networks to optimize both flow throughput and latency.

³ The reported value of 0 ms does not imply the absence of delay, but rather that the measured latency falls below the temporal resolution of the simulation environment.

4.2 AI-Based Multi-User Scheduler

Due to the introduction of OFDMA in IEEE 802.11ax, the scheduling problem has attracted significant research interest. Among classical optimisation approaches, throughput-maximising linear programming schedulers have been tested in NS-3 [20], the Hungarian Algorithm has been applied with different utility functions such as Max Rate, Proportional Fairness, and Shortest Remaining Processing Time [21], and binary-search algorithms have been employed to minimise the maximum transmission time per OFDMA transmission [22].

In the ML domain, unsupervised deep learning has been applied to 5G UL OFDMA scheduling [23], while supervised deep learning has been used to reduce channel sounding overhead in Wi-Fi 6 [24]. Deep Reinforcement Learning has also gained traction, with DQN applied to distributed 5G resource block allocation [25] and Graph Convolutional Networks employed for multi-AP channel allocation [26].

Reinforcement Learning was ultimately selected for this work because the scheduling problem naturally maps to a sequential decision framework (Markov Decision Process⁴): the network state serves as the observation, the RU allocations correspond to actions, and the performance metric defines the reward.

Moreover, RL offers a distinctive advantage for deployment: its online learning capability allows a pre-trained agent running on a commercial AP to continue adapting to its specific environment, traffic patterns, and interference conditions in real-time.

4.2.1 OFDMA Resource Allocation Problem

The OFDMA feature introduced in the IEEE 802.11ax amendment allows the AP to subdivide the available channel bandwidth into adjacent orthogonal subcarriers known as RUs of different sizes (26, 52, 106, 242, 484, 996, or 2x996 tones) and assign them to multiple STAs to perform simultaneous Multi-User transmissions [1].

OFDMA scheduling can be described as the task in which the AP decides how to distribute the RUs, in the channel employed, between the Stations on each DL/UL transmission in order to maximise some performance metric. This must be done in terms of the network state, which includes the Modulation and Coding Scheme (MCS), the buffered data, average input traffic per Station, RU quality per Station, etc. To be specific, this is an optimisation problem where the variable to be optimized represents the performance of the network (e.g. network throughput, latency, etc.). Moreover, this optimisation problem is subject to some constraints:

- There is a limited number of RUs to allocate for a given channel width, e.g. for an 80 MHz channel, possibilities can be one of the following: 1xRU996, or 2xRU484, or 4xRU242, or 8xRU106, or 16xRU52, or 32xRU26.
- Only 1 Station can be allocated to an RU (two Stations must not share RUs).
- Only 1 RU can be allocated to a Station (a Station must not operate in two RUs at the same time).
- The RU allocation is kept during a TXOP. It cannot be changed during the transmission.
- Only some RU configurations are allowed. If some RU of a specific size is allocated, then the “sub-RUs” that can be formed by splitting the allocated RU cannot be allocated.

⁴ Markov Decision Process (MDP): A mathematical framework for modeling decision-making in stochastic environments, used to find an optimal policy for long-term reward maximization.

The objective is to maximise a performance metric (which acts as the Reward), such as the Max Rate (aggregated throughput), a utility function like Proportional Fairness, or a custom Standard/Normalized Reward based on the Goodput Factor to ensure fairness.

4.2.1.1 Assumptions

To reduce the computational complexity of the resource allocation problem and accelerate the ML training process due to hardware limitations, four main assumptions or simplifications are established for this approach:

- **Equal RU size:** Although the IEEE 802.11ax standard allows allocating Resource Units (RUs) of varying sizes within the same TXOP, it is assumed that all available RUs in each TXOP will be of the exact same size. For example, an 80 MHz channel will be divided equally, such as into two 484-tone RUs, four 242-tone RUs, eight 106-tone RUs, etc.
- **Equal RU frequency response:** The typical OFDMA advantage of assigning the specific RU with the lowest attenuation to each station will be omitted, as exploiting this feature would require specific hardware on the AP. By assuming that all RUs have a similar frequency response for all stations, the model is simplified so that the AP only needs to decide which stations will transmit, without evaluating which specific RU is assigned to each one.
- **Single data phase per TXOP:** Normally, an AP can transmit multiple consecutive data frames (known as data phases) during the duration of a single TXOP. To further reduce the decision-making options for the model, it is assumed that only one data phase will occur per TXOP.

Limitation to 4 Stations: The combinatorial nature of the scheduling problem makes exhaustive search impractical for real-time operation. For example, assuming an 80 MHz channel with 4 stations and equal-sized RUs, the action space already consists of 11 possible allocations. If a larger number of stations and variable RU sizes were considered, this action space would grow explosively (as detailed in Section 4.2.2). This elevated complexity motivates the application of Reinforcement Learning, which enables the Agent to discover optimal policies through trial-and-error interaction with the network environment.

4.2.2 Reinforcement Learning Formulation

The scheduling problem is modelled as a Markov Decision Process with the following components:

1. **State space⁵:** The state is represented as a $3 \times N$ matrix, where N denotes the number of stations, comprising the Air Time Ratio (ATR), the averaged Goodput Factor (GF), and the buffer status for each station. The state defines which aspects of the network the Agent observes before making each decision. To ensure effective OFDMA allocation, it is essential to select relevant numerical features that capture meaningful information about network conditions. The magnitudes included in this matrix have been chosen for this purpose.

⁵ State space determines which aspects of the network will be observed by the Agent before taking each decision.

- **Air Time Ratio (ATR):** The ATR_i estimated for a STA_i is defined as the ratio between the measured Input Traffic of the STA and its theoretical transmission rate at the PHY level, assuming a 26-tones RU. In this way, this ATR provides information about the STA transmission requirements, versus its transmission capacity in 26-RU units, which depends on its current MCS and Number of Spatial Streams (NSS). Each sample of measured Input Traffic evaluates the transmitted bytes from the application over a window time. Then, each sample is averaged over an α filter, to obtain the input throughput metric. The expressions are given in Equations 4.8 and 4.9.

$$ATR_i = \frac{InputThroughput\ STA_i}{PhyRate(MCS_i, NSS_i, 26RU)} \quad (4.8)$$

With:

$$InputThroughput\ STA_i(t) = \alpha * InputThroughput\ STA_i(t) + (1 - \alpha) * \overline{InputThroughput\ STA_i(t - 1)} \quad (4.9)$$

- **Goodput Factor (GF_i):** The GF is computed as an exponentially weighted moving average of the ratio between delivered throughput and input throughput, reflects the long-term service satisfaction of each station. The Equations are presented in 4.10 and 4.11.

$$\overline{GF_i(t)} = \alpha \cdot GF_i(t) + (1 - \alpha) \cdot \overline{GF_i(t - 1)} \quad (4.10)$$

With:

$$GF_i = \frac{Throughput\ STA_i}{InputThroughput\ STA_i} \quad (4.11)$$

- **Buffer status:** The buffer of STA_i is the total size in bytes of the enqueued packets to be sent to STA. This measurement indicates how close is a STA of being congested.
2. **Action space:** It must encompass all valid RU allocations permitted by the network's constraints. By assuming equal RU sizes and a similar frequency response across all RUs, the Access Point's decision is significantly simplified: it only needs to select which Stations (STAs) will transmit during the next Transmit Opportunity (TXOP). Because the specific RU assigned to each selected STA is arbitrary, this allocation process effectively becomes a task of grouping the STAs.

The Group Size (GS), defined as the number of STAs scheduled for the upcoming TXOP, dictates the total number of possible allocation combinations. For instance, in a network with 4 STAs operating on an 80 MHz channel (using a 996-tone RU), the valid group sizes are restricted to $GS \in 1, 2, 4$. Ultimately, each specific allocation is mathematically represented as an n -component binary vector, where a value of "1" denotes that the STA is allocated to transmit, and a "0" signifies no allocation.

Group Size	Allocation Vector	Action Number
1	[1, 0, 0, 0]	0
	[0, 1, 0, 0]	1
	[0, 0, 1, 0]	2
	[0, 0, 0, 1]	3
2	[1, 1, 0, 0]	4
	[1, 0, 1, 0]	5
	[0, 1, 1, 0]	6
	[1, 0, 0, 1]	7
	[0, 1, 0, 1]	8
	[0, 0, 1, 1]	9
4	[1, 1, 1, 1]	10

Table 4: Action space for 4 STAs

Group size 1 corresponds to single-user transmissions where one station receives the entire 996-tone RU. Group size 2 splits the channel into two 484-tone RUs. Group size 4 divides the channel into four 242-tone RUs. The group size 3 is not available in this configuration because the 996-tone RU cannot be evenly divided into three valid RU sizes per the standard's tree decomposition rules.

3. Reward function: It is a numeric value that correctly represents the desired network performance to be maximised by the Agent. To properly evaluate the OFDMA scheduling problem, four different reward functions are considered:

- **Max Rate:** A standard utility function from the literature in [21], that maximises the linear sum of the delivered throughputs across all stations. While it effectively achieves high aggregated throughput, it may lead to starvation for stations with lower traffic demands, as the Agent tends to repeatedly select stations with higher traffic. As shown in Equation 4.12.

$$R_{\max rate}(s, a) = \sum_{i=1}^n \text{Throughput}(STA_i) \quad (4.12)$$

- **Proportional Fairness (PF):** Another metric from [21], that introduces fairness by maximizing the sum of the logarithms of the throughputs, rather than their linear sum. Despite improving general fairness, this function does not account for the specific input traffic (actual needs) of each station. As shown in Equation 4.13.

$$R_{PF}(s, a) = \sum_{i=1}^n \log_{10} \text{Throughput}(STA_i) \quad (4.13)$$

- **Standard Reward (SR) - Proposed:** A custom function defined specifically for this study to address the limitations of previous metrics. It modifies the Max Rate approach by incorporating a fairness multiplying factor, which is the ratio between the minimum and maximum Goodput Factor (GF) of the stations. This formulation, shown in Equation 4.14, encourages the Agent to transmit as much data as possible

while effectively ensuring fairness based on the actual traffic requirements of each station.

$$R_{SR}(s, a) = \frac{\min_i(\overline{GF_i})}{\max_i(\overline{GF_i})} \cdot \sum_{i=1}^n \text{Throughput}(\text{STA}_i) \quad (4.14)$$

- **Normalized Reward (NR) - Proposed:** A variant of the Standard Reward designed in for this study to prevent different network use cases from having drastically different maximum reward values. In Equation 4.15, the delivered throughput of each station is normalized by its theoretical maximum PHY rate, and the total sum is divided by the group size. This normalization minimizes the differences in achievable rewards across scenarios, which is a highly recommended practice to improve the stability of the model's training process.

$$R_{NR}(s, a) = \frac{\min_i(\overline{GF_i})}{\max_i(\overline{GF_i})} \cdot \frac{1}{\text{GroupSize}} \cdot \sum_{i=1}^n \frac{\text{Throughput}(\text{STA}_i)}{\text{PHYRate}(\text{MCS}_i, \text{NSS}_i, \text{RU}_i)} \quad (4.15)$$

4.2.3 Deep Q-Network (DQN) Architecture

The Reinforcement Learning algorithm employed is the Deep Q-Network (DQN) [27] operating in Double DQN mode [28]. DQN replaces the classical Q-Learning lookup table with an Artificial Neural Network (ANN) that approximates the optimal Q-function, enabling the model to handle continuous state spaces and large state-action dimensions. The Double DQN variant addresses the Q-value overestimation problem inherent in the original DQN by decoupling action selection from Q-value estimation.

Two key mechanisms ensure training stability: the Target Network, whose parameters are periodically copied from the Q-Network to provide stable target values during backpropagation, and the Experience Replay Memory, which stores transitions (state, action, reward, next state) and randomly samples mini-batches to break temporal correlations in sequential observations.

The neural network architecture is a Feed Forward Network with fully connected layers. The input layer has 12 neurons (corresponding to the 3 state variables per each of the 4 STAs). Four hidden layers with 8, 16, 32, and 16 neurons respectively use the ReLU activation function, with Batch Normalisation applied between each hidden layer and its activation to reduce covariance shift [29]. The output layer has 11 neurons (one per possible allocation action) with no activation function, as the DQN performs a regression estimation of Q-values.

Layer	Configuration
Input Layer	12 inputs (3 states x 4 STAs)
Hidden Layer 1	8 neurons, ReLU + Batch Norm
Hidden Layer 2	16 neurons, ReLU + Batch Norm
Hidden Layer 3	32 neurons, ReLU + Batch Norm
Hidden Layer 4	16 neurons, ReLU + Batch Norm
Output Layer	11 outputs (Q-values per action number)

Table 5: Double DQN Network Configuration Layer

4.2.3.1 Training Hyperparameters

The exploration policy employed is an ϵ -greedy strategy⁶ with linear annealing. An initial warm-up phase of 50,000 random actions ($\epsilon = 1.0$) is conducted to populate the experience replay memory, which has a total capacity of 200,000 transitions⁷, before formal learning begins. Subsequently, the ϵ value is linearly decreased from 1.0 to 0.1 over 500,000 actions, after which it remains constant at 0.1 for the remaining 3,000,000 training actions. Finally, during testing on trained cases, a fully greedy policy ($\epsilon = 0$) is applied to exclusively evaluate the purely acquired knowledge.

Hyperparameter	Value
Total training actions	3,000,000
Warm-up actions	50,000
ϵ (warm-up)	1
ϵ (initial / final) in Linear Decreasing	1.0 / 0.1
ϵ annealing steps	500,000
Experience replay memory size	200,000 transitions
DQN variant	Double DQN
Mini-batch size	512
Target network update interval	Every 10,000 actions
Optimizer	Adam
Learning rate	0.001
Loss function	Mean Squared Error (MSE)

Table 6: Double DQN Network Hyperparameter

For non-trained case tests in dynamic mode, $\epsilon = 0.1$ is maintained to allow online learning capability: a deployed AP can not only execute a pre-trained policy but also continue adapting to its specific environment, traffic patterns, and interference conditions in real-time.

4.2.3.2 Training RL Agents

To assess the training progress, learning curves were monitored for each RL agent. The four evaluated models (Max Rate, Proportional Fairness, Standard Reward, and Normalized Reward) demonstrated stable and predominantly monotonically increasing trajectories, achieving convergence near the 2,000,000-action mark. The absence of divergence across all curves validates the suitability of the chosen DQN hyperparameters and confirms the overall success of the training process.

⁶ The ϵ -greedy strategy selects a random action with probability ϵ (exploration) and the best-known action with probability $1 - \epsilon$ (exploitation).

⁷ A transition is defined as a tuple containing the initial State, the taken Action, the obtained Reward, and the resulting State.

5. Results

5.1 NS-3 Simulation Environment

Both AI strategies presented in this deliverable were implemented and validated using the NS-3 discrete-event network simulator [15]. The integration of ML algorithms with NS-3 required a dual-language architecture combining C++ (for the network simulation) with Python (for the AI/ML frameworks) [30], enabled by the NS3-AI inter-process communication module. This Section describes each component of the simulation environment.

5.1.1 NS-3 Simulator

NS-3 is described on its official website as a "discrete-event, general-purpose network simulator for Internet systems." Developed in C++, NS-3 is an open-source project distributed under the GNU GPLv2 license. Its publicly available source code has contributed to its widespread adoption by both the research community and industry.

The NS-3 simulator is selected for the present project not only due to its popularity, but also because it offers several implementations of IEEE 802.3, IEEE 802.11ax, and IEEE 802.11be features. Furthermore, NS-3 provides fine-grained control over various aspects of the PHY and MAC layers, allowing users to configure detailed parameters to suit specific simulation needs. Importantly, its open-source nature enables users to develop and integrate custom modules, which can introduce new simulation capabilities, such as emulating hardware constraints that a real AP might impose. This flexibility is particularly valuable in a project conducted within an industrial context.

5.1.2 NS3-AI Module

The NS3-AI [31] module establishes a bidirectional interaction between the NS-3 network simulator and ML frameworks through a shared-memory pool. This design enables the simulation environment to leverage advanced AI models for decision-making and optimisation tasks in real-time.

As shown in Figure 5, the NS-3 simulator, implemented in C++, represents the network environment where nodes and protocols operate.

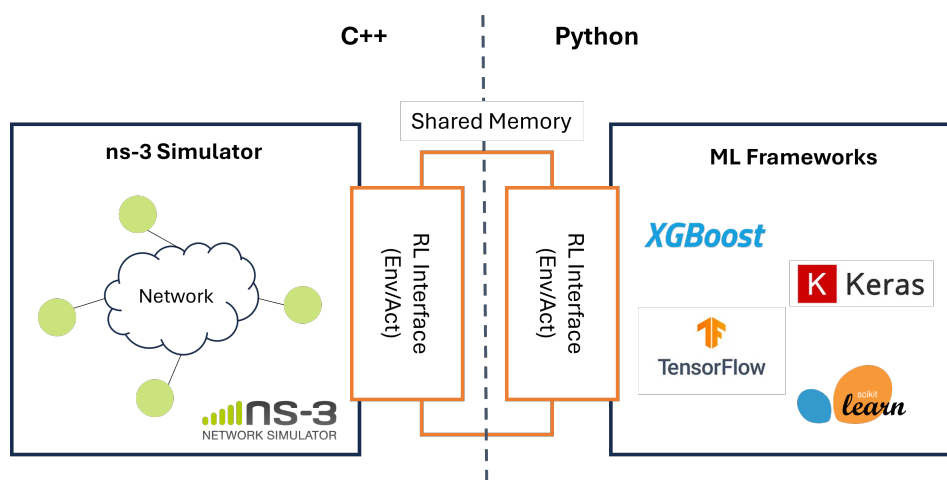


Figure 5: NS3-AI Structure

The RL Interface (Env/Act) exposes the simulation state "Environment" and accepts "Actions" from the learning agent⁸ [32]. The ML component is implemented in Python, providing access to the frameworks described in Table 7:

Tool(s)	Main role	Target	Typical use
Scikit-learn [33]	Classic ML	A wide set of algorithms	Baselines, model selection, classical ML
TensorFlow (Keras) [34]	Deep learning	Neural networks	Images, sequences, text, or deep models
XGBoost [35]	Boosted trees	Gradient boosting	Competitive tabular models and leaderboards

Table 7: ML Framework using NS3-AI

5.1.3 Integration of AI Strategies with NS-3

The dual-language architecture described above supports the two AI strategies developed in this deliverable, each with a distinct data flow:

- **AI-Based Traffic Flow Classifier** (XGBoost): NS-3 simulations can generate packet-level traces that are processed by the feature extraction pipeline implemented in Python through the NS3-AI module. The classifier was trained offline on Ethernet-channel captures of different services and then applied in real-time to Wi-Fi simulation outputs. The classification result determines the best-suited EDCA Access Category for each downlink flow, which is fed back into the NS-3 realm and influences the AP's queuing.

The simulation environment supports multiple OBSS configurations with multiple APs and stations sharing the same channel, enabling evaluation of the classifier under realistic interference conditions.

- **AI-Based Multi-User OFDMA Scheduler** (Double DQN): The integration uses the same NS3-AI module, translating network observations into states, forwarding actions to the OFDMA scheduler, and computing rewards from simulation feedback. The Double DQN network is implemented using TensorFlow/Keras as the DL backend. This architecture allows the agent to interact with NS-3 in real-time, receiving state observations after each TXOP and providing allocation decisions for the next transmission.

⁸ Although derived from RL environment APIs, the NS3-AI RL interface is employed in this project purely as an inter-process communication layer between NS-3 (C++) and Python, thereby supporting the exchange of any structured data through the shared-memory protocol.

5.2 AI-Based Traffic Flow Classifier Simulation Results

This Section presents the simulation results obtained from the evaluation of the AI-Based Traffic Flow Classifier described in Section 4.1.

5.2.1 Simulation Scenario

The experimental setup comprises two Wi-Fi congestion scenarios, both designed to stress-test the classifier under ultra-dense network conditions. The simulation employs a lossless channel model, which eliminates path loss effects and removes any dependency between physical distance and received signal quality. All wireless links share identical parameters: EHT format, MCS 5, 160 MHz bandwidth, 800 ns guard interval, a single spatial stream (NSS = 1), and operation in the 5 GHz band.

The interference environment is modelled through a uniform OBSS interference matrix, where all node pairs (both intra-BSS and inter-BSS) experience a constant interference level of -50 dB. The packet detection (PD) and energy detection (ED) thresholds are set to -80 dBm and -70 dBm, respectively, with a fixed received power differential (DiffRxPwr) of 30 dB, as summarized in Table 8.

Under this configuration, since the channel model is lossless and distance-independent, each STA operates at a fixed SNR determined solely by the uniform interference level rather than by its physical position.

Parameter	Value	Implication
Wi-Fi Standard	IEEE 802.11be (EHT)	
Frequency Band	5 GHz	
Channel Bandwidth	160 MHz	Fixed for all links
MCS Index	5	Uniform link quality across all STAs
Number of Spatial Streams (NSS)	1	Single spatial stream
Guard Interval (GI)	800 ns	Fixed for all links
OBSS interference	-50 dB (uniform)	Fixed for all STAs
Channel Model	Lossless	No distance-dependent attenuation
PDThreshold	-80 dBm	Packet detection sensitivity
ED threshold	-70 dBm	Energy detection sensitivity
DiffRxPwr	30 dB	Fixed received power differential
Theoretical PHY Rate	576.6 Mbps	
Max A-MPDU Size (AP)	128,000 bytes	
Max A-MPDU Size (STAs)	64,000 bytes	
EDCA - VO AC (CWmin/CWmax/AIFSN)	3 / 7 / 2	
EDCA - VI AC (CWmin/CWmax/AIFSN)	7 / 15 / 2	
EDCA - BE AC (CWmin/CWmax/AIFSN)	15 / 1,023 / 3	
EDCA - BK AC (CWmin/CWmax/AIFSN)	15 / 1,023 / 7	
Scheduler (AP)	AP Trigger-Based EDCA	
Scheduler (STA)	EDCA	
Simulation Time	60 s	
Traffic Start Time	t = 2 s	

Table 8: Simulation parameters for scenarios

The initial 2 seconds of the simulation are excluded from the ML model, as they primarily consist of Wi-Fi control packets that are irrelevant for training or testing purposes. Since the application begins operating at the two-second mark, the first usable data vector is generated at second 7⁹. Subsequent vectors are extracted using a five-second sliding window with a one-second step, resulting in a total of 16-feature vectors. These features are then fed into the ML model, which returns a classification and their corresponding EDCA Access Categories (Video, Voice, Best Effort) based on real-time predictions.

5.2.1.1 Metrics

- **Throughput Received:** measures the amount of useful data successfully received per unit of time, expressed in megabits per second.

$$\text{Throughput [Mbps]} = \frac{\text{Bytes} \times 8}{10^6 \times \text{Duration}_{\text{seconds}}}$$

Where:

- Bytes = Number of bytes received during the application simulation time (in bytes).
- Duration = Time of the application simulation (in seconds).

- **Average Packet Delay (Avg. Delay):** The average packet delay represents the mean time it takes for a packet to be delivered from the moment it is sent until it is received.

$$\text{Average Packet Delay} = \frac{\sum_{i=1}^N \text{delay}_i}{N}$$

Where:

- delay_i = It is the individual delay of the packet i , calculated as the time difference between the packet reception and the packet transmission, expressed in milliseconds.
- N = total number of received packets.

⁹ This means that the first classification is performed at second 7, after which the system continuously classifies each flow every second based on its corresponding 5-tuple (source IP, destination IP, source port, destination port, protocol).

5.2.2 Evaluation Results on Simulation Scenarios

The evaluation compares a baseline configuration in which all flows remain in Best Effort (BE) against the proposed ML-based prioritisation which relies on forced UL-BE queue lookup in the AP scheduler

- Scenario 1 considers a single AP with heavy local contention.
- Scenario 2 introduces OBSS by activating a second AP with eight interfering uplink BULK_UDP sources, and an additional 300 Mbps downlink BULK_UDP source inside AP1.

All metrics are computed over 58 seconds of active application traffic.

5.2.2.1 Scenario 1: Single AP with Heavy Local Contention

A single BSS with 1 AP (AP1) and 10 associated STAs.

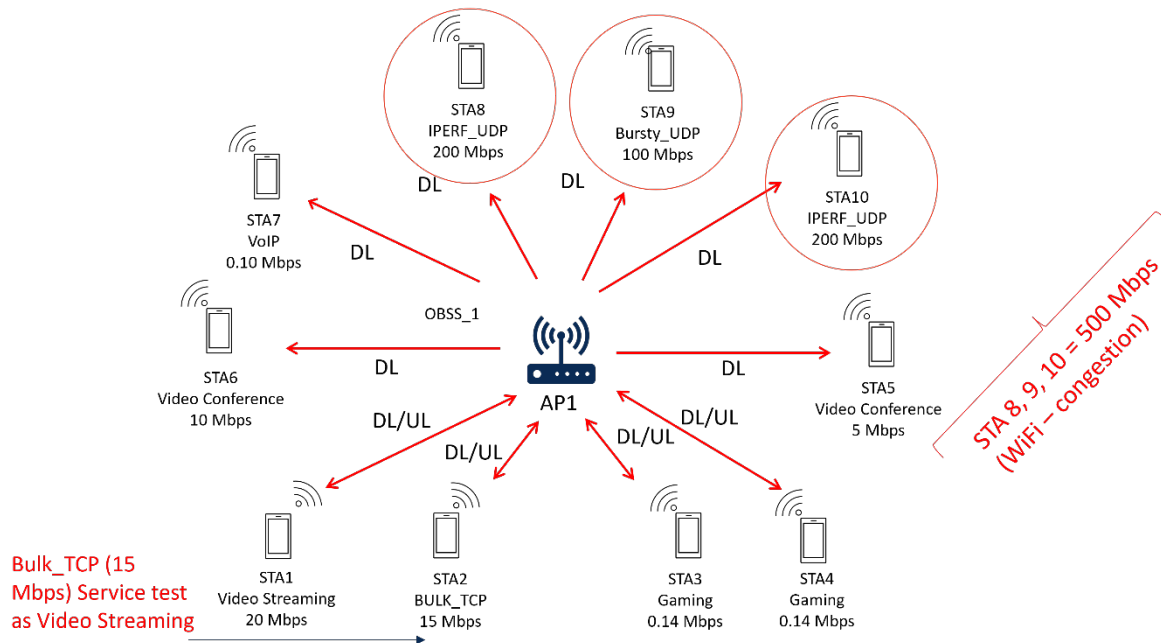


Figure 6: Scenario 1 - A single BSS with 1 AP (AP1) and 10 associated STAs. The red outlines highlight the specific STAs inducing channel congestion by injecting 500 Mbps of intensive UDP traffic (IPERF and Bursty) into the network.

STA #	App Service	Link Direction	Data Rate [Mbps]	Access Category
STA 1	Video_Streaming	DL/UL (ack)	20	VI
STA 2	BULK_TCP	DL/UL (ack)	15	BE
STA 3	Gaming	DL/UL	0.14	VI
STA 4	Gaming	DL/UL	0.14	VI
STA 5	Video_Conference	DL	5	VI
STA 6	Video_Conference	DL	10	VI
STA 7	VoIP	DL	0.1	VO
STA 8	IPERF_UDP	DL	200	BE
STA 9	Bursty_UDP	DL	100	BE
STA 10	IPERF_UDP	DL	200	BE

Table 9: Application Configuration for Scenario 1

This scenario stresses a single BSS under conditions where the aggregate offered load approaches or exceeds the effective wireless channel capacity, as AP1 must simultaneously serve video streaming, gaming, video conferencing, VoIP, and high-rate bulk UDP/Iperf traffic.

The Bulk TCP flow at 15 Mbps is intentionally included to evaluate whether the traffic classifier correctly identifies it as a background data transfer and does not misclassify it as Video Streaming, given the similarity in bitrate, and TCP protocol.

STA #	App Service	BE	ML-Class	Thpt Gain [%]	BE	ML-Class	Delay Change [%]
		RX Thpt [Mbps]	RX Thpt [Mbps]		RX Delay [ms]	RX Delay [ms]	
STA 1	Video_Streaming	0.79	20.15	2447.92	1394.00	5.00	-99.64
STA 2	BULK_TCP	14.91	1.29	-91.38	297.00	453.00	52.53
STA 3	Gaming	0.14	0.14	0.00	295.00	26.00	-91.19
STA 4	Gaming	0.14	0.14	0.01	6.00	1.00	-83.33
STA 5	Video_Conference	5.58	5.58	0.00	8.00	1.00	-87.50
STA 6	Video_Conference	10.96	10.97	0.03	11.00	2.00	-81.82
STA 7	VoIP	0.12	0.12	0.01	839.00	84.00	-89.99
STA 8	IPERF_UDP	196.02	110.11	-43.83	26.00	49.00	88.46
STA 9	Bursty_UDP	11.05	10.27	-7.11	447.00	498.00	11.41
STA 10	IPERF_UDP	196.18	57.62	-70.63	26.00	97.00	273.08
Total		435.89	216.36				

Table 10: Scenario 1: BE Baseline vs. ML-Based Classification.

As shown in Table 10, the applications targeted for priority changes were highlighted. The contrast between BE-only operation and ML-based prioritisation was significant across all traffic classes, which revealed the benefits and the inherent trade-offs of active QoS enforcement in a heavily loaded single BSS environment.

- Video Streaming (STA 1): Under BE-only operation, the Video Streaming flow is severely starved by the high-throughput UDP flows competing for the same channel, achieving only 0.79 Mbps out of the 20 Mbps offered load and suffering an end-to-end delay of 1394 ms, far beyond any practical playback threshold. With ML-based prioritisation, the flow is correctly identified and elevated to the VI access category, recovering full throughput at 20.15 Mbps and reducing delay by 99.64% to just 5 ms.

This represents the most dramatic improvement in the scenario and validates the classifier's ability to distinguish sustained video delivery from generic bulk data traffic.

- Bulk TCP (STA 2): The Bulk TCP flow is correctly classified and assigned to the Best Effort AC (intentionally deprioritized). In comparison, its throughput drops from 14.91 Mbps to 1.29 Mbps, and its delay increases from 297 ms to 453 ms. This degradation is expected and desired behaviour: by confining the Bulk TCP flow to lower-priority queues with larger contention window parameters, the scheduler frees channel time for latency-sensitive services.
- Gaming (STAs 3 and 4): The meaningful improvement lies in delay: STA 3 sees its delay reduced from 295 ms to 26 ms (-91.19%), and STA 4 from 6 ms to 1 ms (-83.33%).

- Video Conferencing (STAs 5 and 6): The impact of ML classification is primarily reflected in delay: STA 5 drops from 8 ms to 1 ms (-87.50%), and STA 6 from 11 ms to 2 ms (-81.82%). While the absolute delay values under BE were already below the critical threshold for Video Conferencing, the ML-based scheduler consistently reduces jitter exposure, resulting in more stable performance.
- VoIP (STA 7): Under BE-only operation, it experiences a delay of 839 ms, more than eight times the 100 ms threshold commonly used as the upper bound for acceptable voice quality (ITU-T G.114). With ML prioritisation, the flow is elevated to the VO access category and delay is reduced to 84 ms (-89.99%), bringing it into an acceptable range.

5.2.2.2 Scenario 2: OBSS Congestion

An Overlapping Basic Service Set (OBSS) environment comprising two access points (AP1 and AP2) and 13 associated STAs.

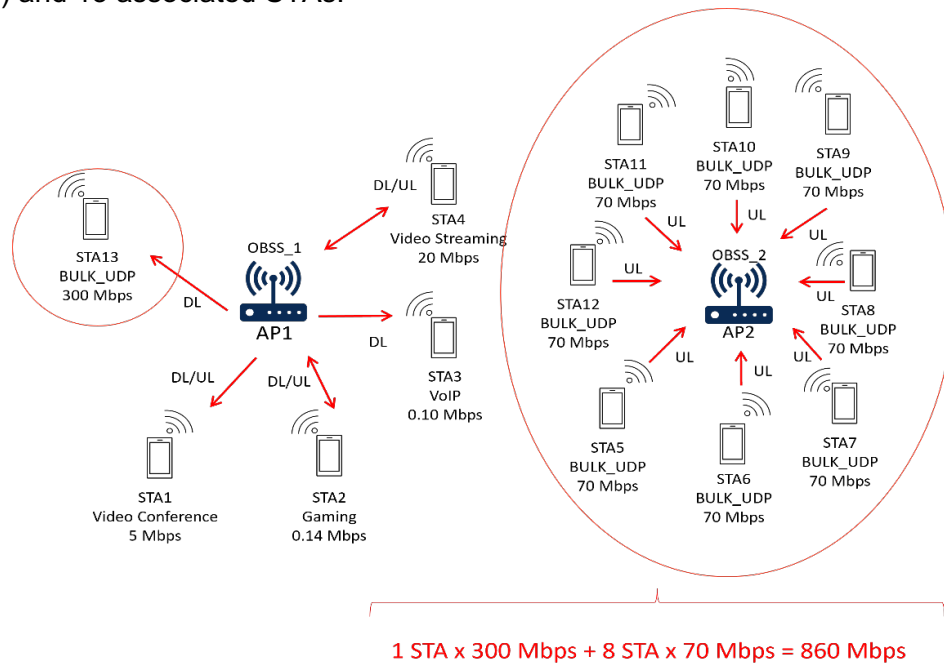


Figure 7: Scenario 2 - An OBSS topology with two APs. The red markings denote the congestion-generating stations: one STA on AP1 injecting 300 Mbps and eight STAs on AP2 each injecting 70 Mbps, resulting in a combined offered load of 860 Mbps that saturates the shared wireless medium.

STA #	App Service	Link Direction	Data Rate [Mbps]	Access Category	AP Connected
STA 1	Video_Conference	DL	5	VI	AP1
STA 2	Gaming	DL/UL	0.14	VI	AP1
STA 3	VoIP	DL	0.1	VO	AP1
STA 4	Video_Streaming	DL/UL (ack)	20	VI	AP1
STA 13	Bulk_UDP	DL	300	BE	AP1
STA 5	Bulk_UDP	UL	70	-	AP2
STA 6	Bulk_UDP	UL	70	-	AP2
STA 7	Bulk_UDP	UL	70	-	AP2
STA 8	Bulk_UDP	UL	70	-	AP2
STA 9	Bulk_UDP	UL	70	-	AP2
STA 10	Bulk_UDP	UL	70	-	AP2
STA 11	Bulk_UDP	UL	70	-	AP2
STA 12	Bulk_UDP	UL	70	-	AP2

Table 11: Application Configuration for Scenario 2.

This scenario evaluates ML-based QoS classification under OBSS interference conditions, where AP1 simultaneously serves latency-sensitive multimedia flows alongside a high-rate downlink Bulk UDP source (STA 13, 300 Mbps). The scenario is particularly challenging because Bulk UDP traffic of this profile was not included in the original classifier training set, making it a direct test of the model's generalisation capability.

The eight Bulk UDP stations associated with AP2 introduce sustained cross-BSS contention, further stressing the medium and compressing the available airtime for AP1's protected flows.

STA #	App Service	BE	ML-Class	Thpt Gain [%]	BE	ML-Class	Delay Change [%]
		RX Thpt [Mbps]	RX Thpt [Mbps]		RX Delay [ms]	RX Delay [ms]	
STA 1	Video_Conference	5.35	5.58	4.32	137	12	-91.24
STA 2	Gaming	0.14	0.14	0.03	340	68	-80.00
STA 3	VoIP	0.12	0.12	0.00	14	1	-92.86
STA 4	Video_Streaming	8.46	21.68	156.27	159	11	-93.08
STA 13	Bulk_UDP	131.36	111.90	-14.82	43	51	18.60
STA 5	Bulk_UDP	28.38	23.24	-18.11	203	250	23.15
STA 6	Bulk_UDP	29.44	22.30	-24.25	195	259	32.82
STA 7	Bulk_UDP	31.15	22.53	-27.67	183	257	40.44
STA 8	Bulk_UDP	29.06	20.73	-28.66	197	281	42.64
STA 9	Bulk_UDP	32.39	23.37	-27.85	176	248	40.91
STA 10	Bulk_UDP	32.17	21.96	-31.74	178	263	47.75
STA 11	Bulk_UDP	30.84	19.20	-37.74	186	296	59.14
STA 12	Bulk_UDP	29.07	19.15	-34.12	198	303	53.03
Total		387.92	311.89				

Table 12: Scenario 2 - BE Baseline vs. ML-Based Classification Results

The results confirm that the classifier generalises effectively to the unseen Bulk UDP profile and successfully protects the latency-sensitive multimedia flows from displacement.

The video streaming flow (STA 4) is the most notable beneficiary: under BE-only operation it achieves only 8.46 Mbps with a mean delay of 159 ms, whereas ML-based prioritisation elevates it to the VI access category, recovering full throughput at 21.68 Mbps and reducing delay by 93.08% to just 11 ms. Similarly, the video conferencing flow (STA 1) improves from 137 ms to 12 ms (-91.24%), gaming (STA 2) from 340 ms to 68 ms (-80.00%), and VoIP (STA 3) from 14 ms to 1 ms (-92.86%).

Regarding the unseen Bulk UDP traffic at STA 13, the classifier mapped it to the closest known behavioural profile (Bursty UDP) and correctly retained it in the BE access category. This prevents it from consuming high-priority queue resources, though the resulting throughput reduction from 131.36 Mbps to 111.90 Mbps (-14.82%) and modest delay increase are expected trade-offs of active QoS enforcement.

The same pattern is observed across the AP2 Bulk UDP stations (STAs 5 to 12), which collectively experience reduced throughput and slightly higher delays as channel time is redistributed in favour of the protected sensitive flows. This behaviour is consistent with the design intent: background bulk traffic is intentionally constrained to preserve service quality for interactive and real-time applications.

5.2.2.3 Average Aggregate Throughput and Delay Trade-off

ML-based prioritisation introduces a clear trade-off between latency reduction and aggregate throughput.

In Scenario 1, latency-sensitive flows benefit from substantial delay reductions: video streaming latency decreases from 1394 ms to 5 ms, accompanied by an increase in received throughput from 0.79 Mbps to 20.15 Mbps (+2447.92%). Gaming latency is reduced from 295 ms to 26 ms, while VoIP latency drops from 839 ms to 84 ms.

However, this improvement comes at the expense of aggregate received throughput, which decreases from 435.89 Mbps to 216.36 Mbps, representing a 50% reduction. This degradation is due to the prioritization of real-time flows into higher access categories, which reduces the probability of channel access for Best Effort traffic.

Scenario 2 exhibits the same behaviour under OBSS conditions. Latency-sensitive flows also experience substantial delay reductions, with Video Streaming delay decreasing from 159 ms to 11 ms, Video Conferencing from 137 ms to 12 ms, Gaming from 340 ms to 68 ms, and VoIP from 14 ms to 1 ms. In addition, Video Streaming throughput increases from 8.46 Mbps to 21.68 Mbps, while Video Conferencing throughput slightly improves from 5.35 Mbps to 5.58 Mbps. Nevertheless, these gains come at the expense of aggregate received throughput, which decreases from 387.92 Mbps to 311.89 Mbps (-19.6%).

In both scenarios, the throughput surrendered by background flows is not wasted but effectively redistributed as airtime for the protected services, confirming that the scheduler trades raw capacity for deterministic latency guarantees where they are most needed.

5.2.2.4 Feature Robustness Under Congested Conditions

During the first 7 seconds of simulation time (5 seconds of active application traffic), the AP operates without ML-driven prioritisation, and all flows compete under standard Best Effort EDCA rules. This initial window allows the classifier to passively accumulate the flow-level feature measurements required to produce predictions. From second 7 onward, the classifier becomes active, and the AP assigns each flow to its corresponding access category, at which point EDCA prioritisation takes effect according to the ML output.

A key observation from this design is that the selected feature set demonstrates robustness under both pre-prioritisation (all flows competing in BE) and post-prioritisation conditions (flows separated into different EDCA ACs). During the first phase, features are extracted from unprioritized, heavily congested traffic, a scenario where inter-packet delays, queue depths, and flow statistics are significantly distorted by contention. During the second phase, features are extracted from traffic that is already being shaped by QoS enforcement, altering the statistical profile of each flow, with respect to the pre-prioritisation phase.

The fact that the classifier achieves 100% prediction accuracy across all flows in both regimes confirms that the chosen features capture application-layer behavioural signatures that remain discriminative regardless of the underlying MAC-layer scheduling state. This property is non-trivial: it indicates that the model generalises beyond the controlled conditions of its training environment and can operate reliably in dynamic, congestion scenarios where prioritisation state may change over time.

5.2.3 Conclusions for AI-Based Traffic Flow Classifier

The simulation results presented in this Section confirm that the proposed AI-based traffic flow classifier effectively addresses the QoS degradation experienced by latency-sensitive applications under heavily congested Wi-Fi environments. Across both evaluation scenarios, the classifier achieved 100% prediction accuracy, correctly identifying and prioritising video streaming, video conferencing, gaming, and VoIP flows while appropriately retaining bulk traffic in the Best Effort access category.

The most significant improvements are observed in packet delay. In Scenario 1 (single BSS with heavy local contention), the ML-based classification achieved an average delay reduction of approximately 88.9% across the protected flows, bringing all latency-sensitive services within acceptable thresholds. This improvement was accompanied by a substantial increase in Video Streaming throughput of 2447.92%. In Scenario 2 (OBSS congestion), a comparable average delay reduction of approximately 89.3% was achieved, demonstrating that the classifier maintains its effectiveness even under inter-BSS contention conditions. In this case as well, Video Streaming throughput showed a significant increase of 156.27%.

The observed reduction in aggregate throughput, 50.4% in Scenario 1 and 19.6% in Scenario 2, does not represent a degradation of network capability but rather a deliberate reallocation of channel resources away from bandwidth-dominant bulk flows toward interactive and real-time services.

From a user-experience perspective, this trade-off is justified: under BE-only operation, latency-sensitive flows suffered delays exceeding acceptable quality thresholds by an order of magnitude, whereas ML-based prioritisation restored them to near-target performance levels.

Furthermore, the classifier demonstrated two important generalisation properties. First, the selected feature set proved robust across both unprioritized and prioritised Wi-Fi traffic regimes, indicating that the model captures application-layer behavioural signatures that remain discriminative regardless of the underlying MAC-layer scheduling state.

Second, the model successfully classified unseen traffic profiles (specifically the 300 Mbps Bulk UDP flow in Scenario 2), by mapping them to the closest known behavioural category, confirming that the classifier does not merely memorise training patterns but generalises to previously unobserved traffic conditions.

These results validate the feasibility of integrating real-time ML-based traffic classification into the Wi-Fi MAC layer as a practical mechanism for automated QoS enforcement in ultra-dense deployment scenarios.

5.3 AI-Based Multi-User Scheduler Simulation Results

This Section presents the simulation results obtained from the evaluation of the RL based OFDMA multi-user scheduler described in Section 4.2. The experiments were designed to assess three core capabilities: (i) resolution of trained network scenarios, (ii) effectiveness of the proposed reward functions, and (iii) generalisation to previously unseen network conditions. All results reported in this Section are derived from experiments conducted on the NS-3 simulator integrated with the Double DQN network via the NS3-AI module.

5.3.1 Simulation Scenario

An AP controlled by the RL agent is instantiated together with 4 associated STAs, all operating with the OFDMA feature enabled over a single spatial stream and under the EDCA protocol. Only traffic belonging to the Best Effort AC is employed in the simulations.

The STAs are placed equidistantly at 10 metres from the AP in a circular arrangement with static positions. Only downlink UDP traffic is considered.

The total simulation time is 6 seconds. During the first second, each STA sends 500 kbps of uplink traffic for association purposes. From seconds 2 to 3, the AP operates in single-user mode to allow metrics (input throughput, ATR, averaged GF) to converge. From seconds 3 to 6, the OFDMA feature is enabled, and the RL Agent begins making allocation decisions. No rate control algorithm operates during the simulations; the MCS per STA remains fixed throughout.

Parameter / Setting	Value
Frequency Band	5 GHz
Channel Bandwidth	80 MHz
Channel Number	42
Control Frames MCS Index	3
Data Frames MCS Index	Dependent on use case
Input Throughput per STA	Dependent on use case
Input Throughput α	0.4
Goodput Factor α	0.1
Input Throughput Measuring Window	50 ms
Queue Max Size	500 packets per STA
Propagation Model	Friis propagation loss model
Error Correction	LDPC enabled
Aggregation	A-MSDU and A-MPDU enabled
Data Phases per TXOP	1 (TXOP Limit = 0)
Transport Layer Protocol	UDP
Distance AP - STA	10 m
Simulation Time	6 s (100 s for non-trained case)

Table 13: Simulation settings and parameter values

5.3.2 Use Cases

Three use cases were defined to evaluate the model under different network conditions.

- Two trained use cases (non-congested and congested).
- One non-trained use case was reserved exclusively for generalisation testing.

Use Case	Parameter	STA 1	STA 2	STA 3	STA 4
Non-congested (Trained)	MCS index	10	10	3	3
	Input Traffic (Mbps)	100	100	10	10
Congested (Trained)	MCS index	10	10	3	3
	Input Traffic (Mbps)	10	10	100	100
Non-trained	MCS index	7	7	7	7
	Input Traffic (Mbps)	100	100	10	10

Table 14: Use cases defined for the three evaluations

As shown in Table 14, in the non-congested case, STAs 1 and 2 use MCS 10 (1024-QAM, 3/4 coding rate) with 100 Mbps input traffic, while STAs 3 and 4 use MCS 3 (16-QAM, 1/2 coding rate) with 10 Mbps. In the congested case, the MCS assignment remains the same, but the input throughputs are swapped: the STAs with MCS 3 now receive 100 Mbps, which exceeds their PHY capacity, generating congestion. The non-trained case employs MCS 7 (64-QAM, 5/6) for all STAs with mixed throughputs (100 and 10 Mbps), placing the network at the boundary between congestion and non-congestion.

The resulting ATR values for each MCS and input throughput combination, which directly determine the state representation observed by the agent, are:

- MCS 3 at 10 Mbps: ATR = 3.33
- MCS 3 at 100 Mbps: ATR = 33.33
- MCS 7 at 10 Mbps: ATR = 1.33
- MCS 7 at 100 Mbps: ATR = 13.33
- MCS 10 at 10 Mbps: ATR = 0.88
- MCS 10 at 100 Mbps: ATR = 8.85

Note that in a Multi-User (MU) OFDMA transmission, the time duration of the data frames is equal for all participating STAs, with the station having the largest buffer determining the frame length. STAs with less data fill their payloads with padding. Grouping stations with similar ATR values minimises this padding overhead and improves transmission efficiency, which is the underlying principle that makes the MU Round Robin (RR) baseline a strong comparator.

5.3.3 Evaluation Results on Trained Use Cases

The results were compared against three baseline algorithms:

- Random Actions (uniform random sampling from the action space): With this comparison, if the obtained results are better than this algorithm, then we can be certain that the RL model is not taking random actions. This is a common technique in the RL field, as comparisons to random algorithms can be appreciated in important papers such as the DQN and Double DQN algorithms.

- Multi-User 4 STAs (all 4 STAs continuously transmitting together in OFDMA): In this mode, four stations (clients) are always scheduled simultaneously on the same OFDMA frame. The access point divides the channel into RUs and assigns one RU to each of the four STAs. All four transmit in parallel, continuously. This maximises spectral efficiency when all four have traffic to send, since every frame carries data from all four users.
- Multi-User Round Robin (STAs grouped by MCS-Input-Throughput pair, groups selected in round-robin manner): Here, stations are first classified into groups based on shared characteristics: their MCS index (modulation and coding scheme, which reflects link quality) and their offered input throughput. Stations with similar MCS and traffic demands are assigned to the same group. Then the AP cycles through these groups in a round-robin manner. Group A gets a TXOP, then group B, then group C, and so on.

The following RL agents (defined in Section 4.2.2) were evaluated in static mode (no online learning during the test, greedy policy with $\epsilon = 0$) on each of the use cases observed during training, and subsequently compared:

- Proportional Fairness (Log. Agent)
- Max Rate (Max Rate Agent)
- Standard non-normalized Reward proposal (Std. Agent)
- Normalized Reward proposal (Norm. Agent).

5.3.3.1 Non-Congested Scenario (Trained)

In this scenario (STA 1,2: MCS 10 at 100 Mbps; STA 3,4: MCS 3 at 10 Mbps), the network operates below its aggregate capacity. The most relevant metric is the global Goodput Factor, representing the percentage of correctly delivered data.

Table 15 compares two groups of models: on the one hand, the non-AI baselines, including Random actions, MU (4 STAs), and MU RR; and on the other hand, the AI models based on RL, which comprise the Log. Agent, Max Rate Agent, Std. Agent (proposed), and Norm. Agent (proposed).

Metric	# STA	Non-AI Baselines			AI Models (RL Agents, $\epsilon = 0$)			
		MU (4 STAs)	Rand. act.	MU RR	Log. Agent	Max Rate Agent	Std. Agent	Norm. Agent
Rx Rate (Mbps)	1	93.83	97.02	102.85	93.83	102.9	102.88	102.89
	2	93.80	96.20	102.84	93.80	102.89	102.92	102.87
	3	9.46	9.71	10.29	9.46	10.5	10.29	10.29
	4	9.46	9.68	10.29	9.46	10.42	10.29	10.29
Rx/Tx Goodput (%)	1	91.20%	94.30%	99.97%	91.20%	100%	100%	100%
	2	91.17%	93.50%	99.96%	91.17%	100%	100%	99.99%
	3	91.93%	94.36%	99.99%	91.93%	100%	99.99%	99.99%
	4	91.93%	94.07%	99.99%	91.93%	100%	99.99%	99.99%
Avg. Latency (ms)	1	15.02	12.50	1.81	15.02	1.63	1.71	1.36
	2	15.02	15.98	1.81	15.02	2.27	1.71	1.36
	3	4.07	4.64	1.49	4.07	31.02	1.72	2.58
	4	4.07	4.9	1.49	4.07	44.12	1.72	2.58

Table 15: Non-congested trained case results

The baseline algorithms MU 4 STAs and Random Actions failed to serve all the input traffic, reaching Goodput Factor values around 91-94%. The Proportional Fairness agent converged to the solution of continuously selecting all 4 STAs simultaneously (identical to the MU 4 STAs baseline), indicating that its fairness formulation is ineffective for this use case.

In contrast, the Max Rate, Standard, and Normalized agents all achieved Goodput Factor values of approximately 100% across all STAs, successfully delivering the totality of the transmitted data. The MU RR baseline also performed well in this scenario, approaching 100% Goodput.

A noteworthy observation is that the average latency metric reveals differences in fairness management among the RL agents, even when throughput results are comparable.

The Standard agent consistently achieved the lowest latency values (approximately 1.71-1.72 ms for all STAs), outperforming the Max Rate agent, which exhibited significantly higher latency for low-traffic STAs (e.g. 31-44 ms for STA 3 and STA 4). This demonstrates that the proposed Standard Reward fosters a more balanced allocation policy that does not starve lower-demand STAs, even though latency is not an explicit component of the reward function.

5.3.3.2 Congested Scenario (Trained)

In the congested scenario, the aggregate input traffic exceeds the network capacity. STAs 3 and 4 operate at MCS 3 (16-QAM, 1/2 coding rate) with 100 Mbps of demand, creating a severe bottleneck as their PHY rates cannot sustain the requested throughput. No algorithm achieves 100% Goodput in this regime.

The metrics employed here are exactly the same as in the previous Section, excluding the latency. Latency has not been included here because it does not provide relevant information in this use case. The reason, among others, is the manner that Wi-Fi queues are implemented in NS3¹⁰.

Metric	# STA	Non-AI Baselines			AI Models (RL Agents, $\epsilon = 0$)			
		MU (4 STAs)	Rand. act.	MU RR	Log. Agent	Max Rate Agent	Std. Agent	Norm. Agent
Rx Rate (Mbps)	1	2.66	3.71	4.74	2.66	4.63	5.02	4.38
	2	2.42	3.69	4.74	2.42	4.71	4.93	4.26
	3	25.32	37.05	47.97	25.32	44.95	49.22	43.29
	4	24.99	35.37	47.96	24.99	45.20	48.4	42.01
Rx/Tx Goodput (%)	1	25.85%	36.05%	46.06%	25.85%	44.99%	48.78%	42.56%
	2	23.52%	35.86%	46.06%	23.52%	45.77%	47.91%	41.40%
	3	24.61%	36.01%	46.63%	24.61%	43.69%	47.84%	42.08%
	4	24.29%	34.38%	46.62%	24.29%	43.93%	47.04%	40.83%

Table 16: Congested trained case results

The MU 4 STAs baseline achieved the worst result, reaching approximately 25% Goodput. The Proportional Fairness agent converged to the same MU 4 STAs solution, confirming that this reward is unable to produce a useful fairness notion in congested scenarios. The proposed Rewards produced substantially better outcomes: the Standard agent reached approximately 48% Goodput for all STAs, outperforming both the Max Rate agent (approximately 44-46%) and the Normalized agent (approximately 41-43%).

All RL agents performed comparably to the MU RR baseline (approximately 46%), which is known to be a strong heuristic when STAs are grouped by similar ATR values.

Differences among the Max Rate, Standard, and Normalized agents remained within a 5% margin, with the Standard agent being slightly superior overall.

¹⁰ All these simulations use DL traffic. The NS3 implements the AP queues based on the EDCA scheme, where there is one queue per EDCA AC, shared by all the STAs. In other words, a queue may be full and contain different amount of packets per STA. In each simulation, there is a first phase of random single user transmissions, so the amount of packets per STA in the queue is random at the beginning of the OFDMA simulation phase.

These results confirm that the proposed Standard Reward produces a more useful notion of fairness than Proportional Fairness, by assigning higher resource allocations to STAs with greater channel requirements without abandoning STAs with lower demands.

The inclusion of the min/max Goodput Factor ratio as a multiplying factor in the reward effectively prevents starvation while still maximising throughput.

5.3.3.3 Model Consistency and Generalisation

A critical design objective was to ensure that the trained RL model does not exhibit bias arising from the enumeration (ordering) of STAs at the ANN input. To evaluate this, every use case was replicated across all 6 possible STA-enumeration permutations (i.e. all distinct assignments of two MCS-Input Throughput pairs A and B to 4 STAs, yielding vectors [A,A,B,B], [B,B,A,A], [A,B,A,B], [B,A,B,A], [A,B,B,A], [B,A,A,B]). These permutations were fed during training in a round-robin technique to prevent any systematic association between a specific ATR value and a fixed STA input index.

The results across the 6 non-congested simulation cases confirmed that the agent treats a STA with a given set of features nearly identically regardless of its position in the input vector. All agents (Standard, Normalized, Max Rate) achieved approximately 100% Goodput Factor across all permutations, with only marginal variation in latency. In the congested cases, the results were also approximately consistent across permutations, with only two isolated exceptions (one per proposed reward) out of 12 congested simulation cases where the agent converged to a suboptimal solution resembling the MU 4 STAs baseline. This confirms that the STA enumeration generalisation strategy was effective.

Furthermore, the same unique RL agent was able to learn to solve both the congested and non-congested use cases simultaneously, rather than requiring a separate model per use case. This demonstrates that the DQN architecture, combined with the proposed state representation (ATR, averaged GF, and buffer status per STA), provides sufficient discriminative capacity for the agent to distinguish between fundamentally different network operating regimes within a single policy.

5.3.4 Evaluation Results on Non-Trained Use Case

To evaluate the agent's ability to generalise beyond training conditions, a third use case was defined with all STAs operating at MCS 7 (64-QAM, 5/6 coding rate) and input throughputs of 100, 100, 10, and 10 Mbps. This scenario was deliberately chosen to sit at the boundary between congestion and non-congestion: with MCS 7, a suboptimal scheduling policy leads to congestion, whereas a good scheduling policy can avoid it.

The ATR values (13.33 and 1.33) were distinct from those seen during training, ensuring the agent encountered genuinely novel state inputs. For RL, only the Standard agent was evaluated, given its slightly superior performance in the trained cases. The simulation time was extended to 100 seconds to allow sufficient adaptation time.

The agent was tested in both static mode (no learning) and dynamic mode (online learning with $\epsilon = 0.1$).

Metric	# STA	Non-AI Baselines			AI Model (RL: Std. Agent)	
		Multi User (4 STAs)	Random Actions	Multi User RR	Static Mode, $\epsilon = 0$	Dynamic Mode, $\epsilon = 0.1$
Rx Rate (Mbps)	1	63.11	76.00	102.88	82.11	92.77
	2	63.11	75.32	102.88	81.55	92.62
	3	6.45	7.65	10.29	8.26	9.32
	4	6.45	7.59	10.29	8.19	9.31
Rx/Tx Goodput (%)	1	61.34%	73.87%	100%	79.81%	90.17%
	2	61.34%	73.21%	100%	79.26%	90.02%
	3	62.68%	74.34%	99.99%	80.27%	90.57%
	4	62.68%	73.76%	99.99%	79.59%	90.47%

Table 17: Non-trained case results

The Static mode achieved approximately 80% Goodput Factor, a 6-20% improvement over both the MU 4 STAs baseline (approximately 61-63%) and Random Actions (approximately 73-74%). This confirms that the knowledge acquired during training transfers meaningfully to unseen network conditions. The Dynamic mode further improved performance to approximately 90% Goodput, demonstrating the agent's online learning capability and its effective adaptation to new conditions. However, the MU RR baseline still outperformed both modes in this scenario, achieving nearly 100% Goodput.

The approximately 10% Goodput improvement from static to dynamic mode is an encouraging result, as it validates one of the key advantages of the RL approach: an AP deployed in production could not only execute a pre-trained scheduling policy, but also continue refining it in real-time, adapting to the specific characteristics of its deployment environment (building structure, user traffic patterns, number of associated STAs).

A longer online training period may allow the dynamic agent to converge closer to the optimal solution.

5.3.5 Conclusions for AI-Based Multi-User Scheduler

The AI-Based Multi-User Scheduler applied Deep Reinforcement Learning (Double DQN) to the OFDMA resource allocation problem introduced in IEEE 802.11ax. A custom state representation combining Air Time Ratio, averaged Goodput Factor, and buffer status per station was proposed, together with two novel reward functions (Standard and Normalized) that incorporate a fairness multiplying factor based on the minimum-to-maximum Goodput Factor ratio.

The experimental results demonstrate that: (i) the trained agents successfully resolved both congested and non-congested use cases within a single policy, achieving approximately 100% Goodput in the non-congested scenario and up to 48% in the congested regime; (ii) the proposed Standard Reward outperformed both Proportional Fairness and Max Rate from the literature, providing substantially better latency balance without sacrificing throughput; (iii) the STA-enumeration permutation strategy eliminated input-ordering bias in the neural network; and (iv) the online learning capability in the non-trained scenario yielded a roughly 10% Goodput improvement over the static agent, validating the potential for real-time adaptation in deployed APs.

6. Summary

This deliverable presents the technical contributions developed under Task 2.3 of the OWIN6G project, addressing the design of AI-driven MAC layer scheduling algorithms for IEEE 802.11 wireless sensor networks. Two complementary strategies were proposed, implemented, and validated through extensive NS-3 simulations.

The first strategy, the AI-Based Traffic Flow Classifier, employs a supervised ML model (XGBoost) to identify application types from flow-level statistical features extracted in real-time at the AP. By dynamically mapping each detected flow to the appropriate EDCA Access Category, the classifier restores QoS differentiation that is otherwise rendered ineffective by applications universally marking their traffic as Best Effort. Simulation results across two scenarios (single BSS with heavy local contention and OBSS congestion) demonstrated 100% classification accuracy, average delay reductions of approximately 89% for latency-sensitive flows (video streaming, video conferencing, gaming, and VoIP), and successful generalisation to unseen traffic profiles. The accompanying DL/UL scheduling mechanism effectively protected TCP-based applications from uplink interference within the TXOP.

The second strategy, the AI-Based Multi-User Scheduler, applied Deep Reinforcement Learning (Double DQN) to the OFDMA resource allocation problem introduced in IEEE 802.11ax. A custom state representation combining Air Time Ratio, averaged Goodput Factor, and buffer status per station was proposed, together with two novel reward functions (Standard and Normalized) incorporating a fairness multiplying factor based on the minimum-to-maximum Goodput Factor ratio. The trained agents successfully resolved both congested and non-congested use cases within a single policy, achieving approximately 100% Goodput in non-congested conditions and up to 48% in the congested regime. The proposed Standard Reward outperformed both Proportional Fairness and Max Rate from the literature, providing substantially better latency balance without sacrificing throughput. Furthermore, the STA-enumeration permutation strategy effectively eliminated input-ordering bias in the neural network, and the online learning capability yielded a roughly 10% Goodput improvement in the non-trained scenario, validating the potential for real-time adaptation in deployed APs.

Together, these two strategies address complementary aspects of MAC layer intelligence: the traffic classifier operates at the flow level to ensure that latency-sensitive applications receive appropriate prioritisation, while the multi-user scheduler operates at the resource level to optimise bandwidth distribution among competing stations. Both approaches were validated using the NS-3 discrete-event simulator integrated with Python-based ML frameworks through the NS3-AI inter-process communication module, demonstrating the feasibility of embedding AI-driven decision-making within the IEEE 802.11 MAC layer for next-generation hybrid optical-wireless 6G networks. These results fulfil the objectives of Task 2.3, specifically the design of efficient MAC layer scheduling algorithms incorporating ML for wireless sensor networks based on existing mobile RF standards.

7. Appendix

7.1 Feature Selection Process

Starting from an initial set of 29 features, previously identified through a feature selection methodology applied in [14], a series of systematic experiments were conducted using the NS-3 network simulator to evaluate the behaviour of these features under varying congestion conditions. The experimental procedure comprised six iterative steps, each designed to refine the feature set based on empirical observations. The ML classifier employed throughout this process was XGBoost. All simulations used a duration of 60 seconds, and flow-level features were extracted using a sliding-window approach. The following Sections describe the rationale and results of each iteration.

7.1.1 Summary Table

Step	Method	Evaluation Scenario	Action	Features Removed	Features Reintroduced	Count	Results
Base	-	-	Initial candidate set from [14]	-	-	29	-
Step 1	Pearson correlation: data rate vs. Features.	Each application type at 3 to 70 Mbps over 1 Gbps Ethernet (point-to-point)	Remove features linearly dependent on data rate. $r > 0.9$	tcp_flag_count, num_packets, reverse_num_packets, total_packets, bytes, reverse_bytes, total_bytes, byte_rate, packet_rate, dt_intensity, avg_bytes, avg_dt_intensity (12)	-	17	100% accuracy on Ethernet channel across all application types with 17 features.
Step 2	Congestion test: Video Streaming on Ethernet	Video Streaming (VS) at 50 Mbps + IPERF UDP at 1 Gbps over 1 Gbps Ethernet channel.	Reintroduce feature to separate Video Streaming from IPERF_TCP under congestion	-	avg_dt_intensity (1)	18	Before: 23/54 VS flows misclassified as IPERF_TCP. After avg_dt_intensity reintroduction: 1/54 misclassified; non-congested Ethernet accuracy maintained at 100%.
Step 3	Congestion test: Video Streaming on Wi-Fi	3 Wi-Fi congested scenarios: i. Single-BSS; ii. OBSS; iii. OBSS + Bulk UDP at 100/200/300 Mbps.	Remove IAT features unstable under Wi-Fi congestion	mean_iat, std_iat (2)	-	16	Before (accuracy): Single-BSS: 99.81%. OBSS: 100%. OBSS + Bulk UDP at 100 Mbps: 100%, at 200 Mbps: 78.24% (Video Streaming recall 12.96%), at 300 Mbps: 82.41% (Video Streaming recall 29.63%). After retraining with 16 features: 100% accuracy in all scenarios.
Step 4	Evaluate eliminated features for Bulk TCP vs. Video Streaming separation	Add Bulk TCP application at 15 Mbps (emulating Video Streaming data rate) in Wi-Fi: i. Single-BSS; ii. OBSS + UDP 300 Mbps	Reintroduce std_iat as discriminative for Bulk TCP vs. Video Streaming	-	std_iat (1)	17	Bulk TCP is predominantly classified as Video Streaming (in Single-BSS). After reintroducing std_iat: 7 Bulk TCP flows still misclassified as Video Streaming. Under OBSS + UDP 300 Mbps: Bulk TCP classified as Bursty_UDP.

Step 5	Train with Bulk TCP app) as solution previous step 4); add discriminative features	Same Wi-Fi congestion scenarios as Steps 3 and 4 with Bulk TCP in training set	Add features for Bulk TCP and Video Streaming separation; remove unstable feature: duration	duration (1)	packet_rate, dt_intensity (2)	18	With 17 features: unsatisfactory Bulk TCP vs. Video Streaming separation. After adding dt_intensity and packet_rate, removing duration (18 features): improved discriminative performance.
Step 6	Adversarial validation: Ethernet vs. Wi-Fi distributions	Adversarial comparison of Bulk TCP distributions: Ethernet training vs. Wi-Fi congested	Remove features with poor cross-environment generalization	max_iat, switching_ratio (2)	-	16	For Bulk_TCP, max_iat and switching_ratio exhibited substantial distributional divergence between Ethernet and Wi-Fi. Final 16-feature model achieved stable classification across all congestion scenarios.

Table 18: Iterative Feature Selection Process

7.1.2 Explanation of Each Iteration

- Step 1 - Pearson Correlation Analysis: Each application type (Iperf TCP, Iperf UDP, Video Conference, Video Streaming and Bursty UDP) was simulated in NS-3 at data rates ranging from 3 Mbps to 70 Mbps over a 1 Gbps Ethernet channel.

A Pearson correlation analysis between the configured data rate and each of the 29 candidate features revealed that 12 of them exhibited a strong linear relationship with data rate ($r > 0.9$). Since under congestion conditions the available throughput varies, these 12 features would change proportionally, introducing a bias in the classifier that reflects network load rather than actual traffic type. To prevent this, the 12 data-rate-dependent features were removed, reducing the set to 17.

- Step 2 - Video Streaming Congestion on Ethernet: To test the 17-feature classifier under stress, a congestion scenario was created on the Ethernet channel by combining a 50 Mbps Video Streaming flow with a 1 Gbps Iperf UDP flow. Under these conditions, 23 out of 54 Video Streaming flows were misclassified as Iperf TCP.

The previously eliminated features were re-examined, and avg_dt_intensity was identified as the most suitable candidate for recovering discriminative power between these two classes. This feature captures the averaged effect of the high initial burst that characterizes DASH-based Video Streaming traffic, a pattern absent in Iperf_TCP. After its reintroduction, the model was retrained with 18 features and misclassifications dropped to a single flow, while accuracy on the non-congested Ethernet channel remained at 100%.

- Step 3 - Video Streaming Congestion on Wi-Fi: The 18-feature classifier, trained on Ethernet data, was evaluated on a Wi-Fi channel (5 GHz, 160 MHz, MCS 5) under three congestion scenarios of increasing severity: single-BSS, OBSS, and OBSS congestion with application Bulk UDP at 100, 200 and 300 Mbps. While the first two scenarios and the 100 Mbps case achieved near-perfect accuracy, the 200 Mbps and 300 Mbps cases caused Video Streaming recall to drop to 12.96% and 29.63%, respectively.

A comparison of feature distributions between the Ethernet training data and the congested Wi-Fi data revealed that mean_iat and std_iat shifted significantly under heavy congestion, deviating from their training distribution. Both features were removed.

After retraining with 16 features, the classifier achieved 100% accuracy in all three Wi-Fi congestion scenarios.

- Step 4 - Bulk TCP Evaluation: To assess classifier robustness against unseen traffic types, a 15 Mbps Bulk TCP flow intentionally configured to mimic the data rate of Video Streaming but absent from the training set was introduced in two Wi-Fi scenarios: Single-BSS congestion and OBSS with UDP at 300 Mbps.

In the first scenario, the classifier predominantly labelled Bulk TCP as Video Streaming, since both are TCP-based and share a similar data rate. To address this without adding Bulk TCP to the training set, the features eliminated in Step 3 were reconsidered.

Between `mean_iat` and `std_iat`, `std_iat` was preferred: being a variability metric, it is less sensitive to absolute congestion-induced delays than `mean_iat`. The Ethernet training data confirmed this choice, showing that Bulk TCP has nearly constant average `std_iat` (0.00037) while Video Streaming ranges between 0.008 and 0.015. After reintroducing `std_iat`, the feature count increased to 17 and residual misclassifications decreased notably, though some persisted because the Bulk TCP application had not been included in the training set.

- Step 5 - Training with Bulk TCP and Feature Refinement: After incorporating Bulk TCP into the training set, the model was retrained using 17 features; however, classification performance remained unsatisfactory. A closer examination of the training data revealed two additional features, `dt_intensity` and `packet_rate`, that exhibited clear distributional differences between Bulk TCP and Video Streaming, with the latter showing markedly higher dispersion in both metrics. Concurrently, the duration feature proved problematic: while stable within the 5-second window of the Ethernet training environment, it became highly variable under Wi-Fi congestion. This created a distribution mismatch between the training and testing sets, degrading the classifier's robustness. Consequently, `dt_intensity` and `packet_rate` were introduced, and duration was removed, resulting in a refined set of 18 features.
- Step 6 - Adversarial Validation¹¹: As a final robustness check, adversarial validation was performed using XGBoost by comparing the Ethernet training distribution of Bulk TCP against its distribution under Wi-Fi congestion. This analysis identified two features (`max_iat` and `switching_ratio`) whose distributions differed substantially between environments, indicating that they would not generalize reliably from Ethernet-trained models to Wi-Fi operational conditions. Both were removed, producing the final set of 16 features.

The resulting model was validated across all previously defined congestion scenarios and confirmed stable classification performance in every case.

¹¹ Adversarial Validation: A technique that uses a binary classifier to evaluate the similarity between training and test datasets. If the model successfully distinguishes between the two, the data distributions differ significantly; if not, they are similar.

7.1.3 Dataset from NS3 Applications

Label	Filename	Train	Test	Grand Total
Bulk_TCP	BulkTCP_10Mbps_Seed10	54		54
	BulkTCP_10Mbps_Seed50	54		54
	BulkTCP_10Mbps_Seed100	54		54
	BulkTCP_10Mbps_Seed500	54		54
	BulkTCP_10Mbps_Seed3		54	54
	BulkTCP_Seed3_5Mbps		3	3
	BulkTCP_Seed3_10Mbps		3	3
	BulkTCP_Seed3_30Mbps		3	3
	BulkTCP_Seed3_50Mbps		3	3
	BulkTCP_Seed3_70Mbps		3	3
Bulk_TCP Total		216	69	285
Bursty_UDP	BurstyUDP_10Mbps_Seed10	54		54
	BurstyUDP_10Mbps_Seed50	54		54
	BurstyUDP_10Mbps_Seed100	54		54
	BurstyUDP_10Mbps_Seed500	54		54
	BurstyUDP_10Mbps_Seed3		54	54
	BurstyUDP_Seed3_10Mbps		3	3
	BurstyUDP_Seed3_30Mbps		3	3
	BurstyUDP_Seed3_50Mbps		3	3
	BurstyUDP_Seed3_5Mbps		3	3
	BurstyUDP_Seed3_70Mbps		3	3
Bursty_UDP Total		216	69	285
Gaming	Gaming_Seed10	54		54
	Gaming_Seed50	54		54
	Gaming_Seed100	54		54
	Gaming_Seed500	54		54
	Gaming_Seed3		54	54
	Gaming_seed4		15	15
Gaming Total		216	69	285
IPERF_TCP	IperfTCP_10Mbps_Seed10	54		54
	IperfTCP_10Mbps_Seed50	54		54
	IperfTCP_10Mbps_Seed100	54		54
	IperfTCP_10Mbps_Seed500	54		54
	IperfTCP_10Mbps_Seed3		54	54
	IperfTCP_Seed3_5Mbps		3	3
	IperfTCP_Seed3_15Mbps		3	3
	IperfTCP_Seed3_30Mbps		3	3
	IperfTCP_Seed3_3Mbps		3	3
	IperfTCP_Seed3_50Mbps		3	3
IPERF_TCP Total		216	69	285
IPERF_UDP	IperfUDP_10Mbps_Seed10	54		54
	IperfUDP_10Mbps_Seed100	54		54
	IperfUDP_10Mbps_Seed50	54		54



	IperfUDP_10Mbps_Seed500	54		54
	IperfUDP_10Mbps_Seed3		54	54
	IperfUDP_Seed3_10Mbps		3	3
	IperfUDP_Seed3_30Mbps		3	3
	IperfUDP_Seed3_50Mbps		3	3
	IperfUDP_Seed3_5Mbps		3	3
	IperfUDP_Seed3_70Mbps		3	3
IPERF_UDP Total		216	69	285
Video_Conference	VideoConference_3Mbps_Seed10	54		54
	VideoConference_3Mbps_Seed100	54		54
	VideoConference_3Mbps_Seed50	54		54
	VideoConference_3Mbps_Seed500	54		54
	VideoConference_3Mbps_Seed3		54	54
	VideoConference_Seed3_10Mbps		3	3
	VideoConference_Seed3_30Mbps		3	3
	VideoConference_Seed3_50Mbps		3	3
	VideoConference_Seed3_5Mbps		3	3
	VideoConference_Seed3_70Mbps		3	3
Video_Conference Total		216	69	285
Video_Streaming	VideoStreaming_Dash_20Mbps_Seed10	54		54
	VideoStreaming_Dash_20Mbps_Seed100	54		54
	VideoStreaming_Dash_20Mbps_Seed50	54		54
	VideoStreaming_Dash_20Mbps_Seed500	54		54
	VideoStreaming_Dash_20Mbps_Seed3		54	54
	VideoStreaming_Seed3_5Mbps		3	3
	VideoStreaming_Seed3_10Mbps		3	3
	VideoStreaming_Seed3_30Mbps		3	3
	VideoStreaming_Seed3_40Mbps		3	3
	VideoStreaming_Seed3_50Mbps		3	3
Video_Streaming Total		216	69	285
VoIP	VoIP_Seed10	54		54
	VoIP_Seed50	54		54
	VoIP_Seed100	54		54
	VoIP_Seed500	54		54
	VoIP_Seed3		54	54
	VoIP_seed4		15	15
VoIP Total		216	69	285
Grand Total		1728	552	2280

7.1.4 Confusion Matrix Training Model

Figures 8 and 9 showed that the classifier provided a clear separation of the considered traffic classes in the NS-3 simulated environment over the Ethernet channel, for both training and testing phases, reaching an overall accuracy of 99.82%. The predominance of values along the main diagonal indicates high accuracy in both training and testing. Nevertheless, these results should be interpreted in the context of a controlled simulation setup.

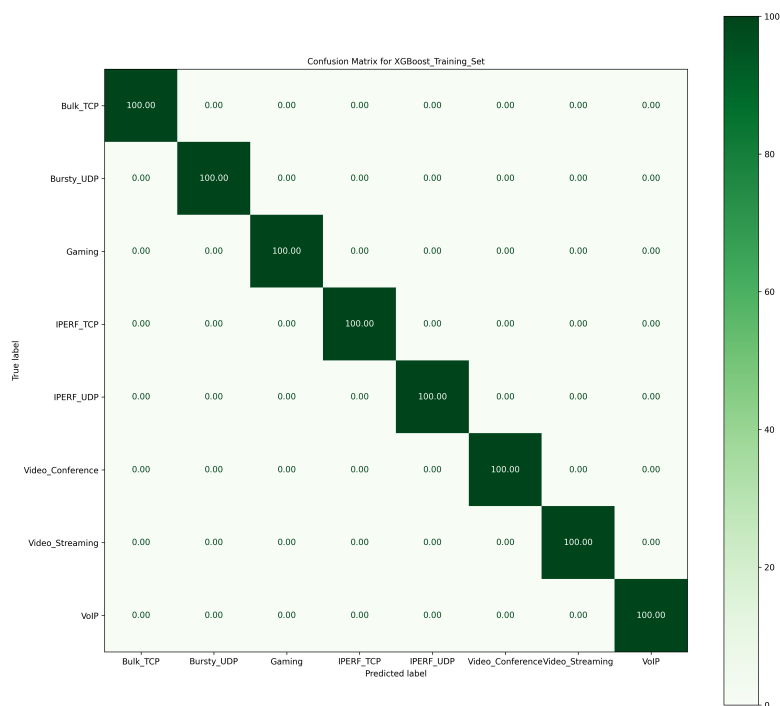


Figure 8: Confusion Matrix – Training Ethernet Channel



Figure 9: Confusion Matrix – Testing Ethernet Channel

8. References

- [1] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on IEEE 802.11ax high efficiency WLANs," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 197–216, Jan. 2019, doi: 10.1109/COMST.2018.2871099.
- [2] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," 2000.
- [3] M. Kulin, T. Kazaz, E. De Poorter, and I. Moerman, "A survey on machine learning-based performance improvement of wireless networks: PHY, MAC and network layer," *Electronics (Switzerland)*, vol. 10, no. 3, pp. 1–64, Feb. 2021, doi: 10.3390/electronics10030318.
- [4] B. Bellalta, "IEEE 802.11ax: High-efficiency WLANs," *IEEE Wirel. Commun.*, vol. 23, no. 1, pp. 38–46, Feb. 2016, doi: 10.1109/MWC.2016.7422404.
- [5] J. Hui and M. Devetsikiotis, "A unified model for the performance analysis of IEEE 802.11e EDCA," *IEEE Transactions on Communications*, vol. 53, no. 9, pp. 1498–1510, Sep. 2005, doi: 10.1109/TCOMM.2005.855013.
- [6] A. Custura, R. Secchi, and G. Fairhurst, "Exploring DSCP modification pathologies in the internet," *Comput. Commun.*, vol. 127, pp. 86–94, Sep. 2018, doi: 10.1016/j.comcom.2018.05.016.
- [7] A. Custura, A. Venne, and G. Fairhurst, "Exploring DSCP modification pathologies in mobile edge networks," in *2017 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, Jun. 2017, pp. 1–6. doi: 10.23919/TMA.2017.8002923.
- [8] S. Szott *et al.*, "Wi-Fi Meets ML: A Survey on Improving IEEE 802.11 Performance with Machine Learning," *IEEE Communications Surveys and Tutorials*, vol. 24, no. 3, pp. 1843–1893, Sep. 2022, doi: 10.1109/COMST.2022.3179242.
- [9] OWIN6G, "Optical and Wireless sensors Networks for 6G (OWIN6G)," 2025. Accessed: Oct. 23, 2025. [Online]. Available: <https://owin6g.eu/>
- [10] M. Camelo, P. Soto, and S. Latre, "A General Approach for Traffic Classification in Wireless Networks Using Deep Learning," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 5044–5063, Dec. 2022, doi: 10.1109/TNSM.2021.3130382.
- [11] S. S. Shaabanzadeh, M. Carrascosa-Zamacois, J. Sánchez-González, C. Michaelides, and B. Bellalta, "Virtual reality traffic prioritization for Wi-Fi quality of service improvement using machine learning classification techniques," *Journal of Network and Computer Applications*, vol. 230, p. 103939, Oct. 2024, doi: 10.1016/j.jnca.2024.103939.
- [12] H. Ozkan *et al.*, "Multimedia traffic classification with mixture of Markov components," *Ad Hoc Networks*, vol. 121, p. 102608, Oct. 2021, doi: 10.1016/j.adhoc.2021.102608.
- [13] F. Wilhelmi, S. Szott, K. Kosek-Szott, and B. Bellalta, "Machine Learning and Wi-Fi: Unveiling the Path Toward AI/ML-Native IEEE 802.11 Networks," *IEEE Communications Magazine*, vol. 63, no. 7, pp. 114–120, Jul. 2025, doi: 10.1109/MCOM.001.2400292.

- [14] F. Rau, C. Herranz, I. Val, P. Georgieva, and J. Perez, "A Novel Flow-Based Online Network Traffic Classification Using Machine Learning," *TechRxiv*, pp. 1–18, Aug. 2025, doi: 10.36227/techrxiv.175459507.75164419/v1.
- [15] NS3, "Web Site - NS3 Simulator," <https://www.nsnam.org/>. Accessed: Oct. 30, 2025. [Online]. Available: <https://www.nsnam.org/>
- [16] D. J. Vergados, A. Michalas, A. Sgora, D. D. Vergados, and P. Chatzimisios, "FDASH: A fuzzy-based MPEG/DASH adaptation algorithm," *IEEE Syst. J.*, vol. 10, no. 2, pp. 859–868, Jun. 2016, doi: 10.1109/JSYST.2015.2478879.
- [17] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
- [18] Scikit-learn developers, "Cross-validation: evaluating estimator performance," Feb. 2025. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html
- [19] M. Committee of the IEEE Computer Society, "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 Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements IEEE Computer Society," 2005.
- [20] M. S. Kuran, A. Dilmac, O. Topal, B. Yamansavascular, S. Avallone, and T. Tugcu, "Throughput-maximizing OFDMA Scheduler for IEEE 802.11ax Networks," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, Aug. 2020, pp. 1–7. doi: 10.1109/PIMRC48278.2020.9217366.
- [21] D. Bankov, A. Didenko, E. Khorov, and A. Lyakhov, "OFDMA Uplink Scheduling in IEEE 802.11ax Networks," in *2018 IEEE International Conference on Communications (ICC)*, IEEE, May 2018, pp. 1–6. doi: 10.1109/ICC.2018.8422767.
- [22] A. Dutta, N. Gupta, S. Das, and M. Maity, "MMRU-ALLOC: An Optimal Resource Allocation Framework for OFDMA in IEEE 802.11ax," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, Aug. 2020, pp. 1–6. doi: 10.1109/PIMRC48278.2020.9217154.
- [23] Y. Li, S. Han, and C. Yang, "User scheduling for uplink OFDMA systems by deep learning," in *IEEE Wireless Communications and Networking Conference, WCNC*, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/WCNC49053.2021.9417487.
- [24] P. K. Sangdeh and H. Zeng, "DeepMux: Deep-Learning-Based Channel Sounding and Resource Allocation for IEEE 802.11ax," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2333–2346, Aug. 2021, doi: 10.1109/JSAC.2021.3087246.
- [25] J. Liu, X. Ma, W. Han, and L. Wang, "Resource Allocation in OFDMA Networks with Deep Reinforcement Learning," in *2020 IEEE 8th International Conference on Information, Communication and Networks (ICICN)*, IEEE, Aug. 2020, pp. 111–117. doi: 10.1109/ICICN51133.2020.9205096.

- [26] K. Nakashima, S. Kamiya, K. Ohtsu, K. Yamamoto, T. Nishio, and M. Morikura, "Deep Reinforcement Learning-Based Channel Allocation for Wireless LANs with Graph Convolutional Networks," *IEEE Access*, vol. 8, pp. 31823–31834, 2020, doi: 10.1109/ACCESS.2020.2973140.
- [27] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.
- [28] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [29] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International conference on machine learning*. PMLR, 2015.
- [30] Python, "Python Programming Language Official Website," <https://www.python.org/>. Accessed: Oct. 23, 2025. [Online]. Available: <https://www.python.org/>
- [31] H. Yin *et al.*, "Ns3-Ai: Fostering Artificial Intelligence Algorithms for Networking Research," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jun. 2020, pp. 57–64. doi: 10.1145/3389400.3389404.
- [32] P. Gawłowicz and A. Zubow, "ns3-gym: Extending OpenAI Gym for Networking Research," Oct. 2018, [Online]. Available: <http://arxiv.org/abs/1810.03943>
- [33] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," 2011. doi: 10.5555/1953048.2078195.
- [34] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," *arXiv preprint arXiv:1605.08695*, 2016, doi: 10.5555/3026877.3026899.
- [35] T. Chen *et al.*, "xgboost: Extreme Gradient Boosting," 2025. [Online]. Available: <https://github.com/dmlc/xgboost>