Sage: A Real-Time AI System for Reducing Latency in NextG Cellular Networks

Aoyu Gong[†]
EPFL

Raphael Cannatà[†]
EPFL

Arman Maghsoudnia
EPFL

Néstor Lomba Lomba
EPFL

Dan Mihai Dumitriu
Pavonis LLC

Haitham Hassanieh
EPFL

Abstract

NextG applications such as AR/VR, industrial automation, cloud gaming, and autonomous robots increasingly demand lower latencies. Current 5G networks, however, incur significant delays due to request-based scheduling, where users must signal demand before the base station can allocate resources for uplink transmissions. In this paper, we present Sage, a real-time AI system that can predict per-user uplink demand at millisecond granularity and proactively allocate resources to reduce uplink latency. Sage proposes traffic trains: a novel abstraction that mitigates distortions to the observed traffic arrivals at the base station and yields stable prediction targets. Sage extracts statistical features from user traffic and retrieves appropriate models from a traffic-aware database of dedicated AI predictors. Sage further executes low-latency inference, error tracking, and online continual learning to dynamically adapt prediction models. Extensive evaluation shows that Sage achieves millisecond-level prediction accuracy with sub-millisecond inference overhead, reducing uplink latency by 2.53× on average across diverse applications while maintaining high resource efficiency.

CCS Concepts

• **Networks** → **Mobile networks**; **Wireless access networks**; **Programmable networks**; • **Computing methodologies** → **Artificial intelligence**.

Keywords

5G, NextG, Cellular networks, Low latency, Real-time AI system, Predictive scheduling

ACM Reference Format:
Aoyu Gong, Raphael Cannatà, Arman Maghsoudnia, Néstor Lomba Lomba, Dan Mihai Dumitriu, and Haitham Hassanieh. 2026. Sage: A Real-Time AI System for Reducing Latency in NextG Cellular Networks. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3789240.3829112

1 Introduction

Over the past decades, cellular networks have achieved remarkable improvements in throughput, to the point where data rates are no longer the primary bottleneck for many applications [16, 17, 79]. The performance frontier has shifted to latency. Modern delay-sensitive applications, like mobile AR/VR, cloud gaming, industrial IoT, autonomous vehicles, mobile sensing, and robot navigation or control, increasingly depend on lower latency [2, 8, 18, 27, 32, 33, 39, 49, 61, 91, 92, 96]. Many of these applications rely heavily on uplink traffic for sending video streams, sensing data, motion control data, etc. and studies show that uplink latency is the dominant portion of the total latency in the cellular network [55, 83]. Our goal in this paper is to reduce uplink latency between users and base stations in Next-Generation (NextG) cellular networks.

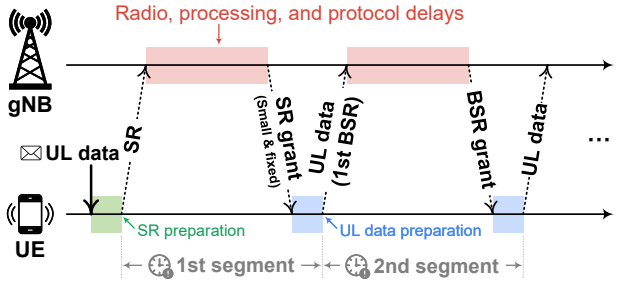


Figure 1: Grant-based access in 5G uplink scheduling.

In today's 5G networks, uplink data is sent through either *grant-based* or *grant-free access*. In *grant-based access*, shown in Fig. 1, the user must request permission to transmit by sending a Scheduling Request (SR), then wait for a grant, transmit a small initial packet with a Buffer Status Report (BSR), indicating the amount of data remaining in its buffer. It then waits again for another grant before sending the remaining data [19]. This incurs at least two round-trip delays, shown as the two latency segments in Fig. 1. In *grant-free access*, uplink resources are pre-allocated to the user such that it can transmit uplink data immediately [56], avoiding the need for the first segment in Fig. 1 and potentially the second segment if the pre-allocated resources are large enough to fit all the data. This reduces latency but wastes resources when there is no data to send [56]. *Grant-based access*, on the other hand, tends to be efficient in terms of allocated resources as it allocates exactly the amount of resources needed by each user at the cost of increasing latency. This problem is exacerbated in Non-Terrestrial Networks (NTNs) for direct-to-satellite 5G connections, where every extra transmission amplifies the latency due to the long propagation delays between devices and satellites [37, 72, 80].

The tradeoff between latency and efficiency is therefore fundamental. *Grant-based access* keeps devices waiting even when data is ready, while *grant-free access* wastes resources and cannot scale well with many users. Neither approach achieves both low latency

[†]Co-primary first authors.

SIGCOMM '26, Denver, CO, USA

ACM ISBN 979-8-4007-2467-1/26/08
https://doi.org/10.1145/3789240.3829112

and high efficiency [107]. In this paper, we show that it is possible to break this tradeoff by allowing the base station to predict *when* and *how much* data a user will send. If the base station can accurately forecast the upcoming traffic, it can pre-allocate just the right amount of resources at the right time, effectively skipping both latency segments shown in Fig. 1.

Prior research has explored traffic prediction in cellular networks [6, 9, 10, 31, 38, 51, 70, 81, 101, 105]. However, most work has focused on large time scales (minutes, hours, or even days) with offline prediction suitable for planning and load balancing, but not millisecond-level decisions. More recent works [83, 99] have proposed real-time prediction and latency reduction frameworks, but only support periodic traffic with small payloads of 100 bytes [83] or < 300 bytes [99] that fit in initial SR grants, as explained in §9.

Building an accurate real-time prediction system is challenging. The base station never sees the "real" traffic arrival process at the user. Instead, it only observes traffic arriving at the base station via SRs, BSRs, and received payloads, which are shaped and delayed based on the 5G protocol configurations, as well as retransmissions caused by decoding errors and packet loss. This distorted view of user-side uplink bursts at the base station makes traffic prediction hard and error-prone. Second, cellular networks must support a wide range of applications with highly different traffic patterns. A single general-purpose AI predictor would be too large to run within millisecond deadlines and too costly to retrain every time a new application emerges. Finally, implementing a fully functional system requires more than just prediction. The system must track prediction errors, adapt scheduling decisions on the fly, handle traffic that is challenging to predict, and update prediction models–all under strict real-time constraints.

In this paper, we present Sage, a full AI system for millisecond-scale uplink traffic characterization, prediction, and proactive scheduling in 5G Radio Access Networks (RANs) that does not require protocol or hardware modifications and can be implemented via software updates. Sage addresses the distorted view of user-side uplink bursts at the base station by proposing the concept of *"traffic trains"*, an abstraction that groups together closely spaced traffic arrivals based on SRs, BSRs, delivered payload sizes, and retransmission feedback. This simple abstraction largely filters out noise and distortions introduced by the 5G system, including buffer quantization, slot-level observability, radio-layer retransmissions, and system configuration diversity, while preserving the intrinsic burst structure of application traffic. It ensures that traffic trains observed at the base station closely track the underlying traffic arrival process at the user.

To handle diverse traffic patterns, Sage uses an AI model database of lightweight, profile-specific AI predictors instead of a single large universal model. Traffic is classified into profiles, each paired with a tailored model, enabling fast inference and incremental updates. Finally, to enable a fully functional system, Sage incorporates a control architecture that runs across the 5G base station, a real-time RAN Intelligent Controller (RIC), a near-real-time RIC, and a shared model database. Sage has five components: (1) *Traffic Characterization* to classify traffic and select an appropriate prediction model. (2) *Model Verification* to track prediction errors and confirm that the selected model continues to perform well. (3) *Prediction and Scheduling* to run inference and pre-allocate resources for incoming traffic when verification succeeds. (4) *Continual Learning* to incrementally adapt the model to current traffic when verification fails. (5) *Database Updating* to update the model database by training and registering new models when continual learning fails.

For hard-to-predict traffic patterns, Sage reverts to *grant-based* or *grant-free access*. For *grant-free access*, Sage introduces an analytical model and uses it to determine an appropriate grant size that balances the tradeoff between latency and resource consumption. Sage's architecture ensures that prediction is used when it is reliable, suspended when it is not, and improved over time through model retraining and database updates.

We implemented a prototype of Sage on our 5G testbed using the open-source srsRAN [78] software stack, which we extended with interfaces to our custom implementations of a real-time RIC and a near-real-time RIC. We further deployed all AI models in the ONNX format [63] to enable real-time inference and online training. We evaluated Sage over the air across several real-world applications, with non-periodic traffic, including surveillance cameras, mobile AR, SLAM, multiplayer games (Counter-Strike 2 and Team Fortress 2), Google Meet, WhatsApp, Zoom, and industrial automation.

Across all applications, Sage reduces latency by an average of 2.53× compared to *grant-based access* while using only 1.59× more resources. It achieves comparable latencies to *grant-free baselines* while using, on average, up to 37.5× fewer resources. With 10 concurrent commercial user equipment running heterogeneous applications, Sage reduces latency by 1.42–2.41× compared to *grant-based access* while using only 1.15–1.55× more resources. Sage achieves a Jain's fairness index above 0.9902 across MAC schedulers and application-arrival rates. As the number of users increases from 2 to 10, Sage remains effective even under heavy load, reducing latency from 2.26× to 1.57× with only from 1.39× to 1.13× more resources, while preserving a Jain's fairness index above 0.9937. For real-time video streaming, Sage reduces network latency per frame by 2.07–2.54× and end-to-end display latency by 1.13–1.23×. Sage predicts traffic arrival times with a median absolute error under 4 ms and arrival sizes with a median relative error under 20 %.

The paper makes the following contributions:

- It introduces the notion of *traffic trains*, an abstraction that reconstructs user-side uplink bursts from distorted base-station-side observations and yields a reliable prediction target.
- It presents a real-time AI system that performs millisecond-level prediction of *traffic trains* at the base station.
- It presents a full system architecture to reduce uplink latency in 5G RANs: traffic characterization, model verification, prediction and scheduling, continual learning, and database updating.
- It introduces an analytical model that characterizes the latency–resource tradeoff as a function of the guaranteed grant size.
- We implement and extensively evaluate Sage over the air.

Project webpage: https://sens.epfl.ch/research/sage

Ethical statement. This work does not raise any ethical issues.

2 Background

The 5G standards refer to base stations as gNBs and user devices as User Equipment (UEs). The standards divide physical resources into frequency-time units called Resource Blocks (RBs). Unlike 4G, which uses fixed-duration time slots of 1 ms, 5G supports flexible

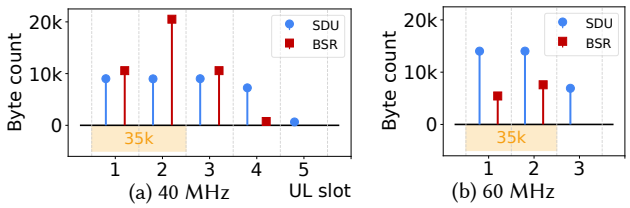


Figure 2: SDUs and BSRs under different channel bandwidth for the same uplink burst of 35K bytes.

slot durations through the numerology parameter μ, where $T_{\text{slot}} = 2^{-\mu}$ ms. With Time Division Duplexing (TDD), each slot is explicitly configured as Uplink (UL) or Downlink (DL).

Uplink scheduling. Because the wireless medium is shared, the gNB must allocate radio resources among competing UEs, a process known as resource scheduling. For UL transmissions, a UE first waits for a Scheduling Request (SR) opportunity (which can take as long as 40 ms in commercial networks) and sends an SR to the gNB to indicate it has pending UL data. The gNB then allocates the UE an initial fixed grant, referred to as the SR grant and denoted by G_{SR}. This initial grant allows the UE to transmit a small amount of data and report its remaining buffer occupancy through a Buffer Status Report (BSR). Although the specifications do not specify G_{SR} [19], it must be sufficient to carry a BSR. A BSR is a 5 or 8 bits field that indicates the amount of data queued in the UE's buffer. Based on this information, the gNB allocates additional resources to the UE in the earliest available UL slots. Choosing G_{SR} involves a trade-off between latency (over-provisioning) and spectral efficiency (under-provisioning). For example, widely used open-source 5G Radio Access Network (RAN) implementations such as srsRAN [78] and OpenAirInterface [60] use SR grants of 512 bytes and 5 RBs, respectively, whereas commercial U.S. operators such as AT&T, Verizon, and T-Mobile typically use 2–3 RBs [83].

Application traffic isolation. A UE can run multiple applications simultaneously, some of which may have low-latency requirements while others do not. In 5G, up to 64 QoS Flow Identifiers (QFIs) are mapped by the SDAP layer to Data Radio Bearers (DRBs) [21]. UEs are mandated to support up to 16 DRBs, which are then mapped to 8 Logical Channel Groups (LCGs). Relevant control signals (e.g., BSRs) are sent per LCG [22]. Hence, one can guarantee traffic isolation by requiring an active low-latency application to be mapped to its own LCG. In practice, UEs run only a few concurrent low-latency applications, typically one. Not every application is required to be mapped to a dedicated LCG, only the foreground low-latency application. Other traffic can be multiplexed on the remaining LCGs, and the dedicated LCG can be reused once the application is no longer active. For simplicity, we will describe the paper assuming one low-latency application per UE.

RAN intelligent controller (RIC). Introduced by the O-RAN alliance [97], the RIC is responsible for controlling and managing radio resources. A RIC can operate at the non-Real-Time (non-RT), near-Real-Time (near-RT), or Real-Time (RT) level [5]. The non-RT RIC handles non-time-critical tasks, including network optimization and analytics. The near-RT RIC handles time-critical tasks such as radio resource management, dynamic network optimization, as well as AI/ML workflows [67]. Although the RT RIC is not yet formally standardized, it has been used for optimizations that require extremely short control-loop times and is therefore often co-located with the gNB [15, 44, 52, 85, 110].

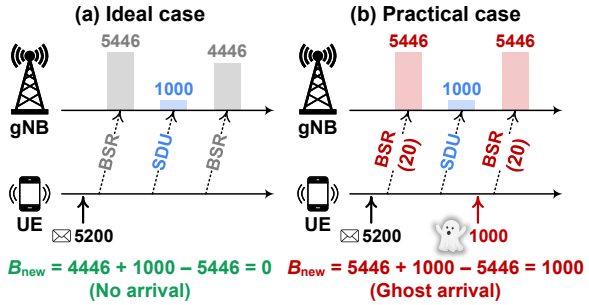


Figure 3: Derived traffic arrival and ghost arrivals.

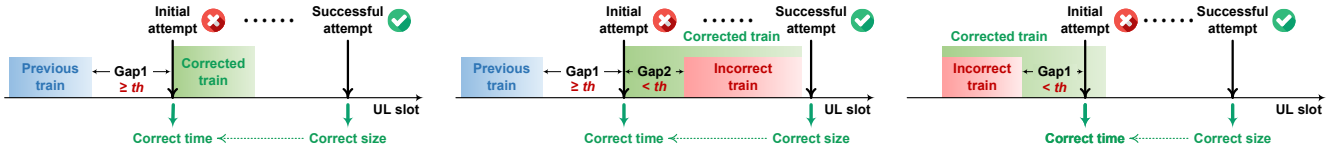
3 Motivation for Traffic Trains

To efficiently perform real-time traffic prediction in NextG systems, it is critical to define prediction targets that are both efficient and robust. A natural choice is to use signals already available at the gNB side, including BSRs and Service Data Units (SDUs). When a UE transmits UL data, its MAC layer segments the data into SDUs, which the gNB then receives. These SDUs reflect the actual payload size being delivered across the radio link. At the same time, the BSR reflects the amount of data queued in the UE's buffer. However, they are unsuitable as prediction targets for two reasons. First, they do not reflect the UE-side arrival process: a single UE-side UL burst might be fragmented into multiple SDUs and trigger multiple BSRs when the allocated grant is insufficient to carry the burst within a single slot. This breaks the mapping between "what arrived at the UE" and "what the gNB observes," leading to redundant predictions for the same burst. Second, BSR and SDU values are entangled with system dynamics like system configurations, scheduling decisions, and modulation/coding schemes. Thus, the same UE-side UL burst can yield very different BSR/SDU values under different network settings. As shown in Fig. 2, changing only the channel bandwidth (from 40 to 60 MHz) can substantially alter the sequence of SDUs and BSRs. This dependency reduces the generalizability of AI predictors and increases the amount of training data needed to cover diverse runtime conditions.

A more stable alternative is for the gNB to derive UE-side UL bursts from BSRs and SDUs, and use these reconstructed events as prediction targets. We refer to this event as a *derived traffic arrival*: an inferred arrival at the UE buffer computed by the gNB from two consecutive BSRs and the SDUs transmitted in between. Specifically, given two consecutive BSRs and the SDUs transmitted in between, the arrival size of this derived traffic arrival is:

$$B_{\text{new}} = \text{Current BSR} + \text{Total SDUs} - \text{Previous BSR},$$

with the arrival time defined as the time slot in which the current BSR is received. As shown in Fig. 3(a), this abstraction filters out system-dynamic effects largely and yields a more consistent view of traffic demand. However, it still suffers from two limitations. The first limitation is the quantized nature of BSRs. BSRs are encoded in discrete levels [19], which mask small buffer changes. For instance, any buffer size between 3910 and 5446 bytes maps to BSR index 20,



(a) Instantiate a missed train. **(b)** Correct an existing train (time and size). **(c)** Correct an existing train (size).

Figure 4: HARQ processes enable retroactive correction of the traffic train sequence by instantiating missed traffic trains in (a), correcting the arrival time and size of an existing traffic train in (b), or only the size in (c).

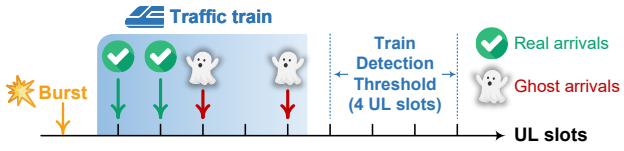


Figure 5: Traffic train and train detection threshold.

which the gNB interprets as 5446 bytes. As long as transmissions keep the buffer within this range, the reported BSR index remains unchanged, and *the gNB may falsely derive new traffic arrivals even though no new data has actually arrived at the UE buffer*, as shown in Fig. 3(b). We define these arrivals as *ghost arrivals*, which inject noise that degrades prediction accuracy by distorting inter-arrival statistics. The second limitation is that derived traffic arrivals are fundamentally constrained by *slot-level observability*. When a single UE-side UL burst spans multiple time slots, as shown in Fig. 2, it can result in multiple derived traffic arrivals. This fragmentation can distort inter-arrival statistics and trigger redundant back-to-back predictions for the same UE-side UL burst.

To overcome these limitations, we introduce the concept of *traffic trains*. Specifically, traffic trains group closely spaced derived traffic arrivals to recover a clearer view of the underlying UE-side arrival process, while minimizing distortions introduced by cellular systems. A traffic train is defined as a sequence of derived traffic arrivals, followed by an idle period consisting of a pre-defined number of consecutive UL slots with no derived traffic arrivals, as shown in Fig. 5. This parameter, referred to as the *train detection threshold*, is configurable at the slice level. Lower thresholds can be used for URLLC slices to capture short bursts that may complete within a single UL slot, while higher thresholds are suitable for eMBB slices to aggregate larger bursts that usually span multiple UL slots. The arrival time of a traffic train is defined as the time at which its first derived traffic arrival is observed following the idle period. Once a new traffic train is detected, this train is considered to end at this detection point. The size of this traffic train is computed using only information available before the detection point. Specifically, it is the total size of SDUs transmitted from the beginning of the train up to the slot at which the final BSR before the detection point is observed, plus the value of that final BSR, minus the value of the last BSR received before the beginning of the train:

$$\sum(\text{SDUs up to final BSR}) + \text{Final BSR} - \text{Pre-train BSR}.$$

Since this final BSR corresponds to the lowest BSR level observed before the detection point, the impact of BSR quantization is minimized for the train. Moreover, by decoupling traffic train detection and its size estimation, we enable early train termination for timely prediction without waiting for the remaining SDUs to be sent.

Sage can also use SRs to improve traffic-train reconstruction. Unlike BSRs and SDUs, an SR does not reveal the amount of buffered data, but it indicates that the UE has UL demand before the corresponding BSRs or SDUs become observable at the gNB. Therefore, when an SR is received after an idle period longer than the train detection threshold, Sage instantiates a provisional traffic train whose arrival time is the SR slot and whose size is finalized once the subsequent BSRs and SDUs are observed.

Traffic trains provide a simple yet powerful abstraction that is not only robust to the diverse system configurations, BSR quantization, and slot-level observability distortions discussed earlier, but also to packet loss and retransmissions. Our traffic train detection exploits Hybrid Automatic Repeat Request (HARQ) processes (a MAC-layer mechanism that allows the gNB to keep track of outgoing grants and request retransmissions for failed packets [19]) as a source of temporal ground truth. In particular, while retransmission delays the successful delivery of SDUs, HARQ preserves the history of all transmission attempts, including the slot of the initial attempt. As a result, although an SDU will only become observable at the gNB after successful decoding, its initial transmission time can be recovered retroactively. We leverage this property to correct the temporal placement of retransmitted SDUs by combining their recovered initial-attempt slot with their sizes obtained upon successful decoding. Retransmitted SDUs can either instantiate new traffic trains that were previously missed because of delayed observations, as shown in Fig. 4a; correct errors in existing trains by adjusting their arrival time and/or sizes, as shown in Fig. 4b and 4c; or merge trains that were previously separated due to packet loss. Specifically, this correction is made by computing (i) $G1$: the gap between the recovered initial-attempt slot and the end of the previous train, and (ii) $G2$: the gap between the recovered initial-attempt slot and the beginning of the next train. Letting th be the train detection threshold, we have

- $G1 \geq th$ and $G2 \geq th \Rightarrow$ instantiate missed train (Fig. 4a).
- $G1 \geq th$ and $G2 < th \Rightarrow$ correct next train (Fig. 4b).
- $G1 < th$ and $G2 \geq th \Rightarrow$ correct previous train (Fig. 4c).
- $G1 < th$ and $G2 < th \Rightarrow$ merge previous and next trains.

Taken together, traffic trains provide a stable abstraction of UE-side UL bursts by filtering out artifacts. This abstraction can transform noisy, fragmented MAC-layer observations into a sequence of burst-level events, facilitating predictions.

4 System Overview

In this section, we present an overview of our system, beginning with the definition of three operational modes:

(1) *Grant-based mode*: the gNB allocates resources to a UE based on SRs and BSRs.

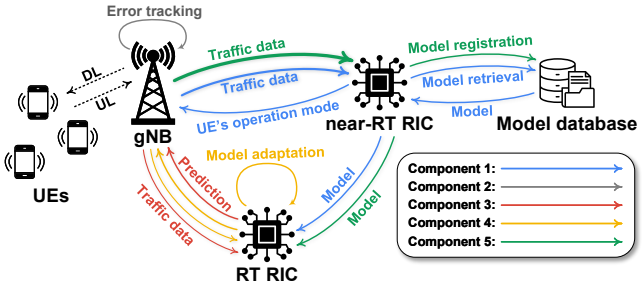


Figure 6: System overview.

(2) *Grant-free mode*: the gNB continuously allocates a guaranteed amount of resources to a UE in every UL slot.
(3) *Prediction mode*: the gNB forecasts upcoming traffic trains, including their arrival times and sizes, based on the most recent observations. It then proactively allocates resources to the UE according to the prediction results.

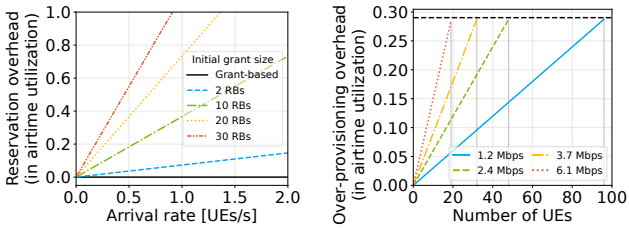
After establishing a connection between the gNB and a UE, the system will traverse five components, as shown in Fig. 6.

Component 1 – Traffic Characterization: The gNB initially assigns the UE to *grant-based* or *grant-free mode* and collects a time series of the UE's traffic trains. This time series is then forwarded to the near-RT RIC, which stores it and triggers the model-selection pipeline once a predefined volume is reached. The pipeline first applies the Augmented Dickey-Fuller test [12, 13] to assess whether the time series is stationary. If the time series is stationary, the near-RT RIC extracts a feature vector, identifies the most similar traffic profile through similarity matching, and retrieves the corresponding prediction model from the model database, as shown in Fig. 6 and detailed in §5.1. The selected model is then forwarded to the RT RIC for subsequent tasks. If the time series is non-stationary (i.e., difficult to predict), the near-RT RIC instructs the gNB to assign the UE to *grant-based* or *grant-free mode* according to slice-specific policies. For UEs operating in *grant-free mode*, we develop an analytical model that characterizes the trade-off between latency and resource consumption to determine appropriate grant sizes. The mathematical derivation of this analytical model is provided in App. A1.

Component 2 – Model Verification: The gNB keeps the UE in *grant-based* or *grant-free mode* and starts submitting its inference tasks to the RT RIC, which executes the assigned prediction model on the collected traffic. Specifically, it performs inference to predict future traffic trains and returns the prediction results to the gNB. The gNB quantifies the prediction error by comparing the results against detected traffic trains. As detailed in §5.3, once a sufficient number of predictions have been evaluated, the gNB assesses if the model's performance satisfies a predefined accuracy threshold.

Component 3 – Prediction & Scheduling: If the model passes verification, the gNB assigns the UE to *prediction mode*, enabling proactive UL resource allocation based on predicted traffic trains. The gNB continues to track the prediction error after assigning the UE to *prediction mode*, verifying if the performance remains above the threshold.

Component 4 – Continual learning: Conversely, if the model fails verification, the system triggers continual learning. During this period, the gNB assigns the UE to *grant-based* or *grant-free mode* and uses the most recent traffic trains to incrementally update the model through supervised learning in the RT RIC as detailed in §5.2. Each continual-learning step is immediately followed by an inference step, after which the updated model is re-evaluated as in Component 2. This cycle, spanning Components 2–4, iteratively continues as the system adapts to the UE's traffic.



(a) Bootstrapping phase. **(b)** Steady-state phase.

Figure 7: Overhead in airtime utilization.

Component 5 – Database updating: If continual learning fails to improve the model's performance after repeated attempts, the gNB requests the near-RT RIC to update the model database. Specifically, the near-RT RIC trains the model on a longer time series of traffic trains and registers the resulting model, together with its feature vector, as a new entry in the database, as described in §5.4. Once training is complete, the near-RT RIC forwards the new model to the RT RIC, enabling subsequent prediction and evaluation as in Components 2 and 3. If the new model still fails, the gNB assigns the UE to either *grant-based mode* or *grant-free mode* using a grant size determined by the analytical model derived in App. A1.

We analytically analyze the impact of Sage on airtime utilization and fairness during the bootstrapping phase (*grant-based* and *grant-free modes*) and the steady-state phase (*prediction mode*) under varying churn rates, numbers of UEs, and bitrates. The mathematical derivation is given in App. A2. We plot the reservation overhead in Fig. 7a as a function of churn rate for varying grant sizes during grant-free bootstrapping. In contrast, grant-based bootstrapping incurs no such overhead. During the steady-state phase, SAGE incurs, on average, a constant multiplicative over-provisioning overhead. The aggregate overhead increases linearly with the number of UEs, as shown in Fig. 7b. This overhead reduces the number of UEs supportable before airtime saturation, compared to grant-based access.

5 System Design

5.1 Traffic-Aware Model Database

We propose using lightweight prediction models tailored to individual traffic profiles, rather than a universal model. A universal model must be provisioned for the most challenging traffic profiles, which can incur high inference latency [41] and become impractical for latency-critical cellular networks. As new traffic profiles emerge, such a model must be trained on an expanding aggregated dataset and grow in complexity to capture increasing variability, leading to repeated retraining with longer training times. Moreover, updates intended to improve performance on new traffic profiles may degrade the performance on previously supported ones, requiring re-evaluation across all traffic profiles. In contrast, lightweight models support modular, profile-specific design and enable incremental updates, because only the model associated with the new profile needs to be trained. This modularity improves flexibility by

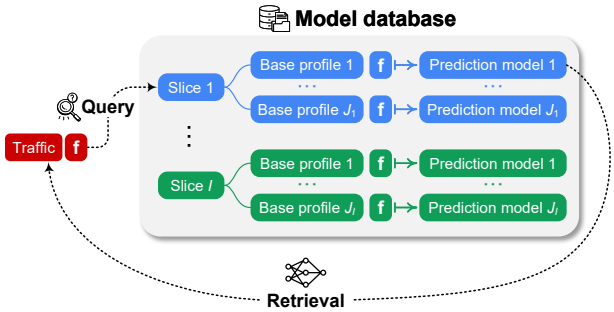


Figure 8: Traffic-aware model database.

allowing individual models to be replaced without affecting others, while reducing computational and energy overhead. These properties make profile-specific lightweight models a practical choice for predictive scheduling.

To achieve this, we propose a model database as shown in Fig. 8, which contains $I \geq 1$ slices. Traffic is first classified by slice because different slices are designed to serve diverse applications with distinct features [42, 69]. Within each slice $i \in \{1, 2, \ldots, I\}$, traffic is further classified into J_i base profiles. These profiles can initially be defined according to target applications and are updated dynamically at runtime, as explained in §5.4. Before deployment, we collect a time series of $M+1$ traffic trains for each base profile: $\{(\tau_0, B_0), \ldots, (\tau_M, B_M)\}$, where τ_m and B_m denote the arrival time and size of the m-th traffic train, respectively. We convert this time series into: $\{(\Delta_1, B_1), \ldots, (\Delta_M, B_M)\}$, where $\Delta_m = (\tau_m - \tau_{m-1})\, T_{\text{slot}}$ for $m \in \{1, 2, \ldots, M\}$. We use inter-arrival time rather than absolute slot indexes, since slot indexes wrap around within each frame and system frame numbers reset every 1024 frames [20], which makes absolute timing ambiguous. Moreover, we do not use inter-arrival slot counts. This is because slot duration varies with numerology, i.e., the same inter-arrival time can correspond to different inter-arrival slot counts under different numerologies.

As a traffic profile is characterized by both its temporal pattern and demand magnitude, we extract statistical features from both $\{\Delta_m\}_{m=1}^{M}$ and $\{B_m\}_{m=1}^{M}$. The features include mean, median, IQR, p_{10}, p_{90}, skewness, MAD, autocorr$_{(1)}$, where IQR is the interquartile range, p_{10} and p_{90} are the 10th and 90th percentiles, MAD is the median absolute deviation, and autocorr$_{(1)}$ is the lag-1 autocorrelation. This leads to two feature vectors $\mathbf{f}_\Delta \in \mathbb{R}^8$ and $\mathbf{f}_B \in \mathbb{R}^8$, which are further concatenated to form the final feature vector $\mathbf{f}$:

$$\mathbf{f} = \mathbf{f}_\Delta \oplus \mathbf{f}_B \in \mathbb{R}^{16}. \tag{1}$$

Such feature vectors serve as reference entries for similarity matching and are used to associate new traffic-train time series with a base profile at runtime. Specifically, when a new time series is collected, the near-RT RIC extracts its corresponding feature vector and computes the norm-log1p distance [89] between this vector and those of all stored base profiles. It then selects the closest profile and retrieves the corresponding pretrained prediction model.

For prediction, we derive the model input and output from a time series of traffic trains. Specifically, for a model with an input sequence length of S, consider the most recent $S+1$ traffic trains

$$\{(\tau_0, B_0), (\tau_1, B_1), \ldots, (\tau_S, B_S)\}.$$

The model input uses normalized inter-arrival times and sizes:

$$\left\{\left(\hat{\Delta}_1, \hat{B}_1\right), \left(\hat{\Delta}_2, \hat{B}_2\right), \ldots, \left(\hat{\Delta}_S, \hat{B}_S\right)\right\}, \tag{2}$$

$$\Delta_s = (\tau_s - \tau_{s-1})\, T_{\text{slot}}, \quad \hat{\Delta}_s = \frac{\Delta_s - \mu_\Delta}{\sigma_\Delta}, \quad \hat{B}_s = \frac{B_s - \mu_B}{\sigma_B},$$

for each $s \in \{1, 2, \ldots, S\}$. Here, μ_Δ and σ_Δ are the mean and standard deviation of inter-arrival times, while μ_B and σ_B are those of sizes. The model outputs the normalized predictions $(\hat{\Delta}'_{S+1}, \hat{B}'_{S+1})$ for the next inter-arrival time and size, which are denormalized to obtain the next predicted traffic train (τ'_{S+1}, B'_{S+1}):

$$\left(\tau'_{S+1}, B'_{S+1}\right) = \left(\tau_S + \left\lfloor \frac{\hat{\Delta}'_{S+1}\sigma_\Delta + \mu_\Delta}{T_{\text{slot}}} \right\rfloor,\ \hat{B}'_{S+1}\sigma_B + \mu_B\right).$$

The system follows a modular design in which predictors function as plug-ins. This design allows operators to assign lightweight models to simple traffic profiles while assigning higher-capacity models to complex ones. Under this plug-in design, Sage can support *a wide range of predictors*, from statistical models to AI-based models, and can readily incorporate future advances in AI.

As shown in Fig. 8, these procedures collectively create a traffic-aware model database, which is initialized offline and queried at runtime. This database allows the system to retrieve the most suitable prediction model based on the current traffic characteristics, where each profile's feature vector serves as a lookup key, and the associated pretrained model acts as the lookup target. The exact number of profiles in realistic deployment depends on the diversity of traffic patterns within each slice, but this does not create a scalability bottleneck: the database can support queries across 10 000 profiles within 200 ms and 50 concurrent queries within 1 s on an Intel NUC. This design ensures that the prediction process is both low-latency and traffic-aware.

5.2 Inference and Continual Learning

After **Component 1**, when the gNB detects a new traffic train for a UE, it submits a prediction task to the RT RIC, which consists of either inference alone or continual learning followed by inference. At runtime, for a UE assigned a model with input sequence length S, the gNB maintains a sliding buffer of the UE's most recent $S+2$ traffic trains $\{(\tau_s, B_s)\}_{s=0}^{S+1}$. This buffer enables the gNB to construct the UE's most recent $S+1$ input data points $\{(\hat{\Delta}_s, \hat{B}_s)\}_{s=1}^{S+1}$ according to Eq. (2). The two procedures are as follows:

- **Inference:** takes $\{(\hat{\Delta}_s, \hat{B}_s)\}_{s=2}^{S+1}$ as input to generate a one-step-ahead prediction of the next traffic train's arrival time and size.
- **Continual learning:** takes $\{(\hat{\Delta}_s, \hat{B}_s)\}_{s=1}^{S}$ as input and $(\hat{\Delta}_{S+1}, \hat{B}_{S+1})$ as the target, forming a supervised training pair. The model is updated for one training epoch, enabling it to incrementally adapt to the current traffic characteristics.

The RT RIC is co-located with the gNB on the same infrastructure, aligning with the emerging AI-on-RAN paradigm [4, 68]. This design ensures that task execution remains isolated from the gNB's main processing pipeline. To improve scalability across UEs, we further design a task scheduler. Assume the gNB has established connections with N UEs. For each UE $n \in \{1, 2, \ldots, N\}$, let E_n denote the elapsed time since UE n's most recent traffic train was detected and $\hat{\mu}_n$ denote the empirical mean inter-arrival time of its past traffic trains. Moreover, let $p_n \in \{0, 1\}$ be a binary indicator of

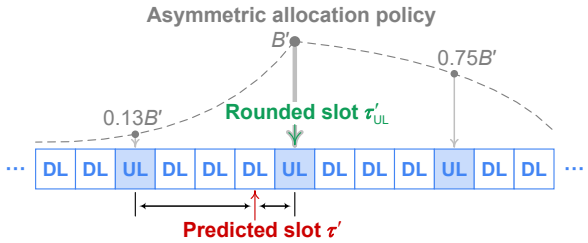


Figure 9: Prediction window.

whether UE n has a pending task. The global system state can be represented as: $[(E_1, \hat{\mu}_1, p_1), \ldots, (E_N, \hat{\mu}_N, p_N)]$. This state updates dynamically upon certain events, including: (i) UEs joining or leaving the network, which changes N; (ii) detection of a new traffic train for UE n, which resets E_n, updates $\hat{\mu}_n$, and sets $p_n = 1$; (iii) dispatch of a pending task for UE n, which sets $p_n = 0$. We consider a single-task buffer per UE, i.e., any pending task is replaced when a new one arrives. This design prioritizes data freshness, which is critical for accurately predicting the next traffic train.

Intuitively, UE n's task has higher urgency when E_n is large and $\hat{\mu}_n$ is small, indicating that another traffic train may arrive soon and that the associated task should be executed promptly. Based on this intuition, we propose a policy that prioritizes tasks based on $\hat{\mu}_n - E_n$. We assume the RT RIC is equipped with multiple concurrent execution threads, with the level of concurrency determined by the hardware parallelism. When $C \geq 1$ execution threads are available and at least one task is pending (i.e., $\sum_{n=1}^{N} p_n > 0$), the scheduler is triggered to make a scheduling decision. Specifically, it selects up to C UEs with pending tasks and the smallest values of $\hat{\mu}_n - E_n$, and dispatches their tasks to the available threads for execution.

5.3 Prediction and Error Tracking

Once inference is complete, the RT RIC notifies the gNB of the predicted next traffic train, including its predicted arrival slot and size: (τ', B'). Due to 5G's flexible frame structure, especially in TDD mode, the predicted slot τ' may not align with a UL opportunity. To address this, the gNB rounds the predicted slot τ' to the nearest UL slot, denoted by τ'_{UL}. To tolerate minor prediction errors and natural variability in traffic arrivals, we introduce a *prediction window* centered at τ'_{UL}. Its half-width, measured in slots, is defined as $W = \Omega\alpha$, where Ω is the base unit measured in slots (e.g., the number of slots in a TDD period) and $\alpha \in \mathbb{N}$.

The gNB pre-allocates resources for all UL slots within the window $[\tau'_{\text{UL}} - W, \tau'_{\text{UL}} + W]$. As UL bursts usually begin at the predicted slot τ'_{UL} and continue into subsequent slots, the resource allocation should not be uniform: fewer resources before the predicted slot τ'_{UL} and more after it. To capture this, we introduce an *asymmetric allocation policy*. Specifically, the resource allocation decays according to exponent γ_L for slots before τ'_{UL} and exponent γ_R for slots after it, where $\gamma_L \leq \gamma_R$. For each UL slot $t \in [\tau'_{\text{UL}} - W, \tau'_{\text{UL}} + W]$, the resource allocation is defined as:

$$A(t) = B' \max\left(0, 1 - \left(\frac{|t - \tau'_{\text{UL}}|}{W + \Omega}\right)^{\gamma}\right),$$

where $\gamma = \gamma_L$ if $t < \tau'_{\text{UL}}$ and $\gamma = \gamma_R$ otherwise. An example with $\Omega = 4$, $\alpha = 3$, $\gamma_L = 0.1$, and $\gamma_R = 1$ is shown in Fig. 9.

We propose an error tracking mechanism that enables the gNB to adaptively switch between operational modes based on prediction quality. Upon detecting the actual arrival slot τ, the gNB computes the prediction error as $e = (\tau'_{\text{UL}} - \tau)^2$. We mainly focus on arrival time prediction errors, as accurate timing is essential for enabling proactive uplink scheduling. If the predicted arrival slot is incorrect, the gNB may miss the UL opportunity to schedule the UE, leading to avoidable delays. While size prediction also matters, moderate over-provisioning can absorb bursts, and under-provisioning can be corrected once the UE reports its remaining buffer occupancy. At runtime, the gNB maintains a sliding buffer of the most recent Q errors, $\{e_q\}_{q=1}^{Q}$, for each UE. A prediction is considered "bad" if $e_q > W^2$, where W is the prediction window's half-width. We define a threshold Q_{mode} such that the gNB assigns a UE to *prediction mode* (i.e., continues to trust the prediction model) if the number of "bad" predictions, Q_{bad}, satisfies:

$$Q_{\text{bad}} = \left|\left\{q \in \{1, 2, \ldots, Q\} \mid e_q > W^2\right\}\right| \leq Q_{\text{mode}}.$$

Otherwise, the gNB assigns the UE to either *grant-based* or *grant-free mode*. This allows for adaptive per-UE control, which tolerates occasional errors while ensuring timely fallback.[1]

To balance efficiency and robustness, we introduce a mechanism to adaptively adjust the window half-width W. Specifically, the gNB compares the number of "bad" predictions Q_{bad} with two thresholds: $Q_- < Q_+$. If $Q_{\text{bad}} \leq Q_-$, the gNB reduces W to reduce resource over-provisioning. If $Q_- < Q_{\text{bad}} < Q_+$, it leaves W unchanged. If $Q_{\text{bad}} \geq Q_+$, it expands W to increase tolerance to prediction errors. The updated window half-width W' is given by:

$$W' = \begin{cases} \max(W_{\min}, W - \Omega), & \text{if } Q_{\text{bad}} \leq Q_-, \\ W, & \text{if } Q_- < Q_{\text{bad}} < Q_+, \\ \min(W_{\max}, W + \Omega), & \text{if } Q_{\text{bad}} \geq Q_+, \end{cases}$$

where $W_{\min}$ and $W_{\max}$ are the minimum and maximum window half-widths. We use a single threshold for operational mode switching, as the decision is binary. By contrast, the prediction window half-width is a multi-level control variable, where using two thresholds introduces hysteresis and prevents oscillation.

5.4 Model Database Online Updating

To ensure long-term adaptability, we support dynamic updates to the model database at runtime. If prediction errors remain persistently high after the continual learning process, the gNB interprets this as a structural deviation between the observed traffic and the matched base profile. In such cases, the gNB requests the near-RT RIC to update the database. Specifically, the near-RT RIC initializes a new model instance from the retrieved model, trains it on a longer time series of traffic trains, and registers the resulting model and its feature vector as a new database entry.

In addition, the near-RT RIC supports extending the model database through external workflows. New traffic profiles and their associated models can be developed offline using application-specific optimizations, such as data augmentation, tailored model architectures, input/output processing, and customized training objectives.

[1] We avoid using moving averages for error tracking, as they react slowly. An occasional outlier can skew the metric for a long time. In contrast, our threshold-based method is fast and local, tolerating isolated errors while quickly responding to persistent ones.

Figure 10: Testbed.

Following validation, the new profiles and models can be integrated into the running system without requiring a restart or interrupting service. The near-RT RIC can be triggered periodically to scan for new external database entries.

6 Implementation

Our system targets the RAN part of the 5G network, using an off-the-shelf core network (Open5GS [48]). It includes one or more gNBs, an RT RIC co-located with each gNB, a near-RT RIC, and a model database. All components are containerized with Docker [14], orchestrated with Kubernetes [87], and deployed via Helm [86].

gNB and RT RIC. We build upon the open-source gNB implementation srsRAN [78] with four key capabilities: (i) metric-collection functionality to collect traffic trains, (ii) an error-tracking mechanism that monitors prediction quality and triggers operational-mode switching, (iii) a communication module using ZeroMQ [103] and Protobuf [28] for communication with the RIC, and (iv) slicing functionality that enables traffic classes to operate in isolation under different policies and operational modes. A lightweight RT RIC, co-located with the gNB, is implemented in C++ with a task scheduler that enables low-latency AI execution.

Near-RT RIC. The near-RT RIC is a centralized controller that manages multiple gNBs. It stores time-series data, applies statistical analysis, retrieves prediction models, and runs the analytical model for *grant-free mode*. When triggered by a gNB, it can retrain models on larger datasets. We implement the near-RT RIC in C++ with the modular xApp framework, where each xApp runs as an independent process and communicates with gNBs through ZeroMQ and Protobuf. Traffic-train time series are stored in TimescaleDB [88] and accessed through an API. The model database is implemented as a repository of prediction models and their feature vectors, organized and indexed using JSON metadata.

AI models. All models are represented in the Open Neural Network Exchange (ONNX) format, a widely supported standard that decouples models from specific training frameworks and enables execution across platforms and languages [63]. Using ONNX Runtime accelerates both inference and training, making AI tasks practical for real-time operation in the RAN. To integrate models, we implement lightweight C++ modules. These modules enable the RICs to load and execute models, while handling input/output conversion between ONNX tensors and standard data structures. AI models function as plug-ins that integrate seamlessly with Sage.

7 Evaluation

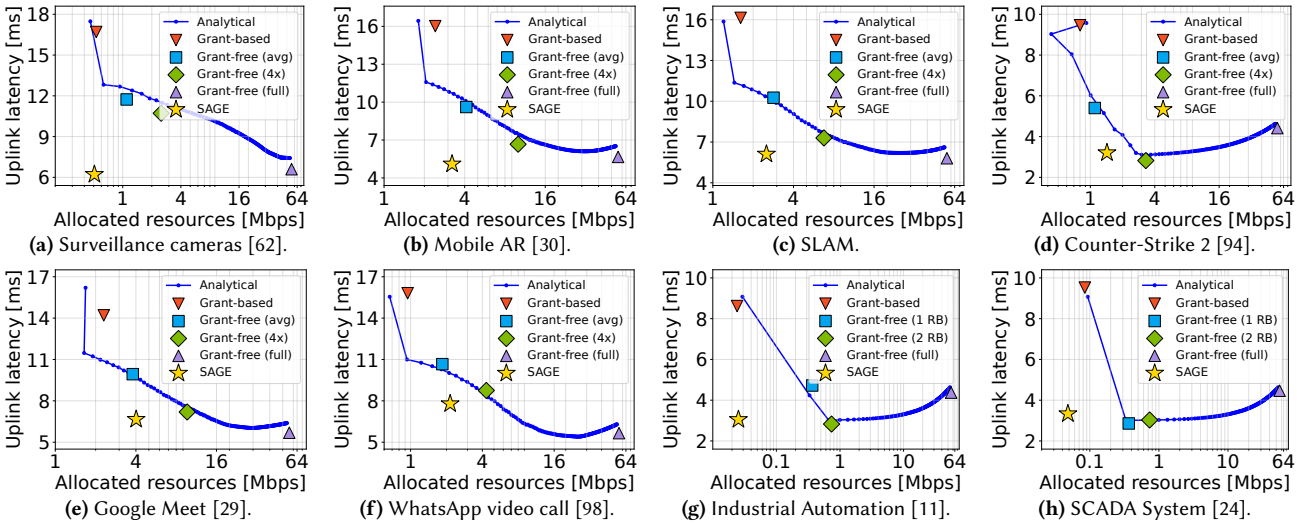
We built a 5G testbed using srsRAN as the gNB, Open5GS as the mobile core, and self-implemented RICs. As shown in Fig. 10, the gNB and RT RIC are co-located on a Dell Precision 5860. The mobile core and near-RT RIC run on an Intel NUC 12. The radio head is a USRP X410, and UE devices are commercial 5G modules. All experiments are conducted over the air in band n78 using numerology $\mu = 1$. UEs are placed a few meters from the gNB in an office building.

We evaluate Sage using 8 real-world applications with different traffic patterns: **(a)** *Surveillance cameras*: streaming the VIRAT Video dataset [62] at 15 FPS with variable bitrate via GStreamer [84], **(b)** *Mobile Augmented Reality (Mobile AR)*: streaming the Ego4D dataset [30] at 30 FPS with variable bitrate via GStreamer, **(c)** *Simultaneous Localization and Mapping (SLAM)*: a mobile robot moving through an automated laboratory and streaming continuous video to an edge server for real-time SLAM processing, **(d)** *Online gaming (Counter-Strike 2)*: multi-participant gameplay on a CS2 server [94], **(e)** *Video conferencing (Google Meet)*: multi-participant calls using Google Meet [29], **(f)** *Video calling (WhatsApp)*: peer-to-peer video calls using WhatsApp [98], **(g)** *Industrial automation*: the "Geek Lounge" dataset [11], which captures industrial control system traffic generated by industrial network equipment, **(h)** *SCADA system*: the MODBUS/TCP traces replayed from [24]. For each application, we collect 5–10 ten-minute traces, pretrain an LSTM model, and evaluate Sage on new experiments at runtime. The model hyperparameters are detailed in App. B1.

7A. Evaluation across different applications. We evaluate Sage across the applications described above. For each application, we compare Sage against four baselines: *grant-based* (▼), *grant-free (avg)* (■), which allocates a guaranteed amount of resources corresponding to the application's average bitrate, *grant-free (4x)* (♦), which allocates four times the average bitrate, and *grant-free (full)* (▲), which allocates the full UL resource grid. For IoT applications, where network usage is very low, we instead compare against *grant-free (1 RB)* (■) and *grant-free (2 RB)* (♦), which allocate a guaranteed budget of 1 RB and 2 RBs, respectively. We consider *grant-based* as it is the conventional access scheme, and adopt realistic NR configurations from srsRAN and two commercial operators. We use *grant-free (avg)* to evaluate whether resource allocation based on long-term average demand is sufficient for bursty traffic, *grant-free (4x)* to assess whether moderate over-provisioning can absorb such bursts, and *grant-free (full)* to provide an intuitive latency lower bound. Moreover, we include an analytical curve from the mathematical model in App. A1, which captures performance across all possible grant sizes and validates our system-level analysis.

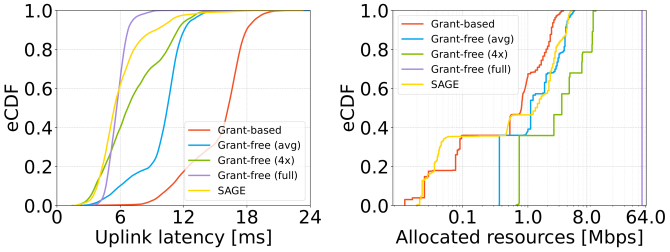
Fig. 11 shows uplink latency versus allocated resources across all 8 applications. Sage achieves consistently lower latency than *grant-based* and provides comparable or better resource efficiency than the *grant-free* baselines. Specifically, Sage reduces latency by up to 3.14× compared with *grant-based* and 1.89× compared with *grant-free (avg)*. Moreover, it reduces allocated resources by up to 4.9× compared with *grant-free (4x)* and 110× compared with *grant-free (full)*. For IoT applications, Sage can match or even surpass the resource efficiency of *grant-based*. This is because SR grants are fixed in size (512 bytes in srsRAN) and often exceed the small payloads of IoT traffic. By predicting actual traffic demands, Sage avoids such over-provisioning and allocates only what is required.

In addition, we highlight the following key observations:
(1) *Average-based baseline falls short.* Allocating resources based on an application's average bitrate may seem sufficient, but bursty traffic quickly invalidates this assumption. An average bitrate of



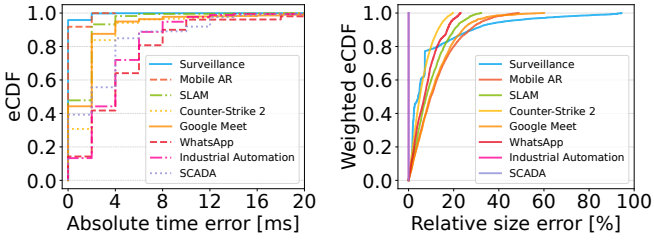
(a) Surveillance cameras [62]. **(b)** Mobile AR [30]. **(c)** SLAM. **(d)** Counter-Strike 2 [94].
(e) Google Meet [29]. **(f)** WhatsApp video call [98]. **(g)** Industrial Automation [11]. **(h)** SCADA System [24].

Figure 11: Uplink latency versus allocated resources across all 8 applications. eCDFs for all 8 applications are shown in Fig. 24.



(a) Uplink latency. **(b)** Allocated resources.

Figure 12: eCDFs of aggregated uplink latency and allocated resources for all 8 applications in Fig. 11.



(a) Absolute arrival-time error. **(b)** Relative arrival-size error.

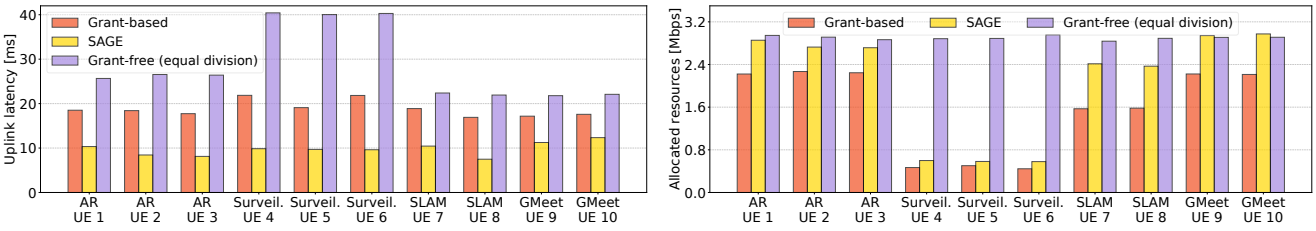
Figure 13: eCDFs of absolute arrival-time error and relative arrival-size error for all 8 applications in Fig. 11.

2 Mbps corresponds to 500 bytes per UL slot in our setup, which is far from enough to serve a burst of 40K bytes. Increasing the allocation to a fixed multiplier (e.g., 4× the average or 2K bytes per slot) still fails to achieve the lowest latency and wastes resources during idle periods. *grant-free (full)* minimizes queueing and scheduling delay, but requires dedicating the full UL resource grid to one UE. (2) *Oversized grants increase uplink latency.* For applications with small payloads, such as Counter-Strike 2, Industrial Automation, and SCADA System, oversized grants can increase uplink latency. When the allocated grant exceeds the available data, the UE applies MAC padding to fill the allocated transport block, which the gNB correspondingly receives and processes [19, 23]. Consequently, *grant-free (full)* incurs additional PHY processing overhead at both the UE and gNB, whereas SAGE achieves lower latency by allocating resources closer to the UE's actual demand.

Fig. 12 aggregates results across all applications, showing the eCDFs of uplink latency and allocated resources. For uplink latency (Fig. 12a), SAGE consistently shifts the distribution left, achieving lower latency than *grant-based* and *grant-free* baselines. For allocated resources (Fig. 12b), SAGE shifts the distribution left relative to *grant-free* schemes, requiring far fewer resources than *grant-free (4x)* and *grant-free (full)*. Quantitatively, SAGE reduces latency by 2.53× compared to *grant-based* and lowers allocated resources by up to 37.5× compared to *grant-free* baselines. Together, these results highlight that SAGE delivers low latency and high resource efficiency. When traffic demand falls below 0.1 Mbps, SAGE can achieve higher resource efficiency than *grant-based*. This is because fixed SR grants over-allocate resources for small packets. We further show the per-application eCDFs in App. B2.

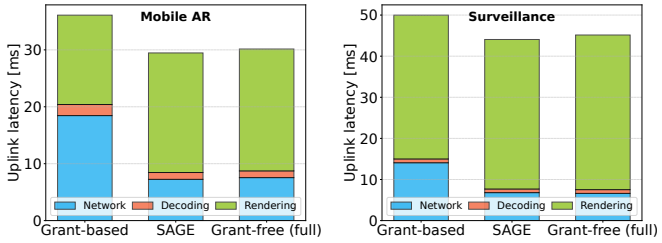
7B. Prediction accuracy. In Fig. 13, we show prediction accuracy using the eCDFs of absolute arrival-time error (the error occurs in 2 ms steps due to the TDD configuration) and relative arrival-size error. For arrival sizes, the eCDF is weighted by bytes to highlight errors on large bursts. Across applications, most predictions achieve millisecond-level accuracy in arrival times and incur less than 20 % relative error in arrival sizes. Together with Fig. 11, these results indicate that timing accuracy is the critical factor. Under- or over-provisioning in arrival sizes can be compensated for or mitigated by the prediction window, but errors in arrival time directly lead to missed scheduling and increased latency.

7C. Application-layer latency. We break down application-layer latency into network, decoding, and rendering latencies in Fig. 15. Compared to *grant-based*, SAGE reduces network latency by 2.07× for surveillance cameras and by 2.54× for mobile AR. Moreover, SAGE achieves performance close to *grant-free (full)*. Interestingly, while decoding and rendering times are determined by the application pipeline, their measured latency appears to increase when



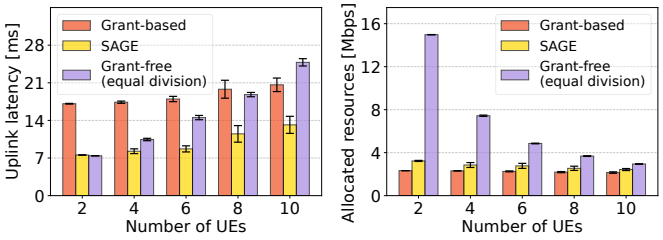
(a) Uplink latency. **(b)** Allocated resources.

Figure 14: Comparison of 10 concurrent UEs running 4 different applications with the PF scheduler, where a new application arrives every 1 s. eCDFs for all 10 UEs are shown in Fig. 28.



(a) Mobile AR. **(b)** Surveillance cameras.

Figure 15: Application-layer latency for video streaming.



(a) Uplink latency. **(b)** Allocated resources.

Figure 16: Comparison across varying numbers of UEs with the RR scheduler, where a new application arrives every 1 s.

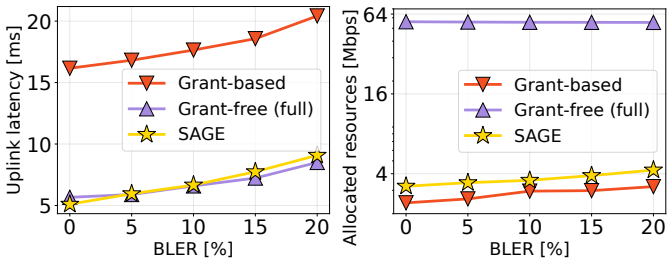
network latency is reduced. This effect arises from fixed application scheduling, which creates more time for the decoding and rendering stages as frames arrive earlier. Nonetheless, a reduction in network latency not only improves overall latency but also enhances robustness by providing subsequent stages with more timing slack.

7D. Multi-UE, multi-application scenario. We evaluate SAGE in a more challenging setting. In this setting, 10 UEs are connected concurrently to the gNB over the air using commercial 5G modems from different vendors, including 5 SIMCom, 3 Telit, 1 Sierra, and 1 Quectel, and run 4 different applications: mobile AR, surveillance camera, SLAM, and Google Meet. All UEs adopt grant-based bootstrapping. We use a 40 MHz channel bandwidth to stress SAGE under limited uplink resources and high contention, and evaluate both Proportional Fairness (PF) and Round Robin (RR) schedulers. We further vary the arrival process, where a new application enters the bootstrapping phase every 1 s or 20 s. We consider *grant-free (equal division)* as the baseline, which partitions the UL resource grid among all 10 UEs and assigns each UE an equal bandwidth share of 10 RBs (bitrate of 2.9 Mbit/s).

Fig. 14 shows the results using the PF scheduler when a new application arrives every 1 s. To show fairness, the eCDF of per-UE latency and allocated resources is shown in Fig. 28. Across all UEs and applications, SAGE consistently reduces latency compared with *grant-based*, achieving a 1.42–2.27× reduction in uplink latency at the cost of only a 1.16–1.54× increase in allocated resources. Compared with *grant-free*, SAGE reduces latency by 1.79–4.19×, while using between 1.02× more and 5.10× fewer allocated resources. *grant-free* can even perform worse than *grant-based*. This happens because when a UE generates a burst that exceeds its reserved resources, the excess bytes need to queue even if other UEs are idle and their reserved resources are unused. The surveillance camera application highlights this limitation most clearly. Under *grant-free*, it experiences the highest latency as bursty large key frames create demand spikes that dominate the overall latency. Thus, *grant-free* provides immediate access, but its scalability quickly becomes limited as the number of UEs increases. Similar trends hold under the RR scheduler and under slower application arrivals, as shown in App. B3. Finally, we quantify fairness using Jain's fairness index in Eq. (4). Across both schedulers and both application-arrival rates, SAGE achieves fairness values above 0.9902 when computed across all UEs and applications.

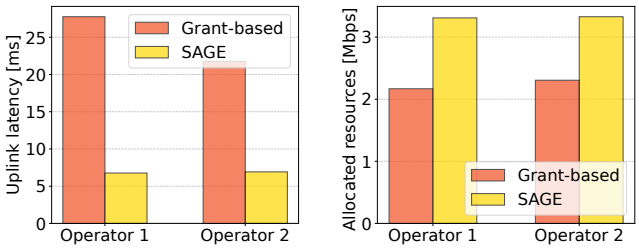
Fig. 16 evaluates the scalability of SAGE as the number of concurrent UEs increases from 2 to 10, with all UEs running mobile AR. Across all UE counts, SAGE consistently reduces uplink latency compared with *grant-based*, achieving latency reductions from 2.26× to 1.57× while increasing allocated resources from 1.39× to 1.13×. The improvement decreases as the number of UEs increases. This is expected because higher contention increases the base uplink latency and reduces the scheduling flexibility available to prediction-based allocation. Even when SAGE accurately predicts upcoming traffic, predicted grants must still be shared fairly among active UEs. Allocated resources by SAGE also decrease as the number of UEs grows, since the scheduler has fewer idle resources available for proactive allocation under higher contention. Nevertheless, SAGE remains effective even at heavy load. With 10 UEs, *grant-based* already uses 73.13 % of the available UL bandwidth, and SAGE uses 82.65 %. Despite this, SAGE reduces latency by 1.57× at the cost of only a 1.13× increase in allocated resources. Similarly, we further compute Jain's fairness index across all UE counts, and all values remain above 0.9937. Overall, these results show that SAGE scales to multi-UE scenarios and remains effective under contention and different traffic, schedulers, application-arrival processes, and UE hardware.

7E. Channel and retransmissions. To test the resilience of traffic trains to varying channel conditions, we conduct experiments under different retransmission rates. In Fig. 18, we show the latency and allocated resources of Mobile AR under varying mean Block



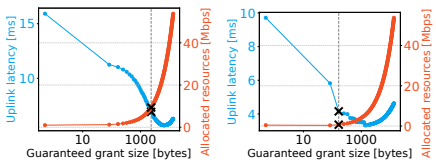
(a) Uplink latency. **(b)** Allocated resources.

Figure 18: Comparison for mean BLER from 0 to 20 %. eCDFs are shown in Fig. 29.



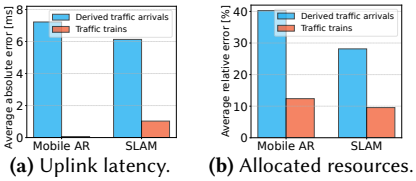
(a) Uplink latency. **(b)** Allocated resources.

Figure 19: Evaluation using two commercial operators' configurations.



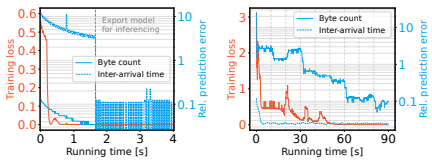
(a) Zoom. **(b)** Team Fortress 2.

Figure 21: Less predictable applications.



(a) Uplink latency. **(b)** Allocated resources.

Figure 22: Comparison of derived traffic arrivals and traffic trains.



(a) IoT traffic. **(b)** Video streaming.

Figure 23: Online continual learning.

Table 1: Comparison for Google Meet with/without best-effort background (BG) traffic saturating the uplink channel.

	Uplink latency	Allocated resources
Without BG traffic	6.7170 ms	4.0066 Mbps
With BG traffic	6.7754 ms	4.0143 Mbps

Error Rates (BLER) from 0 to 20 % using *grant-based*, *grant-free (full)*, and Sage (eCDFs are provided in Fig. 29). Even at higher retransmission rates, Sage manages to achieve latency close to *grant-free (full)*, while maintaining allocated resources close to *grant-based*. We further evaluate Sage's prediction accuracy under highly variable channel conditions (SNR and BLER are shown in App. B4). Sage remains robust, achieving an average resource allocation of 3.45 Mbps and an average latency of 6.99 ms, with prediction errors of 22.25 % in arrival sizes and 0.57 ms in arrival times. Under stable channel conditions, these values are 3.22 Mbps and 5.10 ms, with prediction errors of 14.62 % and 0.16 ms.

7F. Background traffic. To show the isolation between prediction slices and best-effort slices, we measure Sage's performance in two scenarios: (1) a single UE operating in the prediction slice, and (2) the same UE operating in the prediction slice while additional background traffic in the best-effort slice saturates the UL channel. In Tab. 1, we can observe that the presence of background traffic has a negligible impact on Sage's performance. During the evaluation, the background traffic has an average bitrate of 51.98 Mbps.

7G. Commercial network configurations. We evaluate the performance of Sage under system configurations from two national commercial operators, extracted using QCSuper [66]. The key differences between the configuration in §7 and those of the commercial operators are: (1) *SR period*: we set the SR period to the minimum supported by our TDD setup (2 ms), while the commercial operators use 5 ms and 20 ms; (2) *TDD pattern configuration*: we use **DDMU**, while they use **DDDMU**; and (3) *channel bandwidth*: we use 60 MHz, while they use 100 MHz. Fig. 19 shows uplink latency and allocated resources under the two commercial operators' configurations. The longer SR periods in commercial networks increase the average waiting time before UEs can send SRs, making the benefits of Sage more pronounced. Across the two configurations, Sage achieves 3.2–4.1× latency reduction compared to *grant-based*, while using 1.4–1.5× more resources. Although commercial networks are not necessarily optimized for low-latency applications, these results indicate that *Sage remains effective across diverse system configurations without retraining AI predictors, because traffic trains provide a configuration-robust prediction target.*

7H. Less predictable applications. Some applications can exhibit highly irregular traffic patterns, making them difficult to predict reliably. As discussed in §4, UEs are reassigned to *grant-based* or *grant-free mode* under such cases. Our analytical model provides a principled way to automatically determine suitable grant sizes in *grant-free mode* by identifying the Pareto knee points. In Fig. 21, we show this procedure for Zoom [109] and Team Fortress 2 [93].

8 Micro-benchmarks

8A. Traffic trains versus derived traffic arrivals. To show the necessity of traffic trains, we compare predictors trained on derived traffic arrivals to those trained on traffic trains. We evaluate Mobile AR and SLAM, and measure absolute arrival-time error and relative arrival-size error. As shown in Fig. 22, traffic trains provide a significantly more accurate prediction target.

8B. Execution times. We show execution times for the main components of Sage. Inference and Continual Learning (CL) are benchmarked on an AMD Ryzen 7 7840HS using a single thread:

- *Traffic characterization*: 3-second datapoints are collected.
- *Model retrieval*: 149 ms on average.
- *Inference*: 0.051 ms on average across all models in Fig. 11.
- *CL*: 0.451 ms on average across all models in Fig. 11.
- *Initial verification*: 9 datapoints verified; exact timing depends on the application's inter-arrival times.

• *Knee point detection*: 18 ms on average across tested applications, assuming 3 min of collected data.

For reference, the average inter-arrival time measured across tested applications is 59.4 ms. When an application arrives, the one-time pipeline of data collection and model retrieval incurs an initialization delay of 3–10 seconds. This cost is amortized over the lifetime of the connection. Since our predictors are lightweight, we can run 1152.7 inference tasks within 59.4 ms using 30.7 MiB memory, and 131.5 CL tasks within 59.4 ms using 39.2 MiB, all on a single thread.

8C. Continual learning. In Fig. 23, we evaluate continual learning for two workloads: IoT traffic and video streaming. For IoT traffic (Fig. 22a), the model converges within seconds: training loss drops rapidly, and the relative prediction error for both byte count and inter-arrival time stabilizes to below 0.15. Adapted models are exported for inference every 100 epochs, enabling fast adaptation with minimal overhead. For video streaming (Fig. 22b), convergence is slower due to higher variability in the traffic pattern. The training loss initially spikes but steadily decreases, with prediction errors for both targets approaching stable low values within one minute. Such results show that continual learning can quickly adapt models to new traffic patterns at runtime.

Additional microbenchmarks of different traffic predictors are reported in App. C.

9 Related Works

Various models have been developed for cellular traffic prediction, which fall into three categories: statistical, machine learning, and deep learning. Among statistical models, exponential smoothing models [71, 76, 77, 90] have been widely used for cellular traffic prediction for admission control and resource management. Classical machine learning models, such as random forest and Gaussian process regression, have also been applied to cellular traffic prediction [81, 101, 105]. Deep learning models, such as feed-forward neural networks [3, 43], convolutional neural networks [7, 25, 57, 104], and Recurrent Neural Networks (RNNs) [34, 35, 45–47, 58, 75, 82, 102, 108], often outperform these approaches in many cellular traffic-prediction tasks [9, 51]. Most of this work, however, has focused on much longer timescales (minutes, hours, even days) and high-level patterns and is therefore not designed to predict cellular traffic at millisecond granularity.

The closest to our work is [99], which predicts the next 100 UL payloads (based on the previous 1000 slots) to proactively allocate grants and reduce the need for SRs. However, [99] employs a large model that requires 15 to 35 ms per inference when run on a powerful GPU edge server. Moreover, it has been tested only on highly periodic, small-payload traffic (< 300 bytes) [64, 65]. In contrast, Sage provides slot-level predictions using lightweight models, with an average inference latency of 0.051 ms in an end-to-end system that supports automated traffic classification and error tracking.

LRP [83] proposes reducing uplink latency from the UE side by having the UE send a dummy packet ahead of an actual packet arrival. This dummy packet triggers the UE's 5G module to send an SR in advance and receive a grant in time for the actual packet. LRP only requires modifications to UE applications. However, it only supports periodic traffic with known inter-arrival times and sizes smaller than 100 bytes, which avoids the need to predict traffic but significantly limits the range of applications that LRP can support. Sage, on the other hand, focuses on the gNB, predicts traffic sizes to allocate the right amount of resources, and is not limited to periodic or small traffic.

Finally, prior work [1, 74] proposes reactive mechanisms that use SR/BSR signaling to more accurately track the UE's buffer status and avoid outdated BSR values, which can happen in applications with continuous traffic arrivals like file uploads. However, such an approach cannot avoid the two latency segments shown in Fig. 1, which are usually the bottleneck for bursty or periodic traffic. In contrast, Sage takes a complementary proactive approach: by predicting traffic trains directly, it bypasses SRs and BSRs.

10 Conclusion & Future Work

We presented Sage, a real-time AI system that reduces uplink latency in 5G RANs through millisecond-level traffic prediction and proactive scheduling. Sage introduces *traffic trains*, an abstraction that addresses distortions in gNB-side observations introduced by 5G systems and yields a reliable prediction target. Extensive evaluation shows that Sage achieves low uplink latency while maintaining high resource efficiency. Several points are worth discussing:

Bootstrapping modes. The choice between grant-based and grant-free bootstrapping creates a trade-off between scalability and traffic-characterization quality. Grant-based bootstrapping preserves fairness by allocating only requested resources. However, waiting for SR opportunities may delay gNB-side observations, especially under long SR periods, and thus increase uncertainty during traffic characterization. In contrast, grant-free bootstrapping provides cleaner gNB-side observations, since UL opportunities are preallocated and do not depend on SR opportunities. However, it can reduce capacity available to competing UEs under load.

Operators can select the bootstrapping mode according to slice-specific policies. For latency-critical traffic classes, operators can use grant-free bootstrapping to obtain cleaner gNB-side observations and provide low-latency fallback behavior. For delay-tolerant or resource-constrained traffic classes, operators can select grant-based bootstrapping to preserve fairness and spectrum efficiency. Improving traffic-train detection under long SR periods is an important direction for reducing the characterization uncertainty of grant-based bootstrapping.

Channel prediction. Sage focuses on traffic prediction. A complementary direction is to incorporate channel prediction. Forecasting channel quality could allow the system to adapt grant sizes (in RBs) or adjust prediction windows in real time, helping proactive scheduling remain efficient and reliable under fading and interference.

Integration with slice-level scheduling. Our current design focuses on predictive scheduling within a slice. In multi-slice systems, however, predictions from different slices may compete for shared resources, and coordination becomes necessary. One way is to introduce hierarchical scheduling: the near-RT RIC could first aggregate predictions into slice-level demand estimates and then allocate resources across slices according to their priorities and requirements. Within each slice, Sage can still optimize per-UE scheduling.

References

[1] Aby K Abraham. 2015. An optimized method of buffer status reporting for uplink data in LTE. In *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, Coimbatore, India, 1–4. https://doi.org/10.1109/ICECCT.2015.7226188

[2] Alaa A. R. Alsaeedy and Edwin K. P. Chong. 2020. 5G and UAVs for Mission-Critical Communications: Swift Network Recovery for Search-and-Rescue Operations. *Mobile Networks and Applications* 25, 5 (01 Oct 2020), 2063–2081. https://doi.org/10.1007/s11036-020-01542-2

[3] Rodolfo Alvizu, Sebastian Troia, Guido Maier, and Achille Pattavina. 2017. Matheuristic With Machine-Learning-Based Prediction for Software-Defined Mobile Metro-Core Networks. *Journal of Optical Communications and Networking* 9, 9 (2017), D19–D30. https://doi.org/10.1364/JOCN.9.000D19

[4] Ganesh Ananthanarayanan, Matthew Balkwill, Xenofon Foukas, Zhihua Lai, Bozidar Radunovic, Connor Settle, and Yongguang Zhang. 2025. Distributed AI Platform for the 6G RAN. In *Proceedings of the 2nd ACM Workshop on Open and AI RAN (OpenRan '25)*. Association for Computing Machinery, New York, NY, USA, 15–21. https://doi.org/10.1145/3737900.3770167

[5] Bharath Balasubramanian, E. Scott Daniels, Matti Hiltunen, Rittwik Jana, Kaustubh Joshi, Rajarajan Sivaraj, Tuyen X. Tran, and Chengwei Wang. 2021. RIC: A RAN Intelligent Controller Platform for AI-Enabled Cellular Networks. *IEEE Internet Computing* 25, 2 (2021), 7–17. https://doi.org/10.1109/MIC.2021.3062487

[6] Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. 2015. A multi-source dataset of urban life in the city of Milan and the Province of Trentino. *Scientific Data* 2, 1 (27 Oct 2015), 150055. https://doi.org/10.1038/sdata.2015.55

[7] Dario Bega, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Pérez. 2020. DeepCog: Optimizing Resource Provisioning in Network Slicing With AI-Based Capacity Forecasting. *IEEE Journal on Selected Areas in Communications* 38, 2 (2020), 361–376. https://doi.org/10.1109/JSAC.2019.2959245

[8] Gabriel Brown, P Analyst, and H Reading. 2018. Ultra-Reliable Low-Latency 5G for Industrial Automation. *Technol. Rep. Qualcomm* 2 (2018), 52065394. https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/ultra-reliable-low-latency-5g-for-industrial-automation.pdf

[9] Yue Cai, Peng Cheng, Ming Ding, Youjia Chen, Yonghui Li, and Branka Vucetic. 2020. Spatiotemporal Gaussian Process Kalman Filter for Mobile Traffic Prediction. In *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, London, UK, 1–6. https://doi.org/10.1109/PIMRC48278.2020.9217211

[10] Xiaming Chen, Yaohui Jin, Siwei Qiang, Weisheng Hu, and Kaida Jiang. 2015. Analyzing and Modeling Spatio-Temporal Dependence of Cellular Traffic at City Scale. In *2015 IEEE International Conference on Communications (ICC)*. IEEE, London, UK, 3585–3591. https://doi.org/10.1109/ICC.2015.7248881

[11] CS3Sthlm. 2015. Geek Lounge Dataset. https://www.netresec.com/?page=PCAP4SICS. (2015). [Online; Last accessed: 01-Sept-2025].

[12] David A. Dickey and Wayne A. Fuller. 1979. Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *J. Amer. Statist. Assoc.* 74, 366a (1979), 427–431. https://doi.org/10.1080/01621459.1979.10482531

[13] David A. Dickey and Wayne A. Fuller. 1981. Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root. *Econometrica* 49, 4 (1981), 1057–1072. http://www.jstor.org/stable/1912517

[14] Docker Inc. 2013. Docker: Empowering App Development for Developers. https://www.docker.com/. (2013). [Online; Last accessed: 25-Mar-2025].

[15] Salvatore D'Oro, Michele Polese, Leonardo Bonati, Hai Cheng, and Tommaso Melodia. 2022. dApps: Distributed Applications for Real-Time Inference and Control in O-RAN. *IEEE Communications Magazine* 60, 11 (2022), 52–58. https://doi.org/10.1109/MCOM.002.2200079

[16] Ericcson. 2023. *Growth in percent falls, while data traffic rises.* Mobility Report. Ericcson. https://www.ericsson.com/4ae12c/assets/local/reports-papers/mobility-report/documents/2023/ericsson-mobility-report-november-2023.pdf [Online; Last accessed: 24-Dec-2024].

[17] Ericcson. 2024. *Quarterly mobile network data traffic update.* Mobility Report. Ericcson. https://www.ericsson.com/4adb7e/assets/local/reports-papers/mobility-report/documents/2024/ericsson-mobility-report-november-2024.pdf [Online; Last accessed: 24-Dec-2024].

[18] Ericsson. 2019. You need to see our dancing hexapod demo from MWC. (2019). https://www.ericsson.com/en/blog/2019/3/dancing-hexapod-demo-a-success-at-mwc

[19] ETSI. 2024. *Medium Access Control (MAC) protocol specification.* Technical Specification TS 138 321. 3GPP, Sophia Antipolis, France. https://www.etsi.org/deliver/etsi_ts/138300_138399/138321/18.02.00_60/ts_138321v180200p.pdf Version: 18.2.0.

[20] ETSI. 2024. *Physical channels and modulation.* Technical Specification TS 138 211. 3GPP, Sophia Antipolis, France. https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/18.02.00_60/ts_138211v180200p.pdf Version: 18.2.0.

[21] ETSI. 2025. *Evolved Universal Terrestrial Radio Access (E-UTRA) and NR; Service Data Adaptation Protocol (SDAP) specification.* Technical Specification TS 137 324. 3GPP, Sophia Antipolis, France. https://www.etsi.org/deliver/etsi_ts/137300_137399/137324/19.00.00_60/ts_137324v190000p.pdf Version: 19.0.0.

[22] ETSI. 2026. *NR and NG-RAN Overall description.* Technical Specification TS 138 300. 3GPP, Sophia Antipolis, France. https://www.etsi.org/deliver/etsi_ts/138300_138399/138300/19.02.00_60/ts_138300v190200p.pdf Version: 19.2.0.

[23] ETSI. 2026. *Physical layer procedures for data.* Technical Specification TS 138 214. 3GPP, Sophia Antipolis, France. https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/19.02.00_60/ts_138214v190200p.pdf Version: 19.2.0.

[24] Ivo Frazão, Pedro Henriques Abreu, Tiago Cruz, Hélder Araújo, and Paulo Simões. 2019. Denial of Service Attacks: Detecting the Frailties of Machine Learning Algorithms in the Classification Process. In *Critical Information Infrastructures Security*, Eric Luiijf, Inga Žutautaitė, and Bernhard M. Hämmerli (Eds.). Springer International Publishing, Cham, 230–235. https://doi.org/10.1007/978-3-030-05849-4_19

[25] Yin Gao, Man Zhang, Jiajun Chen, Jiren Han, Dapeng Li, and Ruitao Qiu. 2021. Accurate Load Prediction Algorithms Assisted with Machine Learning for Network Traffic. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, Harbin City, China, 1683–1688. https://doi.org/10.1109/IWCMC51323.2021.9498910

[26] Aoyu Gong, Arman Maghsoudnia, Raphael Cannatà, Eduard Vlad, Néstor Lomba Lomba, Dan Mihai Dumitriu, and Haitham Hassanieh. 2025. Towards URLLC with Open-Source 5G Software. In *Proceedings of the 1st Workshop on Open Research Infrastructures and Toolkits for 6G (OpenRIT6G '25)*. Association for Computing Machinery, New York, NY, USA, 7–14. https://doi.org/10.1145/3750718.3750743

[27] Andres J. Gonzalez, Min Xie, Per Hjalmar Lehne, and Pål Grønsund. 2021. Achieving High Throughput and Low Latency with 5G: A Real Implementation Experience. *IEEE Communications Magazine* 59, 10 (2021), 84–90. https://doi.org/10.1109/MCOM.001.2001068

[28] Google. 2001. Protocol Buffers. https://protobuf.dev/. (2001). [Online; Last accessed: 25-Mar-2025].

[29] Google. 2017. Google Meet Video Conferencing. https://meet.google.com/. (2017). [Online; Last accessed: 01-Sept-2025].

[30] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Abrham Gebreselasie, Cristina González, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jáchym Kolář, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Ziwei Zhao, Yunyi Zhu, Pablo Arbeláez, David Crandall, Dima Damen, Giovanni Maria Farinella, Christian Fuegen, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. 2022. Ego4D: Around the World in 3,000 Hours of Egocentric Video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, New Orleans, LA, USA, 18995–19012. https://doi.org/10.1109/CVPR52688.2022.01842

[31] Yan Guo, Shangguang Wang, Ao Zhou, Jinliang Xu, Jie Yuan, and Ching-Hsien Hsu. 2020. User allocation-aware edge cloud placement in mobile edge computing. *Software: Practice and Experience* 50, 5 (2020), 489–502. https://doi.org/10.1002/spe.2685

[32] Saqib Hakak, Thippa Reddy Gadekallu, Praveen Kumar Reddy Maddikunta, Swarna Priya Ramu, Parimala M, Chamitha De Alwis, and Madhusanka Liyanage. 2023. Autonomous vehicles in 5G and beyond: A survey. *Vehicular Communications* 39 (2023), 100551. https://doi.org/10.1016/j.vehcom.2022.100551

[33] Ananya Hazarika and Mehdi Rahmati. 2023. Towards an Evolved Immersive Experience: Exploring 5G- and Beyond-Enabled Ultra-Low-Latency Communications for Augmented and Virtual Reality. *Sensors* 23, 7 (April 2023), 3682. https://doi.org/10.3390/s23073682

[34] Qing He, György Dán, and Georgios P. Koudouridis. 2020. Semi-Persistent Scheduling for 5G Downlink based on Short-Term Traffic Prediction. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. IEEE, Taipei, Taiwan, 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9322125

[35] Qing He, Arash Moayyedi, György Dán, Georgios P. Koudouridis, and Per Tengkvist. 2020. A Meta-Learning Scheme for Adaptive Short-Term Network Traffic Prediction. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2271–2283. https://doi.org/10.1109/JSAC.2020.3000408

[36] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[37] Mohsen Hosseinian, Jihwan P. Choi, Seok-Ho Chang, and Jungwon Lee. 2021. Review of 5G NTN Standards Development and Technical Challenges for Satellite Integration With the 5G Network. *IEEE Aerospace and Electronic Systems Magazine* 36, 8 (2021), 22–31. https://doi.org/10.1109/MAES.2021.3072690

[38] Jin Huang and Ming Xiao. 2022. Mobile Network Traffic Prediction Based on Seasonal Adjacent Windows Sampling and Conditional Probability Estimation. *IEEE Transactions on Big Data* 8, 5 (2022), 1155–1168. https://doi.org/10.1109/TBDATA.2020.3014049

[39] Hongxun Hui, Yi Ding, Qingxin Shi, Fangxing Li, Yonghua Song, and Jinyue Yan. 2020. 5G network-based Internet of Things for demand response in smart grid: A survey on application potential. *Applied Energy* 257 (2020), 113972. https://doi.org/10.1016/j.apenergy.2019.113972

[40] Rajendra K Jain, Dah-Ming W Chiu, and William R Hawe. 1984. *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems.* Technical Report DEC-TR-301. Digital Equipment Corporation.

[41] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. (2020). https://arxiv.org/abs/2001.08361

[42] Benish Sharfeen Khan, Sobia Jangsher, Ashfaq Ahmed, and Arafat Al-Dweik. 2022. URLLC and eMBB in 5G Industrial IoT: A Survey. *IEEE Open Journal of the Communications Society* 3 (2022), 1134–1163. https://doi.org/10.1109/OJCOMS.2022.3189013

[43] Anil Kirmaz, Diomidis S. Michalopoulos, Irina Balan, and Wolfgang Gerstacker. 2020. Mobile Network Traffic Forecasting Using Artificial Neural Networks. In *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, Nice, France, 1–7. https://doi.org/10.1109/MASCOTS50786.2020.9285949

[44] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. 2024. EdgeRIC: Empowering Real-time Intelligent Optimization and Control in NextG Cellular Networks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 1315–1330. https://www.usenix.org/conference/nsdi24/presentation/ko

[45] Tejashri Kuber, Ivan Seskar, and Narayan Mandayam. 2021. Traffic Prediction by Augmenting Cellular Data with Non-Cellular Attributes. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, Nanjing, China, 1–6. https://doi.org/10.1109/WCNC49053.2021.9417547

[46] Varun Kurri, Vishweshvaran Raja, and P. Prakasam. 2021. Cellular traffic prediction on blockchain-based mobile networks using LSTM model in 4G LTE network. *Peer-to-Peer Networking and Applications* 14, 3 (01 May 2021), 1088–1105. https://doi.org/10.1007/s12083-021-01085-7

[47] Jinsung Lee, Sungyong Lee, Jongyun Lee, Sandesh Dhawaskar Sathyanarayana, Hyoyoung Lim, Jihoon Lee, Xiaoqing Zhu, Sangeeta Ramakrishnan, Dirk Grunwald, Kyunghan Lee, and Sangtae Ha. 2020. PERCEIVE: deep learning-based cellular uplink prediction using real-time scheduling patterns. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys '20)*. Association for Computing Machinery, New York, NY, USA, 377–390. https://doi.org/10.1145/3386901.3388911

[48] Sukchan Lee. 2017. Open5GS. https://open5gs.org/. (2017). [Online; Last accessed: 25-Mar-2025].

[49] Jingya Li, Keerthi Kumar Nagalapur, Erik Stare, Satyam Dwivedi, Shehzad Ali Ashraf, Per-Erik Eriksson, Ulrika Engström, Woong-Hee Lee, and Thorsten Lohmar. 2022. 5G New Radio for Public Safety Mission Critical Communications. *IEEE Communications Standards Magazine* 6, 4 (2022), 48–55. https://doi.org/10.1109/MCOMSTD.0002.2100036

[50] John D. C. Little. 1961. A Proof for the Queuing Formula: L = λW. *Operations Research* 9, 3 (June 1961), 383–387. https://doi.org/10.1287/opre.9.3.383

[51] Chunsheng Liu, Tao Wu, Zhifei Li, and Bin Wang. 2021. Individual traffic prediction in cellular networks based on tensor completion. *International Journal of Communication Systems* 34, 16 (2021), e4952. https://doi.org/10.1002/dac.4952 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/dac.4952

[52] Leonardo Lo Schiavo, Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2024. CloudRIC: Open Radio Access Network (O-RAN) Virtualization with Shared Heterogeneous Computing. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 558–572. https://doi.org/10.1145/3636534.3649381

[53] James M. Lucas and Michael S. Saccucci. 1990. Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements. *Technometrics* 32, 1 (1990), 1–12. https://doi.org/10.1080/00401706.1990.10484583

[54] Arman Maghsoudnia, Aoyu Gong, Raphael Cannatà, Dan Mihai Dumitriu, and Haitham Hassanieh. 2026. LatencyScope: A System-Level Mathematical Framework for 5G RAN Latency. (2026). arXiv:cs.NI/2511.21277 https://arxiv.org/abs/2511.21277

[55] Arman Maghsoudnia, Eduard Vlad, Aoyu Gong, Dan Mihai Dumitriu, and Haitham Hassanieh. 2024. Ultra-Reliable Low-Latency in 5G: A Close Reality or a Distant Goal?. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks (HotNets '24)*. Association for Computing Machinery, New York, NY, USA, 111–120. https://doi.org/10.1145/3696348.3696862

[56] Nurul Huda Mahmood, Renato Abreu, Ronald Böhnke, Martin Schubert, Gilberto Berardinelli, and Thomas H. Jacobsen. 2019. Uplink Grant-Free Access Solutions for URLLC services in 5G New Radio. In *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, Oulu, Finland, 607–612. https://doi.org/10.1109/ISWCS.2019.8877253

[57] Jose Mejia, Alberto Ochoa-Zezzati, and Oliverio Cruz-Mejía. 2020. Traffic Forecasting on Mobile Networks Using 3D Convolutional Layers. *Mobile Networks and Applications* 25, 6 (01 Dec 2020), 2134–2140. https://doi.org/10.1007/s11036-020-01554-y

[58] Ahmad M. Nagib, Hatem Abou-Zeid, Hossam S. Hassanein, Akram Bin Sediq, and Gary Boudreau. 2021. Deep Learning-Based Forecasting of Cellular Network Utilization at Millisecond Resolutions. In *ICC 2021 - IEEE International Conference on Communications*. IEEE, Montreal, QC, Canada, 1–6. https://doi.org/10.1109/ICC42927.2021.9500858

[59] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. (2023). https://openreview.net/forum?id=Jbdc0vTOcol

[60] Navid Nikaein, Mahesh K. Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Christian Bonnet. 2014. OpenAirInterface: A Flexible Platform for 5G Research. *SIGCOMM Comput. Commun. Rev.* 44, 5 (oct 2014), 33–38. https://doi.org/10.1145/2677046.2677053

[61] Nokia and Sennheiser. 2020. Low latency 5G for professional audio transmission. (2020). https://d1p0gxnqcu0lvz.cloudfront.net/documents/Nokia_Low_Latency_5G_for_Professional_Audio_Transmission_White_Paper_EN.pdf

[62] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, J. K. Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xioyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, Antonio Torralba, Bi Song, Anesco Fong, Amit Roy-Chowdhury, and Mita Desai. 2011. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*. IEEE, Colorado Springs, CO, USA, 3153–3160. https://doi.org/10.1109/CVPR.2011.5995586

[63] ONNX. 2017. Open Neural Network Exchange. (2017). https://onnx.ai/ [Last accessed on 18-Sept-2025].

[64] Dennis Overbeck. 2022. TransmissionDataSet. https://github.com/overbeckd/TransmissionDataSet. (2022). [Online; Last accessed: 17-Sept-2025].

[65] Dennis Overbeck, Niklas A. Wagner, Fabian Kurtz, and Christian Wietfeld. 2022. Proactive Resource Management for Predictive 5G Uplink Slicing. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. IEEE, Rio de Janeiro, Brazil, 1000–1005. https://doi.org/10.1109/GLOBECOM48099.2022.10001244

[66] P1 Security. 2024. QCSuper release 2.0.1. https://github.com/P1sec/QCSuper. (2024).

[67] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1376–1411. https://doi.org/10.1109/COMST.2023.3239220

[68] Michele Polese, Niloofar Mohamadi, Salvatore D'Oro, Leonardo Bonati, and Tommaso Melodia. 2026. Beyond Connectivity: An Open Architecture for AI-RAN Convergence in 6G. *IEEE Communications Magazine* (2026), 1–6. https://doi.org/10.1109/MCOM.001.2500438

[69] Petar Popovski, Kasper Fløe Trillingsgaard, Osvaldo Simeone, and Giuseppe Durisi. 2018. 5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View. *IEEE Access* 6 (2018), 55765–55779. https://doi.org/10.1109/ACCESS.2018.2872781

[70] Darijo Raca, Jason J. Quinlan, Ahmed H. Zahran, and Cormac J. Sreenan. 2018. Beyond throughput: a 4G LTE dataset with channel and context metrics. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 460–465. https://doi.org/10.1145/3204949.3208123

[71] Aditya R. Raikwar, Rahul R. Sadawarte, Rishikesh G. More, Rutuja S. Gunjal, Parikshit N. Mahalle, and Poonam N. Railkar. 2017. Long-Term and Short-Term Traffic Forecasting Using Holt-Winters Method: A Comparability Approach with Comparable Data in Multiple Seasons. *Int. J. Synth. Emot.* 8, 2 (July 2017), 38–50. https://doi.org/10.4018/IJSE.2017070103

[72] Federica Rinaldi, Helka-Liina Maattanen, Johan Torsner, Sara Pizzi, Sergey Andreev, Antonio Iera, Yevgeni Koucheryavy, and Giuseppe Araniti. 2020. Non-Terrestrial Networks in 5G & Beyond: A Survey. *IEEE Access* 8 (2020), 165178–165200. https://doi.org/10.1109/ACCESS.2020.3022981

[73] S. W. Roberts. 1959. Control Chart Tests Based on Geometric Moving Averages. *Technometrics* 1, 3 (1959), 239–250. https://doi.org/10.1080/00401706.2000.10485986

[74] Flavien Ronteix-Jacquet, Xavier Lagrange, Isabelle Hamchaoui, and Alexandre Ferrieux. 2022. Rethinking Buffer Status Estimation to Improve Radio

Resource Utilization in Cellular Networks. In *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*. IEEE, Helsinki, Finland, 1–5. https://doi.org/10.1109/VTC2022-Spring54318.2022.9860632
[75] Guto Leoni Santos, Pierangelo Rosati, Theo Lynn, Judith Kelner, Djamel Sadok, and Patricia Takako Endo. 2022. Predicting short-term mobile Internet traffic from Internet activity using recurrent neural networks. *International Journal of Network Management* 32, 3 (2022), e2191. https://doi.org/10.1002/nem.2191
[76] Vincenzo Sciancalepore, Xavier Costa-Perez, and Albert Banchs. 2019. RL-NSB: Reinforcement Learning-Based 5G Network Slice Broker. *IEEE/ACM Transactions on Networking* 27, 4 (2019), 1543–1557. https://doi.org/10.1109/TNET.2019.2924471
[77] Vincenzo Sciancalepore, Konstantinos Samdanis, Xavier Costa-Perez, Dario Bega, Marco Gramaglia, and Albert Banchs. 2017. Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. IEEE, Atlanta, GA, USA, 1–9. https://doi.org/10.1109/INFOCOM.2017.8057230
[78] Software Radio Systems. 2022. srsRAN Project. https://github.com/srsran/srsRAN_Project. (2022). [Online; Last accessed: 31-Dec-2024].
[79] Beatriz Soret, Preben Mogensen, Klaus I. Pedersen, and Mari Carmen Aguayo-Torres. 2014. Fundamental Tradeoffs Among Reliability, Latency and Throughput in Cellular Networks. In *2014 IEEE Globecom Workshops (GC Wkshps)*. IEEE, Austin, TX, USA, 1391–1396. https://doi.org/10.1109/GLOCOMW.2014.7063628
[80] Starlink. 2024. *Improving Starlink's Latency*. Technical Report. Starlink Services, LLC. https://www.starlink.com/public-files/StarlinkLatency.pdf
[81] Schyler C. Sun and Weisi Guo. 2021. Forecasting Wireless Demand with Extreme Values using Feature Embedding in Gaussian Processes. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE, Helsinki, Finland, 1–6. https://doi.org/10.1109/VTC2021-Spring51267.2021.9449040
[82] S. Swedha and E. S. Gopi. 2021. LSTM Network for Hotspot Prediction in Traffic Density of Cellular Network. In *Machine Learning, Deep Learning and Computational Intelligence for Wireless Communication*, E. S. Gopi (Ed.). Springer Singapore, Singapore, 35–47. https://doi.org/10.1007/978-981-16-0289-4_3
[83] Zhaowei Tan, Jinghao Zhao, Yuanjie Li, Yifei Xu, and Songwu Lu. 2021. Device-Based LTE Latency Reduction at the Application Layer. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, Virtual, 471–486. https://www.usenix.org/conference/nsdi21/presentation/tan
[84] Wim Taymans, Steve Baker, and Andy Wingo. 2018. GStreamer Application Development 1.10.1. (2018).
[85] Thanuskanth Thangavadivel, Gopalasingham Aravinthan, Van-Quan Pham, Ahan Kak, Huu-Trung Thieu, and Nakjung Choi. 2024. TinyRIC-ML: A Lightweight Real Time ML Platform for O-RAN. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 1584–1586. https://doi.org/10.1145/3636534.3697426
[86] The Helm Authors. 2015. Helm: The Kubernetes Package Manager. https://helm.sh/. (2015). [Online; Last accessed: 25-Mar-2025].
[87] The Kubernetes Authors. 2014. Kubernetes: Production-Grade Container Orchestration. https://kubernetes.io/. (2014). [Online; Last accessed: 25-Mar-2025].
[88] Timescale Inc. 2017. TimescaleDB: An open-source time-series database powered by PostgreSQL. https://www.timescale.com/. (2017). [Online; Last accessed: 25-Mar-2025].
[89] Maximilian Toller, Bernhard C. Geiger, and Roman Kern. 2020. A Formally Robust Time Series Distance Metric. (2020). https://arxiv.org/abs/2008.07865
[90] Quang Thanh Tran, Li Hao, and Quang Khai Trinh. 2020. A comprehensive research on exponential smoothing methods in modeling and forecasting cellular traffic. *Concurrency and Computation: Practice and Experience* 32, 23 (2020), e5602. https://doi.org/10.1002/cpe.5602 e5602 CPE-19-0571.R2.
[91] Luca Turchet, Claudia Rinaldi, Carlo Centofanti, Luca Vignati, and Cristina Rottondi. 2024. 5G-Enabled Internet of Musical Things Architectures for Remote Immersive Musical Practices. *IEEE Open Journal of the Communications Society* 5 (2024), 4691–4709. https://doi.org/10.1109/OJCOMS.2024.3407708
[92] Mikko Uitto and Antti Heikkinen. 2021. Evaluation of Live Video Streaming Performance for Low Latency Use Cases in 5G. In *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, Porto, Portugal, 431–436. https://doi.org/10.1109/EuCNC/6GSummit51104.2021.9482605
[93] Valve Corporation. 2007. Team Fortress 2. https://www.teamfortress.com/. (2007). [Online; Last accessed: 01-Sept-2025].
[94] Valve Corporation. 2023. Counter-Strike 2. https://www.counter-strike.net/. (2023). [Online; Last accessed: 01-Sept-2025].
[95] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., Long Beach, California, USA. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
[96] Mojca Volk and Janez Sterle. 2021. 5G Experimentation for Public Safety: Technologies, Facilities and Use Cases. *IEEE Access* 9 (2021), 41184–41217. https://doi.org/10.1109/ACCESS.2021.3064405
[97] O-RAN WG1. 2024. *O-RAN Architecture Description (O-RAN.WG1.OAD-R003-v08.00)*. Technical Specification TS 103 982. 3GPP, Sophia Antipolis, France. https://www.etsi.org/deliver/etsi_ts/103900_103999/103982/08.00.00_60/ts_103982v080000p.pdf Version: 8.0.0.
[98] WhatsApp. 2018. WhatsApp Video Calling. https://www.whatsapp.com/. (2018). [Online; Last accessed: 01-Sept-2025].
[99] Robin Wiebusch, Niklas A. Wagner, Dennis Overbeck, Fabian Kurtz, and Christian Wietfeld. 2023. Towards Open 6G: Experimental O-RAN Framework for Predictive Uplink Slicing. In *ICC 2023 - IEEE International Conference on Communications*. IEEE, Rome, Italy, 4834–4839. https://doi.org/10.1109/ICC45041.2023.10279730
[100] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. (2023). arXiv:cs.LG/2210.02186 https://arxiv.org/abs/2210.02186
[101] Huiwei Xia, Xin Wei, Yun Gao, and Haibing Lv. 2019. Traffic Prediction Based on Ensemble Machine Learning Strategies with Bagging and LightGBM. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, Shanghai, China, 1–6. https://doi.org/10.1109/ICCW.2019.8757058
[102] Shah Zeb, Muhammad Ahmad Rathore, Aamir Mahmood, Syed Ali Hassan, JongWon Kim, and Mikael Gidlund. 2021. Edge Intelligence in Softwarized 6G: Deep Learning-enabled Network Traffic Predictions. In *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE, Madrid, Spain, 1–6. https://doi.org/10.1109/GCWkshps52748.2021.9682131
[103] ZeroMQ Community. 2007. ZeroMQ: The Intelligent Transport Layer. https://zeromq.org/. (2007). [Online; Last accessed: 25-Mar-2025].
[104] Shanjun Zhan, Lisu Yu, Zhen Wang, Yichen Du, Yan Yu, Qinghua Cao, Shuping Dang, and Zahid Khan. 2021. Cell Traffic Prediction Based on Convolutional Neural Network for Software-Defined Ultra-Dense Visible Light Communication Networks. *Security and Communication Networks* 2021, 1 (2021), 9223965. https://doi.org/10.1155/2021/9223965
[105] Qianqian Zhang, Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Merouane Debbah. 2018. Machine Learning for Predictive On-Demand Deployment of UAVs for Wireless Communications. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Abu Dhabi, UAE, 1–6. https://doi.org/10.1109/GLOCOM.2018.8647209
[106] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. (2023). https://openreview.net/forum?id=vSVLM2j9eie
[107] Hui Zhou, Yansha Deng, Luca Feltrin, and Andreas Höglund. 2022. Analyzing Novel Grant-Based and Grant-Free Access Schemes for Small Data Transmission. *IEEE Transactions on Communications* 70, 4 (2022), 2805–2819. https://doi.org/10.1109/TCOMM.2022.3150787
[108] Yuchao Zhu and Shaowei Wang. 2021. Joint Traffic Prediction and Base Station Sleeping for Energy Saving in Cellular Networks. In *ICC 2021 - IEEE International Conference on Communications*. IEEE, Montreal, QC, Canada, 1–6. https://doi.org/10.1109/ICC42927.2021.9500442
[109] Zoom Video Communications. 2013. Zoom Video Conferencing. https://zoom.us/. (2013). [Online; Last accessed: 01-Sept-2025].
[110] Frederik Jonathan Zumegen, Ish Kumar Jain, and Dinesh Bharadia. 2024. BeamArmor: Seamless Anti-Jamming in 5G Cellular Networks with MIMO Null-steering. In *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications (HOTMOBILE '24)*. Association for Computing Machinery, New York, NY, USA, 121–126. https://doi.org/10.1145/3638550.3641138

APPENDIX

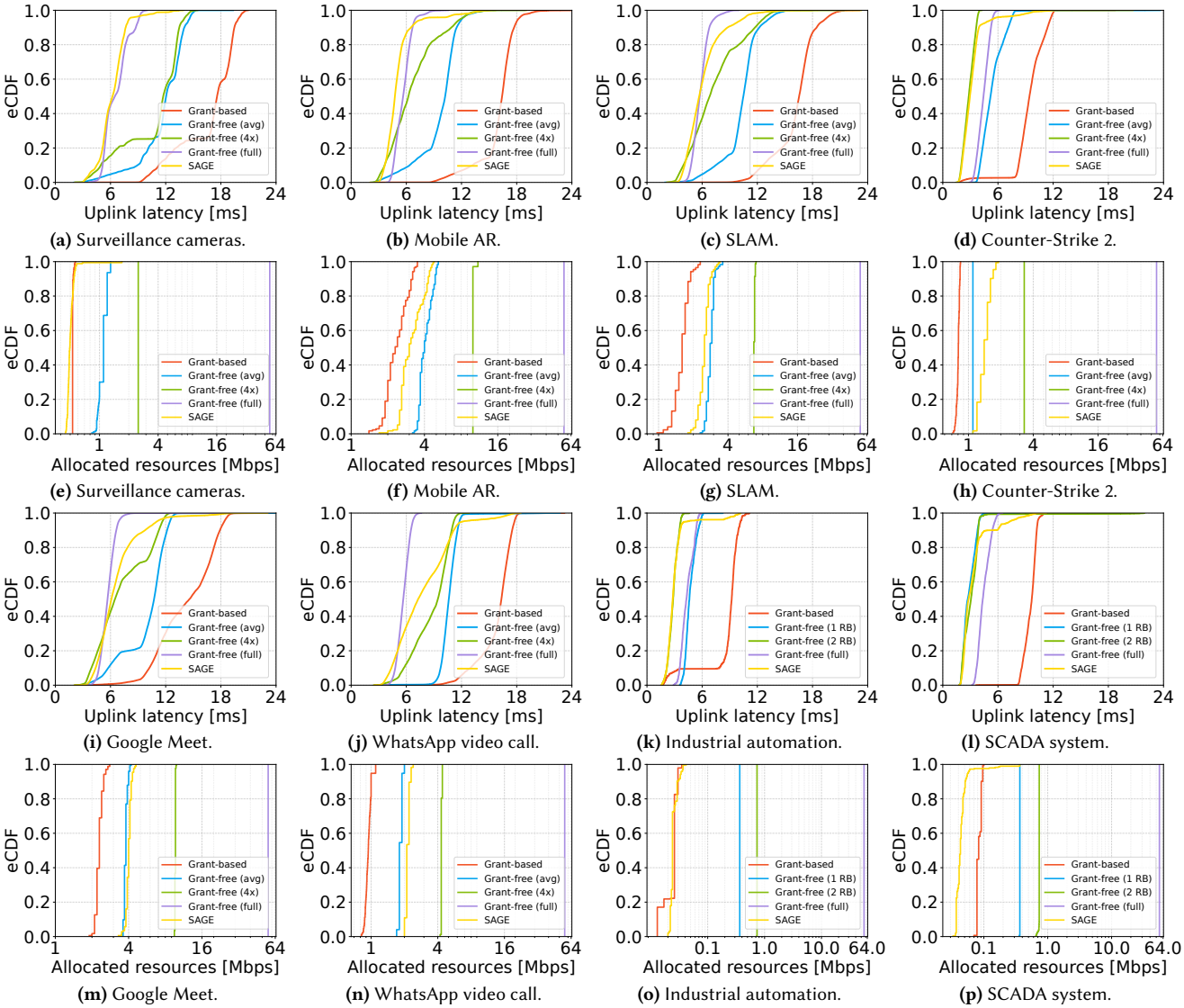
Appendices are supporting material that has not been peer-reviewed.

A Additional Modeling and Analysis

A1 Analytical Model

To begin with, we define three key latency components:

(1) *Base latency* L_{base} is the user-plane uplink latency of transmitting a small packet (e.g., an ICMP packet). It covers processing delays in the UE and gNB protocol stacks, as well as the first latency segment illustrated in Fig. 1.
(2) *Signaling latency* L_{signal} corresponds to the latency segments shown in Fig. 1: the time from a UE sending a UL scheduling signal (SR or BSR) to receiving the corresponding grant.



(a) Surveillance cameras. **(b)** Mobile AR. **(c)** SLAM. **(d)** Counter-Strike 2.

(e) Surveillance cameras. **(f)** Mobile AR. **(g)** SLAM. **(h)** Counter-Strike 2.

(i) Google Meet. **(j)** WhatsApp video call. **(k)** Industrial automation. **(l)** SCADA system.

(m) Google Meet. **(n)** WhatsApp video call. **(o)** Industrial automation. **(p)** SCADA system.

Figure 24: eCDFs of uplink latency and allocated resources for all 8 applications in Fig. 11.

(3) *Transmission latency* L_{trans} is the average interval between two consecutive UL transmissions, which is mainly governed by frame structures and TDD configurations.

The *base latency* can be directly measured using one-way latency measurement tools, whereas *signaling* and *transmission latency* can be derived from the system configuration [26, 54].

Initial grant transmissions. Consider a traffic train of B bytes. The gNB allocates the UE an initial grant of G bytes. Under grant-based access, this initial grant is fixed at $G = G_{\text{SR}}$ (the SR grant). Under grant-free access, $G = G_{\text{gtd}} \in \mathcal{P} \triangleq \{p, 2p, \ldots, P\}$, where G_{gtd} is the guaranteed grant size, p is the payload size corresponding to one RB and P is the maximum payload size that can be transmitted in a single UL slot, both measured in bytes. The initial latency is computed by

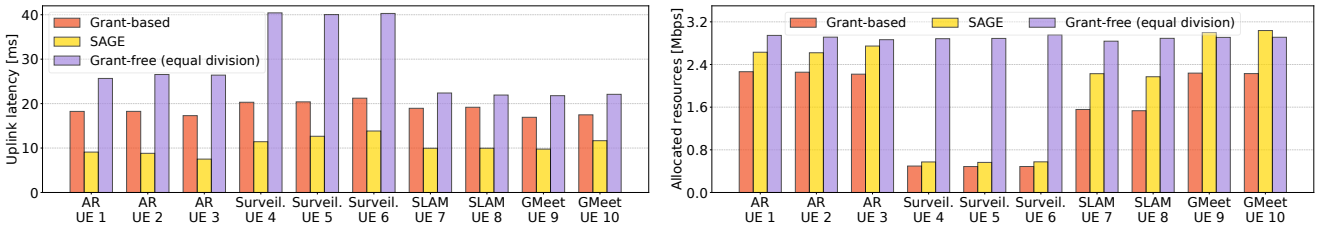
$$L_{\text{init}} = \begin{cases} L_{\text{base}}, & \text{for grant-based access,} \\ L_{\text{base}} - L_{\text{signal}}, & \text{for grant-free access,} \end{cases}$$

where grant-free access bypasses the *signaling latency* L_{signal} as no SR is required. During the second latency segment in Fig. 1 (i.e., the time gap between the first initial grant and the first BSR grant), the gNB can issue additional initial grants whenever UL opportunities arise, which the UE can use to transmit data [54]. Hence, the total number of initial grants is given by

$$N_{\text{init}} = \left\lfloor \frac{L_{\text{signal}}}{L_{\text{trans}}} \right\rfloor.$$

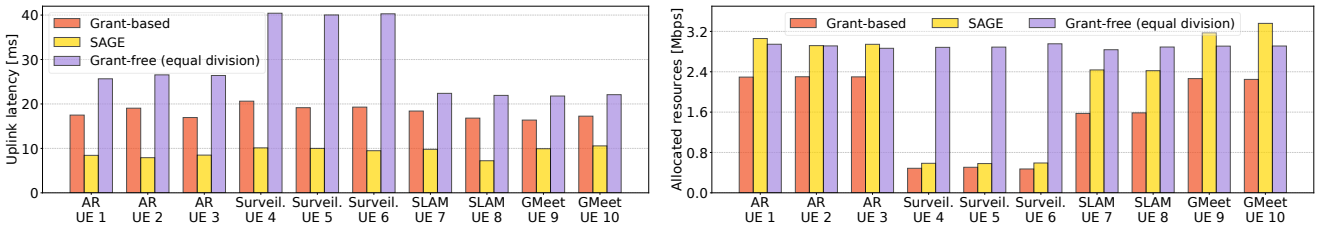
The floor function guarantees that only full UL opportunities are counted. The remaining bytes after these initial grants are

$$B_{\text{remain}} = \max\left(0, B - N_{\text{init}} G\right),$$



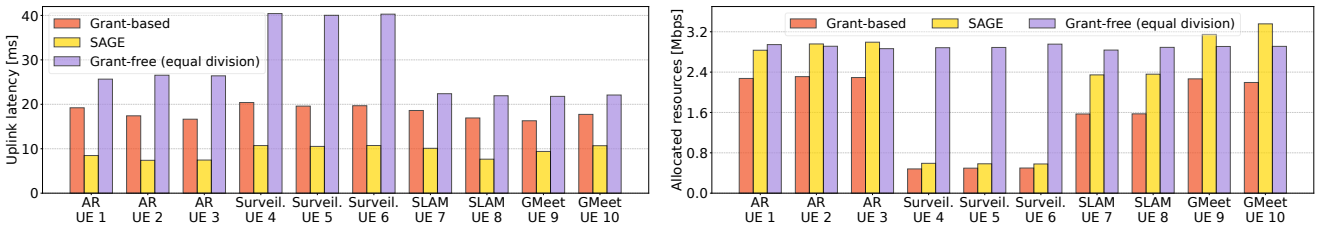
(a) Uplink latency. **(b)** Allocated resources.

Figure 25: Comparison of 10 concurrent UEs running 4 applications with the RR scheduler, where a new application arrives every 1 s.



(a) Uplink latency. **(b)** Allocated resources.

Figure 26: Comparison of 10 concurrent UEs running 4 applications with the PF scheduler, where a new application arrives every 20 s.



(a) Uplink latency. **(b)** Allocated resources.

Figure 27: Comparison of 10 concurrent UEs running 4 applications with the RR scheduler, where a new application arrives every 20 s.

which must be scheduled via subsequent BSR-based allocations.

BSR grant transmissions. If $B_{\text{remain}} > 0$, the remaining bytes are transmitted using UL slots scheduled based on BSR grants. Each such UL slot can carry at most P bytes. The remaining payload is transmitted in full UL slots followed by a final UL slot carrying the remaining bytes. Formally, the number of full UL slots before the final slot is

$$N_{\text{full}} = \max\left(0, \left\lceil \frac{B_{\text{remain}}}{P} \right\rceil - 1\right),$$

and the remaining bytes carried by the final slot are

$$B_{\text{final}} = B_{\text{remain}} - N_{\text{full}} P.$$

Per-slot latency accumulation. The B bytes in the traffic train are not transmitted all at once. Instead, they are transmitted in chunks, each chunk corresponding to the payload carried in a particular UL slot. These chunks experience different latencies because they complete at different times. Let c_i denote the size (in bytes) of chunk i transmitted in the i-th UL slot, and let F_i denote the completion time of chunk i (i.e., the latency experienced by the bytes in chunk i). We partition the B bytes into three chunk categories, aligned with the scheduling phases.

(1) *Initial-grant chunks*: During the initial-grant phase, the UE uses up to N_{init} initial grants. We have

$$c_i = \max\Big(0,\ \min\big(G,\ B - (i-1)G\big)\Big),$$
$$F_i = L_{\text{init}} + (i-1)L_{\text{trans}} + \lambda G,$$

for $1 \le i \le N_{\text{init}}$. The term λG captures the linear growth of processing time with the UL payload size.[2]

(2) *BSR-grant full-slot chunks*: For the following N_{full} full BSR-grant slots, each chunk carries P bytes. We have

$$c_i = P,$$
$$F_i = L_{\text{init}} + (i-1)L_{\text{trans}} + \lambda P,$$

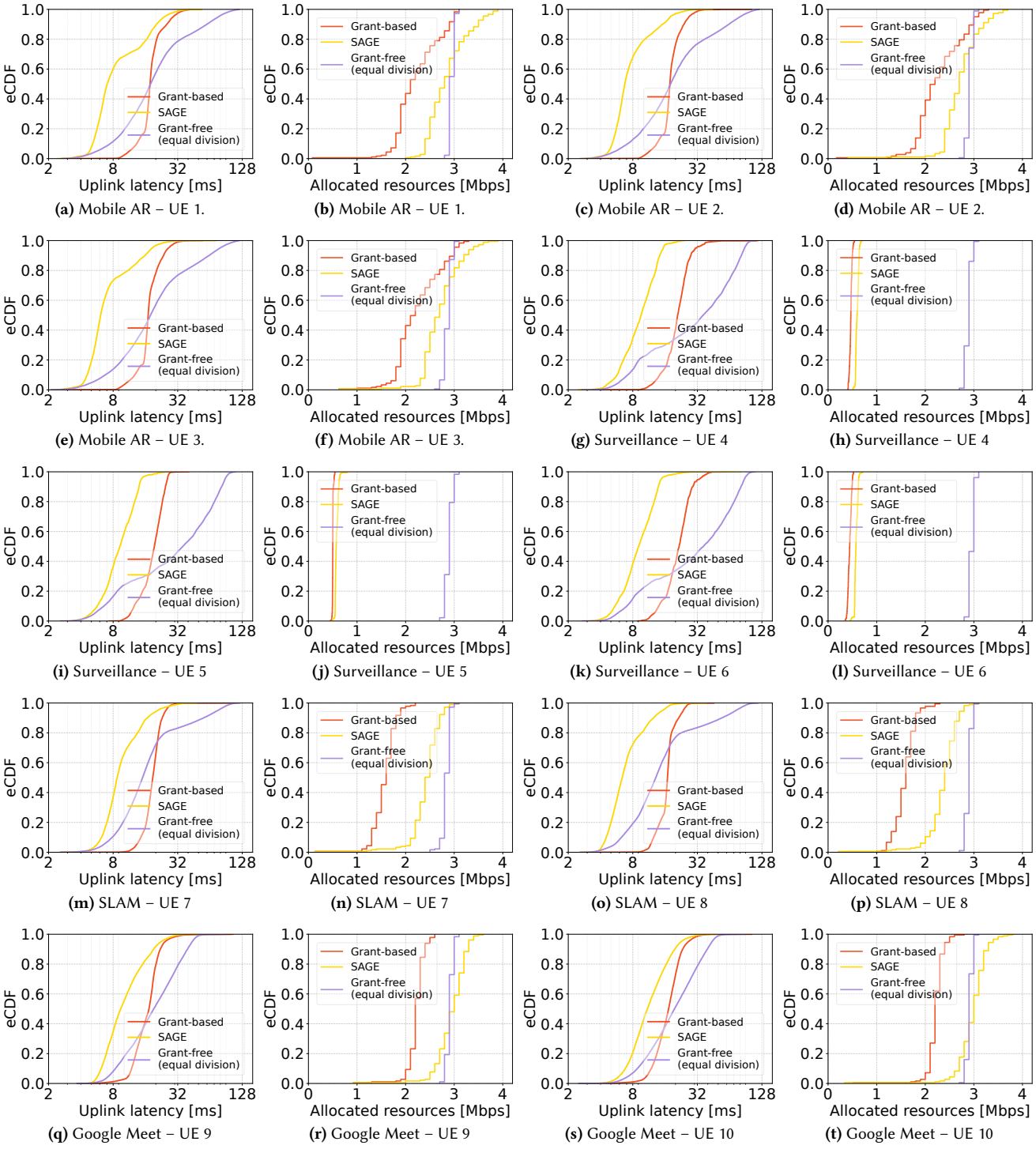
for $N_{\text{init}} + 1 \le i \le N_{\text{init}} + N_{\text{full}}$.

(3) *BSR-grant final-slot chunk*: For the final BSR-grant slot, we have

$$c_i = B_{\text{final}},$$
$$F_i = L_{\text{init}} + (i-1)L_{\text{trans}} + \begin{cases} \lambda G, & \text{if grant-free \& } B_{\text{final}} \le G, \\ \lambda B_{\text{final}}, & \text{otherwise,} \end{cases}$$

for $i = N_{\text{init}} + N_{\text{full}} + 1$.

[2]The coefficient λ can be estimated empirically by measuring L_{base} across different packet sizes and applying linear regression.



(a) Mobile AR – UE 1. **(b)** Mobile AR – UE 1. **(c)** Mobile AR – UE 2. **(d)** Mobile AR – UE 2.

(e) Mobile AR – UE 3. **(f)** Mobile AR – UE 3. **(g)** Surveillance – UE 4 **(h)** Surveillance – UE 4

(i) Surveillance – UE 5 **(j)** Surveillance – UE 5 **(k)** Surveillance – UE 6 **(l)** Surveillance – UE 6

(m) SLAM – UE 7 **(n)** SLAM – UE 7 **(o)** SLAM – UE 8 **(p)** SLAM – UE 8

(q) Google Meet – UE 9 **(r)** Google Meet – UE 9 **(s)** Google Meet – UE 10 **(t)** Google Meet – UE 10

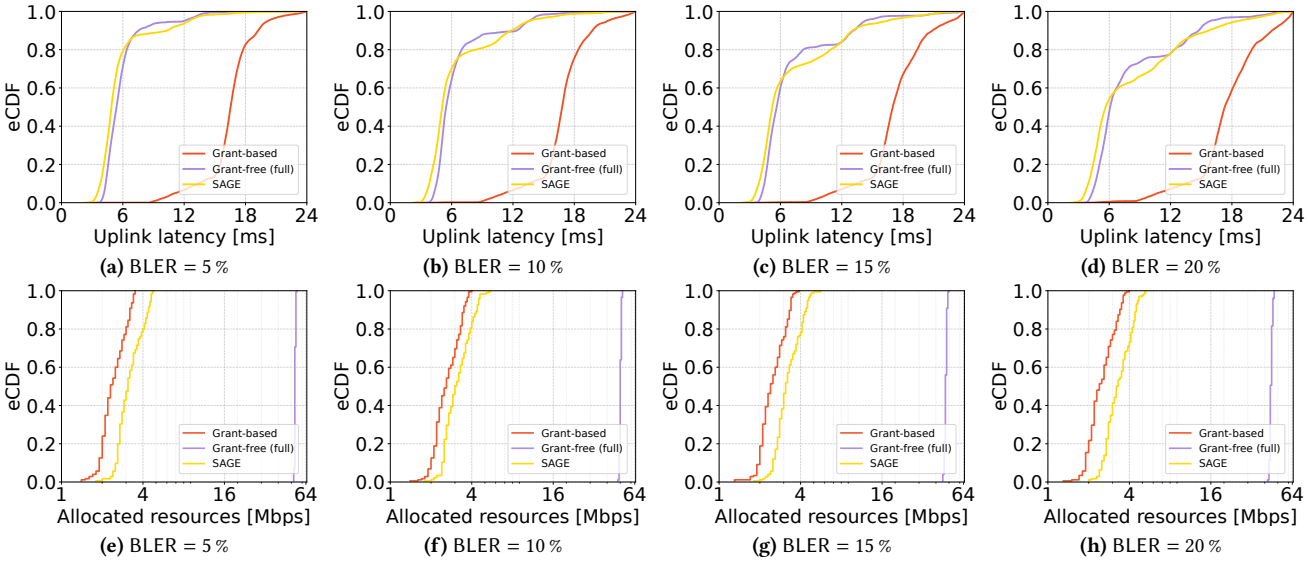
Figure 28: eCDFs of uplink latency and allocated resources for all 10 UEs in Fig. 14.

Uplink latency. Aggregating per-chunk completion times yields the per-byte uplink latency

$$L = \frac{1}{B} \sum_{i=1}^{N_{\text{init}}+N_{\text{full}}+1} c_i F_i.$$

Total allocated resources. The total allocated resources (in bytes) are given by

$$R = N_{\text{init}} G + N_{\text{full}} P + \begin{cases} G, & \text{if grant-free \& } B_{\text{final}} \le G, \\ B_{\text{final}}, & \text{otherwise.} \end{cases}$$



(a) BLER = 5 % **(b)** BLER = 10 % **(c)** BLER = 15 % **(d)** BLER = 20 %

(e) BLER = 5 % **(f)** BLER = 10 % **(g)** BLER = 15 % **(h)** BLER = 20 %

Figure 29: eCDFs of uplink latency and allocated resources for Mobile AR under different BLER values (5, 10, 15 and 20 %).

UL slot usage. The total number of used UL slots is

$$N_{\text{UL}} = N_{\text{init}} + N_{\text{full}} + \mathbf{1}_{\{B_{\text{final}}>0\}}.$$

This analytical model characterizes the latency–resource tradeoff as a function of the guaranteed grant size and is implemented in the near-RT RIC. Given a collected traffic-train time series $\{(\tau_m, B_m)\}_{m=1}^{M}$ from a hard-to-predict application, the near-RT RIC uses the model for each traffic train to compute its uplink latency and total allocated resources. The near-RT RIC further aggregates the results over the entire time series to compute the average uplink latency $L_{\text{app}}(G_{\text{gtd}})$ and average allocated resources $R_{\text{app}}(G_{\text{gtd}})$, both expressed as functions of the guaranteed grant size G_{gtd}.

Knee point detection. To determine a suitable grant size, we apply a knee point detection method to the trade-off curve defined by $\{(R_{\text{app}}(G_{\text{gtd}}), L_{\text{app}}(G_{\text{gtd}}))\}_{G_{\text{gtd}}\in\mathcal{P}}$. The knee point corresponds to the point of diminishing returns, where further increases in allocated resources yield only marginal uplink latency improvements. We first normalize both axes to $[0, 1]$:

$$R'_{\text{app}}(G_{\text{gtd}}) = \frac{R_{\text{app}}(G_{\text{gtd}}) - \min_{G_{\text{gtd}}\in\mathcal{P}} R_{\text{app}}(G_{\text{gtd}})}{\max_{G_{\text{gtd}}\in\mathcal{P}} R_{\text{app}}(G_{\text{gtd}}) - \min_{G_{\text{gtd}}\in\mathcal{P}} R_{\text{app}}(G_{\text{gtd}})},$$

$$L'_{\text{app}}(G_{\text{gtd}}) = \frac{L_{\text{app}}(G_{\text{gtd}}) - \min_{G_{\text{gtd}}\in\mathcal{P}} L_{\text{app}}(G_{\text{gtd}})}{\max_{G_{\text{gtd}}\in\mathcal{P}} L_{\text{app}}(G_{\text{gtd}}) - \min_{G_{\text{gtd}}\in\mathcal{P}} L_{\text{app}}(G_{\text{gtd}})}.$$

We then define the line connecting the two extreme points,

$$x_{\min} = \left(\min_{G_{\text{gtd}}\in\mathcal{P}} R'_{\text{app}}(G_{\text{gtd}}),\ \min_{G_{\text{gtd}}\in\mathcal{P}} L'_{\text{app}}(G_{\text{gtd}})\right),$$

$$x_{\max} = \left(\max_{G_{\text{gtd}}\in\mathcal{P}} R'_{\text{app}}(G_{\text{gtd}}),\ \max_{G_{\text{gtd}}\in\mathcal{P}} L'_{\text{app}}(G_{\text{gtd}})\right).$$

For each point $x(G_{\text{gtd}}) = (R'_{\text{app}}(G_{\text{gtd}}), L'_{\text{app}}(G_{\text{gtd}}))$, its perpendicular distance to this line is

$$d(G_{\text{gtd}}) = \frac{\left|(x_{\max} - x_{\min}) \times (x_{\min} - x(G_{\text{gtd}}))\right|}{\|x_{\max} - x_{\min}\|},$$

where $\times$ represents the 2D cross product. The knee point is then identified as $G^{\star}_{\text{gtd}} = \arg\max_{G_{\text{gtd}}\in\mathcal{P}} d(G_{\text{gtd}})$. The near-RT RIC then sends this grant size $G^{\star}_{\text{gtd}}$ to the gNB, which uses it to configure resource allocations for UEs operating in *grant-free mode*.

A2 Scalability and Fairness Analysis

This section quantifies the additional UL resource allocation introduced by SAGE and its impact on airtime utilization and fairness. Since SAGE operates in two main phases, bootstrapping and steady state, we analyze each separately.

Bootstrapping. As mentioned in § 4, bootstrapping can operate in either *grant-free mode* or *grant-based mode*, depending on slice-specific policies and Service Level Agreements (SLAs). Under *grant-based mode*, bootstrapping does not introduce additional allocated resources beyond those used by a traditional 5G network. Under *grant-free mode*, the gNB assigns each bootstrapping UE a fixed grant of R_{res} RBs in every UL slot, regardless of whether the UE has data to transmit. The UE can request extra resources through BSRs when its demand exceeds the reserved grant. To analyze the *reservation overhead in airtime utilization*, we use Little's Law [50], which relates the average number of items in a queue to their arrival rate and time in the system. So, the average number of UEs in the bootstrapping phase can be given by:

$$N_{\text{UE, boot}} = \lambda T_{\text{boot}}, \tag{3}$$

where $N_{\text{UE, boot}}$ is the average number of UEs in the bootstrapping phase, λ is the average arrival rate of low-latency UEs, and T_{boot} is the average time that a UE remains in the bootstrapping phase. By Eq. (3), the reservation overhead in airtime utilization, denoted by O_{boot}, is:

$$O_{\text{boot}} = \frac{N_{\text{UE, boot}} R_{\text{res}}}{R_{\text{total}}} = \frac{\lambda T_{\text{boot}} R_{\text{res}}}{R_{\text{total}}},$$

where R_{total} is the total number of RBs available in an UL slot. Fig. 7a shows O_{boot} for different values of λ and R_{res}, assuming $T_{\text{boot}} = 10$ s and $R_{\text{total}} = 273$ RBs, corresponding to the number of RBs in a 100 MHz bandwidth configuration. We vary λ from 0.01 to 2 UEs per second and R_{res} from 2 to 30 RBs (up to 20 % of the UL resource grid capacity).

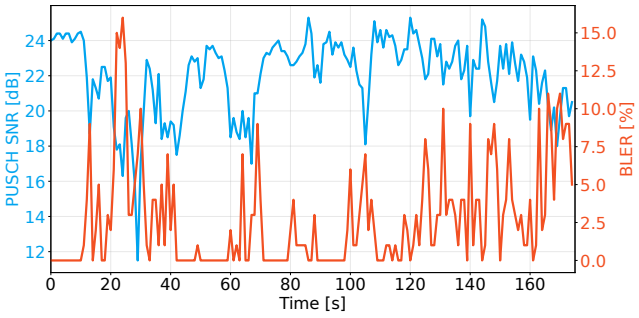


Figure 30: Uplink (PUSCH) Signal-to-Noise Ratio (SNR) and Block Error Rate (BLER) over time for the varying channel experiment.

A UE's bootstrap reservation temporarily reduces the capacity available to other UEs. The magnitude of this effect can be directly controlled by R_{res}. For any additional resources requested by BSRs, UEs in the bootstrapping phase compete with other UEs through the gNB's existing MAC schedulers. Thus, SAGE does not modify the MAC schedulers' policies on unreserved resources. An operator can set $R_{\text{res}} = 0$, that is, use grant-based bootstrapping, to eliminate the reservation overhead. For slices with low-latency requirements, an operator may instead choose a non-zero R_{res} to reduce latency during the bootstrapping phase.

Steady state. During the steady-state phase, SAGE allocates resources according to predicted traffic trains. When a prediction overestimates the actual demand or uses a prediction window to tolerate timing uncertainty, some proactively allocated resources may remain unused. We capture this effect with an empirical over-provisioning factor, represented by γ, which quantifies the additional resources allocated due to prediction errors. It is defined as the ratio of the average allocated resources achieved by SAGE to those achieved by grant-based access. The over-provisioning overhead in airtime utilization, denoted by O_{steady}, is:

$$O_{\text{steady}} = \frac{N_{\text{UE, steady}}(\gamma - 1)R_{\text{GB, UE}}}{R_{\text{total}}},$$

where $N_{\text{UE, steady}}$ is the average number of UEs in the steady-state phase and $R_{\text{GB, UE}}$ is the average allocated resources per UE under grant-based access. Across the applications presented in Fig. 11, we observe an average resource-inflation factor of 1.59×. Accordingly, Fig. 7b shows O_{steady} for different values of $N_{\text{UE, steady}}$ and $R_{\text{GB, UE}}$, assuming $\gamma = 1.59$ and $R_{\text{total}} = 273$ RBs. We vary $N_{\text{UE, steady}}$ from 1 to 100 and $R_{\text{GB, UE}}$ from 2 to 10 RBs, corresponding to 1 to 6 Mbit/s. The additional allocation during the steady-state phase represents the cost of achieving lower latency via proactive grants.

During the steady-state phase, SAGE submits proactive grants to the same underlying MAC scheduler and therefore preserves the operator-selected scheduling policy (e.g., round-robin, proportional fair, etc.) and inter-slice priorities. Nevertheless, proactive scheduling may introduce different amounts of additional resource allocation across UEs due to differences in their traffic patterns and prediction accuracy. We compute Jain's fairness index [40] over the per-UE resource-inflation factors as follows:

$$J = \frac{\left(\sum_{n=1}^{N_{\text{UE}}} \gamma_n\right)^2}{N_{\text{UE}} \sum_{n=1}^{N_{\text{UE}}} \gamma_n^2}, \qquad \gamma_n = \frac{R_{\text{SAGE}, n}}{R_{\text{GB}, n}}, \tag{4}$$

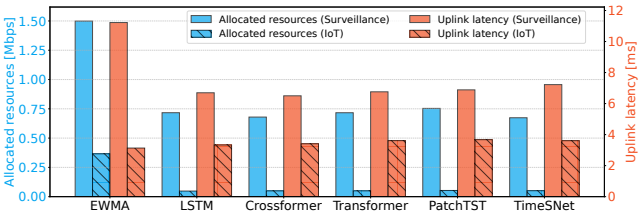


Figure 31: Comparison across different predictors.

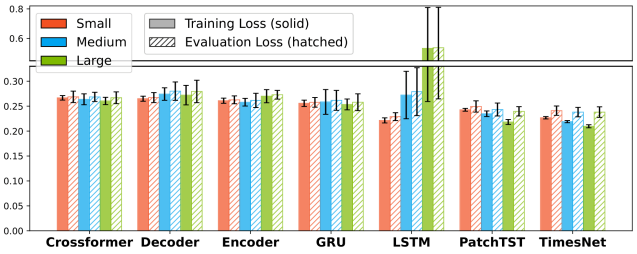


Figure 32: Training and evaluation loss across different architectures and model sizes.

where $R_{\text{SAGE}, n}$ and $R_{\text{GB}, n}$ denote the average allocated resources to UE n by SAGE and grant-based access. Thus, γ_n captures the relative resource-allocation increase introduced by predictive scheduling for UE n. A fairness index of 1 indicates that SAGE allocates proactive resources fairly across all UEs, introducing the same relative over-provisioning overhead for each UE. In contrast, a value close to $1/N_{\text{UE}}$ indicates highly unfair allocation, where the over-provisioning overhead is concentrated on only a small subset of UEs. The multiple-UE evaluation in §7D and App. B3 shows that Jain's fairness index remains above 0.99 across all cases. These results indicate that SAGE allocates proactive resources fairly across all UEs and applications.

B Additional Results

B1 Model Hyperparameters

Tab. 2 reports the hyperparameters of the LSTM models trained for all 8 applications in Fig. 11. As discussed in §5.1, SAGE supports profile-specific lightweight model design. Applications with simpler traffic patterns, such as surveillance cameras and SCADA systems, use shorter input sequences and smaller hidden sizes, resulting in compact models with fewer parameters. In contrast, applications exhibiting higher traffic variability or longer temporal dependencies, such as mobile AR and industrial automation, use longer input sequences and larger hidden sizes to capture more complex dynamics, at the cost of larger models. This design balances prediction accuracy and inference efficiency.

B2 eCDF for Each Application

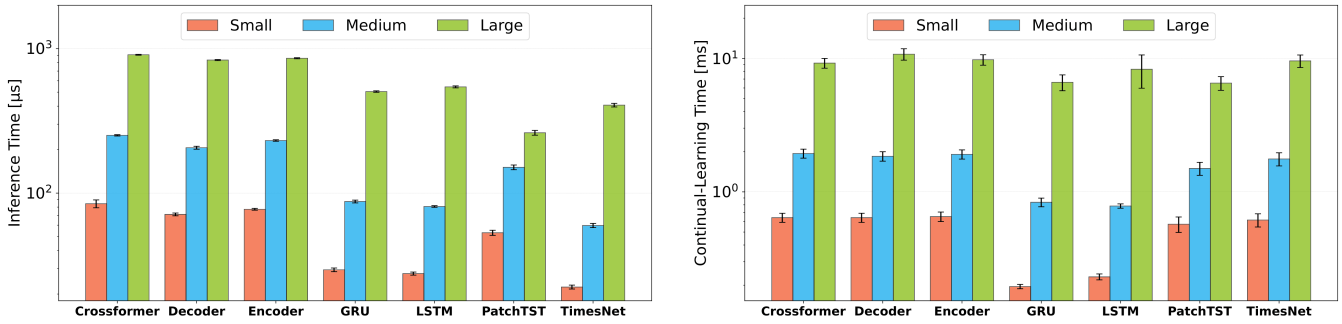
Fig. 24 shows the eCDFs of uplink latency and allocated resources for all 8 applications in Fig. 11.

B3 Multiple UEs and Applications.

Across Fig. 25 to 27, SAGE maintains low latency across heterogeneous applications and schedulers. Compared with *grant-based*,

Application	Input length	Input size	Hidden size	Number of layers	Output size	Total parameters
Surveillance cameras	16	2	16	2	2	3490
Mobile AR	64	2	64	2	2	50818
SLAM	64	2	64	2	2	50818
Counter-Strike 2	32	2	32	2	2	13122
Google Meet 2	32	2	32	2	2	13122
WhatsApp	64	2	64	2	2	50818
Industrial Automation	64	2	64	2	2	50818
SCADA System	16	2	16	2	2	3490

Table 2: Hyperparameters of the LSTM models trained for all 8 applications in Fig. 11.



(a) Inference execution time

(b) Continual learning execution time

Figure 33: Inference and continual learning execution times across different architectures and model sizes.

Sage reduces latency by 1.50–2.41× and increases allocated resources by 1.15–1.55×. Compared with *grant-free*, Sage reduces latency by 1.90–4.25×, while using between 1.15× more and 5.14× fewer allocated resources.

B4 Channel and Retransmissions

Fig. 29 shows the eCDFs of uplink latency and allocated resources for *grant-based*, *grant-free (full)*, and Sage under 5, 10, 15 and 20 % BLER. Fig. 30 shows the uplink (PUSCH) Signal-to-Noise Ratio (SNR) and Block Error Rate (BLER) over time for the varying channel experiment.

C Additional Micro-benchmarks

Evaluation across predictors. Predictors act as plug-ins in our system. To demonstrate this modularity and benchmark predictor performance, we evaluate classical and modern time-series forecasting models: EWMA [53, 73], LSTM [36], Crossformer [106], Transformer [95], PatchTST [59], and TimesNet [100]. Fig. 31 shows that modern AI-based predictors achieve similar performance but consistently outperform EWMA, which both over-allocates resources and incurs higher latency. These results highlight the importance of AI-driven control and validate Sage's plug-in design.

Training and evaluation losses. We evaluated the training and evaluation losses of commonly used architectures for time-series forecasting. For each architecture, we consider three model sizes: small, medium, and large. The architectures and their numbers of trainable parameters are reported in Tab. 3.

Architecture	Small	Medium	Large
Crossformer	25.6K	76.4K	400.2K
Decoder	34.1K	102.1K	534.3K
Encoder	25.6K	76.4K	400.2K
GRU	9.9K	35.2K	187.9K
LSTM	13.1K	49.9K	250.5K
PatchTST	25.8K	76.6K	401.1K
TimesNet	20.8K	58.0K	304.8K

Table 3: Number of trainable parameters.

Fig. 32 reports training and evaluation losses across different architectures and model sizes. The models are trained on video-streaming traffic with a batch size of 256 and evaluated using 5-fold cross-validation. Overall, most models achieve comparable performance, although some more complex architectures yield slightly lower losses. Increasing model size, however, does not necessarily improve performance. In several cases, larger models incur higher losses. In particular, the large LSTM model fails to converge.

Execution times. Fig. 33 shows the execution times of inference and one-epoch continual learning (see §5.2) across different architectures and model sizes. We conduct this microbenchmark on an Intel NUC 12 with an Intel Core i7 processor. For small models, inference completes within 100 µs, and continual learning completes within 1 ms. As expected, inference and continual learning times increase with model size. Larger models may not improve accuracy and can incur higher execution costs, increasing the risk of missed prediction deadlines and making them impractical for real-time systems.