5G Radio Access Network Slicing in Massive MIMO Systems for Industrial Applications

Haorui Peng*, Emma Fitzgerald*†, William Tärneberg*, Maria Kihl*
*Department of Electrical and Information Technology, Lund University, Lund, Sweden
†Institute of Telecommunications, Warsaw University of Technology, Warsaw, Poland
{haorui.peng, emma.fitzgerald, william.tarneberg, maria.kihl}@eit.lth.se

***Abstract*—A key enabler for Industry 4.0 is Fifth Generation Wireless Specifications (5G), within which network slicing is a promising technique to ensure customized quality of service for specific end-user groups in industrial scenarios. Massive Multiple Input Multiple Output (MIMO) plays a significant role in 5G but network slicing for massive MIMO has not yet been addressed. In this paper, we propose a network slicing scheme for a 5G Radio Access Network (RAN) with massive MIMO technology. Our simulations show that it is feasible to provide guaranteed performance in terms of high reliability, low latency, and a large number of connections by deploying our proposed scheme at the mMedium Access Control (MAC) layer of a massive MIMO base station.**

***Index Terms*—Network Slicing, 5G, RAN, Massive MIMO, Industrial 4.0**

I. Introduction

Wireless communication for factory automation is receiving growing interest in the context of Industry 4.0. Factory automation requires networks to provide extremely high reliability, high capacity, large throughput, and low latency as well as security and safety compliance to meet the performance requirements of different applications [1]. Today, the most common networking solutions for factory automation systems are wired networks, such as controller area networks, Modbus and industrial Ethernet. However, wires inhibit the mobility of end devices, and installation, management and maintenance of cables come with significant costs. While wireless networks overcome these shortcomings, most wireless standards, especially in unlicensed spectrum, struggle to meet critical requirements as cable networks can. As such, the design of industrial wireless standards is a vibrant research area.

5G, in combination with network slicing, brings a novel alternative solution that overcomes the flaws of most wireless networks [2]. The use cases that 5G covers meet many performance requirements for industrial automation, such as ultra-reliability and low latency [3], while also providing higher mobility, flexibility, and elasticity in the manufacturing system. A Global Mobile Suppliers Association (GSA) white paper [4] argues that network slicing will play a significant role in addressing the performance requirements in vertical industries. As defined in [5], network slicing permits tailoring of the network on the same physical infrastructure according to specific performance requirements from customers. A network slice allows isolated access to only the customers who are subscribed to it. Network slicing covers the industrial communication demand of channel separation even when using the same infrastructure. Studies on network slicing are very diverse, depending on the networking or communication technologies that they focus on, as well as the applications served by the slices. Network slicing is generally implemented by configuring a set of Virtual Network Functions (VNF) and connecting them via virtual networks on a programmable infrastructure, such as is done in [6] and [7].

Some papers have proposed network slicing schemes in the RAN. However these solutions are highly dependent on the chosen RAN architecture. For instance, in [8], a framework is introduced to enforce network slicing in the RAN. The authors implemented a slice resource manager and resource mappers in the framework to perform a two-level scheduling process. [9] presents a radio resource slicing framework based on LTE-A air interfaces, while [10] proposes a slicing plane for RAN considering inter-cell interference and resource grid isolation. To the best of our knowledge, there are no RAN-based network slicing schemes proposed for 5G networks that use massive MIMO technology. Massively many connections and low latency communications are from two different traffic categories that massive MIMO can serve, but few studies have treated these two types of traffic separately and accommodated them on the same infrastructure.

In this paper, we propose a RAN-based network slicing scheme for 5G networks using massive MIMO technology. Our slicing scheme uses a coordinated two-level scheduler and specifically targets industrial scenarios, aiming to provide Quality of Service (QoS) guarantees on par with wires for factory automation. Although a two-level scheduling mechanism was exploited in some slicing solutions such as [8], we address a different type of radio resource abstraction based on massive MIMO and focus on industry use cases. Our results demonstrate that our scheme facilitates wireless communication for factory automation, meeting the QoS requirements of

This work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Maria Kihl and Emma Fitzgerald are also partially supported by the SEC4FACTORY project, funded by the Swedish Foundation for Strategic Research (SSF), and the 5G-PERFECTA Celtic Next project funded by Sweden's Innovation Agency (VINNOVA). Maria Kihl and Emma Fitzgerald are part of the Excellence Center at Linköping-Lund on Information Technology (ELLIIT) strategic area.

low latency and high reliability and enabling massively many connections for the specific scenarios we investigate. We also guarantee radio resource isolation between the slices.

II. Targeted System

In this section, we describe the targeted system that we use to evaluate our proposed network slicing scheme. We focus on a factory automation scenario. An industrial facility has numerous sensors, controllers and actuators, here called industrial units, which are all connected via a 5G network using massive MIMO antenna technology. We assume that the distance between the industrial units is small enough that they can all be covered by the same Base Station (BS); that is we have a single cell environment. The industrial units have different priorities, for instance the control units should have higher priority since their application is more time critical. Thus, we categorise the industrial units and subscribe them to two different network slices on top of our single cell system.

We take as our system setting massive MIMO with Time Division Duplex (TDD) and Orthogonal Frequency-Division Multiplexing (OFDM) modulation [11]. The time-frequency space of a single massive MIMO system can be divided into so-called coherence blocks, where a coherence block is the largest time period during which the channel can be viewed as time-invariant and the channel frequency response is approximately constant. A coherence block is shared by uplink data, downlink data and uplink pilot transmissions. The uplink pilots are used by the BS to estimate each end-user's Channel State Information (CSI), used by the BS for precoding needed to process the input and output data. Thus a pilot is needed for a given end-user to transmit data successfully and we will consider the uplink pilots as the resources that the end-users require before a transmission can start.

Each coherence block can host at most p mutually orthogonal pilot signals, and pilots with the same length yield the same performance. Ideally, the maximum number of end-users a system can serve is $K = p$, giving equal performance to the users. However, in practice, some end-users may be assigned multiple pilots, giving a more accurate CSI estimation. Hence the number of users that can be served can be less than p.

In this paper, we assume that each industrial unit in our factory automation scenario uses either the Ultra-Reliable and Low-Latency Communication (URLLC) or Massive Machine Type Communication (mMTC) traffic types. The actuation units and controllers are distributed and their feedback loops are connected via the mobile network. Sensors are also deployed to measure the states of the actuators. Communications for the control loops are all categorised as URLLC type traffic. Meanwhile, there are also numerous devices sending out occasional monitoring messages about the plant. These monitoring devices are installed and distributed throughout the whole plant and require massively many connections. They are thus categorised as mMTC.

We separate the industrial units into two network slices with different performance requirements and priorities. We define two network slices: 1) a URLLC slice, denoted by S_1, in which the end-users send periodic transmission requests; and 2) an mMTC slice S_2, serving a large number of end-users with aperiodic transmissions. We assume that the total number of connected industrial units exceeds the number of antennas of the massive MIMO system. It is, therefore, impossible to provide a sufficient number of resources (pilots) for each individual unit in each coherence block. Instead, a dynamic scheme is needed in order to meet the QoS requirements.

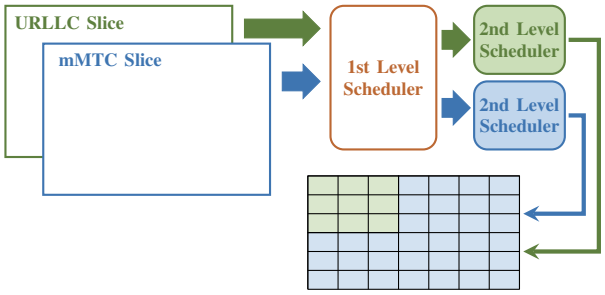

Fig. 1: An example of the two-level MAC scheduler tailoring the time-frequency resource elements of a coherence block for two network slices. The first level scheduler allocate the resources to each slice and each second level scheduler assigns the pilots to the subscribed end-users.

III. Proposed slicing method

In this section, we present our RAN-based network slicing method. The proposed method: 1) enables the coexistence of two traffic classes with different QoS requirements; 2) isolates part of the radio spectrum resources for one high priority slice in order to meet a certain network performance; 3) prevents the end-users being interfered with the users of the second, low-priority, slice; and 4) can be generalized to more than two slices, facilitating a network slicing solution for several traffic classes with different priorities and QoS requirements.

We assume that during one slot time T_{slot}, the channel is time-invariant; hence a slot time is equal to one coherence interval. A device may transmit on both the uplink and downlink if it is granted a pilot signal within a coherence block. We also assume that the resources in the coherence block are sufficient to complete a full packet transmission from each end-user assigned a pilot by the scheduler.

The number of customers served by each slice is K_1 and K_2, respectively, giving that $K = K_1 + K_2$. We consider that each user in S_1 acquires p_1 pilot signals and p_1 may be larger than one in order for the BS to obtain more accurate CSI and provide higher channel reliability. In S_2 we assume that $p_2 = 1$, to serve as many devices as possible for mMTC. During each coherence interval, we consider that there are $\tilde{K}_1$ and $\tilde{K}_2$ end-users with transmission requests. To serve all the devices, the following condition should be satisfied.

$$\tilde{K}_1 p_1 + \tilde{K}_2 p_2 \leq p. \tag{1}$$

Our proposed network slicing method is designed as a two-level MAC scheduler, as illustrated in Fig. 1. The first level of

the MAC scheduler is responsible for inter-slice scheduling. It divides the radio spectrum into two and maps dedicated resources to each slice. The second level enforces intra-slice scheduling and allocates the pilots assigned by the first level to the devices within the slice. The end-users interface only with the slices they are subscribed to and receive pilots from the second level schedulers.

The first level scheduler conducts a simple priority scheduling wherein slice S_1 always takes precedence over slice S_2 to guarantee its QoS requirements. The scheduler always assigns the required number of pilots to slice S_1 until all pilots are in use. If there are any pilots unassigned after this, they will be assigned to slice S_2. Each second level scheduler will then use a scheduling algorithm for intra-slice resource allocation to individual users. In this paper, we will investigate three commonly deployed scheduling algorithms.

First Come First Served (FCFS): The scheduler maintains two queues for all the incoming requests from the two slices and serves them in order of arrival time. The advantage of FCFS is that the reliability of S_1 will always be guaranteed when the traffic does not exceed the highest load that the system can tolerate, and there will not be any resource waste. However, it requires the scheduler to be aware of the full queuing information and would cost extra computation and signalling to maintain and acquire this information.

Round Robin with partial queuing information (RR_Q): In this method, the scheduler does not need to acquire full queuing information. During each coherence interval, it instead iterates through all the devices in each slice and assigns pilots to whichever users are signalling for transmission. In this way the overhead would be lower than FCFS since RR_Q does not assign plots in order of arrival time.

Round Robin without queuing information (RR_NQ): In this method the scheduler does not acquire any queuing information; instead it assigns pilots to each device subscribed to the slice, whether it signals for transmission or not. This yields a static scheduling approach. The scheduler groups a number of slots into a frame for pilot allocation such that a frame time is equal to the deadline of each request in S_1. In this way, we guarantee that every request will be assigned a pilot within its deadline. During a frame, the scheduler iterates through all the devices in S_1 assigning pilots. If there are pilots left during a frame, they will be allocated to S_2. This does not incur any overhead for searching for active devices. However it will result in a large number of wasted pilots and is not applicable to S_2, since S_2 hosts an enormous number of end-users. Accordingly, the other methods are utilized within S_2 when RR_NQ is applied within S_1.

The second level scheduler of each slice can choose a different scheduling algorithm. We will show in the following sections that the chosen scheduling algorithm of one slice has very little impact on the QoS in the other slice.

IV. Simulation

We evaluated our proposed network slicing method using simulations. The complete code of our simulation program, experiments and data will be shared on Github [12] upon publication. The implementation is based on a massive MIMO MAC layer simulator [13]. The simulation program is written in Python and is extensible with any applicable scheduling algorithm on both levels of the scheduler. It also supports custom traffic profiles on each slice by specifying the inter-arrival distribution, deadline, required number of pilots and number of end-users.

TABLE I: Network parameters and variables used in the simulation.

Variable name	Value	Symbol
Simulation length	10000 ms	L
Slot time	0.5 ms	T_{slot}
Available pilot signals per slot	12	p
Mean inter-arrival time in S_1	1 ms or 10 ms	d_1
Mean inter-arrival time in S_2	50 ms	d_2
Number of pilot each device requires in S_1	1 or 3	p_1
Number of pilot each device requires in S_1	1	p_2
Traffic load in S_1	[0.1:0.1:1.1]	ρ_1
Traffic load in S_2	[0.1:0.1:1.5]	ρ_2
Number of devices in S_1	$\rho_1 p d_1 / p_1 T_{slot}$	K_1
Number of devices in S_2	$\rho_2 p d_2 / p_2 T_{slot}$	K_2
Schedulers	FCFS, RR_Q or RR_NQ	

A. Simulation model

In our simulation, a predefined number of end-users (representing industrial units) are subscribed to each of the two slices described in Section II. The end-users belonging to a slice have a specific arrival distribution representing the assumed traffic profile for these types of industrial units. The scheduler then assigns pilots to end-users as described in Section III.

Each end-user in the URLLC slice S_1 sends transmission requests with a near-constant period of value d_1 times a small, normally distributed variance $\omega_1 \sim \mathcal{N}(1, 0.05)$. The other slice S_2 however follows a Poisson distribution with a larger average inter-arrival time d_2, representing aperiodic transmission requests from monitoring sensors. The deadlines of the transmission requests are the same as their average inter-arrival times. If a pilot is granted before the deadline is reached, the connection between the device and the BS is initiated and data transmission succeeds within the corresponding coherence time. If a deadline is missed, the packet is dropped.

The massive MIMO MAC layer simulator provided in [13] is an event-based simulator with basic traffic generation functions and scheduling implementations. We extended it so that the end-users are separated into different slices according to different traffic profiles and the pilots are allocated via our two-level scheduling process. Each transmission request is sent when a request event is triggered according to the aforementioned traffic profiles of the end-users. A request is labeled with its slice type, arrival time (the same as the time when the request event is triggered) and deadline (the time when the request expires if no pilot is assigned to it).

Every T_{slot} a new scheduling event is triggered, in which p pilots are released to be assigned to the requests. Each scheduling event handles all the active requests. It marks a request as expired if it has passed its deadline without receiving a pilot. Afterwards it assigns pilots to the requests with later deadlines using the two-level scheduler, until all pilots are used for the current slot time. A request that gets sufficient pilots for its CSI transmission is marked as served. In slice S_2, each end-user takes only one pilot in our simulation ($p_2 = 1$), obtaining the least sufficient CSI quality for the transmission. However in slice S_1, which requires higher reliability, each end-user may need more than one pilot to guarantee accurate CSI, yielding $p_1 \geq 1$, depending on the demands of the users. All unhandled request events are marked as pending and are still active until the next scheduling event.

B. Experiments

In our experiments, we tested different combinations of the scheduling algorithms described in Section III to investigate how an increase of traffic load in each slice affects the QoS performance. We define the traffic load ρ_i in slice S_i in terms of the number of users K_i that is subscribed to it.

$$\rho_i = \frac{K_i p_i T_{slot}}{p d_i}, \; i = 1, 2. \tag{2}$$

The network parameters we used in the simulation are shown in Table I. These parameters are based on those of our 100-antenna test-bed [14] under a high mobility scenario. Additionally, we evaluated our method according to three different aspects: the performance of the URLLC slice, the performance of the mMTC slice and the isolation between the two slices. In the following we describe the experimental setup for the three evaluations.

1) Latency and reliability performance in URLLC slice: To evaluate the latency and reliability performance versus the growth of traffic load ρ_1 in slice S_1, we kept the traffic load ρ_2 in S_2 a constant value 0.5 and applied FCFS as the second level scheduler for S_2. In each simulation, ρ_1 is generated via the profile shown in Table II. In this experiment the end-users in S_1 required a short latency and a moderate channel reliability. All three scheduling algorithms were applied in S_1 for the evaluation.

2) Isolation between the two slices: In this experiment, we investigated whether the performance of slice S_2 will be affected by the selection of the second level scheduler in the higher priority slice S_1. Therefore in this case we kept the traffic load ρ_1 in slice S_1 a constant value 0.5 and applied the parameters in Table II to generate the traffic load in S_2.

3) Latency and connection performance in mMTC slice: In this experiment, we evaluated the QoS performances in slice S_2. We investigated how the latency performs with the growth of the traffic load ρ_2 in S_2. Thus we also had a constant traffic load $\rho_1 = 0.5$ in slice S_1. In this experiment, we applied the RR_NQ scheduler in S_1 and evaluated both the FCFS and RR_Q methods in slice S_2. The traffic load ρ_2 was generated by following the parameters in Table II. In this experiment, we also investigated the limit of the number of end-users that the system can accommodate in S_2 with our parameters.

TABLE II: Parameters used for the three evaluations. Entries with dashes are varied in the corresponding evaluation.

Slice	Variable	Eval. (1)	Eval. (2)	Eval. (3)
S_1	ρ_1	–	0.5	0.5
	p_1	1	3	1
	d_1	1 ms	10 ms	10 ms
	scheduler	–	–	RR_NQ
S_2	ρ_2	0.5	–	–
	p_2	1	1	1
	d_2	50 ms	50 ms	50 ms
	scheduler	FCFS	RR_Q	–

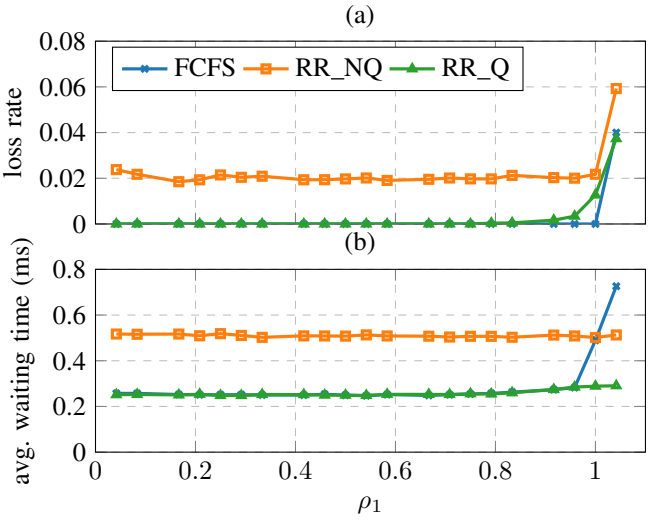

Fig. 2: Slice S_1 (a) loss rate and (b) waiting time with three different schedulers. The RR_NQ method has around 2% packet loss even when the system is lightly loaded. The RR_NQ method has a higher latency compared to the other schedulers. When using FCFS,the average waiting time of the requests grows rapidly once the system is fully loaded.

V. Results and Discussion

In this section, we present and discuss the results of our evaluation. We focus on the average waiting time and the loss rate of the requests to each slice when applying our network slicing scheme. We note that the 95% confidence intervals of the average waiting times from our experiments are all within 1.6% of the mean.

A. Latency and reliability performance in URLLC slice

First, we discuss how different scheduling algorithms perform versus the increasing traffic load ρ_1. Since S_1 has a higher priority there will not be any impact from the traffic load of S_2 on it. Thus, there should not be any packet loss when ρ_1 is less than 1.0. However as shown in Fig. 2(a), the RR_NQ scheduler does have an impact on the loss rate even when the traffic load is small. This loss is due to the variance

of the inter-arrival time of the requests from S_1, considering a case where two subsequent requests from the same device have an inter-arrival time slightly less than d_1 due to the introduced variance. The second request would possibly fail to be allocated a pilot if the first one just arrived at or shortly after the allocation slot in the previous frame. Thus this loss rate is independent of the arrival period and depends only on the arrival rate variance. It is also possible to have such an impact of loss rate on the RR_Q method, especially when the deadline is short (e.g. 1 ms) and the variance on the inter-arrival time is relatively large, as we can see from Fig. 2(a) that RR_Q has an increase in loss rate when S_1 is heavily loaded. This variance is likewise reflected in the variance of loss rate when using the RR_NQ scheduler.

It is also shown in Fig. 2(b) that the RR_NQ scheduler has an effect on the average waiting time of the requests. The average waiting time is equal to half a slot time for the FCFS and RR_Q schedulers since the BS iterates through all the arrival requests every slot time. However with RR_NQ the average waiting time is half a frame time because the BS iterates through all the devices each frame. Furthermore, when ρ_1 reaches 1.0, the waiting time of the FCFS scheduler increases rapidly and almost reaches the deadline of each request on that slice when $\rho_1 = 1.1$. This is because the waiting time with FCFS depends on the queues of all the arrival requests as maintained by the BS, but in the other two cases, it depends on the length of each queue on the device side. Note that this waiting time is calculated from when a packet arrives to the moment when it is assigned a pilot.

There is a trade-off between the computation and signalling cost, and the performance of the different schedulers. The RR_NQ saves on signalling however it is not applicable to critical applications that require ultra-reliability. FCFS gives a good performance, especially when the system is not fully loaded, but it costs heavily in computation and signalling. When the system is almost fully loaded, FCFS may not guarantee reliability as bursty traffic can quickly fill up the arrival queues.

B. Isolation between the two slices

The effect on the QoS performance of S_2 of different schedulers in S_1 reflects the isolation performance of our slicing scheme. Fig. 3 shows a slight difference with the RR_NQ scheduler than with the other two. Under the same traffic load, the average waiting time in S_2 also depends on the traffic profile in S_1.

As shown in Eq. (2), the number of end-users K_1 can be larger when d_1 is longer for the same traffic load. However, with K_1 customers, it takes the base station $K_1 p_1/p$ slots to serve all the devices in S_1, which is longer than it would take if the traffic profile had a shorter deadline, and this would slightly increase the average waiting time of S_2. For the same reason, this difference in waiting time will be smaller when the deadline is shorter since the waiting time is bounded to the frame time when using the RR_NQ scheduler. This means that isolation is not guaranteed by the scheduler since the performance of the waiting time in S_2 depends on the deadlines of the traffic in S_2.

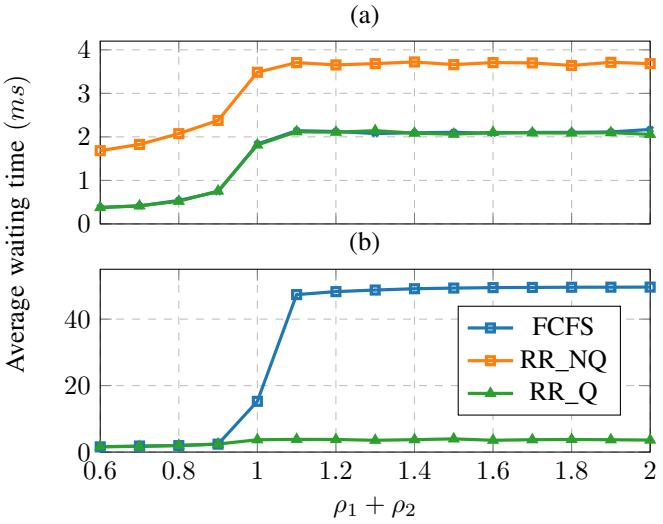

Fig. 3: Slice S_2 average waiting time (a) when applying three different schedulers in S_1 along with RR_Q in S_2 and traffic parameters in Eval. (2) of Table II, among which scheduler RR_NQ results in a higher waiting time than the other configurations; (b) when applying FCFS and RR_Q in S_2 along with RR_NQ in S_1. The x axis shows the overall system traffic load $\rho_1 + \rho_2$ where in this case ρ_1 is constant at 0.5. All the requests have the deadline of 10ms in this experiment.

C. Latency and connection performance in mMTC slice

To investigate the impact of different scheduling algorithms used in S_2, we plotted the waiting time of S_2 as ρ_2 increases. In this case we have a constant traffic load of 0.5 in S_1, which causes a traffic block in S_2 starting from $\rho_2 = 0.5$, as shown in Fig. 4. The 95% confidence intervals on loss rate are within 5% of the values shown in the figure. Fig. 4 illustrates loss rate versus number of connections in S_2 with corresponding overall system traffic load at the top of the figure. Using our specified traffic profile, the system can host up to 600 end-users simultaneously without any loss under both the FCFS and RR_Q schedulers. In S_2, there is no impact from the type of scheduler because RR_NQ is not applicable to this slice and the mean inter-arrival time is long compared to S_1.

The schedulers' performance in S_2 is however similar to S_1 as shown in Fig. 3. There, the waiting time with the round robin method does not increase when the system is overloaded. This is because the system blocks devices exceeding those that can be accommodated in the round robin. The waiting time performance is thus the same as when the traffic load $\rho_1 + \rho_2 = 1$ since we only calculate waiting time for those requests that were assigned pilots.

The simulation results show that the performance of S_2 is affected by the traffic loads in both slices and by the choice

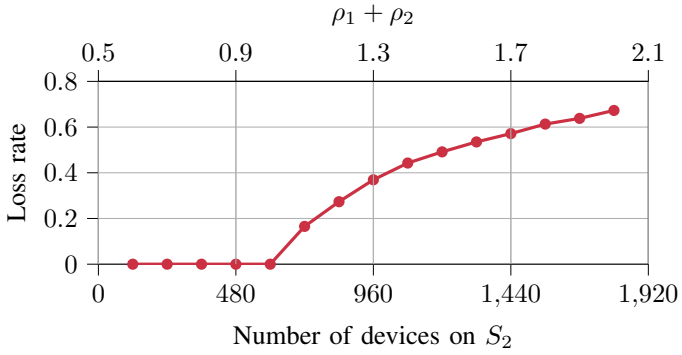

Fig. 4: Slice S_2 loss rate versus number of connections. With the corresponding parameters, the system can host up to 600 end-users in S_2 without any loss. This is equivalent to an overall traffic load $\rho_1 + \rho_2 = 1$.

of its own scheduler, but not by the choice of scheduler in S_1. Since S_2 does not require ultra-reliable performance, we may allow some packet loss in order for the system to host a larger number of connections to the network in S_2.

VI. Conclusion

In this paper we have presented a resource slicing scheme for a single-cell RAN based on massive MIMO. We proposed a two-level MAC scheduler that divides the radio resources into two parts and maps them to each slice, and is also responsible for intra-slice resource allocation. We considered three different scheduling algorithms (FCFS, RR_Q and RR_NQ) that could be applied in our slicing scheme. Simulation experiments were performed to evaluate the system performance in accordance with the user requirements of each slice.

Since massive MIMO systems have all the signal processing and channel estimation only on the BS side, it may cause lower power efficiency if the computation on the MAC scheduling consumes a lot. RR_NQ scheduling has the lowest overhead since it is not necessary for the BS to maintain a queue of all incoming requests or for the end-users to signal for transmission requests. However this method does not guarantee ultra-reliability since there can be packet loss due to the statistical scheduling approach. Further, it is not applicable for the case of massive Internet of Things (IoT) connections since the number of devices is far more than the available pilot resources. FCFS scheduling is more reliable and applicable but may have much higher overhead on the BS for maintaining the arrival queues. Meanwhile the RR_Q method has a large signalling overhead.

Our slicing scheme can be extended to host more than two slices or apply other scheduling algorithms to meet diverse performance requirements. The next step of our work is to extend the proposed scheme to a complete network slicing approach that combines RAN slicing with Core Network (CN) slicing in an optimal way.

References

[1] G. Zennaro, A. Stojanovic, and M. Giordanino, "Report on vertical requirements and use cases," 5G-TRANSFORMER Project, Tech. Rep. 761536, 2017, accessed: Mar, 4, 2020. [Online]. Available: http://5g-transformer.eu/index.php/deliverables/

[2] M. Gidlund, T. Lennvall, and J. Åkerberg, "Will 5g become yet another wireless technology for industrial automation?" in *2017 IEEE International Conference on Industrial Technology (ICIT)*, March 2017, pp. 1319–1324.

[3] M. Maternia, S. E. El Ayoubi, M. Fallgren, P. Spapis, Y. Qi, D. Martín-Sacristán, Ó. Carrasco, M. Fresia, M. Payaró, M. Schubert, J. S. Bedo, and V. Kulkarni, "5G PPP use cases and performance evaluation models," 5Gppp, Tech. Rep., April 2016, accessed: Oct. 23, 2019. [Online]. Available: http://www.5g-ppp.eu/

[4] GSA, "5g network slicing for vertical industries," Global mobile Suppliers Association, White Paper, September 2017, accessed: Oct. 23, 2019. [Online]. Available: https://www.huawei.com/minisite/5g/img/gsa-5g-network-slicing-for-vertical-industries.pdf

[5] GSMA, "An introduction to network slicing," GSM Association, Tech. Rep., 2017, accessed: Oct. 23, 2019. [Online]. Available: https://www.gsma.com/futurenetworks/resources/an-introduction-to-network-slicing-2/

[6] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, May 2017.

[7] C. Bektas, S. Monhof, F. Kurtz, and C. Wietfeld, "Towards 5g: An empirical evaluation of software-defined end-to-end network slicing," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec 2018, pp. 1–6.

[8] A. Ksentini and N. Nikaein, "Toward enforcing network slicing on RAN: Flexibility and resources abstraction," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 102–108, 2017.

[9] A. Aijaz, "A radio resource slicing framework for 5g networks with haptic communications," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2285–2296, 2017.

[10] A. Gudipati, L. E. Li, and S. Katti, "RadioVisor: A slicing plane for radio access networks," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 237–238.

[11] T. L. Marzetta, E. G. Larsson, H. Yang, and H. Q. Ngo, *Fundamentals of Massive MIMO*. Cambridge University Press, 2016.

[12] Massive MIMO slicing. Accessed: Oct. 23, 2019. [Online]. Available: https://github.com/HaoruiPeng/slicing-simulator

[13] Massive MIMO simulator. Accessed: Jun. 28, 2019. [Online]. Available: https://github.com/jost95/massive-mimo-simulator

[14] J. Vieira, S. Malkowsky, K. Nieman, Z. Miers, N. Kundargi, L. Liu, I. Wong, V. Öwall, O. Edfors, and F. Tufvesson, "A flexible 100-antenna testbed for massive MIMO," in *2014 IEEE Globecom Workshops (GC Wkshps)*, Dec 2014, pp. 287–293.