

AWaRe-SAC: Proactive Slice Admission Control under Weather-Induced Capacity Uncertainty

Dror Jacoby*, Yanzhi Li*, Shuyue Yu, Nicola Di Cicco, Hagit Messer, Gil Zussman, and Igor Kadota

Abstract—Millimeter-wave (mmWave) links are increasingly utilized in wireless x-haul transport to meet growing service demands. However, the inherent susceptibility of mmWave links to weather-related attenuation creates uncertainty about future network capacity which can significantly affect Quality of Service (QoS). This creates a critical challenge: how to make admission control decisions for slices with QoS requirements, balancing acceptance rewards against the risk of future QoS-violation penalties due to capacity uncertainty? To address this, we develop a proactive slice admission control framework that tightly integrates: (i) a predictor that leverages historical link measurements to forecast short-term attenuation and quantify uncertainty; and (ii) an admission control algorithm that incorporates both the predictions and uncertainties to maximize rewards and minimize QoS-violation penalties. We compare our framework against baseline, state-of-the-art, and idealized oracle algorithms using real-world mmWave x-haul data and residential traffic traces. Simulations suggest that our framework can achieve revenues that are 250% larger than baseline algorithms and 75% larger than state-of-the-art algorithms.

I. INTRODUCTION

Beyond-5G networks increasingly rely on mmWave frequencies, particularly V Band (57–71 GHz) and E Band (71–76 GHz and 81–86 GHz), for short-range deployments requiring high bandwidth and low latency [1]. However, these higher frequencies are more susceptible to weather-induced attenuation, with rain causing significant signal degradation, limiting reliability [2]–[4]. Satisfying stringent QoS requirements of dynamic traffic demands under time-varying link capacities creates complex challenges for network operators, driving the need for resilient, adaptive strategies to mitigate communication disruptions and optimize network performance.

Most existing slice admission control algorithms (discussed in Sec. II) assume that link capacities are static and known a priori, and, thus, are inadequate for modern mmWave networks. In contrast, our work explicitly integrates weather-aware link capacity forecasting and proactive slice admission in mmWave backhaul. Admission decisions account not only for immediate rewards (accrued from slice admission) but also

Dror Jacoby and Hagit Messer are with the School of Electrical Engineering, Tel Aviv University. Yanzhi Li and Igor Kadota are with the Department of Electrical and Computer Engineering, Northwestern University. Shuyue Yu and Gil Zussman are with the Department of Electrical Engineering, Columbia University. Nicola Di Cicco is with the Department of Electronics, Information and Bioengineering, Politecnico di Milano. This work was supported in part by NSF and Center for Smart Streetscapes (CS3) under NSF Cooperative Agreement EEC-2133516, NSF grant AST-2232455, NSF grant AST-2132700, NSF grant CNS-2433807, NSF grant CNS-2148128 and by funds from federal agency and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program, and BSF grant T-2025115. *These authors contributed equally to this work.

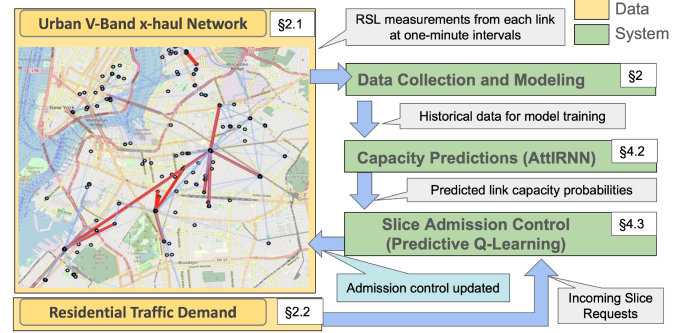


Fig. 1: Overview of the AWaRe-SAC framework workflow.

for future penalties arising from potential QoS violations (due to stochastic link capacity degradation).

We develop an Adaptive Weather-aware Real-time Slice Admission Control (AWaRe-SAC) framework that tightly integrates: (i) **Prediction phase** which leverages historical link measurements to forecast short-term link attenuation and quantify estimation uncertainty. (ii) **Slice admission control phase** which incorporates the sequence of predictions and uncertainties to proactively admit Slice Requests (SRs) aiming to maximize rewards and minimize QoS-violation penalties. Fig. 1 presents the AWaRe-SAC framework applied to a segment of the NYC Mesh [5] mmWave network deployed in New York City (NYC). Due to space constraints, we use appendices in [6] for details of the mmWave network infrastructure, SR generation mechanism, choice of QoS-violation penalty functions, and more. **Our contributions include:**

- **Slice Admission with Time-varying Resources:** Contrasting with the traditional slice admission problem with static network resources, our network has a unique challenge: QoS-violations may be unavoidable. Our novel problem formulation and AWaRe-SAC framework explicitly account for QoS-violation risk and ensuing penalties, including algorithms that address slice underprovisioning. AWaRe-SAC proactively balances current admission rewards and predicted QoS-violation penalties by tightly coupling forecasting of capacity levels probability distributions with a Q-learning admission control algorithm.
- **Evaluation using Real-World Datasets:** We use two datasets: (i) link measurements from the NYC Mesh network, capturing weather-induced attenuation in V Band; and (ii) residential traffic traces from 520 units across 28 buildings in NYC, capturing real user demand patterns. Our predictor leverages the significant short-term temporal correlation of weather-induced attenuation. Slices generated

from traces can have longer durations. The combination of these datasets provides the first realistic benchmark for slice admission algorithms under weather-induced capacity variations. We validate AWaRe-SAC on this benchmark, demonstrating 15–20% improved prediction accuracy and up to $3.5\times$ higher revenue.

II. RELATED WORK

Slice Admission Control. Prior work on slice admission control explores Reinforcement Learning (RL)-based methods, including tabular Q-learning [7], deep RL [8], [9], and multi-agent methods [10], as well as non-RL approaches such as optimization and adaptive methods [11], [12]. Most prior work on slice admission control assumes static network resources, whereas real-world x-haul networks can be sensitive to weather-induced attenuation, rendering existing solutions inadequate for mmWave networks. Our novel framework not only aims to “pack” as many slices as possible in the available resources, but also minimizes the risk of future underprovisioning penalty. Introducing time-varying network resources and underprovisioning penalties, unique to our mmWave x-haul scenario, significantly changes both the constraints and the objective of the admission control problem.

Furthermore, prior work has primarily evaluated admission control algorithms on slices created from simulations or predefined distributions (e.g., Poisson processes) [7], [10]. A main challenge for generating realistic slices is the lack of standardized mechanisms for creating slices from network traces. We address this challenge in Sec. IV-B.

Forecasting Network Resources and Traffic Demand. Traffic demand forecasting has evolved from statistical models to data-driven methods [13], with deep learning frameworks explicitly optimizing cost and QoS [14], [15]. A similar trend appears in rain-induced attenuation forecasting, where data-driven methods like RNNs [16] have been adopted to leverage historical data for rain mitigation and nowcasting applications [17], [18], complementing traditional time-series approaches. In prior work, we utilized weather-aware capacity predictions for network routing [19]. In contrast, in this work, we forecast future capacity *distributions* and use uncertainty estimates to proactively mitigate capacity shortage risks.

III. PROBLEM FORMULATION AND ORACLE SOLUTION

Consider a mmWave network with star-like topology¹ (similar to Fig. 1, detailed in [6, Appendix B-A]) supporting different 5G services including Ultra-Reliable and Low Latency Communications (URLLC), Enhanced Mobile Broadband (eMBB), Massive Machine-Type Communications (mMTC), and Best-Effort (BE). *Users* generate traffic flows with different QoS requirements. *Tenants* aggregate user flows and generate SRs associated with different 5G service types.

¹To extend the AWaRe-SAC framework to general network topologies, the problem formulation would need to account for the impact of accepting individual slices on the entire network, which includes adding flow conservation constraints to (1)–(6).

The *infrastructure provider* utilizes an admission control algorithm to decide whether to accept or reject arriving SRs to a particular mmWave link, aiming to maximize revenue.

Link Capacity. Time is divided into time-slots with index $t \in \{1, \dots, \mathcal{H}\}$, where $\mathcal{H}$ is the time-horizon and the slot duration is 1 minute. Capacity measurements, denoted by $\{C_t\}_{t=1}^{\mathcal{H}}$, fluctuate over time, potentially dropping to zero during severe rain fading events. We describe the measurements collected from real-world mmWave links in Sec. IV-A.

Slice Requests. SRs are ordered chronologically with index $i \in \{1, \dots, N\}$, where N is the number of SRs generated within the $\mathcal{H}$ time-slots. In Sec. IV-B, we describe the SR generation model, based on real-world network traces. SR i is characterized by service type (URLLC, eMBB, mMTC, or BE), throughput requirement d_i , duration D_i , sum reward $R_i = \rho_i d_i D_i$ for admitting SR i , where ρ_i is the price per unit throughput per slot (which varies for different service types), and a per slot penalty P_{it} incurred when d_i cannot be fulfilled in slot t . Let t_i represent the slot when SR i is generated. Since several SRs can be generated in the same slot, it follows that $t_1 \leq t_2 \leq \dots \leq t_N$. Let the binary decision variable $z_i \in \{0, 1\}$ indicates whether the algorithm admits SR i . If admitted, $z_i = 1$, then SR i becomes active during $t \in \{t_i, t_i + 1, \dots, t_i + D_i - 1\}$. The indicator $k_{it} \in \{0, 1\}$ indicates if SR i is active in slot t . It follows that $k_{it} \leq z_i, \forall i, t$.

Underprovisioning and Penalty. We consider admission control algorithms that do not admit new SRs when current admitted SRs are being underprovisioned. In slot t , underprovisioning occurs when $C_t < \sum_{i=1}^N k_{it} d_i$. In this case, admission control algorithms attempt to allocate the available capacity C_t to active SRs, aiming to minimize penalty. Let the decision variable $f_{it} \in [0, 1]$ be the fraction of the requested throughput d_i assigned to the active SR i in time slot t . The penalty for assigning $f_{it} < 1$ in slot t is a convex piecewise-linear function with m_i segments defined as $P_{it} = \max_{k \in \{1, \dots, m_i\}} \{a_{ik}(1 - f_{it}) + b_{ik}\}$ with penalty coefficients $a_{ik} \geq 0$ and $b_{ik} \leq 0$ that may vary for different 5G service types. Penalty coefficients are chosen such that $P_{it} = 0$ when $f_{it} = 1$ and P_{it} increases sharply as f_{it} approaches 0 to encourage distributing the available capacity across all active SRs. In time slot t , admission control algorithms select f_{it} to minimize $\sum_{i=1}^N P_{it}$ under the constraint $\sum_{i=1}^N k_{it} d_i f_{it} \leq C_t$.

Perfect-information MILP formulation (Oracle) Assuming perfect knowledge of past, present, and future SRs and link capacity fluctuation, we design an Oracle admission control algorithm that maximizes the cumulative reward minus penalty. Specifically, we formulate a Mixed-Integer Linear Program (MILP) over the finite horizon $\mathcal{H}$ that serves as an idealized benchmark. The MILP is related to the classical Multidimensional Knapsack Problem, with two key distinctions. First, the knapsack capacity (i.e., the available link capacity C_t) changes over time. Second, items can be fractionally packed $f_{it} < 1$ with a penalty $P_{it} > 0$. The MILP formulation shown below jointly determines: SRs to admit $z_i, \forall i$, and (ii) fractions $f_{it}, \forall i, t$, of the SRs throughput d_i to allocate. Key notation

used in this work is summarized in [6, Appendix A].

$$\min \sum_{t=1}^{\mathcal{H}} \sum_{i=1}^N P_{it} - \sum_{i=1}^N z_i R_i \quad (1)$$

$$\text{s.t. } z_i = 1 \rightarrow P_{it} \geq a_{ik}(1 - f_{it}) + b_{ik}, \quad \forall i, t, k \quad (2)$$

$$\sum_{i=1}^N k_{it} d_i f_{it} \leq C_t, \quad \forall t \quad (3)$$

$$P_{it} \geq \delta \rightarrow z_{i'} = 0, \quad \forall i, t, \forall i' \text{ such that } t_{i'} = t \quad (4)$$

$$f_{it} = 0, P_{it} = 0 \quad \forall i, t \text{ such that } k_{it} = 0 \quad (5)$$

$$\delta > 0, 0 \leq f_{it} \leq 1, z_i \in \{0, 1\}, P_{it} \geq 0 \quad (6)$$

The objective function (1) minimizes penalty minus reward, which is equivalent to maximizing revenue. Constraint (2) uses an epigraph reformulation of the penalty function. Constraint (3) enforces the time-varying capacity limit. Constraint (4) prevents the admission of new SRs while underprovisioning, i.e., when $P_{it} \geq \delta > 0$. Constraints (2) and (4) can be implemented using standard big- M linearization or indicator constraints.

Although the Oracle solution to (1)-(6) cannot be applied in practice, due to the assumption of perfect information, it provides a benchmark for our evaluation in Sec. VI. Further, it guides the design of the AWARe-SAC framework that leverages short-term predictions of C_t to inform its online admission control decisions. Next, we describe both datasets utilized in the design and evaluation of AWARe-SAC.

IV. DATASETS: LINK ATTENUATION AND SLICE REQUESTS

A. mmWave Link Measurements

Dataset. In this paper, we use networking logs collected by NYC Mesh [5] containing Received Signal Level (RSL) measurements at one-minute resolution from 25 directional V-band (58–70 GHz) links with distances from hundreds of meters to several kilometers. RSL measures received power, in dBm, which is directly affected by rain-induced attenuation. Details about the mmWave network infrastructure (partially shown in Fig. 1) and the dataset (that we made publicly available [5]) can be found in [6, Appendix B].

Weather-Induced Capacity Fluctuation. Most prior work on slice admission control assumes static link capacity, whereas real-world networks can be sensitive to weather-induced attenuation. V-band links (shown in red in Fig. 1) are highly susceptible to precipitation, which can cause severe absorption and scattering [4], [20], leading to sharp drops in RSL and link capacity C_t , posing a critical challenge for network operators. Fig. 2 showcases this impact on a representative 69 GHz link (March 09–10, 2024): while the signal remains stable during dry periods, heavy rain results in attenuation exceeding 30 dB, leading to temporary outages. To map RSL to discrete link capacity values, we adopt a standard hysteresis-based Adaptive Coding and Modulation (ACM) approach, described in Section V-A and detailed in [6, Appendix D-1].

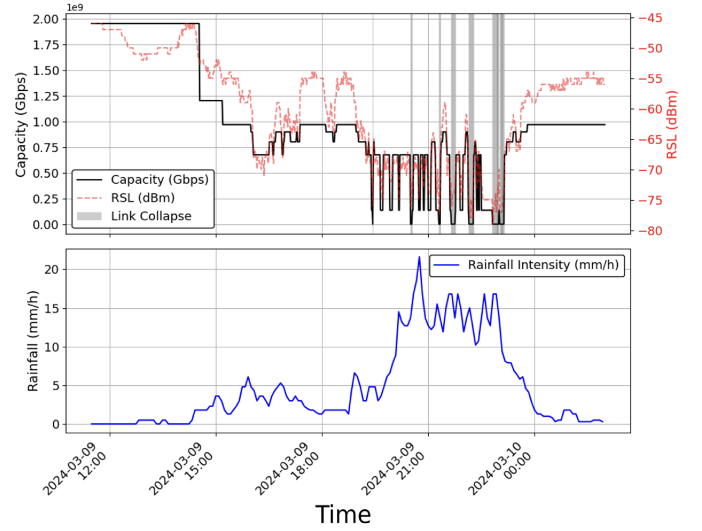


Fig. 2: RSL and capacity fluctuation during rainfall event.

B. Modeling Slice Requests from Network Traces

Standardized mechanisms for slice generation have not yet been established in the industry. We propose a SR generation pipeline based on network traces.

Dataset and 5G Services. For over one month, we collected user traffic demands from 520 Columbia University residential units in off-campus buildings. These residential buildings accommodate graduate students, postdocs, and faculty and their families. We anonymized privacy-sensitive fields, discarded personally identifiable information, and publicized the dataset [21] with network traces. In this paper, we group packets with the same 5-tuple (i.e., source IP, destination IP, source port, destination port, transport protocol) as a flow. We associate the most recent DNS lookup to each flow. Based on the DNS domains, TLS Server Name Indications (SNI), port numbers, and IP addresses, we associate each flow with a specific 5G service according to the following mapping: (i) **eMBB** service is designed for high-speed, high-capacity data transmission. We associate eMBB SRs with flows from video streaming and gaming services, including YouTube, Netflix, HBO, Hulu, Steam, and PlayStation; (ii) **URLLC** service is designed for ultra-reliable, low-latency communication services. We associate URLLC SRs with flows from video conferencing and chatting applications, including Zoom, Snapchat, WhatsApp, and GoogleMessages; (iii) **mMTC** service caters to the connectivity needs of numerous IoT devices. We did not identify IoT flows in our data; (iv) **BE** service is designed for non-critical communication. We associate BE SRs with flows from Spotify, Gmail, and Google Docs.

Slice Request Characterization. For management simplicity, we assume that there is only a limited number of SR types, each type characterized by 5G service, duration D_i , throughput d_i , reward R_i , and penalty P_{it} . Rewards and penalties are defined by tenants and infrastructure providers based on the 5G service. In Sec. VI we evaluate AWARe-SAC using different R_i and P_{it} . We compute SR duration D_i and throughput d_i based on flow statistics. We consider three D_i options: the 90th

percentile of flow duration for each of the three 5G service categories. For each D_i , we consider four d_i options: the 25th, 50th, 75th, and 95th percentiles of aggregated flow throughput. Throughput values are scaled up by a factor of 4× to ensure that demands are aligned with the available link capacity in our evaluation, and to anticipate increased bandwidth consumption due to the growing adoption of data-intensive services. In total, we generate 12 types of SRs.

Slice Request Generation. With the above principles, we consider flows in chronological order and generate SRs as follows: For each incoming flow, if its throughput cannot be accommodated within active SRs, then a new SR is generated using the flow's rounded-up throughput. When a flow lasts beyond its SR, the flow is treated as an incoming flow.

V. SYSTEM DESIGN

AWaRe-SAC integrates an AttIRNN predictor with a two-tier SR admission controller. AWaRe-SAC Workflow: (1) Recent RSL measurements $\mathbf{X}_t$ are fed into the trained AttIRNN predictor. (2) The predictor outputs short-term capacity distributions. (3) Admission control maximizes expected revenue using either: (i) Locally Optimal or (ii) Predictive Q-Learning.

A. AttIRNN Predictor

AttIRNN is a multi-step probabilistic predictor of RSL values that leverages the significant temporal correlation of weather-induced fluctuations over short periods (see Fig. 2). AttIRNN uses a short-term horizon of $H = 5$ minutes. AttIRNN transforms the sequence of RSL predictions into capacity distributions that are input to admission control.

Prediction Problem Formulation. Given the (static) link characteristics $\mathbf{x}_s \in \mathbb{R}^S$ (e.g., frequency and length [4]) and the T -step input history $\mathbf{X}_t = \{x_{t-T+1}, \dots, x_t\} \in \mathbb{R}^T$ which represents the T most recent RSL measurements, AttIRNN learns a mapping $F_\Theta : (\mathbf{X}_t, \mathbf{x}_s) \mapsto \hat{\mathbf{Y}}_t$, where $\hat{\mathbf{Y}}_t = \{(\hat{\mu}_{t+1}, \hat{\sigma}_{t+1}^2), \dots, (\hat{\mu}_{t+H}, \hat{\sigma}_{t+H}^2)\} \in \mathbb{R}^{H \times 2}$ represents the H -step predicted mean-variance pairs with $\hat{\mu}_{t+h}$ estimating the future expected RSL x_{t+h} and $\hat{\sigma}_{t+h}^2$ capturing the inherent uncertainty of the prediction at step $t+h$.

Learning Setup. The training set is built with a sliding window of length $T+H$, producing supervised pairs $(\mathbf{X}, \mathbf{Y})$ where $\mathbf{X} = \{x_1, \dots, x_T\}$ and $\mathbf{Y} = \{y_1, \dots, y_H\}$ with $y_h = x_{T+h}$. At inference, these map to time-indexed $\mathbf{X}_t$ and $\hat{\mathbf{Y}}_t$. Model parameters Θ are learned by minimizing the Gaussian negative log-likelihood (NLL)

$$\mathcal{L}(\Theta) = \frac{1}{2} \sum_{(\mathbf{X}, \mathbf{Y}) \in \mathcal{D}} \sum_{h=1}^H \left[\frac{(y_h - \hat{\mu}_h)^2}{\hat{\sigma}_h^2} + \log \hat{\sigma}_h^2 \right], \quad (7)$$

which drives accurate mean predictions and calibrated predictive uncertainty, assuming independent² Gaussian prediction errors $y_h - \hat{\mu}_h \sim \mathcal{N}(0, \hat{\sigma}_h^2)$ at each step h .

AttIRNN Architecture. Fig. 3 depicts the AttIRNN architecture centered around an attention-augmented encoder-decoder

framework [22]. The encoder processes the input sequence $\mathbf{X}$ via LSTM cells [23] to generate hidden states $\{\phi_j^E\}_{j=1}^T$. Given a link's static features $\mathbf{x}_s \in \mathbb{R}^S$, AttIRNN computes an embedding $\mathbf{g}_s$ via a fully-connected (FC) layer. This embedding enables link-specific adaptation while leveraging shared patterns learned across multiple links. At each decoder step h , attention aligns the previous decoder state ϕ_{h-1}^D , static context $\mathbf{g}_s$, and encoder states via:

$$\beta_{hj} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\phi_{h-1}^D; \mathbf{g}_s; \phi_j^E] + \mathbf{b}_a), \quad (8)$$

$$\tilde{\beta}_{hj} = \exp(\beta_{hj}) / \sum_{j'=1}^T \exp(\beta_{hj'}), \quad (9)$$

yielding context $\mathbf{g}_h = \sum_{j=1}^T \tilde{\beta}_{hj} \phi_j^E$ and augmented context $\tilde{\mathbf{g}}_h = [\mathbf{g}_h; \mathbf{g}_s]$. The decoder then autoregressively generates predictions through states $\{\phi_h^D\}_{h=1}^H$:

$$\phi_h^D = \text{LSTM}(y_{h-1}, [\phi_{h-1}^D, \tilde{\mathbf{g}}_h]), \quad (10)$$

$$\hat{\mathbf{y}}_h = \mathbf{W}_o[\phi_h^D; \tilde{\mathbf{g}}_h] + \mathbf{b}_o, \quad (11)$$

where $\hat{\mathbf{y}}_h = (\hat{\mu}_h, \hat{\sigma}_h^2)$ with positive $\hat{\sigma}_h^2$ via softplus. Here y_{h-1} is ground truth at training (teacher forcing) and previous prediction $\hat{\mu}_{h-1}$ at inference.

Discrete Capacity Distributions. During regular operations, mmWave links use ACM to map their current RSL to a discrete link capacity. We consider the standard hysteresis-based ACM detailed in [6, Appendix D-1] with capacity levels C_l , with $C_0 < C_1 < \dots < C_{L-1}$, and associated RSL hysteresis thresholds $C_l^\downarrow$ (to transition from C_l to C_{l-1}) and $C_l^\uparrow$ (to transition from C_l to C_{l+1}).

At slot t , AttIRNN uses the current (known) capacity $C_{l'}$ and the H -step mean-variance pairs $\hat{\mathbf{Y}}_t = \{(\hat{\mu}_{t+h}, \hat{\sigma}_{t+h}^2)\}_{h=1}^H$ to generate probability mass functions $\tilde{\mathbf{p}}_{t+h} = \{\tilde{p}_{l,h}\}_{l=0}^{L-1}$ over the capacity levels $C_l \in \{0, \dots, L-1\}$ and over the short-term horizon $h \in \{1, \dots, H\}$, where $\tilde{p}_{l,h} \in [0, 1]$ represents the probability that the link capacity at time slot $t+h$ is C_l . To obtain $\tilde{p}_{l,h}$, we use the assumption of independent Gaussian prediction errors to compute the h -step transition probabilities from the current level $C_{l'}$ to C_l given by

$$\tilde{p}_{l,h}(l' \rightarrow l) = \Phi\left(\frac{\max(C_l^\uparrow, C_{l'}^\downarrow) - \hat{\mu}_{t+h}}{\hat{\sigma}_{t+h}}\right) - \Phi\left(\frac{\min(C_l^\downarrow, C_{l'}^\uparrow) - \hat{\mu}_{t+h}}{\hat{\sigma}_{t+h}}\right) \quad (12)$$

where $\Phi(\cdot)$ is the standard normal CDF. Then, assuming a first-order Markov model, the short-term capacity distribution $\tilde{\mathbf{p}}_{t+h}, \forall h \in \{1, \dots, H\}$ can be computed iteratively as

$$\tilde{p}_{l,h} = \sum_{j=0}^{L-1} \tilde{p}_h(j \rightarrow l) \tilde{p}_{j,h-1}, \quad h = 2, \dots, H, \quad (13)$$

initialized by $\tilde{p}_{l,1} = \tilde{p}_1(l' \rightarrow l)$. These short-term capacity distributions that account for prediction uncertainty $\hat{\sigma}_h^2$ are input to slice admission control algorithms discussed next.

B. Slice Admission Control Algorithms

We present three slice admission algorithms: (i) a Rate Control mechanism for SR underprovisioning, (ii) a Locally Optimal algorithm that leverages capacity distributions for revenue maximization, and (iii) a Predictive Q-Learning algorithm that incorporates capacity forecasts into RL.

²While the errors are assumed independent and Gaussian per horizon step, the actual predictions are not assumed independent.

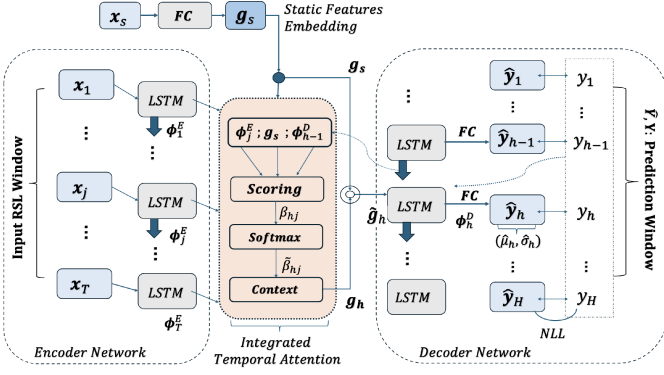


Fig. 3: AttIRNN. Seq2Seq-based predictions with temporal attention for H -step prediction using T -step history.

Rate Control for SR Underprovisioning in Slot t . Denote by $\mathcal{N}_t = \{i \in \{1, \dots, N\} | k_{it} = 1\}$ the set of active SRs at the beginning of slot t . In slot t , underprovisioning occurs when $C_t < \sum_{i \in \mathcal{N}_t} d_i$. In this case, instead of admitting new SRs, admission control algorithms select fractions $f_{it}, \forall i \in \mathcal{N}_t$ of the active SRs throughput request d_i aiming to minimize the QoS-violation penalty in slot t subject to the link capacity constraints, as described by the linear problem below

$$\min \sum_{i \in \mathcal{N}_t} P_{it} \quad (14)$$

$$\text{s.t. } P_{it} \geq a_{ik}(1 - f_{it}) + b_{ik}, \forall k, i \in \mathcal{N}_t \quad (15)$$

$$\sum_{i \in \mathcal{N}_t} d_i f_{it} \leq C_t \quad (16)$$

$$0 \leq f_{it} \leq 1, \quad P_{it} \geq 0 \quad (17)$$

This rate control mechanism is integrated into both the Locally Optimal and Predictive Q-learning algorithms described next.

Locally Optimal (LO). In each slot t , LO considers three inputs: (i) the set of active SRs $\mathcal{N}_t$; (ii) the batch of arriving SRs, denoted by $\mathcal{A}_t = \{i \in \{1, \dots, N\} | t_i = t\}$; and (iii) the capacity distributions $\hat{\mathbf{p}}_{t+h}$. The goal of LO is to admit the subset of arriving SRs, $\mathcal{A}_t^* \subseteq \mathcal{A}_t$, that maximizes the expected short-term revenue $\mathbb{E} \left[\sum_{i \in \mathcal{A}_t^*} R_i - \sum_{\tau=t}^{t+H} \sum_{i \in \{\mathcal{N}_t \cup \mathcal{A}_t^*\}} P_{i\tau} \right]$. In slot t , there are $2^{|\mathcal{A}_t|}$ SR subsets that can be admitted (each subset is associated with a reward $\sum_i R_i$) and L^H possible sequence of capacity levels C_l over the short-term horizon (each sequence is associated with a probability $\prod_{h=1}^H \hat{p}_{l,h}$). Each of the $2^{|\mathcal{A}_t|} L^H$ combinations is associated with a penalty $\sum_{\tau} \sum_i P_{i\tau}$ that may affect the expected short-term revenue.

To reduce complexity, a key insight is that for each SR subset, the penalty $\sum_i P_{i\tau}$ obtained from (14)-(17) only depends³ on the capacity level $C_\tau \in \{C_0, \dots, C_{L-1}\}$ and not on τ . Hence, to obtain $\sum_i P_{i\tau}, \forall \tau \in \{t, \dots, t+H\}$, we only need to solve (14)-(17) L times, instead of L^H times. When $|\mathcal{A}_t|$ is small, LO can easily compute the $2^{|\mathcal{A}_t|}$ possible rewards and corresponding expected short-term revenues to obtain the optimal SR subset $\mathcal{A}_t^* \subseteq \mathcal{A}_t$. When $|\mathcal{A}_t|$ is large (in our evaluation it can be ~ 100 SRs), LO finds an approximate solution by sorting the set $\mathcal{A}_t$ in decreasing order of rewards R_i and adding SRs to the admitted set $\mathcal{A}_t^*$, one at a time,

³Here, we make the pessimistic assumption that no SR will expire within the short-term horizon H .

if their addition increases the short-term expected revenue, thereby reducing complexity from $2^{|\mathcal{A}_t|}$ to $|\mathcal{A}_t|$.

In contrast with Oracle, the LO algorithm does not assume knowledge of future SR batches. As a result, LO may admit low-reward SRs in slot t that prevent the admission of high-reward SRs in the near future, $t+h$, leading to suboptimal overall performance. Predictive Q-learning addresses this limitation by interacting with the environment to learn the “quality” of taking specific actions.

Predictive Q-Learning (PQL). PQL extends standard Q-learning [24] by explicitly incorporating predictions of future link capacity into the SR admission process. In each slot t , PQL considers the same three inputs as LO. PQL sorts arriving SRs within $\mathcal{A}_t$ in decreasing order of rewards R_i . Let $i(j) \in \{\text{sort}(\mathcal{A}_1), \text{sort}(\mathcal{A}_2), \dots, \text{sort}(\mathcal{A}_H)\}, \forall j \in \{1, 2, \dots, N\}$ represent the re-ordered SR index i sorted in decreasing value of rewards R_i within each set $\mathcal{A}_t$. PQL evaluates the addition of SRs to the admitted set $\mathcal{A}_t^*$, one at a time, using an ϵ -greedy policy in which PQL selects a random action $a_{i(j)} \in \{0, 1\}$ (to reject or admit SR $i(j)$) with probability ϵ (exploration) or PQL selects actions according to its Q-table $Q(s_{i(j)}, a_{i(j)})$ with probability $1 - \epsilon$ (exploitation). The exploration rate ϵ decays over iterations j . After selecting action $a_{i(j)}$, PQL receives reward $r(s_{i(j)}, a_{i(j)})$, updates its state from $s_{i(j)}$ to $s_{i(j+1)}$, and considers the admission of SR $i(j+1)$.

The Q-table $Q(s_{i(j)}, a_{i(j)})$ estimates the expected long-term revenue of taking action $a_{i(j)} \in \{0, 1\}$ in state $s_{i(j)} = (\mathbf{n}, \mathbf{A}_{\text{new}}, c_f)$, where $\mathbf{n} = (n_1, \dots, n_{12})$ records the number of active SRs of each type (as defined in Section IV-B), $\mathbf{A}_{\text{new}}$ is a one-hot encoding vector that indicates the type of SR $i(j)$, and c_f represents the number of future time slots, within the short-time horizon H , in which capacity constraints C_t are satisfied if SR $i(j)$ is admitted. If SR $i(j)$ is rejected, the reward $r(s_{i(j)}, a_{i(j)} = 0) = 0$. If SR $i(j)$ is admitted, the reward $r(s_{i(j)}, a_{i(j)} = 1) = R_{i(j)} - R_{i(j)}(1 - c_f/H)\lambda$, where $\lambda > 0$ is a risk-penalty scaling coefficient that controls the cost of anticipated capacity violations and $(1 - c_f/H)$ approximates the probability of capacity violation over the short-term horizon. Penalty scaling with $(1 - c_f/H)$ allows PQL to account for the impact of current admission decisions on future underprovisioning.

The Q-table is updated iteratively according to

$$Q_{\text{new}}(s_{i(j)}, a_{i(j)}) = Q(s_{i(j)}, a_{i(j)}) + \alpha_j \left[r(s_{i(j)}, a_{i(j)}) + \max_{a' \in \{0, 1\}} Q(s_{i(j+1)}, a') - Q(s_{i(j)}, a_{i(j)}) \right]. \quad (18)$$

where α_j is the learning rate at iteration $j \in \{1, \dots, N\}$, determined by the occurrence count $o(s_{i(j)}, a_{i(j)})$ of the state-action pair $(s_{i(j)}, a_{i(j)})$. The learning rate is updated as $\alpha_j = 0.5/o(s_{i(j)}, a_{i(j)})$. This choice of α_j ensures that the learning rate decreases over time as the agent becomes more experienced with the state-action pair, which is crucial for the convergence of the Q-learning algorithm. The discount factor is set to 1 to maximize long-term utility. Through iterative updates, PQL gradually learn SR admission policies that are

Algorithm 1 Predictive Q-Learning (PQL) for SR Admission

```

1: Initialize Q-table  $Q(s, a) \leftarrow 0, \forall (s, a)$ ; occurrence counts  $o(s, a) \leftarrow 0, \forall (s, a)$ 
2: Initialize exploration parameters  $\epsilon$ , decay_rate, min_epsilon
3: for each time slot  $t$  do
4:   (Optional) If current allocations violate  $C_t$ , Rate Control, otherwise Continue
5:   Let  $\mathcal{A}_t$  be the set of SRs arriving at slot  $t$ 
6:   if  $\mathcal{A}_t \neq \emptyset$  then
7:     Sort  $\mathcal{A}_t$  in decreasing order of rewards
8:     for each SR  $i(j)$  do
9:       Compute predicted future capacity violations  $c_f$  in next 5 minutes
10:      Observe state  $s_{i(j)} \leftarrow (\mathbf{n}, \mathbf{A}_{\text{new}}, c_f)$ 
11:      Use  $\epsilon$ -greedy policy to select action  $a_{i(j)}$ 
12:      Execute  $a_{i(j)}$  (admit if 1; reject if 0)
13:      Compute reward
14:       $r(s_{i(j)}, a_{i(j)}) \leftarrow 0$  if  $a_{i(j)} = 0$ 
15:      else  $r(s_{i(j)}, a_{i(j)}) \leftarrow R_{i(j)} - R_{i(j)}(1 - c_f/H)\lambda$ 
16:      Observe next state  $s_{i(j+1)}$ 
17:      Increment occurrence count  $o(s_{i(j)}, a_{i(j)}) \leftarrow o(s_{i(j)}, a_{i(j)}) + 1$ 
18:      Update learning rate  $\alpha_t = 0.5/o(s_{i(j)}, a_{i(j)})$ 
19:      Update Q-value  $Q(s_{i(j)}, a_{i(j)})$  using (18)
20:   Decay exploration rate:  $\epsilon \leftarrow \max(\epsilon \cdot \text{decay\_rate}, \text{min\_epsilon})$ 

```

robust to future capacity fluctuations and converges to a policy $Q^*(s, a)$ that maximizes the long-term expected revenue. The detailed description of PQL is given in Algorithm 1.

VI. EVALUATION

In this section, We evaluate AttIRNN prediction performance and the SR admission control performance. Using both real-world datasets described in Sec. IV, we evaluate AWARe-SAC using: (1) up to 30 hours of SR generation; and (2) up to 34 hours of measurements from six V-band links, spanning three precipitation events (March–April 2024), with rainfall intensities ranging from 5.45 to 45.7 mm/h. To characterize the volatility of each simulated scenario, we compute its Coefficient of Variation (CV), defined as the ratio of the standard deviation to the mean of the RSL measurements over 1-hour intervals. Higher CV indicates more volatile links, often due to intense rainfall.

In evaluating AttIRNN, we compare prediction baselines and architectural variations, with results averaged across all links and grouped by prediction horizon and CV. In evaluating the SR admission control algorithms, each simulated scenario consists of 1 hour of RSL measurements combined with 1 hour of SR generation. The volatility of the scenario is computed using CV. To characterize the maximum underprovisioning time fraction of each simulated scenario, we consider a fictional, non-admissible SR algorithm that admits all SRs and we compute the fraction of time slots the simulated scenario violates QoS. Details on SR generation and RSL measurements are provided in [6, Appendix B].

A. AttIRNN prediction performance

We compare AttIRNN against established statistical and deep learning models, including: (i) Naive approach [25] which predicts $\hat{x}_{t+h} = x_t, \forall h, t$, being most effective in stable conditions; (ii) ARIMA [26], which is a well-known baseline; (iii) TSMixer [27], which is a recent MLP architecture that uses a Mean Squared Error (MSE) loss function; and (iv) DeepAR [28], which utilizes static features and a NLL loss (7) for probabilistic forecasting. For AttIRNN, we use $H = 5$ horizon and $T = 15$ lookback. The static feature vector $\mathbf{x}_s$ includes link length, frequency, antenna type, and link ID to

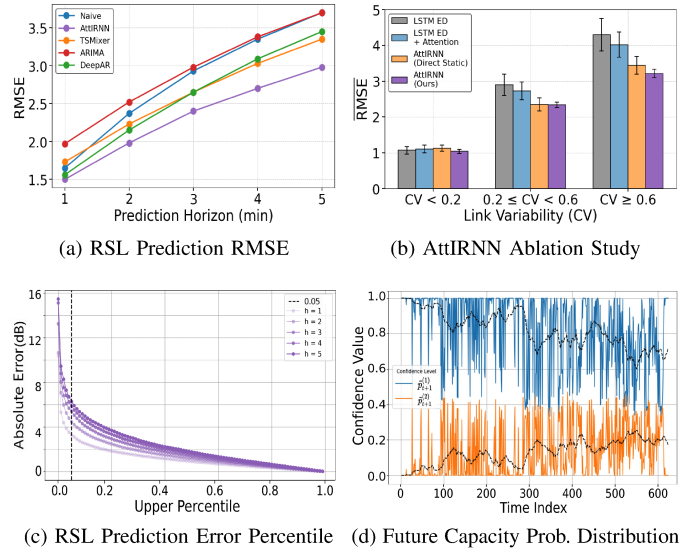


Fig. 4: (a) RSL Prediction RMSE for different SR algorithms as the short-term horizon increases; (b) Comparison of AttIRNN with LSTM encoder-decoder and temporal attention variants for different CV regimes; (c) RSL Prediction Error Percentile for as the short-term horizon increases; and (d) Evolution of the predicted probability distribution of link capacities over time.

account for physical link characteristics that influence signal attenuation patterns. The AttIRNN architecture uses single-layer LSTMs with hidden dimension 64 for encoder and decoder, with link embeddings of dimension 8.

We evaluate the different prediction models (Fig. 4a) using Root Mean Square Error (RMSE) for prediction horizon h :
$$\text{RMSE}_h = \sqrt{\frac{1}{N} \sum_t (x_{t+h} - \hat{\mu}_{t+h})^2}.$$

Training, Validation, and Evaluation Datasets. We split the data from the 25 V-band links, collected within November 2023–April 2024, into: (i) training set, 19 links within November 2023–March 2024; (ii) validation set, 6 held-out links within November 2023–March 2024; and (iii) evaluation set, on the same 6 held-out links but within March–April 2024. All models were optimized via grid search during the training phase. Training uses Adam optimizer, batch size 128, up to 120 epochs with 10 epoch patience early stopping. Models were implemented in PyTorch on an NVIDIA RTX 2080 Ti. See [6, Appendix E] for details.

RSL Prediction RMSE. Fig. 4a shows that AttIRNN consistently achieves the lowest RMSE for all prediction horizons. While most models perform comparably at $h = 1$, statistical baselines (Naive, ARIMA) degrade rapidly, exceeding 3.5 dB at $h = 5$. Fig. 4b displays the mean RMSE across CV regimes, with error bars representing the standard deviation over 10 runs. The Direct-Static variant isolates metadata impact by feeding the static embedding $\mathbf{g}_s$ directly into the decoder, effectively bypassing the Integrated Temporal Attention layer. This configuration confirms that physical link characteristics are primary drivers of accuracy, particularly at high link variability where $\text{CV} \geq 0.6$. While this outperforms the

LSTM ED and LSTM + Attention baselines that lack explicit static integration, our full AttIRNN (Ours) achieves the lowest RMSE and demonstrates stable training across all regimes. These results confirm that static integration provides the vital link-aware context while integrated attention captures dynamic temporal fluctuations to further refine the prediction.

Prediction Error Percentile. Fig. 4c shows the absolute RSL prediction error percentile for $h \in \{1, \dots, H\}$. The dashed vertical line indicates that 95% of all predictions exhibit an absolute error of less than 4 dB at $h = 1$ and 6 dB at $h = 5$. The rarity of large prediction errors ensures reliable capacity forecasting during typical rain-induced attenuation patterns.

Future Capacity Probability Distribution. Fig. 4d shows the evolution of the two most probable values of the capacity probabilities $\tilde{p}_{l,h=1}$ over $l \in \{0, 1, \dots, L-1\}$, denoted as $\tilde{p}_{t+1}^{(1)}$ and $\tilde{p}_{t+1}^{(2)}$. Periods where $\tilde{p}_{t+1}^{(1)}$ drops coincide with rises in $\tilde{p}_{t+1}^{(2)}$, demonstrating the expected shifting of probability mass between discrete capacity levels as link conditions deteriorate during the rain event. This behavior confirms the model's ability to capture dynamic uncertainty across discrete capacity values rather than producing overconfident point estimates.

B. SR Admission Control Performance

We compare the following SR algorithms: (i) Random, which admits each SR with probability 0.5; (ii) Naive Greedy, which admits SRs whenever capacity allows; (iii) Locally Optimal, described in V-B. To reduce complexity, we consider LO that finds an approximate solution by admitting arriving SRs one at a time, in decreasing order of rewards; (iv) Naive Q-learning [7], which assumes static link capacity $C_t = C, \forall t$; (v) Predictive Q-learning, described in Sec. V-B, with $\lambda = 0.5$. Naive and Predictive Q-learning employ lightweight table-based updates and converge within 100k training steps. These Q-learning algorithms have an average runtime of 6 ms per-time-slot when handling arriving batches of 100 SRs, making them suitable for real-time deployment. (vi) DeepNetSlice [29] which is a Deep Reinforcement Learning model that employs an actor-critic architecture with Graph Convolutional Networks to capture dynamic capacity states, allowing the agent to optimize long-term SR admission decisions under fluctuating capacity. However, DeepNetSlice does not leverage capacity predictions for SR admission, what limits its ability to mitigate underprovisioning penalties. (vii) Oracle benchmark, described in Sec. III. When underprovisioning occurs, all SR admission control approaches use Rate Control (14)-(17) to minimize their penalty. For fairness, all methods are evaluated using the same reward and penalty function.

SR Generation and Underprovisioning Penalty. Simulations include the 12 types of SRs described in Sec. IV-B, four per 5G service. The price per throughput unit per slot is $\rho_i \in \{2.5, 5, 10\}$ for BE, eMBB, and URLLC SRs, respectively. The piecewise-linear penalty functions have $m_i = 2$ segments each, with slope and intercept of the segments determined by the equations $a_{ik}(1 - f_{it}) + b_{ik}$. The coefficients are in Table I. Additional details about the penalty coefficients utilized in our evaluation can be found in [6, Appendix C].

TABLE I: Coefficients of the Penalty Functions used in Fig. 5.

	Segment 1		Segment 2	
5G Service	Slope	Intercept	Slope	Intercept
URLLC	6	-2	2	0
eMBB	3	-1	1	0
BE	1.5	-0.5	0.5	0

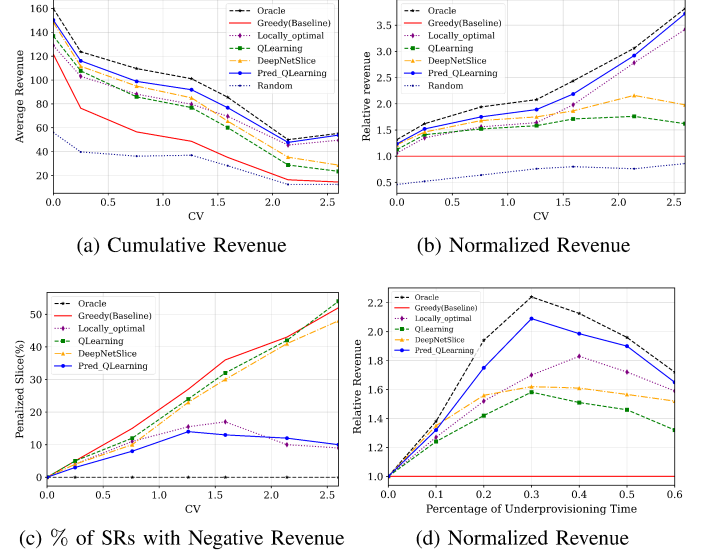


Fig. 5: (a) Revenue of different SR algorithms as CV increases; (b) Same as (a) with revenue normalized with respect to Naive Greedy; (c) Percentage of SRs with negative revenue as CV increases; (d) Revenue as the maximum underprovisioning time fraction increases.

Q-Learning Training and Evaluation. For the SR admission control experiments, we train Q-learning using all available SR generation and link-capacity data described in Sec. IV. Specifically, Q-learning updates its Q-table online and converges over the entire collection of SR generation and mmWave link-capacity traces. After convergence, we freeze the learned Q-table and evaluate the resulting admission policy on each individual simulated scenario, where each scenario consists of one hour of SR generated from residential traffic traces combined with one hour of measured link capacity from a V-band mmWave link. All baseline admission control algorithms are evaluated on the same set of scenarios, ensuring a fair comparison across methods.

Impact of Link Fluctuation. Fig. 5a shows the cumulative revenue for different SR algorithms under varying link CVs. In general, higher CV results in lower revenue due to added underprovisioning penalties. Fig. 5b normalizes revenue of the different SR algorithms with respect to the Naive Greedy baseline. For low CVs, all SR algorithms (except Random) achieve similar revenues. As CV increases, the revenue of Naive Greedy and Random become more similar. More importantly, for high CVs, predictive SR algorithms, namely Predictive Q-learning and Locally Optimal, can achieve revenues $3.5\times$ larger than Naive Greedy, while reactive SR algorithms, namely Naive Q-learning and DeepNetSlice, can achieve

revenues $1.7\times$ larger than Naive Greedy. This significant difference between predictive and reactive SR algorithms is due to the tight integration of link capacity forecasting into SR admission control. Notably, Predictive Q-learning consistently outperforms all SR algorithms, with revenue that remains close to the (idealized) Oracle for all values of CV, suggesting that short-term capacity prediction combined with Q-learning can achieve near-optimal performance.

To further highlight the benefit of predictive SR algorithms, Fig. 5c shows the percentage of admitted SRs whose cumulative penalties P_{it} exceed their lump sum reward R_i . As CV increases, all reactive SR algorithms experience increasing penalty rates. In contrast, predictive SR algorithms maintain substantially lower penalty rates, suggesting that explicit prediction can significantly reduce costly QoS-violation penalties.

Impact of Underprovisioning. Recall that each simulated scenario is associated with a maximum underprovisioning time fraction. A low underprovisioning fraction indicates a simulated scenario with sufficient resources. A high underprovisioning fraction signals overload. As shown in Fig. 5d, when the fraction is low, all policies can admit (almost) all SRs and achieve similar rewards. However, when this fraction is between 0.2 and 0.4, (i.e., the operator has moderate resource availability), Predictive Q-learning greatly outperforms others by making better admission decisions. For high underprovisioning fractions, where resources are extremely scarce, all SR algorithms (including Oracle) degrade due to limited capacity.

In [6, Appendix E], we discuss additional simulation results that evaluate predictive SR algorithms under different piecewise-linear penalty functions and when integrated to different RSL prediction algorithms. The results suggest that the proposed Predictive Q-learning algorithm consistently outperforms other SR algorithms across diverse scenarios.

VII. CONCLUSION

This paper addressed proactive slice admission control in mmWave x-haul, where weather-induced attenuation makes future capacity uncertain and can turn purely reactive admission into costly QoS-violation penalties. We introduced AWaRe-SAC, a tightly coupled framework that (i) forecasts short-term link capacity as probabilistic distributions and (ii) exploits these distributions in an admission controller to balance immediate slice rewards against anticipated underprovisioning risk. Using real-world measurements from the NYC Mesh deployment and slice requests derived from residential traffic traces, our evaluation showed that incorporating capacity uncertainty into admission decisions yields substantial gains, particularly in volatile conditions. Looking forward, AWaRe-SAC opens several directions for network optimization under uncertainty: extending from a star-like topology (as in Fig. 1) to general multi-hop networks, and integrating joint demand and capacity forecasting.

REFERENCES

- [1] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, 2013.
- [2] P. Nagaraj, "Propagation Characteristics of mmWave and Terahertz Frequencies in 6G Systems," *IEEE Trans. Wireless Commun.*, vol. 13, no. 2, pp. 1–18, 2023.
- [3] Y. Niu, Y. Li, D. Jin, L. Su, and A. V. Vasilakos, "A survey of millimeter wave communications (mmWave) for 5G: opportunities and challenges," *Wireless Netw.*, vol. 21, pp. 2657–2676, 2015.
- [4] ITU-R, "Specific Attenuation Model for Rain for use in Prediction Methods (Recommendation ITU-R P.838-3)," International Telecommunication Union, Geneva, Switzerland, Tech. Rep., 2005.
- [5] D. Jacoby, S. Yu, Q. Hu, Z. Hine, R. Johnson, J. Ostrometzky, I. Kadota, G. Zussman, and H. Messer, "Openmesh: Wireless signal dataset for opportunistic urban weather sensing in new york city," *Earth System Science Data Discussions*, vol. 2025, pp. 1–27, 2025.
- [6] D. Jacoby, Y. Li, S. Yu, N. D. Cicco, H. Messer, G. Zussman, and I. Kadota, "AWaRe-SAC: Proactive slice admission control under weather-induced capacity uncertainty," *arXiv:2601.05978*, 2026.
- [7] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.
- [8] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Dynamic SDN-based radio access network slicing with deep reinforcement learning for URLLC and eMBB services," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2174–2187, 2022.
- [9] Q. Liu, N. Choi, and T. Han, "OnSlicing: Online end-to-end network slicing with reinforcement learning," in *Proc. ACM CoNEXT*, 2021.
- [10] M. Sulaiman, A. Moayyedi, M. Ahmadi, M. A. Salahuddin, R. Boutaba, and A. Saleh, "Coordinated slicing and admission control using multi-agent deep reinforcement learning," *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 2, pp. 1110–1124, 2022.
- [11] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, and S. Guo, "A survey of intelligent network slicing management for industrial IoT: Integrated approaches for smart transportation, smart energy, and smart factory," *IEEE Commun. Surv. Tutor.*, vol. 24, no. 2, pp. 1175–1211, 2022.
- [12] M. Zhao, Y. Zhang, Q. Liu, A. Kak, and N. Choi, "AdaSlicing: Adaptive online network slicing under continual network dynamics in open radio access networks," in *Proc. IEEE INFOCOM*, 2025, pp. 1–10.
- [13] C. Gutterman, E. Grinshpun, S. Sharma, and G. Zussman, "RAN resource usage prediction for a 5G slice broker," in *Proc. ACM MobiHoc*, 2019.
- [14] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive network management in sliced 5G networks with deep learning," in *Proc. IEEE INFOCOM*, 2019, pp. 280–288.
- [15] A. Collet, A. Banchs, and M. Fiore, "Lossleap: Learning to predict for intent-based networking," in *Proc. IEEE INFOCOM*, 2022.
- [16] D. Jacoby, H. Messer, and J. Ostrometzky, "Spatio-temporal model for predicting multivariate weather-induced attenuation in wireless networks," *IEEE Trans. Instrum. Meas.*, 2025.
- [17] D. Jacoby, J. Ostrometzky, and H. Messer, "Short-term prediction of the attenuation in a commercial microwave link using LSTM-based RNN," in *Proc. EUSIPCO*, 2021, pp. 1628–1632.
- [18] M. A. Samad and D.-Y. Choi, "Learning-assisted rain attenuation prediction models," *Appl. Sci.*, vol. 10, no. 17, p. 6017, 2020.
- [19] I. Kadota, D. Jacoby, H. Messer, G. Zussman, and J. Ostrometzky, "Switching in the rain: Predictive wireless x-haul network reconfiguration," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 3, pp. 1–26, 2022.
- [20] D. Nandi and A. Maitra, "Study of rain attenuation effects for 5G Mm-wave cellular communication in tropical location," *IET Microw., Antennas Propag.*, vol. 12, no. 9, pp. 1504–1507, 2018.
- [21] S. Yu, T. Koch, I. Mammadov, H. Cao, G. Zussman, and E. Katz-Bassett, "Internet service usage and delivery as seen from a residential network," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 9, no. 2, pp. 1–30, Jun. 2025.
- [22] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
- [23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [24] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.

- [25] R. Hyndman and A. Koehler, “Another look at measures of forecast accuracy,” *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.
- [26] R. J. Hyndman and Y. Khandakar, “Automatic time series forecasting: the forecast package for R,” *J. Stat. Softw.*, vol. 27, pp. 1–22, 2008.
- [27] S.-A. Chen, C.-L. Li, N. Yoder, S. O. Arik, and T. Pfister, “Tsmixer: An all-mlp architecture for time series forecasting,” *arXiv:2303.06053*, 2023.
- [28] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, “DeepAR: Probabilistic forecasting with autoregressive recurrent networks,” *Int. J. Forecast.*, vol. 36, no. 3, pp. 1181–1191, 2020.
- [29] A. Pasquali, V. Lomonaco, D. Bacciu, and F. Paganelli, “Deep reinforcement learning for network slice placement and the deepnetslice toolkit,” in *Proc. IEEE ICMLCN*, 2024, pp. 76–81.

APPENDIX A NOTATION AND CLARIFICATION

Table V summarizes the key notation used in this work. Additionally, we elaborate on several technical details and definitions from the paper.

1) *Training vs. Inference Clarification: Training and Inference Notation.* During training, samples $(\mathbf{X}, \mathbf{Y}) \in \mathcal{D}$ are indexed using a relative prediction horizon $h = 1, \dots, H$. Accordingly, the model produces $\hat{\mathbf{Y}} = \{(\hat{\mu}_h, \hat{\sigma}_h^2)\}_{h=1}^H$, where h denotes the offset relative to the input window.

At inference time (Sec. V-A), predictions are generated online. At each evaluation time step t , the model outputs $\hat{\mathbf{Y}}_t = \{(\hat{\mu}_{t+h}, \hat{\sigma}_{t+h}^2)\}_{h=1}^H$, which correspond to absolute future time indices. These predictions are used immediately by AWaRe-SAC to make admission control decisions.

Teacher Forcing. In (10), AttIRNN uses teacher forcing: during training, $\hat{\mu}_{h-1} = y_{h-1}$ (ground truth), while at inference, $\hat{\mu}_{h-1}$ is the previous prediction. This allows the model to optimize all H steps jointly during training while generating predictions sequentially at inference.

Markov Approximation in Capacity Space. For capacity propagation (Eq. (13)), we use first-order Markov approximation for $h > 1$: $P(C_{t+h}|C_{t+h-1}, \hat{\mu}_{t+h}, \hat{\sigma}_{t+h}) \approx P(C_{t+h}|C_t, \{\hat{\mu}_{t+i}, \hat{\sigma}_{t+i}\}_{i=1}^h)$. This approximation is both computationally tractable and well aligned with the ACM mechanism. In particular, ACM state transitions are governed by hysteresis threshold crossings relative to the current capacity state. The predicted mean and uncertainty $(\hat{\mu}_{t+h}, \hat{\sigma}_{t+h})$ capture the likelihood of these threshold crossings and therefore provide sufficient information for estimating future capacity transitions.

APPENDIX B DATASET DESCRIPTION AND AVAILABILITY

We use two publicly available datasets to support reproducibility and enable future work [5], [21].

Wireless Measurements (OpenMesh): To capture weather-induced attenuation across a city-scale mmWave wireless network, we utilize the OpenMesh dataset [5], comprising one-minute RSL logs from NYC Mesh’s wireless network. NYC Mesh⁴ is a community-driven wireless network deployed in

NYC. The dataset is hosted on Zenodo and formatted according to the OpenSense CML v1.1 NetCDF specifications.⁵ Extended details regarding the network topology, data formats, and standardization are provided in [5].

Residential Traffic for Slice Modeling: To generate realistic slice requests, we use residential traffic traces derived from 520 NYC residences [21]. These traces capture genuine user behavior, including video streaming and peer-to-peer activity. We map raw traffic flows to representative 5G service categories (eMBB, URLLC, and BE) and extract empirical distributions of throughput and duration.

A. mmWave Network infrastructure

In this appendix, we describe the NYC Mesh network infrastructure used for our analysis and the methodology for link selection, expanding on the dataset overview provided in Section IV.

Network Topology and Geography. Our dataset is derived from the NYC Mesh network, a community-owned urban network aiming to provide affordable internet service. NYC Mesh employs a hybrid tiered architecture blending mesh and star topologies, where fiber-connected Supernodes and wirelessly interconnected Hub Nodes form a redundant backbone to distribute connectivity to individual Member Nodes. The infrastructure is concentrated in the dense urban environment of NYC (specifically Brooklyn and Lower Manhattan), as visualized in Fig. 1. This deployment offers a unique testbed for analyzing weather effects typical of metropolitan areas, with further details provided in the OpenMesh dataset [5].

Data Subset and Link Selection. From the OpenMesh dataset [5], we constructed a targeted subset focusing strictly on V-band (60–70 GHz) mmWave links. This selection exploits the heightened sensitivity of these frequencies to atmospheric attenuation, which is advantageous for opportunistic weather sensing where lower frequencies often fail to capture light precipitation. Furthermore, analyzing these bands is critical for ensuring the high availability required by next-generation 5G/6G networks. The final subset was filtered for consistent reporting, with the resulting distribution of path lengths and frequencies illustrated in Fig. 6.

B. Slice Request Generation from Network Traces

As discussed in Sec. IV-B, we used network traces collected from 520 Columbia University residential units to generate the SRs utilized in the evaluation of AWaRe-SAC in Sec. VI. We will detail the mechanism we developed to generate SRs from user demand network traces.

Step 1. Collect and anonymize network packet traces. Packets from/to the 520 off-campus residential units are mirrored from several aggregation switches managed by Columbia University to our servers in a nearby lab. We followed established practices to anonymize sensitive information such as IP

⁴NYC Mesh Community Network.

⁵M. Fencel *et al.*, “OpenSense: Data Formats and Conventions for Commercial Microwave Link Data v1.1,” 2024.

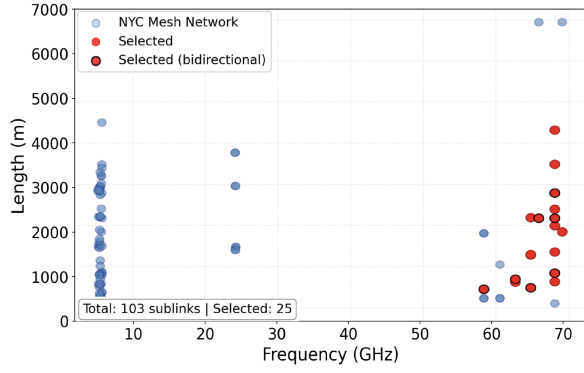


Fig. 6: Scatter plot of link characteristics from the NYC Mesh network. The distribution shows path length versus operating frequency for all 103 available sublinks. The selected subset (red markers) isolates the used V-band links in this work.

addresses and MAC addresses (more details about the network topology and data collection can be found in [21]).

Step 2. Identify flows for eMBB, URLLC, and BE slices.

Following established practices [21], we aggregate packets into flows and identify services accessed by flows. A flow is defined as the packets with the same 5-tuple (i.e., source IP, destination IP, source port, destination port, transport protocol) within an hour. We associate flows with the query domain of the most recent DNS packet with corresponding IP addresses. We also associate TLS flows with their SNI fields. We also annotate flows with their corresponding service when possible, based on DNS/SNI domains, ports, destination IP addresses, and transport protocols. We begin with a curated list of approximately 200 keywords from DNS/SNI domains for 100 popular services. For example, when we see ‘nflx’ in the associated domain, we know that the flow is used for Netflix service. We also use well-known port numbers and ASes to infer the associated services.

As there are no universally accepted industry standards or real-world implementations of network slicing, we identify widely used services that align with the characteristics of eMBB, URLLC, and BE and treat them as representative use cases for the three slicing categories. For the purpose of this study, we include only traffic flows caused by these selected services. For each flow, we know its start time, duration, average throughput, and the service it is used for.

Step 3. Determine slice types based on the throughput and duration statistics.

We assume that, for management simplicity, an infrastructure provider offers only a limited set of slice types, similar to how cloud providers today offer predefined virtual machine options. We further assume that providers will select reasonable options for slicing throughput and duration based on real traffic statistics. Therefore, for this study, we use three duration options—the 90th percentile of flow duration for each of the three slicing service categories. For throughput options, we use the 25th, 50th, 75th, and 95th percentiles of aggregated throughput for flows with the same

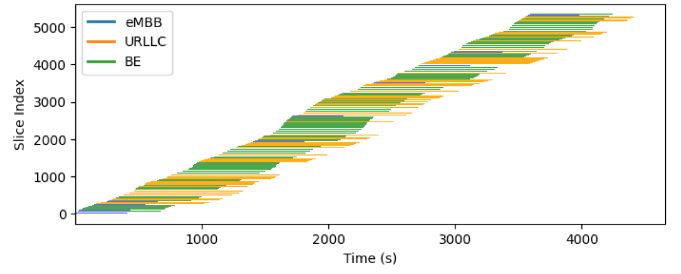


Fig. 7: An example of what slices look like over a one-hour interval. Each line denotes the start and end of a slice, and each color corresponds to a slicing service category.

start time and duration, as tenants are likely to group those flow requests from users when requesting slicing resources. These throughput values are scaled up by a factor of 4× to ensure that demands are aligned with the available link capacity in our simulation, and to anticipate increased bandwidth consumption due to the growing adoption of data-intensive services. The required throughputs for all SRs in our measurement are 0.4, 8.8, 19.2, and 27.2 Mbps. In total, we have 12 slice types.

Step 4. Generate slices. Finally, we generate slices in a manner consistent with expected tenant behavior. A tenant receives flow requests from users, each specifying the service type and required network resources; however, tenants do not know in advance how long a user will maintain a flow, similar to how today’s services lack prior knowledge of session duration. Upon receiving a flow request, the tenant attempts to use existing slicing resources efficiently by placing the flow into an active slice with sufficient available capacity. If no such capacity exists, the tenant requests a new slice to accommodate the flow. Additionally, if a slice expires while the user is still using the service (i.e., the flow has not yet ended), we treat the remaining portion as a new flow request.

For one month of data, we use traffic collected from the same hour of each day (10am-11am) to ensure consistent traffic behavior. For each one-hour traffic trace, we generate slices and treat the result as a single slicing instance for simulation. In total, this yields 30 slicing instances used in our evaluation. We show an example of what slices look like in Fig. 7. X axis represents time, and Y axis represents the slice index. Each line denotes a slice, showing the start and end of a slice. Each color represents a slicing service category.

APPENDIX C

CONVEX PIECEWISE-LINEAR PENALTY FUNCTION

This appendix details the penalty model used to quantify service degradation when admitted slices are underprovisioned. As discussed in Sec. III, we represent the penalty for underprovisioning SRs using a convex piecewise-linear function. Specifically, for each active slice i and time slot t , the penalty is defined as $P_{it} = \max_{k \in \{1, \dots, m\}} \{a_{s(i),k}(1 - f_{it}) + b_{s(i),k}\}$, where $f_{it} \in [0, 1]$ denotes the fraction of requested throughput allocated to the slice. This construction preserves convexity and captures increasing service degradation as the

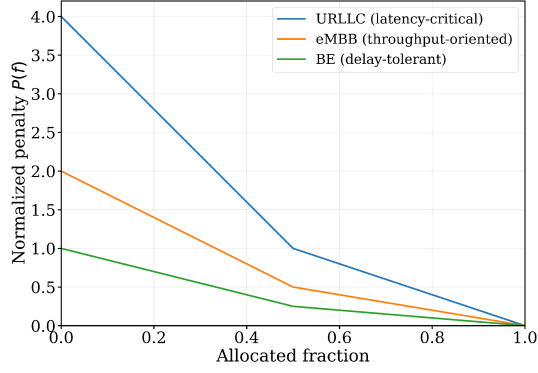


Fig. 8: Penalty P_{it} versus allocated fraction for different service types.

allocated fraction decreases. The coefficients satisfy $a_{s,k} \geq 0$ and $b_{s,k} \leq 0$ and are chosen such that $P_{it} = 0$ when $f_{it} = 1$.

In the evaluation of AWARe-SAC (Sec. VI), we implement the penalty model using service-type-dependent penalty functions via a standard epigraph reformulation:

$$\begin{aligned} P_{it} &\geq a_{s(i),1}(1 - f_{it}) + b_{s(i),1} = \kappa \rho_i(1 - f_{it}), \\ P_{it} &\geq a_{s(i),2}(1 - f_{it}) + b_{s(i),2} = \kappa \rho_i(2 - 3f_{it}), \end{aligned} \quad \forall i, t. \quad (19)$$

We use two linear segments ($m = 2$) for all service types. ρ_i denotes the price per throughput unit per slot of slice i , and $\kappa > 0$ is a global penalty severity coefficient that controls how strongly the operator penalizes underprovisioning. Larger values of κ correspond to more conservative operation with higher sensitivity to QoS violations. The coefficients $(a_{s,k}, b_{s,k})$ capture different tolerance levels to underprovisioning across service types. Latency-critical services incur steeper penalties than best-effort traffic. Fig. 8 illustrates the resulting penalty functions used in Sec. VI. All penalties satisfy $P_{it} = 0$ when $f_{it} = 1$ and increase monotonically as f_{it} decreases, with a steeper slope under severe underprovisioning.

APPENDIX D

STANDARD HYSTERESIS-BASED ADAPTIVE CODING AND MODULATION (ACM)

As introduced in Sec. V-A, we utilize a standard hysteresis-based ACM mechanism to transform RSL values into discrete link capacities. While the principles of ACM are well-established in wireless communications, this appendix details the specific RSL thresholds and Modulation and mapping parameters employed in our analysis.

1) *Predictive ACM Mapping*: Fig. 9 illustrates the impact of a rain event on signal levels, highlighting the transitions between discrete capacity levels $\{C_l\}$ governed by hysteresis thresholds $(C_l^\downarrow, C_l^\uparrow)$. For instance, an RSL drop below -53.5 dBm triggers a reduction to level C_6 , a state that persists until the signal recovers above -51.5 dBm. This

TABLE II: Hysteresis Thresholds for Predictive ACM

Level (l)	Capacity (Gbps)		Level Up (dBm)		Level Down (dBm)	
	af60	wave	af60	wave	af60	wave
7	1.95	1.00	–	–	-52.5	-59.5
6	1.20	0.94	-49.5	-56.5	-55.5	-62.5
5	0.97	0.88	-53.5	-60.5	-59.5	-64.5
4	0.90	0.67	-57.5	-63.5	-63.5	-67.5
3	0.80	0.42	-61.5	-65.5	-67.5	-70.5
2	0.67	0.2	-65.5	-68.5	-71.5	-73.5
1	0.2	0.15	-69.5	-71.5	-75.5	-76.5
0	0.00	0.00	-72.5	-73.5	$-\infty$	$-\infty$

mechanism follows standard ACM principles⁶, adapting the link to the highest sustainable Modulation and Coding Scheme supported by the instantaneous RSL; specifically, we map these thresholds directly to discrete capacity values corresponding to standard modulation levels to focus on the resulting realizable throughput. Crucially, we extend this mapping to the predictive domain—a concept for proactive routing explored in [19]. However, unlike [19], which typically relies on point estimates, we apply the ACM thresholds directly to the *entire* predicted RSL distribution. We model the future RSL as $y_{t+h} \sim \mathcal{N}(\mu_{t+h}, \sigma_{t+h})$, where the parameters are optimized via NLL minimization during training. By propagating these probabilistic forecasts through the discrete thresholds, we compute the explicit probability of each capacity level being available, providing a risk-aware input for our proactive admission control.

Derivation of Thresholds and Capacity Values. Table II lists the discrete capacity levels $\{C_l\}$ and hysteresis thresholds $(C_l^\downarrow, C_l^\uparrow)$ utilized in our ACM scheme. To ensure a robust evaluation, we standardized these parameters by grouping the 68–69 GHz links into two representative clusters based on device type and path length: (1) **AF60-LR** (1.8–2.2 km, 4 links), max capacity $C_7 = 1.95$ Gbps; and (2) **Wave** (0.8–1.4 km, 2 links), max capacity $C_7 = 1.0$ Gbps.

Because raw capacity logs often lack discrete consistency, we derived the standardized mapping empirically by analyzing a single representative link from each cluster. For each RSL interval, we assigned the capacity level C_l corresponding to the *empirical mode* of that representative link’s historical performance. To apply this standard mapping across the remaining links in each cluster, we normalized their signal scales to neutralize installation-specific variations (e.g., antenna alignment). Specifically, we aligned the nominal (clear-sky) RSL for every link to a standardized reference point: 4 dB above the first degradation threshold $C_L^\downarrow$ (the point where the link drops from maximum capacity). This adjustment ensures that the modeled ACM transitions are driven strictly by dynamic atmospheric attenuation rather than static link budget discrepancies.

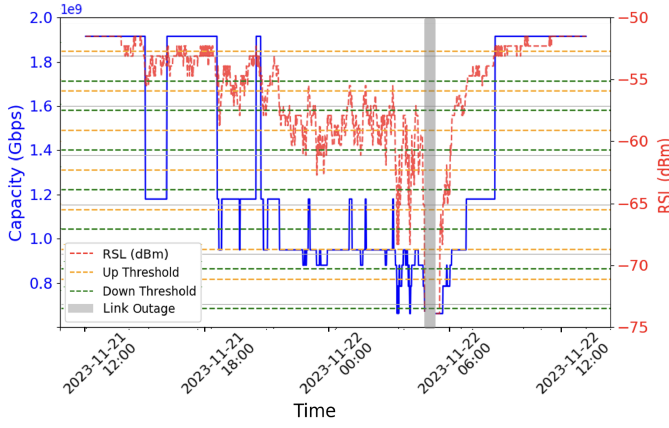


Fig. 9: Capacity levels during rain event (af60lr, Nov 21-22, 2023). RSL $\rightarrow$ capacity conversion via hysteresis (Table II).

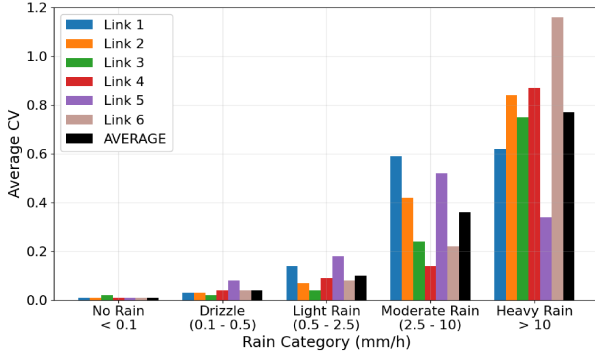


Fig. 10: Average CV categorized by rainfall intensity (WMO).

APPENDIX E EVALUATION DETAILS

Prediction Modeling Strategy and Baselines. To address multi-step short-term forecasting ($H = 5$ min), we adopt a **univariate cross-learning framework**. While we initially explored multivariate architectures setup, including spatio-temporal attention [16], [19] and TSMixer [27] with multi-link prediction objectives, our analysis indicated diminishing returns given our specific link topology and available measurements, ultimately favoring univariate frameworks. This aligns with findings that complex multivariate biases often induce overfitting when covariate correlations are limited, further discussed in [27]. Consequently, for the neural evaluators (DeepAR, TSMixer, AttIRNN) in Sec. VI-A, we train a single global model utilizing a learnable **link-embedding** to capture link-specific dynamics without full multivariate overhead. For Neural Baselines, we adapt **TSMixer** [27] and **DeepAR** [28] from their official PyTorch implementations. Specifically, we adopt the TSMixer-Ext architecture by concatenating flattened embeddings with time-series patches, while for DeepAR we utilize a NLL head for probabilistic regression to output distribution parameters. While in these neural models we

⁶L. Bao, J. Hansryd, T. Danielson, G. Sandin, and U. Noser, “Field trial on adaptive modulation of microwave communication link at 6.8 GHz,” in Proc. EuCAP, 2015, pp. 1–5

freeze the trained weights for real-time deployment, the **Adaptive ARIMA** model, implemented via `pmdarima`, explicitly adapts to evolving patterns using a rolling-window approach to handle non-stationarity. Its optimal order is dynamically selected via AIC [26], with parameters refit every 10 minutes during evaluation. Despite these differing adaptation strategies, both frameworks successfully support real-time forecasting at the minute-scale resolution required for our evaluation.

Data Split and Balancing Strategy: To mitigate the inherent class imbalance where dry intervals vastly outnumber precipitation events, we prioritize rainy periods during training data selection. Training solely on raw chronological traces would bias the model toward predicting zero attenuation (the dominant class). We therefore employ standard statistical wet-dry classification methods to filter the dataset, ensuring a balanced representation of rain and non-rain conditions, adopting the multi-link methodology from [16]. Notably, while this classification could be derived from external meteorological observations, we rely exclusively on signal-based statistics to preserve the system’s stand-alone capability, avoiding dependence on auxiliary weather sensors.

TABLE III: Hyperparameter Grid Search Space and Configurations.

Hyperparameter	Search Range	AttIRNN	DeepAR	TSMixer
<i>General Training Parameters</i>				
Lookback Window (T)	{5, 10, 15, 20} min	15 min	15 min	20 min
Batch Size	{64, 128, 256}	128	64	128
Learning Rate	$\{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$	10^{-3}	10^{-3}	5×10^{-4}
Dropout Rate	{0.0, 0.1, 0.2, 0.5}	0.1	0.1	0.5
<i>Model Complexity Parameters</i>				
Hidden Dimensions	{32, 64, 128}	64	128	32
Layers / Blocks	{1, 2, 3, 4}	1	2	2
Link Embedding Dim.	{4, 8, 16}	8	8	4
<i>Note: “Layers/Blocks” refers to encoder-decoder layers for RNN models and mixer blocks for TSMixer.</i>				

Hyperparameter Optimization. All deep learning models were optimized via grid search to ensure fair comparison, selecting the configuration that minimized validation loss. To strictly prevent data leakage and preserve temporal ordering, the validation set was constructed using the final 10% of the chronological data from the training links, rather than random shuffling. Table III outlines the hyperparameter spaces explored via grid search and the final configurations selected for each model. For architectural depth, we tuned the number of stacked components: encoder-decoder layers for AttIRNN variants, recurrent layers for DeepAR, and mixer blocks for TSMixer (denoted as “Layers/Blocks” in the tables). To strictly isolate architectural contributions, ablation variants inherit AttIRNN’s hyperparameters, whereas baselines are individually optimized for competitive fairness.

Evaluation Setup. We evaluate our predictor and admission-control policy on real-world data from six V-band links excluded from the training set. The test dataset covers 34 hours across three distinct rain events in March–April 2024:

- **Event I** (9 Mar, 03:15–13:45): Mean rain rate 5.45 mm/h, peak 21.6 mm/h, duration 10.5 h.
- **Event II** (23 Mar, 02:00–19:00): Mean rain rate 5.96 mm/h, peak 27.4 mm/h, duration 17 h.

TABLE IV: Prediction Performance across CV and horizon.

CV Range	Step	ARIMA		DeepAR		AttIRNN			
		RMSE	$q_{.95}$	RMSE	$q_{.95}$	RMSE	$q_{.95}$	$\hat{\sigma}$	$\tilde{p}^{(1)}$
< 0.2	t+1	0.30	0.87	0.38	0.98	0.32	0.82	0.18	0.96
	t+3	0.52	1.11	0.55	1.11	0.52	1.07	0.24	0.93
	t+5	1.34	1.44	0.64	1.36	0.58	1.25	0.39	0.91
[0.2, 0.6]	t+1	2.31	4.48	2.00	3.70	1.79	3.26	1.08	0.90
	t+3	3.81	6.40	3.16	5.76	2.88	5.18	1.28	0.84
	t+5	4.47	9.20	4.08	8.32	3.95	7.88	1.46	0.79
> 0.6	t+1	2.83	6.33	2.58	5.94	1.93	5.56	2.00	0.87
	t+3	4.60	10.83	3.91	10.31	3.28	9.20	2.35	0.82
	t+5	5.22	13.71	4.45	12.48	4.16	10.84	2.60	0.72

- **Event III** (3 Apr, 10:00–16:00): Mean rain rate 5.80 mm/h, peak 45.7 mm/h, duration 6 h.

The ground truth precipitation data was sourced from public archived datasets, averaging NOAA and Weather Underground records. We use CV-based metrics to normalize across diverse link characteristics (e.g., frequency, antenna type, path loss), rather than analyzing error sensitivity to individual features.

CV-Based Evaluation. We use the CV to characterize link volatility independently of specific links or time periods. Higher CV indicates more frequent capacity fluctuations due to signal variations during intense rainfall. This enables fair comparison across heterogeneous links and varying weather conditions. Fig. 10 presents the average RSL CV for all evaluated links, grouped by rainfall intensity⁷. Dry periods exhibit near-zero CV, indicating stable capacity, while CV increases sharply with rainfall intensity. Link 5 shows anomalously high CV under heavy rain due to signal outages. In general, heavier rainfall yields higher CV values, while non-rain conditions show near-zero CV, indicating minimal capacity changes. Intense rain increases RSL fluctuations in the links, thereby reducing capacity a relationship shown in Fig. 2. It quantifies throughput stability, with low CV indicating consistent performance and high CV representing significant fluctuations which often require proactive admission control. Table IV aggregates prediction results for ARIMA, DeepAR, and AttIRNN over the entire evaluation period, grouping them into three CV bands (low, medium, high) and three horizons ($h = 1, 3, 5$), with AttIRNN’s confidence metrics also reported. It also reports AttIRNN’s confidence metrics $\hat{\sigma}_{t+h}$ and the mean $\tilde{p}^{(1)}$ to quantify prediction uncertainty. When link variability is low ($CV < 0.2$), all models perform similarly; but at medium ($0.2 \leq CV < 0.6$) and especially high variability ($CV \geq 0.6$), AttIRNN demonstrates a clear advantage, achieving the lowest RMSE and sharply reduced 95th-percentile absolute errors, indicating both higher accuracy and a lower frequency of significant prediction errors.

Impact of Penalty Severity Coefficient. Fig. 11a shows the impact of the penalty severity coefficient κ on the relative revenue achieved by different algorithms as the penalty for underprovisioning increases. As the penalty coefficient grows, the performance gap among predictive models narrows, since

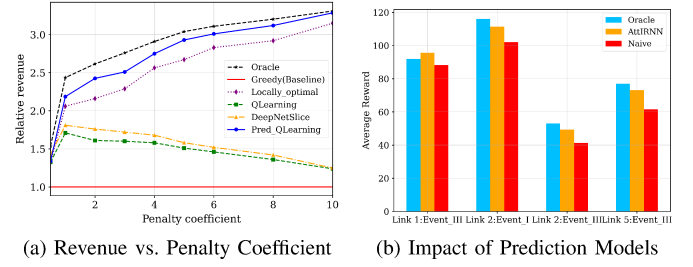


Fig. 11: Revenue under different penalty coefficient or prediction models.

all predictive approaches become increasingly conservative and tend to reject any slice that may introduce future penalties. Consequently, their admission decisions converge, leading to similar revenue outcomes. In contrast, the naive greedy, Q-learning, and DeepNetSlice models perform poorly as they fail to anticipate future capacity constraints and therefore incur substantial penalties.

Impact of Prediction Models. The previous results use AttIRNN as the capacity prediction model. The Naive model is easy to adopt in a system, assuming $\hat{x}_{t+h} = x_t$, and serves as a baseline; despite its simplicity, it can perform reasonably well in noisy or volatile conditions. AttIRNN represents a more realistic learning-based predictor, as it has been shown to outperform other baseline models. The Oracle is used to evaluate the model with zero prediction error, serving as the ideal predictor benchmark. We evaluate the impact of these three prediction models on system performance. Fig. 11b shows that prediction accuracy influences the performance of predictive Q-learning. The Naive model tends to overestimate future capacity, which can lead to overly aggressive slice admissions and degraded performance due to increased underprovisioning penalties. AttIRNN achieves performance close to the Oracle on average. In particular, moderate underestimation of future capacity by AttIRNN can be beneficial, as it encourages more conservative admission decisions that reduce the risk of underprovisioning penalties under volatile network conditions.

⁷World Meteorological Organization, “Manual on Codes: International Codes, Volume I.1, Annex II to the WMO Technical Regulations,” World Meteorological Organization (WMO), Geneva, Switzerland, Tech. Rep. WMO-No. 306, 2017.

TABLE V: Parameters and Decision Variables

Symbol	Description
Perfect-Information MILP Parameters and Variables	
$\mathcal{H}$	Time-horizon.
t	Time index $t \in \{1, \dots, \mathcal{H}\}$.
C_t	Capacity measurement at time slot t .
N	Total number of SRs generated within the time-horizon.
i	Chronological index of SR generation $i \in \{1, \dots, N\}$.
d_i	Throughput requirement of SR i .
D_i	Duration of SR i .
R_i	Lump sum reward for admitting SR i .
P_{it}	Per-slot penalty incurred when SR i is underprovisioned in slot t .
t_i	Slot when SR i is generated.
z_i	Binary decision variable: 1 if SR i is admitted at any time slot, 0 otherwise.
κ	Global penalty severity coefficient.
k_{it}	1 if slice i is active at time slot t , 0 otherwise.
m_i	Number of intervals in the convex piecewise linear penalty function for SR $i \in \{1, \dots, N\}$.
a_{ik}, b_{ik}	Coefficient and intercept of the k -th linear segment, $k \in \{1, \dots, m_i\}$, of the penalty function of SR $i \in \{1, \dots, N\}$.
f_{it}	Fraction of requested throughput assigned to SR $i \in \{1, \dots, N\}$ at time slot $t \in \{1, \dots, \mathcal{H}\}$.
ρ_i	Price per throughput per time unit paid by a network SR.
Capacity Prediction (AtfIRNN) Notation	
$\mathbf{X}_t$	The last- T RSL samples $\{x_{t-T+1}, \dots, x_t\}$.
T	Look-back window hyperparameter (number of historical RSL samples).
$\mathbf{Y}_t$	The next- H future RSL values $\{x_{t+1}, \dots, x_{t+H}\}$.
$\mathbf{x}_s$	Static feature vector of link specifications.
$(\mathbf{X}, \mathbf{Y})$	Generic $\mathbf{X}_t, \mathbf{Y}_t$ supervised input-output training pair from sliding window.
ϕ_j^E	Encoder hidden states summarizing the input window.
$\mathbf{g}_s$	Static embedding of link attributes $\mathbf{x}_s$ (learned vector representation).
ϕ_h^D	Decoder hidden state at future step h .
$\mathbf{g}_h, \tilde{\mathbf{g}}_h$	Attention-weighted encoder context and enriched context combining temporal-static features.
$\beta_{hj}, \tilde{\beta}_{hj}$	Alignment scores and normalized attention weights, respectively.
$\hat{\mathbf{y}}_h$	Stochastic prediction at horizon h : mean $\hat{\mu}_h$ and variance $\hat{\sigma}_h^2$.
$\hat{\mu}_{t+h}$	Predicted mean RSL at time $t+h$.
$\hat{\sigma}_{t+h}^2$	Predicted variance RSL at time $t+h$.
C_l	Discrete capacity level ($l = 0, \dots, L-1$).
$C_l^\downarrow, C_l^\uparrow$	Lower and upper RSL thresholds for capacity level C_l .
$\Phi(\cdot)$	The standard normal Cumulative Distribution Function (CDF).
$\tilde{p}_h(l' \rightarrow l)$	One-step transition probability from capacity level $C_{l'}$ to C_l at time horizon h .
$\tilde{p}_{l,h}$	Estimated likelihood of the link operating at capacity level C_l at time horizon h .
H	Short-term time horizon for capacity prediction.
Slice Admission Control Notation	
$s_{i(j)}$	System state associated with SR $i(j)$, represented as $(\mathbf{n}, \mathbf{A}_{\text{new}}, c_f)$.
$a_{i(j)}$	Action taken for SR $i(j)$, defined as $a_{i(j)} \in \{0, 1\}$ (reject or admit).
$\mathcal{N}_t$	Set of active SR indices at time slot t .
$\mathcal{A}_t$	Set of SRs arriving at time slot t .
$\mathbf{n}$	Vector $\mathbf{n} = (n_1, \dots, n_{12})$ recording the number of active SRs of each type.
$\mathbf{A}_{\text{new}}$	One-hot encoding vector indicating the type of the currently considered SR $i(j)$.
c_f	Predictive indicator of future capacity availability.
$Q(s, a)$	Q-value function estimating the expected long-term revenue of taking action a in state s .
$r(s_{i(j)}, a_{i(j)})$	Immediate reward obtained by taking action $a_{i(j)}$ in state $s_{i(j)}$.
λ	Risk-penalty scaling coefficient.
α_j	Learning rate at iteration j , computed as $\alpha_j = 0.5/o(s_{i(j)}, a_{i(j)})$.
$o(s_{i(j)}, a_{i(j)})$	Occurrence count of the state-action pair $(s_{i(j)}, a_{i(j)})$.
ϵ	Exploration probability in the ϵ -greedy policy.