Accepted for presentation in: 2023 IEEE International Conference on Communications (ICC)

Towards Open 6G: Experimental O-RAN Framework for Predictive Uplink Slicing

Robin Wiebusch, Niklas A. Wagner, Dennis Overbeck, Fabian Kurtz, Christian Wietfeld
Communication Networks Institute, TU Dortmund University, Otto-Hahn-Strasse 6, 44227 Dortmund
Email: {robin.wiebusch, niklas.wagner, dennis.overbeck, fabian.kurtz, christian.wietfeld}@tu-dortmund.de

***Abstract*—Traditional cellular Radio Access Networks (RANs) are associated with significant costs and low agility due to proprietary hard- and software resulting in vendor lock-ins. Open RAN promises to change this by harnessing open source and Commercial of-the-Shelf (COTS) solutions. Hence, the Open Radio Access Network (O-RAN) Alliance, a consortium of partners from industry and research, aims to identify and close gaps in 3rd Generation Partnership Project (3GPP) specifications. It defines a software-centric RAN architecture with open interfaces to increase interoperability, strengthen innovation and lower market entry barriers towards future 6G infrastructures. A core concept in this context is the near-Real-Time RAN Intelligent Controller (RIC), a virtual platform for hosting so-called xApps. These software-based network functions provide functionalities such as monitoring or network slicing. Using proactive resource management, slicing has the potential of enabling concurrent service profiles such as Ultra-Reliable Low Latency Communication (URLLC) and Enhanced Mobile Broadband (eMBB), while rising spectral efficiency and lowering latency. Thus, this work introduces an O-RAN-based framework for predictive uplink slicing. An xApp is presented, harnessing deep learning to dynamically reconfigure RAN scheduling via the RIC's E2 interface. The evaluation is performed on the challenging example of URLLC traffic from the Smart Grid domain via an experimental laboratory setup. O-RAN introduces additional interfaces, yet the framework performs about on par with proprietary solutions with latencies down to 5 ms.**

I. Introduction

Cellular networks such as Long Term Evolution (LTE) are traditionally specified by the 3rd Generation Partnership Project (3GPP) [1]. They include proprietary interfaces as well as hard- and software components, particularly in the Radio Access Network (RAN), often preventing operators from mixing equipment of different vendors. Such a lock-in to specific manufacturers potentially increases costs while closed and proprietary interfaces, e.g. in the front-haul, may limit innovation by raising barriers to market entry. Open RAN is a concept which aims to address this by open sourcing aspects currently not fully covered by 3GPP. Here, the international Open Radio Access Network (O-RAN) Alliance [2], formed by operators, incumbent and novel vendors as well as research institutes, has emerged as the most prominent project. They focus on open source software deployed on Commercial of-the-Shelf (COTS) hardware to lower market barriers, reduce costs and accelerate innovation cycles towards 6G [3]. As depicted in Fig. 1, O-RAN core components are the non- and near-Real-Time (RT) RAN Intelligent Controller (RIC). Acting as the RAN's operating system, they serve to host virtualized network functions known as r/xApps. Examples include basic tasks such as monitoring as well as novel services such as Machine Learning (ML)-driven approaches to e.g. slicing. This promises faster deployment of innovative solutions regardless of vendor, shortening the cycle from research to RAN integration. Closely associated with these efforts is the O-RAN Software Community (OSC), developing specification compliant open software. This work introduces an O-RAN compliant solution for predictive uplink slicing, fully based on open source software. By partitioning uplink resources into multiple virtually dedicated networks, i.e. slices, we aim to meet requirements as diverse as Ultra-Reliable Low Latency Communication (URLLC) and Enhanced Mobile Broadband (eMBB). For evaluation, critical Smart Grid communications are used. We employ a Long Short-Term Memory (LSTM) ML model to predict uplink resource requirements, removing delays incurred by Scheduling Request Occasions (SROs). Hence, the slicing xApp running on the near-RT RIC needs to provide accurate predictions within milliseconds.

The work is structured as follows: First, Sec. II provides an overview of related work. Sec. III then describes the novel predictive resource assignment and underlying ML concepts. Next, Sec. IV introduces the employed O-RAN framework integrating ML-driven slicing functionality as an xApp. The experimental setup and results are given in Sec. V, highlighting achieved gains in efficiency and performance. Finally, a conclusion and an outlook are given in Sec. VI.

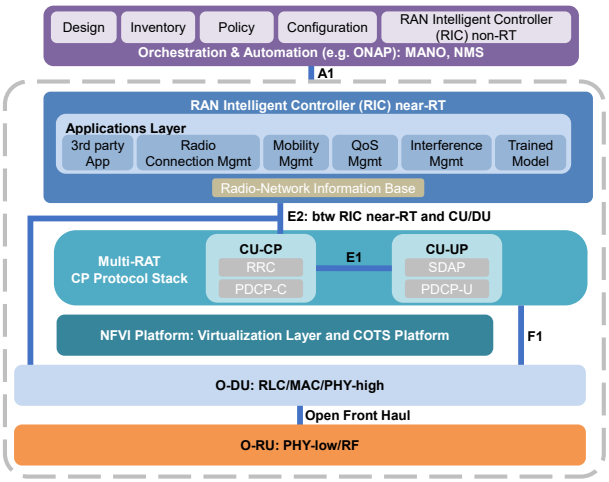

Fig. 1: O-RAN architecture [2] with non- and near-RT RIC hosting network functions such as this work's predictive uplink slicing xApp

II. Related Work

As the open RAN concept is relatively new, related works mostly concentrate on implementing the framework and evaluating its performance. Pioneers in this area are the authors of [4], who embedded the E2 interface into srsRAN and provided a policy-driven dynamic downlink slicing scheduler as an xApp. In [5] the authors focus on the acquisition and collection of data to be used with ML models for xApps. The underlying scenario comprises multiple base stations and user devices within a testbed, evaluating scalability of the open RAN concept. An ML model is trained to perform RAN control via RIC control messages to update base station configurations at runtime. Promising results for maximizing transmit rate while minimizing packet queue times are achieved. The authors of [6] provide a setup comprising three User Equipments (UEs) as well as a 5G Non-Standalone (NSA) base station. This combination of the O-RAN framework with srsRAN also includes a customized xApp for monitoring Key Performance Indicators (KPIs). Also, the deployment of a downlink slicing xApp is demonstrated by using the E2 interface to allocate resources in the RAN. Building on the modified monitoring xApp, control and report messages are used to configure a Next Generation Node B (gNB) in different slices through asynchronous E2 control messages. RAN parameters are adapted by the control service based on metrics collected by the report function. The E2 agent is a key contribution, handling the signaling between near-RT RIC and eNB/gNB. These works provide open source resources, which enabled the research within this paper.

Research of ML-based resource management is driven by the evolution towards 6G. Several frameworks are proposed to increase spectral efficiency and promote self-organization, e.g. in [7]. Network slicing is a particularly relevant use case to intelligently provide resources, according to custom Service Level Agreements (SLAs) or Quality of Service (QoS) parameters. Several works research network slicing by combining software-defined RANs with ML. In [8], the authors formulate and solve dynamic resource re-assignment among slices with the Lyapunov technique. Two different slices are taken into consideration with heterogeneous requirements concentrating on delay and throughput. The authors of [9] present a Deep Q-Network (DQN)-based framework for dynamic slicing and scheduling on the base station level. A self-optimization scheme is provided enabling adjustments to current network and traffic conditions at runtime. Yan et al. [10] propose an intelligent Resource Scheduling Strategy for RAN slicing, which inherits a collaborative learning framework. Here, an LSTM model is used for large timescales, as well as the Asynchronous Advantage Actor Critic algorithm for small timescales. Results show improvements over other prediction algorithms in terms of resource utilization. The authors of [11] examine network slicing when using a Deep Reinforcement Learning agent, correlating industrial production activity and network utilization. Thereby, the spectral efficiency of resource allocation is improved and overall network performance is increased. Within this work, we aim to achieve similar improvements in spectral efficiency by utilizing the concept of Proactive Grants (PGs), specified by the 3GPP for 5G, and prediction techniques to allocate resources precisely on the uplink channel, mitigating delays.

III. Machine Learning driven Proactive Network Slicing

This section describes the developed approach to proactive uplink resource management based on ML.

A. Proactive Resource Management using Machine Learning

In a conventional reactive scheduling scheme, when a UE intends to transmit data it requests physical resources in the Physical Uplink Control Channel (PUCCH) during SROs. The physical resources are then granted by the Evolved Node B (eNB) via Downlink Control Information (DCI) on the Physical Downlink Control Channel (PDCCH). As depicted on the left side of Fig. 2, the waiting time for SROs is a main contributor to the end-to-end delay in the uplink communication between UE and eNB. In contrast, in [12] a proactive scheduling scheme is developed. The proactive approach mitigates the problem of large scheduling delays by allocating PGs via the PDCCH in advance, making it obsolete for the UE to request resources. Instead, as depicted on the right side of Fig. 2, the UE can send its data in the moment the packet is generated. However, to proactively allocate resources without causing spectral inefficiencies, a highly accurate, real-time capable model for predicting timing and payload of the UEs' data is required. As resources are scheduled every ms, predictions must be available at this interval, too. Therefore, the prediction process as well as scheduler and LSTM model interfaces should minimize delays. Otherwise large prediction horizons would result in low accuracies and thus low spectral efficiency. In the previous work, a Representational State Transfer (REST) interface is deployed for communication between scheduler and ML model.

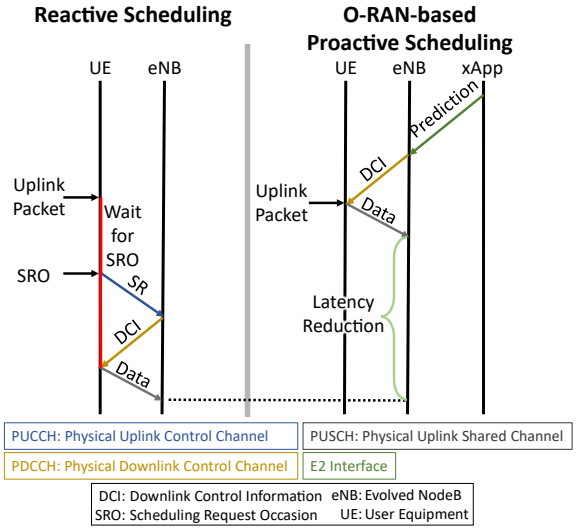

Fig. 2: Reactive vs. proactive schemes for scheduling RAN uplink resources as studied in this work

B. Deep Learning for Real-Time Payload Prediction

As LSTM models are considered suitable for time series prediction [13], we employ them in this work to proactively manage physical resources by anticipating uplink data traffic. To rule out the prediction model as a main source of differences in test results, the model designed and trained in [12] is redeployed. For training the model, Tensorflow with a Keras backend is utilized [14]. This dataset is split into $\frac{2}{3}$ training and $\frac{1}{6}$ validation subsets, whereas $\frac{1}{6}$ of the data is used for traffic generation by the UE during runtime. Hyperparameter tuning using the hyperband tuner algorithm in combiation with cross-validation is performed to achieve a robust model with a prediction accuracy of 92.45 % on the validation dataset. Table I shows the resulting parameters of the LSTM. Apart from prediction accuracy, it is important that the model produces several multi-step predictions simultaneously and in real-time. This is caused by the xApp requiring predictions within pre-defined intervals, which can be faster than they become available. Tensorflow provides a scalable REST server for interacting with the underlying ML model. Crucially, the REST server handles requests simultaneously by creating multiple model instances. Hence, several partly overlapping predictions are made concurrently, with the model designed to generate 100 values in advance based on 1000 measured payloads. Our scheduler utilizes the most recent prediction for each Transmission Time Interval (TTI).

Therefore, the basis for predictions performed by the model is the arrival process of the individual, possibly fragmented, payloads at the eNB. Since reactively scheduled packets have non-controllable additional delays introduced by the Scheduling Request (SR)-based scheduling as shown in Fig. 2, the time of arrival at the UE's transmission buffer is not known by the eNB. Thus, if uncompensated, these payloads get predicted only with a similar delay as they arrived, resulting in a diverging behavior. Hence, a convergence algorithm is developed to improve the timing accuracy. It monitors whether the last predicted payload arrived via a proactive grant and gradually shifts the predictions towards earlier grants until this offset is overcompensated, resulting in one reactive grant. Afterwards, the convergence behaviour improves.

IV. O-RAN-based Real-Time Network Control

The following section describes the evolution of the previously discussed closed-loop uplink slicing towards open RAN interfaces based on a near-RT RIC-based xApp.

TABLE I: Training Parameter Settings

Learning Rate	10^{-4}
Layer Structure	LSTM (vanilla, 64 Units) Dense Layer (Activation: linear)
Batch Size	4
Epochs	16
Loss	MSE
Optimizer	ADAM

A. O-RAN Architecture as a Basis for the xApp

A major innovation included in the open RAN architecture is the specification and integration of two RICs. The non-RT RIC is embedded in the Orchestration & Automation Framework. Its purpose is to monitor and control the RAN for non time-critical functionalities, such as the online training of deep learning traffic models. Connected to the non-RT RIC via the A1 interface, the near-RT RIC on the other hand serves time-critical, low latency RAN control. Since the framework presented in this work manages highly dynamic, time critical network slicing resource allocation every ms, the near-RT RIC (simply called RIC in the following) is of special interest. Importantly, the RIC itself does not contain control functionalities. Instead, it serves as a platform for the deployment of scalable xApps, where control logics can be implemented in a microservice-like fashion. Furthermore, multiple xApps with different functionalities can be deployed simultaneously. Messages between the xApp and the RAN units, called E2 nodes, are managed by the RIC Message Router (RMR), which is responsible for routing messages over the E2 interface between the xApps and E2 nodes. The E2 Application Protocol (E2AP) defines message procedures and these four service types for the E2 interface [2]:

- *Report*: Based on negotiated trigger events or on predefined timers, the E2 node sends data to the xApp.
- *Insert*: Notifies the xApp of an event in the E2 node.
- *Policy*: Defined in a subscription message, policies specify E2 node behaviour during trigger events.
- *Control*: Using a control request, the xApp can set parameters or behaviour in the E2 node. The control request can be sent as a response for an insert message or due to internal xApp processes. Requests have to be followed by a control acknowledge from the E2 node.

This work uses control requests, as they can be triggered autonomously by the xApp. In this case, the xApp sends a control request over the E2 interface whenever the LSTM model generates a new packet size prediction. Whereas the E2AP defines general procedures, service-specific communication is defined in an E2 Service Model (E2SM) individually for each service. The E2SM defines data and message types using Abstract Syntax Notation One (ASN.1) and is known by both xApp and E2 node. Both parties can compile the E2SM into implementation-specific data types. To serve our xApp written in C, for instance, the Service Model is compiled into C-specific data types using the asn1c compiler [15], regardless of the programming language of the E2 Node.

B. Designing an xApp for Proactive Network Slicing

Fig. 3 depicts the process of evolving the previously discussed uplink slicing scheme towards open RAN platforms and interfaces. In [12], the slicing control logic as well as the connectivity to the Tensorflow REST server is integrated directly into an srsRAN 21.10 [16] eNB LTE stack and evaluated via Software-Defined Radios (SDRs).

In this work, the slicing function is bundled in a newly developed Slice-Aware Machine Learning-based Ultra-Reliable

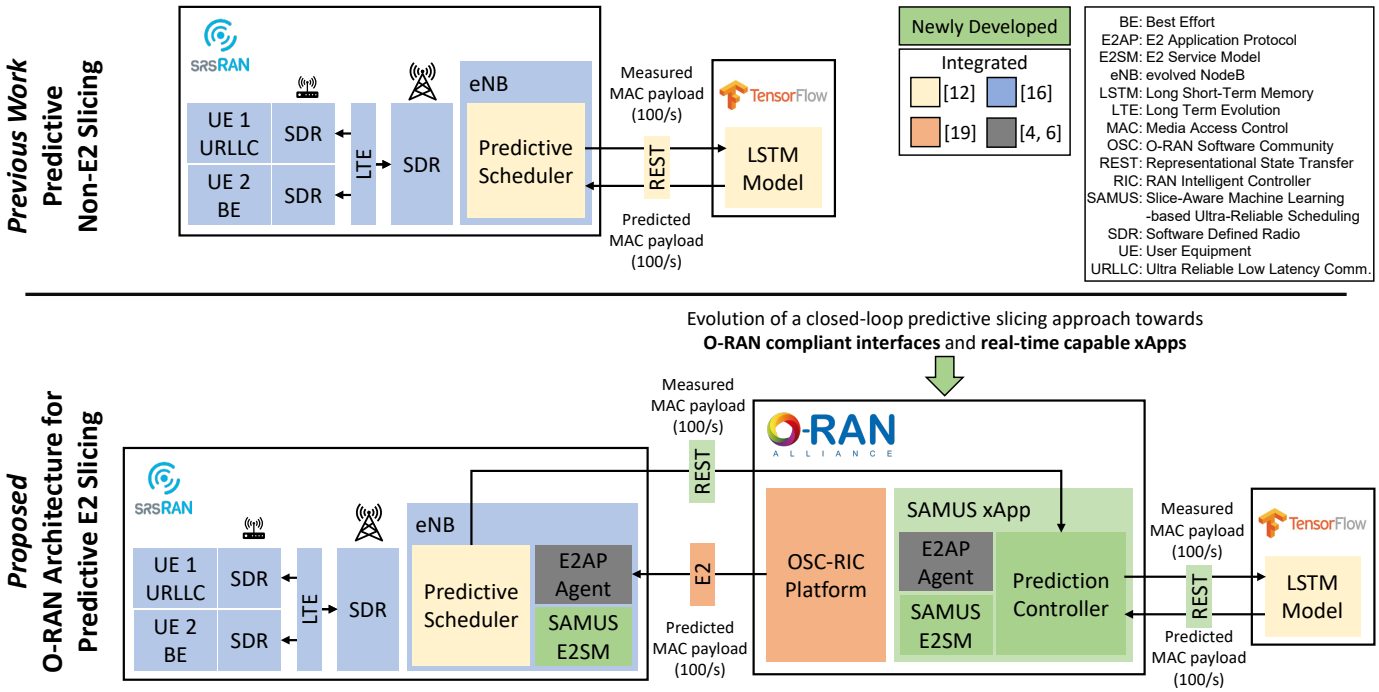

Fig. 3: Depiction of the proposed O-RAN compliant framework for ML-driven uplink slicing and its entirely open source-based elements and interfaces in contrast to a more traditional non-O-RAN setup presented in previous work

Scheduling (SAMUS) xApp which is integrated in an O-RAN compliant RIC platform. The concept of SAMUS was first introduced by [17] as a simulative framework for predictive uplink slicing. The developed xApp and the integration into the O-RAN and srsRAN-based framework works as follows: A near-RT RIC provided by the OSC is deployed as a platform for the proposed xApp. For scalability, RIC components are bundled as Docker containers and deployed using Kubernetes v1.16. The OSC provides software for the deployment of the Kubernetes cluster [18]. Therein, RIC components based on the O-RAN Cherry release are employed [19].

A core component of the SAMUS xApp is the prediction controller. It is equipped with a REST server, which receives a series of 1,000 consecutively measured Medium Access Control (MAC) layer payloads sent by the scheduler for each UE within 10 ms intervals. Payloads are identified by the corresponding UE's International Mobile Subscriber Identity (IMSI) as well as the TTI number of the first payload value. As the IMSI is not natively known by the eNB, it retrieves this from communications between UE and Evolved Packet Core (EPC) during attachment. Also, a mapping between IMSI and Temporary Mobile Subscriber Identity (TMSI) is established. The controller identifies a UE's slice based on the IMSI. Our xApp offers three modes: static proactive slicing, predictive proactive slicing and a combination of both. Using static slicing, the xApp returns future payload values based on preconfigured settings. This guarantees UEs a constant bandwidth for each TTI. For predictive slicing, historic values are utilized to derive forecasts. To do so, the prediction controller sends measured payloads to the Tensorflow model's REST server. There, predictions for 100 future payloads are generated and sent back via the Hypertext Transfer Protocol (HTTP) response. These predictions are multiplied by a configurable overprovisioning factor to increase UE throughput and reduce scheduling delay. Also, payloads are limited to a maximum of 700 Byte per TTI to ensure a minimum throughput for other UEs. Lastly, a combination of static and predictive slicing is implemented by statically defining a constant payload baseline and adding the predicted (and scaled) values on top. The predicted payload series is then encoded into an E2 control request via a newly developed SAMUS E2SM. It consists of the predicted values as well as the IMSI and the first TTI they correspond to. Using the E2AP agent provided by [4], E2 control requests are sent over the RIC's E2 interface to the eNB stack. There they are received and decoded (using the SAMUS E2SM) by an E2 agent developed by [6]. Next, E2 control requests are sent to the scheduler to dynamically assign physical resources as required by the UE's packets. As depicted on the right side of Fig. 2, predictions are utilized to schedule uplink resources in the moment the packet is generated, consequently reducing scheduling delays. Thus, predictions must be available for every TTI. To support RAN control requests in real-time, the E2 interface, which is based on the Stream Control Transmission Protocol (SCTP), is specified to support control message intervals from 20 ms to 1 s. An SCTP packet is only sent after a Selective Acknowledgement (SACK) has been received. The utilized E2 agent sends SACKs with the standard delay of 200 ms, resulting in large control message intervals. Hence, we tweaked the E2 agent to send SACKs immediately, yielding control messages at intervals from 10 to 20 ms.

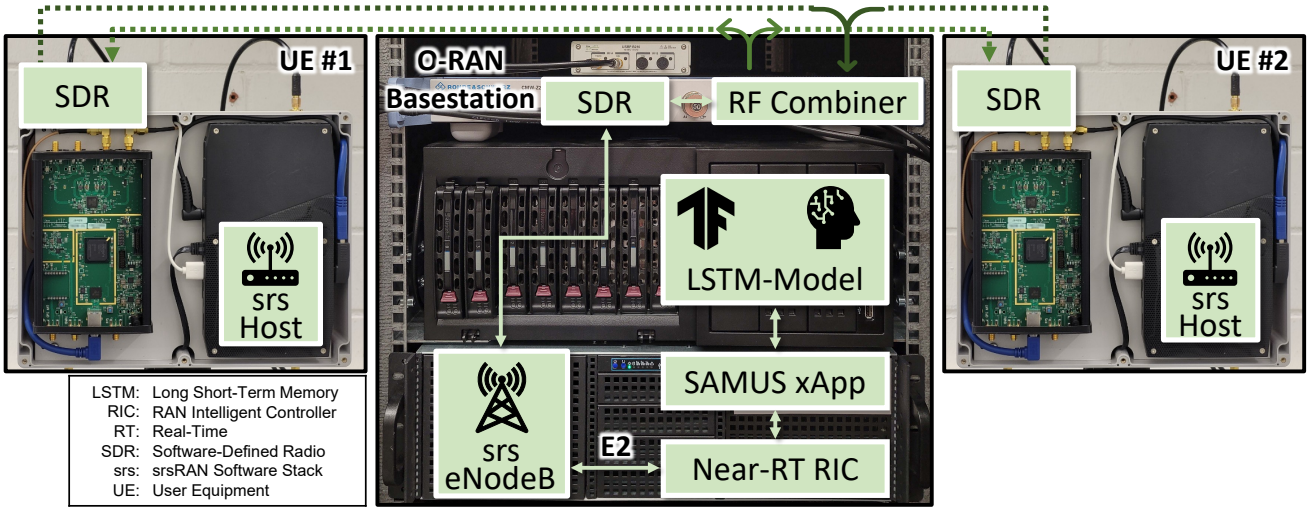

Fig. 4: Experimental setup for evaluating the proposed O-RAN framework and xApp for ML-driven predictive uplink slicing

V. Evaluation Setup and Results

In the following we discuss the evaluation setup and results.

A. Evaluation Scenario and Experimental Setup

The evaluation is performed on a physical test setup as depicted in Fig. 4. EPC, eNB and the O-RAN RIC including xApp are deployed on a shared server (AMD Ryzen 5900X processor, 32 GB RAM, Ubuntu 18.04) to reduce latencies. Two compact computers (Intel i7-6770HQ, 16 GB RAM) host one UE each. They connect to Ettus Research USRP B210 SDRs in a wired setup, enabling reproducible results. An RF combiner connects the SDRs of eNB and UEs. Our ML model is deployed on a seperate server (dual AMD EPYC 7552 48-Core Processors, 512 GB RAM) and creates a series of 100 prediction values within 15 to 35 ms. For evaluation, the following real-world scenario is used: One UE is deployed in a URLLC slice, transmitting critical International Electrotechnical Commission (IEC) 60870 traffic (captured within a realistic Smart Grid laboratory, c.f. [12]). During the experiment, runtime data traffic is generated using Tcpreplay [20]. Data packets are captured on both eNB and URLLC UE using tshark. Devices are synchronized using the Precision Time Protocol (PTP). In a second slice another UE sends up to 10 Mbps of Best Effort (BE) data via iPerf2, effectively trying to utilize all available uplink resources. As no delay requirements are imposed for the BE slice, this UE's resources are always scheduled reactively. Hence, less throughput is achieved than generated, since bandwidth is allocated to the URLLC UE. The setup is parametrized so that all proactive slicing approaches achieve the same throughput in the BE slice. This way it is possible to compare the resulting URLLC delays. For predictive slicing, the xApp scales forecasted values up by a constant overprovisioning factor of 4 to reduce URLLC delay. Using static slicing, a constant value of 50 Byte per TTI is allocated. After mapping to Physical Resource Blocks (PRBs), this results in an allocated bandwidth of roughly 0.5 Mbps. Combining static and predictive scheduling, the static portion is parametrized to result in about 0.25 Mbps allocated bandwidth. To reduce unconditionally allocated resources below the minimum of one PRB per TTI, a PRB is only proactively scheduled e.g. every second TTI to halve the assigned bandwith. Total allocated bandwidth also depends on the predicted payload per TTI. The eNB is configured to a bandwidth of 3 MHz, 15 PRBs and to a constant Modulation and Coding Scheme (MCS) of 23 for the uplink, resulting in a Transport Block Size (TBS) of 7480 bit. This yields a total bandwidth of 7.48 Mbps for user and control plane data. User plane throughput with two UEs as measured with iPerf2 is 5.99 Mbps. srsRAN is used in Frequency Division Duplex (FDD) mode and each evaluation is based on over 10,000 samples, gathered by multiple measurements.

B. Results of the Empirical Evaluation

We compare a reactive, Round Robin (RR) slicing approach with four PG-based methods, namely static slicing, predictive non-E2 and predictive (as is and in combination with static) E2 slicing. For RR the URLLC UE is always granted PRBs with higher priority whenever it requested these resources during the prior SRO. Fig. 5 shows the resulting mean data rates. Using the reactive strategy, the BE UE achieves a mean data rate of 5.4 Mbps, which is higher than the achieved mean data rate of any proactive slicing approach (approx. 4.9 Mbps). This illustrates the nature of proactive resource allocation: Due to prediction errors, resources are seldom allocated exactly to the per-TTI requirements of the URLLC UE, leading to spectral inefficiencies and thus reduced throughput in the BE slice. On the other hand, a RR-based scheduler can tailor the allocated PRBs exactly to requirements. None of the approaches achieve the maximum possible user plane throughput of 5.99 Mbps, though. The reason for this is a scheduling overhead consisting of unallocated PRBs and reserved but unused bandwidth for the URLLC slice to reduce delay peaks. As described in sec. V, the proactive approaches are parametrized to achieve equivalent BE slice throughputs. This allows a fair comparison of end-to-end one-way URLLC delays, as given in Fig. 6.

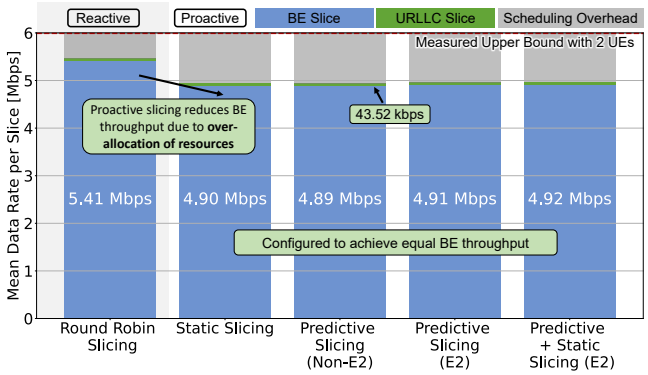

Fig. 5: Observed data rates of reactive, static and predictive proactive O-RAN-based uplink slicing for BE traffic

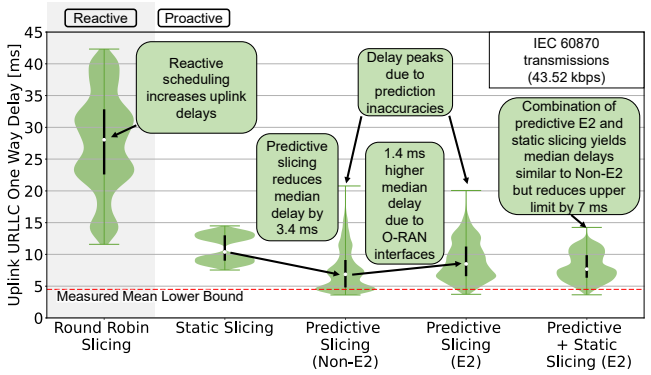

Fig. 6: Measured delays for reactive, static and predictive proactive O-RAN-based uplink slicing for URLLC Smart Grid communications

Delays are defined as the time packets require to reach the Packet Data Network Gateway (P-GW) on eNB side after being sent by the UE application. The mean lower bound is the average delay measured when proactively assigning all resources to the URLLC slice. The same performance could be achieved by deploying a perfectly accurate prediction model.

As expected, the reactive approach has larger uplink delays due to waiting periods for SROs. The median delay as well as the upper and lower bounds are significantly higher relative to proactive approaches. Among the latter, delays of predictive solutions improve upon static slicing. Non-E2 predictive slicing lowers the median delay by 3.4 ms. However, delays also show higher peaks, indicating violations of optimal QoS. This is due to larger prediction errors. Yet, the shape of the violins indicates that these errors occur rarely. Most notable, the predictive E2 approach results in a delay increase of 1.4 ms compared to the non-E2 approach. This is caused by latency of the E2 interface, which is required in O-RAN compliant architectures, as well as overhead introduced by the xApp. Thus, resource management on TTI basis can be performed via the proposed O-RAN compliant framework without strongly impacting performance. Furthermore, a combination of predictive E2 and static slicing significantly lowers peak delays.

VI. Conclusion and Outlook

This work presents a novel, O-RAN-based framework for predictive uplink slicing. A near-RT RIC xApp is combined with a Tensorflow ML model to perform real-time, proactive RAN uplink resource management. This is evaluated and tested using srsRAN radio via a real-world scenario. Performance is shown to be comparable to a setup without O-RAN compliance, although introducing several new interfaces. Hence, uplink delays down to 5 ms are achieved with high spectral efficiency. In future work, the setup will be ported and tested using 5G and 6G radios, further lowering end-to-end delays. Additionally, non-RT RIC features will be harnessed for online ML capabilities.

Acknowledgment

This work has been partly funded by the Federal Ministry of Education and Research (BMBF) via the project *6GEM* under funding reference 16KISK038 and is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) via the project *5Gain* under funding reference 03EI6018C.

References

[1] *3rd Generation Partnership Project (3GPP)*, 2022. [Online]. Available: https://www.3gpp.org/.

[2] *O-RAN ALLIANCE Specifications*, 2022. [Online]. Available: https://orandownloadsweb.azurewebsites.net/specifications.

[3] N. H. Mahmood *et al.*, "Machine Type Communications: Key Drivers and Enablers towards the 6G Era," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, Jun. 2021.

[4] D. Johnson, D. Maas, and J. Van Der Merwe, "NexRAN: Closed-Loop RAN Slicing in POWDER -A Top-to-Bottom Open-Source Open-RAN Use Case," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization*, New Orleans, LA, USA: ACM, 2021, pp. 17–23.

[5] L. Bonati *et al.*, *OpenRAN Gym: An Open Toolbox for Data Collection and Experimentation with AI in O-RAN*, 2022.

[6] P. S. Upadhyaya *et al.*, *Prototyping Next-Generation O-RAN Research Testbeds with SDRs*, 2022. [Online]. Available: https://arxiv.org/abs/2205.13178.

[7] B. Khodapanah *et al.*, "Framework for Slice-Aware Radio Resource Management Utilizing Artificial Neural Networks," *IEEE Access*, vol. 8, pp. 174 972–174 987, 2020.

[8] A. Papa *et al.*, "Optimizing Dynamic RAN Slicing in Programmable 5G Networks," in *IEEE Int. Conf. on Comm. (ICC)*, 2019, pp. 1–7.

[9] B. Casasole *et al.*, "QCell: Self-optimization of Softwarized 5G Networks through Deep Q-learning," in *IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 01–06.

[10] M. Yan *et al.*, "Intelligent Resource Scheduling for 5G Radio Access Network Slicing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7691–7703, 2019.

[11] M. Zambianco, A. Lieto, I. Malanchini, and G. Verticale, "A Learning Approach for Production-Aware 5G Slicing in Private Industrial Networks," in *IEEE Int. Conf. on Comm. (ICC)*, 2022, pp. 1542–1548.

[12] D. Overbeck, N. A. Wagner, F. Kurtz, and C. Wietfeld, "Proactive Resource Management for Predictive 5G Uplink Slicing," in *IEEE Global Communications Conference (GLOBECOM)*, 2022, pp. 1–6.

[13] A. M. Nagib *et al.*, "Deep learning-based forecasting of cellular network utilization at millisecond resolutions," in *IEEE International Conference on Communications (ICC)*, 2021, pp. 1–6.

[14] Martín Abadi *et al.*, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. [Online]. Available: https://www.tensorflow.org/.

[15] *ASN.1 C compiler*, 2019. [Online]. Available: https://gitlab.eurecom.fr/oai/asn1c/-/tree/velichkov_s1ap_plus_option_group.

[16] I. Gomez-Miguelez *et al.*, "srsLTE: an open-source platform for LTE evolution and experimentation," Oct. 2016, pp. 25–32.

[17] C. Bektas, D. Overbeck, and C. Wietfeld, "SAMUS: Slice-Aware Machine Learning-based Ultra-Reliable Scheduling," in *IEEE International Conference on Communications (ICC)*, 2021, pp. 1–6.

[18] *Cherry Release of O-RAN Cluster Deployment Tools*, 2022. [Online]. Available: https://github.com/o-ran-sc/it-dep/tree/cherry.

[19] *Cherry Release of RIC Deployment Tools*, 2022. [Online]. Available: https://github.com/o-ran-sc/ric-plt-ric-dep/tree/cherry.

[20] *Tcpreplay v4.4.2*, 2022. [Online]. Available: https://github.com/appneta/tcpreplay.